Tartan Autonomous Underwater Vehicle
Design and Implementation of TAUV-20: Kingfisher

Tom Scherlis, Joey Wood, Advaith Sethuraman, Joy Sodon, Maheer Sayeed, Rishabh Jain and Shubhankar Agarwal.

Carnegie Mellon University

Abstract - TartanAUV (TAUV) is a second year Robosub team driven by undergraduate students from diverse backgrounds and disciplines. We set out to use the knowledge we acquired last year to design an enduring AUV platform to use for the next several years. This meant a complete overhaul of our engineering processes, and the construction of our second vehicle TAUV-20: Kingfisher. Kingfisher is extendable, easy to maintain and repair, and larger than its predecessor Albatross. To support the addition of a Doppler Velocity Logger, rotating sonar, and stereo vision, we rewrote our software stack with 3D perception and navigation in mind. We replaced our off-the-shelf external controller with an integrated model predictive controller, allowing us to more accurately execute complex 6DOF maneuvers.

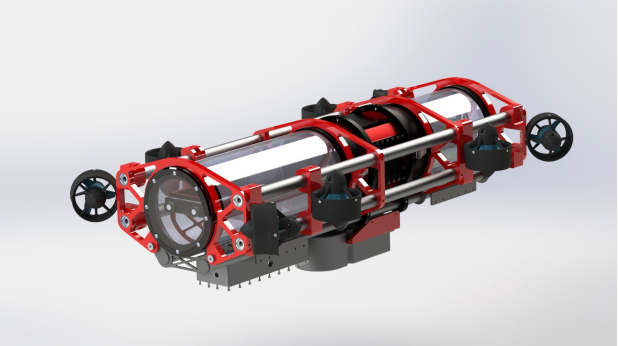

Figure 1: Kingfisher

1 Competition Strategy

As a second-year team, TartanAUV's driving goal was to design an AUV that is sustainable for the future. The first year was a great learning experience for us and had a great impact on our strategy this year. One of the biggest learnings was to focus on excelling in a few tasks, rather than faintly trying all tasks. Therefore, we set out to build a solid base for our AUV accompanied by rigorous testing on all fronts: mechanical, electrical, and software. The Mechanical and Electrical Team focused on designing a modular and versatile AUV, which can be improved and adjusted to run at the competition in future years. This decision was made upon the discovery that competition tasks only varied slightly from year to year. We concluded that creating a vehicle that could be slightly adjusted to achieve the new tasks was the most prudent course of action. The Software Team focused on developing a generalized autonomy stack to move away from a per-task hardcoded stack. We wanted to make it easy for new members to contribute and also develop generalized autonomy algorithms that can be leveraged for multiple tasks (e.g. MPC controller). Our software team firmly believes that investing in a state-of-the-art simulator is one of the most rewarding investments to succeed in Robosub. A software team can do testing with much faster iterations in a simulator. We devoted significant efforts into developing a stable simulator, which later on helped us debug our localization, mapping, vision, and trajectory planning algorithms.

Keeping in mind how we nearly missed a perfect buoy run due to hard-coded qual gate code in our first year, we chose to focus on completing the first several tasks with a high degree of precision, completing the new vehicle, and securing a Top-10 rank. This meant accomplishing software and maneuverability based tasks, with a hopeful ambition to add manipulators later on in the team's schedule. We specifically focused on the gate, buoys, and path-following tasks. We designed and rigorously tested these scenarios in our simulator. Manipulator ideas and designs were at their infancy when COVID-19 shutdowns hit, completely stalling their development. We also tried experimenting with our upgraded sensor suite which included: 360 sonar, stereo cameras, and DVL.

2 Design Creativity

2.1 Mechanical Design Drivers

This year, we fully redesigned our mechanical system, looking to improve on the major limitations of our first-year (2018-2019) vehicle. We identified accessibility to internal electronics, modularity of vehicle internals, and thermal management of our internal electronics as three major areas of improvement. Namely, the need to accommodate a larger main computer (from the NVIDIA Jetson Tx2 to the NVIDIA Jetson Xavier), two sets of stereo cameras, a DVL (Doppler Velocity Log), Hydrophone acoustic system, and upgraded IMU (Internal Measurement Unit) necessitated a larger enclosure. We also wanted the flexibility to improve and adapt our perception and compute hardware as we advance as a team and employ more advanced algorithms in future years, so we intentionally oversized the electronics enclosure.

2.2 Enclosure System

The core of our AUV is a pressure vessel, consisting of two transparent acrylic tubes joined by a central aluminum "midcap." The tubes are joined to the midcap with temporary double o-ring seals, so they can be easily removed in a field environment for on-the-fly electronics servicing. The midcap is a large (7" diameter) 5-axis CNC machined part, which constituted a significant manufacturing challenge for our team. The midcap contains two o-ring tube seals on each end, and four face seals around the circumference of the part, which seal four removable panels containing cable pass-throughs. The enclosure is separated into two halves, one for battery and power electronics and one for compute and perception electronics. This year, we decided to move the battery from a separate sealed enclosure into the main enclosure for ease of wiring and to reduce possible points of failure in sealing and electrical connections.

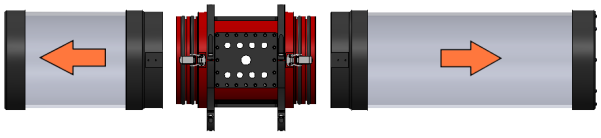

Figure 2: Kingfisher Enclosure System CAD Model

2.3 Structural System

The main driver for our redesigned structural system was the ability to easily service internal electronics, especially on-the-fly both at competition, and at our testing facilities in Pittsburgh. This drove us to a folding chassis design, where the superstructure of the AUV folds away, allowing access to electronics on 3 sides (left, right, and top). This design allows us to have a compact profile during operation and shipping, but also open access with minimal obstructions during electronics servicing. We hope this design allows us to quickly iterate and experiment with new compute and perception hardware. The main acrylic electronics enclosure is reinforced underneath with a steel weldment, termed the "spine." The spine is designed with maximum rigidity in mind, as it is the primary bracket on which vibration-sensitive perception hardware (IMU, Stereo Cameras, DVL) are mounted. Even small deflections of these sensors relative to one another can throw off the localization algorithms, so the spine weldment is as stiff as possible.

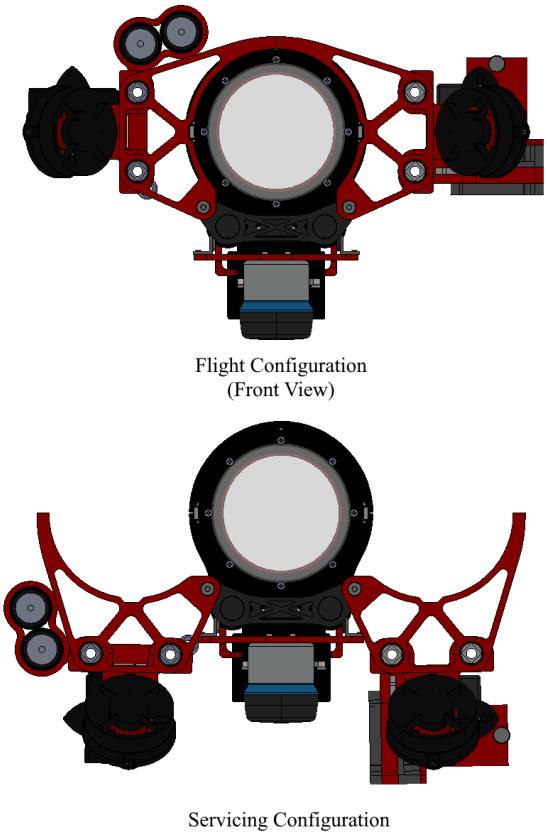

Figure 3: Kingfisher Structural System CAD Model

2.4 Electronics System

2.4.1 Compute

Kingfisher's electronics package is designed to support the complex needs of our software team, by providing the primary computer, communications systems, power and battery management, and sensing systems for the vehicle. The electrical design is optimized around simplicity, compute performance in defined areas like GPU and multithreading ability, I/O inter-

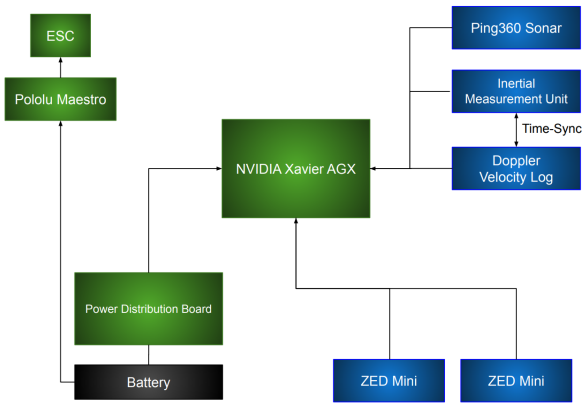

Figure 4: Kingfisher Electronics System Overview

face availability, and power/space requirements. To satisfy these goals, we settled on an NVIDIA Jetson AGX Xavier as our primary single-board-computer (SBC). This is an upgrade over the Tx2 unit we used last year, and the AUV had to be designed around fitting this larger computer. The upgrade was driven by a desire for improved CPU performance and I/O availability, allowing us to upgrade our cameras from a single monocular camera in Albatross to a set of two stereo camera systems this year (4 cameras total).

2.4.2 Power

The vehicle's power system is managed by our custom power distribution board (PDB), shown in Figure 4. The PDB supports up to 200A power draw, while measuring current draw and battery state-of-charge information. It also supports an external hardware kill-switch that can immediately disable all power to the thrusters based on either a TTL input from our magnetic kill switch, or a software pin controlled by the Xavier in the case of a critical software fault.

2.4.3 Sensing

We removed the off-the-shelf Pixhawk unit that we used last year with an Xsens MTi-200-VRU Inertial Measurement Unit (IMU), which provides significantly improved attitude and heading information to our custom Guidance, Navigation, and Control (GNC) system. The Xsens was chosen because of its proprietary software designed to improve resistance to unmodelled magnetic field distortion in the environment, which is notoriously challenging to deal with. This was a significant problem for us last year, so we spent extensive time testing and verifying Xsens IMU in magnetically noisy environments. The only alternative is high performance MEMS or Fiber Optic Gyroscopes (FOGs), although these are notoriously expensive and hard to integrate into the sensing system. The Xsens is also capable of operating in a large variety of time synchronization modes. Additionally, we purchased and have begun integrating a Blue Robotics Ping360 sonar.

The most significant upgrade to our electronics this year was the addition of a Teledyne Pathfinder Doppler Velocity Log (DVL). This sensor provides accurate linear velocity data which can be fused with inertial and visual data to provide a high degree of position sensing accuracy. To make this work, the DVL needs to be precisely synchronized with the IMU. We achieve this using the Xsens IMU's sync port, which allows us to output a regular sample signal to the DVL, and receive an accurate timestamp from the IMU indicating when the sample occurred with respect to the IMU's clock. We have not yet attempted synchronization between the Xavier and the IMU, although we intend to use either a direct Pulse-Per-Second (PPS) clock sync between the devices, or use the ETHZ clock bias estimator: Cuckoo Time Translator [1].

2.4.4 Communications

A significant upgrade this year was the transition from slimrun cat5 ethernet cable and Fischer connectors to Blue Robotics' Fathom Tether and Subconn connectors. This cable is neutrally buoyant, leak resistant, and more mechanically durable than the slimrun ethernet. Likewise, the wetmate connector is far more reliable in a marine environment. We also constructed a top-side box, with a router and connections for power and the fathom tether spool to provide wireless access to operator laptops. Internally, we use a combination of RS-232 and USB to communicate with sensors and actuators.

2.5 Software Systems

The focus for this year's software systems is creating flexible code for autonomy. We draw inspiration for our system design from well established autonomous systems such as self-driving cars and exploratory mobile robots. The software stack is split into State Estimation, Perception, and GNC (Guidance, Navigation and Control).

2.5.1 State Estimation

Underwater state estimation is known to be a challenging problem due to being in a GPS denied, low visibility, and dynamic environment. We use an IMU, DVL, and Depth Sensor in conjunction with an Extended Kalman Filter. However, in precision tasks, these sensors can still develop drift. We have conducted experiments in MATLAB involving the use of Pose Graph based SLAM methods [3], [4], [7] to provide a framework for loop closure and landmark-based localization.

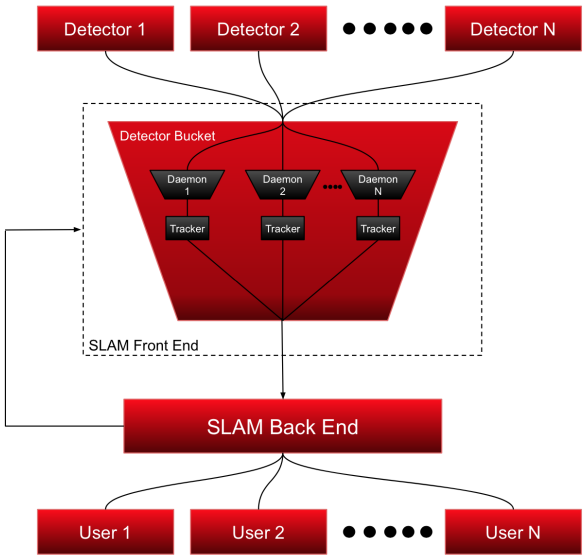

Figure 5: Kingfisher Perception System Overview

The Pose Graph minimizes an error function involving the constraint between nodes:

$$e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j)$$

The objective function we must minimize over the set of constraints C becomes:

$$F(x) = \sum_{(i,j)\in C} e_{ij}^T \Omega_{ij} e_{ij}$$

The system of equations is solved using a nonlinear Levenberg-Marquadt optimizer. A full pose graph implementation will be integrated before the coming season. Currently, we use a combination of robot pose and landmark Kalman Filters.

2.5.2 Perception

Regardless of what the specific challenge is, Kingfisher must be able to perceive the world and accurately gather information. The perception system is managed by the Detector Bucket (Figure 5). A detector will report a detection to the Bucket, which will spawn a Detector Daemon to filter, and track the data. This abstraction allows for asynchronous sensor information and minimizes the latency between detection and updating the robot's beliefs.

We use the Mahalanobis Distance and the Kuhn–Munkres (Hungarian) algorithm [5] to effectively match and reject detections. This is useful because the Mahalanobis Distance can incorporate a sensor's reported error uncertainty. Any efficient integer programming algorithm can be used, including graph-based methods such as Joint Compatibility Branch and Bound.

Our vision detection system is composed of Deep Neural Networks and classical Computer Vision detectors. For simple detection tasks such as the gate, we can use traditional line-fitting, which reduces overhead and complexity.

The default tracker for detectors is a constant position Kalman Filter Tracker in 3D . In the event of dynamic objects, the encapsulated tracker can be replaced by a higher order filter. Using a configurable object manifest, any prior information about objects (dimensions, location, quantity) are incorporated into the robot's beliefs. The end result is a 3D perception system that allows for accurate navigation (Figure 7).

2.5.3 Guidance, Navigation and Control

We use a hierarchical nonlinear model-based control strategy. At the low level, we use a differentially flat dynamics model with flat outputs corresponding to position and heading as well as their derivatives. A low-level PID attitude controller is used to stabilize the roll and pitch axes, while a custom linear Model Predictive Controller (MPC controller) with a 2-second horizon based on the OSQP quadratic programming toolbox is used to provide position and heading tracking control. A second order linear model based on the flat outputs is used, so the output of the Model Predictive Controller includes acceleration in x, y, z as well as yaw velocity. These are combined with the roll/pitch velocity inputs from the attitude controller and fed through our inverse dynamics model to determine a 6DOF force/torque wrench. Finally, the Thruster Allocation Matrix (TAM) is applied to the wrench to determine 8 thruster force commands which are each fed through the inverse thruster dynamics model.

High level guidance is provided using a Motion Utilies system, which provides easy-to-use trajectory primitives for mission applications. While a trajectory primitive is active, the Motion Utilities object acts as a reference trajectory server providing a 2 second horizon reference trajectory to the MPC controller. We have created a small toolbox of trajectory primitives, including S-Curve (finite jerk) linearly interpolated trajectories, minimum snap trajectories computed by a custom 5th order polynomial spline optimizer based on OSQP, simple ray trajectories for following a straight line indefinitely, and search area trajectories based on a polynomial bounding region within which to perform a grid search.

Quadratic Programming allows extremely fast control frequencies, easily satisfying our requirement of 20 Hz with a 2 second horizon time on our embedded computer. Details regarding our MPC and Trajectory optimization implementations can be found in its own paper [2]. We are currently working on adding parameter adaptation using EKF to allow our inverse dynamics controller to learn and update model parameters online to reduce model innacuracies.

2.6 Simulator

We realized early-on that having an offline testing framework is very important for the software team to iterate faster and develop a more robust software stack. Our software stack is explicitly designed such that the simulator is a drop-in replacement for our low level sensor drivers package, and all other software including controls, state estimation, and perception are tested in the simulator. CFD analysis is used as a starting point for model identification of the vehicle, allowing us to create an accurate digital twin of our AUV for simulations (Figure 6). Our simulator is based on the UUV Simulator [6] and with an added software layer to fine-tune for Robosub.

2.7 Frameworks

We have been working on a set of frameworks aimed to improve software reliability within our codebase. First, a fault tracking service was designed to allow ROS nodes to set and unset fault statuses in order to centralize information about system health. Some faults can be defaulted to active, and only cleared after an initialization and verification phase, for instance. This provides a quick way of verifying that all systems are running nominally before switching to autonomous modes or even arming the vehicle.

Finally, we use continuous integration to run automated build and test jobs to prevent regressions in development. These automated checks have been immensely useful as we have approach 300 commits and 30 finished pull requests on our codebase this year.

3 Experimental Results

We began testing new controls frameworks in the high bay of Carnegie Mellon's Robotics Institute early in the year. However, once the quarantine began, we transitioned to simulation (Figure 6). We created mock navigation and perception tasks in our simulator to test our software stack offline. Some of these tasks include navigating to a point of interest, identifying the gate and navigating through it safely, and testing our perception pipeline on a variety of new objects (Figure 7). We wanted to ensure that our system could handle higher level tasks that would always show up in Robosub competitions, regardless of the specific challenge for that year. We rely heavily on detailed visualizations and data analytics to gauge the performance of our systems (Figure 8).

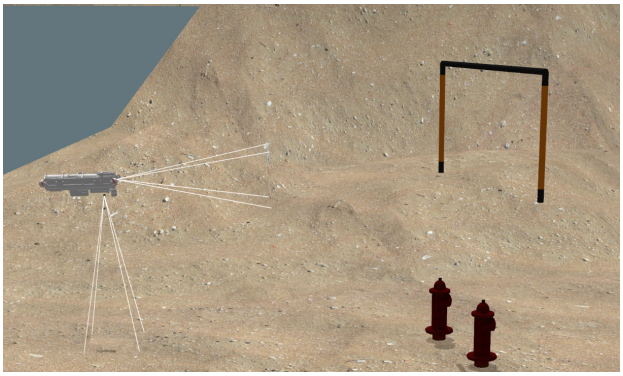

Figure 6: Kingfisher in a Simulator Environment

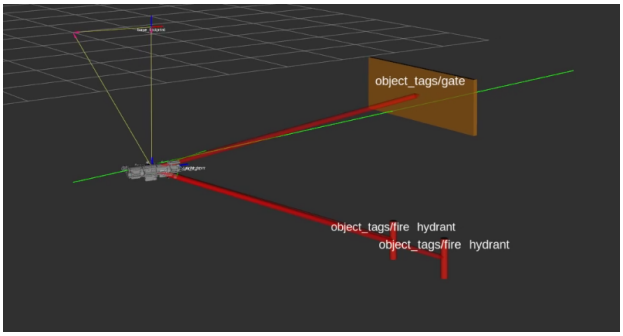

Figure 7: Kingfisher Internal Perception System

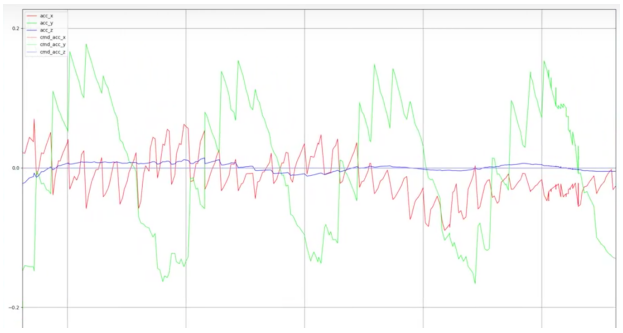

Figure 8: Kingfisher Control Input Visualization

4 Acknowledgements

TartanAUV would like to thank our parent organization, RoboClub, as well as faculty members Michael Kaess and George Kantor for their incredible mentorship and support. We would like to thank Chuck Whittaker and the Field Robotics Center for training and access to the Robotics Institute water testing facilities. We would like to thank Jim Bain (CMU ECE), Margaret Noel (CMU ECE), and Catherine Copetas (CMU SCS) for their crucial help in negotiating sponsorship deals. We would like to thank our gold sponsors: Leidos, Carnegie Mellon Student Senate, and Altium designer, our silver sponsors: Blue Origin, Fischer connectors, Blue Trail Engineering, and the Robotics Institute, and our bronze sponsors: NVIDIA, Robot Perception Lab, Advanced Circuits, Connect Tech inc, Molex, and IMC. We would also like to thank Vortex NTNU for their initial simulator setup and Harvey Mudd Robosub Team (MuddSub) for a high degree of collaboration. We look forward to everyone's continued support in our effort to bring Robosub to the Carnegie Mellon community.

References

[1] Cuckoo Time Translator, ETH Zurich. https://github.com/ethz-asl/cuckoo_time_translator.

[2] A.Shek and T. Scherlis. Highly Dynamic Quadcopter Control For Drone Racing. 2020. http://tomscherlis.com/wp-content/uploads/2020/05/drone_control_16899_final_report.pdf.

[3] F. Dellaert and M. Kaess. Square root SAM: Simultaneous location and mapping via square root information smoothing. 2006. https://www.cc.gatech.edu/~dellaert/pub/Dellaert06ijrr.pdf.

[4] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, Wolfram Burgard. A Tutorial on Graph-Based SLAM. 2018. http://www2.informatik.uni-freiburg.de/~stachnis/pdf/grisetti10titsmag.pdf.

[5] H.W. Kuhn. The Hungarian method for the assignment problem. 1955. https://web.eecs.umich.edu/~pettie/matching/Kuhn-hungarian-assignment.pdf.

[6] Musa Morena Marcusso Manhães, Sebastian A. Scherer, Martin Voss, Luiz Ricardo Douat, and Thomas Rauschenbach. UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation. In *OCEANS 2016 MTS/IEEE Monterey*. IEEE, Sep 2016. https://doi.org/10.1109%2Foceans.2016.7761080.

[7] S. Thrun and M. Montemerlo. The graph SLAM algorithm with applications to large-scale mapping of urban structures. 2006. http://robots.stanford.edu/papers/thrun.graphslam.pdf.

A Components List

Components	Vendor	Model/Type	Specs	Cost (if new)
Buoyancy Control				
Frame	Various	Aluminum		150
Waterproof Housing	(Generic)	6in Acrylic Tube		150
Waterproof Housing	Custom	Aluminum Midcap		Sponsored
Waterproof Connectors	Blue Trail Engineering	Cobalt Series		Sponsored
Waterproof Connectors	Subconn	DBH8F/DOM8M		300 per pair
Thrusters	Blue Robotics	8x T200		
Motor Controls	Blue Robotics	Generic ESC		
High Level Controls	custom			
Actuators	Blue Trail Engineering	Waterproof Servo		Sponsored
Propellers				
Battery	Various	4s LiPo 16Ah		Already Owned
Converter				
Regulator	Texas Instruments	Various		
CPU	Nvidia	Jetson Xavier		Sponsored
Internal Comm Network	RS232/USB			
External Comm Network	Ethernet			
Programming Language 1	Python			
Programming Language 2	C++			
Programming Language 3	MATLAB			
Compass	See IMU			
IMU	Xsens	MTi-200-VRU		Sponsored
DVL	Teledyne	Pathfinder		Sponsored (Leidos)
Camera(s)	Stereo Labs	2x ZED mini		
Hydrophones	Aquarian	AS-1		
Manipulators				
Algorithms: vision	Yolov3 Template Matching			
Algorithms: acoustics	FFT			
Algorithms: localization/mapping	EKF Pose Graphs			
Algorithms: Autonomy	Model Predictive Control (MPC)			
Open source software	OSQP OpenCV ROS Gazebo UUV-Simulator			
Team size (number of people)	7			
HW/SW expertise ratio	1:1			
Testing time: simulation	$>$ 100 hours			
Testing time: in-water	0 hours (Pandemic)			