The Cobalt AUV: Design and Development Palouse Robosub

James Irwin, Brandon Kallaher, Ryan Summers, Christian Ziruk, et al.

***Abstract*—This paper aims to convey the design of the 2016-2017 Palouse RoboSub AUV - Cobalt. Included is the theory behind the design process and the implementation from electrical, software, and mechanical engineering perspectives. The electrical aspect of the paper goes over the change from a system with centralized control boards to a more distributed board design scheme, while the mechanical section covers the design decisions made to accommodate changes in the electrical design and new system additions. In terms of software, the paper goes over the new features and tools created to help increase development speed, including the conversion of the system to use the ROS (Robot Operating System) framework.**

***Keywords*—journal, LaTeX, paper, RoboSub, Palouse, WSU, AUV, engineering, Cobalt, AUVSI, ROS.**

I. Design Strategy

In order to facilitate effective communication both between teams and between sub-team members, the team primarily focused on design processes, communication, and improved information flow. Included in this philosophy was the extensive development of an online wiki [2] for containing documentation and design resources. This helped the sub-teams to quickly and effective communicate designs with each other. By developing standardized ways of introducing new ideas into the project, the team established the foundations for good engineering practices. This was a guiding principle for the fundamental design strategy employed by the team for the 2016-17 competition year, and differs substantially from the engineering-focused mindset of previous competition seasons.

A. Software

At the beginning of this year, the software team spent a significant amount of time to find the root causes of the difficulties experienced in previous years and identified new ways to circumvent them. As a result, the software team developed 4 main goals to focus on over the course of the 2016-17 competition season:

1) Implement (and stick to) a well-defined software workflow
2) Focus on core functionality first to ensure a solid foundation
3) Grow the software team
4) Develop a simulator to accelerate development

Workflow: For goal 1, the software team wanted to create a structured environment for collaboratively working on code. Historically, the team would regularly encounter compiler errors, incomplete code, and various other issues in the "stable" branch of the codebase, as well as confusion over what each team member was working on and its state of completion. The team determined that a well-defined process for assigning tasks and merging code was needed. In addition, the team felt it important to implement a review process to improve code quality. Although the software team has attempted this goal in the past, they failed largely because people would simply choose not to follow it - particularly when faced with looming deadlines. To combat this, tool assistance was used wherever possible to enforce the workflow. Tools such as Github's pull requests and issue tracker, Reviewable's code review tool, and TravisCI's continuous integration server have been used extensively throughout the year, and Github's branch permissions helped to ensure these checks all passed. The software team lead set up a Debian software repository so that a single command could be used to download all the necessary software and configure a computer for RoboSub development, whereas previously the team lead maintained a bash script for performing initial installation. In addition, the Debian repository allowed the computer's main update system to ensure everyone used the same software packages.

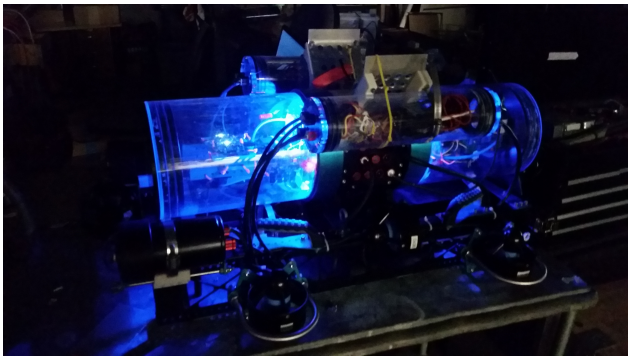


Fig. 1: Cobalt sits in the lab after a successful pool test.

Task Priority: Goal 2 can be interpreted on several different levels: a system architecture level, a task-priority level, and the individual module level. Since the software team's creation, a multi-process system architecture has been used where multiple computer processes run to implement the complex functionality of the AUV, and it is necessary for these processes to communicate with each other to some degree. Another task is the coordination and management of the individual programs. To implement this communication, the software team has historically tried to develop a custom message-passing framework, which was usually based on the the ZeroMQ library [12]. However, each year it was scrapped to be replaced

with a ”better” custom system, which resulted in large yearly efforts to debug the communication framework. This resulted in spending more time fighting and debugging the networking and process management tools than writing the actual code for the AUV. In an attempt to stop this cycle, the software team decided to try out a popular software library designed for robotics coined the *Robot Operating System* (ROS) [9].

Another lesson from the team's history revolved around the workflow. The software team historically followed a top-down design model, and usually spent most effort on vision processing and the high-level decision making programs (AI) while simultaneously neglecting essential lower-level components, such as the control system and thruster modules. As a result, these critical components were often buggy and, in the end, prevented the practical use of any vision processing results or AI decisions. The software team chose to follow a bottom-up design this year, which entailed creating the smallest building blocks first and then building more complex functionality on top of them. At the start of the 2016 fall semester, the software team started out by writing the software for our sensors and thrusters, and thoroughly tested those modules. Next, the team debugged the control system to the point of having stable movement, and by the 2017 spring semester they started work on vision processing and decision-making. This work ordering resulted in tangible progress much earlier in the year, and allowed immediate pool testing of vision and decision-making code as soon as it was written.

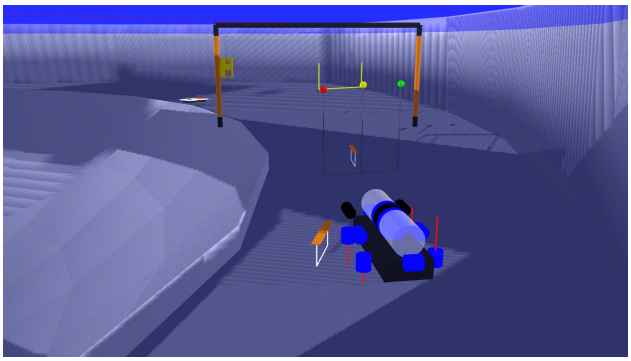


Fig. 2: Gazebo Simulator

An issue that has constantly plagued the software team is that members would write large, elaborate software modules with the preemptive intent to be efficient and customizable (premature optimization and modularity). This was combined with little practical integration testing until the end of the year, at which point it was often discovered that the prime functionality of the module was buggy. Compounding this complexity was a lack of documentation, which made software difficult to maintain. To combat this, the software team followed the KISS principal (keep it simple, stupid) and decided that the main priority would be to get the basic functionality of the module running and well-tested, and only perform optimization and add modularity when needed. This has resulted in faster software development and code that is much easier to maintain.

Team Growth: Historically, the software team was composed of 2-3 members, typically juniors or seniors, which resulted in a lot of software turnover and was a lot of work for such a small team. Goal 3 is a necessary step to ensure the survival of the team on a long-term basis, but also increase the team's software output. This year the software team attempted to simplify the learning curve for new members, and also focused most of the first semester towards on-boarding new members. Simplifying software setup (as discussed in goal 1) helped immensely with getting new members started. In addition, the software veterans created many smaller example programs to help members ”learn the ropes” before having to understand the complex minutiae of a real module. During the first few weeks, the team explicitly disallowed veteran members from writing code. Instead, they assisted the new recruits to write software modules. Although time consuming, this was found to be an effective method of teaching the idiosyncrasies of how we write software. It also helped new members engage with the veteran members and feel part of the group. The software team has also maintained a wiki [2] to document high-level code design and organization.

Simulator: Although a side goal last year, goal 4 became a major objective for this year's team. The ability to perform preliminary tests of total system integration without running on real hardware has had a huge impact in software development speed. Progress of the simulator is described further in section II, while an example image from the simulator is visible in Figure 2.

B. Electrical

In previous years, concurrent development between the software and electrical teams has been fraught with difficulty. The electrical team naturally desires having the ability to make modifications to the AUV to prototype hardware, while the software team needs a stable AUV to test algorithms on. When its not known whether a bug is a result of hardware or software, the time required to debug problems on the AUV increases drastically. This year, the electrical team took great pains when implementing upgrades so that Cobalt has been able to undergo a complete overhaul to the electrical design, while simultaneously maintaining a stable development platform for the software team. The motivation for the rework of the electrical system revolved around 3 key goals:

1) Stability
2) Reliability
3) Ease of Implementation

Stability: Maintaining an AUV that is continuously available to other parts of the club is crucial to success at the competition. If the AUV is taken out of commission as a result of electrical work, significant setbacks can be experienced by other portions of the club. By establishing a number of guiding principles to follow when working on electrical portions of the AUV design, the team has vastly improved stability over previous years. In order to maintain a stable software testing platform, the electrical team focused the first two months on finalizing all remnant projects from the previous year's design cycle. This provided the rest of the club with an AUV that was

capable of performing basic functions in a reliable fashion, i.e. moving around the pool, determining orientation, and sensing depth. By keeping the initial requirements simple, the first iteration of the electrical framework came together quickly and provided the first initial, stable platform for the software team. Once the AUV was functional, the electrical team transitioned towards individually upgrading specific projects. Instead of removing and replacing sensors for testing, a 'ride-along' methodology was employed where multiple sensors were installed on the AUV. This allowed for prototype software and hardware to be run in realistic testing environments (i.e. when the AUV is moving around in the pool), while it also ensured that the software team would have access to hardware that was known to be stable. When the new sensors and electrical equipment were properly tested and verified, components were swapped out and older sensors and hardware components were removed without the software team having to compensate. This streamlined the testing and verification process between the electrical and software teams and resulted in a large productivity increase throughout the year.

Reliability: Reliability was a core design requirement throughout the electrical team's design process. When there was confusion about whether the code was broken or if the hardware was actually the problem, time was lost and errors were made. Reliability is composed of two distinct parts: designing with intent and hardware verification. By designing with reliability as a key principle, steps can be taken early in the design process to facilitate a more reliable system. Additionally, if reliability principles are put in place with the first revisions of hardware systems, dependability problems can be quickly addressed and remedied before designs are finalized. This permits the design process to be much more flexible on both the electrical and mechanical teams. However, designing with an intent for reliability is meaningless unless projects are extensively tested to ensure they meet the specification. This year, the electrical team focused on ensuring that all projects met rigid standards on electrical connectivity and specification performance before projects were integrated into the AUV. This resulted in a vast improvement in usability of AUV hardware over previous years.

Ease of Implementation: Because the RoboSub competition requires so many different electrical subsystems to be completed in a short development period, the electrical team chose to leverage popular hardware platforms. By utilizing the Arduino framework based on Atmel AVR microcontrollers for all of the embedded systems, development time was drastically reduced and code maintainability was improved. This allowed for the electrical team to take on more ambitious projects, while worrying less about the smaller details. The overarching philosophy put in place was integration as opposed to building from scratch. Inertial sensors were selected that provided embedded fusion algorithms as opposed to raw IMU readings, and projects were abstracted into individual components connected to the main computer by individual USB connections. By focusing on systems that were easy to implement, the electrical team was able to ensure that projects met specification and performed reliably for the software team.

C. Mechanical

The mechanical team's primary focus for Cobalt was optimization. With fatigue failure occurring on certain components, it was necessary to redesign and manufacture enclosures for new and old electronics systems in order to have an ideal lifespan. This route allowed our team to maintain most of last year's AUV while focusing on areas in need of improvement. Focusing on smaller projects allowed the team to practice and develop their mechanical engineering skills on smaller more focused projects. By not designing and manufacturing an entire AUV, the mechanical team was able to focus on three major areas of improvement:

1) Robustness
2) Ease of Maintenance
3) Modularity

Robustness: In order to be successful the team needed to understand the AUV and the systems within it. To kickoff the season, the team focused its energy on understanding the areas in need of redesign due to the lack of robustness. By spending the first month learning faults within Cobalt and why they occurred, the team was able to develop a strong foundation for future design decisions that would be made for this year's AUV. The design of the new enclosures were designed with appropriate material selection and appropriate sealing considerations. In previous years, material and sealing was decided based on earlier iterations of the team's AUVs. With access to many different materials and machining capabilities, using appropriate material for a greater fatigue life is optimal. Spending additional time to make the AUV more robust allows the AUV to be in a usable state much longer and used for testing for years to come.

Ease of Maintenance: Certain systems require more attention than others on the AUV. When designing for ease of maintenance it was critical to understand which enclosures would need to be accessed at a moments notice versus an enclosure that would need to be opened due to a critical failure. The team approached this challenge by dividing the enclosures and main hull into different categories based on need of accessibility. From there the choice in sealing method was made between O-rings and Epoxy. Cobalt reflects the use of both O-rings and epoxy. However due to the need for accessibility, O-rings were a more prevalent choice for most enclosures.

Modularity: Due to the nature of our team's hardware and electronics choices, being dynamic and having modularity within the design is critical to system integration. The design of the main hull and all enclosures are placed in sections of the AUV that allow other systems to be added to it at anytime. By incorporating modularity in needed areas of our AUV, the team can continue to test and implement necessary hardware without a complete redesign. This approach saves time and gives other sub-teams a chance to test systems while we continue to add to the AUV.

II. Vehicle Design

A. Software

1) Overview: A high value was placed on small, simple programs to do specific tasks. A ROS node was written for each sensor (various IMUs, depth sensor). A control system node receives sensor data describing the state of the AUV (orientation, depth), as well as goals (roll, pitch, yaw, depth, forward, and strafe), and sends normalized thruster values, as seen in Appendix B. These normalized thruster commands are received by our thruster module, which uses a serial port to talk to the hardware that controls the thrusters. This basic layout can be seen in Appendix A. In addition, we started development of a particle filter for integrating IMU data, depth reading, vision processing, and hydrophone measurements to estimate our position in the pool. Although this shows promise in the simulator, its appearance at the competition is dependent on the state of hydrophone progress.

2) Simulator: Last year we started the development of a simulator primarily for testing the control system, vision processing, and higher-level decision making logic. The simulator was developed in the Unity [11] game engine because of its potential for realistic imagery. However, it was difficult to install and use, and added yet another programming language to our codebase (C#). We chose to rewrite our simulator using the Gazebo7 [1] simulator because of its realistic physics modeling, ease of installation and interfacing with ROS despite that fact that the graphics are much poorer than Unity. We wanted the simulator to provide a solid integration test-bench, therefore several bridge modules were written to present an interface that exactly mimics the software interfaces on the real hardware, including the serial ports. This allows us to realistically test that every component of the system talks correctly without needing the real AUV present.

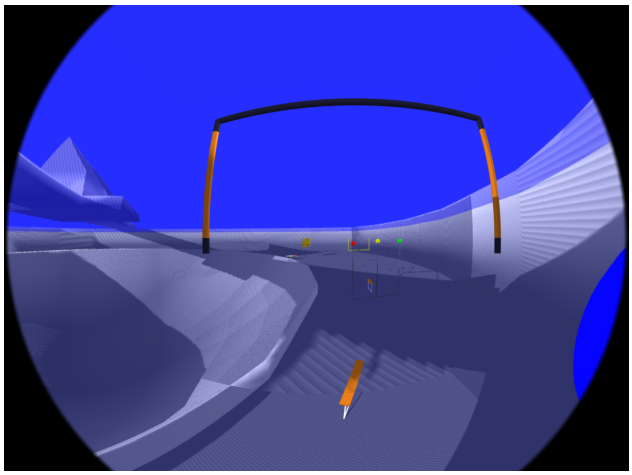


Fig. 3: Simulated Camera View

3) Vision: Previously the object detection system used hand coded filters to programmatically detect the location of objects in the pool. It was found that this style of object detection is not sufficient, even after extensive tuning. In order to solve this problem, object detection systems using machine learning methods to detect objects in the pool were investigated. Theoretically, machine learning methods of object detection are much more reliable than hand coded filters because the algorithms create more powerful models and are able to use more feedback than can be accomplished by hand. This has been shown by image tagging and classification competitions such as ImageNet [3] and PASCAL VOC [8] where algorithms are able to recognize hundreds of objects in real-world pictures.

When evaluating which algorithm to use, speed was an important concern as there is limited computing power available on the AUV and deep object detection is a computationally expensive operation. Additionally, the object detection system should run at the same rate as our camera input (5 frames per second) so that the AI system is able to run with adequate feedback. Training speed was also a concern, while there are significant computing resources available through the Intelligent Robotics Learning Lab (IRLL), there are only about 3 days available to collect, label, and train with data at the competition. Thus, any framework chosen must be able to converge in less than 2 days. Cobalt has also been outfitted with a Jetson TX2 [7] to help the object detection subsystem run as quickly as possible.

Three machine learning algorithms for object detection were tested for their accuracy, evaluation speed, and training speed. These algorithms were evaluated using only an i7-7700K quad core CPU running at 4.2GHz on a hand labeled dataset designed for the RoboSub competition. First and foremost, YOLO (You Only Look Once) [5] was evaluated using the Darknet [4] framework, a deep object detection framework boasting the fastest speeds, with real time performance on a dedicated GPU. Both standard YOLO v2 and YOLO-tiny v2 were tested [5]. Both were able to perform with good accuracy on the dataset at relatively fast speeds. YOLO evaluated at a rate of 0.2 FPS (frames per second) and YOLO-tiny evaluated at a rate of 1 FPS. Notably, the Darknet framework only allows evaluation on one thread when running in CPU mode. Second, YOLO-tiny v2 was tested in the TensorFlow [10] framework. TensorFlow allows for execution on all CPU threads and as a result YOLO-tiny evaluated at 8 FPS. Lastly, Faster R-CNN was tested. Faster R-CNN was the most accurate, but also the slowest of the frameworks tested. With a back end of TensorFlow, Faster R-CNN evaluated at 0.17 FPS.

After testing, Darknet YOLO was chosen to integrate into Cobalt for its speed and ability to train with multiple GPUs. Because of its lower CPU performance, training the network with Darknet and moving the network weights to TensorFlow for prediction was considered but rejected because the Jetson TX2 should allow Darknet to run sufficiently once it has been integrated.

4) Localization: In the past Cobalt has had no way of knowing its precise X and Y position within its environment, though Z position can be found relatively accurately using depth sensors. This year brought the development of a hydrophone system that should be able to determine the position of the AUV relative to the acoustic pinger in the pool allowing the system to determine the general position of the sub. Unfortunately, the hydrophones only report data

once every 2 seconds (because the pinger only pings once every 2 seconds) and they are quite susceptible to noise which results in low frequency and somewhat inaccurate position measurements. To improve this, the localization system uses both a particle filter and a Kalman filter to perform sensor fusion and produce a more accurate and higher frequency estimate of the sub's position.

Since the team does not have a DVL (Doppler Velocity Logger) or another way to determine the velocity of the sub, a nine state Kalman filter was chosen to fuse acceleration and orientation data from the IMU and depth data, to estimate the subs velocity. Additionally, once the particle filter is converged and stable, its position estimate is input into the Kalman filter. A Kalman filter was chosen due to its simplicity, efficiency, and accuracy using linear, Gaussian data.

Localization uses the particle filter to fuse the hydrophone data and depth data, as well as the velocity estimated by the Kalman filter. A particle filter was chosen as it handles low frequency, non-linear input well.

The simulator was essential to the development of the localization system since the hydrophone system was being developed in parallel. It allowed changes to be made and rapidly tested using unit tests that would be difficult or impossible to do in real pool testing.

5) AI: Although it has not been the main focus of the year, the AI is an important piece of the software as it sets high level goals such as "go to depth", "maintain orientation", "yaw right", and "move forward." Problems such as "is this orange object a path marker or a start gate?" were designated to be the domain of the vision processing system in order to simplify the higher-level decision making logic. This was in an attempt to make modifying the behavior of the AUV on-the-fly as simple as possible. Python is used to keep the code simple, straightforward, and mutable. Going forward, the AI will be the main focus of the software team as the other parts are being polished.

B. Electrical

Overview: The electrical design of the AUV is composed of two main components. The first priority of the electrical team was to properly and safely route power to the thrusters and main hull. Because Cobalt is utilizing Blue Robotics T200 thrusters, there were much higher power requirements as each thruster can draw up to 25 Amps. To accommodate for the large current requirements on the battery, all power routing was done in an external power compartment. A dual relay setup was used to allow for the thrusters to be independently shut down while maintaining power to the main hull and the hardware controllers. The overall system diagram of power flow throughout the submarine can be seen in diagram in Appendix D.

The second set of responsibilities for the electrical team was to provide sensor data to the software team. This year, the electrical system on Cobalt follows a simple, expandable model for changing hardware, adding functionality, and verifying and testing projects independently. To do this, each project was considered as a separate entity that connected to the main computer through a USB port. Each USB port then enumerated as a serial interface for the software team to interface with. This also allowed for simplifying cabling within the AUV because each electrical peripheral only had a single cable running to it. This has increased the reliability of individual projects and has reduced clutter within the hull of the AUV.

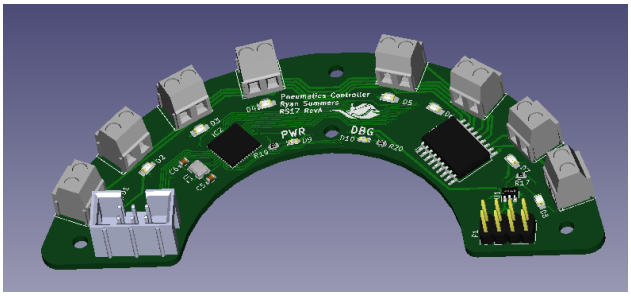


Fig. 4: The pneumatics circuit board was designed through close collaboration between mechanical engineering specifications and electrical engineering designers.

Peripherals

Cobalt is composed of a number of electrical peripherals connected through a common USB hub including up to three inertial measurement units (IMUs), four independent, high-precision depth sensors orientated throughout the AUV, two cameras mounted on the bow, controllable LED strips for providing mission feedback information, and a remote pneumatics controller. Microcontrollers were eliminated throughout the previous revisions of Cobalt in favor of software controllers. This was in order to reduce both developmental overhead as well as information latency. By writing software drivers that interfaced directly with IMU serial interfaces, code was easier to test and verify, information was acquired faster, and unit testing could be put in place for continuous integration. It also reduced the hardware design overhead as it removed the hardware support for an embedded device. Cobalt transformed from running over six independent microcontroller programs last year to only utilizing three simple controllers. Additionally, all controllers were written utilizing the ROS-Serial interface, which allows the embedded hardware to communicate with the rest of the software system as if they were independent programs. The overall communication architecture for the electrical system can be observed in Appendix C.

Throughout the year, the electrical team chose to utilize common designs to eliminate all possible rework. Because all electrical peripherals communicate through a common serial-USB interface, a generic adapter board was created for interface conversion. This board mounts on top of all other boards for small form factor and high reliability of signal continuity. The universal adapter shown in Fig. 5 resulted in a much smaller electronics footprint within the AUV and gave the mechanical team more freedom in designing the internals of the AUV. The pneumatics controller board shown in Fig. 4 was one of the first collaborations between the electrical

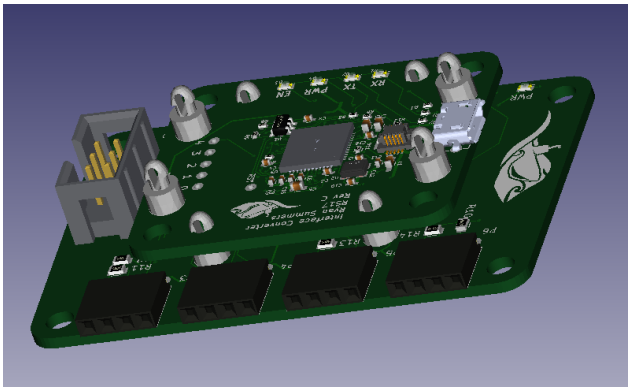


Fig. 5: The interface converter board sits aboard the depth sensor controller and communicates through a mezzanine connector.

engineers and the mechanical engineers to create a circuit board that fit into the mechanical design of the AUV.

IMU Selection: One of the most important tasks for the electrical team was the selection of a reliable IMU to provide orientation information to the software team with minimal error. After learning the difficulties of implementing custom fusion algorithms in previous years, the team decided to utilize IMUs with integrated fusion algorithms. The BNO055 proved to be a low-cost IMU that still provided reliable and consistent results. It allowed for easy implementation, as it has a serial UART communication interface. Two individual BNO055s were installed in the AUV in opposite locations to minimize environmental impact as a result of magnetic transients introduced by the power relays and thruster currents.

C. Mechanical

Overview: With a complete AUV that was built for all movement tasks, there were still a few places in need of optimization. As mentioned earlier, the Blue Robotics T200 thrusters with integrated BlueESC did not work at last years competition resulting in seven dead thrusters. It was urgent to adapt new enclosures to accommodate new, discrete ESCs. Additionally, other enclosures were in need of changes to adapt to new electronics hardware that was being updated. Two main enclosures and updates were made to Cobalt in order to meet the need of the competition being the ESC case and Pneumatics case.

ESC Case: The ESC case started out within a tubular enclosure, than evolved to a rectangular one due to the number of connectors required and to improve accessibility to the ESCs. The connector choice was a hybrid of SUBCONN In-line and Blue Robotics Bulkhead connectors. Originally the design was going to only have SUBCONN Bulkhead connector, but due to the connector being too large for the case, a hybrid solution meets the requirements and was less expensive overall. The case itself is designed to hold four Afro ESCs, requiring two case assemblies to be manufactured. By doing this an equal weight distribution was maintained and divided the eight thrusters between two cases. The material choice was 6061 Aluminum for the outside and cover, ABS for power distribution and ESC fixtures, and a Buna N O-ring for the seals of the lid and bulkhead connectors. With the choice of material having a high machinability, the case was manufactured on campus at the Cougar Student Machine Shop. Due to case size and the machines work volume, the Haas Mini Mill was the ideal choice to machine out the lid and base for the case. Using MasterCam the case was able to be machined using dynamic tool paths to reduce machine time. Also reducing machine time was the choice to place all holes for connectors on one side, thereby reducing tool and part set up within the machine. All fixtures within the case were either laser cut for parts that only required 2D vector files and others were 3D printed. A view of the ESC case can be seen in Fig. 6.

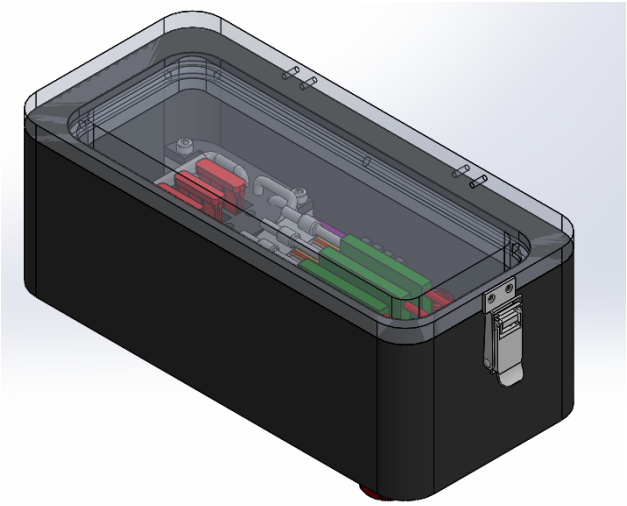


Fig. 6: An ESC Case

Pneumatics: Air is the primary source for actuation on Cobalt. Purchasing reliable motors for an underwater application can be hard to come by for the specific needs of Cobalt. Holding six solenoids, the pneumatics case is placed on the bottom to help balance the AUV. Directing the air to each of the solenoids through the manifold was a difficult task. Our manifold acts as the air supply channel and the base to the enclosure. To minimize the volume of the enclosure it was ideal to incorporate the manifold as the base to the enclosure. Initial design considerations incorporated a off the shelf manifold, but this proved to be a waste of space and make the assembly much heavier. Since this case will be accessed for maintenance, a face seal was used with a bolt pattern to compress the O-ring between the base and lid. The material choice for this case consisted of 6061 aluminum for the manifold and Buna-N for O-ring and gasket material. The most difficult part to manufacturing this assembly is the manifold. With three different sides that require material removal, three different set ups were needed. As with the ESC case, this enclosure used Mastercam's dynamic tool paths for

the roughing operations alongside contour and peck drilling operations for the holes and pockets. The Pneumatics case can be seen in Fig. 7.

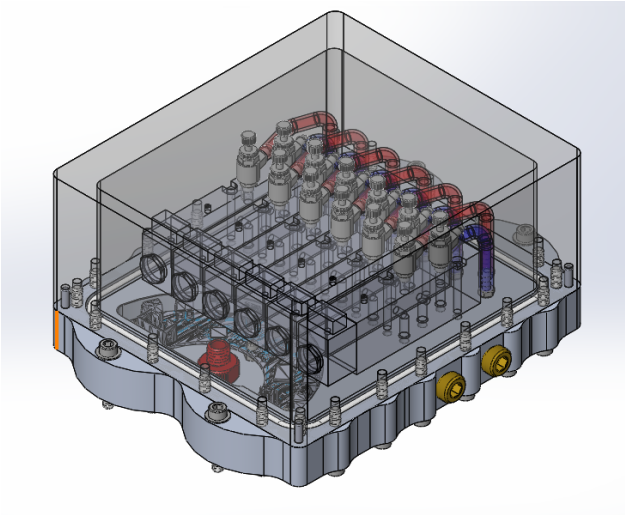


Fig. 7: Pneumatics Case

III. Experimental Results

Simulator Testing

Within the Gazebo simulator, the localization engine, vision processing, and control system have been verified to perform as designed. Additionally, the AUV can successfully (and repeatedly) complete the start gate and buoy tasks.

Orientation Testing

After a number of pool tests with the completed preliminary AUV, it was noted that Cobalt had a noticeable drift in yaw over time. A number of precision tests were conducted on yaw authority, and it was observed that the AUV yaw could drift up to 180 degrees over the course of a few minutes. However, pitch, roll, and depth sensing were consistently correct. After a preliminary investigation, it is believed that the relays controlling power may be causing a level of static magnetic distortion that the BNO055 is not able to properly filter out. This results in the magnetometer reading of the IMU being inaccurate, which automatically puts the sensor into a relative orientation mode. During these periods, the sensor primarily utilizes the gyroscope for determining yaw. The BNO055, being a relatively in-expensive IMU, has a gyroscope drift of up to three degrees per second, and it is believed that this error contributed to the noticed yaw drift. The sensors have since been moved further away from the power relays, relays with smaller active current (and thus smaller magnetic fields) have also been put in place, but the team has not yet had the opportunity to see if these changes result in corrections to the IMU orientation readings.

Object Detection

The YOLO framework has shown excellent performance in both speed and accuracy on example footage collected from pool tests. However, testing on previous competition footage has shown that the network fails to generalize to the murky competition environment. The software team plans to improve the system by training on existing competition footage and on new data collected at this year's competition. Unfortunately, the object detection system has not been evaluated live during a pool test as necessary hardware has yet to be integrated.

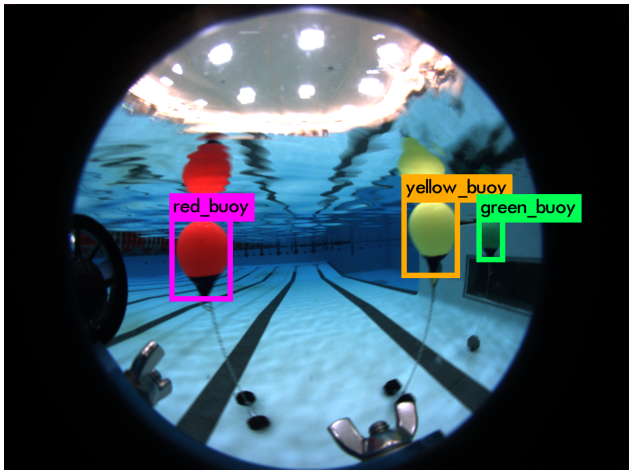


Fig. 8: Object Detection

IV. Acknowledgements

The authors would like to acknowledge the following people for their assistance with the project this year:

General Team Members

Peter Brennan, Mykhailo Bykhovtsev, Courtney Cadenhead, Kayl Coulston, Dustin Crossman, Justin Dominiak, Brandon Evans, Andrew Feistner, Omar Finol-Evans, Derek Fisk, Edoardo Franco-Vianelli, Adolfo Garcia, Daniel Goto, Alex Gu, Alex Hirte-Uhlorn, Gunnar Jensen, Sean Kallaher, Daylan Kelting, Stasia Kulsa, Edward Kuo, Alex Lanphere, Cal Merriman, Drew Miller, Brian Pecha, Kimi Phan, Zachary Pratt, Ben Songras, Mark Summers, Courtney Snyder, Lucy Tran, Johnny Wang, Liudi Yang

Mentors

Dr. Aaron Crandall, Aaron Darnton, Mike Kapus, Dr. Patrick Pedrow, Alex Read, Dr. Matthew Taylor

REFERENCES

[1] ”Gazebo”. Gazebosim.org. N.p., 2017. Web. 21 June 2017.

[2] ”Home”, Palouse RoboSub Technical Documentation, 2017. [Online]. Available: http://robosub.eecs.wsu.edu/wiki/. [Accessed: 21- Jun- 2017].

[3] ImageNet, 2016. [Online]. Available: http://image-net.org. [Accessed: 21- Jun- 2017].

[4] J. Redmon, ”Darknet: Open Source Neural Networks in C”, Pjreddie.com. [Online]. Available: https://pjreddie.com/darknet/. [Accessed: 21- Jun- 2017].

[5] J. Redmon, ”YOLO: Real-Time Object Detection”, Pjreddie.com, 2016. [Online]. Available: https://pjreddie.com/darknet/yolo/. [Accessed: 21- Jun- 2017].beginthebibliography10

[6] Leslie Lamport, *LaTeX: a document preparation system*, Addison Wesley, Massachusetts, 2nd edition, 1994.

[7] ”NVIDIA Jetson Modules and Developer Kits for Embedded Systems Development”, Nvidia.com, 2017. [Online]. Available: http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html. [Accessed: 21- Jun- 2017].

[8] ”The PASCAL Visual Object Classes Homepage”, Host.robots.ox.ac.uk. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/. [Accessed: 21- Jun- 2017].

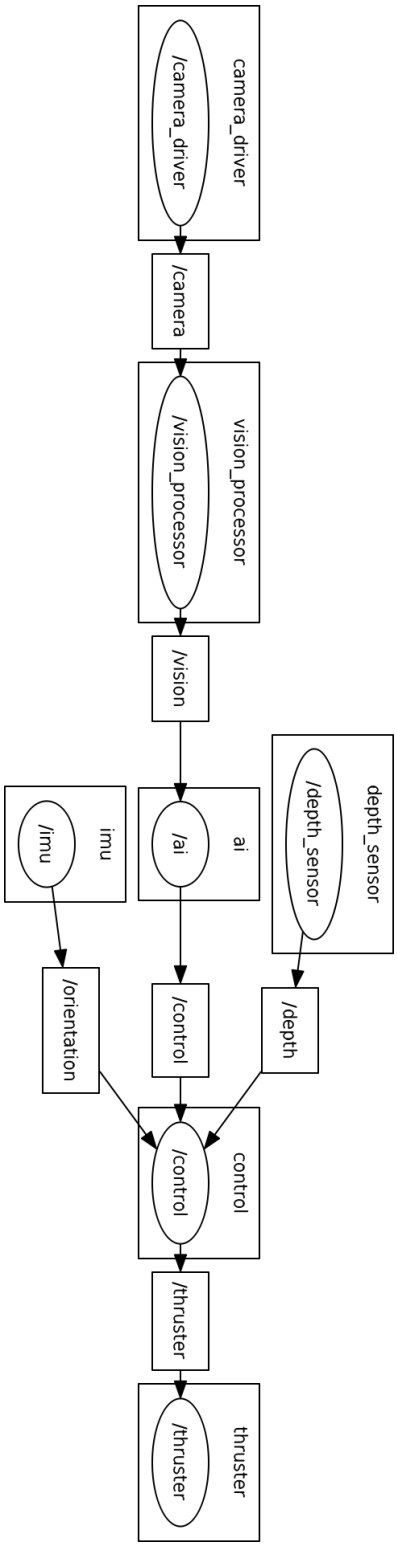
[9] Robot Operation System, ROS.org. [Online]. Available: http://www.ros.org/. [Accessed: 20-Jun-2017].

[10] ”TensorFlow”, TensorFlow, 2017. [Online]. Available: https://www.tensorflow.org/. [Accessed: 21- Jun- 2017].

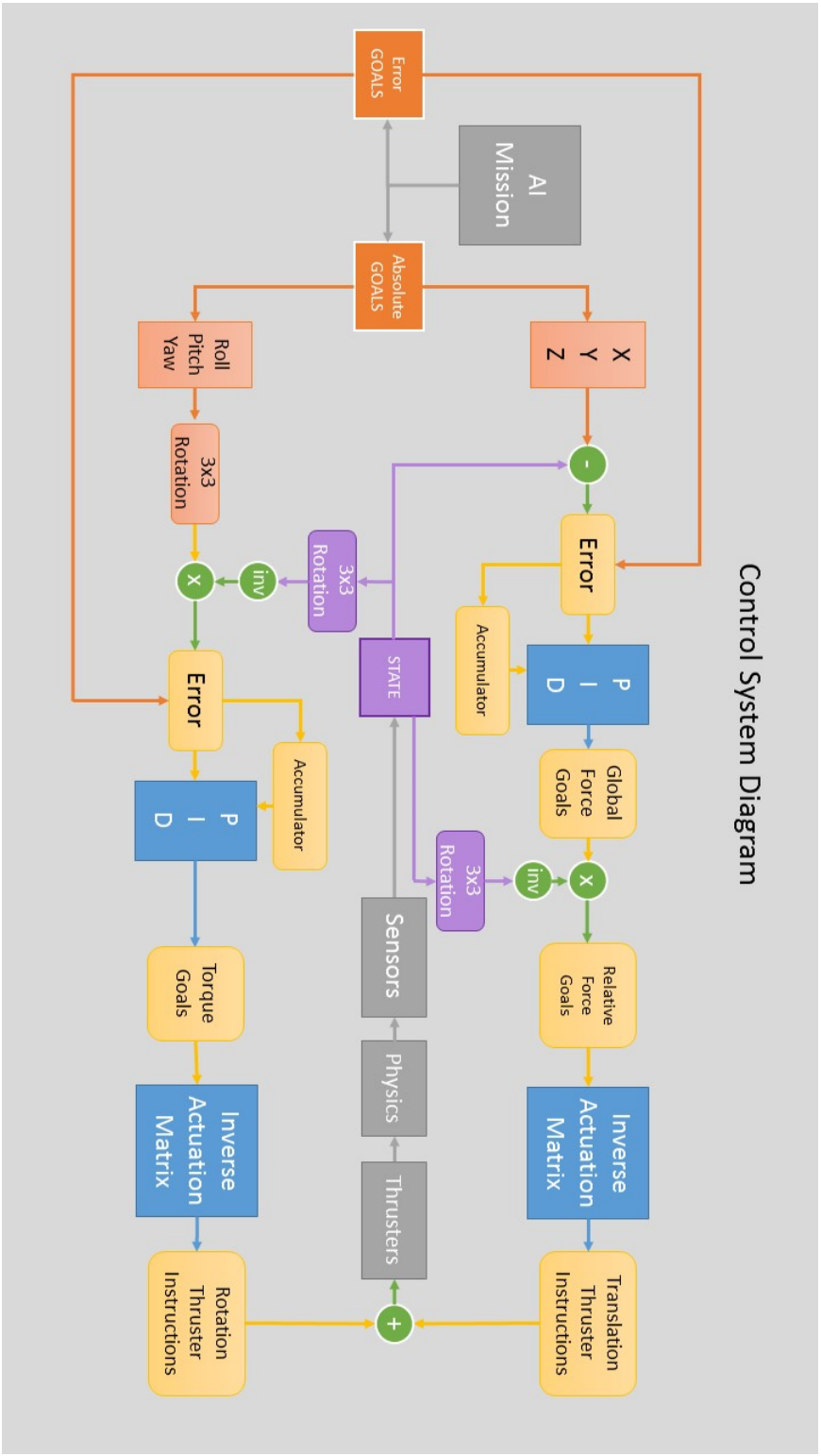
[11] ”Unity - Game Engine”, Unity, 2017. [Online]. Available: https://unity3d.com/. [Accessed: 21- Jun- 2017].

[12] zeromq, Distributed Messaging - zeromq. [Online]. Available: http://zeromq.org/. [Accessed: 20-Jun-2017].

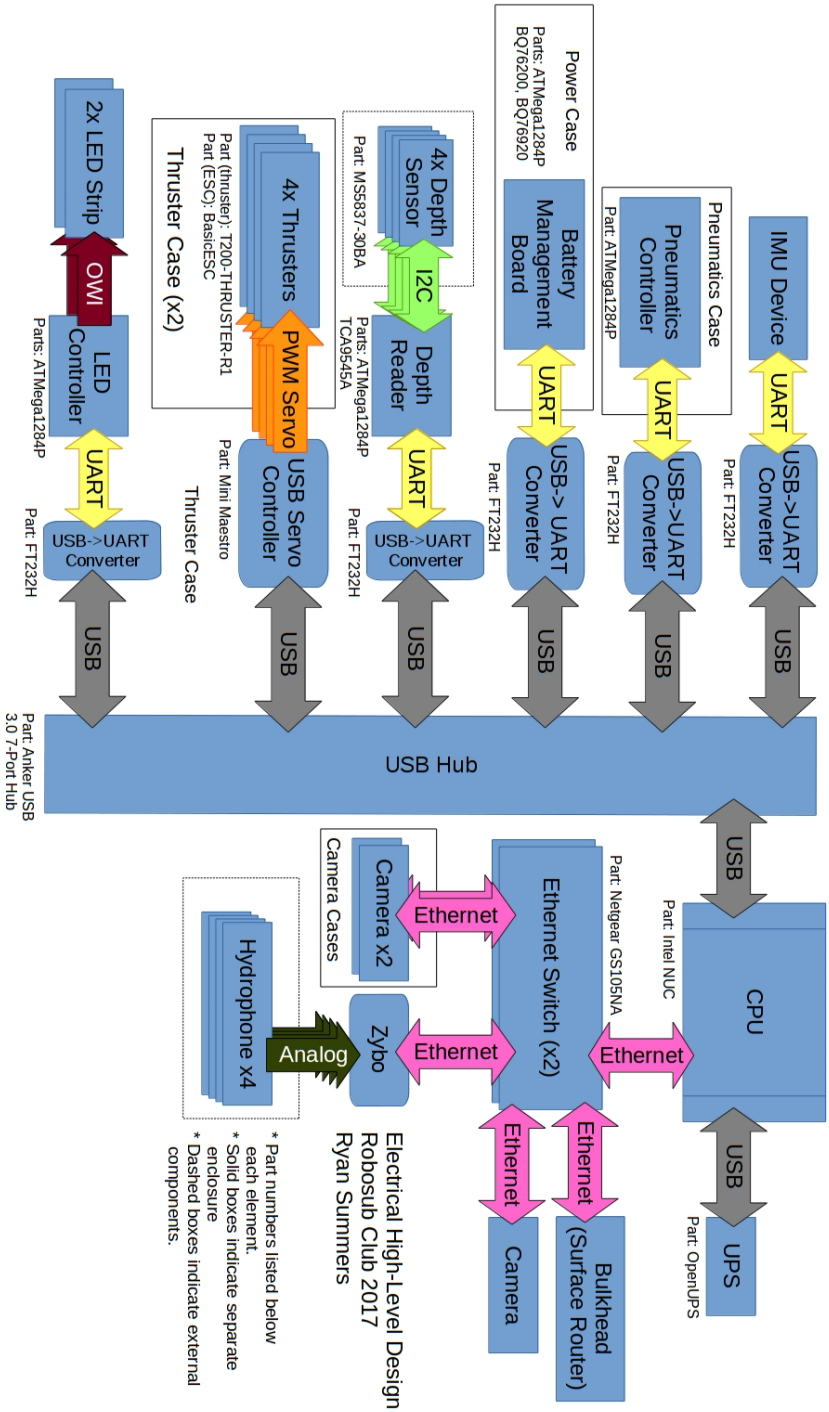
APPENDIX A
ROS NODE GRAPH



APPENDIX B
CONTROL SYSTEM DESIGN



APPENDIX C
ELECTRICAL COMMUNICATIONS DIAGRAM



APPENDIX D
POWER DIAGRAM