Software Protocol Converters as Enablers for Interoperability in Heterogeneous Multi-Robot Systems

Nathan K. Wagner[ac]; Otávio A. R. da Cruz[a,b]; Pedro H. M. Pereira[a,b]; Carlos S. Junior [a]; Vitor Fontena[a]; Cassio Teixeira[a]

[a] Center for Embedded Devices and Research in Digital Agriculture (CEDRA), São Leopoldo, Brazil

[b] Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil

[c] Universidade do Vale do Rio do Sinos (UNISINOS), São Leopoldo, Brazil

A paper from the Proceedings of the
17th International Conference on Precision Agriculture and 11th Brazilian Congress on Precision and Digital Agriculture
13-15 July 2026
Porto Alegre, Brazil

Abstract.

Advancements in agricultural robotics are driving a transition toward ecosystems composed of heterogeneous platforms from multiple manufacturers. In this context, robotic platforms from multiple vendors must coexist and collaborate to perform complex tasks, including autonomous monitoring and precision weeding. However, this evolution introduces a fundamental challenge: the lack of a standardized communication stack capable of supporting cross-platform integration. The widespread use of proprietary systems and vendor-specific middleware, which are inherently incompatible, creates a critical barrier to interoperability. This challenge can be illustrated through an autonomous farming scenario composed of an Unmanned Aerial Vehicle (UAV) and a Ground Control Station (GCS). The UAV performs field monitoring using the MAVLink protocol for flight telemetry and georeferencing data streaming. In contrast, the GCS serves as a central supervisory management system based on RabbitMQ. Since the UAV cannot natively interact with GCS management services, the incompatibility between MAVLink's message-oriented data and RabbitMQ's message-oriented architecture creates a communication gap. This requires an intermediate translation layer, such as a ROS2-based bridge, to enable communication between both systems. This research highlights software-based protocol converters as architectural components for interoperability in heterogeneous multi-robot systems. While MAVROS is commonly used to bridge MAVLink and ROS2, the main contribution of this research is the development of a middleware that connects ROS2 to RabbitMQ. This bridge acts as a vital translation layer between robotic agents and high-level management systems, allowing platforms

to operate as interconnected components without requiring many modifications to existing hardware. By centralizing data into a common exchange format, the proposed middleware allows UAV-generated data, such as detected weed coordinates, to be published directly to the RabbitMQ broker and then consumed by other agents to trigger autonomous missions. This approach positions software-side converters as the primary enablers for scalable agricultural ecosystems. If integration with new platforms or external systems is required, the architecture only demands the development of specific converters, effectively dissolving communication silos and allowing for the seamless expansion of heterogeneous fleets in the field.

Keywords.
Interoperability, Heterogeneous Multi-Robot Systems, Software-side Converters, Precision Agriculture

Introduction

Agriculture is a structural pillar of the economy in several countries and a central component of global food security. According to projections from the Food and Agriculture Organization (FAO) of the United Nations, food production will need to increase by approximately 60% by 2050 to meet the demand of a world population estimated at 9.7 billion people (FAO, 2017). In Brazil, agribusiness accounts for approximately one quarter of the national Gross Domestic Product and nearly half of total exports, positioning the country among the world's leading producers of soybean, maize, coffee, and animal protein (IBGE, 2025). The economic scale and strategic role of the sector make agriculture both a public policy priority and an increasingly relevant domain for technological innovation.

The sector faces structural challenges, including rural labor shortages, pressure to reduce the use of pesticides and herbicides, and the need for efficient monitoring of large cultivated areas. Agricultural robotics has emerged as one of the main technological responses to these challenges (Oliveira, Moreira, & Silva, 2021). Robotic systems can support autonomous monitoring, weed detection, and selective input application, with demonstrated potential to significantly reduce herbicide consumption when compared with conventional broadcast spraying (Rahimi Azghadi, et al., 2025).

However, the advancement of these solutions has occurred in a fragmented manner, with platforms developed independently by different manufacturers. A typical scenario involves heterogeneous Unmanned Aerial Vehicles (UAVs), and Ground Control Stations (GCSs) that must cooperate in the execution of agricultural operations, even though they may have been designed using distinct architectures, proprietary communication stacks, and incompatible middleware technologies (Ren, et al., 2024). This configuration introduces a fundamental challenge: the lack of standardized communication mechanisms, which acts as a barrier to interoperability.

Different approaches have been proposed to mitigate this interoperability barrier. One possibility is the adoption of a unified communication standard by all agents involved in the system. However, this strategy has historically been difficult to achieve in robotics, mainly due to the reluctance of manufacturers to abandon proprietary communication stacks and platform-specific middleware. A second approach relies on centralized gateways capable of connecting multiple middleware technologies through a common intermediate representation. A third approach, widely adopted in practical robotic systems, consists of using protocol converters. These converters are specialized components that act as intermediate translation layers between specific pairs of heterogeneous technologies, without requiring modifications to the original platforms (Aragão, Moreno, & Bernardino, 2016).

In the UAV domain, this latter strategy is exemplified by MAVROS, a consolidated solution for integrating the Micro Air Vehicle Link (MAVLink) protocol with the Robot Operating System 2 (ROS 2) ecosystem (MAVROS, 2024). Similar protocol converters have been proposed in

industrial domains for integrating heterogeneous middleware technologies (Sim, Song, Shin, & Kim, 2021). Nevertheless, an important gap remains insufficiently explored in the literature: the integration between ROS 2 and message brokers used by management systems, such as RabbitMQ. In agricultural robotic ecosystems, data generated by robotic agents must be delivered to GCSs and supervisory applications, while mission commands and coordination instructions must also be sent back to the agents. Therefore, interoperability is not limited to robot-to-robot communication, but also includes communication between robotic middleware and management-level messaging infrastructures.

This article proposes a software bridge between ROS 2 and RabbitMQ, acting as a translation layer between robotic agents and the management system. The proposed approach reduces the communication silo between these two middleware technologies and enables the integration of new platforms through the addition of specific converters, without requiring changes to existing components. The main contributions of this work are: (i) the design and implementation of a bidirectional ROS2-to-RabbitMQ converter based on asynchronous communication, using a predefined JSON schema as an intermediate semantic representation for message standardization; (ii) a proof of concept of the bridge on an embedded platform composed of two Raspberry Pi nodes interconnected through a local wireless network and a SiK Telemetry radio link, exercising both ROS 2 and MAVLink communication paths; and (iii) the experimental characterization of conversion integrity and processing latency, demonstrating end-to-end message preservation and submillisecond conversion times in both directions.

Background

Interoperability

ISO/IEC 21823-1:2019 defines interoperability in Section 3.4 as "the ability for two or more systems or applications to exchange information and to mutually use the information that has been exchanged" (ISO/IEC, 2019).The standard identifies five interoperability levels: transport, syntactic, semantic, behavioural, and policy. All five must be addressed jointly for system integration to be effective. Among these, three are particularly relevant for the integration of heterogeneous multi-robot systems: transport interoperability, which refers to the ability of systems to exchange data through a shared communication network; syntactic interoperability, which concerns agreement on message structure, encoding, and format; and semantic interoperability, which refers to the consistent interpretation of the meaning of the exchanged information. In heterogeneous multi-robot systems, effective integration requires these dimensions to be simultaneously satisfied, which is difficult because these platforms are typically designed by different manufacturers and rely on distinct communication stacks.

This fragmentation becomes particularly evident at the level of robotic middleware. Each middleware typically ensures interoperability only among the components that adopt its own communication model, data structures, and execution assumptions. Manufacturers and system designers tend to encourage adoption of a specific middleware, while interoperability across different technologies is rarely addressed as a central design concern (Aragão, Moreno, & Bernardino, 2016). As a result, robotic systems based on different middleware often remain isolated from one another, even when they operate within the same application domain. For agricultural robotics in particular, where aerial platforms, ground vehicles, sensors, supervisory systems, and management applications may rely on different protocols and middleware technologies, interoperability cannot be restricted to network connectivity alone. It must also include message structure alignment, semantic standardization, and translation mechanisms capable of enabling heterogeneous agents to exchange data and commands consistently.

Communication Technologies

Heterogeneous multi-robot systems involve communication technologies with different characteristics and design objectives, organized around distinct functional levels. At the

embedded platform level, lightweight telemetry protocols are commonly used to support direct communication between unmanned vehicles and control stations. MAVLink is a representative example of this category, widely adopted for telemetry, command exchange, and mission-related communication for UAVs (MAVLINK, 2024). At the level of coordination among robotic agents, publish-subscribe middleware solutions such as ROS 2 have become widely adopted by providing distributed discovery, configurable Quality of Service (QoS), and low-latency peer-to-peer communication. ROS 2 relies on the Data Distribution Service (DDS) as its underlying communication middleware, which supports data-centric publish-subscribe interaction among distributed robotic systems (Macenski, Foote, Gerkey, Lalancette, & Woodall, 2022).

At a third level, robotic solutions deployed in enterprise or management-oriented environments must interact with supervisory systems, administrative platforms, and data management applications. These systems commonly adopt message brokers designed for enterprise integration, supported by the publish-subscribe communication paradigm, which provides full decoupling between publishers and subscribers in time, space, and synchronization (Eugster, Felber, Guerraoui, & Kermarrec, 2003). Brokers such as RabbitMQ, Apache Kafka, and ActiveMQ implement this paradigm, supporting messaging models in which producers and consumers are temporally decoupled, with the broker providing message persistence, configurable routing, and asynchronous communication. In agricultural robotics, robotic agents rely on telemetry protocols or robotic middleware, while management systems rely on broker-based asynchronous messaging. This layered communication landscape creates a structural integration challenge that motivates the use of dedicated translation mechanisms between communication layers.

Protocol Converters

Protocol converters are components that act as intermediate translation layers between technologies that do not communicate natively, addressing incompatibilities related to message formats, transport models, and communication paradigms. By isolating protocol-specific logic from the rest of the system, converters provide modularity and reduce dependency on modifications to embedded hardware, enabling integration to be incrementally extended as new platforms or technologies are introduced.

A well-known example of this strategy is MAVROS, which operates as a converter between MAVLink and ROS 2, providing a connection proxy for Ground Control Stations, automatic conversion between aerospace reference frames such as NED and ENU, and bidirectional mapping of MAVLink messages into ROS topics (MAVROS, 2024). In industrial domains, a converter between the Open Platform Communications Unified Architecture (OPC UA) and DDS has been proposed to integrate shop-floor systems with real-time communication infrastructures (Sim, Song, Shin, & Kim, 2021). These examples illustrate how protocol converters have been successfully applied to bridge heterogeneous communication technologies in specific domains. However, in agricultural robotics, there is still a lack of a reference architecture that defines how protocol converters should be structured, which functional components they should include, and how they should address common integration requirements such as telemetry, commands, georeferenced data, status information, and asynchronous supervision messages. This gap limits the reuse, extensibility, and systematic development of converters for heterogeneous agricultural robotic ecosystems.

Therefore, such architecture should provide mechanisms capable of connecting robotic communication, robot-to-robot coordination, and management-level message exchange, translating between robotic middleware and message brokers while preserving the semantic consistency of the exchanged data.

Proposed Architecture

The proposed architecture is intended to enable interoperability among heterogeneous robotic systems operating in agricultural environments. To achieve this objective, the proposed architecture adopts RabbitMQ as the standardized communication mechanism at the management layer, while allowing robotic platforms to preserve their native communication stacks. Interoperability is therefore achieved through an intermediate middleware layer responsible for protocol conversion and message standardization.

For clarity, the proposed architecture can be described from two perspectives. First, the system can be organized into architectural layers, which define the high-level interaction among robotic platforms, protocol converters, and management services. Second, this middleware can be described in terms of internal functional modules, which specify how messages are received, standardized, converted, and forwarded across heterogeneous communication domains. The next sections provide a review of these perspectives.

Architectural Layers

As shown in Figure 1**Error! Reference source not found.Error! Reference source not found.**, the proposed architecture is organized into three layers: the robotic platforms layer, the protocol converters layer, and the management layer.

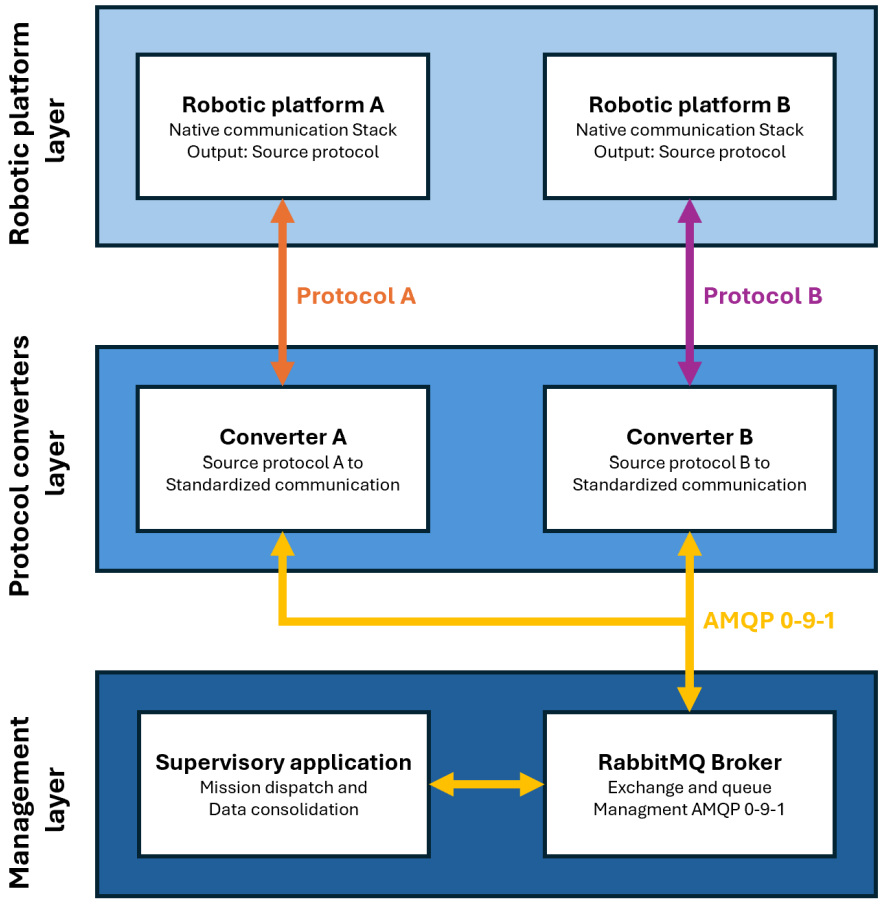

Figure 1 - Layered view of the proposed architecture.

The platforms layer comprises the heterogeneous agents deployed in the agricultural environment. These agents may include aerial robots, ground robots, or other platforms. Each

platform may operate according to its own native communication stack, here generically represented as source protocols. The proposed architecture does not require these protocols to be mutually compatible, since communication with the rest of the system is mediated by the intermediate protocol converters layer.

The protocol converters layer is responsible for bidirectionally translating messages between each source protocol and the system's standardized communication model**.** This layer acts as the core interoperability mechanism, since it isolates protocol-specific logic from the remaining system components. As a result, integrating a new robotic platform requires only the implementation of the corresponding converter, while preserving the native protocol of the source platform.

The management layer comprises the centralized communication and coordination services of the system. In the proposed architecture, RabbitMQ is adopted as the standard communication backbone of this layer. RabbitMQ implements AMQP 0-9-1, a message-oriented protocol based on exchanges, queues, routing keys, and message acknowledgements. Through this infrastructure, messages produced by heterogeneous robotic agents can be received, routed, buffered, and delivered to the appropriate consumers. This layer also includes the supervisory application, which is responsible for consolidating data received from the field and dispatching mission commands or coordination instructions to the robotic agents.

The communication flow supported by the architecture is bidirectional. In the upstream direction, robotic platforms generate telemetry, status information, perception outputs, and operational events, which are translated by the protocol converters layer and forwarded to the management infrastructure. In the downstream direction, commands and mission instructions generated by the supervisory application are published through RabbitMQ, translated by the conversion layer, and delivered to the corresponding robotic platform through its native communication interface. This bidirectional model allows the system not only to collect and consolidate data from heterogeneous agents, but also to coordinate their actions during agricultural operations.

From a generic perspective, the architecture can be interpreted as a mapping between multiple source protocols and a common management protocol. Each robotic platform uses a native source protocol, while the management infrastructure adopts RabbitMQ as the message broker, which implements the AMQP 0-9-1 protocol as the standardized communication mechanism. For each source protocol, interoperability is achieved through the implementation of translation mechanisms between the robotic platform and the management protocol. This design allows the architecture to be incrementally extended as new platforms and communication technologies are introduced.

Middleware Modules

While the layer-based view describes the overall organization of the system, the middleware itself can be described through a set of internal functional modules. As shown Figure 2, the middleware is organized into three main modules: ingress handling, converters, and egress handling. In addition, the conversion process relies on a JSON schema that acts as a common semantic schema for message standardization.

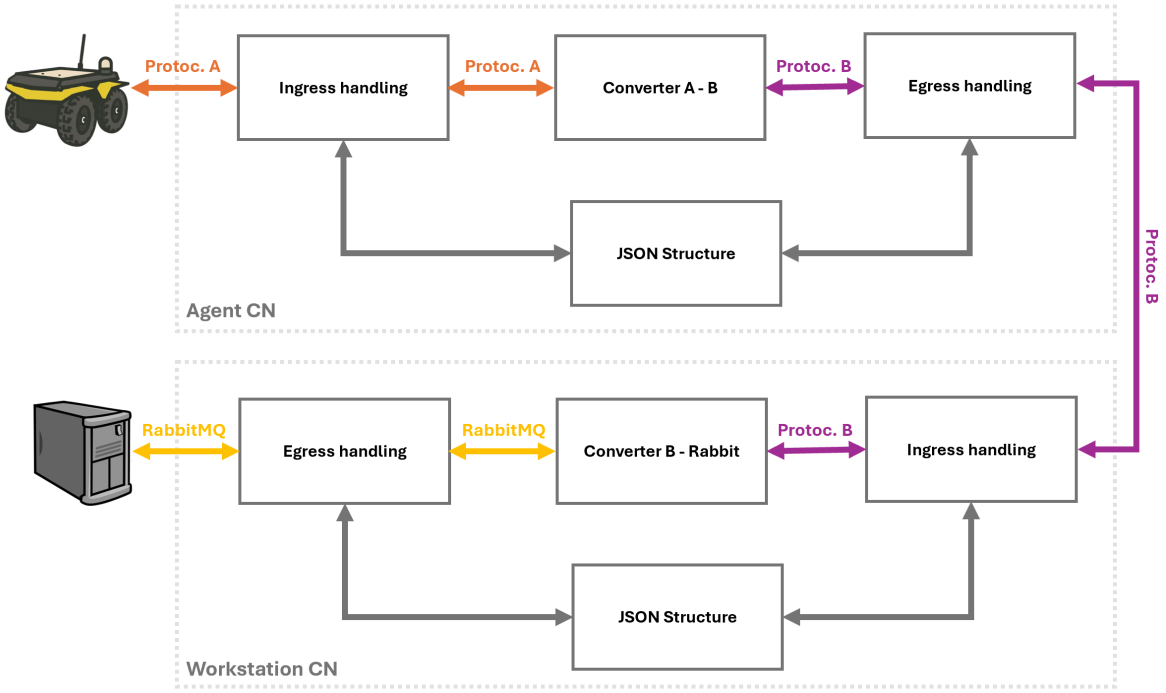

Figure 2 - Modular view of the middleware.

The ingress handling module is responsible for receiving messages from external communication interfaces and preparing them for internal processing. This module implements the required communication stack for each supported source protocol and performs initial processing tasks, such as payload reception, structural verification, and metadata extraction. Its role is to ensure that incoming messages are correctly captured and normalized before reaching the conversion stage.

The converter module is responsible for transforming messages from the source representation into the destination representation. This transformation may include syntactic conversion, serialization adaptation, field mapping, and protocol-specific restructuring. However, a central aspect of the proposed middleware is the use of a JSON structure as an intermediate JSON schema. Rather than performing direct point-to-point transformations between arbitrary protocols, incoming messages are first mapped to this JSON schema, which acts as a normalized semantic representation. This intermediate schema provides a common description of the exchanged data, enabling different protocols to interoperate through a shared representation rather than via isolated, tightly coupled conversions.

The adoption of a JSON schema provides two main benefits. First, it promotes semantic standardization, since messages originating from distinct communication stacks can be represented according to a common set of fields and meanings. Second, it improves scalability, since the integration of a new protocol only requires the implementation of mappings between that protocol and the JSON schema, rather than the development of dedicated converters for every possible protocol pair. In this sense, the JSON schema acts as a exchange model within the middleware.

The egress handling module is responsible for forwarding the converted message to the appropriate destination. Depending on the communication direction, this may involve publishing standardized messages to RabbitMQ, forwarding protocol-adapted commands to a robotic platform, or dispatching messages to another middleware interface. This module may also apply destination-specific processing, such as routing configuration, delivery constraints, and security annotations, to ensure that the message reaches the correct endpoint through the appropriate

communication channel.

Together, these modules define a middleware architecture that separates reception, semantic standardization, conversion, and transmission into distinct responsibilities. This modular organization reduces coupling among system components and facilitates future expansion. When a new robotic platform or communication protocol must be incorporated, the required adaptations remain localized within the ingress, conversion, and egress logic associated with that protocol, while the remainder of the architecture remains unchanged.

The proposed design, therefore, combines two principles. At the architectural level, interoperability is achieved by introducing a conversion layer between heterogeneous robotic platforms and a standardized management infrastructure based on RabbitMQ. At the middleware level, extensibility is achieved by organizing the conversion process into functional modules and by adopting a JSON-based semantic schema as a common internal representation.

Experiments

In this work, the proposed middleware is instantiated as RabbitROS, a software component developed to evaluate the architecture in practice. The evaluation was conducted through two independent and complementary experiments. The first experiment verifies the functional integrity of message conversion among the supported communication protocols. The second experiment characterizes the processing latency introduced by the protocol converters layer.

The physical platform consists of two Raspberry Pi 5 boards with 8 GB of memory. The first Raspberry Pi acts as the GCS, hosting the RabbitMQ broker, RabbitROS (the bidirectional converter between RabbitMQ and ROS), and an instance of MAVROS. The second Raspberry Pi acts as the robotic agent, representing the UAV node in the experimental setup. This node receives messages from the GCS according to the protocol exercised in each scenario.

In the scenarios involving ROS 2 communication, the robotic agent operates as a ROS 2 node connected to the GCS through a wireless local network. In the scenarios involving MAVLink communication, the robotic agent hosts a local MAVROS instance, which performs bidirectional conversion between MAVLink and ROS 2. MAVLink messages are exchanged through a Holybro SiK Telemetry Radio V3 link, connected to both Raspberry Pi boards through USB interfaces.

Communication between the two Raspberry Pi boards occurs through two distinct physical media. The first is a local IEEE 802.11 wireless network, over which DDS operates using UDP transport for ROS 2 traffic. The second is the SiK radio link, also connected through USB interfaces at both endpoints, which is used for MAVLink traffic. This configuration allows the evaluation to cover both network-based robotic middleware communication and radio-based telemetry communication.

All messages exchanged among the agents follow a predefined JSON schema. This schema establishes a common vocabulary of supported message types, fields, and structures, enabling message standardization across the RabbitROS. The JSON schema therefore acts as an intermediate semantic representation, ensuring that messages exchanged through different communication protocols can be interpreted consistently by the conversion layer.

Protocol Conversion Validation

The first experiment aims to verify the integrity of message conversion across the conversion chain, ensuring that each message generated at the source reaches its destination without changes in content, format, or semantic meaning. The evaluation comprises two test scenarios, each corresponding to a distinct conversion direction. The first scenario is restricted to RabbitROS converter between RabbitMQ and ROS 2, whereas the second scenario covers the complete conversion chain involving MAVLink.

In both scenarios, the message follows a round-trip path between the GCS and the robotic agent. For comparison purposes, the message is recorded at three points along the path: the original message published by the generator at the GCS, the message stored at the robotic agent after the first conversion sequence, and the message stored again at the GCS after the return path. This procedure allows the verification of whether the conversion chain preserves the original structure and semantic content of the message after both upstream and downstream transformations.

In the first scenario, illustrated in Figure 3, a generator running at the GCS publishes a JSON message to RabbitMQ (step 1). RabbiROS consumes the message from the broker and converts it into a ROS 2 topic (step 2), through which the message is transmitted to the robotic agent over the wireless network (step 3). The robotic agent receives the message through ROS 2, stores it, and immediately republishes it on the same topic toward the GCS (step 4). At the GCS, the RabbitROS translates the message back to AMQP and records it at the end of the path.

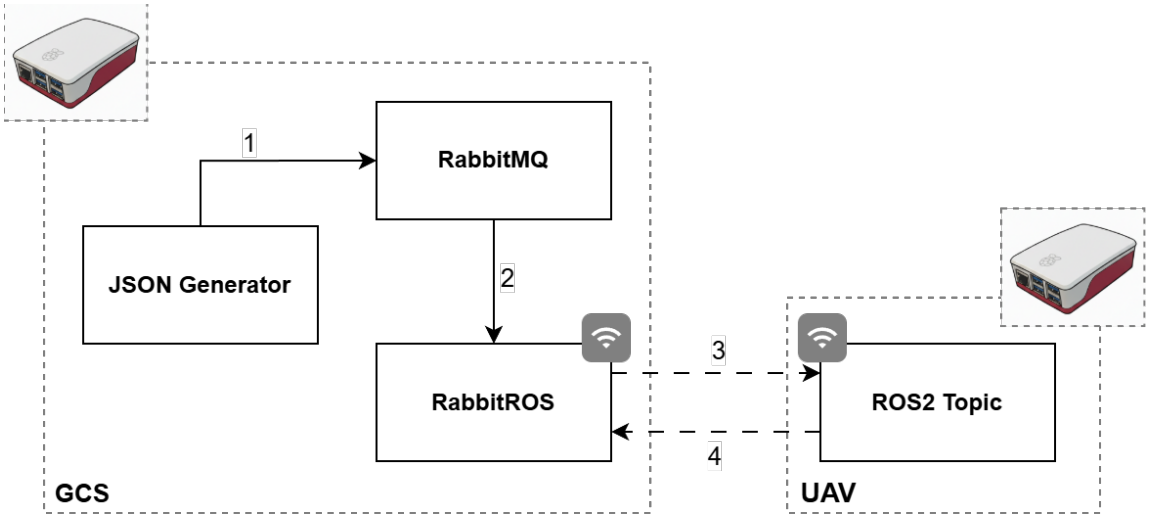

Figure 3 - Round-trip between RabbitMQ and ROS 2 through the WiFi link.

The second scenario, illustrated in Figure 4, evaluates the conversion path involving MAVLink, with conversion chains deployed on both sides of the communication link. The generator publishes the message to RabbitMQ (step 1). Within the GCS, the message is then successively converted to ROS 2 by the RabbitROS (step 2) and to MAVLink by the local MAVROS instance (step 3), after which it is transmitted through the SiK Telemetry radio link (step 4). At the robotic agent, a local MAVROS instance converts the received MAVLink message back to ROS 2 (step 5), where a ROS 2 node stores the message and republishes it (step 6). The message is then reconverted by the same local MAVROS instance to MAVLink and retransmitted through the SiK link back to the GCS (step 7). Finally, at the GCS, the reverse conversion chain restores the message to the AMQP representation and records it at the end of the path (step 8). This scenario validates not only the integrity of the bidirectional MAVLink-to-ROS 2 conversion on both sides of the communication link, but also the integrity of the end-to-end MAVLink communication path.

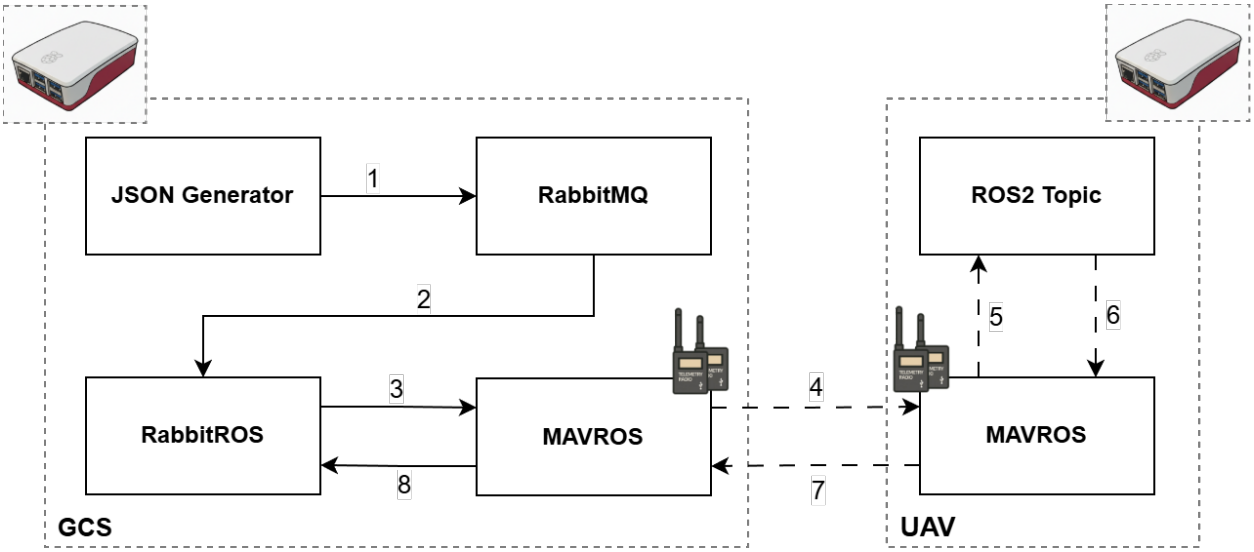

Figure 4 - Round-trip between RabbitMQ and MAVLink through the SiK Telemetry radio link.

Each scenario was executed five times, with 100 messages transmitted per execution at a publication rate of one message per second. The validation metric consists of a byte-by-byte comparison between the original JSON message and the corresponding versions recorded at the robotic agent and at the GCS at the end of the round-trip path.

The adopted acceptance criterion is the complete equivalence among the three recorded versions for all transmitted messages, meaning that a conversion path is considered valid only if every message preserves its original content across all conversion stages. During each execution, the three versions of every message are recorded in a CSV file, enabling subsequent verification, traceability, and reproducibility of the validation process.

Processing Latency Characterization

The second experiment characterizes the latency introduced exclusively by the RabbitROS. In this context, processing latency is defined as the elapsed time between the reception of a message in each source protocol and its publication in the destination protocol, within the same process. Transmission latency between agents is not included in this measurement, since the objective is to isolate the computational cost associated with the conversion procedure itself.

Because of this methodological decision, the measured scenarios are restricted to internal conversions that do not depend on radio-based communication. The experiment therefore considers the two direct conversion directions supported by the RabbitROS: RabbitMQ-to-ROS 2 and ROS 2-to-RabbitMQ. Both scenarios are executed entirely within the GCS, without involvement of the robotic agent, as illustrated in Figure 5.

In each scenario, two timestamps are recorded by the RabbitROS: the first immediately after the message is received at the component input, and the second immediately after the message is published through the destination protocol. The CT-L1 timestamps are labeled t_1 (input) and t_2 (output), while the CT-L2 timestamps are labeled t_3 (input) and t_4 (output). The processing latency for each scenario is therefore defined as $t_2 - t_1$ for CT-L1 and $t_4 - t_3$ for CT-L2.

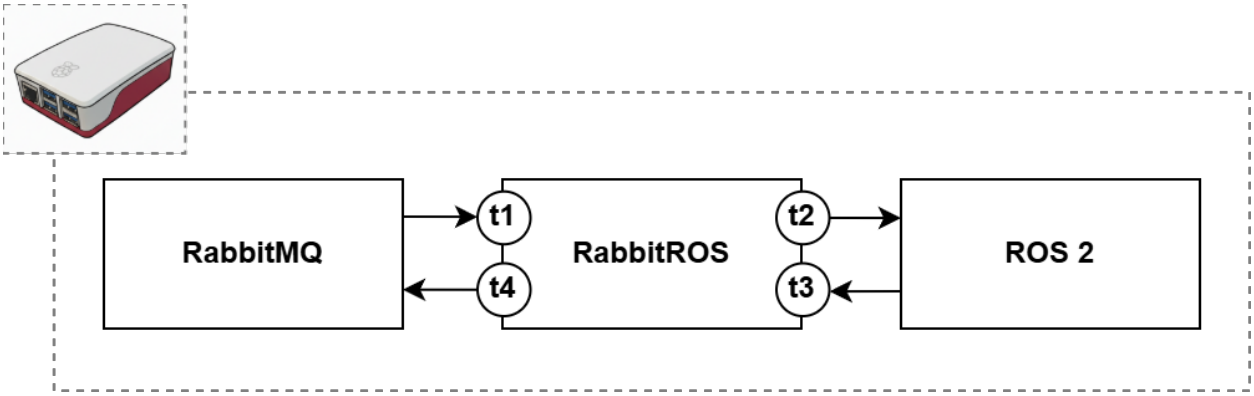

Figure 5 - Latency characterization conversions within the RabbitROS.

To investigate the influence of message size on communication performance, each scenario was executed using three different payload sizes: approximately 50 bytes, 150 bytes, and 250 bytes. The upper limit of 250 bytes was adopted to remain consistent with the 253-byte payload limit imposed by MAVLink v2.0. Although MAVLink is not directly exercised in this latency characterization experiment, this constraint preserves methodological consistency with the most restrictive transport protocol considered in the overall architecture.

Each combination of conversion direction and payload size was executed five times, with 100 messages transmitted per execution at a publication rate of one message per second. For each execution, the mean, standard deviation, minimum, and maximum latency values are reported, enabling a statistical characterization of the RabbitROS behavior under different payload sizes and conversion directions. The collected timestamps are recorded in CSV files to support subsequent analysis, traceability, and reproducibility.

Results and Discussion

This section presents the results obtained in both experiments, followed by a discussion of the observed implications.

Protocol Conversion Validation

Both validation scenarios were executed in full, totaling 500 messages per scenario (five executions of 100 messages each, at a rate of one message per second). In both scenarios and across all executions, the byte-by-byte comparison between the original JSON message and the versions recorded at the UAV and at the return to the GCS resulted in complete equivalence, with no discrepancies observed.

The result is particularly notable in the second scenario, which involves the full conversion chain on both sides of the radio link. Across each round-trip path, the message undergoes six successive conversions and two radio transmissions: from the RabbitROS to MAVROS at the GCS, transmission to the UAV, from MAVROS back to ROS 2 at the UAV, then in the reverse direction back to the RabbitROS at the GCS. The preserved integrity throughout this chain demonstrates the functional correctness of both the converters and the canonical JSON schema adopted as the common intermediate representation.

Processing Latency Characterization

Table 1 and Table 2 report the latency statistics observed in the two scenarios of the second experiment, for the three evaluated payload sizes.

Table 1 - Processing latency in scenario CT-L1, in milliseconds.

Size	Mean	Standard deviation	Minimum	Maximum
50 B	0.112	0.010	0.090	0.180
150 B	0.111	0.008	0.090	0.150
250 B	0.113	0.010	0.090	0.160

Table 2 - Processing latency in scenario CT-L2, in milliseconds.

Size	Mean	Standard deviation	Minimum	Maximum
50 B	0.832	0.114	0.750	1.770
150 B	0.838	0.099	0.750	1.490
250 B	0.831	0.098	0.750	1.540

The reported results correspond to the final configuration of the RabbitROS, in which the AMQP channel operates without explicit broker confirmation (`publisher_confirms=False`), while preserving the durability of published messages (`delivery_mode=PERSISTENT`), and in which the ROS 2 callbacks are executed in the same asyncio loop that operates the AMQP client, eliminating the synchronization overhead between distinct threads.

Two results are immediately apparent. The first is the insensitivity of latency to payload size in both scenarios: the variation between 50 B and 250 B remains within margins below 0.01 ms, indicating that the JSON serialization cost is negligible in the evaluated range, and that the processing time is dominated by constant overheads of I/O and queuing. The second is the asymmetry between the two directions: the ROS 2 to RabbitMQ conversion is approximately seven times slower than the inverse path. This residual asymmetry stems primarily from the cost of `aio_pika.publish()`, which encodes the message and writes it to the local socket toward the broker, combined with synchronous logging in the critical path. CT-L1, in contrast, requires only an asynchronous local publication on a ROS 2 topic, without interaction with the broker.

Conclusion

This work presented a software-based protocol converter that establishes a translation bridge between ROS 2 and RabbitMQ, addressing a gap in the literature regarding the integration of robotic middleware and enterprise message brokers. The proposed architecture adopts RabbitMQ as the standardized communication mechanism at the management level, while preserving the native communication stacks used by the heterogeneous robotic agents, enabling interoperability without requiring modifications to the existing platforms.

The experimental evaluation, conducted on a physical platform composed of two Raspberry Pi 5 boards and a SiK Telemetry radio link, demonstrated the functional correctness of the proposed architecture and characterized its performance. The validation experiment reported complete message integrity across all 1,000 transmissions, including the scenario involving six successive conversions and two radio transmissions. The latency experiment characterized the computational cost of the conversion in the submillisecond range in both evaluated directions, with insensitivity to payload size in the exercised range and a residual asymmetry of approximately seven times between the directions, attributable to the cost of publishing in the AMQP broker. These results empirically support the argument that software-based protocol converters constitute a viable and efficient solution for the integration of heterogeneous robotic

fleets in agricultural ecosystems.

Three main directions are identified for future work. The first is the extension of the latency characterization to scenarios involving MAVLink, currently restricted to functional validation in the present work. The second is the evaluation of the RabbitROS under high-load regimes, with publication rates compatible with dense telemetry flows, allowing the investigation of queuing and contention effects. The third is the validation in agricultural field conditions, with real robotic platforms operating in production environments, a necessary step for the transition of the presented contribution from a controlled proof of concept to a production-ready solution.

References

Aragão, M., Moreno, P., & Bernardino, A. (2016). Interoperability for Robotics: A ROS–YARP Framework. Front. Robot. AI. doi:10.3389/frobt.2016.00064

FAO. (2017). *The future of food and agriculture: Trends and challenges.* Food and Agriculture Organization of the United Nations. Fonte: https://www.fao.org/global-perspectives-studies/fofa/en/

IBGE. (2025). *Driven by Agriculture, GDP advances 1.4% in Q1. 2025.* Instituto Brasileiro de Geografia e Estatística. Fonte: https://agenciadenoticias.ibge.gov.br/en/agencia-news/

ISO/IEC. (2019). INTERNATIONAL ELECTROTECHNICAL COMMISSION, ISO/IEC 21823-1:2019 — Internet of things (IoT) — Interoperability for IoT systems — Part 1: Framework. Geneva: ISO.

MACENSKI, S., FOOTE, T., GERKEY, B., LALANCETTE, C., & WOODALL, W. (maio de 2022). Robot Operating System 2: Design, architecture, and uses in the wild. Science Robotics. *v 7*, p 66. doi:10.1126/scirobotics.abm6074

MAVLINK. (2024). MAVLink Developer Guide. Acesso em 22 de Maio de 2026, disponível em https://mavlink.io/en/.

MAVROS. (2024). MAVLink to ROS 2 Bridge. Acesso em 22 de Maio de 2026, disponível em https://mavros.readthedocs.io/en/latest/

Oliveira, L., Moreira, A., & Silva, M. (2021). Advances in Agriculture Robotics: A State-of-the-Art Review and Challenges Ahead. Robotics. doi:https://doi.org/10.3390/robotics10020052

RABBITMQ. (2024). AMQP 0-9-1 Model Explained. Acesso em 22 de Maio de 2026, disponível em https://www.rabbitmq.com/tutorials/amqp-concepts

Rahimi Azghadi, M., Olsen, A., Wood, J., Saleh, A., Calvert, B., Granshaw, T., . . . Philippa, B. (2025). Precision robotic spot-spraying: Reducing herbicide use and enhancing environmental outcomes in sugarcane. *Computers and Electronics in Agriculture*. doi:https://doi.org/10.1016/j.compag.2025.110365

Ren, Z., Chen, J., Chen, T., Xie, P., Xu, Y., Deng, J., . . . Jiao, W. (2024). Integrating UAV, UGV and UAV-UGV collaboration in future industrialized agriculture: Analysis, opportunities and challenges. *Computers and Electronics in Agriculture*. doi:10.1016/j.compag.2024.109631

Sim, W.-B., Song, B.-K., Shin, J.-H., & Kim, T.-G. (2021). Data Distribution Service Converter Based on the Open Platform Communications Unified Architecture Publish–Subscribe Protocol. *Electronics (MDPI)*.

Acknowledgments

This work has been partially funded by the project AgroBot supported by the Center for Embedded Devices and Research in Digital Agriculture (CEDRA) of SENAI-RS, with financial resources from the PPI IoT/Manufatura 4.0 / PPI HardwareBR of the MCTI, grant number 056/2023, signed with EMBRAPII. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.