TEAM SIMPLEXITY 2021 UUV

Ryan Meagher, Tyler Meagher, Alex Battikha, Hudson Kim, Thomas Fernandez

***Abstract* -- Team Simplexity is a first-year private team started by Ryan Meagher in San Diego, CA. The team consists of four middle schoolers and a graduate student from the ASU Fulton School of Engineering. We have created a UUV that utilizes cutting edge software and hardware. Our UUV incorporates the ModalAI VOXL companion computer combined with ModalAI's Flight Core. Two 4K high resolution cameras are at the heart of our vision processing & navigation system. The PX4 firmware running on the Flight Core controls the propulsion system and interfaces to several offboard sensors. ROS2 runs our vision processing pipeline on the VOXL allowing OpenCV to process raw images, and in turn, publish commands to our navigational system. We have a propulsion system which utilizes the BLHeli32 firmware on two 4-in-1 ESCs. These are connected to six Blue Robotics T200 thrusters. We built several custom test apparatuses to characterize several aspects of the UUV to ensure we have a solid foundation to build upon for years to come.**

Competition Strategy

As a first year team our initial goal was to build a robot with the capability to reliably navigate through the gate after the random coin flip. We planned to perform a style maneuver before passing through the gate. A significant amount of engineering work had to be performed before even these simple tasks could be achieved. We realized having a strong understanding of each component in the vehicle would be key in achieving a successful outcome. This involved research, prototyping, and testing. We also sought advice from outside experts. Our strategy was to look at past teams designs to understand what had been successful, but look for areas where we might innovate and gain a competitive advantage.

By leveraging proven off the shelf components we could reduce the engineering time it took to get a functional UUV ready for the competition. A major portion of our effort was learning about, and designing, a propulsion system that could complete our selected missions effectively. We also wanted to provide a strong foundation to build upon for future competitions.

One early competition strategy we decided upon was to design a propulsion system that could support two complete runs in the twenty minute time period. This put specific speed requirements on the design and influenced our hull, propulsion, and battery design. Doing this allowed us to double our chances if something went wrong during our initial run.

Image recognition and underwater navigation are large parts of the competition requirements to complete missions successfully. We decided that using cutting edge image sensors as well as having the processing power to utilize the high bit-rate video streams would be an important strategy. For this we turned to a local drone manufacturer, *ModalAI*, who develops cutting edge hardware and software for aerial drones. At the heart of their solution is Qualcomm's high performance Snapdragon processor with "Quad-core cpu up to 2.15GHz, a GPU, and 2 DSPs" along with custom hardware acceleration blocks[1]. This solution provides a world class imaging processing platform ideally suited for autonomous vehicles.

Our competition strategy for underwater navigation was to use multiple IMUs, *PNI's* RM3100, a military grade magnetometer [2], along with vision localization to navigate successfully through the different missions.

With our vision system driving our localization, we determined having three degrees of freedom in the propulsion system would be necessary, as well as sufficient, to perform the missions in this competition. As the robot approaches the gate, a path marker, or the Make your Grade buoy, having the ability to make slight corrections in the horizontal plane without rotation would allow the camera to better track

images. This made image navigation significantly easier and more reliable.

The final strategy we used for the competition was to design a UUV that could spin about the Z axis without movement in its (x,y,z) position. This would allow greater positional accuracy after completing the "Coin Flip" task and the "Style Maneuver" before continuing to the next task.

Design Creativity

Mechanical and CAD

This year we leveraged an off-the-shelf enclosure and frame structure from *Blue Robotics* to speed the development process. After working through the mission requirements, we determined a six thruster propulsion design would be the best way to meet the 3 DOF motion requirement as well as the axial spin requirement. Figure 1 shows the general thruster placement and propeller rotations.

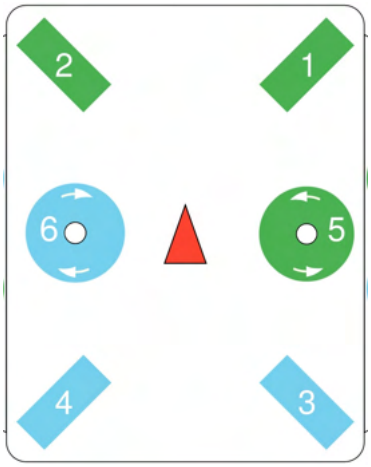

Figure 1: Thruster orientation and propeller rotation direction

Using this configuration allows the UUV to move in any horizontal x-y plane direction (including strafing) without any rotation. It also allows for rotation about the Z axis with no movement in the (x,y,z) position.

We used Solidworks to create a 3D CAD model of our design. Figure 2 is a rendering of our design.

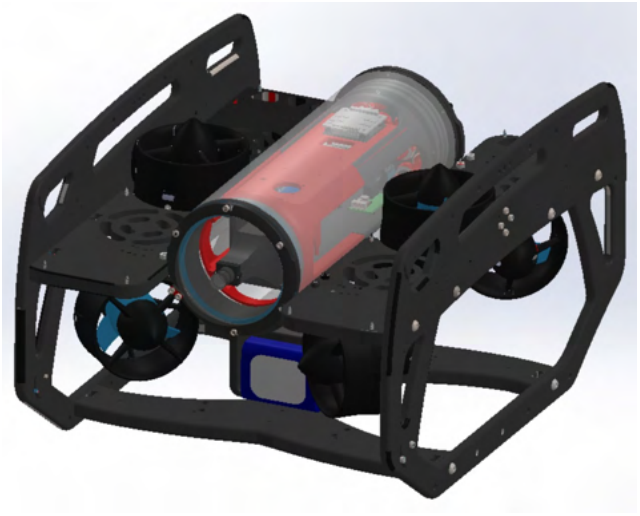

Figure 2: 3D CAD model of our custom UUV

We utilized a small waterproof cylindrical enclosure from *Blue Robotics* with a 100 mm diameter x 285 mm length with a half dome attached to one end of the enclosure that extends 73.5 mm at its apex. Two pressed fit o-rings make a watertight seal with the cylinder. To maximize the space that was available for our electronics, we prototyped several methods of mounting our electronics using 3D printed parts. This was an iterative process throughout the season that culminated with a design that was modular and offered solid usability. Figure 3 is an exploded view of that design.

Figure 3: Exploded view of our custom 3D printed enclosure mounting system

To create the foundation of our enclosure we designed a symmetrical cylindrical piece that ran the length of the enclosure. This cylindrical piece had slots in it that were used to slide flat panels with hardware components attached to both sides of the panel. In order for this to work we had to optimally position our hardware for ease of use when plugging in ethernet, power, or other connectors.

This took several iterations but we ended with a design that met all the requirements. This included ease of assembly and disassembly, and allowing for the plug and play mentality. We still have to do thermal analysis to ensure our cooling fan is sufficient to keep the temperatures near the plastic parts in a range where they will not deform. We also need to measure how mechanical vibrations caused by the fan will affect our IMU measurements.

The slots in the cylindrical piece were also used as the base for our camera mount. We designed the camera mount in such a way that the camera points in a forward direction perfectly centered with regards to the enclosure. This allowed for the positioning of our forward-facing high resolution camera to be only 1 mm away from the apex of the half dome. In doing this we could avoid problems related to the curvature of the dome with respect to camera orientation so as to avoid the issue where a "flat object normal to the optical axis cannot be brought properly into focus on a flat image plane"[6].

Doing this increased the quality of the raw images being fed into our vision processing pipeline. In addition to this, the camera mount was designed in such a way that a second high resolution camera could be pointed downward on the same y-axis as our forward facing camera. Figure 4 is a picture of our assembled electronics enclosure.

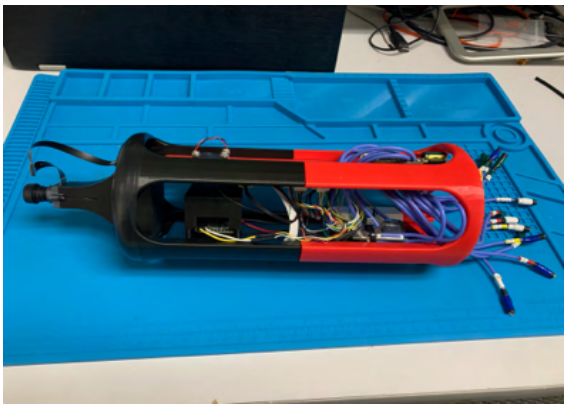

Figure 4: Electronics in custom 3D printed enclosure

An important feature of any waterproof enclosure is getting electrical wires through the enclosure without compromising the integrity of the watertight seal. Our feedthroughs include the six, three phase wires to our thrusters, a depth sensor, two battery connectors, and ethernet cable for communication with a ground station during testing. To save on cost we built several of our own custom feedthroughs. This included two battery power feedthroughs, and an ethernet feedthrough for the tether.

To accomplish this we used hollow aluminum o-ring feedthroughs from *Blue Robotics*. We then created a barrier in any wire passing through that feedthrough with a solid solder joint. Then followed by an epoxy potting of the wire in the feedthrough. This ensured that a nick in the shielding of the wire would not create a path for water to travel into our enclosure. We used a vacuum tester to verify our feedthroughs were watertight. Figure 5 is a picture of the process of making a homemade feedthrough.

Figure 5: Custom feedthroughs

Electronics

Figure 6 is a wiring diagram of our electronics.

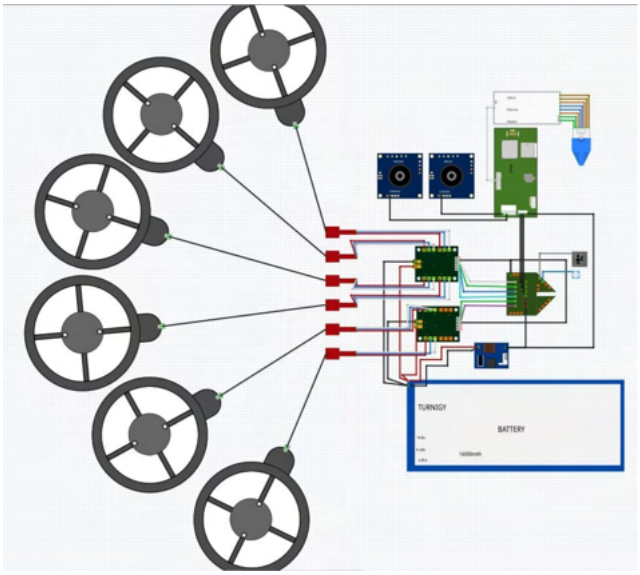

Figure 6: UUV wiring diagram

At the heart of our electronics is *ModalAI's* VOXL platform. This is a high performance companion computer, described earlier, that is in control of the autonomous operation of the UUV. It communicates with the ground station over a 1GBit ethernet link, when tethered, through a USB to ethernet adapter. It interfaces to two 4K, 30 fps, high resolution cameras over high-speed MIPI interfaces for image processing and localization. It also interfaces with the Flight Core over two 1Mbps RS-232 interfaces. The VOXL is powered by an offboard 5V power regulator that is connected to the 16Ah, 4S LiPo battery. A fan is used to cool the VOXL's Snapdragon CPU.

The *ModalAI* Flight Core has a *STMicroelectronics* "high-performance Arm Cortex-M7 32-bit RISC core operating at up to 216 MHz frequency" [2]. This MPU runs the PX4 firmware and interfaces to the two 4-in-1 ESC, the MS5837 depth sensor, 3 IMUs, *PNI's* RM3100 Magnetometer, and the safety switch. The *Furling32* 4-in-1 ESC controls up to four *Blue Robotics* T200 thrusters. The ESC supports 45 amps of current on each channel at up to 20V.

Software

Figure 7 is a diagram of the different software running on our UUV

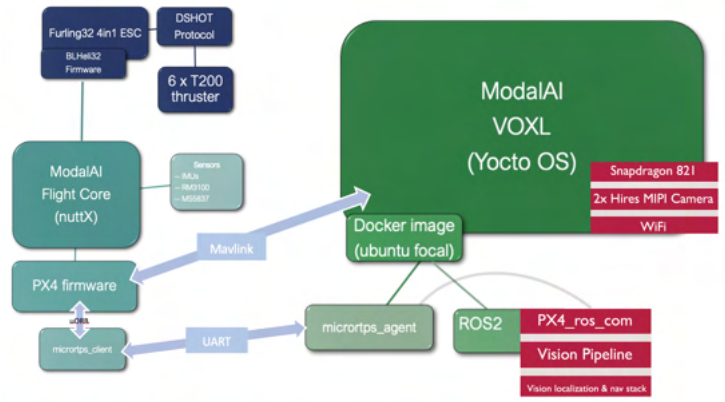

Figure 7: UUV software diagram

The software is divided over several different processors. The VOXL, the Flight Core, and the ESC all have independent software stacks running on them. In addition, the VOXL runs multiple software stacks on its multiple ARM cores, and independent firmware on its two DSPs. The following sections describe these in more detail.

The microprocessor on this ESC runs the BLHeli32 firmware. This firmware allows setting the rotation of the motors, supports bidirectional operation, as well as many other parameters. The maximum update rate of the motor speed is also controlled by the firmware's communication protocol. The Flight Core communicates with the ESC over a 1 wire interface per thruster. This ESC supports the DSHOT protocol, a digital protocol, and unlike the traditional PWM protocol, allows us to query telemetry data from the ESC (like temperature, voltage, current and RPMs). The update rate of the ESC determines the maximum frequency of any control loop used in the propulsion system. Anything faster than this frequency will have no effect. We are still experimenting to determine the appropriate update rate for our propulsion system.

The Flight Core runs the PX4 firmware. The PX4 firmware is an open source professional drone autopilot software project. In order to support our UUV, we needed to customize the PX4 firmware. This included creating custom airframe and mixer files to deal with the unique propulsion system on our vehicle. We also made a custom attitude module to independently control our vehicle based on the sensor data it

receives. A large part of this attitude control came from creating an accurate MS5837 driver and utilizing the compass readings from the *PNI* RM3100.

The PX4 firmware with custom modifications was flashed onto the *ModalAI* Flight Core. The Flight Core runs the PX4 firmware on top of the nuttX RTOS. The drivers in the PX4 software identified sensors connected via SPI, I2C & UART. The PX4 system uses a uORB publish/subscribe messaging system. These published messages were turned into mavlink messages via the PX4 mavlink module and sent via UART to our VOXL companion computer. These mavlink messages are then read and forwarded to a Docker container, and when connected via ethernet, they are also forwarded to the ground station.

The VOXL is running Yocto Jethro Linux. It provides us the ability to use Docker to run any supported containers (e.g. any OSes that support armv8). We choose to create a Focal Docker image (Ubuntu 20.04). This gave us access to ROS2, in particular the Foxy distro of ROS2 was used on the VOXL. Having access to ROS2 gave us the ability to leverage the px4_ros_com ROS2 package. This was ideal as "the Fast DDS interface in the PX4 Autopilot can be leveraged by any applications running and linked in DDS domains" which is why PX4 developers recommend switching to ROS2 [7].

To enable communication between our px4_ros_com package and the PX4 uORB messaging schema on the Flight Core a bridge had to be created. This is done by a server/client interaction via the UART interface between the micrortps_client on the Flight Core and the micrortps_agent running in the Ubuntu docker container on the VOXL.

However, all the serial ports on the VOXL are internally mapped to the Sensors DSP (sDSP). This enables low-level and time-sensitive interaction to sensors and telemetry communications via the sDSP's real-time operating system [8]. While this may free up CPU cycles on the applications processor it does not allow applications to talk directly to the serial ports with reads and writes which is required by the micrortps ROS2 bridge.

Therefore to enable this bridge we had to create custom software on Qualcomm's Hexagon sDSP as well as services in the linux kernel that would parse UART messages and forward these messages via UDP to the micrortps_agent and vice versa parsing UDP packets from the micrortps_agent and creating UART messages to the micrortps_client.

Even though the Docker container is running on-top of the host OS, we are able to have access to the host network as well as camera devices connected to the VOXL. This enabled us to make use of camera streams in our ROS2 vision pipeline which is directly used in the nav2 stack to control our vehicle via the micrortps_bridge.

We created a custom autonomous run process within ROS2. We trigger this via WiFi to start the mission. We have not completed this part of the design, but will be working on it over the coming weeks.

Experimental Results

To characterize the T200 thrusters running on our UUV we made a thruster tester apparatus. This involved transferring the force from the thruster to an S-type load cell through a pivot arm. The load cell distance from the pivot point was equal to that of the thrusters. This resulted in compression and tension forces applied to the load cell that were directly proportional to the force created by our T200 thrusters. In addition, a high precision 20V, 50 amp, variable DC power supply was used to characterize the T200's power usage at different operating voltages and PWM values. For the electronics, an ESP32 microcontroller was connected to a HX711 load cell amplifier to measure the force imparted on the load cell by the thrusters. The ESP32 was also connected to a pair of ESCs to control the speed of the thrusters by changing the PWM values. A webserver running on the ESP32, communicated over WiFi to a host computer which provided the user interface. It also provided the ability to store the experimental results. Figure 8 is a picture of the apparatus.

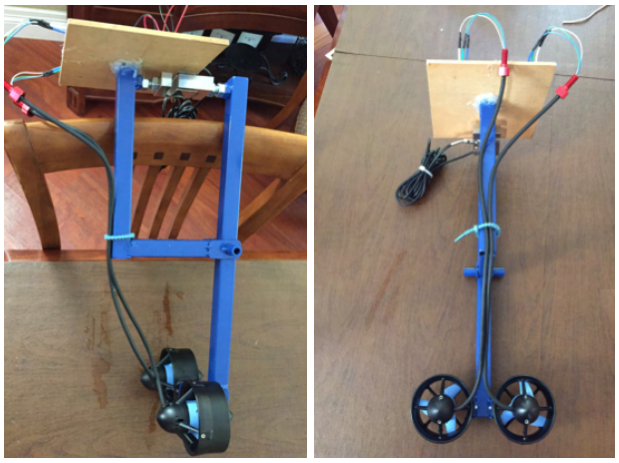

Figure 8: Thruster test apparatus

The user interface allowed for running manually as well as running in an automatic mode. When in the automatic mode, our microcontroller would send pwm signals from 1500 to 1900 and 1500 to 1100 microseconds in steps of +/- 25 microseconds to the ESC. Measurements were taken at each step and the mean force, current, and standard deviations were recorded every 5 seconds. These measurements were written to a csv file on the host computer for later analysis. We ran the experiment over several different voltages. We processed the results to come up with a characterization graph of the T200 thrusters as shown in Figure 9. This is a plot of the mean force, in pounds of thrust, vs the PWM signal length, in microseconds. Each graph represents a different operating voltage.

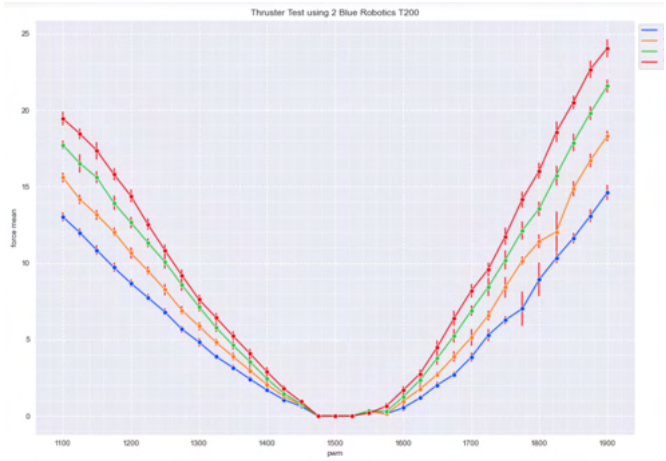

Figure 9: Blue Robotics T200 thruster characterization

We drew two conclusions from this testing. First, we determined a 16Ah, 4S, LiPo battery would provide the required power and energy storage for our UUV to complete 2 runs in a 20 minute time span. Second, we determined that our four horizontal thruster configuration would have greater than 25lbs of static force at max power. We plan to verify this on our actual UUV in the coming weeks.

For getting accurate depth measurements we used TE Connectivity's MS5837 30 Bar pressure sensor . The MS5837-30Ba is "a new generation of high resolution pressure sensors with I2C bus interface for depth measurement systems with a water depth resolution of 2 mm"[3]. To test our depth sensor we prototyped an apparatus that made use of an ESP32, the MS5837, and a small pressure vessel. This apparatus allowed us to test the software, and develop accurate drivers needed for the PX4 firmware. The small pressure vessel allowed simulating different depths without going in the water. Figure 10 shows our depth sensor prototype and our pressure test vessel.

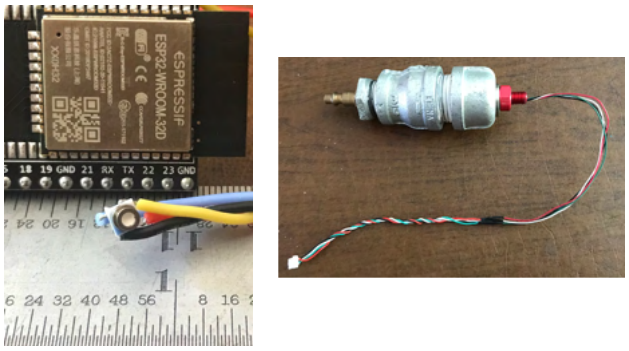

Figure 10: Our depth sensor prototype and pressure testing vessel

We also developed an apparatus and method to measure our center of gravity (CG) and center of buoyancy (CB). Using a scale, the internal 3D compass, and our CAD models we can experimentally find the CG and CB locations. Measuring the force and heading from multiple points both in, and out, of the water allows us to determine these locations with the help of our CAD model. Our two PVC ballast tubes and our battery holder allow for adjustments in the CG and CB. This is important so we can maintain

symmetry of our propulsion system. This task is ongoing, and we plan to make more measurements over the coming weeks. Figure 11 are pictures of this testing.

Figure 11: Center of gravity and center of buoyancy testing

We have just started testing our propulsion system in the water in a tethered configuration. We plan to continue this work in the coming weeks as well as get our initial testing on our vision localization system. In the future we would like to get a software simulation environment working for out of the water navigation testing. Figure 12 is our completed UUV.

Figure 12: Team Simplexity's UUV

Acknowledgements

We would like to thank our mentors and parents for all their support. Without their help and encouragement we would not have been able to undertake such a major project. We would also like to thank ModalAI, Blue Robotics, Bulgin, and Beto at MacArtney for their generous support.

References

1. ModalAI. *VOXL.* https://www.modalai.com/pages/voxl (ModalAI, 2021).
2. PNI. *RM3100 Breakout Board.* https://www.pnicorp.com/product/rm3100-breakout-board/ (PNI 2018).
3. ModalAI. *Flight Core - PX4 Drone Flight Controller.* www.modalai.com/products/flight-core (ModalAI, 2021).
4. ModalAI.*4k High-resolution Sensor for VOXL (IMX377 M12-style Lens).* www.modalai.com/collections/accessories/products/4k-high-resolution-sensor-for-voxl-imx377-m12 (ModalAI, 2021).
5. TE Connectivity. *MS5837-30BA spec sheet.* (TE Connectivity, 2019).
6. Riedl, Max J. *Optical Design Fundamentals for Infrared Systems*. SPIE Press. pp. 40 (2001).
7. PX4. *ROS 2 User Guide (PX4-ROS 2 Bridge).* docs.px4.io/master/en/ros/ros2_comm.html (ModalAI, 2021)
8. ModalAI. *VOXL Serial IO.* docs.modalai.com/voxl-serial-io/ (ModalAI, 2021).

Appendix A: Component Specifications

Component	Vendor	Model/Type	Specs	Number	Cost	Status
Buoyancy Control	Home Depot	2" schedule 40 PVC	280 PSI	2	8	Installed
Frame	Blue Robotics	BlueROV2 frame	Black HDPE	1	339	Installed
Waterproof Housing	Blue Robotics	4" diameter acrylic with end caps	Acrylic	1	224	Installed
Waterproof Connectors	Custom Made / Blue Robotics	Penatrator	Aluminum + Epoxy	10	5	Installed
Thrusters	Blue Robotics	T200		6	179	Installed
Motor Control	Furling32	45A 4-in-1	45A / 6S/ 4 Ch	2	50	Installed
High Level Control						
Actuators						
Propellers	Blue Robotics	Included with T200		6	0	Installed
Battery	Turnigy	16Ah, 4S Lipo	16Ah, 4S Lipo	1	145	Installed
Converter						
Regulator	ModalAI	Power Module	5V / 6A	1	50	Installed
CPU	ModalAI	VOXL		1	499	Installed
CPU	ModalAI	Flight Core	PX4`	1	249	Installed
Internal Comm Network	Ethernet, I2C, SPI, RS-232, USB					Installed
External Comm Interface	1Gbit Ethernet, WiFi					
Compass	PNI	RM3100		1	25	Installed
Depth Sensor	TE Connectivity	MS5837	30 Bar	1	9	Installed
IMU	Multiple	Included on VOXL and Flight Core		5	0	Installed
DVL						
Vision	Sony	IMX214	4k30fps	2	$49	Installed
Acoustics						
Manipulator						
Algorithms: vision	OpenCV					Installed
Algorithms: acoustics						

Algorithms: localization and mapping	SLAM					Planned
Algorithms: autonomy						
Open Source Software	PX4, NuttX, ROS2, Yocto, Ubuntu, , ROS2, OpenCV					Installed
Team Size	5					
Expertise ratio	3 Hardware / 3 Software					
Testing Time: Simulation						
Testing Time: in water						
Inter-Vehicle Communication						
Programming Languages	C, C++, Python, Shell Scripting, Matlab, HTML, JavaScript					