Design and Development of Web-Based USV Ground Control Station for Ocean Observation

Ali Musthofa Baharudin[1], Setyawan Ajie Sukarno [2], Mohammad Harry Khomas Saputra [3]
(Received: 3 May 2026 / Revised: 5 May 2026 / Accepted: 11 May 2026 / Available Online: 8 June 2026)

***Abstract*— Ocean observation increasingly relies on Unmanned Surface Vehicles (USVs) for wide-area maritime data collection, yet existing Ground Control Station (GCS) systems remain limited by desktop-based accessibility constraints, high-latency HTTP communication, unoptimized relational databases, and the absence of integrated oceanographic sensor visualization. This study aims to design and develop a web-based GCS that unifies telemetry monitoring, bidirectional remote control, CTD and ADCP sensor visualization, and waypoint-based mission planning within a single browser-based interface. The system was developed using the Rapid Application Development (RAD) methodology, with MQTT and WebSocket adopted for low-latency communication and PostgreSQL with TimescaleDB for time-series data management. Functional testing confirmed that all core features operated correctly. Performance evaluation under 4G-LTE field conditions showed an average command round-trip latency of 178 ms and end-to-end monitoring latency below 250 ms. TimescaleDB hypertable conversion improved write throughput by 2.5×, reduced long-range query latency by 3.75×, and achieved a storage compression ratio of 875×. The proposed architecture provides a replicable reference for web-based USV GCS development across ocean observation applications.**

Keywords—Ground Control Station, MQTT, Ocean Observation, Unmanned Surface Vehicle, WebSocket

**Corresponding Author*: musthofali26@gmail.com

I. INTRODUCTION

Ocean observation has become an increasingly critical component of maritime research and environmental management, including activities such as water quality monitoring, hydrographic surveys, and marine surveillance, all of which require reliable and scalable data acquisition platforms [1][2][3]. The adoption of Unmanned Surface Vehicles (USVs) has emerged as a practical solution to these demands, offering continuous and wide-area data collection with reduced human risk and operational cost [1][3].

As USV deployments grow in scale and complexity, with the global USV market experiencing rapid growth driven by increasing demand across defense, scientific, and commercial maritime sectors [4], the need for capable and flexible Ground Control Station (GCS) systems to support these operations has become increasingly critical. A GCS serves as the central interface for telemetry monitoring, sensor data visualization, and mission control from a remote location [5]. However, most existing GCS implementations remain desktop-based, restricting operator mobility and limiting system access to a single device at a time [6][7] [8]. Widely adopted open-source platforms such as Mission Planner require local installation on dedicated devices [9][10].

This constraint is particularly significant in multi-operator field deployments where situational awareness must be maintained across different locations simultaneously [6][11][12]. Web-based GCS architectures have been proposed as a more accessible alternative, enabling operators to monitor unmanned vehicles from any device through a standard web browser without requiring dedicated software installation [7][11]. Although existing web-based implementations have demonstrated reliable real-time telemetry visualization through WebSocket communication [7][11], they remain limited to unidirectional monitoring and do not support bidirectional remote control, waypoint-based mission management, or integration of domain-specific sensors. Table 1 summarizes a comparative overview of existing GCS systems against the proposed system.

As shown in Table 1, although desktop-based platforms such as Mission Planner (MP) and QGroundControl (QGC) provide comprehensive mission planning and remote-control capabilities, both rely exclusively on MAVLink over serial or UDP connections and store telemetry in flat binary log files without any time-series database backend [9][10]. Furthermore, most systems continue to rely on HTTP for data exchange, which introduces high latency in continuous real-time scenarios due to its stateless, connection-per-request nature [13][14][15].

Comparative studies confirm that MQTT over WebSocket achieves substantially lower and more stable end-to-end latency than HTTP, making it a more appropriate communication protocol for web-based GCS deployments that demand low-latency and bidirectional data exchange [14][15]. Beyond communication limitations, existing USV GCS implementations generally focus on basic navigation parameters such as GPS and IMU data, without integrating oceanographic sensor visualization [16][17].

[A]li Musthofa Baharudin. Department of Automation Engineering, Politeknik Manufaktur Bandung, Bandung, 40135, Indonesia. E-mail: musthofali26@gmail.com
Setyawan Ajie Sukarno. Department of Automation Engineering, Politeknik Manufaktur Bandung, Bandung, 40135, Indonesia. E-mail: ajie@ae.polman-bandung.ac.id
Mohammad Harry Khomas Saputra. Department of Automation Engineering, Politeknik Manufaktur Bandung, Bandung, 40135, Indonesia. E-mail: harry@ae.polman-bandung.ac.id

This is particularly problematic for ocean observation missions, which require the simultaneous monitoring of multiple physical parameters, with Conductivity, Temperature, and Depth (CTD) sensors and Acoustic Doppler Current Profilers (ADCPs) serving as the primary instruments for measuring water column properties and current patterns [18][19]. Prior USV studies have confirmed that existing systems address monitoring and control as isolated functions through desktop or mobile interfaces, without a unified operational platform, and without structured storage of mission data for post-mission analysis [9][12][16][17].

These shortcomings are further compounded by the fact that continuous sensor data streams require efficient storage to support time-range queries, yet conventional relational databases used in existing GCS implementations tend to degrade in write speed and query performance as data volume grows [20][21][22]. Time-series database solutions offer faster write throughput, more efficient time-range queries, and better storage utilization for sequential sensor data [21][22].

To address these limitations, this study proposes a web-based GCS for USV ocean observation that unifies telemetry monitoring, bidirectional control, and oceanographic sensor visualization within a single integrated platform. This work contributes three primary advances. The first contribution is a unified web-based GCS architecture integrating real-time telemetry, bidirectional remote control, CTD and ADCP visualization, and mission planning within a single platform.

The second contribution is the application of MQTT and WebSocket for low-latency bidirectional communication between the USV and the browser-based operator interface. The third contribution is the adoption of TimescaleDB within a web-based USV GCS for ocean observation, an integration not previously reported in existing literature. The findings are expected to serve as a practical and replicable reference for future web-based GCS development across various USV platforms and ocean observation applications.

TABLE 1.
COMPARISON OF THE PROPOSED SYSTEM WITH EXISTING GCS IMPLEMENTATIONS

Feature	This Work	MP*	QGC [10]	Eka [7]	Ardi [11]	Liang [8]	Demetillo [16]	Sukarno [17]	Setiawan [9]
Web-based Platform	✓	✗	✗	✓	✓	✗	✗	✗	✗
IoT Pub/Sub Protocol	✓	✗	✗	**P**	**P**	✗	✗	**P**	✗
Real-time Telemetry	✓	✓	✓	✓	✓	✓	✓	✓	✓
Bidirectional Control	✓	✓	✓	✗	✗	✓	✗	✗	**P**
Waypoint Mission Planning	✓	✓	✓	✗	✗	✓	✗	✗	**P**
CTD/ADCP Sensor Integration	✓	✗	✗	✗	✗	✗	✗	✗	✗
Time-Series Database	✓	✗	✗	✗	✗	✗	✗	✗	✗
USV Platform	✓	**P**	**P**	✗	✗	✗	✓	✓	✓

✓: Supported; ✗: Not supported; **P**: Partially supported
* Mission Planner (ArduPilot open-source GCS); used in USV studies including [9]

Despite these findings, no existing web-based GCS for USV ocean observation has adopted a time-series database, leaving this gap unexplored in the literature [20][21][22]. These limitations indicate that existing systems have not simultaneously addressed unified platform integration, low-latency communication, oceanographic sensor visualization, and efficient time-series data management within a single interface. This study addresses these gaps by designing and developing a web-based GCS for USV ocean observation that unifies telemetry monitoring, bidirectional remote control, CTD and ADCP sensor monitoring, and waypoint-based mission planning within a single integrated platform, employing MQTT and WebSocket for low-latency communication and PostgreSQL with TimescaleDB for time-series data management.

II. METHOD

A. Software Development Methodology

The system was developed using the Rapid Application Development (RAD) methodology as it supports fast iteration, continuous user feedback, and early validation of system functions [23]. RAD is particularly suitable for this study because the GCS requires repeated refinement between the frontend, backend, and hardware communication flow. Its iterative nature ensures that the final implementation remains aligned with operational needs and real-time data exchange requirements. The methodology consists of three main phases, as illustrated in Figure 1. In the Requirement Planning phase, the functional requirements for the ground control station were identified, focusing on mapping the telemetry data structure from all sensor modules installed on the USV, including oceanographic sensors, GPS, IMU, and battery monitoring.

This phase also involved designing the command payload format to ensure seamless bidirectional communication between the ground control station and the vehicle. Subsequently, the RAD Design Workshop stage involved an iterative design-and-refine approach to develop the frontend and backend. Iteration cycles were conducted to validate the bidirectional communication flow via the MQTT broker, covering both telemetry reception and remote command transmission. Each iteration produced a testable build evaluated against the defined functional requirements, with identified discrepancies addressed in the subsequent cycle.

On the hardware side, the USV is equipped with a GSM module as the network interface, enabling the vessel to continuously transmit telemetry and oceanographic sensor data to the server using the MQTT protocol [13], [26]. The backend, developed in Go, serves as the core processing layer. It handles incoming messages from the MQTT broker, processes telemetry payloads, and distributes data to other system components [26]. It also manages data validation and synchronization to ensure that only structured and reliable information is forwarded to the database and the frontend.

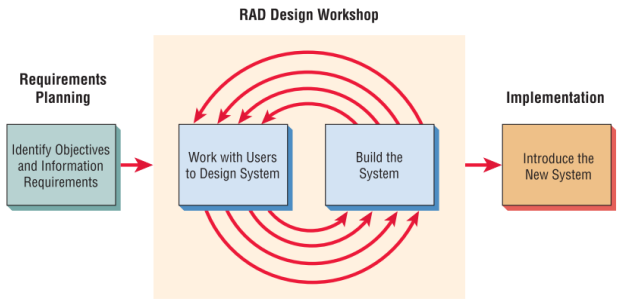

Figure 1. RAD Development Phases [23]

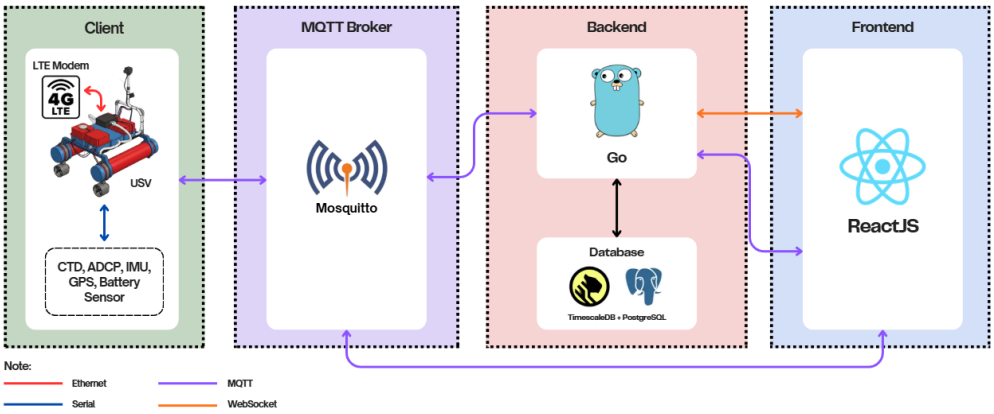

Figure 2. Overall System Architecture of the Web-based GCS

During integration testing, any discrepancies in the hardware-side data format were promptly resolved through backend and frontend adjustments to maintain system synchronization throughout the design process. In the final Implementation phase, the system was deployed into the operational environment for evaluation. Comprehensive testing was conducted using Black Box Testing to verify all functional features [24] and Performance Testing to quantitatively measure system stability [25].

B. System Architecture

The web-based GCS architecture for the USV was designed to support bidirectional real-time communication between the vessel and the operator while efficiently managing continuous multi-sensor telemetry data. The architecture is divided into two main sides: the hardware side residing on the vessel and the Virtual Private Server (VPS) side acting as the central processing hub, as illustrated in Figure 2.

The backend exposes a REST API for non-real-time operations such as historical data retrieval, mission management, and authentication, and concurrently runs a WebSocket server that relays continuous sensor streams from the broker directly to the frontend. All incoming telemetry is persisted in PostgreSQL with TimescaleDB, which leverages hypertable partitioning to maintain query performance under continuous data ingestion [20], [22]. The frontend, built with ReactJS, subscribes to the WebSocket stream and renders live sensor data on an interactive dashboard, including geospatial visualization and vessel control instruments.

C. Sequence Diagram

Figure 3 illustrates the sequence diagram for waypoint-based mission planning and execution. The operator defines missions through the Frontend, which are sent to the Backend via REST API for validation and persistence in the database.

Once confirmed, the mission payload is published to the MQTT Broker and forwarded to the USV for sequential execution, with progress updates streamed back to the Frontend via WebSocket in real time. This closed-loop mechanism ensures that the operator maintains full situational awareness throughout the mission. The integration of REST API for mission configuration and MQTT for real-time execution reflects the hybrid communication strategy adopted in the proposed architecture. In addition to mission planning, the system also handles continuous data monitoring, as illustrated in Figure 4. The USV continuously publishes telemetry and sensor data to the MQTT Broker, which the Backend subscribes to, validates against the database, and persists as sensor_log entries in TimescaleDB.

The Backend simultaneously broadcasts the incoming data through the WebSocket server, delivering real-time updates to the Frontend dashboard. Figure 5 illustrates the sequence diagram for remote control and command execution. The Frontend first establishes a WebSocket connection to the MQTT Broker, enabling persistent bidirectional communication throughout the operation. Command messages from the Frontend are then forwarded through the WebSocket connection to the MQTT Broker and subsequently delivered to the USV for execution. The USV publishes an acknowledgment response, which is relayed back to the Frontend via WebSocket, enabling real-time command status feedback on the user interface.

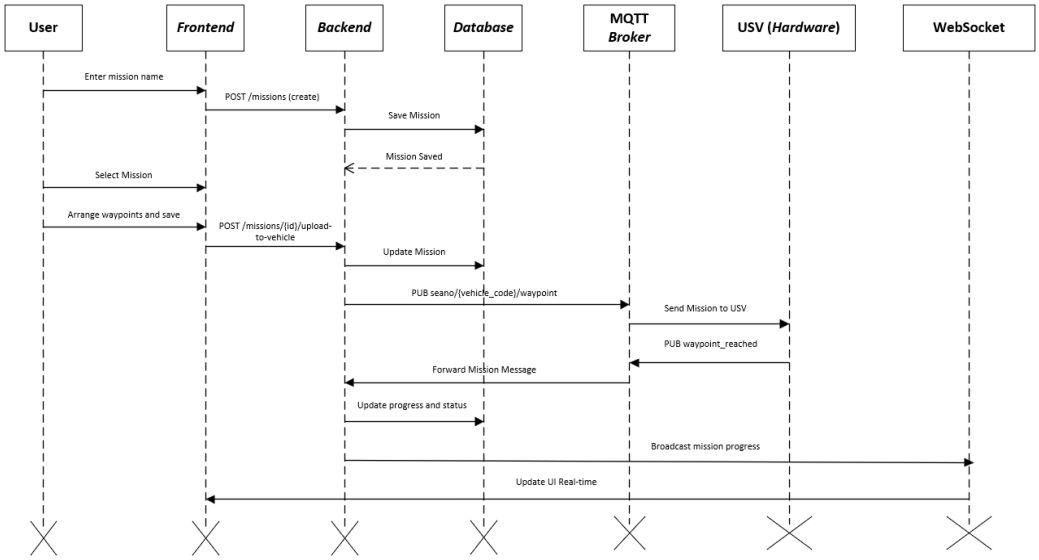

Figure 3. Sequence Diagram for Waypoint-Based Mission Planning and Execution in the Web-based GCS

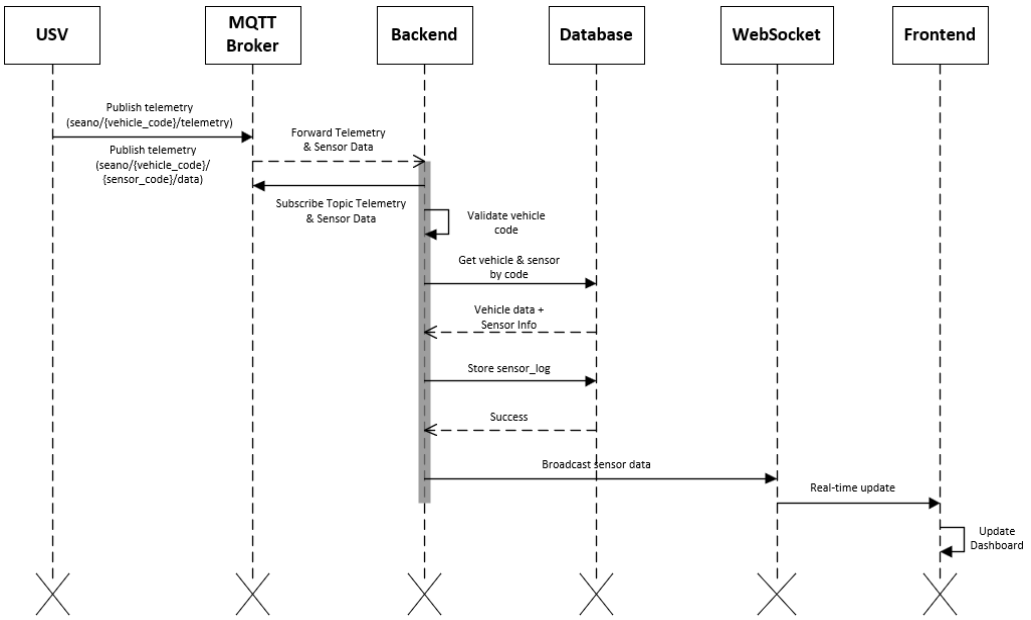

Figure 4. Sequence Diagram for Real-Time Telemetry and Sensor Data Monitoring in the Web-based GCS

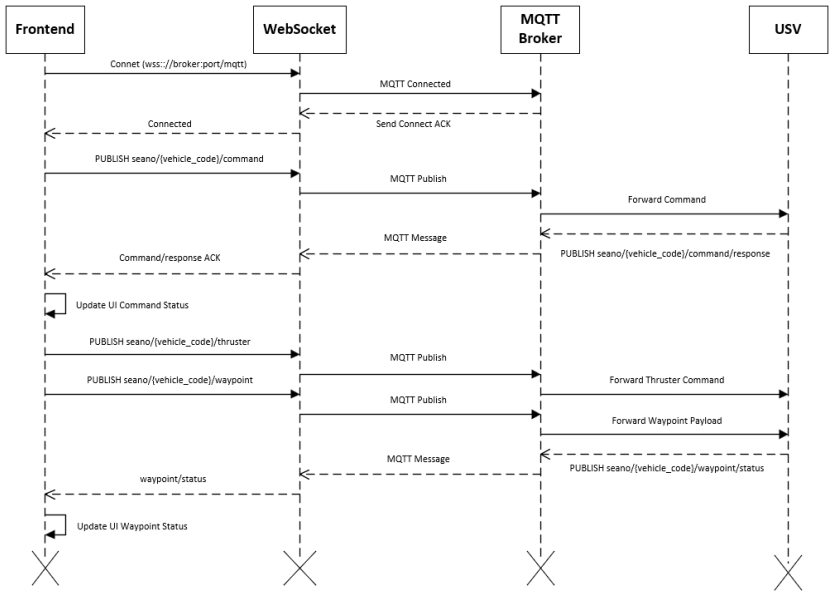

Figure 5. Sequence Diagram for Remote Control and Command Execution in the Web-based GCS

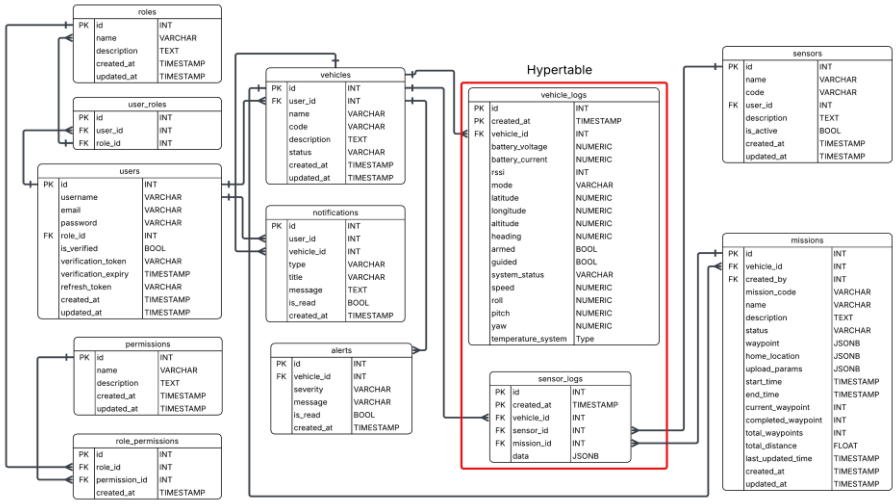

Figure 6. Database Schema of the Web-based GCS with TimescaleDB Hypertable Implementation

D. Database schema

The database schema of the proposed web-based Ground Control Station is designed to support both structured relational data and continuous time-series data generated by the USV system, as illustrated in Figure 6. The schema is implemented using PostgreSQL with the TimescaleDB extension to ensure efficient storage and retrieval of continuous telemetry and sensor data streams [20][21][22]. Core entities such as users, roles, permissions, vehicles, and missions are modeled using normalized relational tables to maintain data integrity and support system functionalities, including authentication, access control, and mission management. The relational structure ensures referential integrity across entities, enabling consistent tracking of vehicle assignments and mission ownership. Hypertable partitioning is applied to time-series tables, allowing TimescaleDB to manage ingestion rates and query performance independently from the relational layer [20][22].

This hybrid schema design enables the system to handle both transactional operations and continuous sensor data ingestion within a single unified database infrastructure. To handle real-time data ingestion, telemetry information is stored in the *vehicle_logs* hypertable, which captures aggregated navigation and system parameters such as position, heading, speed, battery status, and operational modes. In parallel, environmental and oceanographic sensor data are stored in the *sensor_logs* hypertable to accommodate heterogeneous payload formats from various sensors, enabling flexible integration without requiring frequent schema modifications.

E. User Interface

The user interface of the proposed web-based Ground Control Station is designed to support three primary operational features: real-time monitoring, remote control, and mission planning. The interface adopts a responsive design to ensure accessibility and usability across different devices while maintaining real-time interaction capabilities.

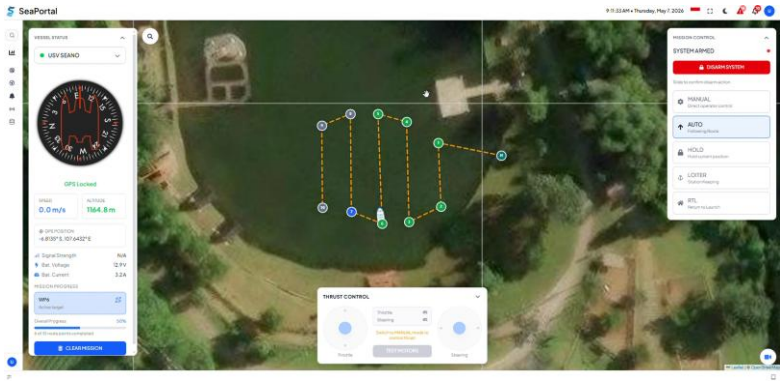

Figure 7. Control Interface of the Web-based GCS

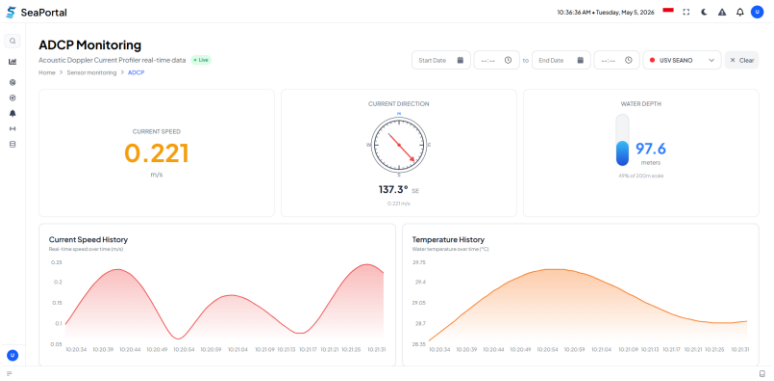

Figure 8. ADCP Monitoring Interface of the Web-based GCS

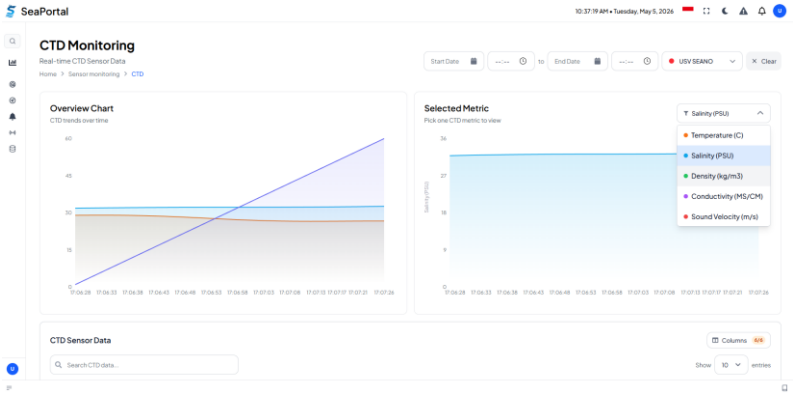

Figure 9. CTD Monitoring Interface of the Web-based GCS

The control interface, shown in Figure 7, provides operators with direct command inputs for vehicle movement and system status monitoring in real time. All interface components are unified within a single web-based platform, allowing operators to switch seamlessly between monitoring, control, and mission planning without requiring dedicated software installation. The sensor monitoring interface is divided into dedicated pages for each oceanographic instrument. The ADCP monitoring page, shown in Figure 8, displays real-time current speed, current direction, and water depth alongside historical trend charts. The CTD monitoring page, shown in Figure 9, presents an overview chart of all CTD parameters over time and allows operators to select individual metrics such as salinity, temperature, density, conductivity, and sound velocity for detailed visualization.

Both pages support time-range filtering, enabling operators to retrieve and review historical sensor data for post-mission analysis. Data updates across both pages are delivered continuously via WebSocket, eliminating the need for manual refresh. The mission planner interface, presented in Figure 10, allows operators to define and manage autonomous missions. Users can create waypoint-based routes directly on the map, adjust waypoint parameters, and configure mission properties such as speed and radius. Once configured, the mission can be uploaded to the USV for execution, with mission progress visualized on the map in real time. The unified interface design ensures that all operational functions are accessible within a single platform, reducing operator workload and supporting more efficient decision-making during field deployments.

Figure 10. Mission Planner Interface of the Web-based GCS

TABLE 2.
ENVIRONMENT SPECIFICATION

Component	Specification
USV Onboard – Edge Computing	
• Board	NVIDIA Jetson Orin Nano 8GB
• Autopilot	ArduPilot (CUAV X7+)
• Middleware	ROS Noetic
• FC Bridge	MAVROS
• Connectivity (USV)	Orbit 4G LTE Modem
• Connectivity (Client)	Orbit 4G LTE Modem
VPS – Server Side	
• Operating System	Ubuntu 22.04.5 LTS
• CPU	AMD EPYC 9354P — 4 vCPU (32-core physical, 1 thread/core)
• RAM	16 GB
• Backend	Go 1.24
• MQTT Broker	Mosquitto 2.0
• Database	PostgreSQL 15.13 + TimescaleDB 2.23.1
• Frontend	ReactJS 19.1.1
• Web Server	Nginx

F. *Environment Specification*

The system was implemented and evaluated across two main environments, as summarized in Table 2. On the USV onboard side, the autopilot telemetry is bridged to the MQTT communication pipeline, with CTD and ADCP data being simulated on the NVIDIA Jetson Orin Nano to replicate the actual sensor hardware integrated within the USV Command and Control Unit (CCU) [17]. The USV platform used in this study is shown in Figure 11. On the server side, the VPS environment provides the central processing hub for MQTT message routing, time-series data persistence, and real-time WebSocket streaming to the frontend.

III. RESULTS AND DISCUSSION

A. Functional Testing

Functional testing was conducted using a black-box approach to verify whether each core function of the proposed system operated according to the defined requirements. Black-box testing evaluates system behavior from an external perspective, focusing on input-output correctness without examining internal implementation details [24]. The evaluation covered three operational domains: real-time monitoring, bidirectional remote control, and waypoint-based mission planning.

The results in Table 3 show that all ten tested features were executed successfully, confirming that the proposed web-based GCS supports the complete operational workflow of a USV ocean observation mission within a single integrated browser-based interface.The monitoring tests demonstrated that telemetry data from the USV, including position, heading, speed, battery status, and operational mode, could be received and displayed on the dashboard in real time. These results are consistent with previous web-based GCS studies that confirmed the ability of WebSocket-based architectures to sustain real-time multi-sensor visualization through persistent browser connections [7][11].

Beyond navigation telemetry, the monitoring tests also confirmed that CTD and ADCP oceanographic sensor data were successfully displayed in real time on their dedicated interfaces. CTD sensors are fundamental to oceanographic investigation, providing continuous measurements of conductivity, temperature, and depth that are essential for characterizing water column properties [18]. ADCP sensors complement this by delivering current velocity profiles across multiple depth layers, which are critical for understanding ocean circulation patterns during observation missions [19]. The successful real-time visualization of both sensor streams within the same web-based platform directly addresses the objective of building an integrated ocean observation GCS.

The control tests showed that commands sent from the browser interface were correctly received and executed by the USV, and acknowledgment responses were returned and displayed in real time. This result aligns with MQTT-based USV research that demonstrated the protocol's effectiveness for remote vehicle control and sensor monitoring over internet networks [27]. Unlike earlier MQTT-based control systems that focused primarily on remote actuation, the present study embeds bidirectional control within a broader operational framework that also encompasses oceanographic sensor visualization and mission management.

These results collectively confirm that the proposed architecture addresses the key limitations of existing GCS implementations. Most prior systems handle monitoring, control, and mission planning as separate functions on desktop platforms [16][17], while the proposed system consolidates all three within a single web-based interface. The integration of oceanographic sensor monitoring further extends the platform's relevance for ocean observation missions, going beyond generic vehicle telemetry to support domain-specific marine data acquisition. The success of all functional tests therefore confirms that the implemented system is functionally valid and aligned with the research objective of this study.

TABLE 3.
FUNCTIONAL TESTING RESULT

No	Feature	Test Scenario	Expected Result	Status
1	Monitoring	Receive telemetry data	Data displayed in real time on the dashboard	Success
2	Monitoring	CTD data visualization	CTD parameters visualized dynamically on the dashboard	Success
3	Monitoring	ADCP data visualization	ADCP parameters rendered and updated in real time	Success
4	Monitoring	Mission tracking	Mission route visualized and tracked on the map in real time	Success
5	Control	Send command to USV	Command successfully transmitted and executed by the USV	Success
6	Control	Control USV via Virtual Joystik	USV movement direction successfully controlled via virtual joystick interface	Success
7	Control	Receive command response	Acknowledgment message received and displayed in the interface	Success
8	Mission Planner	Create waypoint	Waypoint successfully created and visualized on the map	Success
9	Mission Planner	Upload mission	Mission data successfully transmitted and stored	Success
10	Mission Planner	Execute mission	USV follows the predefined waypoint sequence correctly	Success

TABLE 4.
REMOTE COMMAND TRANSMISSION LATENCY (MQTT OVER WEBSOCKET)

No	Command	T_sent	T_ack	Latency (ms)
1	Arm	2026-03-02 08:08:54.947+07	2026-03-02 08:08:55.123+07	176 ms
2	Disarm	2026-03-02 08:09:27.128+07	2026-03-02 08:09:27.311+07	183 ms
3	Change Mode: Auto	2026-03-02 08:09:55.037+07	2026-03-02 08:09:55.223+07	186 ms
4	Change Mode: Manual	2026-03-02 08:10:15.462+07	2026-03-02 08:10:15.603+07	141 ms
5	Change Mode: Hold	2026-03-02 08:10:37.116+07	2026-03-02 08:10:37.293+07	177 ms
6	Change Mode: Loiter	2026-03-02 08:11:09.395+07	2026-03-02 08:11:09.128+07	233 ms
7	Change Mode: RTL	2026-03-02 08:11:46.261+07	2026-03-02 08:11:46.437+07	176 ms
8	Joystick Forward	2026-03-02 08:15:39.290+07	2026-03-02 08:15:39.423+07	133 ms
9	Send Waypoint Mission	2026-03-02 08:20:57.120+07	2026-03-02 08:20:57.315+07	195 ms

The mission planner tests confirmed that waypoint creation, mission upload, and mission execution all functioned correctly through the browser interface. This is a key contribution because it transforms the system from a passive monitoring dashboard into an active mission management platform. Conventional desktop-based GCS platforms such as Mission Planner and QGroundControl support mission planning but require local software installation on dedicated devices, which limits deployment flexibility in field environments [9][10]. The proposed web-based architecture removes this dependency by enabling mission planning directly through any standard browser, making the system accessible across multiple devices simultaneously.

B. Performance Testing

Performance testing was conducted to quantitatively evaluate end-to-end communication latency and database storage efficiency under field conditions using a 4G-LTE cellular network, replicating actual USV deployment scenarios where both telemetry monitoring and remote command transmission rely on mobile network infrastructure. All evaluations were performed within the same integrated system instance in which monitoring, control, mission planning, and sensor visualization operate concurrently, confirming that the unified architecture does not introduce cross-component interference between functional components.

(1) Remote Command Transmission Latency

Remote command transmission latency was measured as the round-trip time from the moment a command was sent from the browser interface to the moment the acknowledgment response was received, covering the full communication path from the browser through the MQTT broker over WebSocket directly to the USV and back. Nine command types were evaluated as presented in Table 4, covering vehicle arming and disarming, operational mode changes, joystick-based directional control, and waypoint mission upload.

(2) End-to-End Monitoring Latency

End-to-end monitoring latency was measured as the elapsed time from the moment telemetry data was published by the USV to the moment it was received and rendered at the browser interface, covering the full communication path from the USV to the MQTT broker, where the Go backend subscribes to incoming telemetry across two data streams: vehicle log data, which encompasses navigation telemetry parameters, and sensor log data, which encompasses oceanographic measurements from CTD and ADCP payloads, before pushing the data to the browser via WebSocket.

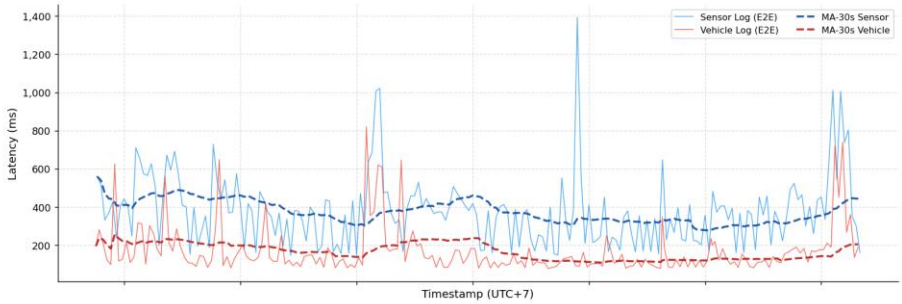

Figure 11. End-to-End Monitoring Latency over 4G-LTE Field Network with 30-Second Moving Average

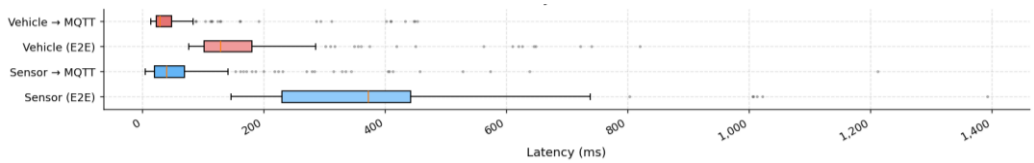

Figure 12. Latency Distribution by Communication Segment over 4G-LTE Field Network

TABLE 5.
TIMESCALEDB PERFORMANCE EVALUATION

Test	Metric	Regular Table	Hypertable	Performance Gain
Write Throughput	10,000 rows insert time	532.43 ms	209.24 ms	2.5×
Write Throughput	Rows/second	18,784 rows/s	47,796 rows/s	2.5×
Query A	1-hour range (1 vehicle)	3.998 ms	6.281 ms	-
Query B	24-hour avg/minute aggregation	13.984 ms	13.106 ms	1.07×
Query C	7-day row count	93.256 ms	24.890 ms	3.75×
Storage (compressed chunks)	Old chunk size	23 MB	64 kB	875×

The results show that round-trip latency ranged from 133 ms to 233 ms with an overall average of 178 ms, where mission upload commands exhibited slightly higher latency due to larger payload size compared to simpler binary state commands such as Arm and Disarm. These results confirm that all command types can be executed with sub-250 ms round-trip latency under 4G-LTE field conditions, which is acceptable for remote supervisory control of USV operations where commands are issued on-demand rather than continuously [27]. The low command transmission overhead is consistent with the design characteristics of MQTT over WebSocket as a lightweight protocol optimized for constrained IoT environments, while the higher average round-trip latency compared to the monitoring segment reflects the bidirectional nature of command-acknowledgment paths under LTE network conditions [13][14][15].

The time-series results in Figure 11 show that vehicle log latency remained consistently below 250 ms, while sensor log latency ranged between 300–500 ms due to larger payload size per record. The USV to MQTT broker segment exhibited very low and tightly distributed latency for both streams, confirming that the MQTT broker introduces minimal overhead consistent with its design as a lightweight IoT protocol [13][26]. Occasional spikes exceeding 1000 ms are attributed to transient network congestion rather than systematic protocol failures, and the overall end-to-end latency is within an acceptable range for supervisory control and ocean observation applications [1][7][14][15].

(3) Database Performance

Database performance was evaluated by comparing a standard PostgreSQL regular table against a TimescaleDB hypertable under three workloads: write throughput, time-range query performance, and storage efficiency, with results presented in Table 5. The hypertable completed insertion of 10,000 rows in 209.24 ms compared to 532.43 ms for the regular table, representing a 2.5× improvement attributable to time-based chunk partitioning that reduces write amplification under high-frequency ingestion workloads [20][21].

Query performance results show that the hypertable introduces a marginal overhead for short-range queries but delivers a 3.75× improvement for 7-day range queries, reflecting the advantage of chunk-based partitioning for long-term telemetry retrieval patterns. The most operationally significant result is storage compression, where hypertable compression reduced chunk size from 23 MB to 64 kB, a ratio of 875×, ensuring that months of continuous multi-sensor data can be retained on a standard VPS without costly storage expansion [20][21][22]. These results confirm that TimescaleDB integration provides meaningful advantages for the high-frequency data ingestion and long-duration storage requirements of the proposed GCS, representing a novel contribution not previously reported in existing USV GCS literature [20][21][22].

IV. CONCLUSION

This study presented the design and development of a web-based Ground Control Station for a USV ocean observation platform that unifies telemetry monitoring, bidirectional remote control, CTD and ADCP sensor visualization, and waypoint-based mission planning within a single browser-based interface. The system was evaluated through functional and performance testing to verify its operational validity and quantify its communication and storage characteristics.

Functional testing confirmed that all ten tested features functioned correctly, demonstrating that the proposed architecture successfully consolidates monitoring, control, and mission planning functions that prior systems have addressed only in isolation on desktop platforms. The integration of CTD and ADCP sensor visualization within the same operational interface represents a domain-specific advance over existing USV GCS implementations, which have focused predominantly on generic navigation telemetry without supporting the multi-parameter oceanographic data streams required for sustained ocean observation missions. Performance testing revealed that end-to-end monitoring latency remained below 250 ms for vehicle telemetry and between 300–500 ms for oceanographic sensor data under 4G-LTE field conditions, confirming that the MQTT and WebSocket communication architecture provides a stable low-latency channel suitable for real-time supervisory control.

Remote command round-trip latency averaged 178 ms across all nine command types, with sub-250 ms performance for all commands, validating the bidirectional responsiveness of the system under operational network conditions. These findings quantitatively demonstrate that MQTT over WebSocket delivers communication performance sufficient for remote USV supervision, extending prior comparative protocol studies to the specific context of a unified web-based GCS.

The adoption of TimescaleDB within the proposed system yielded a 2.5× improvement in write throughput, a 3.75× improvement in long-range query performance, and a storage compression ratio of 875× compared to a standard PostgreSQL table. These results establish a quantitative basis for recommending time-series database integration in future web-based GCS designs for continuous ocean observation, an approach that has not been previously reported in existing USV GCS literature. The proposed system therefore addresses the identified research gaps and provides a practical and replicable architectural reference for web-based USV GCS development across diverse ocean observation applications.

REFERENCES

[1] A. A. Priadi, T. Cahyadi, L. Barasa, S. A. Sukarno, N. Suranta, and M. Yando, "Development of IoT-Based Monitoring for Unmanned Surface Vessels (USV)," *J. Eng. Sci. Technol.*, vol. 20, no. 5, pp. 1–10, 2025.

[2] R. G. Patterson *et al.*, "Uncrewed Surface Vehicles in The Global Ocean Observing System: a New Frontier for Observing and Monitoring at The Air–Sea Interface," *Front. Mar. Sci.*, vol. 12, pp. 1–23, 2025, doi: 10.3389/fmars.2025.1523585.

[3] V. Bolbot, A. Sandru, T. Saarniniemi, O. Puolakka, P. Kujala, and O. A. V. Banda, "Small Unmanned Surface Vessels—A Review and Critical Analysis of Relations to Safety and Safety Assurance of Larger Autonomous Ships," *J. Mar. Sci. Eng.*, vol. 11, pp. 1–38, 2023, doi: 10.3390/jmse11122387.

[4] C. Barrera, I. Padron, F. S. Luis, O. Llinas, and G. N. Marichal, "Trends and Challenges in Unmanned Surface Vehicles (USV): From Survey to Shipping," *Int. J. Mar. Navig. Saf. Sea Transp.*, vol. 15, no. 1, pp. 135–142, 2021, doi: 10.12716/1001.15.01.13.

[5] P. Gogendeau *et al.*, "An autonomous surface vehicle for acoustic tracking , bathymetric and photogrammetric surveys," *Ocean Eng.*, vol. 331, pp. 1–15, 2025, doi: 10.1016/j.oceaneng.2025.121201.

[6] F. Kilic, M. Hassan, and W. Hardt, "Prototype for Multi-UAV Monitoring–Control System Using WebRTC," *drones*, vol. 8, pp. 1–24, 2024, doi: 10.3390/drones8100551.

[7] P. Eka, W. Lestari, M. Ridwan, R. F. Ramadhan, and F. Nisfa, "A Real-Time IoT-Integrated Ground Control Station (GCS) for Unmanned Aerial Vehicle (UAV) Monitoring System," *J. Electr. Electron. Eng.*, vol. 9, no. 2, pp. 109–122, 2025, doi: 10.21070/jeeeu.v9i2.1710.

[8] X. Liang, S. Zhao, G. Chen, G. Meng, and Y. Wang, "Design and Development of Ground Station for UAV/UGV Heterogeneous Collaborative System," *Ain Shams Eng. J.*, vol. 12, pp. 3879–3889, 2021, doi: 10.1016/j.asej.2021.04.025.

[9] J. D. Setiawan *et al.*, "Development and Performance Measurement of an Affordable Unmanned Surface Vehicle (USV)," *Automation*, vol. 3, pp. 27–46, 2022, doi: 10.3390/automation3010002.

[10] C. Ramirez-atencia and D. Camacho, "Extending QGroundControl for Automated Mission Planning of UAVs," *sensors*, vol. 18, no. 2339, pp. 1–23, 2018, doi: 10.3390/s18072339.

[11] N. Ardi, S. M. Razali, and A. H. Thohari, "Web-Based Ground Control Station for Real-Time Monitoring Quadcopter UAVs," *Proc. 8th Int. Conf. Appl. Eng. (ICAE 2025)*, pp. 132–150, 2025, doi: 10.2991/978-94-6463-982-7_9.

[12] F. Sotelo-torres, L. V Alvarez, and R. C. Roberts, "An Unmanned Surface Vehicle (USV): Development of an Autonomous Boat with a Sensor Integration System for Bathymetric Surveys," *Sensors*, vol. 23, pp. 1–26, 2023, doi: 10.3390/s23094420.

[13] B. A. Enache, C. K. Banica, and A. G. Bogdan, "Performance Analysis of MQTT Over WebSocket for IoT Applications," *Sci. Bull. Electr. Eng. Fac.*, vol. 1, no. 48, pp. 46–49, 2023, doi: 10.2478/sbeef-2023-0008.

[14] B. Amirkhanov, G. Amirkhanova, M. Kunelbayev, S. Adilzhanova, and M. Tokhtassyn, "Evaluating HTTP, MQTT Over TCP and MQTT Over WebSocket for Digital Twin Applications: A Comparative Analysis on Latency, Stability, and Integration," *Int. J. Innov. Res. Sci. Stud.*, vol. 8, no. 1, pp. 679–694, 2025, doi: 10.53894/ijirss.v8i1.4414.

[15] K. T. M. Tran, A. X. Pham, N. P. Nguyen, and P. T. Dang, "Analysis and Performance Comparison of IoT Message Transfer Protocols Applying in Real Photovoltaic System," *Int. J. Networked Distrib. Comput.*, pp. 1–13, 2024, doi: 10.1007/s44227-024-00021-4.

[16] A. T. Demetillo and E. B. Taboada, "Real-Time Water Quality Monitoring For Small Aquatic Area Using Unmanned Surface Vehicle," *Eng. Technol. Appl. Sci. Res.*, vol. 9, no. 2, pp. 3959–3964, 2019, doi: 10.48084/etasr.2661.

[17] S. A. Sukarno, H. Rudiansyah, H. T. Maulana, M. Kadir, and F. S. Adi, "Design and Development of a Command and Control Unit (CCU) on an Unmanned Surface Vehicle (USV) for Water Observation," *Int. J. Mar. Eng. Innov. Res.*, vol. 10, no. 4, pp. 1124–1132, 2025, doi: 10.12962/j25481479.v10i4.

[18] S. Xiao, M. Zhang, C. Liu, C. Jiang, X. Wang, and F. Yang, "CTD Sensors for Ocean Investigation Including State of Art and Commercially Available," *sensors*, vol. 23, no. 586, pp. 1–22, 2023, doi: 10.3390/s23020586.

[19] Ø. Bergh *et al.*, "A Modular Smart Ocean Observatory for Development of Sensors, Underwater Communication and Surveillance of Environmental Parameters," *sensors*, vol. 24, no. 6530, pp. 1–14, 2024, doi: 10.3390/s24206530.

[20] S. Rinaldi, F. Bonafini, P. Ferrari, A. Flammini, E. Sisinni, and D. Bianchini, "Impact of Data Model on Performance of Time Series Database for Internet of Things Applications," *IEEE Instrum. Meas. Technol. Conf.*, pp. 5–10, 2019, doi: 10.1109/I2MTC.2019.8827164.

[21] A. Khelifati, M. Khayati, A. Dignös, D. Difallah, and P. Cudré-Mauroux, "TSM-Bench : Benchmarking Time Series Database Systems for Monitoring Applications," *Proc. VLDB Endow.*, vol. 16, no. 11, pp. 3363–3376, 2023, doi: 10.14778/3611479.3611532.

[22] G. Piotr and D. Mrozek, "Comparative Analysis of Time Series Databases in the Context of Edge Computing for Low Power Sensor Networks," *Comput. Sci. – ICCS 2020*, pp. 371–383, 2020, doi: 10.1007/978-3-030-50426-7_28.

[23] K. E. KENDALL and J. E. KENDALL, *Systems Analysis and Design*, 8th ed. Upper Saddle River, NJ: Prentice Hall, 2011.

[24] A. Verma, A. Khatana, and S. Chaudhary, "A Comparative Study of Black Box Testing and White Box Testing," *Int. J. Comput. Sci. Eng.*, vol. 5, no. 12, pp. 301–304, 2017, doi: 10.26438/ijcse/v5i12.301304.

[25] S. Pargaonkar, "A Comprehensive Review of Performance Testing Methodologies and Best Practices: Software Quality Engineering," *Int. J. Sci. Res.*, vol. 12, no. 8, pp. 2008–2014, 2023, doi: 10.21275/SR23822111402.

[26] K. Il Ko and M. H. Lee, "MQTT-Based Architecture for Real-Time Data Collection and Anomaly Detection in Smart Livestock Housing," *sensors*, vol. 25, no. 7186, pp. 1–21, 2025, doi: 10.3390/s25237186.

[27] R. A. Atmoko, D. Yang, M. Y. Alfiani, and L. Subiyanto, "Controlling Unmanned Surface Vehicle Using MQTT Protocol," *J. Comput. Networks, Archit. High Perform. Comput.*, vol. 1, no. 2, pp. 21–28, 2019.