Open-hardware open-source low-cost underwater ROV

Vid Rijavec, Danijel Skočaj

University of Ljubljana, Faculty of Computer and Information Science
E-mail: vid.rijavec@gmail.com, danijel.skocaj@fri.uni-lj.si

Abstract

As monitoring coastal regions becomes ever more topical due to unforeseen changes in ecosystems there still has not been a major breakthrough in marine robotics when it comes to mass accessibility. In this work we present our low-cost prototype of an underwater remotely operated vehicle (ROV) for coastal exploration and monitoring. We present the mechanical, electrical and software components of the developed ROV and describe the results of field tests. We made all the data, schematics, software, and documentation needed for construction of a such low-cost vessel publicly available.

Keywords

ROV, submarine, low-cost, Raspberry Pi

1 Introduction

Robotics has been making major strides in worldwide adoption, especially in unmanned ground and aerial vehicles (UGVs and UAVs) where prices have been dropping and with household robots entering mass production. The same cannot be said regarding autonomous underwater vehicles (AUVs) and remotely operated underwater vehicles (ROVs). The potential for such vessels includes controlling and monitoring large underwater areas without a large crew of divers and by users without diving certification. A number of species are already on the decline due to climate change [1], and those are being replaced by others, affecting ecosystems in unpredictable ways. As such, having access to more effective and low-cost AUVs and ROVs could help in detecting changes sooner, allowing for faster response and potential food chain collapse prevention. Similar applications involving monitoring coastal regions by utilising such an underwater robot are numerous.

There are many commercial options for buying a research ROV. High-end versions include the iBubble EVO [2] or the Blue Robotics BlueROV2 [3], with projects like the Blue Dot Model C [4] presenting a more affordable but less capable option. Given that even the low-cost options still aren't truly affordable most enthusiasts still prefer to construct their own do-it-yourself (DIY) research vessel to customize it to their needs. However, most of these attempts tend to still be bulky and analog.

Our vessel, shown in Fig. 1, is designed as a combination of the classical AUV torpedo-like design and the typical boxy ROV designed for high movement accuracy, by combining the hydrodynamic hull shape with multiple control thrusters to allow for more degrees of freedom than a singular rear propeller. Our intention was to construct a fast-moving ROV that can explore a coastline from shore, one that's controlled by a pilot using a tablet computer but offers advanced stabilization functions and an autopilot for ease of use. As such our design is a compromise between the two underwater vehicle types, giving the vessel greater speed at the cost of some maneuverability.

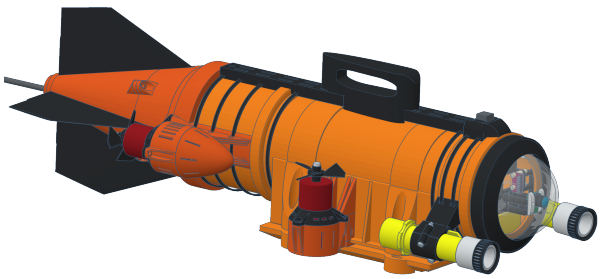

Figure 1: **ROV CAD Model.** The visible components of the vessel are: a PVC pressure hull, external LED lights, modified bilge pump thrusters (allowing movement in 4-DoF), rear stabilizers, a top handle, and a transparent camera dome.

The paper outlines the system and the specific components we've developed for it, with the rest of the code, schematics and documentation being available on Github [5].

2 Mechanical design

Figures 2, 3 and 4 present an overview of the ROV and its component parts, along with the tablet computer used by the pilot. The entire vessel is approximately 74 cm long and weighs just over 10 kg with batteries.

Building a sturdy and resistant pressure hull requires a rigid and non-permeable material that can be effectively glued to other parts. Out of all the tested low-cost components, the most cost-effective parts were determined to be PVC-U sewer pipe sections. We've used a cleaning section of pipe to make up the majority of the hull, as it has

an external hatch which is practical for accessing internal electronics.

The front of the vessel is capped with an acrylic dome with a thickness of 3 mm, to allow the internal camera to see the underwater environment. It's attached to the sewer pipe hull using epoxy glue and silicone caulking. The curvature of the dome later proved to be detrimental to the video quality as it added unwanted distortions and reflections. A more optimal design would involve the use of a thicker but flat (but smaller) front window. The video is then processed and sent over a length of rugged CAT 5E cable.

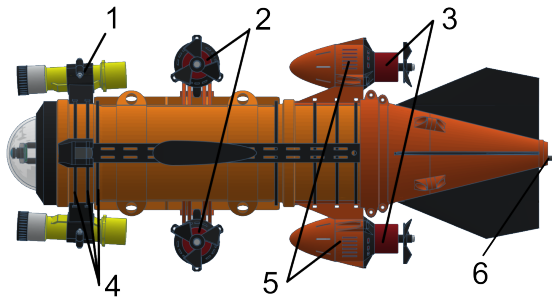

Figure 2: **Top view of the vessel. 1.** PLA flashlight mount. **2.** Depth control thrusters. **3.** Horizontal movement thrusters. **4.** Zip ties. **5.** PLA thruster nacelles. **6.** CAT 5E cable.

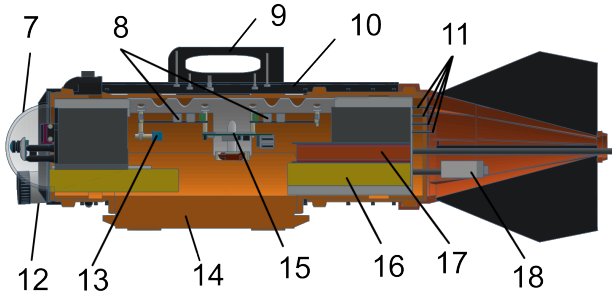

Figure 3: **Internal cutaway of the vessel. 7.** Transparent dome containing the Pi camera and IMU. **8.** Cytron MDD10A motor drivers. **9.** Dual purpose top fin and handle. **10.** Steel structural rail. **11.** Motor wires. **12.** Diving flashlights with diffusers. **13.** Voltage divider and ADC. **14.** Access hatch. **15.** Single board computer. **16.** 2x 5 Ah 3S LiPo batteries. **17.** 5 V 10 Ah Power bank. **18.** Water pressure sensor.

Submersible DC bilge pump motors intended for clearing out any water that seeps into a ship's bilge are a common choice for DIY ROV construction. The main appeals are the low cost, ease of modification and full water ingress protection. A simple improvement is to remove the pump's impeller and housing, replacing them with a more efficient propeller [6] to achieve higher (and reversible) thrust. Our prototype uses nylon boat propellers sized at 56 mm. These motors do have a drawback - the o-ring seal around the shaft which exerts increasing amounts of drag with increasing pressure [7]. The seals fully block the shaft from rotating at around 25 m, which limits the sub's operational depth. As such, using fully flooded brushless motors may be a better fit if greater depths are needed.

Rear wings aid with stabilization when moving at speed and were also chosen to be constructed out of PVC due to its light weight. The tail itself is not pressurized (typically called an outer or light hull which allows for water ingress) and houses the water pressure sensor, the tail fins, and protects the power/data cables along with providing better hydrodynamic performance.

3 Hardware and electronics

The electronics system as seen on the cutaway in Fig. 3 is centered around a single board computer, specifically the Raspberry Pi 2 Model B+, due to its low cost and lower TDP than later versions. The key features were GPIO pin access and the RJ45 socket which allows for an Ethernet link between the vessel and the control app on the surface. The surface component is a low-cost Lenovo tablet computer encased in a 3D printed housing that connects to the Ethernet link using an external USB dongle as seen in Fig. 4.

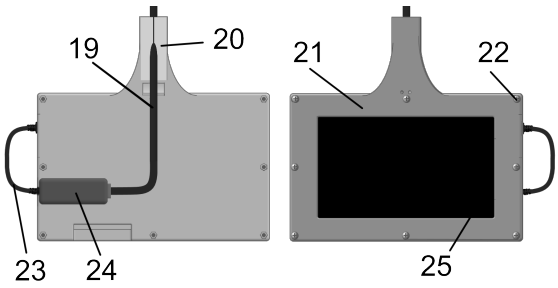

Figure 4: **Tablet-based remote control system. 19.** Reinforced CAT 5E cable (70 m total). **20.** Cable clamp. **21.** 3D printed splash-proof housing. **22.** M3 screws. **23.** Micro USB connector. **24.** Ethernet network card. **25.** Lenovo Tab 3.

3.1 Propulsion

As the ROV uses DC brushed motors (with a current rating of 5 A), we chose low cost Cyron MDD10A motor drivers to connect them to the Pi. Since the current rating of the drivers is twice the required maximum we can count on lower internal heating.

3.2 Sensors

The ROV is equipped with various sensors: an IMU, a water pressure (depth) sensor, a battery voltage sensor, an internal air pressure and a temperature sensor. We've also used a Pi Camera Module v2, and later the D160 fish eye modification for video recording.

We've used the MPU9250 IMU, which offers an accelerometer, gyroscope and magnetometer. The expansion board used is the GY-91 which also contains a BMP-280 pressure and temperature sensor, so that is used to log the state of internal air as well.

For the water pressure sensor we've selected a low-cost transducer intended for measuring the pressure inside hoses with a capability of 0.8 MPa (which equates

to roughly 80 m depth of seawater). As the sensor is analog and the Pi has no analog reading pins we've also added a 10 bit 4-channel ADC to link the two together. As the ADC has spare channels we can also measure battery level using a voltage divider circuit.

4 Software

We've based our software around a socket.io link, which allows communication between the C# Unity app running on the remote control tablet, and the node.js server running on the submarine itself, which controls most of the functionality. To avoid having a rooted system on the tablet and a static IP, the Raspberry Pi instead acts as a DHCP server and assigns a known IP to any Ethernet connected device.

Not all of the system can be managed with node.js directly, as the libraries that allow full Pi Camera configuration have only been provided for Python and C++. Since the system requires the usage of GPU splitter ports to simultaneously stream and record video while taking pictures, the camera control part ended up being implemented in Python, which is then launched and managed from the node.js server.

4.1 Stabilization

We implemented two automatic functions: depth hold and heading hold. Establishing proportional control between the detected depth and target depth turned out to be enough to archive a satisfactory depth hold.

A more advanced system was required for holding a specific heading. An AHRS IMU fusion system [8] was added to get a good internal belief state regarding the submarine's orientation. We've then implemented a 360-degree-capable proportional rotational system, which allows the pilot to input a single forwards or backwards speed while the vessel holds orientation.

Full PID controllers did not turn out to be required for adequate stabilization due to high drag and inherent damping and were also infeasible to properly tune during operation.

The system manages the aforementioned sensors and streams their data to the tablet, uses them internally for stabilization and also records them into a log file for later use.

4.2 Remote control app

We've implemented a front-end interface for controlling and configuring the submarine from a tablet computer, which was developed using Unity. The video arriving from the camera in a mjpeg format is displayed using an adapted version of the Mjpegprocessor library [9], while the rest of the telemetry and commands get routed through the socket.io library [10]. Fig. 5 shows the GUI while streaming the image of a calibration board, which indicates the camera field of view.

The main functionality is remote control, which is done either with manual joysticks or by selecting a direction using the bottom compass bar and a depth using a vertical depth indicator. In order to provide as much

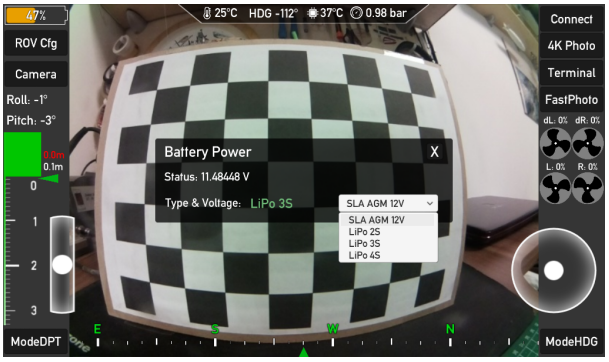

Figure 5: Layout of the tablet interface, showing the relevant telemetry and video feed.

on-site debugging power, the app can set all stabilization and video recording parameters and allow for rudimentary terminal access.

4.3 Visualization and analysis

We built a Godot engine visualizer to attempt to process the recorded log file data and reconstruct the ROV's odometry to some degree as seen in Fig. 6.

The logged data gives us two absolute measurements: rotation (based on the fused IMU data) and depth (calculated from water pressure readings). It's not possible to completely determine the movement path using just these two values, however, we also logged the individual thrusters' speeds which were used to infer movement velocity. While it's not possible to compensate for drift due to waves and currents the resulting movement path is still good enough in most cases.

While the data was logged every 100 ms, the visualization ran at a higher frequency (16 ms for 60 frames per second) so the data had to be linearly interpolated for smooth movement. A synced on-board video of a dive with the telemetry visualization which can be viewed on Youtube [11].

Figure 6: The ROV being rendered in Godot, visualizing a recorded dive and drawing a trail behind the vessel.

5 Evaluation and testing

The finalized ROV shown in Fig. 7 performed as required and is capable of exploring a coastal area while recording clear video as shown in Fig. 8. An edited

compilation of video recordings that sums up research, construction and testing can be viewed on Youtube [12].

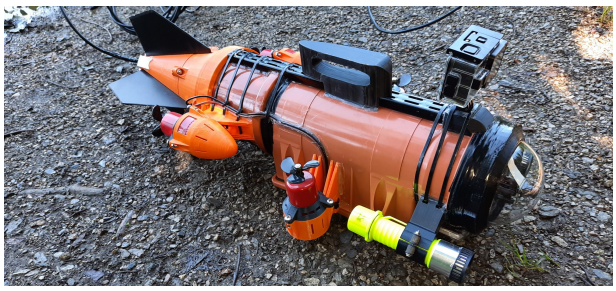

Figure 7: The ROV with an action camera mounted on the top rail.

To determine the effects of different buoyancy on the system's operation, tests were done in both salt and fresh water. The difference ended up being minor, however, the depth calculation can be off by 2.5 % due to the extra mass of salt water. We therefore added parameters that can be changed in the mobile app to compensate for the changing environment.

We've used two different action cameras to record external video during testing: the Apeman A80 and later a GoPro Hero 9 Black combined with a red filter as the previous setup proved unsatisfactory. Having a red filter allows us to do an initial colour correction by restoring the lower wavelengths which get absorbed far more in water and lead to over-saturation of green and blue.

The recorded data shows that the air pressure in the ROV stays relatively constant regardless of depth, which would mean that there is minimal flex in the pressure hull as intended. The internal air temperature also stayed close to the external water temperature. It would appear the lack of isolation in the hull provides for excellent cooling and would allow for higher TDP computers to work without issues.

Figure 8: Fan mussles which were found in large numbers when testing in Izola, some of them decayed due to parasite infestation, confirming news sources [13].

Another addition to the system would be a sonar system for sea floor and obstacle detection, however, there are no affordable sensors that are capable of being integrated into a custom system at this time. One would need to construct a custom driver circuit for a commercial fish finder sonar transducer (e.g. TL88E, XF02), as conventional sonar sensors designed for AGV/AUV atmospheric usage lack the required power and do not operate in effective frequencies, which lowers the range and accuracy under water to unusable levels.

6 Conclusion

In this paper, we described the process of building a low-cost underwater DIY ROV with open-source hardware and software, useful for light research and monitoring of shallow coastal regions. We've achieved this goal by choosing a crush depth of 25 m and used less resistant and subsequently cheaper materials, as well as using 3D printed components to fill the gap between other parts. The total cost of the system ended up being around 400 €, excluding the tablet computer and action cameras.

The vessel is stable enough to record useful video due to the included IMU and sensor fusion, while being able to utilize its high top speed to rapidly traverse a chosen area and locate points of interest. The external mechanical design is robust enough to handle expected conditions, however the electrical internal setup could be further improved by designing a dedicated printed circuit board.

7 Acknowledgements

The authors would also like to thank professors Dr. Nikolaj Zimic and Dr. Matjaž Vidmar for their help and contributions in the research phase of the project.

References

[1] Marine Food Webs Are on the Brink of Collapse Because of Climate Change, https://futurism.com/marine-food-webs-brink-collapse-because-climate-change

[2] iBubble - Your Personal Underwater Cameraman, http://ibubble.camera/product/ibubble-autonomous-underwater-drone

[3] Blue Robotics, http://bluerobotics.com

[4] Blue Dot Underwater Drones, https://www.bluedotrov.com

[5] Source code, schematics and documentation, https://github.com/MoffKalast/RaspberryUnderwaterROV

[6] Homebuilt ROVs - Seafox Retrofit Thrusters, https://www.homebuiltrovs.com/seafoxretrofitthrusters.html

[7] Homebuilt ROVs Mayfair 750 GPH Bilge Pump Thruster Testing, http://www.homebuiltrovs.com/mayfair750test.html

[8] ahrs - npm, https://www.npmjs.com/package/ahrs

[9] SampleUnityMjpegViewer, https://github.com/DanielArnett/SampleUnityMjpegViewer

[10] socket.io-unity, https://github.com/floatinghotpot/socket.io-unity

[11] ROViz visualization, https://www.youtube.com/watch?v=M0sli2o5RJY

[12] Construction and testing video, https://www.youtube.com/watch?v=DufHhX7p4Xk

[13] Slovenian sea researchers uncover a number of deceased fan mussles, https://www.sta.si/2797119