# Qubo: The Tortoise Who Lived

Sean Gillen, Yichao Peng, John Rogers, Ben Hurwitz, Greg Harris, Camden Miller, Alex Jiao, Beth Priege, David Linko, Jeremy Weed, Justin Becker, Justin Kanga, Justin Mayle, Kevin Wittmer, Kyle Montemayor, Ross Baehr, Dr. David Akin

***Abstract*—Qubo is an autonomous underwater vehicle (AUV) designed, modeled, fabricated, assembled, and tested by an interdisciplinary team of students at the University of Maryland in College Park, MD. The vehicle, which measures 21.00" x 24.25" x 12.50" and weighs approximately 35 pounds, runs off of either a pair of 4-cell lithium polymer battery packs or a 20V external supply capable of up to 30A. It features internal battery charging, two orthogonal cameras for vision, and a modular design for future expansion. Eight thrusters are used for motion with six degrees of freedom. The primary runs a ROS based software system that allows for flexible execution of competition tasks. An embedded system, controlled through a TI Tiva C controller board, handles the lower level tasks and sensor processing. Qubo has been designed for simplicity and flexibility, primarily with the goal of navigating the tank this year, and leaving the Robotics At Maryland team in a good position for future competitions.**

## I. Introduction

ROBOTICS AT MARYLAND is the University of Maryland's AUVSI Robosub team. our team of 25 undergraduate students has worked for three years to create our most advanced AUV to date. Tortuga IV, the club's previous vehicle, finally became obsolete in 2014. It had accrued several years of patches, updates, and lost documentation, meaning that a clean slate design was required. The name "Qubo," which means "turtle" in Swahili, was chosen for a robot that would be smaller and more modular than our previous design. Two years of team turnover due to graduation and budgetary struggles hampered this effort though. These factors left less than one year to design, build, and test Qubo. The short time frame has necessetated compromises, However Qubo will compete at Robosub 2017, and the modularity and scalability of the basic deisgn mean that Qubo will see many upgrades over the course of a long life.

## II. Design Strategy

The current version of Qubo was conceptualized over the last week of February 2017, with a preliminary design review the first week of March. This left only months until competition, meaning that the design was pared down to meet time constraints. The main goal was to have a vision system that could navigate the simplified Qubo through the course. The electrical system was drastically simplified to decrease design time and the chances of failure. Mechanically, Qubo lost a manipulator, torpedoes, a separate battery hull, and a separate motor controller hull, among several reductions. The software was simplified as well to a single $I^2C$ bus and minimal other communications, with a major focus on controls and vision. "Feature creep" was targeted and excised, keeping Qubo streamlined and focused on the most essential tasks. Estimates based on the 2016 competition showed that if we could successfully navigate the course, even without accomplishing any other tasks, Qubo would earn enough points to have placed. This confirmed navigation as our main priority. Due to time restrictions, decisions were made with less information that otherwise would be warranted, and modeling was less complete than desired. Simulations were run from the software side, but with the number of team members dwindling as some were lost to class work, the remainder worked overtime to put Qubo in the water.

Robotics At Maryland is with the Institute for Systems Research at the University of Maryland, College Park, in conjunction with the Space Systems Laboratory at the Neutral Bouyancy Research Facility under the Department of Aeorspace Engineering. Email: president@ram.umd.edu.

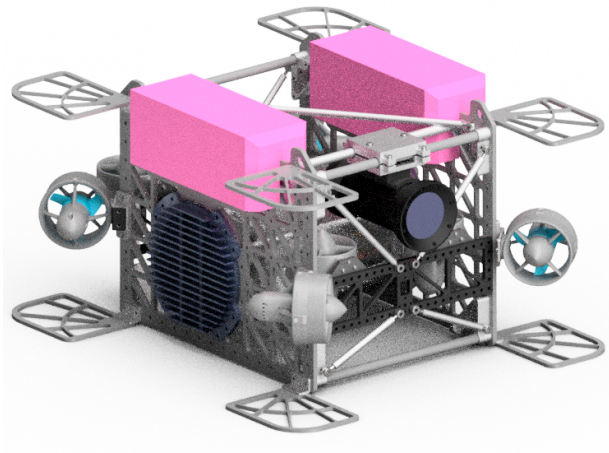

Fig. 1. CAD Rendering of Qubo

## III. Mechanical System

Qubo's mechanical system is composed of a main pressure hull, eight Blue Robotics T200 thrusters, two camera hulls, and an aluminum frame. The main pressure hull houses the electronics system and the computers. Two camera hulls are placed in the vertical center plane, one forward facing, and one downward facing. The robot relies heavily on vision for maneuvering. Mechanisms for firing a torpedo, dropping markers, and manipulating objects will be added in the future.

The mechanical design of Qubo emphasizes modularity and ease of machining. One of the lessons learned from Qubo's predecessor, Tortuga IV, was that a simple frame design can reduce the serviceability of and limit access to internal components. In Qubo, the frame was designed with

respect to the hardware configuration. The design eliminates the spatial dependency of systems, and allows for quick part removal, swapping, and modification. Additionally, the robot was designed to reduce machining complexity. Only a few of parts require CNC milling, while others can be manually machined or 3D printed.

### A. Propulsion

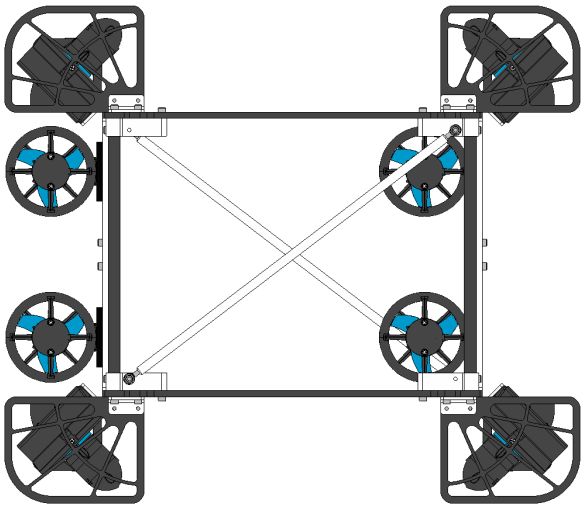

Fig. 2. Top View of Frame and Thruster Configuration

The propulsion system is intended to give the robot fine control of its movements. In RoboSub, the ability to perform forward, strafing, and yaw motion while maintaining depth is critical. On Qubo, four downward facing thrusters are placed in a rectangular pattern. Together, they maintain a horizontal operating plane which other four thrusters move on. The sideway thrusters are aligned to provide vector thrust 45 degrees from the center plane. Ideally, they provide more thrust for forward motion, and direct control for yaw. Further, the high degree of freedom allows the robot to perform complex maneuvers if so desired.

This configuration requires the thruster to quickly reciprocate and deliver similar forward and backward thrust. The T200 thruster from Blue Robotics is an affordable option that meets the required specification[1]. Each thruster is mounted via an adapter that can be quickly detached and swapped in case of a malfunction.

### B. Frame

The frame consists of two 16” x 12” x 0.25” water-jetted aluminum panels with four main aluminum rods holding them together. The water jetted pattern offers option to mount additional hardware. The front and back plate provide additional mounting space. Each side is reinforced with cross beams. Stress analysis shows the frame can withstand several times its operational load.To protect the thrusters in the event of a collision, laser-cut Delrin bumpers are fitted onto the frame. As for corrosion consideration, all fasteners used are stainless steel and the aluminum members are protected by a sacrificial anode.

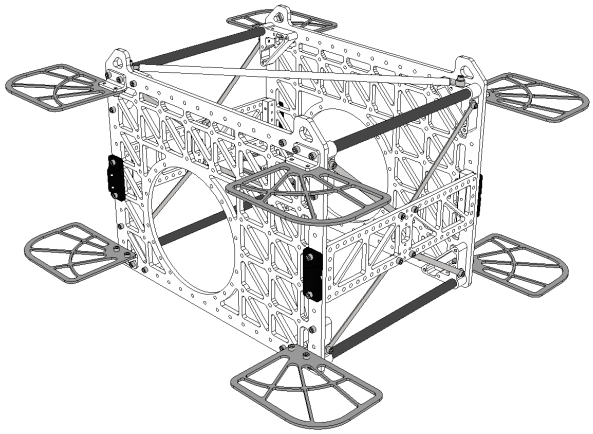

Fig. 3. CAD Rendering of Frame

### C. Main Pressure Hull

The main pressure hull is an enclosure that houses major electronic and control components. The hull measures 7 inches in outer diameter, 6.5 inches in inner diameter, and 13 inches long. It is comprised of a CPU-side endcap, a connector-side endcap, and an acrylic tube. The connector-side endcap is responsible for mounting and cooling of eight Electronic Speed Controllers (ESCs). Each ESC corresponds to a Blue Robotics Penetrator with thruster cable running through. The computer, Jetson TX100, is cooled by the CPU-side endcap, whose heat fins ensures continual operation in water and on land. Also on the inside of the CPU endcap is a four-pin scaffold on which the two batteries and main electronic board are mounted to. The pins work in conjunction with rods on the connector side to correctly alignment. The exterior of the CPU endcaps has six more Blue Robotics Penetrators, a ball valve for pressure equalization, and several spots for SubConn wet-mate electrical connectors.

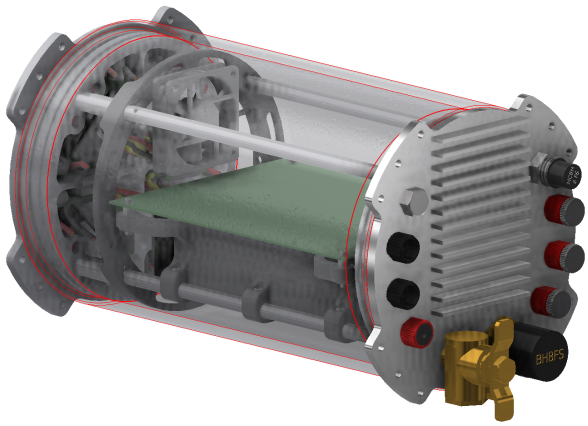

Fig. 4. CAD Rendering of Main Pressure Hull

The endcaps are CNC milled from blocks of 6061 aluminum for optimal heat dissipation and strength. The rough outline

of the parts is defined to accommodate the 12 mounting holes on the frame side panel, so that both endcaps can be fixed on the frame. Stress analysis was conducted on the hull to ensure that no parts will fail due to fatigue in its lifetime. For thermal considerations, thermal pads on heat generating components increase conduction to the endcap, and two 60 mm fans force internal air circulation. Before fabrication, a thermal CFD of the entire hull verified the system will remain in its safe operating temperature.

### D. Camera Hull

The two cameras hulls onboard Qubo were taken from Tortuga and fitted with new Mako G cameras and Blue Robotics penetrators for ethernet connection. There is also a slimmer hull built for the camera. This is used as a backup hull.

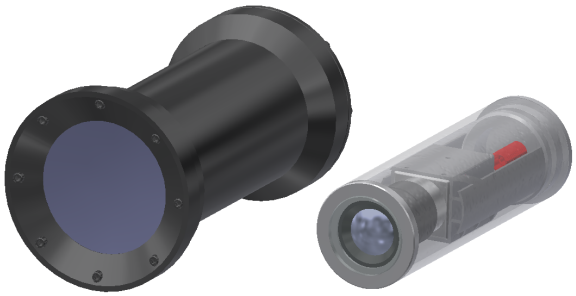

Fig. 5. Modified Tortuga camera hull (left), Qubo camera hull (right)

## IV. Electrical System

The electrical subsystem of Qubo handles all of the power distribution and signal routing required for the AUV to function effectively. It runs off two four-cell lithium-polymer batteries which it subdivides into three power rails for a variety of uses, with the embedded system generating a fourth voltage level. A relay is used as a kill-switch mechanism to the thrusters, which allows all non-moving subsystems to remain on for debugging purposes in case of a catastrophic failure. A backplane/daughterboard structure was implemented to allow more flexibility and modularity within the system, with the daughterboard handling the input voltage and switching between the batteries and shore power. The daughterboard also carries the charging solution, which will be discussed in more detail later. It allows the operator to run the system using the batteries, or run the system using the shore power connection while charging the batteries, giving maximum flexibility.

There were two primary goals that drove most of the design choices for the electrical subsystem. The first was simplicity; because Qubo required a late redesign, fewer moving parts would decrease the likelihood of individual part failure, thereby increase the chances of successful integration. This lead to a significant reduction in the number of components that were used, both in terms of chips as well as instrumentation, leaving Qubo with a much reduced capacity. It also significantly reduced costs and the time required for design cycles, allowing for significantly more iterations. The second goal limit the number of times the main hull needed to be opened. This was emphasized for two reasons. First, the act of opening and closing the o-ring seal inherently weakens the seal itself, increasing the risk of failure. The second was because of prior year competition feedback that the competition space was very dusty, which could interfere with the o-ring sealing or other internal components. By following these two design parameters, Qubo's electrical system has been designed to be both robust and flexible.

TABLE I
System-level Electrical Parameters

| Parameter | Typ. Value [Unit] |
|---|---|
| Input Voltage | 13-20 [V] |
| Max. System Current | 34 [A] |
| Max. 12V and 5V Current | 4 [A] |
| Max. Thruster Current | 25 [A] |
| Max. Operating Temperature | 65 [°C] |
| Nominal Thruster Current | 9 [A] |
| Nominal 12V Current | 1.5 [A] |
| Nominal 5V Current | 1.35 [A] |
| Steady-State Ambient Temperature | 42 [°C] |

### A. Batteries

With an emphasis on simplicity and efficiency, Qubo's batteries were selected to minimize the number of DC/DC conversions that would be required to power everything. The critical components for the system were the Jetson TX1 computer and the BlueRobotics T200 thrusters; the computer's carrying board can handle a maximum of 17V, while the thrusters can take up to 20V. There was also a physical space constraint, meaning the batteries needed to be small. However, Qubo also needed to make full runs, and potentially long test runs, without constantly recharging. Finally, the thrusters can draw a lot of current, so the battery had to be able to source at least 30A - this accounts for 4A for the four translation thrusters, 1A between the four balance thrusters, 3A on the 12V rail (the maximum computer draw), and 4A on the 5V rail (the maximum for the entire rail), plus a safety factor. A 25A fuse is used as hardware protect against over-current events. All of this culminated in the choice of a pair of 14.8V lithium-polymer battery packs, each with a 5000mAh (74Wh) capacity, for a total of 10Ah, at a steady discharge of 45C (and a burst discharge of 90C), way above the system's requirements. However, with a secondary emphasis on future adaptability, these batteries allow Qubo to expand significantly without too many changes to the batteries themselves.

### B. Power Daughterboard

The power daughterboard handles the switching of the input power between the parallel four cell lithium polymer battery packs and our "shore" power system. The shore power is a high current, adjustable, AC-to-DC voltage converter which is used to simultaneously power the entire vehicle out of the water and charge the batteries, all internal to the electrical hull.

TABLE II
BATTERY SPECIFICATIONS

| Parameter | Value [Units] |
|---|---|
| Capacity | 5000 [mAh], 74 [Wh] |
| Configuration | 4S1P |
| Chemistry | Lithium-Polymer |
| Voltage | 12-16.8 [V] |
| Discharge Rate | 45 [C] |
| Max. Burst Discharge Rate | 90 [C] |
| Weight | 480 [g] |
| Dimensions | 154 x 46 x 30 [mm] |

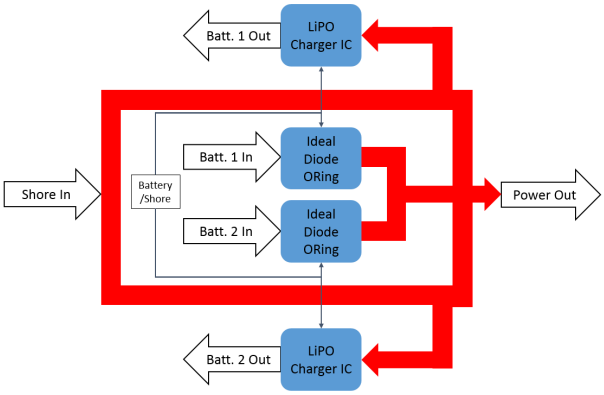


Fig. 6. Schematic block diagram of the power daughterboard.

The subsystem uses a waterproof switch to alternate between shore and battery powered operation. The switch alternates between shore and battery modes by turning on and off the LTC4364 ideal diode ORing chips and the MAX1737 battery charge management ICs. Coupled with the two different power inputs, this switch allows for three modes of operation: shore power, battery power, and system off. The daughterboard also handles to proper charging and discharging of the batteries themselves. The block schematic diagram can be seen in Fig. 6.

When the switch is positioned to battery power, the system will run only off of the batteries. It is important to note that in this mode, it is critical that the shore power connection not be in, otherwise undesired behavior may occur. Once the system is assembled, the batteries are connected and no longer removed. Their output flows through these ideal diode ORing ICs, one per battery, which provide a number of critical features. First and foremost, they allow us to safely parallelize battery packs. This is done through an ideal diode structure, which disallows reverse current flow into the batteries, preventing either pack from attempting to charge the other. These ICs also provide over- and under-voltage protections as well as over-current sensing. Finally, the Linear chips also offer a restart feature that automatically restarts the chip after a designer-specified time limit if any of the fault conditions (such as incorrect voltage or current) are triggered. This is important because it can be done without the intervention of the embedded system, adding an additional redundant safety measure. The embedded system monitors the battery voltage by monitoring the thruster voltage rail, which is merely the battery voltage. (These rails are discussed further in the Backplane section.) Lithium polymer batteries are highly susceptible to over- and under-voltage conditions since there is no innate protection. Therefore it is imperative that this monitoring be done effectively. The cells are considered fully-charged at 4.2V per cell, and should not be discharged below 3.0V per cell. In the interest of adding a safety factor, Qubo's batteries are cutoff at 3.25V per cell via the LTC4364 ICs which is backed up in the embedded system. This prevents us from easily accidentally over-draining them with some headroom for leakage.

The third mode is the shore power, or charging, mode, which requires shore power be plugged in. (Note that if shore is not plugged in, this becomes the third mode: system off.) When the switch is turned to the charging mode, the LTC4364 ICs are electrically held in shutdown mode while the charging ICs, the MAX1737, are powered on. This cuts off the connection from the batteries to the load directly via the ideal diodes. The MAX1737 is a lithium-polymer battery charge controlling IC, and, coupled with some passives, uses a switch-mode technique with a large inductor to generate nearly-arbitrarily high currents to charge these battery packs with up to four cells. It handles a number of potential downfalls with LiPo charging, including the aforementioned under- and over-voltage conditions, current control and management, thermal concerns, and timing. It uses an internal state machine to move between a constant-current "fast charging" mode, a constant-voltage "full-charge" mode, and a "top-off" state; movement between these states is dependent on internal current and voltage measurements as well as on the timer capacitors that can be added as an additional safeguard. Qubo's charging system currently runs at 4A of charging current, which is about 0.8C for the batteries detailed previously. This is a fast charging current [2], but with a 5Ah capacity, a high current was required to charge them to a reasonable level within a reasonable amount of time. The fast-charge stage will charge to about 80% of the maximum, about 4Ah, which takes about an hour, followed by a maximum of 45 minutes for the "full-charge" stages, and a second 45 minutes of "top-off". With full batteries, Qubo should be able to run for between 45 minutes and an hour of continuous movement; charging them will take approximately 2-2.5 hours. This leads to the next advantage of the MAX1737: the input voltage both charges the batteries and provides power to the load simultaneously. This allows us to continue to test either in water or in air while the batteries are charging, eliminating the lost time that would otherwise be required.

### C. Backplane

The backplane exists to route power and signals between the various subsystems and components that compose Qubo. The block diagram for this board is shown in Fig. 7. Power is applied via the power daughterboard through a Samtec

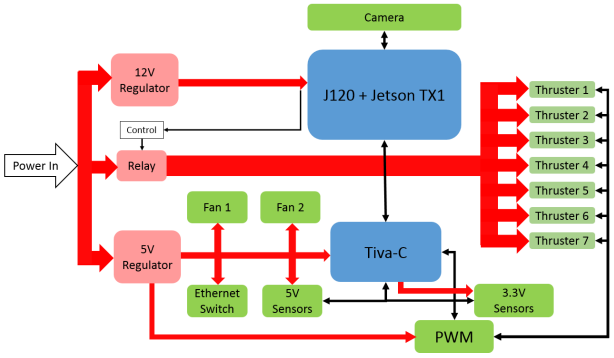


Fig. 7. Schematic block diagram of the backplane.

eighty pin connector - these 80-pin connectors were chosen for an earlier version of Qubo, and will be highly valuable in later revisions with more daughterboards, allowing for a wide variety of signals. Power is routed into three rails: 5V, 12V, and the thruster voltage, which is simply the battery voltage. There is also a 3.3V bus that is generated by the Tiva C embedded processing board. This Tiva C is the second major component of the backplane; it is this embedded computer, discussed later, that handles all of the lower-level sensors and control. The third primary component of this subsystem is the power relay that allows us to externally cut off power to the thrusters if something goes wrong while allowing the software to continue operating and logging data.

Going back to the battery selection, it was decided that the driving constraint for the battery selection was the thrusters, and that everything else would run off of a converted bus. This became the driving constraint for our bus voltages. While the Jetson can handle a range that includes the 14.8V for the battery, it was decided that because, one, shore power would need to be above the battery's maximum voltage of 16.8V, and two, the Jetson is fragile and expensive, that a 12V was dedicated to the Jetson alone. This would provide a stable power bus for the high-end computer, decreasing the chances of issues with power. Everything else runs off of a 5V bus, where "everything" refers to the Tiva C, controlling ICs, sensors, fans, and an ethernet switch (for talking to multiple ethernet-controlled cameras), in total drawing less than 4A at steady-state. This simplified the overall electrical design by minimizing the number of different possible voltages. The lone exception was our temperature sensor, the BME280; it runs off the 3.3V line sourced by the Tiva C embedded board. In a future revision, this will be revised to be a 5V alternative. The cameras were the last to be powered; they run directly off the battery voltage, before the relay, thus avoiding the 12V bus and the kill-switch mechanism.

There are a number of control and monitoring elements on the board as well. A 16-channel PWM generator chip controls our thrusters through Afro 12A electronic speed controllers over the $I^2C$ bus. ACS713 current sensors are on each voltage bus and on each thruster output for real-time current monitoring. These are coupled with a set of INA219 voltage sensors so that the embedded system can determine real-time system-wide power draw. This is critical for the percise control of our thruster outputs which can be implemented with just current readings but works much better with the power draw. There is a single BME280 temperature sensor near the power generating ICs; with the two internal fans, this is enough to get a good reading on the internal temperature without worrying about local variations.

### D. Computers

Qubo has two computers on board, one for low level control of sensors and other hardware, and the other for high level computations. The high-level computer is a Nvidia Jetson Tx1. The Jetson has a beefy 1 GHz 4 core ARM processor, on-board Maxwell GPU with 1 TFLOP of power, and only draws around 10W of power under normal loads at 12V. This makes it ideal for handling our large computational needs without putting excess strain on our power system or causing thermal problems, especially since the primary use for the main computer will be the vision processing.

The low-level computer is a TI Tiva C microcontroller. The Tiva is powered by an 80 MHz ARM cortex-M4 processor with 256 KB of flash and 32 KB RAM. This controller runs what is called Qubo's embedded system; it handles the monitoring of all systems, reading sensors, and commanding our thrusters. The two computers are connected directly via USB, and communicate using our custom protocol, Qubobus. The Tiva C has four different $I^2C$ bus terminations which are both 5V and 3.3V tolerant, which lends more flexibility to the system, and can also communicate over many other protocols, including USB, SPI, and CAN.

### E. Thrusters

The BlueRobotics T200 thruster was chosen to drive Qubo. There were several reasons for this choice. First, they were in widespread use in the 2016 competition, and research indicated that teams were very satisfied with their performance. Second, their technical requirements made them an easy fit into Qubo, with a wide range of input voltages, PWM controls, and simple three-phase design. This made integration with the third-party motor controllers physically straightforward (though the software integration was rather more involved). Further, the thrust-to-power curves were friendly to our design, since our light-weight robot requires significantly less power draw than previous thruster choices. The third reason was that the Space Science Laboratory was using them for another vehicle, and was planning to do extensive underwater testing with them that we could use as well. For these reasons, Qubo runs on eight T200 thrusters, giving it a full six degrees of freedom with redundancy.

### F. Electrical and Mechanical Integration

The integration of the electrical and mechanical subsystems was one of the trickier tasks in Qubo's development. Originally, the electrical hull housed the backplane, power distribution, and main computer stack only, requiring bulkheads for every other instrument and device external to that hull to be connected via a bulkhead. This allowed for maximum modularity, though it was far more complicated to design. With the

shift to the pared-down version three, most of the bulkheads were removed in favor of simpler and cheaper penetrators, which a relatively easy to replace in the middle of competition. Further, with far few external electrical components; only the cameras and shore power, it was an easy way to cut costs. The shore power connection remained a bulkhead so that it could be disconnected easily, as did the tether connection. The Teledyne Wet-Mate-able Impulse series cabled and bulkheads were chosen for their widespread use in industry, at prior Robosub competitions, and the recommendation of the SSL graduate students and Dr. Akin. An interesting solution for the cameras was found by using an RJ-50 cable. The cameras, the Allied Vision Mako-G series, are ethernet-controlled with a 12V input and less than 0.4W of power drawn, and required a more interesting solution to their external connection. An RJ-50 cable (a 10-pin ethernet-like cable) was used to provide both data via four twisted-pairs and power and ground via the last pair. By using a single cable, the penetrators were better able to maintain a seal around the cable itself, reducing the chances of leakage or failure.

## V. Software

The software system for Qubo is divided into two categories depending on which computer they are run on. High level software runs on the Jetson and embedded software runs on the Tiva.

Both of these code bases were written with several guiding principles in mind, the first being modularity. We wanted to maximize the amount of code that can be used between iterations of Qubo and make adding new capabilities straightforward. This goal was at times at odds with the desire to make our code perform as fast as possible. While we do strive to write fast code, we made some compromises in the name of flexibility that reduce the performance of our code.

We also wanted to make our code easy for new members to parse and contribute to. To aid this we attempted to keep our code simple and small. We made active efforts to revise our code every few months and to remove experiments or unfinished projects from the main branches of our code base.

In addition all of our code is free and open sourced, available at Github.

### A. Embedded Software

The Embedded system was written using the FreeRTOS real time operating system. We chose FreeRTOS because we needed a system that would be able to react in real time to multiple inputs, and one that would be able to seemlessly handle multiple tasks at once. Through the software examples Texas Instruments provided, we were able to create a simple Makefile build system that utilized TI's Tiva C libraries in conjunction with FreeRTOS. Using FreeRTOS, we are able to create multiple "threads" of execution, called tasks, that run on the Tiva. Each of these tasks handles a different aspect of the embedded system, such as communicating with each sensor, commanding each of the thrusters, and maintaining communication with the Jetson. Using FreeRTOS queues, semaphores, and notifications, we are able to communicate between tasks safely and effectively.

The embedded software's source code is separated into into directories called tasks, lib, interrupts, and include. Each subdirectories have its own include directory that contains headers that pertain only to that directory. The tasks directory has one file for each task, following a similar structure. The lib directory contains code that can be used multiple times and from any task, such as sending or receiving data on the $I^2C$ bus and interacting with hardware. The interrupts directory contains interrupt handlers that are written and then linked to in a startup file. The global include directory contains headers that don't have associated source files. These may be constants or extern variables that can easily be shared across the system.

Drivers included from TI's software are stored in a directory above the source code. The driver headers' location is passed into the compiler and reachable from any source file we write. Our microcontroller has all the drivers we need stored on a ROM chip, so we were able to preserve the flash memory for our own code.

*1) Qubobus:* Qubobus is a byte stream protocol that describes implements the format of messages between the Jetson and embedded system. There is a handshake procedure to connect the devices on startup. It includes protocol version negotiation, checksum for data integrity, and KeepAlive for connection integrity.

### B. High Level Software

The high-level software was written using the Robotic Operating System (ROS) as a base. The primary advantage of ROS is the large library of utilities it comes with. This includes OpenCV for image processing, Actionlib for handling long lived inter process coordination, and the Gazebo simulator are all modules that we used heavily. In addition to the obvious utilities, ROS also benefits from large community support. There are many forums around to help answer questions or address problems that we come across.

The high-level software is divided into four different section as seen in figure8.

*1) Vehicle Layer and Simulation:* The vehicle layer block is a hardware abstraction layer that handles all parts of the system that are not vehicle agnostic. In practice this means it contains a thruster management node, and nodes for our different sensors. The output of this node is a 3D orientation, a depth reading, and either one or two camera feeds. The thruster management node takes commands from the control system to it's six degrees of freedom, which it then breaks out to the corresponding real or simulated thrusters.

The way in which we handle our camera feeds provides an example of the trade offs made between flexibility and performance. We decided to use a camera node which takes our cameras and publishes the feed as a ROS message that is consumed by our vision node, rather than have our vision node directly pull data from our cameras. This decision was made to accommodate our simulator, which uses ROS messages to publish all vision data. By adding a camera node to abstract

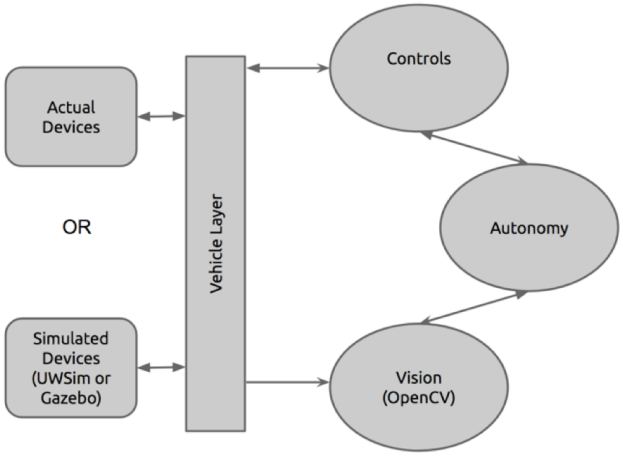


Fig. 8. High level software overview


Fig. 9. View from the robot

away the camera interface we are able to use the exact same vision code for our simulator as we do for our real robot, at the cost of taking a performance hit.

For our simulation we used the Gazebo package because of it's popularity and tight integration with ROS. To add realistic underwater simulation capabilities to our simulator we used a package called UUVSimulator, developed by a research team in Germany [3]. This package allowed us to simulate our robot's buoyant forces and account for nonlinear drag terms. The realistic drag and buoyancy terms greatly aided in the development and tuning of our control system. Furthermore Gazebo allows the simulation of cameras and can place arbitrary meshes in the robot's environment. This allows a crude simulation of the robots visual environment. While not useful for tuning vision algorithms, it is very useful for testing our vision systems integration with our controls.

*2) Autonomy:* The autonomy node is where all the decisions are made and is the node responsible for calling all other parts of the vehicle to action. We implemented a state machine architecture, which performs well in the deterministic environment of RoboSub. Each task has one state in the state machine, and includes several recovery states if we lose sight of an obstacle. An addition advantage to using a state machine is its relative flexibility. If we want to add a different task to our run, it's as simple as adding a new state to the existing machine.

*3) Vision:* Our vision node handles all the image processing required by this competition. It offers ROS services and Actionlib actions to the autonomy node to call. Our actions are meant to locate the target in the frame, and output the distance in pixels from the center of the frame to the target. We have made great efforts to make our vision system as modular as possible. The result is a system where you can "plug and play" different algorithms, without needing to rewrite the boilerplate required to communicate with the rest of our software system. This has made prototyping different algorithms a much more efficient process.

Fig. 10. view after MOG background subtraction

We're also aware that no matter how well we prepare our algorithms, we will need to make adustments to them at competition. To this end, for every algorithm we develop we also develop a "tuner" program. The tuner allows us to run the algorithm step by step, and to change parameters and see the results in real time. When we couple this with ROS's ability to save a play back camera feeds from our runs, we are able to make the changes we need quickly and efficiently.

One algorithm we've found especially fruitful is the Mixture Of Gaussians (MOG) background subtraction algorithm. This allows us to filter out the noise inherent in the course (algae on the walls, water ripples, divers, etc.), which has been very useful when used in conjunction with our normal vision detection algorithms. After we do background subtraction we are able use Haar lines to detect the gate and channel tasks, and a simple blob detector for the buoys. Preliminarily these techniques seems to work very well at robustly identifying our targets.

*4) Controls:* The control system is kept simple with a PID controller around each of our Degrees of Freedom. We get most of our movement queues from the vision system. For example, in the buoy task we have the vision process output

the distance in pixels from the center of the cameras frame to the buoy, and then feed this as the error term to our yaw and heave DOFs. When the error is sufficiently small, we surge forward. Our vehicle is naturally stable about it's pitch and roll axes, and we actually found our control system works better when we make no attempt to activley control these DOFs.

## VI. Vehicle Testing

### A. Mechanical

Many of the mechanical parts are tested before fabrication. All the components are assembled in Autodesk Inventor, down to bolts, to eliminate interference. Multiple 3D printed parts was produced for test fit and machining discussion. Extensive stress analysis was conducted on frame and pressure hulls to ensure a high factor of safety. Additional thermal CFD simulation was conducted on main electronic hull to examine systems cooling performance during operation.

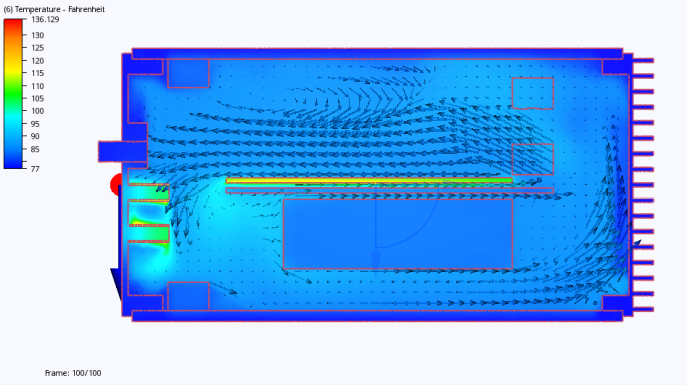


Fig. 11. Main Pressure Hull Thermal CFD in Autodesk Simulation

For physical testing, we conducted part by part study. Performance of the magnesium sacrificial anode was tested by placing it with a piece of aluminum in the tank for an extensive period. The result allowed us to estimate the usage life for an anode piece. We also produced a test hull to observe if Blue Robotics penetrator can prevent ingress of water at depth of 20 ft. Lastly, we tested the marine epoxy for its sealing capabilities using similar methods. In the coming weeks, we aim to perform more system level testing with a fully assembled robot.

### B. Electrical

The electrical subsystem's testing consisted of modeling critical circuitry using various flavors of SPICE to begin with, and then after fabrication various electrical tests are done to look for potential concerns and issues. Fig. 12 shows the layouts that were designed for the 5V rail using the LT8640 and Linear Technology's LT Power CAD II, with similar modeling being done for the 12V rail. CAD modeling and thermal modeling were done in NX as well to determine localized hot spots and internal air flow, as well as physical separation and spacing of the components.

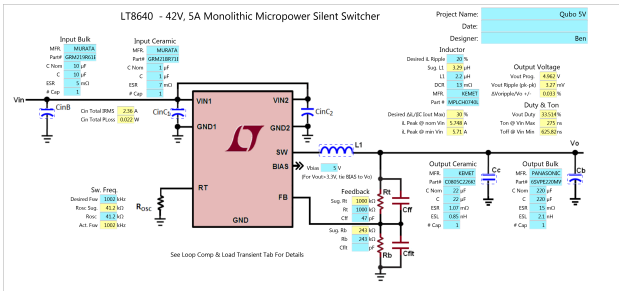


Fig. 12. The schematic for the 5V rail.

### C. Software

Testing of the software system benefited immensely from our use of the Gazebo simulator. Not only were we able to test higher level functionality while waiting for Qubo's hardware to be complete, but we were also able to sanity check new features without needing to deploy the robot. Of course there is no replacement for in water testing, but our simulator greatly sped up our testing process.

Another very valuable resource has been the collection of competition footage from previous years. These videos allow us to test our vision algorithms on the actual conditions we can expect at robosub, which allows us to develop more robust algorithms in advance of the competition. We will also make sure to keep video logs of the runs to allow future teams additional data.

### D. Complete System

No complete system integration testing has taken place as of this writing. Mechanical models have been completed of the entire system in NX, and complete system software modeling has also been done using Gazebo.

## VII. Conclusion

The AUV called Qubo was conceptualized, designed, fabricated, assembled, and tested over a five month period in 2017 by a small team of student engineers at the University of Maryland. It was made to be a simple and robust, with a focus on vision and navigation tasks. The platform is conceived to be adaptable, with many upgrades possible over the coming years. This version of Qubo is just the start of many generations of this AUV, and with a strong base, these newer iterations can attempt more tasks and with better efficiency than before. New members will be able to be quickly integrated into the team, and with a fully functional robot to test and see results on, they will remain involved for longer. The lessons learned from these years of hard work will be used for many generations of projects to come, both at Maryland at beyond.

## TEAM PHOTO

Fig. 13. The Robotics At Maryland team.

## ACKNOWLEDGMENTS

Robotics at Maryland would like to thank Dr. David Akin, our faculty advisor. His expertise in autonomous systems design was invaluable to our project. We would also like to thank the Space Science Laboratory graduate students, who lent us their considerable RoboSub experience and their valuable time. The University of Maryland Robert H. Smith School of Business, Electrical and Computer Engineering Department, Clark School of Engineering, and Computer Science Department, all donated generously to support Robotics at Maryland. DesignME contributed the use of their machines, Terrapin Works offered their support, guidance, electrical fabrication and assembly tools and equipment, and CCTV Camera World donated several cameras that proved very useful. We appreciate their generosity and dedication to robotics education.

## REFERENCES

[1] "T200 thruster documentation," http://docs.bluerobotics.com/thrusters/t200/, Blue Robotics Inc., 2017, accessed: 2017-03-10.

[2] S. Keeping, "Designer's guide to lithium battery charging," https://www.digikey.com/en/articles/techzone/2016/sep/a-designer-guide-fast-lithium-ion-battery-charging, Digi-Key Corp., September 2012, accessed: 2017-02-27.

[3] M. M. M. Manhães, S. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach, "UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation," in *OCEANS'16 MTS/IEEE Monterey*, September 2016, pp. 1–8.