RoboSub 2026 Technical Design Report

Underwater Robotics at Berkeley
University of California, Berkeley
Berkeley, California, USA

***Abstract*—This paper covers the design, manufacturing, testing, and implementation of Tardigrade, an autonomous underwater vehicle created by a team of graduate and undergraduate students at UC Berkeley. This year, the team expanded on the foundation built during RoboSub 2025, building on top of the existing mechanical, electrical, and software infrastructure to adapt Tardigrade to the new challenges presented by RoboSub 2026.**

I. Competition Strategy

Our strategic vision for the Robosub 2026 competition consisted of two main principles: improving and refining our existing design from last year's competition, and designing new subsystems to ensure we could complete every core task. Due to the massive growth our team saw this year, we were able to devote more manpower to these goals than in previous years and successfully increase the complexity of our existing AUV, Tardigrade, while maintaining its functionality.

To complete these two goals, we focused our efforts on refining both our frame and dropper designs, improving our existing electrical system and its organization, and revamping our perception/controls algorithms with YOLO and Behavior Trees. Meanwhile, we also worked on overhauling our torpedo launching system and adding a claw mechanism so that we could complete this year's Resupply task.

Shifting our focus towards completing at least the core aspect of every task meant we had to deal with increased complexity in Tardigrade, particularly in the software stack, which has to account for the state of each mechanism and activate them accordingly. However, completing each core task allows us to score more points without solely relying on one subsystem. This gives us a more solid base to build from as we refine these designs for future competitions.

II. Design Strategy

A. Mechanical Design

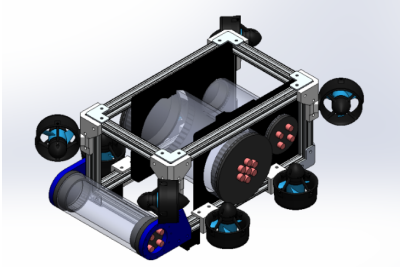
1) Frame: Our robot's frame remains similar to past iterations, with a base of aluminum 80/20 T-slots that allow modularity; we are able to slide our components and adjust spacing as needed to best fit the requirements of the competition. The quickest way for us to manufacture complex parts was through 3D printing, so frame components are made of PETG filament, which is stronger and more resistant to water intrusion. Combined with watertight housings for electronics, batteries, and cameras (cylindrical for optimized sealing, even under pressure), our materials were chosen primarily to be adaptable to changes in design, as well as high strength and low weight. Additionally, we use Solidworks simulations and dunk testing in order to evaluate the buoyancy of our robot, making small alterations by adding foam blocks and pool noodle pieces until we reach positive buoyancy.

This year, we simplified internal electrical integration and increased accessibility of electrical components by pushing all wires out of one side of the robot, allowing us to easily isolate the wires that correspond to each individual task and ensuring control of the robot's functions for each aforementioned task.

Tardigrade's thruster configuration consists of eight thrusters arranged to provide six degrees of freedom. This allows complete freedom of movement in an underwater environment and completion of tasks "with style" to score additional points. The thrusters are also placed close to Tardigrade's center of mass, which makes small, precise rotations more consistent at the cost of making larger rotations more difficult. Facilitating small adjustments is the more important of the two because the torpedo, claw, and dropper tasks rely on precise stabilization and positioning for optimal scoring.

Tardigrade's frame design provides a strong base on which we can build our robot's three primary subsystems: the torpedo, dropper, and claw.

Fig. 1. Isometric Frame View



2) Dropper:

3) Torpedo: For the torpedo system, our main goal was to increase the accuracy of the torpedo and the distance it could travel, so that we could fire through the designated openings of the deploy task from the far distance (1 foot) and maximize the number of points we could earn. Our torpedo launching system consists of a mounting block that attaches the system to the frame of Tardigrade, springs that act as the basis of our launching mechanism, a latching system

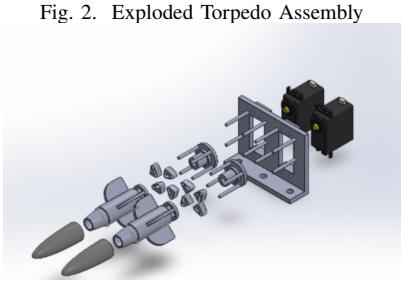
actuated by two waterproofed servos, and two identical torpedoes. The spring loaded design is simple, and allows us to fire more accurate shots with limited access to precise machining tools.

We started our torpedo design process by choosing the profile of our nose cone[1]. We settled on the elliptical profile as our nose cone shape, as this was the most hydrodynamic choice, which means we could increase distance traveled by minimizing the drag forces the torpedo experienced.

We then designed the torpedo to be hollow on the inside and split into two pieces that could be slotted and glued together. This was so we could put weights inside to optimize buoyancy, as a torpedo with neutral buoyancy fires with the most accuracy. We also designed the torpedoes to be small in comparison to the deploy task openings to increase our window of success for the torpedo making it through.

Our latching mechanism, made up of four latches that fit into slots at the base of the torpedo, uses a waterproof servo as an actuator. This servo rotates, causing the latches to shift until they release the torpedo, allowing for the spring to expand and the torpedo to launch. This design creates stability and consistency when firing, and reduces the drag felt by the torpedo, as our previous iteration of this design had to have slots on the outside of the torpedo that made the design significantly less hydrodynamic. Our new latching system increases the distance that the torpedo can travel and the accuracy with which it fires.

Fig. 2. Exploded Torpedo Assembly

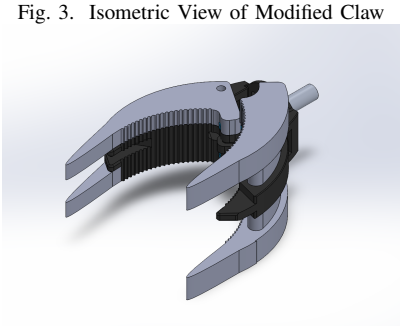


4) Dropper: The dropper mechanism is simple by design: a servo-operated shutter that opens to drop pre-loaded objects at the desired locations. The mechanism is attached to the back of Tardigrade, which both increases accuracy by putting it closer to the downward facing camera, and maintains a more even distribution of weight around the center of the robot.

The dropper objects are 3D printed in a teardrop-like shape with higher infill density at the rounded end to facilitate a straight, vertical path through the water. While the lower weight of 3D printed parts is typically a benefit, the dropper objects need to be heavy to overcome buoyancy and remain stable while falling through water. To add weight, the dropper objects are printed in two parts and a copper weight is inserted during assembly, which increases the stability of the objects and the accuracy of the mechanism.

5) Claw: Due to limitations in strength, consistency, and space within the chassis, the team decided to opt out of a magnetically coupled lead-screw mechanism in favor of modifying a stock Blue Robotics Newton Gripper. This actuator delivers significantly more clamping force while resolving previous waterproofing issues. The aforementioned space constraints within the chassis restrict mounting to a vertical orientation, as any other orientation would require a complete redesign of the robot. Furthermore, to ensure it

Fig. 3. Isometric View of Modified Claw



could grasp the competition objects, the team needed a reliable way to expand the gripper's baseline jaw width. After considering fully replacing the claw arms versus creating modular attachments to slot into the existing actuator, the team ultimately chose the modular design. Because grabbing an object places the highest stress at the pivot point, keeping the stronger stock plastic claw at the base minimizes the risk of mechanical failure. The weaker, 3D-printed PETG attachments are then restricted to the tips, where forces are significantly lower.

Finally, prototyping revealed that PETG plastic did not provide enough grip to successfully pick up objects. To resolve this, we introduce compliant grippers, which deform to maximize surface contact and prevent the claw from slipping. Compliant grippers also allow the claw to adapt to varying object orientations and offer a greater margin for error in alignment as Tardigrade approaches an object. Printing limitations precluded the possibility of printing compliant TPU claws, so we adopted a simpler approach using weatherstrips as padding instead. Its effectiveness was verified through further prototyping and testing.

B. Electrical Design

The primary objective of this year's electrical design is accessibility. While working with the robot last year, we often found it challenging to remove and replace electrical components, as we did not have designated space in our main electronics housing for each component. This year, we aimed to change that by putting a shelf in place to separate the power distribution board from the rest of the electronics; our Jetson slots onto the top of the shelf to ensure its stability while the robot moves, while other, smaller components like the Pixhawk flight controller are placed on the side of the shelf opposite the Jetson.

In terms of overall electronic design, we are using a 22.2V 6S LiPo battery from BlueRobotics, held in a separate housing unit from the rest of the electrical components. The battery is connected to the main electronics housing unit, where its wire is spliced to connect to the power distribution board, a buck converter to step down the input voltage to the Jetson, and the Pixhawk power module, which ensures the Pixhawk operates at a steady voltage.

The power distribution board was custom-made by the team and contains eight ports to which our ESCs connect. The ESCs are then connected to the Pixhawk's pin header via PWM wire, while the ESC power wires are connected to those of the thrusters via snap connectors. The robot's subsystem servos are wired to the pixhawk pin header as well. To maximize efficiency in terms of space usage, we cut down the ESC power wires as much as possible and organized the ESC-thruster connectors to be atop and to the sides of the Jetson. This allows us to keep the thruster wires away from the rest of the electronics, thus making them more easily accessible and mitigating the potential for wiring issues.

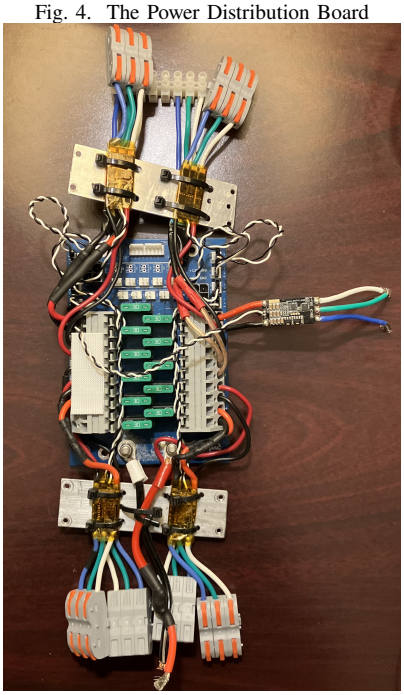
To enable communication between the Jetson and the Pixhawk, we are employing the Micro XRCE-DDS agent; its low overhead and CPU usage is ideal for the Pixhawk, which can focus on motor control and low level control loops while leaving the computer vision to the Jetson.

With respect to sensing and perception, we have a housing unit for our main camera, the Zed 2, in the very front of the robot. The Zed is equipped with the depth perception necessary for the robot to complete its tasks underwater. In addition, we have a second, smaller, waterproof camera, the Arducam, placed on the underside of the robot that faces downward. The Arducam's sole purpose is to detect lane markings in the pool and help keep Tardigrade on course in a straight line. The most significant hardware addition we have had since last year's competition was the Vectornav VN-100 IMU; last year, we relied on the Pixhawk's internal IMU, but found it difficult to calibrate. The VN-100 comes with a plug-and-play SDK that fits in seamlessly with the rest of our electronics, simply connecting to the Jetson and providing constant, reliable data about the robot's motion.

C. Software Design

1) Perception: When competing underwater, there are certain considerations we need to take into account to ensure our robot is able to perceive its surroundings clearly and accurately. While the classical CV methods achieved major goals during the majority of our testing, there were instances where the accuracy of our robots ability to pinpoint targets within the pool was compromised by the distortion of the water or inconsistent lighting within the pool environment. This meant our robot's main method of visualizing was inhibited and susceptible to malfunctioning when experiencing the standard competition testing environment. In order to ensure that the perception of our robot is capable of accurately detecting objects despite these interferences, we established our goal this year to be centered around developing a more robust form of perception.

Fig. 4. The Power Distribution Board



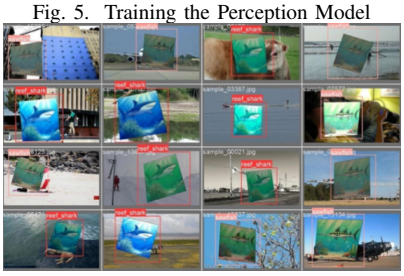
This year, we transitioned away from a classical manual feature gate-only classification towards a more robust YOLO object detection model. From competition requirements involving tasks with object detection in diverse underwater environments, we decided to pursue this approach. We trained models using generated data and augmented changes like blur and lighting distortion to match underwater environment conditions. These decisions were made to fit the challenges where we needed to detect multiple objects. YOLO was chosen since it provides higher accuracy for real-time object detection and better generalization using our data generation pipeline. Multiple model variants like YOLO nano were evaluated during this process, involving many performance metrics that led to making these decisions.

A synthetic data generation pipeline was modified to fit this year's objects for detection. This generated 30,000 images to train the model and the pipeline was created due to limited data available. The team added randomized augmentations like stretching objects over backgrounds and simulating blur to improve detection accuracy. Additionally, we decided to do multi-model evaluation to consider different tradeoffs with speed and accuracy that fit the main challenges we wanted to attempt.

The main challenge in our final design transition was having to change our existing features with a classical CV approach from past years to working with new YOLO based models. These iterations mainly used classical CV, which

required the team to modify existing workflows to fit the current design. Despite the reworking, the resulting model produced an improved detection accuracy overall. Further, when compared to our prior classical CV approach, the YOLO model showed a much better performance when tested with footage containing inconsistent pool lighting and distortion, fulfilling our goal of developing robustness within our robot's perception.

Fig. 5. Training the Perception Model



2) Controls: The Robosub competition consists of many different tasks over a considerable amount of time; because of this, the possibility of error occurring throughout the course is inevitable. To help mitigate accumulating errors and improve the consistency of our robot, we decided to implement a behavior tree control algorithm. The behavior tree receives real-time data from our IMU and camera stream, and treats it as an estimate of the current position of the robot. This gives us a clean representation of what actions aren't succeeding and creates failsafes to prevent them.

We couple our behavior tree algorithm with ROS2 Foxy on our Ubuntu 20.04 machine. We opted for ROS2 Foxy specifically because it was compatible with our onboard computer, and it allows us to simplify movement actions. For our broader movement logic, we first initiate a ROS2 topic that receives IMU and camera data, which is then subscribed to by the behavior tree. A script turning this data into a more structured pose estimate is fed back into the behavior tree, which then controls the orientation of the robot based on its position in the tree.

This allows us to develop a high-level overview of what the robot's course of action should be, and perfect its movement with the implementation of fail-safes throughout the behavior tree. The most significant challenge in creating this system was turning the IMU and camera data into a reliable state estimation. Due to IMU noise and the inconsistency of camera lighting during testing, we were often stuck over- or under-adjusting parameters based on real conditions. As a solution, we continuously tested different lighting and IMU noise in simulated environments, where we then made tweaks to our code, resulting in a much more consistent performance across the board.

III. Testing Strategy

A. Mechanical

1) Torpedo:

a) Simulated: For initial torpedo testing while we were working through iterations of our design, we used the computational fluid dynamics software, Fidelity by Cadence, to run simulations of our torpedo firing underwater.

b) Empirical: Empirical torpedo testing was done by firing the torpedo underwater and measuring how far it launched first, then testing its accuracy by firing it repeatedly at a board and marking where it hit.

2) Dropper:

a) Empirical: Dropper objects have been only tested empirically, letting the object fall through a container filled with water to see the path the object will fall into and how straight this path is relative to its initial dropping position. Repetition of this strategy has proven the dropper shape to be reliable.

3) Claw:

a) Empirical: Initial testing of different PETG claw prototypes across various objects weights and surface profiles demonstrated insufficient results, revealing the need for compliant weather strip padding. Subsequent testing with compliance on the same objects yielded much more consistent and reliable results.

B. Electrical

1) Simulated: Simulated testing has proven difficult for the team in recent years due to knowledge gaps from the pandemic. For example, the PDB was custom-made by the team prior to the pandemic and its creators did not pass down any documentation on it. As a result, the vast majority of the electrical testing thus far has been empirical.

However, the electrical team is currently working on building a KiCAD schematic and layout for the Tardigrade's entire power distribution system that will enable us to better understand how robust it is and what future improvements can be made to ensure its integrity and efficiency.

In addition, having a proper simulation system allows us to avoid much of the grunt work of empirical testing such as removing, disconnecting, and rewiring electronic components that often prove tedious.

2) Empirical: While performing empirical testing, it became apparent that there were multiple potential points of failure that needed to be addressed. Firstly, the three-way splice of the battery wire (to the Jetson via buck converter, pixhawk power module, and power distribution board) presented challenges with sharing current between components. During last year's competition, we had issues powering the Pixhawk; it would begin the arming sequence, but power off abruptly, and then need to re-arm. As a result,

this year, we worked to prioritize the integrity and robustness of all connections between components and power.

The addition of the Vectornav VN-100 introduced a new layer of complexity to our electronics system. While last year, we relied on the Pixhawk's internal IMU, the VN-100 required us to reconfigure some of our connections to the Jetson; the VN-100 and Zed-2 both connect to the Jetson's USB port, necessitating the use of a USB to USB-C dock so that we could utilize the USB port for the camera and the USB-C port for the IMU. As a result, it was crucial to ensure that the VN-100 properly powered on once the Jetson did and that it calibrated properly once connected to power.

C. Software

1) Simulated: We primarily focused on testing the data generation pipeline and the YOLO object detection model. For the YOLO model, we tested it on synthetic images of the gate placed in a background similar to real underwater environments, making sure to mimic inconsistent environments by adding blur, varying angles, and different lighting conditions. We then evaluated how often the model correctly classified and localized the gate using bounding boxes. Meanwhile, for the data generation pipeline, we visually inspected generated images, ensured that every background image contains a correctly placed gate with an accurate bounding box annotation, and verified that the YOLO label formatting is correct.

2) Empirical: Our empirical testing strategy involved evaluating the YOLO object detection model through real pool images and footage collected from past testing. Unlike the images used for simulated testing, the footage contained real underwater distortions like inconsistent pool lighting, reflections, movement, and natural blur. We evaluated the model on how often it correctly classified and bounded the gate in this real footage. To measure improvement, we compared the object detection accuracy rates of the previous CV model and the current YOLO model on the real footage, allowing us to determine whether the YOLO model generalizes better to real pool environments.

3) Integration: Integration testing validates the full software stack on the assembled vehicle on dry land before any in-water session, using the sequenced launch description tardigrade_test.launch.py. The launch exploits ROS 2 OnProcessStart event handlers to bring nodes up in strict dependency order: the micro-XRCE-DDS agent to the Pixhawk, then the ZED 2 camera, then the visual-odometry bridge that forwards pose to the Pixhawk, then the offboard mode-keeper, and finally the mission node. This ordering isolates a class of defects that neither simulated nor empirical perception testing can reproduce - DDS topic-discovery races, QoS-policy mismatches between PX4's BEST_EFFORT and TRANSIENT_LOCAL topics and downstream subscribers, micro-XRCE-DDS agent timeouts, and driver-respawn behavior under transient failure.

Each integration run executes against a written checklist of expected topic frequencies, vehicle-status arming preconditions, and pose-convergence behavior, with a go/no-go gate for water entry. Failure of any precondition terminates the run before deployment and triggers capture of the preceding 30 seconds of bagged topic data for offline analysis. Every in-water run is similarly recorded as a ros2 bag of all sensor topics, mission state, and actuator commands, tagged with pool, water-clarity estimate, lighting condition, and tether status, enabling post-session reproduction of any observed failure mode in the simulator.

IV. Acknowledgments

None of this would have been possible without the continuous support from our alumni, our staff advisors, the Etcheverry Hall Machine Shop Staff, the UC Berkeley College of Engineering, the UC Berkeley Engineering Student Council, and the UC Berkeley EECS Makerspaces, who made sure we had all the resources and assistance we would need to get to Robosub 2026. Alongside them, we would also like to thank Solidworks, Cadence, Polymate, and HOTO Tools for sponsoring our team and giving us the funding, software and tools required to make Tardigrade a reality.

V. References

[1] G Crowell Sr., "The Descriptive Geometry of Nosecones" http://servidor.demec.ufpr.br/CFD/bibliografia/aerodinamica/Crowell_1996.pdf