Technical Design Report

Vivek Anand, Eliot Berry, Jacob Bongaarts, Zoe Dearing, Christopher Hill, Hayden Hirsch, Daniel Marchione, John McGhee, Adam Jirka, Eli Ng, Kai Paio, Micah Paio, Eloise Pelham, Kyan Schmidt, Zachary Shaw, Bryce Wilcock

Abstract **– The Gibson Ek Circuit Trees are excited to launch our first underwater robot at robosub 2024. We built our robot with an off-the-shelf BlueROV2 heavy configuration kit, later named Sir Swims-a-Lot, and modified it to run autonomously. Our team had little experience working with underwater robotics, so we planned for minimal modifications to the off-the-shelf model.**

Competition goals

2024 marks the first year of our team's participation in RoboSub. Our team is inexperienced in the field of underwater robotics so it was a learning process. We started later in the year than we would have hoped, so there was limited time. Taking these considerations, we opted to use an off the shelf BlueROV2 heavy configuration as opposed to building from scratch to give us the greatest chance of a competition ready submersible. We centered our strategy around consistency in performance of the simpler tasks.

We made minimal modifications to our robot with the goal for it to be testing ready as soon as possible. Minimal sensors and mechanism modification would also allow us time to make sure our submersible would be able to consistently complete our target tasks.

In the process of defining our competition strategy we identified a few key tasks which we felt Sir Swims-a-Lot would be able to complete given our limited time and resources. The tasks we identified were completable with the baseBlueROV2 heavy configuration without major changes to hardware and are as follows: enter the pacific with style spins, and circumnavigation of the hydrothermal buoy.

Design Strategies

Because of our limited experience with underwater robotics. Buying the robot gave us more room to think about competition specific needs, like modifications to make the robot autonomous, competition task preparation, and programming. While the BlueROV2 was not optimized for the competition it gave us a strong and reliable foundation to allow us to learn and work off of.

Fig 1. AUV model.

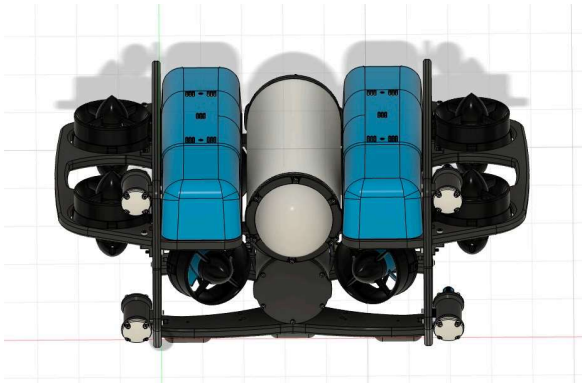

Fig 2. AUV model.

Building the robot

We assembled the BlueROV2 kit, following the build instructions. During the process, we had a problem with screws breaking in the fairings, so we 3D printed two new fairings to ensure a secure attachment.
One problem with the purchase of the off the shelf submarine, was that it didn't have a kill switch. Blue Robotics sells a switch, but only for small currents. After a lot of research we found a kit which comes with a MOSFET PCB, the Blue Robotics switch, and convenient mounting inside the enclosure, which intercepts the power from the battery to the submarine system. The MOSFET allows the switch to handle the larger current. This would minimize complexity, and meet the requirements for Sir Swims-a-Lot to be competition ready.

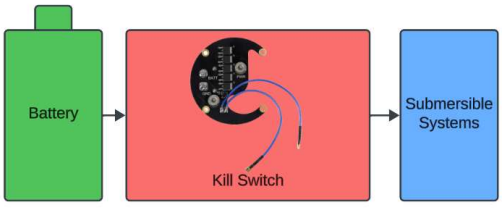

Fig 3. Kill switch design.

Apart from the kill switch modification and 3D printing replacement fairings, we didn't make any major changes to the base submarine.

Sensors

Sensors were a major point of uncertainty when we first started researching the construction of an AUV. One of the main questions we had was, "How do sensors work underwater?" We began testing various sensor types which would work underwater.

Through testing, we decided on the ultrasonic matrix as a potential sensing method because it would give the submarine the most accurate and consistent results.

Since we are now using the BlueROV2, which comes with sensors, we don't have a need for an ultrasonic matrix anymore. We learned a lot about how sensors work, so even though we aren't using the physical prototype it was still a valuable learning experience.

Coding

Our coding team started the year by researching which languages and libraries we would be using for our AUV. We considered using ROS2, as one of our coaches had experience with it, and its broad functionality, such as being able to connect multiple different scripts in different languages together in an efficient and clean way. The flexibility made the program much more complex than we were prepared for

this year, and there were simpler ways to program the BlueROV2. For these reasons, we switched our focus to BlueOS and its Navigator library, a convenient program for our BlueROV2. BlueOS had more user-friendly documentation and pre-built APIs that would integrate python code nicely with our submersible right out of the box.

Our next step after deciding on the libraries and language was motor control and navigation. The BlueOS Navigator library and its documentation made this process very streamlined and we were able to interface with the submarine's systems without much difficulty. We created a program to read and process input data from the submarine to simplify our navigation code.

Testing

Our goal for testing the robot was to complete and test code as much as possible, so that we could make changes and improvements rapidly. Preliminary checks were completed prior to placing the submersible in the water in order to minimize the chance of it sustaining damages. These checks included: safety consideration review, battery, pressure, network, and connection tests prior to placing the submersible underwater.
We ran multiple tests in local pools to experiment with code and confirm motor configuration. During our first test we discovered our electronics and battery enclosures were not properly sealing, so we were unable to get the submersible in the water. A week later we completed our first tests.

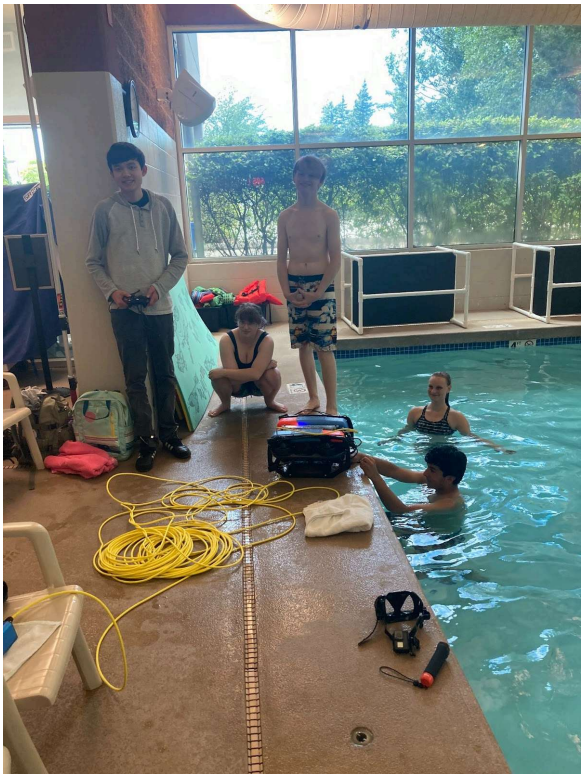

Fig 4. Pool testing with the team

We completed additional tethered pool tests to ensure the motors were properly configured. Once we understood how our motor control code interacted with the robot while it was in the water we were able to progressively test more complex code.

Throughout our experimentation and testing we learned the software provided and built into our robot. We learned how to use QGroundControl to control the robot with a joystick and how the robot moves in the water. The BlueOS Portal allowed us to test motors and sensors directly, interact with the file system, shell and network enabling us to learn to operate the robot in tethered and untethered configurations, upload and

execute new code and operate the robot safely & autonomously.

Component	**Vendor**	**Custom/Purchased**	**Cost**	**Year**
BlueROV2	Blue Robotics	Purchased	$4250	2024
BlueROV2 Heavy Configuration Retrofit Kit	Blue Robotics	Purchased	$790	2024
Fathom ROV Tether (ROV-ready)	Blue Robotics	Purchased	$400	2024
Lumen Subsea Light for ROV/AUV	Blue Robotics	Purchased	$160	2024
Xbox Wireless Controller	Blue Robotics	Purchased	$65	2024
BlueROV2 Spares Kit	Blue Robotics	Purchased	$350	2024
Subsea Buoyancy Foam: R-3312 (16 in x 8 in x 1 in) x2	Blue Robotics	Purchased	$140	2024
Subsea Buoyancy Foam: R-3312 (24 in x 8 in x 2.5 in) x2	Blue Robotics	Purchased	$320	2024
Newton Subsea Gripper	Blue Robotics	Purchased	$640	2024
H6 PRO Lithium Battery Charger	Blue Robotics	Purchased	$185	2024
Lithium-ion Battery (14.8v, 18Ah)	Blue Robotics	Purchased	$380	2024
BlueROV2 MOSFET Battery Switch	FieldWerx	Purchased	$125	2024

[1]A. Amer, O. Álvarez-Tuñón, H. Ugurlu, J. Le Fevre Sejersen, Y. Brodskiy, and E. Kayacan, "UNav-Sim: A Visually Realistic Underwater Robotics Simulator and Synthetic Data-generation Framework." Accessed: Jun. 28, 2024. [Online]. Available: https://arxiv.org/pdf/2310.11927

[2]"Docs | Home," *blueos.cloud*. https://blueos.cloud/docs/ (accessed Jun. 28, 2024).

[3]"ROS 2 Documentation — ROS 2 Documentation: Rolling documentation," *docs.ros.org*. https://docs.ros.org/en/rolling/index.html

[4]"BlueROV2 Assembly," *Blue Robotics*. https://bluerobotics.com/learn/bluerov2-assembly/