Remote Control of a BlueROV using Underwater Acoustic Transmissions

Nicolas Ferraresso
Dept. of Information Engineering
University of Padova
Padova, Italy
nicolas.ferraresso@studenti.unipd.it

Davide Costa
Dept. of Information Engineering
University of Padova
Padova, Italy
costadavid@dei.unipd.it

Damiano Varagnolo
Dept. of Information Engineering
University of Padova
Padova, Italy
damiano.varagnolo@unipd.it
and *Dept. of Engineering Cybernetics*
NTNU
Gløshaugen, Trondheim, Norway
damiano.varagnolo@ntnu.no

Filippo Campagnaro
Dept. of Information Engineering
University of Padova
Padova, Italy
filippo.campagnaro@unipd.it

Michele Zorzi
Dept. of Information Engineering
University of Padova
Padova, Italy
zorzi@dei.unipd.it

***Abstract*—Underwater robotic systems rely on acoustic communication due to the severe attenuation of radio-frequency signals in water, but the limited bandwidth, high latency, and strong channel variability of acoustic links significantly constrain teleoperation. This paper presents a ROS 2-based teleoperation architecture for control of a BlueROV2 Heavy platform over low-data-rate underwater acoustic channels.**

The proposed system integrates software-defined acoustic modems with the `rmw_desert` middleware, achieving interoperability between ROS 2 nodes and underwater acoustic hardware without requiring modifications to the vehicle platform. Two modulation schemes, FHSS and BPSK, are experimentally evaluated across sixteen configurations in both a controlled laboratory tank and a shallow-water river deployment at approximately $85\,\mathrm{m}$. Two experimental setups are used: one optimized for packet delivery rate assessment using high packet counts, and one providing sub-millisecond clock synchronization via hardware timing modules for latency and jitter evaluation. Performance is assessed in terms of packet delivery rate, end-to-end latency, and jitter using timestamped logs and bootstrap confidence intervals.

BPSK consistently achieves lower end-to-end latency than FHSS, with a median reduction of approximately $700\,\mathrm{ms}$ to $800\,\mathrm{ms}$. Both schemes exhibit low and comparable jitter, with median values below $100\,\mathrm{ms}$ across all configurations, indicating stable timing under the evaluated command rates. All configurations exhibit multi-second end-to-end latency, reflecting fundamental constraints of the acoustic channel rather than system integration. These results demonstrate the feasibility of ROS 2-based acoustic teleoperation in real-world conditions and identify latency as the primary limiting factor in performance, providing guidance for the design of underwater communication stacks in bandwidth-constrained robotic systems.

***Index Terms*—underwater acoustic teleoperation, ROS 2, BlueROV2, software-defined modem, packet delivery rate, underwater robotics, DESERT Underwater**

I. Introduction

Underwater robotic platforms are increasingly deployed in missions that require operation in confined, cluttered, or hazardous environments, including marine environmental monitoring, offshore infrastructure inspection, scientific exploration, and defense operations. Among commercially available systems, the BlueROV2 [1] has emerged as a widely adopted platform due to its open-source architecture, modular design, and full six-degrees-of-freedom (6-DOF) maneuverability, making it a common choice in both academic research and industrial contexts.

Conventional remotely operated vehicles (ROVs) depend on a physical tether to ensure high-bandwidth and low-latency communication with a surface operator. Although this method is functional, tethered operation introduces significant operational limitations. The tether mechanically restricts maneuverability, generates hydrodynamic drag, and presents hazards in environments populated with obstacles such as submerged structures. Moreover, deployment and recovery of long umbilical cables require specialized equipment and trained staff, increasing logistical complexity and operational cost. These limitations motivate the development of a wireless underwater teleoperation system.

Replacing the tether requires a reliable underwater wireless communication link. Radio-frequency signals attenuate extremely rapidly in seawater, limiting practical communication ranges to only a few centimeters even at low frequencies [2]. Optical communication can provide high data rates, but it requires precise alignment and degrades rapidly in turbid water conditions [3]. Acoustic communication remains the

only mature technology capable of sustaining underwater wireless links over distances of hundreds of meters [4], [5]. However, underwater acoustic channels impose severe constraints, including very limited bandwidth (typically from a few hundred to a few thousand bits per second), long propagation delays, high packet error rates caused by multipath interference and ambient noise, and strongly time-varying channel characteristics [6]. These properties make acoustic teleoperation a significant engineering challenge.

Existing commercial acoustic modem solutions further limit accessibility. Such systems are typically proprietary, costly, and poorly integrated with open-source robotic frameworks such as ROS 2 [7]. Their closed internal signal processing pipelines prevent customization of physical-layer parameters and communication protocols, restricting experimental flexibility and hindering reproducibility.

This paper presents a complete low-cost acoustic teleoperation system for the BlueROV2. The proposed architecture integrates ROS 2 with the DESERT Underwater framework [8] through a custom middleware layer, `rmw_desert` [9], enabling unmodified ROS 2 application code to communicate transparently over an underwater acoustic channel. The acoustic hardware is based on software-defined Subsea Modem (SuM) [10], [11], whose fully accessible software stack allows direct customization of modulation schemes and communication parameters.

The end-to-end control pipeline originates at a surface station equipped with an Xbox controller. Operator inputs are processed by a custom C++ ROS 2 control node, encoded into compact integer-array packets to minimize bandwidth usage, and transmitted acoustically via the `rmw_desert`/DESERT stack to the underwater vehicle. On the receiving side, a Python-based bridge decodes command packets and forwards them either to a Gazebo simulation environment [12], [13] or to the physical BlueROV2 through MAVLink/MAVROS [14]. This architecture preserves standard ROS 2 communication semantics while transparently replacing the transport layer with an acoustic link.

The system was first developed and validated through progressively realistic configurations, then quantitatively evaluated in physical deployments:

1) fully simulated environment, with both the acoustic channel (DESERT) and the BlueROV2 (Gazebo) simulated: used for functional validation of middleware integration;
2) hardware-in-the-loop configuration using Gazebo and two SuM separated by $10\,\mathrm{m}$: used to validate the command translation pipeline prior to field deployment;
3) short-distance baseline test with modems separated by $60\,\mathrm{cm}$, while the BlueROV2 operates out of water: quantitative evaluation under controlled conditions;
4) real-world deployment with transducers separated by $85\,\mathrm{m}$: quantitative evaluation under realistic field conditions.

Quantitative performance metrics are reported for configurations 3 and 4. Configurations 1 and 2 were used exclusively for functional validation, no performance metrics are reported from these stages.

In this study, two physical-layer implementations are compared. The first, hereafter denoted **FHSS**, employs a frequency hopping spread spectrum scheme with FSK modulation, inspired by the JANUS framing structure (STANAG 4748) [15], but configured with non-standard parameters to match the deployed transducer characteristics. The second, hereafter denoted **BPSK**, is a binary phase-shift keying scheme built upon the liquid-dsp framework [16], using a rate $\frac{1}{3}$ convolutional forward error correction (FEC). Performance is evaluated across two deployment scenarios in terms of packet delivery rate, end-to-end command latency, jitter, and effective teleoperation responsiveness.

The main contributions of this work are:

- the design and implementation of a low-cost, ROS 2-native acoustic teleoperation architecture for underwater vehicles;
- an end-to-end experimental validation of acoustic teleoperation at $85\,\mathrm{m}$ in a real river environment;
- a comparative evaluation of FHSS and BPSK modulation schemes within the same use case.

The remainder of this paper is organized as follows. Section II reviews related work on underwater acoustic communication and ROS 2-based middleware architectures. Section III describes the proposed system architecture. Section IV details the experimental setup and methodology. Section V presents the results and Section VI discusses the results. Finally, Section VII concludes the paper and outlines future research directions.

II. Background and Related Work

A. Underwater Acoustic Communication

Underwater wireless communication is fundamentally constrained by the physical properties of seawater. Unlike terrestrial environments, where radio-frequency and optical technologies are widely used, these approaches are severely limited underwater. Radio-frequency signals suffer from extremely high attenuation in seawater, restricting their practical communication range to only a few centimeters even at very low frequencies [2]. Optical communication can provide high data rates in controlled conditions, but in open-water environments requires precise transmitter–receiver alignment and its performance rapidly degrades in the presence of turbidity and light scattering [3]. As a result, acoustic signaling has become the dominant technology for underwater wireless communication. Acoustic waves propagate efficiently in water and can support communication over distances ranging from hundreds of meters to several kilometers [4]. This capability makes acoustic communication the only practical general-purpose solution for medium and long-range underwater links.

Despite this advantage, underwater acoustic channels present several challenges. The speed of sound in water is approximately $1500\,\mathrm{m/s}$, introducing significant propagation

delays even at moderate distances. Shallow-water environments typically exhibit strong multipath propagation due to reflections from the surface, seabed, and nearby structures. These reflections generate time-dispersed signal arrivals, leading to inter-symbol interference and limiting achievable symbol rates. Consequently, practical underwater acoustic systems often operate at data rates ranging from a few hundred to a few thousand bits per second, with packet error rates significantly higher than those observed in terrestrial wireless networks [2].

A comparative analysis of underwater communication technologies (acoustic, optical, and radio-frequency) [6] shows that acoustic communication remains the only viable solution for most applications requiring ranges beyond a few meters.

Historically, research in underwater acoustic communication has relied on proprietary commercial modem platforms that are costly and closed-source. This lack of openness has been identified as a barrier to reproducible experimentation and rapid innovation in the field [17]. To address this limitation, recent efforts have explored software-defined modem architectures that expose programmable interfaces and allow researchers to modify modulation schemes and protocol parameters.

The SuM platform adopted in this work follows this paradigm [10], [11]. The modem exposes a programmable TCP interface and supports configurable modulation schemes implemented entirely in software. This flexibility enables experimentation with different physical-layer configurations without hardware modifications.

In this study, two physical-layer configurations of the SuM platform are evaluated: an FHSS-based scheme inspired by the JANUS standard (STANAG 4748) [15], and BPSK. Further details on their configuration are provided in Section IV-B.

B. ROS 2 Middleware for Underwater Robotics

ROS 2 [7] has become the de facto standard framework for modular robotic systems. Its communication architecture is based on a publish–subscribe model implemented through the Data Distribution Service (DDS), supporting decoupled interactions among sensors, controllers, planners, and visualization tools.

However, DDS was originally designed for high-bandwidth, low-latency networks such as Ethernet or Wi-Fi. Its discovery mechanisms, message framing, and quality-of-service negotiation generate overhead that becomes prohibitive in extremely constrained environments such as underwater acoustic channels. Standard DDS implementations therefore cannot operate efficiently under severe bandwidth and latency limitations.

ROS 2 mitigates this limitation through the ROS Middleware (RMW) abstraction layer, which decouples the application interface from the underlying transport implementation. By implementing a custom RMW backend, it is possible to replace DDS entirely while preserving the standard ROS 2 programming model.

In this work, the communication infrastructure is provided by the DESERT Underwater framework [8]. Built on NS-2 and NS-Miracle, DESERT implements a complete underwater networking stack, including physical-layer models and higher-layer protocols. A key feature of the framework is that simulation and real hardware deployments share the same codebase, enabling seamless transition from simulation to physical modem testing through configuration changes.

The `rmw_desert` middleware [9] implements the ROS 2 RMW interface to convert ROS 2 messages into compact, compressed packets compatible with underwater acoustic modems, with or without the DESERT stack. It serializes and compresses messages into packets suitable for low-bandwidth transmission and forwards them to the modem interface. On the receiving side, the inverse process decompresses and deserializes the packets transparently. This allows application nodes to work without needing to know about the communication method used. The DESERT stack is particularly well suited for multi-hop and multi-node underwater network scenarios, where it supports distributed communication over acoustic links.

Although `rmw_desert` has been demonstrated in prior communication experiments [9], its integration into a complete end-to-end ROV teleoperation system has not been reported. The architecture proposed in this work integrates the full control pipeline, from the human operator interface and ROS 2 control nodes to the acoustic middleware, software-defined modems, and vehicle actuation. The system is experimentally validated in a real deployment with two modems separated by $85\,\mathrm{m}$ and a BlueROV2 operating in an underwater environment. To the best of the authors' knowledge, no prior work has reported end-to-end latency and jitter measurements for ROS 2-integrated acoustic teleoperation of an ROV under real field conditions.

III. System Architecture

The underwater acoustic channel provides only a few hundred bits per second of usable data rate.

The system architecture is designed under a fundamental constraint: The underwater acoustic channel provides only a few hundred bits per second of usable data rate. Consequently, every component in the communication pipeline must operate efficiently within this limitation. The system is organized into four stages: the surface control node, the `rmw_desert` middleware layer, a Python bridge, and the vehicle-side software stack, as illustrated in Fig. 1.

Two design decisions are central to bandwidth efficiency. First, all thruster commands are aggregated into a single integer-encoded message, guaranteeing atomic actuation and reducing packet size. Second, an event-triggered transmission policy ensures that a new command is sent only when the thrust vector changes beyond a configurable threshold, avoiding unnecessary channel utilization during steady-state operation.

A. Surface Control Node

The surface system is implemented as a ROS 2 process in C++ and deployed inside a Docker container based on the official ROS 2 image `osrf/ros:kilted-desktop-full`,

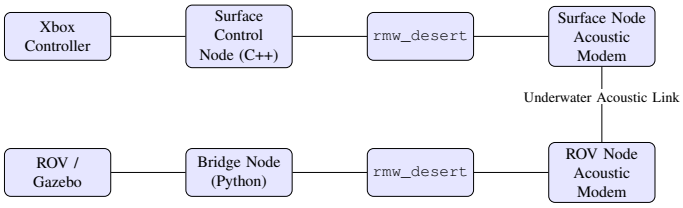

Fig. 1. End-to-end data flow across system components.

which provides a preconfigured environment with the full ROS 2 desktop installation and development tools, ensuring reproducibility across deployments. A controller reader thread polls the Xbox controller via the Simple Direct-Media Layer (SDL2) library [18] and normalizes axis inputs using dead-zone filtering. A publisher node computes a thrust vector by projecting controller inputs through a configurable keymap matrix and encodes the result as a vector of integer values before transmission. Commands are published on a single ROS 2 topic using the standard message type `std_msgs/msg/Int32MultiArray`, an array of 32-bit integers. The resulting message is intercepted by `rmw_desert` before reaching the network stack.

B. Acoustic Middleware Layer

`rmw_desert` [9] replaces the standard DDS transport by serializing and compressing ROS 2 messages using CBOR encoding with a 6-byte packetization header, forwarding them either through the DESERT Underwater framework [8] or directly to the SuM platform [10], [11] that provides a programmable, software-defined physical layer supporting both modulation schemes evaluated in this work.

C. Python Bridge

Since only one RMW implementation can be loaded per process, a Python bridge decouples the `rmw_desert` transport context from the standard ROS 2 middleware used on the vehicle side. Two processes communicate over a local TCP socket: the first subscribes to incoming commands under `rmw_desert` and serializes them, while the second receives and converts them into `mavros_msgs/msg/OverrideRCIn` messages, a MAVROS specific message type used to override RC input channels. MAVROS then translates these messages into MAVLink commands for the flight controller. All commands are timestamped at reception and logged for latency analysis.

IV. Experimental Setup

This section describes the experimental environments, hardware configuration, test procedures, and evaluation metrics used to assess the proposed teleoperation system. Two physical deployment environments and two acoustic modulation schemes were evaluated across two independent experimental setups, each comprising two trials per combination, yielding sixteen experimental runs in total. Identical control inputs were applied across all conditions to ensure comparability.

A. Experimental Environments

Two distinct deployment scenarios were considered for each setup, selected to span a range of channel conditions from controlled baseline to realistic field deployment.

The first scenario was conducted in a controlled indoor water tank, while the BlueROV2 Heavy platform remained outside the water. The two acoustic transducers were submerged with a separation of approximately $60\,\mathrm{cm}$. This intentionally short range does not represent an operational deployment distance, but rather serves to validate the full software stack and communication pipeline while eliminating acoustic propagation effects. At this separation, near-field effects introduce multipath interference that is not representative of far-field operation.

The second scenario was conducted in a natural shallow-water environment (Piovego Canal, Padova, Italy), with a water depth estimated between $1.5\,\mathrm{m}$ and $2\,\mathrm{m}$. The two acoustic transducers were positioned at an estimated distance of approximately $85\,\mathrm{m}$, based on GPS measurements (see Figure 2). They were submerged at a depth of approximately $1\,\mathrm{m}$ and were not physically attached to the vehicle.

The riverbed consists primarily of irregular sediment and debris, resulting in strong bottom reverberation and a highly reflective acoustic environment. The water depth is low, further constraining the acoustic propagation geometry and increasing multipath severity. This setting represents a challenging and operationally realistic deployment scenario.

Fig. 2. On the left, a map showing the positions of the two modems (surface modem on the left, vehicle-side modem on the right). On the right, the actual deployment.

B. Hardware and Software Configuration

Experiments were conducted using an unmodified BlueROV2 Heavy platform. Acoustic communication was provided by two software-defined SuM [10] equipped with custom transducers having a carrier frequency of $32\,\mathrm{kHz}$.

The surface control system, including the ROS 2 node and `rmw_desert` middleware, was deployed inside a Docker container to ensure environment reproducibility and consistent runtime configuration across all trials. In the evaluated configuration, the thrust vector is encoded as a `std_msgs/msg/Int32MultiArray` of six integer values. The `rmw_desert` middleware applies CBOR serialization and adds its packetization header, resulting in each transmitted command occupying 13 or 14 bytes on the acoustic link, depending on the specific thrust vector values.

Clock synchronization between the surface and vehicle-side systems differed between the two experimental setups. In the

first setup, synchronization was achieved using an external remote NTP server with a measured clock offset below $50\,\mathrm{ms}$ between the devices, as determined by timestamp comparison. In the second setup, synchronization was provided by two TM2500C [19] hardware timing modules, disciplined via NTP over a dedicated LAN connection. The theoretical offset under this configuration is below $1\,\mathrm{ms}$ however, direct measurement at this precision was not available.

Link-layer retransmissions were disabled for all experiments. Consequently, packet losses directly reflect channel and link-layer reliability without the confounding effect of recovery mechanisms. This choice also represents a realistic constraint for time-critical teleoperation, where retransmitted commands may arrive stale.

Physical experiments were conducted by bypassing the DESERT network stack to eliminate software-induced overhead and interfacing `rmw_desert` directly with the SuM.

The configuration of the surface control node defines both the polling interval and the transmission policy. In the evaluated setup, joystick inputs are sampled at a fixed interval of $0.1\,\mathrm{s}$ and transmitted only when a change in the command is detected.

Consequently, the measured end-to-end latency includes an additional sampling-induced delay, uniformly distributed in the range $[0, 100]$ ms, with an expected value of $50\,\mathrm{ms}$. This contribution is independent of the acoustic channel and should be considered when interpreting latency results.

C. Acoustic Modulation Configurations

Two physical-layer configurations implemented on the SuM platform are evaluated. The first, FHSS, is configured with a $32\,\mathrm{kHz}$ carrier frequency, $10\,\mathrm{kHz}$ bandwidth, and $192\,\mathrm{kHz}$ sampling rate. Through its frequency hopping scheme and a $\frac{1}{2}$ FEC rate, the application-level bit rate is approximately $192\,\mathrm{bit/s}$. The carrier frequency was selected to coincide with the peak transmit voltage response (TVR) and receive voltage response (RVR) of the deployed transducer, while the bandwidth of $\pm 5\,\mathrm{kHz}$ corresponds to approximately $5\,\%$ deviation from the transducer's resonance peak.

The second, BPSK, operates at a $32\,\mathrm{kHz}$ carrier frequency, $192\,\mathrm{kHz}$ sampling rate, and $1.2\,\mathrm{kHz}$ bandwidth giving an application level bit rate of approximately $400\,\mathrm{bit/s}$ after encoding.

D. Trial Protocol

Two independent experimental setups were used, differing in network topology and clock synchronization.

In the *reliability setup*, all devices were connected via a shared mobile hotspot and synchronized to a remote NTP server, resulting in a clock offset bound of approximately $50\,\mathrm{ms}$. While this is insufficient for accurate latency measurement, it does not affect packet delivery rate (PDR), which depends only on matching packet identifiers. This setup is therefore used exclusively for PDR evaluation.

In the *latency and jitter setup*, devices were connected through a dedicated wired LAN with TM2500C local NTP references, achieving sub-millisecond clock synchronization, enabling accurate latency and jitter analysis.

As shown in Table I, the reliability setup includes substantially higher packet counts, providing greater statistical power for PDR estimation.

In contrast, the latency and jitter setup experienced interference from vessel traffic during river deployment, reducing packet delivery independently of modulation performance. For this reason, PDR results from this setup are not considered representative of link reliability. Laboratory PDR results from the latency setup, which were not affected by interference, are reported separately as a consistency check.

Within each setup, two trials were conducted for every environment–modulation combination, and are treated as independent observations.

TABLE I
TOTAL NUMBER OF TRANSMITTED COMMANDS PER EXPERIMENTAL CONDITION, AGGREGATED ACROSS THE TWO TRIALS OF EACH SETUP.

Environment	Scheme	Setup 1	Setup 2
Laboratory	BPSK	505	216
Laboratory	FHSS	419	205
River	BPSK	211	142
River	FHSS	237	146

E. Control Input Sequence

To ensure repeatability, a predefined joystick input sequence was applied consistently across all modulation schemes and environments. The sequence activates one control axis at a time, maintaining each direction for approximately $10\,\mathrm{s}$ before returning to the neutral state. Specifically, the inputs follow a cyclic pattern: Neutral $\rightarrow$ Up $\rightarrow$ Neutral $\rightarrow$ Right $\rightarrow$ Neutral $\rightarrow$ Down $\rightarrow$ Neutral $\rightarrow$ Left.

The use of a fixed sequence eliminates operator variability as a biasing factor and enables direct comparison of communication metrics across conditions. A qualitative manual joystick test with the BlueROV2 in water was additionally performed to assess real-world usability but no quantitative metrics were collected during this test due to variability in human input.

F. Performance Metrics

Performance was evaluated by recording timestamps at two points in the communication chain:

- the moment a joystick command was received by the surface control node (t^{tx});
- the moment the decoded command was published on the vehicle-side ROS 2 topic (t^{rx}).

All timestamps were logged on both sides of the link and stored as CSV files. The following metrics were computed independently for each of the eight experimental configurations:

- Packet Delivery Rate (PDR);
- End-to-end latency;
- Jitter.

PDR is reported from the reliability setup exclusively. End-to-end latency and jitter are reported from the latency

TABLE II
RELIABILITY, LATENCY, AND JITTER METRICS ACROSS ALL EXPERIMENTAL CONFIGURATIONS. T1/T2 = TRIAL 1/TRIAL 2.

Config	Lab T1 FHSS	Lab T2 FHSS	Lab T1 BPSK	Lab T2 BPSK	River T1 FHSS	River T2 FHSS	River T1 BPSK	River T2 BPSK
Reliability Setup								
N sent	195	224	135	370	115	122	83	128
N matched	195	222	135	369	98	113	78	117
PDR (%)	100.0	99.1	100.0	99.7	85.2	92.6	94.0	91.4
Latency & Jitter Setup								
Lat mean (ms)	2728.0	2721.7	1983.1	2031.5	2858.0	2789.9	2108.0	2045.8
Lat std (ms)	43.6	55.4	53.0	46.5	144.5	38.1	52.1	51.4
CI mean 95%	[2719.7, 2736.3]	[2710.8, 2732.6]	[1972.6, 1993.7]	[2022.7, 2040.2]	[2826.3, 2893.3]	[2775.9, 2804.5]	[2095.5, 2120.6]	[2021.2, 2078.7]
Lat median (ms)	2726.0	2720.0	1984.0	2031.0	2809.0	2787.0	2111.5	2037.0
CI median 95%	[2717.0, 2735.0]	[2701.0, 2731.0]	[1968.0, 1995.0]	[2023.0, 2045.0]	[2801.5, 2829.5]	[2770.0, 2809.0]	[2095.0, 2126.0]	[2009.0, 2057.0]
Jitter median (ms)	28.0	72.0	30.5	30.0	27.0	31.0	30.0	35.0
Jitter max (ms)	131.0	180.0	176.0	220.0	365.0	111.0	171.0	172.0

and jitter setup, which provides the sub-millisecond clock synchronisation required for reliable temporal measurement.

G. Simulation and Hardware-in-the-Loop Validation

During development, preliminary validation was conducted using both full simulation and Hardware-in-the-Loop (HIL) configurations [20]. In simulation, the acoustic channel was emulated using the DESERT framework to verify middleware integration and message routing. In the HIL configuration, acoustic modems operated physically while ROV dynamics were simulated in Gazebo, enabling validation of the command translation pipeline prior to field deployment. These stages were used exclusively for functional validation. No quantitative performance metrics reported in this paper were collected during simulation or HIL experiments. All results derive solely from the physical deployments described above.

V. EXPERIMENTAL RESULTS

This section presents the quantitative results from the physical experiments described in Section IV. PDR is reported from the reliability setup, while end-to-end latency and jitter are obtained from the latency and jitter setup, as justified in Sections IV-B and IV-D. Results are reported for all eight experimental configurations, with a summary in Table II. All scripts used for analysis and figure generation are available in the project repository [21].

A. Packet Delivery Rate

PDR is defined as

$$\mathrm{PDR} = \frac{N_{\mathrm{rx}}}{N_{\mathrm{tx}}} \times 100\,\% \tag{1}$$

where N_{rx} and N_{tx} denote the number of received and transmitted commands, respectively. With retransmissions disabled, PDR directly reflects channel reliability.

In the laboratory environment, both schemes achieved near-perfect delivery. BPSK reached $100\,\%$ (Trial 1) and $99.7\,\%$ (Trial 2), while FHSS achieved $100\,\%$ (Trial 1) and $99.1\,\%$ (Trial 2). This performance is consistent with the short propagation distance ($60\,\mathrm{cm}$) and absence of interference.

As a consistency check, laboratory PDR from the latency and jitter setup was also examined. The overall PDR was approximately $97.3\,\%$, indicating results consistent with those observed in the reliability setup and suggesting that the network infrastructure does not significantly impact delivery.

In the river deployment, performance degraded for both schemes. BPSK achieved $94\,\%$ (Trial 1) and $91.4\,\%$ (Trial 2). FHSS exhibited greater variability, with $85.2\,\%$ (Trial 1) and $92.6\,\%$ (Trial 2). This degradation is attributed to the shallow-water multipath environment described in Section IV. Results are shown in Fig. 3.

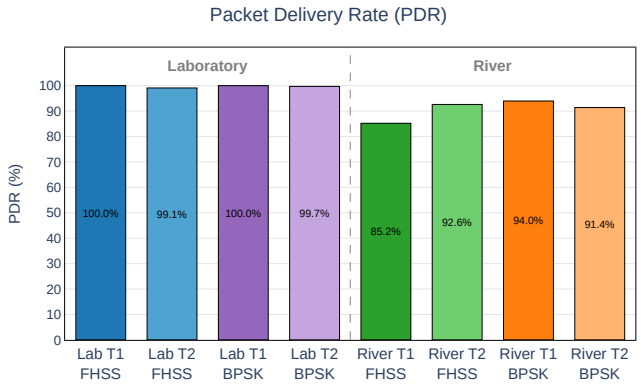

Fig. 3. Packet Delivery Rate per configuration (reliability setup).

B. End-to-End Latency

The end-to-end latency for the k-th command is defined as

$$L_k = t_k^{\mathrm{rx}} - t_k^{\mathrm{tx}} \tag{2}$$

where t_k^{tx} and t_k^{rx} are the transmission and reception timestamps, respectively. Clock synchronization is described in Section IV-B.

At the river deployment distance ($85\,\mathrm{m}$), the acoustic propagation delay is approximately $57\,\mathrm{ms}$ (assuming $c = 1500\,\mathrm{m/s}$), which is negligible relative to the observed multi-second latencies. Due to the skewed nature of latency distributions, the median is used as the primary statistic.

In the laboratory, BPSK exhibited median latencies of $1984\,\mathrm{ms}$ (Trial 1, 95% CI [1968, 1995]) and $2031\,\mathrm{ms}$ (Trial 2, 95% CI [2023, 2045]). FHSS showed consistently higher latency, with medians of $2726\,\mathrm{ms}$ (Trial 1, 95% CI [2717, 2735]) and $2720\,\mathrm{ms}$ (Trial 2, 95% CI [2701, 2731]).

In the river deployment, latency remained comparable to laboratory values. BPSK achieved medians of 2111.5 ms (Trial 1, 95% CI [2095, 2126]) and 2037 ms (Trial 2, 95% CI [2009, 2057]). FHSS recorded 2809 ms (Trial 1, 95% CI [2801.5, 2829.5]) and 2787 ms (Trial 2, 95% CI [2770, 2809]).

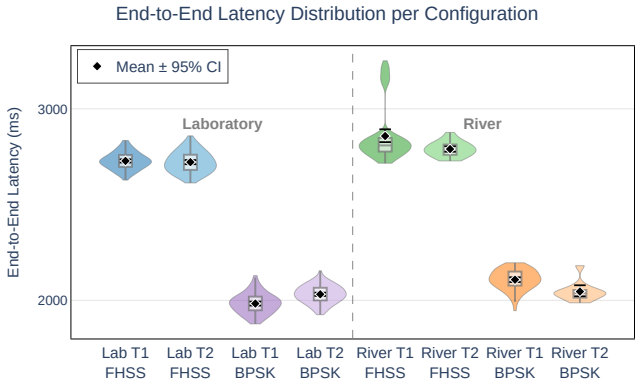

Fig. 4. Latency distributions per configuration (latency and jitter setup).

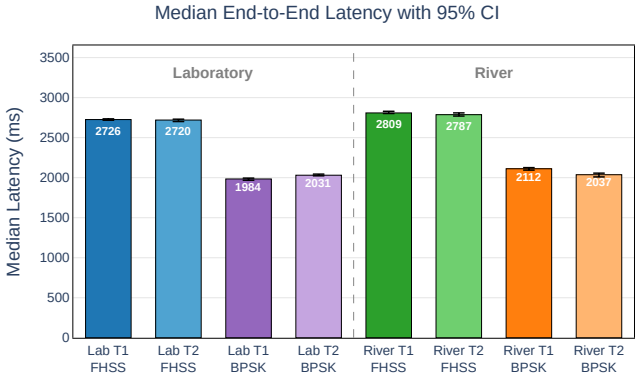

Fig. 5. Median latency with 95 % confidence intervals (latency and jitter setup).

C. Jitter

Jitter is defined as the mean absolute difference between consecutive delays [22]:

$$J = \frac{1}{N-1} \sum_{k=2}^{N} |L_k - L_{k-1}| \tag{3}$$

where L_k is the latency of the k-th successfully received packet, and k is indexed over successfully received packets only.

In the laboratory, both schemes exhibited low jitter. BPSK recorded medians of 30.5 ms and 30 ms, with maxima of 176 ms and 220 ms. FHSS recorded medians of 28 ms and 72 ms, with maxima of 131 ms and 180 ms.

In the river deployment, jitter remained low. BPSK recorded medians of 30 ms and 35 ms, with maxima of 171 ms and 172 ms. FHSS recorded medians of 27 ms and 31 ms, with a higher maximum of 365 ms in Trial 1, consistent with the increased latency variability observed in that configuration.

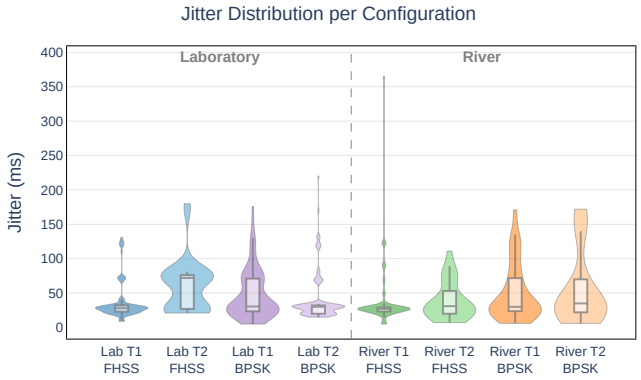

Fig. 6. Jitter distributions per configuration (latency and jitter setup).

VI. Discussion

This section interprets the experimental results presented in Section V in terms of modulation behavior, environmental effects, inter-trial variability, and implications for acoustic teleoperation. Overall, modulation scheme selection has a dominant impact on latency: BPSK consistently achieves lower end-to-end delay than FHSS across both controlled and field environments, while both schemes exhibit comparably low jitter under the improved clock synchronization of the latency and jitter setup.

A. BPSK vs. FHSS: Modulation Scheme Comparison

The latency difference between BPSK and FHSS follows directly from their physical-layer design. FHSS prioritizes robustness via frequency hopping and a $\frac{1}{2}$ FEC rate, at the cost of reduced throughput (192 bit/s). BPSK, employing a rate $\frac{1}{3}$ convolutional FEC, provide higher throughput (400 bit/s).

Including framing overhead (12 B for BPSK and 10 B for FHSS), the transmission time for a 14 B payload is approximately 0.52 s (BPSK) and 1 s (FHSS). This explains the observed latency gap of approximately 700 ms to 800 ms across all conditions. The acoustic propagation delay of approximately 57 ms at the 85 m river deployment distance is comparatively small relative to total latency, but is consistent with the observed increase in median latency between laboratory and river conditions for both schemes.

Jitter analysis shows consistently low values for both schemes (27 ms to 72 ms). This indicates that, under sub-millisecond synchronization, timing variability is primarily driven by channel effects rather than modulation-specific behavior.

In terms of reliability, the near-perfect PDR achieved by both schemes under laboratory conditions suggests that the packet loss observed in the river deployment is largely driven by the acoustic channel environment.

B. Operational Implications for Acoustic Teleoperation

Observed latencies range from $1984\,\mathrm{ms}$ to $2809\,\mathrm{ms}$, which is unsuitable for direct closed-loop teleoperation tasks such as station keeping or obstacle avoidance. The system therefore requires a move-and-wait interaction model or rely on onboard autonomy for fine control. Within these constraints, BPSK is preferable due to its consistently lower latency.

Jitter remains low across all configurations, with median values below $100\,\mathrm{ms}$ and maxima below $365\,\mathrm{ms}$. At the evaluated command rate, this variability does not significantly affect actuation regularity, indicating that jitter is not a limiting factor under these conditions.

With retransmissions disabled, each packet loss corresponds to a missed command execution. At the configured rate (one command every $10\,\mathrm{s}$), the worst-case river FHSS condition ($85.2\,\%$ PDR) implies that roughly one in seven updates is lost. This may be acceptable for coarse navigation but is unsuitable for precision control. Enabling retransmissions would improve reliability but increase and destabilize latency, introducing a fundamental trade-off.

Even under nominally identical configurations, inter-trial variability is observed without consistent directional trends. This is consistent with non-stationary channel behavior driven by temperature gradients, flow velocity, and surface agitation, which modify sound propagation conditions over time. As a result, single-trial measurements are insufficient to characterize performance, and variability must be explicitly considered.

C. Limitations

Several limitations of this study should be considered.

The number of trials per condition was limited to two, which is insufficient to fully characterize the statistical distribution of inter-trial variability. While variability is clearly demonstrated, its magnitude and structure cannot be robustly quantified.

The system operates as a unidirectional open-loop teleoperation architecture. Commands are transmitted without an acoustic feedback channel for the vehicle state, limiting operator awareness and constraining the complexity of feasible tasks. This architecture reflects a simplified control model and does not capture the challenges of fully bidirectional acoustic interaction.

ArduSub's native RC override persistence causes the vehicle to hold the last received thrust command indefinitely in the absence of new packets. As a result, packet loss does not produce immediate loss of actuation, but may cause unintended motion. This behavior partially decouples the reported communication metrics from actual vehicle control fidelity: PDR and latency characterize the acoustic channel but do not directly reflect degradation in trajectory execution.

Physical-layer parameters, including carrier frequency, bandwidth, and FEC configuration, were selected based on hardware constraints rather than systematic optimization. Their impact on the performance across varying channel conditions remains unexplored.

Finally, experiments used an event-triggered command sequence with a $0.1\,\mathrm{s}$ polling interval. Such a high command update rate is generally unsuitable for real underwater deployment, as it increases the likelihood of redundant or unintended commands due to minor controller fluctuations and leads to unnecessary channel load. The chosen rate was instead used to minimize control latency and avoid the additional delay that would result from a lower-rate sampling strategy (e.g., $1\,\mathrm{s}$ polling), which would introduce a fixed quantization delay in command updates.

VII. Conclusions and Future Work

This paper presented a ROS 2-based underwater teleoperation architecture enabling control of a BlueROV2 Heavy platform over low-bandwidth acoustic communication. The system integrates software-defined acoustic modems with a custom middleware layer, `rmw_desert`, allowing direct interaction between ROS 2 nodes and underwater acoustic hardware without requiring modifications to the vehicle platform.

Experimental validation was conducted across sixteen configurations spanning two environments (laboratory tank and shallow-water river canal), two modulation schemes (BPSK and FHSS), two independent trials per condition, and two dedicated experimental setups optimized respectively for reliability and latency measurement, for a total of 2081 transmitted commands. The system successfully demonstrated acoustic teleoperation at approximately $85\,\mathrm{m}$ range in a real-world river environment with the vehicle fully operational.

The evaluation reveals three main findings.

First, BPSK consistently achieves lower end-to-end latency than FHSS across all configurations and environments, with median reductions of approximately $700\,\mathrm{ms}$ to $800\,\mathrm{ms}$.

Second, both schemes exhibit comparably low median jitter across all conditions ($27\,\mathrm{ms}$ to $72\,\mathrm{ms}$), indicating that jitter does not represent a limiting factor at the evaluated command rates.

Third, while both schemes achieve near-perfect packet delivery in the laboratory, river performance is degraded by environmental effects. In this setting, BPSK maintains higher and more consistent reliability than FHSS, whose lower-bound performance corresponds to a non-negligible packet loss rate.

A. Future Work

Several directions emerge for extending this work.

a) Real-world vehicle-integrated deployment: A key next step is the integration of the acoustic modem into a fully untethered ROV platform. Onboard deployment would enable evaluation under realistic operational conditions, including vehicle motion, self-noise, and orientation dynamics, providing a more complete evaluation of communication performance in mission-like scenarios.

b) Command timeout and failsafe mechanism: A timeout-based failsafe is not directly applicable under the current event-triggered transmission scheme, as channel silence is indistinguishable from a deliberate command hold. A more robust approach would require a periodic keep-alive mechanism at the communication layer, analogous to the MAVLink `HEARTBEAT`, enabling safe fallback behavior in the event of link loss.

c) Higher-rate control: Future work should investigate system behavior under higher command rates and continuous control inputs, where increased traffic density may expose additional limitations related to buffering, scheduling, and acoustic channel contention. While current results indicate sufficient baseline reliability, scaling the update rate will likely require adjustments in modulation parameters or available bandwidth.

d) Physical-layer optimization: A systematic exploration of physical-layer parameters, including carrier frequency, bandwidth, spreading factor, and FEC could identify configurations better suited to teleoperation workloads. Such optimization should be performed in conjunction with channel characterization to capture environment-dependent trade-offs between latency and reliability.

REFERENCES

[1] *BlueROV2 Technical Specifications*, Blue Robotics. [Online]. Available: https://bluerobotics.com/store/rov/bluerov2/

[2] M. Stojanovic and J. C. Preisig, "Underwater Acoustic Communication Channels: Propagation Models and Statistical Characterization," *IEEE Communications Magazine*, vol. 47, pp. 84–89, Feb 2009.

[3] C. Pontbriand, N. Farr, J. Hansen, J. C. Kinsey, L.-P. Pelletier, J. Ware, and D. Fourie, "Wireless Data Harvesting Using the AUV Sentry and WHOI Optical Modem," in *OCEANS 2015 - MTS/IEEE Washington*, 2015, pp. 1–6.

[4] E. Sozer, M. Stojanovic, and J. Proakis, "Underwater Acoustic Networks," *IEEE Journal of Oceanic Engineering*, vol. 25, no. 1, pp. 72–83, Jan 2000.

[5] D. Cosimo, A. Montanari, D. S. Terracciano, L. Bazzarello, R. Costanzi, F. Campagnaro, and M. Zorzi, "Comparative Analysis of Throughput Maximization Strategies in Underwater Acoustic Networks: Results from At-Sea Experiments," in *IEEE International Workshop on Metrology for the Sea (MetroSea)*, 2024, pp. 52–57.

[6] A. Pal, F. Campagnaro, K. Ashraf, M. R. Rahman, A. Ashok, and H. Guo, "Communication for Underwater Sensor Networks: A Comprehensive Summary," *ACM Transactions on Sensor Networks*, vol. 19, no. 1, Dec 2022.

[7] *ROS 2*, Open Robotics. [Online]. Available: https://www.ros.org/

[8] F. Campagnaro, R. Francescon, F. Guerra, F. Favaro, P. Casari, R. Diamant, and M. Zorzi, "The DESERT Underwater Framework v2: Improved Capabilities and Extension Tools," in *IEEE Third Underwater Communications and Networking Conference (UComms)*, 2016, pp. 1–5.

[9] D. Costa, F. Campagnaro, and M. Zorzi, "Robot Operating System (ROS) Talks Underwater: An Open-Source Communication Middleware to Control Underwater Vehicles," in *IEEE International Workshop on Metrology for the Sea (MetroSea)*, 2025, pp. 123–127.

[10] *Subsea Modem (SuM)*, SubSeaPulse. [Online]. Available: https://modem.subseapulse.com/sum-doc/

[11] A. Montanari, V. Cimino, D. Spinosa, F. Donegà, F. Marin, F. Campagnaro, and M. Zorzi, "PSK Modulation for Underwater Communication and One-Way Travel-Time Ranging with the Low-Cost Subsea Software-Defined Acoustic Modem," in *Proceedings of the 18th International Conference on Underwater Networks & Systems*. New York, NY, USA: Association for Computing Machinery, 2025.

[12] O. Robotics, "Gazebo Simulator," https://gazebosim.org/home, 2023, version 3.0.

[13] Centrale Nantes ROV Team, "BlueROV2 ROS 2 Simulation and Control Framework," https://github.com/CentraleNantesROV/bluerov2.

[14] MAVLink, "MAVROS," https://github.com/mavlink/mavros.

[15] *STANAG 4748: Digital Underwater Signalling Standard for Network Node Discovery & Interoperability*, NATO Std. STANAG 4748, 2017.

[16] J. D. Gaeddert, "liquid-dsp," https://liquidsdr.org/.

[17] F. Campagnaro, F. Steinmetz, and B.-C. Renner, "Survey on Low-Cost Underwater Sensor Networks: From Niche Applications to Everyday Use," *Journal of Marine Science and Engineering*, vol. 11, no. 1, Jan 2023.

[18] SDL Community, "Simple directmedia layer," https://www.libsdl.org/.

[19] TimeMachines Inc., "Gps ntp+ptp network time server with 10mhz output (tm2500)." [Online]. Available: https://timemachinescorp.com/product/gps-ntpptp-network-time-server-10mz-output-tm2500/

[20] N. Ferraresso, "Piloting the BlueROV2 Using an Xbox Controller Over Acoustic Communication via ROS2," Master's thesis, Università degli Studi di Padova, 2024. [Online]. Available: https://hdl.handle.net/20.500.12608/91673

[21] ——, "Acoustic Control for BlueROV2," https://github.com/Magform/acoustic_bluerov2.

[22] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," Internet Engineering Task Force (IETF), RFC 3550, 2003. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3550