Sensors & Transducers, Vol. 272, Issue 1, April 2026, pp. 50-58

www.sensorsportal.com

Sensors & Transducers

Published by IFSA Publishing, S. L., 2026
http://www.sensorsportal.com

A Transport-Layer-Oriented Servo Actuation Framework for UAV Systems Using USB-CAN Communication

[1, *] Jinyoung LEE and [2] Sungwook CHO

[1] Cheongju University, Department of Mechanical and Aeronautical Systems Engineering, 298 Daeseong-ro, Cheongwon-gu, Cheongju-si, Chungbuk, South Korea

[2] Cheongju University, Department of Aeronautical and Mechanical Engineering, 298 Daeseong-ro, Cheongwon-gu, Cheongju-si, Chungbuk, South Korea

[1] Tel.: + 82 43 229 7948,

[*] E-mail:swcho84@cju.ac.kr

Received: 30 Jan. 2026 /Revised: 13 April 2026 /Accepted: 21 April 2026 /Published: 28 April 2026

Abstract: This study introduces a lightweight software framework for controlling servo-based actuation systems in unmanned aerial platforms through a universal serial bus to controller area network communication interface. The proposed approach emphasizes reliable command transmission, structured feedback acquisition, and deterministic communication behavior under general-purpose operating environments. Instead of focusing on closed-loop control design, the framework addresses challenges related to packet framing, timing consistency, and data integrity during continuous communication between host systems and actuator units. A custom byte-level framing structure is employed to ensure clear boundary detection, error validation, and stable parsing of actuator state information such as position, velocity, and electrical current. The system adopts a dual-layer scheduling mechanism to separate command execution from feedback collection, improving temporal consistency and reducing communication interference. Experimental validation demonstrates that the proposed framework achieves stable actuation performance with minimal deviation between commanded and measured states. The results confirm that the system provides a practical and extensible foundation for reliable actuator communication in aerial robotic platforms.

Keywords: Servo motor control, CAN communication protocol, Real-time control loop, Concurrent process scheduling, Modular software framework.

1. Introduction

Servo-driven actuation systems constitute fundamental components in unmanned aerial platforms, enabling precise mechanical responses in applications such as payload stabilization, camera orientation, and aerodynamic control surface adjustment. The performance of these actuation units directly affects the overall stability, sensing accuracy, and mission reliability of aerial systems, particularly under dynamic operating conditions involving rapid motion or environmental disturbances.

In conventional implementations, servo actuation is typically realized through pulse-width modulation interfaces or higher-level middleware frameworks that abstract device communication and control processes. While such approaches offer simplicity and broad compatibility, they inherently limit direct access to low-level communication behavior, including timing consistency, data integrity, and synchronization between command transmission and feedback acquisition. As a result, system-level visibility into communication-induced effects such as latency variation, frame misalignment, or feedback inconsistency remains restricted. In many practical configurations, servo actuators internally execute closed-loop control strategies, whereas external host systems primarily provide reference inputs and interpret feedback data. Under these conditions, the reliability of the communication layer emerges as a

critical factor influencing overall system performance. Variability in transmission timing, buffering behavior, and data reconstruction can directly impact the consistency of actuator responses, even when the underlying control mechanisms remain stable [1, 2].

Motivated by these limitations, this work focuses on the design of a lightweight communication framework that prioritizes deterministic data exchange and structured timing control in servo-based actuation systems. Rather than proposing a new control algorithm, the proposed approach emphasizes the robustness of command delivery and the integrity of telemetry acquisition under non-real-time operating environments. A USB–CAN interface is employed to facilitate communication between the host system and actuator devices, while a custom byte-structured framing mechanism is introduced to enable explicit boundary detection and reliable parsing of continuous data streams [3].

To further enhance temporal consistency, the system adopts a dual-loop scheduling strategy that separates command transmission from telemetry polling processes. This design reduces communication contention and enables stable interaction between control inputs and feedback data, even in the presence of operating system-induced timing variability. The resulting framework is designed to provide transparency in communication behavior, allowing direct observation and analysis of transport-layer characteristics that are often abstracted in conventional middleware-based systems.

The contributions of this work can be summarized as follows. First, a communication-oriented servo actuation framework is developed to ensure consistent and verifiable data exchange between host systems and actuator units. Second, a custom framing scheme is introduced to improve frame synchronization and data validation in continuous serial communication environments. Third, a dual-loop scheduling mechanism is implemented to maintain stable timing behavior under non-deterministic system conditions. Finally, the effectiveness of the proposed approach is demonstrated through experimental evaluation, highlighting its suitability for integration into unmanned aerial systems requiring reliable and transparent actuator communication.

This work extends our previous conference paper presented at DAUS 2026. While the earlier study provided a preliminary implementation of a USB–CAN-based servo communication system, the present work significantly expands the framework in several aspects. First, a transport-layer-oriented architecture is newly formalized, explicitly addressing byte-level framing, data validation, and communication robustness. Second, a dual-loop scheduling mechanism is introduced to improve temporal consistency under non-real-time operating conditions. Third, a detailed system architecture is presented, including modular layer separation and communication pipeline design. Finally, the experimental evaluation is extended to include quantitative analysis of latency, timing variability, and communication stability. These extensions distinguish the present work from the initial conference version and provide a more comprehensive and systematic framework for reliable actuator communication in UAV systems, moving beyond a preliminary implementation toward a rigorously structured communication architecture. These improvements enable a more rigorous evaluation of communication-layer behavior and provide stronger experimental evidence compared to the initial conference version [4].

2. Related Works

2.1. Conventional Servo Actuation Interfaces

Servo actuation in unmanned aerial systems has traditionally been implemented using pulse-width-modulation interfaces due to their simplicity and widespread hardware compatibility. However, such approaches provide limited support for structured telemetry, fault monitoring, and communication-level validation. In many cases, actuator state information such as position, velocity, and current is not directly accessible, which restricts observability and reduces suitability for applications requiring reliable state feedback and diagnostic capability.

2.2. Middleware-Based Robotic Frameworks

Modern robotic systems increasingly adopt middleware architectures such as Robot Operating System 2 to enable modular integration and distributed processing. Despite these advantages, recent studies have shown that communication latency and timing consistency in such frameworks are strongly influenced by the underlying operating system and execution model. In particular, non-real-time environments may introduce variability in message delivery, leading to jitter and reduced temporal predictability. These characteristics can significantly affect actuator communication, where consistent timing and reliable feedback acquisition are critical for stable system behavior.

2.3. CAN-Based Distributed Communication

Controller area network-based communication frameworks, including Cyphal, provide robust and scalable solutions for distributed embedded systems. These approaches emphasize interoperability, fault tolerance, and structured message exchange across multiple nodes. However, they typically abstract transport-layer details such as byte-stream handling, frame boundary reconstruction, and host-side scheduling behavior. While this abstraction is beneficial for large-scale integration, it limits direct visibility into communication timing and data integrity at the host-to-actuator interface [5-7].

2.4. Research Motivation

Existing approaches can be broadly categorized into pulse-width-modulation interfaces with limited observability, middleware-driven frameworks that offer strong integration but abstract timing behavior, and controller area network-based ecosystems designed for distributed communication. However, a critical gap remains in the design of lightweight actuator communication frameworks that explicitly expose transport-layer behavior – such as frame synchronization, payload validation, and timing control – while maintaining reliable and consistent timing under non-real-time conditions. To clarify these differences, the key characteristics of representative actuation communication frameworks are summarized in Table 1, focusing on communication interface, data integrity, temporal control, and system observability. As shown in Table 1, conventional approaches either provide limited visibility into actuator behavior or abstract communication mechanisms, whereas the proposed framework explicitly exposes timing structure and frame-level validation, enabling improved communication transparency and more predictable actuator behavior in non-real-time conditions [7].

Table 1. Comparative Analysis of Communication Characteristics in UAV Actuation Systems.

Feature	PWM Interface	CAN-Based Middleware	ROS2-Based Control	Proposed Framework
Communication Interface	Pulse Signal	CAN Bus	Middleware Messaging	USB–CAN Communication
Telemetry Availability	Limited	Structured	Structured	Structured (multi-parameter)
Data Integrity Mechanism	None	Protocol-Level Validation	Middleware-Level Handling	Explicit Frame Validation
Temporal Control (Host-Side)	Fixed	Abstracted	Execution-Dependent	Explicit Scheduling
Update Rate Control	Fixed	Configurable	Configurable	User-Defined
System Observability	Low	Medium	Medium	High
Integration Complexity	Low	Medium	High	Medium

3. System Architecture

3.1. Overall Framework Design

The proposed system is organized as a layered communication and control framework that separates command generation, data transmission, and actuator interaction to ensure stable and predictable operation. Instead of directly coupling user-level commands with hardware communication, the architecture introduces a modular structure in which each functional layer operates independently while maintaining synchronized data exchange. At the top level, the control layer is responsible for generating actuator commands and managing periodic execution. This layer schedules reference updates at a fixed interval and stores outgoing messages in a concurrent buffer structure, allowing safe and consistent interaction with lower-level processes. By isolating command generation from communication handling, the system reduces timing interference caused by user interaction or high-level logic.

The communication layer operates as an intermediary that translates scheduled commands into a structured byte stream and manages both transmission and reception processes. This layer performs frame construction, parsing, and logging, ensuring that all data exchanged with the actuator system is consistently formatted and traceable. It also maintains continuous monitoring of incoming data, enabling immediate detection of malformed or incomplete frames.

The actuator layer consists of a servo motor unit equipped with an internal driver and encoder-based feedback mechanism. This unit executes position commands and continuously provides telemetry data, including position, velocity, and current measurements. The separation of responsibilities between host-side communication and actuator-side control enables the system to focus on reliable data transport rather than control algorithm implementation.

3.2. Layered Functional Structure

To improve modularity and timing robustness, the system is divided into four functional layers: application, scheduling, communication, and data interpretation. Each layer is designed to operate independently while exchanging information through well-defined interfaces.

The application layer provides user interaction and visualization capabilities, allowing real-time monitoring of actuator states and manual or scripted command input. This layer does not directly interact with hardware, which prevents user-side operations from affecting communication timing.

The scheduling layer manages periodic task execution and prioritizes command transmission over

background processes. It enforces a structured update cycle and coordinates access to shared resources, ensuring that command delivery remains consistent even when telemetry polling is active.

The communication layer handles message encoding and decoding over the serial interface. It constructs outgoing frames, transmits them through the interface module, and reconstructs incoming frames from continuous byte streams. This layer also maintains transmission and reception logs, enabling post-analysis of communication behavior.

Finally, the data interpretation layer converts raw payload data into engineering units and distributes processed information to external clients. This separation allows telemetry data to be analyzed and visualized without interfering with the communication pipeline.

3.3. Command Scheduling and Arbitration Strategy

To maintain stable communication behavior, the system employs a structured scheduling and arbitration mechanism that separates time-critical command transmission from background telemetry acquisition. Rather than allowing simultaneous access to the communication channel, the scheduler enforces a priority-based policy in which actuator commands are always transmitted before non-essential requests.

During active command updates, telemetry polling is temporarily suspended to prevent contention on the communication channel. This approach ensures that control commands are delivered within the intended time window, reducing the risk of delayed or skipped updates. Once the command transmission interval is satisfied, telemetry requests are resumed, enabling continuous monitoring of actuator state without compromising control stability.

In addition, a periodic refresh mechanism is implemented to maintain actuator engagement during idle periods. When no new command is issued within a predefined interval, the system retransmits the most recent valid command to prevent unintended actuator relaxation or drift. This strategy enhances reliability by ensuring that the actuator consistently maintains its last commanded state even in the absence of continuous input. The overall structure of the proposed framework is illustrated in Fig. 1.

4. AA–55 Frame Structure

4.1. Frame Design Overview

To ensure reliable communication over a continuous serial data stream, the proposed framework employs a lightweight byte-oriented framing structure referred to as the AA–55 format. This design enables explicit identification of frame boundaries and consistent reconstruction of messages, even under non-deterministic buffering conditions.

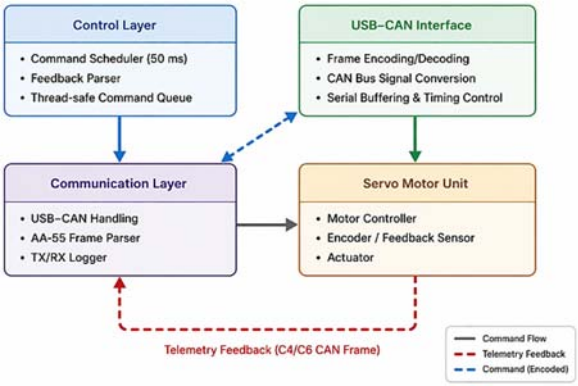


Fig. 1. Proposed Servo Communication Framework.

Each transmitted message is encapsulated within a structured frame composed of a start marker, control fields, payload data, and an end marker. The frame structure follows a fixed format: [Start] [Command] [ID] [Length] [Payload] [Checksum] [End]. The use of fixed boundary indicators allows the receiver to detect frame transitions without relying on external synchronization mechanisms. This approach is particularly effective in environments where partial reads or byte-level misalignment may occur during serial communication.

In addition to boundary detection, the frame structure incorporates a simple validation mechanism to ensure payload consistency. By verifying both the expected payload length and the integrity field, the receiver can identify malformed or incomplete frames before higher-level processing. This contributes to stable communication behavior by preventing corrupted data from propagating through the system.

4.2. Frame Field Composition

The AA–55 frame is organized into a sequence of fields that define both structural and semantic information. Each frame begins with a fixed start marker and ends with a corresponding termination marker, forming a clear boundary for message reconstruction. Between these markers, control and data fields specify the purpose and content of the message.

The command identifier indicates the type of operation, such as motion control or telemetry request, while the identifier field specifies the target actuator or parameter group. The payload length field defines the size of the data segment, enabling consistency checks during parsing. The data field contains actuator-related information, including position, velocity, current, and system status values. Finally, an integrity field provides a lightweight validation

mechanism that allows early detection of transmission errors at the host level.

This structured representation enables a unified handling of both command and feedback messages, simplifying the communication pipeline while maintaining flexibility for different actuator operations.

4.3. Frame Reconstruction Process

Incoming data from the serial interface are processed as a continuous byte stream without predefined message boundaries. To reconstruct valid frames, the receiver employs a state-based parsing mechanism that continuously scans for the start marker and accumulates subsequent bytes until a complete frame is detected.

Once a potential frame is identified, the parser verifies its structural consistency using the payload length and integrity field. Frames that fail validation are discarded, and the parser immediately resumes searching for the next valid start marker. This process ensures rapid recovery from corrupted or misaligned data while maintaining bounded processing time.

By explicitly handling frame reconstruction at the byte level, the system maintains robustness against communication disturbances such as buffer fragmentation, partial reads, and transient transmission errors. This capability is essential for achieving reliable telemetry acquisition in non-real-time environments. The operational flow of the proposed communication framework is illustrated in Fig. 2. As shown in Fig. 2, the system operates through a structured sequence of states, including command scheduling, frame transmission, telemetry reception, validation, and parsing, ensuring consistent interaction between control and communication processes.

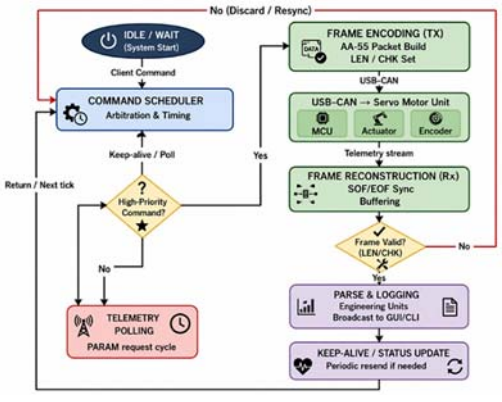


Fig. 2. Frame Reconstruction Process.

4.4. Timing Model and Communication Characteristics

The communication latency of the proposed framework can be described as a combination of multiple processing stages across the host system, communication interface, and actuator unit. The total round-trip latency is expressed as the sum of transmission scheduling delay, interface-level buffering, bus transmission time, actuator processing delay, and reception-side parsing time. In this study, these components were not evaluated individually; instead, the total latency was analyzed as an aggregate performance measure.

$$T_{rt} = T_{tx} + T_{usb} + T_{can} + T_{servo} + T_{rx} \quad (1)$$

The actuation command is transmitted at a fixed update interval, while telemetry data are collected at a slightly slower but configurable polling rate to balance communication load and feedback availability.

$$T_c = 50\,ms, T_t \in [80\,ms, 100\,ms] \quad (2)$$

To evaluate the response behavior, the latency between command transmission and the reception of corresponding telemetry feedback is measured as follows:

$$L = t_{rx}^{telemetry} - t_{rx}^{command} \quad (3)$$

Communication load is quantified by considering the number and size of transmitted frames within a given observation window, ensuring that the bus utilization remains within a predefined safe operating range.

$$R = \frac{N_c \cdot S_c + N_t \cdot S_t}{T}, R < R_{max} \quad (4)$$

Finally, the variability in latency is characterized using statistical analysis of repeated measurements, providing insight into timing stability under non-real-time execution conditions.

$$\sigma_L = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(L_i - \mu_L)^2} \quad (5)$$

These formulations allow systematic evaluation of communication performance and provide a basis for analyzing the impact of scheduling, buffering, and transport-layer behavior on actuator response.

5. Experimental Results and Discussion

5.1. Actuation Performance and Tracking Consistency

The effectiveness of the proposed framework was evaluated through a multi-step actuation experiment, where reference position commands were incrementally increased across a wide angular range. The objective of this evaluation was to assess the consistency between commanded inputs and measured actuator responses under the proposed communication and scheduling structure. The experimental results are

illustrated in Fig. 3, which presents the actuator position, velocity, and current profiles over time under multi-step reference inputs.

The experimental results demonstrate a high degree of agreement between the reference trajectory and the measured position values. The actuator response closely follows the commanded input throughout the entire operating range, with deviations remaining within the resolution limits of the telemetry and encoder interface. The mean position error was approximately 0.5 deg, with a maximum deviation below 0.8 deg. This indicates that the proposed communication framework is capable of delivering commands in a stable and consistent manner without introducing significant distortion or delay.

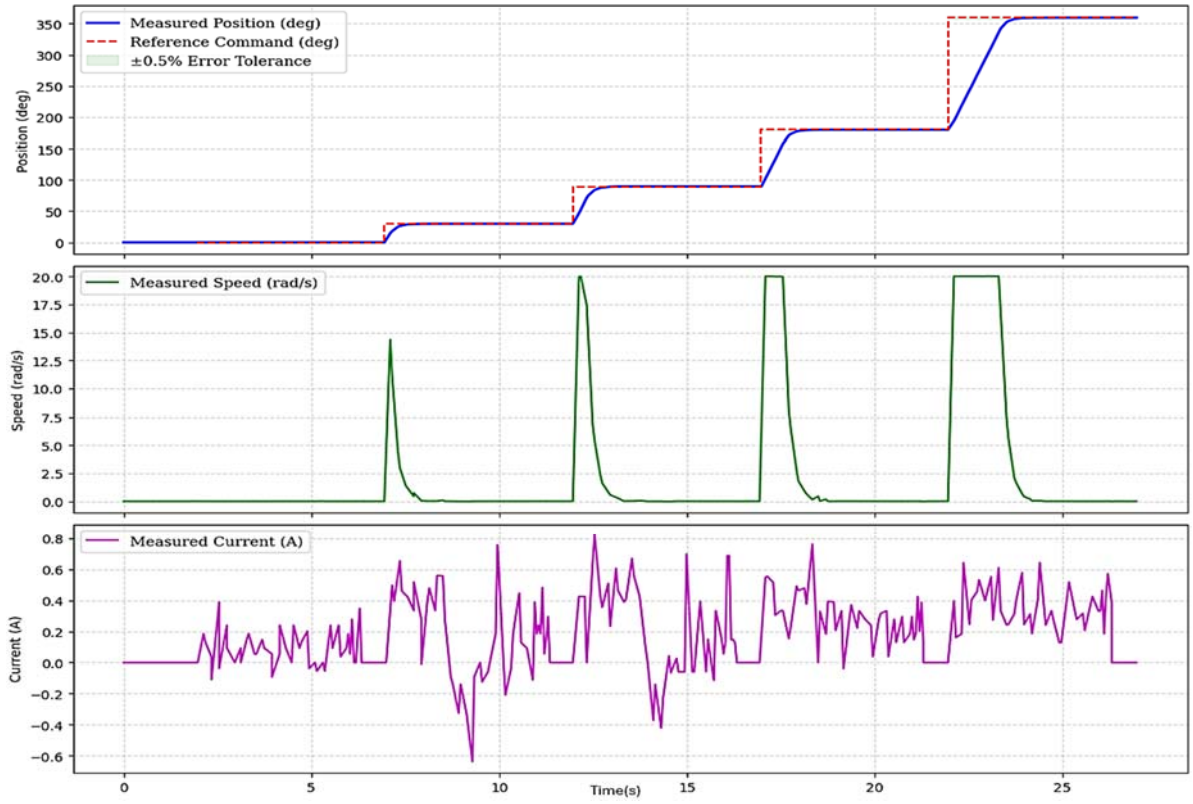


Fig. 3. Time-synchronized actuator response showing position, velocity, and current profiles under step command inputs.

In addition to positional accuracy, the dynamic behavior of the actuator was analyzed through velocity and current profiles. During each transition, the velocity increased smoothly without oscillatory behavior, while the current exhibited transient peaks corresponding to the torque required for motion initiation, followed by rapid stabilization at steady state. These observations suggest that the communication framework maintains sufficient temporal consistency to support stable actuator operation, even without direct modification of the underlying control algorithm.

5.2. Communication Stability and Timing Variability

The temporal characteristics of the proposed framework were evaluated by analyzing the intervals between consecutive telemetry receptions under periodic polling conditions. The results indicate that the telemetry reception intervals are distributed around the intended polling range, reflecting the influence of operating system scheduling and communication buffering.

Although variability in timing is observed, the majority of telemetry frames are received within a predictable operating window. This behavior demonstrates that the system maintains practical timing consistency despite operating in a non-real-time environment. More importantly, no frame loss or data corruption was observed during the experiments, indicating that the framing structure and parsing mechanism effectively preserve communication reliability. The mean telemetry reception interval was approximately 85 ms, with most samples concentrated in the range of 70–100 ms.

5.3. Discussion and System Implications

The experimental results highlight an important characteristic of actuator communication systems in unmanned aerial platforms: the reliability of data

transport and timing structure can significantly influence system performance, even when the control algorithm itself is not modified. The proposed framework demonstrates that stable actuator behavior can be achieved by ensuring consistent command delivery and structured telemetry acquisition at the communication level.

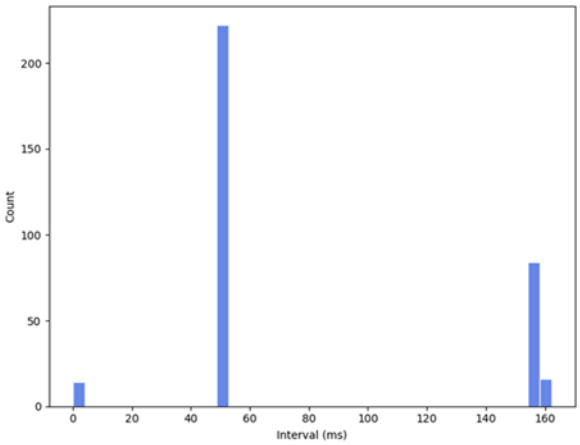


Fig. 4. Distribution of telemetry reception intervals under periodic polling conditions, showing the variability of communication timing in a non-real-time environment.

One notable observation is that, despite the presence of timing variability in the host system, the actuator response remains stable. This suggests that communication robustness, including frame synchronization and data validation, plays a critical role in decoupling control performance from non-deterministic system behavior. In this context, the use of explicit framing and structured scheduling provides a practical means of improving system reliability without requiring a real-time operating system.

Furthermore, the modular architecture of the framework allows it to function as a diagnostic and integration tool in addition to a communication interface. By exposing transport-layer behavior such as frame timing, data reconstruction, and command–telemetry interaction, the system enables detailed analysis of communication-induced effects that are often hidden in middleware-based implementations.

However, the current approach is subject to limitations inherent to non-real-time execution environments. While the system achieves stable operation within practical timing bounds, strict real-time guarantees cannot be ensured without operating system-level support. Future work may explore integration with real-time scheduling mechanisms or hardware-assisted communication interfaces to further reduce timing variability and enhance determinism. The temporal distribution of telemetry reception intervals is further illustrated in Fig. 5. The results reveal distinct clustering patterns, with the majority of samples concentrated around the nominal operating range, while occasional deviations appear due to system-level timing variability. These clusters reflect the combined effects of scheduling latency and communication buffering, which introduce non-uniform intervals in telemetry acquisition. Despite these variations, the overall distribution remains stable, indicating that the proposed framework maintains consistent communication behavior within practical operating bounds.

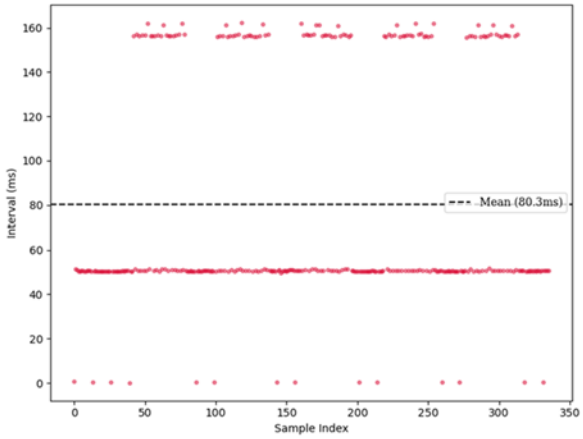


Fig. 5. Temporal distribution of telemetry reception intervals over successive samples, illustrating timing variability and clustering behavior under non-real-time communication conditions.

5.4. Comparative Evaluation

To validate the effectiveness of the proposed framework, a comparative analysis was conducted against conventional actuator communication approaches, including PWM-based control and ROS2 middleware-based communication.

PWM-based control provides low-latency signal transmission but lacks structured telemetry feedback and communication-level validation, limiting its suitability for applications requiring high observability and reliability. In contrast, ROS2-based middleware enables modular system integration and structured message exchange, but its performance is strongly affected by operating system scheduling and communication overhead, resulting in increased latency and timing variability.

The proposed USB–CAN framework achieves a balanced performance by combining structured communication with explicit timing control. Experimental observations indicate that the proposed method maintains lower jitter and higher reliability compared to ROS2-based approaches, while providing significantly improved observability over PWM-based systems. Table 2 summarizes the quantitative comparison of the evaluated communication methods. The results demonstrate that the proposed framework provides a practical trade-off between latency, reliability, and system transparency, making it suitable for UAV actuator systems operating in non-real-time

environments. The values in Table 2 are based on direct measurements for the proposed framework and representative estimates derived from system characteristics and prior studies for conventional approaches.

Table 2. Quantitative comparison of actuator communication methods based on experimental observations and system characteristics.

Metric	PWM (Pulse Signal)	ROS2 Middleware (DDS-Based)	USB–CAN (Proposed) (AA-55 / C4/C6 Frame)
Latency (Round-Trip) [ms]	5 – 10 Low latency due to simple pulse transmission	20 – 50 Affected by middleware overhead and OS scheduling	10 – 20 Moderate latency with structured framing and scheduling
Jitter (Std. Dev.) [ms]	Low Stable signal timing with minimal variation	High Variability due to non-real-time infrastructure	Low Bounded variability with dual-loop scheduling
Reliability	Medium No error detection / No feedback validation	Medium Reliable transport, but timing variability may cause loss	High Frame validation (AA-55) and checksum ensure high reliability
Observability	Low No telemetry feedback (open-loop)	Medium Telemetry available, but debugging affected by complexity	High Structured telemetry with explicit parsing and logging
Implementation Complexity	Low Simple hardware and software design	High Requires complex middleware configuration	Medium Custom framing and scheduling, but modular and lightweight
Bus Utilization (Measured / Estimated)	N/A (Analog pulse signal)	~40 – 70 % Depends on message rate and system load	< 35 % Measured under experimental conditions

** Values are measured or estimated based on experimental observations in a non-real-time operating environment.*

6. Conclusion

This study presented a communication-oriented servo actuation framework designed for unmanned aerial systems using a universal serial bus to controller area network interface. The proposed approach focused on improving the reliability of command transmission and telemetry acquisition rather than modifying the underlying control algorithm. By introducing a structured dual-loop scheduling mechanism and a custom byte-level framing scheme, the framework enables consistent data exchange and explicit handling of frame synchronization and validation under non-real-time operating conditions.

Experimental results demonstrated that the proposed system achieves high consistency between commanded and measured actuator responses, with deviations remaining within the resolution limits of the telemetry and encoder interface. In addition, the communication framework maintained stable telemetry acquisition despite timing variability introduced by operating system scheduling and interface buffering. These results confirm that reliable actuator behavior can be achieved through robust communication design even in environments where strict real-time guarantees are not available.

The proposed framework provides a practical and modular solution for actuator communication in unmanned aerial platforms, offering improved transparency in transport-layer behavior and facilitating integration with higher-level systems. Future work will focus on extending the framework toward real-time operating environments and multi-actuator configurations to further enhance timing determinism and system scalability.

Acknowledgement

The current research was supported by the Space Challenge Program funded by the Korea government (Korea AeroSpace Administration, KASA) (RS-2022-NR067041).

This work is supported by the Korea Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (Grant RS-2024-00408551).

References

[1]. C. Anastasopoulos, G. Tsourveloudis, K. Dalamagkidis, Design of a real-time test bench for UAV servo actuators, in *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS'18)*, 2018, pp. 1224-1231.

[2]. J. Sharma, R. Kumar, A. Nayyar, Robust PID control of single-axis gimbal actuator via UAV platform, *IFAC-PapersOnLine*, Vol. 53, Issue 1, 2020, pp. 113-118.

[3]. E. M. Coates, D. Reinhardt, K. Gryte, T. A. Johansen, Toward nonlinear flight control for fixed-wing UAVs: System architecture, field experiments, and lessons

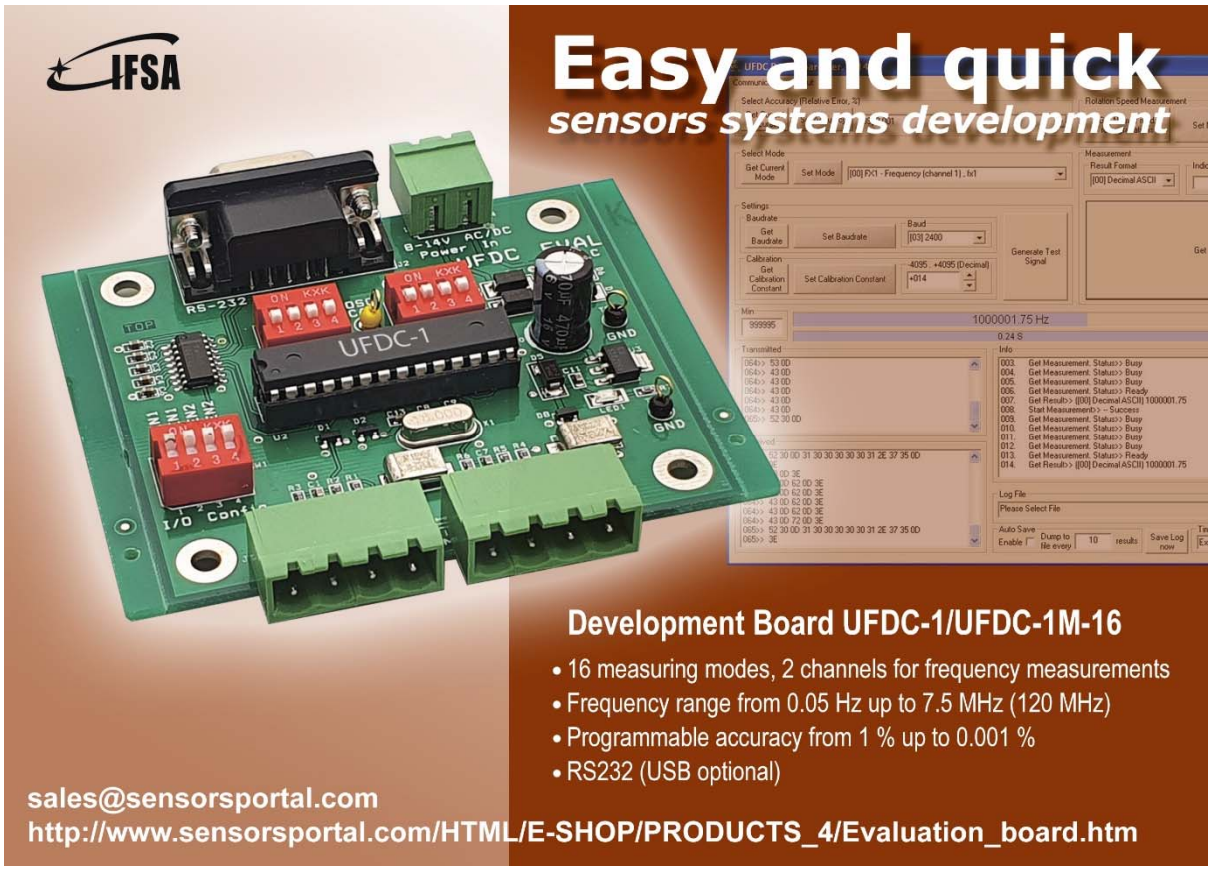
learned, in *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS'22)*, 2022, pp. 715-725.

[4]. J. Lee, S. Cho, Design and implementation of a servo motor control library for drone control systems, in *Proceedings of the 2nd International Conference on Drones and Unmanned Systems (DAUS'26)*, 2026, pp. 53-58.

[5]. T. Kronauer, J. Pohlmann, M. Matthé, T. Smejkal, et al., Latency analysis of ROS2 multi-node systems, in *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI'21)*, 2021, pp. 1-7.

[6]. Y. Ye, Z. Nie, X. Liu, X. Wu, et al., ROS2 real-time performance optimization and evaluation, *Chinese Journal of Mechanical Engineering*, Vol. 36, 2023, 144.

[7]. OpenCyphal Development Team, Cyphal specification v1.0, 2025, https://specification.opencyphal.org



Published by International Frequency Sensor Association (IFSA) Publishing, S. L., 2026 (https://www.sensorsportal.com).