OtterROS: Picking and Programming an Uncrewed Surface Vessel for Experimental Field Robotics Research with ROS 2*

Thomas M. C. Sears[1], M. Riley Cooper[1], Sabrina R. Button[1], and Joshua A. Marshall[1]

***Abstract*— There exist a wide range of options for field robotics research using ground and aerial mobile robots, but there are comparatively few robust and research-ready uncrewed surface vessels (USVs). This workshop paper starts with a snapshot of USVs currently available to the research community and then describes "OtterROS", an open source ROS 2 solution for the Otter USV. Field experiments using OtterROS are described, which highlight the utility of the Otter USV and the benefits of using ROS 2 in aquatic robotics research. For those interested in USV research, the paper details recommended hardware to run OtterROS and includes an example ROS 2 package using OtterROS, removing unnecessary non-recurring engineering from field robotics research activities.**

I. Introduction

Aquatic mobile robots have the potential to become vital tools for environmental monitoring, infrastructure assessment, emergency response, shipping and transportation. Unfortunately, researchers with an interest in uncrewed surface vessel (USV) autonomy have been limited by the number of available platforms that are suitable for research purposes. Environmental conditions such as waves and current, affordability, and software compatibility further limit the selection of USVs for researchers seeking to field test their designs. This paper presents our group's effort to take one commercial product—the Otter USV by Maritime Robotics [1]—and convert it to a system that is suitable for conducting field robotics research using Robot Operating System (ROS) 2.

Offroad Robotics is a field and mobile robotics research group located on the North-East shore of Lake Ontario (Kingston, Ontario, Canada). Motivated to explore robotics in our local waterways, we decided to procure a USV. Our initial choice was the Clearpath Robotics Heron USV because it was well known and used by the robotics research community; but the Heron was discontinued before we could initiate a purchase. As a result, we conducted a survey of commercially available USVs to identify those capable of open water operations and with suitable software interfaces.

*This project was funded in part by the NSERC Canadian Robotics Network (NCRN) under NSERC project NETGP 508451-17, the Vanier Canada Graduate Scholarships program, and Queen's University's USSRF program. The Otter USV was purchased with generous support from the Ingenuity Labs Research Institute's Research Equipment Fund.

[1]T. M. C. Sears, M. R. Cooper, S. R. Button, and J. A. Marshall are all with the Department of Electrical & Computer Engineering and the Ingenuity Labs Research Institute, Queen's University at Kingston, Ontario, Canada. {thomas.sears, joshua.marshall}@queensu.ca

The authors do not work for, consult, own shares in or receive funding from any company or organisation that would benefit from this article, and have disclosed no relevant affiliations beyond their academic appointments.

Fig. 1: The Otter USV (left) during experiments in Lake Ontario. Encounters with pleasure craft are common, but occasionally we cross paths with history.

Ultimately, the Otter was selected because it exceeded the Heron in every capacity, although it lacked ROS support.

In this paper, we share hardware modifications, software developments, and lessons learned in field testing with the Otter USV over the last two years, in an effort to lower the barrier of entry to USV research for others in the field robotics community. This includes discussions about:

- why the Otter was selected and how it has performed as an experimental robotics platform;
- the architecture and use of OtterROS, a new ROS 2 package that can interface with the Otter;
- hardware necessary to enable ROS on the Otter; and
- a control package built on OtterROS, which serves as an example of how the Otter USV and OtterROS can be used to conduct research in real environments.

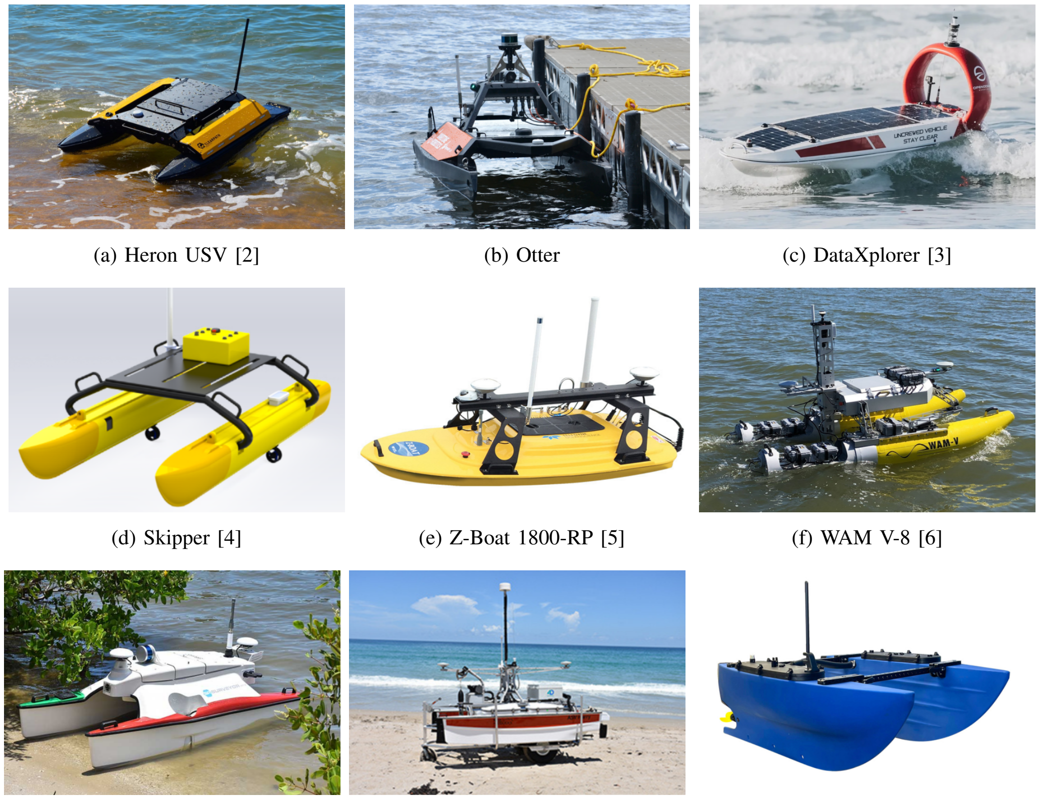
This invitation to use OtterROS also includes the opportunity to share developments and algorithms that further advance autonomy for the Otter USV. Full details and source code for OtterROS can be found at: `https://github.com/offroad-robotics/otter_ros`.

II. Background

In 2022, our research team chose to purchase an Otter USV after surveying available surface vessels for field robotics research. ROS 2 compatibility was sought, but it was uncommon amongst the identified USVs and unavailable on the Otter. Table I highlights the findings of this USV survey. This section discusses what to consider when selecting a USV in research and why we selected the Otter for our work.

TABLE I: Survey results for USVs capable of operating in open water. They vary in size, design, price, and performance. Software interfaces are varied and may require manufacturer support to enable robotics work. Configuration includes steering (S), propulsion (P), and hull (H) design. A "–" indicates information unavailable at time of writing.

USV Name	**Manufacturer**	**Dimensions L×W×H [m]**	**Draft [m]**	**Relative Price**	**Software Interface**	**Endurance [hours]**	**Top Speed [m/s]**	**Payload [kg]**	**Configuration**
Heron (*discontinued*)	Clearpath Robotics (Canada)	1.4×1.0×0.3	0.12	$	ROS MOOS-IvP C++	2	1.7	10	S: Differential Drive P: Water Jet H: Catamaran
Otter	Maritime Robotics (Norway)	2.0×1.1×1.1	0.32	$$	Backseat Driver	20	3.0	30	S: Differential Drive P: Propellers H: Catamaran
DataXplorer	Open Ocean Robotics (Canada)	3.6×0.9×1.6	0.5	–	–	>24	3.0	60	S: Rudder P: Propellor H: Monohull
SKIPPER	Independent Robotics (Canada)	1.2×0.9×1.0	0.3	$	ROS2	6	1.0	25	S: Differential Drive P: Propellers H: Catamaran
Z-Boat 1800-RP	Teledyne Marine (USA)	1.8×1.0×1.1	0.28	$$	Backseat Driver	4.5	5.0	30	S: Differential Drive P: Propellers H: Monohull
WAM V-8	Marine Advanced Robotics (USA)	2.5×1.2×1.0	0.1	$$$	ROS MOOS-IvP	10	1.5	45	S: Differential Drive P: Propellers H: Catamaran
SR-Surveyor M1.8	SeaRobotics Corporation (USA)	1.8×0.9×–	0.2	–	–	7	2.1		S: Differential Drive P: Propellers H: Catamaran
SR-Utility 2.5	SeaRobotics Corporation (USA)	2.5×1.2×–	0.1	–	–	20	3.8	60	S: Differential Drive P: Propellers H: Catamaran
Blue Boat	BlueRobotics (USA)	1.2×0.9×0.5	–	<$	Python C++ Rust	>24	3.0	15	S: Differential Drive P: Propellers H: Catamaran



(a) Heron USV [2] (b) Otter (c) DataXplorer [3]

(d) Skipper [4] (e) Z-Boat 1800-RP [5] (f) WAM V-8 [6]

(g) SR-Surveyor M1.8 [7] (h) SR-Utility 2.5 [7] (i) Blue Boat [8]

Fig. 2: Images of the USVs identified in the survey. Note: Images are not at a constant scale; see Table I for dimensions.

A. USVs for Field Robotics

What defines a *useful* USV of course varies by user. Our group considered mechanical, electrical, and administrative factors that would affect our ability to use a USV for field work. We evaluated each of the USVs listed in Table I against the criteria in Table II. These factors can guide other groups in acquiring a USV suitable for their robotics work.

Table II also highlights our group's specific considerations and how the Otter addressed each factor. Multiple USVs met our needs on each individual factor, but the Otter was the best overall match. The portability, runtime, and payload capacity of the Otter were of particular interest.

B. Middleware for USV Autonomy

As is similar across many domains of robotics, multiple software libraries are publicly available and used for USV guidance, navigation, and control. MOOS, developed at the Oxford Mobile Robotics Group (MRG), is a cross-platform robotics middleware originally developed for maritime applications [9]. MIT's Laboratory for Autonomous Marine Sensing Systems (LAMSS) manages MOOS-IvP, which adds many autonomy features on core MOOS [9]. The Underwater Systems and Technology Laboratory at the University of Porto, Portugal, created DUNE (Unified Navigation Environment), for low-level control of USVs [10]. Neptus, a ground control system with a graphical interface, allows operators to control vehicles that run DUNE. MOOS and DUNE are two options among many with origins in USV and uncrewed underwater vehicle (UUV) research, and have been deployed in numerous published works, including [11], [12].

Since 2009, the Robot Operating System (ROS) has grown to become the de facto software development and research interface for robots [13]. With the latest iteration of ROS 2, the ease of implementation and flexibility of ROS has been matched with significant performance enhancements suitable for the challenges of real-world deployment [14]. ROS 2 is now widespread in ground and aerial mobile robotics.

Although our group was predisposed to choosing ROS 2, it is worth comparing these three main options before committing significant development effort. Other groups looking to test their own guidance, navigation, and control algorithms may consider the elements of Table III in selecting their USV software architecture. Despite ROS 2 being uncommon in USV research, Offroad Robotics found that its general popularity, available documentation, and ease of development made it the most desirable for academic research work.

C. Data Interface with the Otter USV

Many USV manufacturers provide some form of interface with their onboard computer (OBC), and often oblige the "backseat driver" philosophy. This was popularized by MOOS, where low-level control and vehicle autonomy are separated [15]. In this perspective *control* is the domain of manufacturers and aims to manage the vehicle's actuators to achieve certain heading, speed, or other system states. The way the vehicle achieves these states is not necessarily of interest to the operator. *Autonomy*, or mission planning, is the realm of the backseat driver system, which uses high level navigation data to decide what values to send back to the control system.

MOOS was intended for mission planning, so the backseat driver paradigm often does not provide access to the low-level telemetry or actuator control. Furthermore, this architecture does not prescribe what information should be provided to an external computer. Our recommendation is to speak with manufacturers to fully understand what can and cannot be accessed. This greatly impacts which USVs are useful for field robotics—navigation and planning experiments may only require high-level telemetry from the USV, while control research may demand direct access to motor commands.

Maritime Robotics provides a backseat driver interface through the Otter's onboard local area network. The OBC broadcasts navigation and system state information, including position, orientation, and speed, which any device can receive. The OBC also listens for incoming commands for external control[4]. These commands range from path following to motor thrust control, which is particularly valuable for control research. The backseat driver architecture is illustrated as the communication between the Translation and System layers of Fig. 3. In this diagram, the "Payload" computer is separate from the OBC, which in our case runs ROS 2 through our package OtterROS.

III. The Otter USV for Robotics Research

The Otter USV is marketed towards "efficient and precise data acquisition, environmental monitoring, and surveillance in sheltered, coastal and shallow areas" [1], which is a good starting point for a field research robot. This section highlights how the physical configuration of the Otter and the data interfaces available are suitable for robotics research.

A. Mechanical Design

The Otter uses a modular design, with separate waterproof boxes housing different subsystems. A typical Otter will have an "OBS" and "Targa" box, with the "Payload" box reserved for hydrographic instruments. We purchased this box empty to create our own robotics payloads. The breakdown of compartment contents and locations is shown in Fig. 4.

Commercial off-the-shelf components can be used inside the Otter's sealed compartments. This reduces the need to purchase IP67-rated parts or fabricate individual housings for each component. Connecting the compartments is more difficult because it requires IP67-rated connectors and cables. Sourcing waterproof cables can be a costly and slow process, which other USVs avoid by having a single sealed compartment. To minimize a need for many exterior cables, power

[2]Source: A combination of lived experience and data from `https://weather.gc.ca/marine/index_e.html`

[3]A snapshot of search result count from March 2024 from Google scholar and GitHub for "DUNE Unified Navigation Environment", "MOOS-IvP", and "ROS 2" (and "ROS 2 USV").

[4]This is an optional add-on from Maritime Robotics and must be enabled.

TABLE II: Factors to consider in USV selection for robotics research and why we at Offroad Robotics selected the Otter USV.

Factor	**Our (Offroad Robotics) specific considerations**	**Otter rationale**
Mechanical factors		
Wave conditions	Coastal, 1 m high[2]	Rated for Sea State 2 (0.5 m), so suitable for most days.
Water current	St. Lawerence River, Great Cataraqui River	Max speed greater than anticipated currents for upstream travel
Operation depth	Lakes and rivers in Ontario, no hard requirement	32 cm draft allows transit through most waterways
Transportation and storage	Indoor storage, fits in regular vehicle, launch with two people	Fully assembled, fits inside elevator, cargo van, and can be lifted by two people
Payload space	Computer and sensing equipment, sonar-ready	Large compartments for electronics, designed for hydrographic surveys
Reconfigurability	Mounting points, machinable, open surface area	Room for sensors above and below chassis, aluminum structure readily modified
Electrical and data factors		
Runtime	Full day testing, overnight experiments	20-h rating (at 1 m/s), swappable batteries
Power	Simultaneous use of computers, active sensors while driving	4 kWh provides energy for full-day operations
Remote operations	Cover local lakes and rivers, within line of sight	0.5–1.0 km maximum range for Wi-Fi connection
OBC interface (physical)	Onboard connection (e.g., Ethernet, USB, ...) for data transfer	UDP connection over Ethernet for reliable two-way communication
OBC interface (software)	ROS 2 (preferred), Python, or similar compatibility	No ROS 2 support, but NMEA messages over UDP can be adapted for ROS 2
System telemetry	Navigation, state, system data	External interface provides location, speed, and other vehicle data
Remote commands	Manual inputs (preferred), speed/heading control	External interface allows motor force control and activation of all built-in modes
Other factors		
Availability	No export control, shippable to Canada	Maritime Robotics has local distributors, can also sell directly
Price	Upfront cost, maintenance, operations	$100,000 budget, commercial components readily serviced, minimal consumables

TABLE III: Comparison of major open source USV control software.

	DUNE	**MOOS-IvP**	**ROS 2**
First release	2007	2001	2017
Maintainer	LSTS	MRG and LAMSS	Open Robotics
Development origins	USVs	USVs	Generic
Published use[3]	~500	~1,000	~30,000 (~100 with "USV")
Documentation	Limited	MIT User Guide	ROS 2 Docs, YouTube, ...

and data hubs are used inside the payload box in order to use a single external power and Ethernet cable connection.

B. Power and Data Interfaces

The Otter provides a daisy chain connection for power and data to an external payload box as illustrated in Fig. 4. The Ethernet connects other devices to the Otter local area network. The power connection is an unregulated DC supply, so to simplify payload integration, Maritime Robotics sells a regulator board that provides 5 V, 12 V, 30 V, and variable voltage outputs.

Communication between compartments and devices on the Otter is through a local area network. USV-to-shore communications is with a high-power, directional Wi-Fi system. New devices can be easily added to the network and accessed remotely, making ROS control simple.

The backseat data is transmitted as custom NMEA messages, encoded according to Maritime Robotics specifications. From the Otter, two regularly transmitted messages contain position (latitude, longitude), speed, course-over-ground, orientation, and angular velocity. Additional messages indicate the vehicle mode, motor state, battery capacity, and current power consumption. These data are broadcast over the network at a rate determined by the Otter OBC.

Input commands are similarly encoded as NMEA messages to set the control mode and settings for the vessel. Maritime Robotics provides a number of command options:

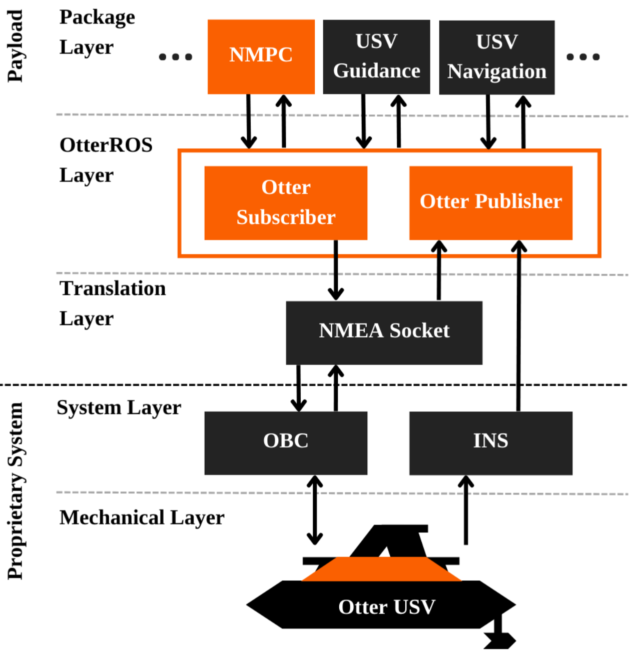


Fig. 3: Adapting the Otter's external interface for ROS 2 with *OtterROS*. The Payload computer running ROS 2 communicates with the Otter over a local network connection. OtterROS manages the interface between the USV and new programs in the Package Layer.

drift, where the motors are turned off, *manual*, where the motor inputs are provided directly, and high level control, such as *course and speed* and GNSS-based *station keeping*.

Although the Otter provides a motor control interface and some navigation data, there are hardware and software system limits. Drawbacks to the Otter backseat driver include:

- no accelerometer data from the navigation system;
- no position uncertainty or GPS state information;
- unsigned motor RPM values (i.e., cannot see which direction propellers are spinning); and
- limited sample rate (1–20 Hz) for incoming data.

Despite these drawbacks, many of which could be addressed

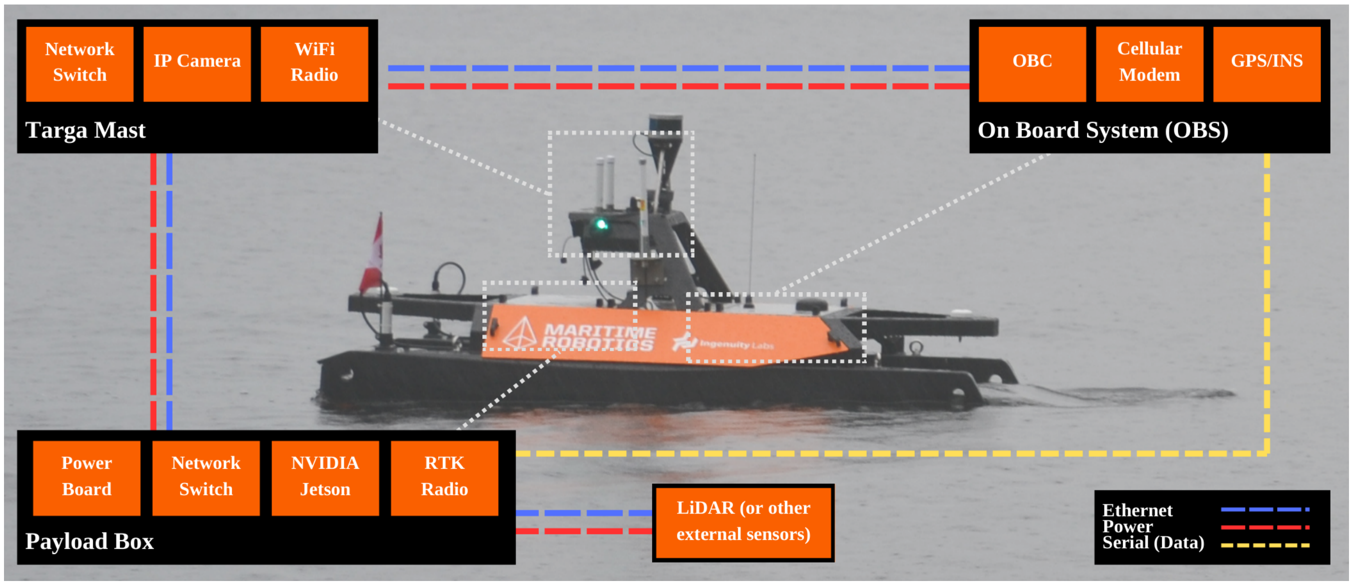


Fig. 4: The Otter uses a compartmental design to house electrical systems. Connections between compartments must be made with waterproof connectors and cables. Locations of each compartment indicated in grey.

in future software updates from Maritime Robotics, the vehicle has been successfully used in numerous field experiments.

IV. OtterROS

To use ROS 2 with the Otter USV, a conversion from the OBC backseat driver interface to ROS was required. *OtterROS* enables this communication through ROS 2 topics. New autonomy and sensing algorithms can be built on top of OtterROS to take advantage of the wide range of ROS 2 compatible hardware and software. This section presents an overview of OtterROS and provides an example of how it can be used. For complete details, see the OtterROS documentation and code [16].

A. Implementation

OtterROS uses a publisher-subscriber architecture to let users communicate with the OBC through ROS 2 topics. Lower-level communication between the OBC and OtterROS is encoded as NMEA messages through UDP, so a custom version of `pynmeagps` [17] is included as a translator for the OtterROS package. New programs can focus entirely on the ROS 2 topics.

A base launch file provides an easy way of starting OtterROS. Additional parameters provide optional connectivity for Velodyne Puck LiDAR and SBG Ellipse-D sensors, and for bagging (i.e., data recording). Bagging is saved as an *mcap* format, which allows easy data sharing and replay on machines that do not have OtterROS.

B. Available Data (Publisher)

OtterROS publishes all available data from the OBC to the topics listed in Table IV. Topics are updated asynchronously as data are received by OtterROS (the actual rate is determined by settings in the Otter OBC). Standard position and orientation topics allow for quick use in ROS 2 visualization tools or in third-party packages.

TABLE IV: Topics and data published by OtterROS. Standard ROS 2 message types are used when possible.

Topic	Message Type	Populated Parameters
`otter_gps`	NavSatFix	Latitude, longitude, altitude
`otter_gps_time`	OtterTime	GPS time
`otter_imu`	Imu	Orientation, angular velocity
`otter_status`	OtterStatus	Motor RPM, temperature, power consumption
`otter_cogsog`	COGSOG	Course, speed over ground, local speed vector

C. External Commands (Subscriber)

Commands to the Otter are sent through ROS topics. The OtterROS subscriber is designed to monitor specific command topics and, upon seeing a change, relay the appropriate command to the OBC. It is left to the node publishing to the command topic to determine how often the commands should be sent. Available commands, and their required inputs, are shown in Table V.

D. Development Example: Otter Control

We developed a Nonlinear Model Predictive Controller (NMPC) for the Otter USV. This application is briefly presented here as an example of a ROS 2 package built on OtterROS. Complete code is available at [16] in the `otter-control` package. Details about the NMPC implementation are omitted in this paper, but can be found in [18].

1) OtterROS Interface: The NMPC controller subscribes to `otter_gps`, `otter_imu`, and `otter_cogsog` from OtterROS. These messages provide position, orientation, and speed information, respectively. As these messages are published sequentially from OtterROS, the `message_filters` package is used to synchronize near-simultaneous data.

TABLE V: Topics watched by the subscriber to be sent to the Otter. Commands are relayed to the OBC immediately and only when there is an update to the topic.

Command (`topic`)	**Parameters**
Drift (`drift_cmds`)	On/off flag
Motor Input (Manual) (`control_cmds`)	Surge force $x \in [-1, 1]$ Sway force y ignored Torque $z \in [-1, 1]$
Station Keeping (`station_keeping_cmds`)	NavSatFix (Lat, Lon) Speed $\in [0, v_{\max}]$
Course and Speed (`course_speed_cmds`)	Course $\in [0, 360]$ Speed $\in [0, v_{\max}]$

This controller was designed to calculate motor inputs force in order to follow waypoint defined paths, so we use the Otter's *manual* back seat command. We do this by periodically publishing calculated surge and torque values to `control_cmds`. When the OtterROS subscriber sees the change, the command is sent over the network to the OBC.

2) Use and Customization: ROS 2 provides numerous ways to launch multiple nodes. The most convenient way to work with custom nodes and OtterROS is to use the OtterROS launch file `otter_launch_base`, which starts the core OtterROS publisher and subscriber and begins logging data. Custom nodes can then be run (or additional launch files launched) that interact with OtterROS. The NMPC controller package includes launch file examples that show how to do all of this in a single launch script.

Because the NMPC node is built in Python, it can be quickly modified and updated, and can easily take advantage of external packages (e.g., `numpy`). As an example, the controller uses an external nonlinear optimization package (CasADi [19]) for real-time optimization.

3) Validation: We tested the NMPC against the manufacturer's built-in waypoint following controller (which, to the best of the authors' knowledge, is a cascaded PI/PD speed/heading controller) on a figure eight path. The NMPC outperformed the supplied controller, demonstrating the utility of OtterROS. We also note that with our current hardware configuration (i.e., the NVIDIA Jetson AGX Orin developers kit), the NMPC ran at 10 Hz with a 4 s horizon.

V. Otter USV Integration

Specific hardware may vary by USV and need (e.g., computing platform, sensors), but mechanical and electrical integration with the Otter will be similar for any build. If followed, this section will enable OtterROS to be rapidly deployed on the Otter USV.

A. Computing Platform

At the heart of OtterROS is a computer platform running ROS 2. We selected the NVIDIA Jetson AGX Orin (developers kit) because it provides the most compute power in this form factor, met the power and space requirements, and is easy to use with USB, Ethernet, Wi-Fi, display, and GPIO interfaces. With growing use of machine learning tools in robotics, the Jetson GPU-focused platform was also preferred over a CPU-focused system like the Intel NUC.

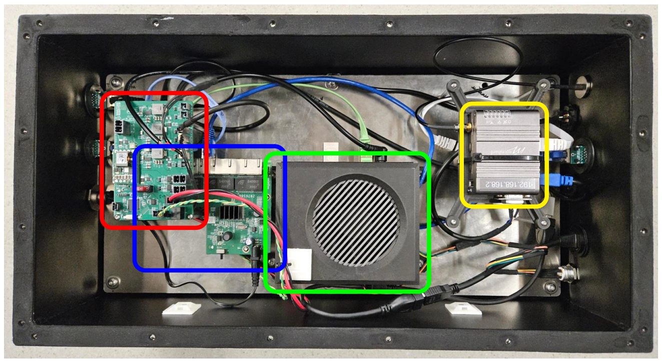


Fig. 5: Hardware inside the aft (payload) container. Bulkhead connectors are visible on port (left) and starboard (right) walls for connections to other boxes and systems. Main components are the power board (red), network switch (blue), and NVIDIA Jetson inside the mounting cradle (green). An optional RTK radio (yellow) is also shown.

TABLE VI: List of components that comprise the Otter payload box for OtterROS operations. Part names and numbers can be found at [21].

Description	**Part name**	**Count**
Power distribution	MR power board	1
Power input cable	MR bulkhead power harness	1
Power harness (internal)	45130-0203 to device	3
Network switch	Netgear GS308	1
Ethernet connectors	ROP-5SPFFH-TCU7001	3
Network cables	Cat6 (generic)	4
Payload computer	NVIDIA Jetson AGX Orin Developers Kit	1
USB bulkhead	UA-30PMFL-LC7B20	1
Switch and indicator	MPB16-CARE-6-JR-0V	1

NVIDIA releases their *JetPack SDK* to take advantage of the unique capabilities of the Jetson with "Jetson Linux" [20]. JetPack 5 is built on Ubuntu 20.04, so OtterROS was developed for ROS 2 *Foxy Fitzroy* (the latest ROS 2 version compatible with Ubuntu 20.04). Users looking to use the latest version of ROS 2 should consider using the JetPack 6 Developers Preview, which is built on Ubuntu 22.04 and compatible with ROS 2 *Humble Hawksbill*.

B. Integration and Supporting Hardware

Integrating the Jetson with the Otter USV required additional hardware, cabling, and a mounting solution. The layout of our payload box is shown in Fig. 5, which highlights the most significant components: the Jetson, the Maritime Robotics power board, and a network switch. A list of internal and external parts is presented in Table VI. Full details of our design, including part numbers, drawings, and assembly instructions, can be found at [21].

The base plate supporting all components is mounted on posts epoxied to the bottom of the box. New plates can be easily swapped for different payloads. This elevated design also protects against any leaks that may form during extended testing, as water can pool underneath the plate without making electrical contact with any components.

When possible, devices were removed from housings and mounted directly to the base plate. The Jetson developers kit is an example where this cannot happen, so a screwless 3D printed "cradle" was designed to securely hold it down (pictured in Fig. 5). Metal standoffs were used to maximize use of vertical space in the compartment and allow part placement above the large bulkhead connectors.

C. Optional Hardware

Additional hardware for RTK corrections and navigation experiments was included in our Otter refit. The Microhard pX2 radio with direct serial connection to the OBS navigation system provides a relay for RTK corrections from a shore-side GPS base station. This device is not IP rated, so is stored inside the payload box. A Velodyne Puck LiDAR was added for navigation experiments and demonstrates how external devices may be added to this design. The Puck is rated IP67, but the power and Ethernet terminations are not, so an adapter box was 3D printed with holes for IP67 connectors. After installing the cables, gaps were sealed with silicone and the box mounted to the underside of the Targa. Hardware designs for both systems are available at [21].

VI. Field Testing Experiences

We have tested the Otter in a wide range of conditions, across all seasons, and for different robotics experiments. In this section we present lessons learned about USV field work and experiences (both positive and negative) with the Otter USV over two years of testing.

A. Research Use Cases

1) Wave Mapping: Initial work with the Otter targeted spatiotemporal mapping of waves [22]. This exposed the Otter to waves that were approximately 50 cm high along the coast of Lake Ontario. During these experiments, the Otter was operated with the built-in waypoint-following controller and successfully navigated circular and linear paths. Data from the onboard navigation system was collected through OtterROS by the Jetson for offline processing.

2) Model Predictive Control: Control experiments with the Otter focused on the use of a nonlinear model predictive controller (NMPC) and data driven system modelling to reduce path following errors [18]. Data collection through OtterROS provided the input for offline system modelling. The NMPC used CasADi [19] for nonlinear optimization and provided raw motor inputs to the OBC through OtterROS.

3) Coastline Localization: Navigation research using LiDAR scans for coastline matching used OtterROS as the connection between USV navigation data and LiDAR data. By building on ROS 2 and OtterROS, the LiDAR data was easily included by launching existing ROS 2 packages from Velodyne. This work benefited from ROS 2 bagging, which enabled simulated experiments for continued development of the algorithms without a need to return outside.

B. Test Conditions

The Otter was launched numerous times in Lake Ontario (Kingston, Ontario, Canada), in the Great Cataraqui River (Kingston, Ontario, Canada), and in Gull Lake (Milton, Ontario, Canada). Operating conditions varied, as shown in Fig. 6, and included:

- wave conditions ranging from flat to 50 cm high;
- air temperatures from –5 °C to 25 °C;
- heavy rain; and
- water temperatures near freezing, ice on the surface.

C. System Performance

In our experiments, the Otter performed well in most conditions, but certain challenges were identified after extensive testing. We also encountered environmental and human factors that affected experiments, which we believe would be common to most USV field experiments. These challenges, and when possible, solutions, are presented in Table VII.

VII. Conclusions

Options are are limited for robotics researchers looking to work with USVs in ROS 2. To this end, this paper presents a survey of commercial USVs and the process that researchers at Offroad Robotics underwent in selecting a suitable vessel for field robotics research applications. The development of OtterROS, a ROS 2 package for the Otter USV by Maritime Robotics, is discussed and a guide for its integration on the Otter USV is given for researchers looking to leverage these developments. Lessons learned after extensive field testing are shared for researchers active or interested in USV field work, which apply to any USV. With the release of OtterROS and the discussions herein, we hope to help build a growing community of USV robotics researchers.

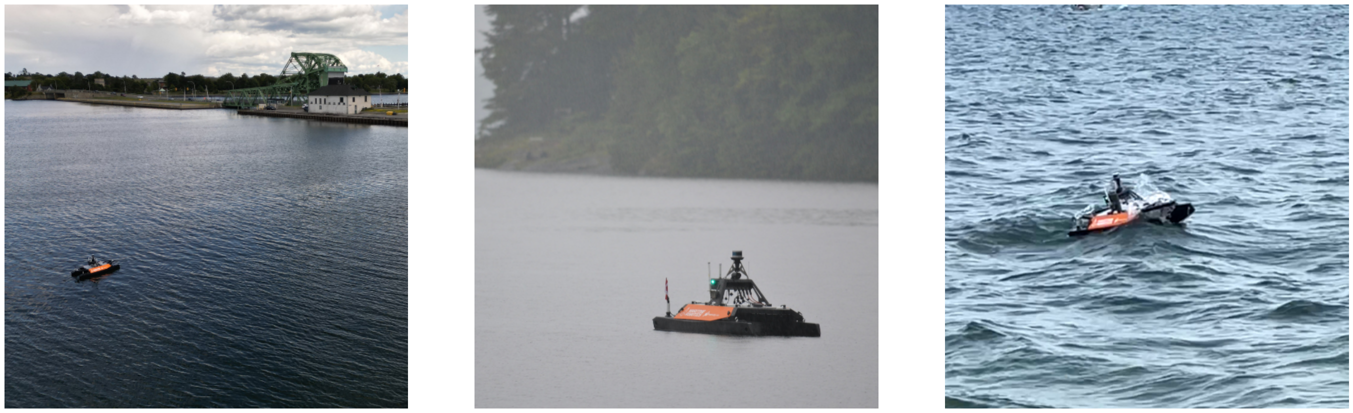
Acknowledgments

Field robotics experiments in Gull Lake (near Minden, Ontario) took place on the traditional territories of the Anishinabee and Mississauga[5]. Experiments in Kingston, Ontario took place on the traditional territories of the Algonquin, Anishinaabe, Haudenosaunee, and Huron-Wendat[5].

References

[1] Maritime Robotics, "The Otter," https://www.maritimerobotics.com/otter.
[2] Clearpath Robotics, "Heron USV," https://clearpathrobotics.com.
[3] Open Ocean Robotics, "DataXplorer," https://www.openoceanrobotics.com.
[4] Independent Robotics, "Skipper USV," https://www.independentrobotics.com.
[5] Teledyne Marine, "Z-Boat 1800-RP," https://www.teledynemarine.com.
[6] Ocean Power Technologies, "WAM V-8," https://wam-v.com/.
[7] Sea Robotics, "SR-Surveyor and SR-Utility," https://www.searobotics.com/.
[8] Blue Robotics, "Blue Boat," https://bluerobotics.com/.
[9] M. R. Benjamin, H. Schmidt, P. M. Newman, and J. J. Leonard, "Autonomy for unmanned marine vehicles with MOOS-IvP," in *Marine Robot Autonomy*, M. L. Seto, Ed. New York, NY: Springer New York, 2013.

[5]`https://native-land.ca`



(a) Strong wind and currents cause waves at the base of the Great Cataraqui River.

(b) Heavy rain and calm water in Gull Lake (Minden, Ontario, Canada).

(c) Large waves along the coast of Lake Ontario in Kingston, Ontario.

Fig. 6: The Otter USV has performed well across a range of environmental conditions.

TABLE VII: Summary of USV field testing challenges and experiences.

Factor	**Explanation and suggested solution**
Otter USV	
Motor failure	Mechanical failure in a motor from unknown cause. Possible propeller strike or due to rapid control commands (to be determined).
Motor delay	The Otter USV has an input delay of approximately 2 s when starting from stationary. A small initial input may mitigate power on delay.
Communication range	Difficult to gauge when communications will be lost. Constant monitoring required if Otter is not on a return path.
Local network dropouts	In cold conditions, the Otter suffers from network dropouts spanning a few seconds. No solution yet.
Environmental	
Motor blockage	Jammed motors may mean the USV will be unable to return to shore. Scout ahead before deploying boat and seek local experts to avoid any (possible) vegetation.
Ecosystem damage	USVs can access areas inaccessible by other powered craft, causing damage to mostly undisturbed environments. Researchers should connect with local ecosystem experts to determine if, where, and when it is appropriate for field testing.
Human	
Legality	Unlike UAVs, we have not found any rules about USV usage. To avoid any conflict, consult with local authorities (e.g., Coast Guard, government agencies) before starting experiments.
Runtime	The Otter has more endurance than most laptop equipment and researchers. Access to power, shelter, and washrooms is critical for a full day of testing (especially in cold conditions).
Public interaction	Popular with public, so boaters and swimmers frequently approach the Otter. Limited signage provides no warning to nearby public and no built in audio connection to base station. Requires supervision and patience for interrupted experiments.
Sailing right of way	When far from shore and encountering another vessel, it is hard to gauge safe passage. Stopping the Otter provides the safest way to let the other vessel pass.

[10] J. Pinto, P. S. Dias, R. Martins, J. Fortuna, E. Marques, and J. Sousa, "The LSTS toolchain for networked vehicle systems," in *MTS/IEEE OCEANS Conference & Exposition*, 2013, pp. 1–9.

[11] J. Fonseca, M. Aguiar, J. B. de Sousa, and K. H. Johansson, "Algal bloom front tracking using an unmanned surface vehicle: Numerical experiments based on baltic sea data," in *IEEE OCEANS Conference & Exposition*. IEEE, 2021, pp. 1–7.

[12] T. Paine and M. Benjamin, "An ensemble of online estimation methods for one degree-of-freedom models of unmanned surface vehicles; applied theory and preliminary field results with eight vehicles," in *International Conference on Intelligent Robots and Systems (IROS)*, October 2023.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*. Kobe, Japan, 2009.

[14] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[15] M. R. Benjamin, H. Schmidt, P. M. Newman, and J. J. Leonard, "Nested autonomy for unmanned marine vehicles with MOOS-IvP," *Journal of Field Robotics*, vol. 27, no. 6, pp. 834–875, 2010.

[16] M. R. Cooper, S. R. Button, and T. M. C. Sears, "OtterROS: A ROS 2 package for advanced robotics with the Otter USV from Maritime Robotics," https://github.com/offroad-robotics/otter_ros.

[17] SEMU Consulting, "Python library aimed primarily at the subset of the NMEA 0183 v4 protocol for parsing of GNSS messages." 2021. [Online]. Available: https://github.com/semuconsulting/pynmeagps

[18] M. R. Cooper, "Data-driven nonlinear model predictive control of an autonomous uncrewed surface vessel," M.A.Sc. Thesis, Queen's University, 2024, [Online]. Available: https://hdl.handle.net/1974/32811.

[19] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[20] NVIDIA, "JetPack SDK," https://developer.nvidia.com/embedded/jetpack, (accessed Feb 23, 2024).

[21] Offroad Robotics, "OtterMods: Parts, drawings, and models for the Otter USV," https://github.com/offroad-robotics/otter_mods.

[22] T. M. C. Sears, M. R. Cooper, and J. A. Marshall, "Mapping waves with an uncrewed surface vessel via Gaussian process regression," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 5214–5220.