Integration of Physical and AI Models Using Open and Interoperable Standards: A Model-Based Methodology for Autonomous Robot Development

Sebastian Rojas-Ordoñez[1, 2] Mikel Segura[1] Ekaitz Zulueta[2]

[1]IKERLAN, S. COOP, Spain, {jsrojas,msegura}@ikerlan.es
[2]EHU/UPV, Spain, jrojas015@ikasle.ehu.eus - ekaitz.zulueta@ehu.eus

Abstract

The development of Cyber-Physical Systems, particularly in the field of autonomous mobile robotics, often relies on proprietary environments, which limit flexibility and interoperability. This paper proposes a modular and open-source methodology that enables the modeling and simulation of such systems using non-proprietary tools. The methodology integrates Functional Mock-up Interface standard for model exchange, Open Neural Network Exchange standard for Convolutional Neuronal Networks algorithms integration, ROS2 as robotic middleware, and Gazebo as the simulation environment. To validate the approach, we applied it in the development of a mobile robot that navigates autonomously by following traffic signals. This implementation demonstrates that these open technologies can be effectively combined, overcoming common integration barriers among proprietary tools. The proposed workflow provides a practical alternative to proprietary solutions and demonstrates the feasibility of integrating open standards for the development of autonomous robotic systems. *Keywords: Autonomous robotic systems, FMI, ONNX, ROS2, Gazebo.*

1 Introduction

Model-Based Engineering (MBE) has transformed the way engineers and researchers design and develop complex systems (Holtmann, Liebel, and Steghöfer 2024). It is a strategy intended to reduce development time and to improve the quality of the system. To this end, the reuse of models and tests is promoted, thus avoiding iterative manual work and optimising the testing phase. Today, both industry and academia rely on MBE to predict system behavior, reduce the need of expensive physical prototypes, and drive advances in mobile robotics (Amorim 2019). At the same time, Convolutional Neural Networks (CNNs) are being increasingly applied in these systems, particularly for handling image processing tasks that support robot navigation and decision-making (Javaid et al. 2023).

Nowadays, many developments in robotics rely on commercial tools and proprietary platforms. While these tools offer powerful capabilities, their high cost and limited interoperability often restrict innovation (Haessler, Giones, and Brem 2023), increasing the demand for open-source alternatives that provide more flexible and accessible solutions.

In this context, the Modelica Association plays a key role in promoting and maintaining open standards for Cyber-Physical System (CPS) development, such as the Modelica Language for modeling dynamic systems and the Functional Mock-up Interface (FMI) (Modelica Association 2025) for ease the model exchange and co-simulation between different modeling and simulation (M&S) tools. These tools enable the development of complex systems, allowing for predictive modeling before physical prototyping, thus, reducing costs and minimizing development risks (Moshood et al. 2024). The integration of mathematical models with computer simulations has become a critical factor in ensuring the reliability and performance of modern systems (Idoko et al. 2024). However, applying these M&S capabilities to modern robotics frameworks is not exempt from challenges (Ray and Ramirez-Marquez 2020).

Regarding the evolution of robotics middleware, ROS2 emerged as a leading development framework for robotic applications. Particularly, ROS2 Humble (Open Robotics 2025b) offers enhanced features, including improved capabilities, robust security measures, and better support for multi-robot systems. When combined with advanced simulation environments like Gazebo (Open Robotics 2025a), it enables the validation of control algorithms and vision systems in realistic scenarios. However, the integration of Modelica-based models with these robotics frameworks remains challenging (Bardaro et al. 2017).

Additionally, the use of CNNs introduces a new set of challenges for systems based on traditional M&S approaches. As AI-based perception becomes increasingly relevant in CPSs, developers face the need to integrate machine learning and deep learning models into existing robotics and control workflows. The Open Neural Network Exchange (ONNX) (ONNX community 2025) standard addresses part of this challenge by offering a common, open format for representing and deploying trained neural networks across different platforms. However, the seamless integration of AI models in simulation tools, robotic middleware, and control systems remains an area of active research. This highlights the need for flexible and modular development methodologies that can facilitate the integration of AI capabilities into CPSs (Parwej 2024).

In summary, our research addresses these integration challenges by proposing an open-source methodology that uses:

- FMI standard for exchanging validated physical models via FMUs.
- ONNX standard for exchanging AI-validated models for image processing.
- ROS2 Humble as robotics middleware.
- Gazebo for 3D simulation.

By focusing on the integration of these non-proprietary components, our methodology offers a cost-effective, modular, and interoperable solution for the development of CPS, focusing in autonomous mobile robotics. This approach not only ensures effective communication between control algorithms, vision processing, and physical simulation while laying the foundation for future improvements in adaptive and AI-driven robotic systems.

The paper is organized as follows: Section 2 reviews the background and related work; Section 3 details the methodology and its architecture; Section 4 describes the proof of concept used to implement and demonstrate the proposed methodology, specifically focusing on the integration of 3D simulation, control, and image processing components; Section 5 presents the results and their implications; and Section 6 concludes with final remarks and directions for future research.

2 Background and Related Work

The integration of simulation, control, and image processing has become essential in developing modern robotic systems (Nevliudov, Tsymbal, and Bronnikov 2021). Recent open-source tools and integration developments have enabled innovative approaches for designing and validating complex systems (Santos et al. 2024; Robles, Martín, and Díaz 2023). However, ensuring seamless integration between the different components is still difficult. This section examines key technologies and related work, highlighting current limitations and opportunities for improvement in simulation in MBE, open-source tools in robotics, model exchange using FMI, and the integration of AI models through ONNX in robotic applications.

2.1 Simulation in Model-Based Engineering

MBE utilizes models as central tools to understand, design, and manage complex systems. This methodology enables effective analysis, early detection of potential issues, and improved efficiency through clear, structured information exchange (Holtmann, Liebel, and Steghöfer 2024). MBE supports critical tasks such as capturing requirements, system design, simulation-based validation, code generation, and result verification, focusing on specific system components to optimize key areas and mitigate development risks (Khandoker et al. 2022). For instance, (Sekar and Baras 2022) describe a model-based framework for robotic grasping, demonstrating how MBE facilitates iterative improvements and early error detection, ultimately reducing prototyping costs and risks.

Recent implementations, such as the one presented by Matlab/Simulink (MathWorks 2024) that combine ROS2 with control systems, show the potential and limitations of current approaches. In this use case, the system employs Simulink models to control simulated robots and manage data exchange, including camera feed, odometry, and speed commands; the focus of this use case involves an Image Processing subsystem for sign detection and a Sign Tracking Logic subsystem, implemented with a Stateflow chart, that translates detected sign information into linear and angular velocity commands.

While these implementations demonstrate MBE's capability for robust, iterative design, it is necessary to highlight the need for more flexible, platform-independent solutions (Corallo et al. 2022).

2.2 Open-Source Tools in Robotics

Open-source platforms have transformed robotics development by providing alternatives to traditional proprietary solutions. Authors of (Patel, Liarokapis, and Dollar 2022) discuss this transformation through open robotic hardware, documenting significant advances and diverse implementation practices across a range of applications. Similarly, (Soori, Arezoo, and Karimi Ghaleh Jough 2024) demonstrates how frameworks such as ROS2 accelerate industrial integration, particularly in Industry 4.0.

These advances in hardware and systems framework have prepared the way for the evolution of complementary software tools that further streamline robotics development. Market analyses by (Cornille 2024) confirm that including visualization tools like Rviz and Rerun.io, alongside physics simulators such as Gazebo and MuJuCo, it is possible to reduce development cycles and improve system reliability.

2.3 Model Exchange Using FMI

The FMI is an open standard that enables model exchange and co-simulation across different M&S environments, facilitating the integration of components from several domains, such as mechanical, electrical, and control systems. This capability is crucial for the development of complex systems where multiple M&S tools may work together. It defines a format for packaging such models, called Functional Mock-up Units (FMU). A FMU represents the practical implementation of the FMI standard, thus, it is a software component that complies with the FMI standard, making it portable and reusable across compatible simulation environments.

Multiple studies have demonstrated the effectiveness of the FMI standard in practical applications and system simulations. For instance, (Oudart et al. 2020) present a model-based toolchain that leverages the FMI standard for CPS co-simulation. Their approach effectively separates physical and logical models, achieving up to 40% reduction in development time. Additionally, (Shahsavari et al. 2021) introduce MCX, an open-source framework for digital twins, highlighting the capability of FMUs to encapsulate system dynamics in a reproducible manner.

Building on these results, integrating FMI with robotics frameworks has emerged as a natural next step to bridge the gap between simulation and control. One notable contribution in this direction is the `fmi_adapter` package (Lange et al. 2020), which provides a concrete implementation for connecting FMI with ROS and Gazebo. This tool enables two primary approaches for incorporating FMUs into ROS-based architectures:

1. **Node-Based Use:** As shown in Figure 1, this approach represents how to create a ROS node containing an FMU. The architecture shows subscribers (left side) connected to the FMI adapter (middle, blue), which interfaces with the FMI library to communicate with the Application FMU (right, green). This implementation is helpful to create the necessary ROS subscribers and publishers to handle all model variables.

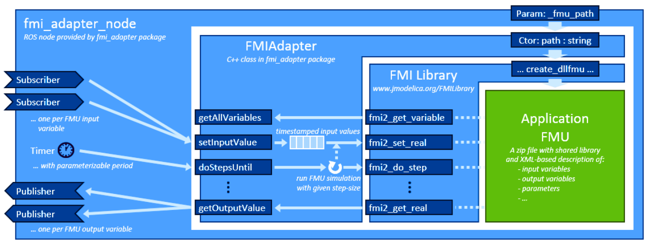

Figure 1. Node-Based Use
(Lange et al. 2020)

2. **Library-Based Use:** Figure 2 illustrates an alternative implementation using a C++ class (`fmi_adapter::FMIAdapter`). In this approach, the application node (left, green) directly interfaces with the FMI adapter components (blue). This method provides more direct control over the FMU lifecycle by eliminating the ROS node wrapper layer, allowing for more precise timing control and custom integration patterns.

Additionally, for Gazebo integration, (Lange et al. 2020) presented `gazebo-fmi` plugins, that enable the incorporation of FMU within the simulation environment. For co-simulation scenarios, `gazebo-fmi` components follow a master-slave architecture where one component, acting as the master, coordinates the execution of other components, which act as slaves. In this context, the main

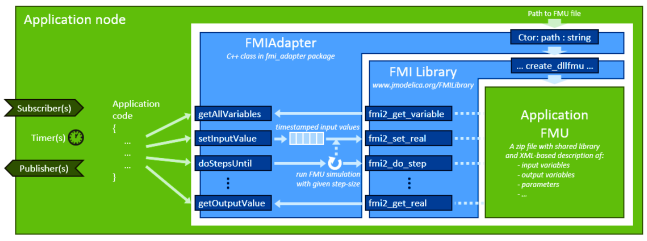

Figure 2. Library-Based Use
(Lange et al. 2020)

physics engine of Gazebo, acting as master, coordinates the overall simulation timing, while individual FMUs, acting as slaves, perform their specific calculations using their built-in solvers. The `gazebo-fmi` suite includes two key plugins:

1. **gazebo-fmi-actuator plugin:** For enabling the simulation of complex actuator dynamics such as reflected inertias, velocity/torque curves, and series elastic actuators.

2. **gazebo-fmi-single-body-fluid-dynamics plugin:** Designed to simulate fluid-body interactions, this plugin uses FMUs to compute lift, drag, and other fluid forces on a body.

2.4 AI Integration Using ONNX

Integrating artificial intelligence into robotic control systems presents significant opportunities in modern robotics, such as enabling adaptive decision-making, real-time optimization, and complex task automation. However, this integration also introduces technological challenges, including platform fragmentation, implementation complexity, and the need for reliable interoperability between AI models and robotic hardware (Patrício, Varela, and Silveira 2024). ONNX has emerged as a widely accepted standard for deploying AI models across various platforms. Its use enables standardized implementation of deep learning models, reducing dependence on customized and proprietary solutions. For instance, (El-Hussieny 2024) demonstrate a deep learning-based model predictive control strategy for a 3-DOF biped robot leg using ONNX, achieving real-time performance. Similarly, (Yue 2024) propose a real-time dynamic motion compensation design in vision-guided robots, that leverages ONNX to enhance task precision and system reliability, highlighting how standardized AI model deployment can improve task precision.

However, current approaches to integrating AI models with robotic control systems often rely on proprietary solutions or require complex custom implementations. This is particularly evident in applications that combine perception with dynamic control, where maintaining consistent performance across multiple subsystems can be difficult (Xu et al. 2024). Indeed, the robotics community has identified the need for more standardized, open approaches to AI integration that can reduce implementation complexity while maintaining system reliability (Licardo, Domjan, and Orehovački 2024).

3 Proposed Methodology

To address the challenges outlined in Section 2, this paper proposes a development methodology for robotic systems based on open-source tools and following MBE principles. Our approach focuses on integrating physical and control models using FMUs, and AI algorithms via ONNX models, all within a Linux-based environment. ROS2 is proposed as the communication middleware, and Gazebo as the simulation platform. In this way, we aim to facilitate the development of such a complex systems in a cost-effective, modular, and interoperable manner.

It is important to note that our methodology assumes these models are already developed and available; therefore, the focus is on their integration into a robotic framework. This integration applies both to simulated environments and to deployment on physical hardware. The proposed methodology is guided by the following key principles:

Interoperability and Open-Source: The use of open-source tools and standards enables the development of a modular and scalable system architecture. This approach eliminates dependency on proprietary environments and promotes collaboration between robotic system developers.

Unified Integration of Physical and AI Models: This proposal unifies the integration of FMUs, containing physical models or control algorithms, with ONNX files, containing machine learning or deep learning algorithms. The integration is performed through ROS2, this way we obtain several advantages:

- As many robotic developments incorporate ROS2, integrating these components does not require modifying previously developed systems. Instead, adding new functionalities simply requires incorporating additional components.
- It reduces development time and costs, as pre-developed models can be incorporated into the system without the need to rewrite code. In other words, it acts as a code generation mechanism, with FMU and ONNX serving as the final code elements.
- It provides flexibility, allowing individual modules to be updated or replaced without affecting the entire system.

Modular Architecture: Our design adopts a modular appoach, where independent subsystems are wrapped within ROS2 nodes. ROS2 uses a communication model based on the publisher-subscriber pattern, where nodes can publish messages to topics or subscribe to them to receive information. This communication is built on top of Data Distribution Service (DDS), a middleware that enables automatic discovery of nodes and message exchange without requiring nodes to know each other directly. Gazebo provides the virtual environment for system simulation. This modular architecture reduces the complexity of integrating diverse technologies. As a result, our proposal is easily adaptable to a wide range of robotic applications.

4 Proof of Concept

To validate the effectiveness of the proposed methodology, we applied it to a scenario in which a robot navigates from point A to point B by following traffic signals that guide its path. We obtained this use case from a Simulink example originally designed to validate communication between Gazebo and Simulink. Our purpose is to reuse some of its elements to demonstrate how FMUs and ONNX models can be integrated for the development of CPS, showing that the same components and tools can be used both in virtual testing environments and in real hardware deployment.

As our approach does not involve the development of the models themselves, our initial intention was to reuse the odometry and signal-tracking logic by exporting them as FMUs, as well as exporting the image processing module as an ONNX model. However, we encountered some difficulties that required some adjustments. First, we were unable to generate FMUs directly from Simulink, since our development environment was Windows-based, while the final application is intended to run on Linux. FMUs generated on Windows cannot be seamlessly integrated into a Linux environment. To address this, we replicated these modules in OpenModelica under Linux and generated the FMUs from there.

Regarding image processing, Simulink relies on a proprietary toolbox that cannot be exported. To overcome this limitation, we recreated the functionality using a Python implementation, using PyTorch to enable ONNX export.

On the other hand, we were able to reuse the original Gazebo scenario. Therefore, our validation strategy is to test whether the robot behaves in the same way in both the original Simulink setup and our integration, now relying exclusively on non proprietary tools. This way, in this proof of concept, ROS 2 orchestrates the system, integrating FMUs for control and odometry tasks, and an ONNX model for image processing, with Gazebo serving as the simulation platform. The following subsections outline the processes involved in applying the proposed methodology.

4.1 Application Scenario

Figure 3 shows the simulated environment in Gazebo, where the map and layout of the simulation area can be appreciated. In this scenario, the mission of the mobile robot is to navigate from point A to point B while following visual signals. The yellow dashed line represents the expected path that the robot should follow based on the traffic signs positioned throughout the environment.

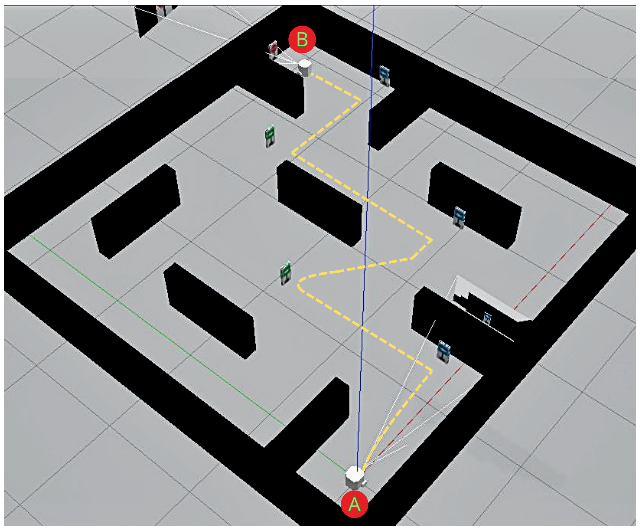

Figure 3. Simulated Gazebo environment showing the navigation path with traffic signs to guide the navigation of the robot.

The robot's navigation is guided by a set of traffic signals strategically placed throughout the environment, as shown in Figure 4. These signals provide the instructions for the robot's navigation behavior:

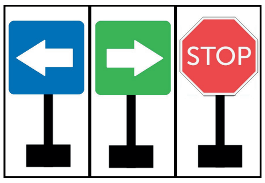

Figure 4. Traffic signals for robot navigation: Left arrow (blue) robot to turn left, right arrow (green) robot to turn right, and stop sign (red) robot to Stop.

4.2 Robot Pose Estimation (Odometry) FMU

The odometry management is handled via a two-stage process:

1. **Odometry Processing:** The system subscribes to the `/odom` topic in ROS2 to receive position and orientation data from the simulator. As shown in Figure 5, the FMU block `ReadOdometry` processes six input parameters: quaternion orientation (q_w, q_x, q_y, q_z) and cartesian position (x, y). The position coordinates (x, y) are directly mapped to the outputs ($posX$, $posY$), while the quaternion components are transformed into a 2D orientation angle ($theta$) using Euler angle conversion. This results in three output parameters ($posX$, $posY$, $theta$) that completely describe the robot's pose in 2D space.

2. **ROS2 Integration:** After understanding the functionality of the odometry FMU, we integrated it into a ROS2 node. This integration was performed by creating a node named (`fmu_odometry_node`) that subscribes to the robot's default `/odom` topic.

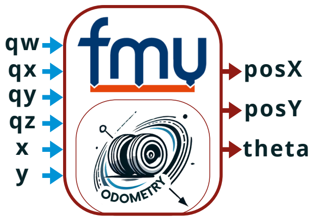

Figure 5. I/O port configuration of the odometry FMU block showing six inputs (four quaternion components and two position coordinates) and three outputs (2D position and orientation angle).

The node processes the odometry data and publishes the resulting pose information to a custom topic named `/fmu_odom_out`, as illustrated in Figure 6 the integration of the system is verified using rqt a ROS2's native introspection and visualization tool.

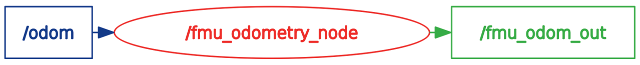

Figure 6. ROS2 node structure for the odometry FMU input subscription to `/odom` and output publication to `/fmu_odom_out`.

4.3 Traffic Sign Detection ONNX

The image processing subsystem employs a two-stage process:

1. **Vision System:** The ONNX model shown in Figure 7, processes RGB images with dimensions of 480×640×3 pixels received from the `/camera/image_raw` topic. The model takes this input tensor [1 480 640 3] and outputs two key parameters: `blobX` for the sign's horizontal position and `blobSize` for its dimensions.

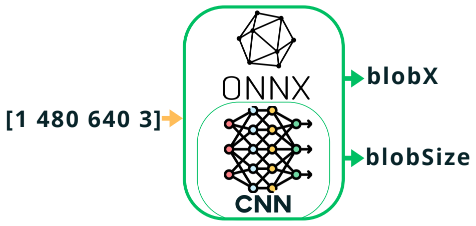

Figure 7. ONNX model structure showing the CNN architecture with its input tensor [1 480 640 3] (representing batch size, height, width, and RGB channels) and its two output parameters: `blobX` and `blobSize`.

2. **ROS2 Integration:** After understanding the functionality of the ONNX model, we integrated it into a ROS2 node. This integration was performed by creating a node that processes the camera input data and publishes the processed information through two custom topics: `/onnx_blobSize`, which provides

continuous updates of the traffic sign dimensions, and `/onnx_blobX`, which delivers the sign's horizontal position in the frame, as illustrated in Figure 8 the integration of the system is verified using rqt a ROS2's native introspection and visualization tool.

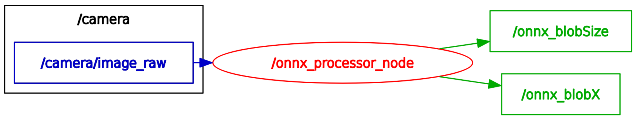

Figure 8. ROS2 node architecture for ONNX integration, illustrating input from the camera feed and output topics for sign detection parameters.

4.4 Autonomous Navigation Control FMU

The control system integrates the data from the FMU odometry model and the ONNX model subsystems through an FMU that manages the robot's control logic:

1. **Controller Design:** The SignTrackingLogic FMU incorporates decision-making logic to process visual inputs from the ONNX node and pose information from the odometry FMU. This FMU calculates the relative position of detected traffic signs and generates corresponding movement commands. Figure 9 shows the I/O configuration, where inputs include `blobSize`, `blobX`, and `pose` (i.e., *posX*, *posY*, *theta*), and outputs include linear velocity (`v`), angular velocity (`w`), and a stop command (`stopRobot`).

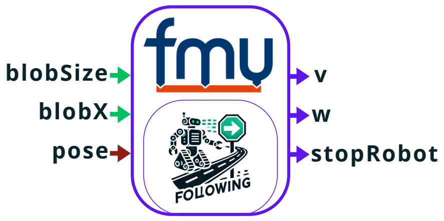

Figure 9. I/O port configuration of the SignTrackingLogic FMU block.

2. **ROS2 Integration:** After understanding the functionality of the SignTrackingLogic FMU, we integrated it into a ROS2 node. This integration was performed by creating a node named (`fmu_sign_tracking_node`) that subscribes to three topics: `/onnx_blobSize`, `/onnx_blobX`, and `/fmu_odom_out`. The node processes this information and publishes the resulting control commands to the `/cmd_vel` topic, as illustrated in Figure 10 the integration of the system is verified using rqt a ROS2's native introspection and visualization tool.

4.5 Integration of FMUs and ONNX

The final integration phase unifies the FMU-based control components and the ONNX-based image processing module within ROS2. This is accomplished through a layered

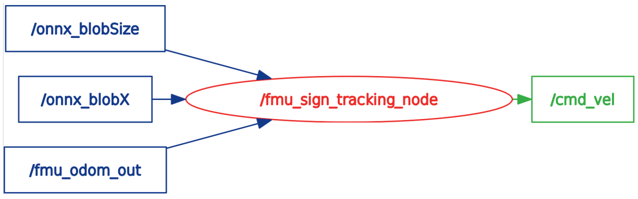

Figure 10. ROS2 node architecture for the SignTrackingLogic FMU, showing input subscriptions from image processing and odometry nodes, and output publication to robot control.

architecture that ensures communication and synchronization between nodes while maintaining system modularity and scalability. The overall integration is validated using Gazebo as the simulation environment.

4.5.1 System Architecture in Linux

Figure 11 illustrates the complete system architecture, which comprises FMU Nodes for odometry processing and sign tracking logic, an ONNX Node for image processing and Gazebo Plugins for simulation interfacing through ROS2 as middleware.

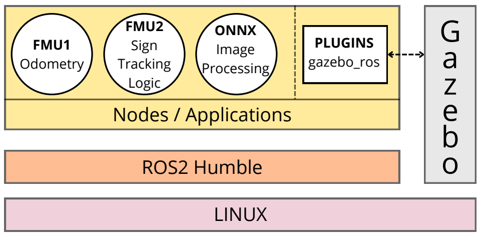

Figure 11. System architecture in a Linux environment.

4.5.2 Integration Implementation

The integrated system is deployed as follows:

1. **Component Deployment:** FMUs are executed within dedicated ROS2 nodes, the ONNX model is deployed within a computer vision node, and Gazebo interfaces with these components via the ROS2 bridge.

2. **Data Flow Management:** As shown in Figure 12, the system maintains synchronized communication through topic-based message passing between nodes, enabling coordinated execution of FMUs and ONNX models.

This integration setup shows the practical feasibility of combining ROS2, FMUs, ONNX models, and Gazebo within a Linux-based open-source environment. The resulting system supports modular development, clear separation of concerns, and synchronized communication across components, aligning with the principles outlined in our proposed methodology.

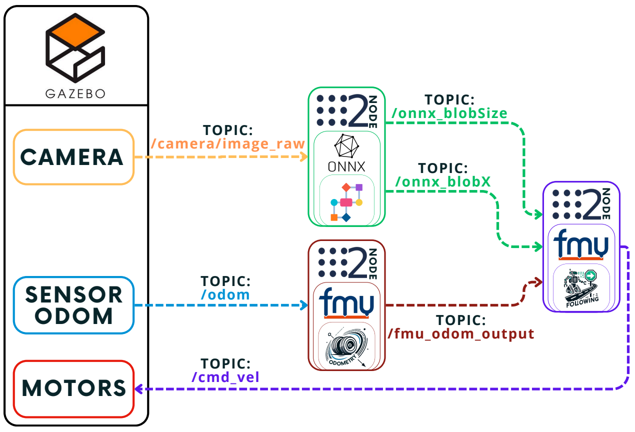

Figure 12. Data flow diagram inter-node communication.

5 Results and Discussion

This section presents and analyzes the results from our implementation, evaluating both the performance of individual components and the overall system integration. In addition to the standard performance metrics, we compare the original outputs of the FMU and ONNX models with those obtained after integration within the ROS2 nodes, thereby validating that our integration preserves the model characteristics and enhances system's interoperability.

5.1 Odometry FMU Performance Analysis

To validate the accuracy of our solution, we compared the roll angle output from the original model with that of our integrated FMU-ROS2 implementation. The roll angle was chosen because it requires quaternion-to-Euler conversion, while x and y coordinates pass directly through the system without transformation. This comparison is essential for verifying that the mathematical operations and coordinate transformations are preserved after integration. Figure 13 shows the comparison of roll angle signals.

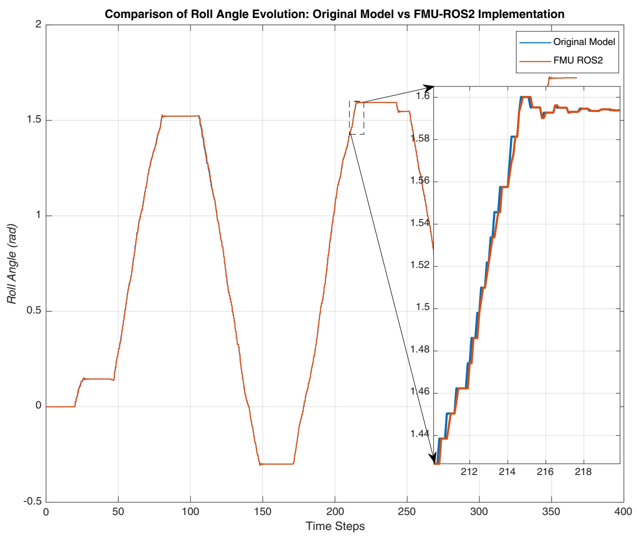

Figure 13. Comparison of roll angle evolution between the original model and FMU-ROS2 implementation.

From this comparison, we can conclude that the quaternion-to-Euler conversion accuracy is maintained. The signals exhibit slight variations, specifically on the order of 0.002 rad. These variations are acceptable, as the operational range goes from -0.4 rad to 1.8 rad. They are likely caused by differences in floating-point computations or by the overhead introduced during ROS 2 message passing.

5.2 ONNX Vision Processing Performance

To validate the accuracy of our ONNX implementation for sign detection, we compared the blobSize(3) output of the original ONNX model with our ONNX-ROS2 solution. This output was chosen as a representative measure since similar detection patterns are observed in other components of the blobSize and blobX vectors. Figure 14 shows the blob size signals.

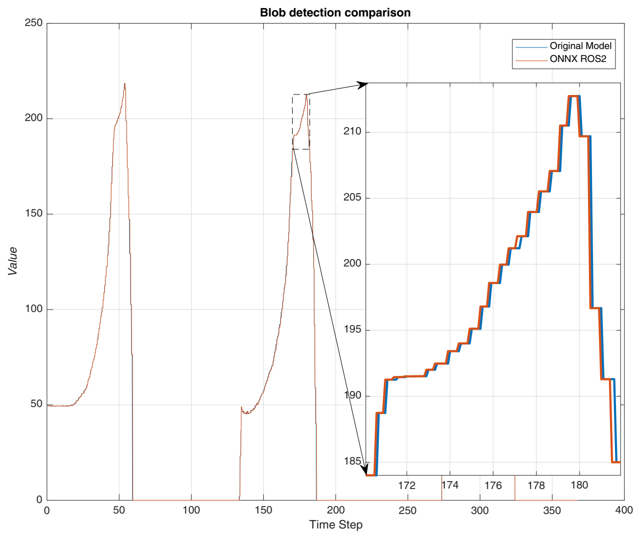

Figure 14. Comparison of blob detection between the original model and the ONNX-ROS2 implementation.

The comparison reveals that the detection accuracy is preserved. Similar to previous comparison, the signals also exhibit slight variations, specifically on the order of 1 pixel. These variations are acceptable, as the operational range goes from 0 to 215 pixels. They are likely caused by differences in floating-point computations or by the overhead introduced during ROS 2 message passing.

5.2.1 ROS2-Based System Integration

The overall system integration was verified using rqt, ROS2's native introspection and visualization tool. Figure 15 displays the runtime communication graph generated by rqt, validating the proposed system architecture as depicted in Figure 12.

The comparison between the proposed architecture and the actual implementation shows successful architecture validation, including the establishment of all intended communication pathways, correct node relationships, and proper topic naming and message routing. In terms of sys-

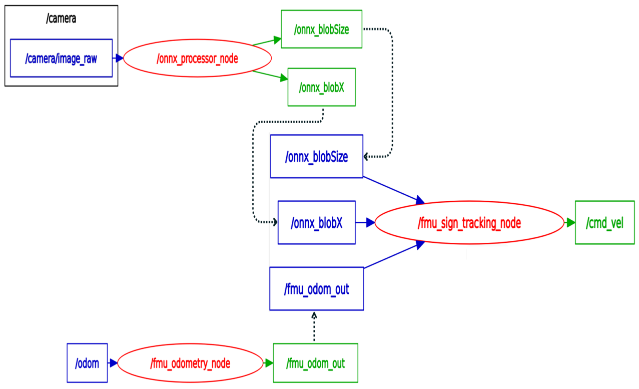

Figure 15. Runtime communication graph generated by rqt, showing node interactions and topic connections in the ROS2 environment.

tem integration, the data flow between the ONNX vision processing and FMU nodes is effective, and synchronization between odometry and control components is maintained, ensuring reliable message passing across the system.

5.2.2 Trajectory Analysis and Navigation Performance

To evaluate the effectiveness of the integrated system, we first analyzed the robot's trajectory through its X-Y position data. Figure 16 shows the complete path, where we can identify the navigation phases.

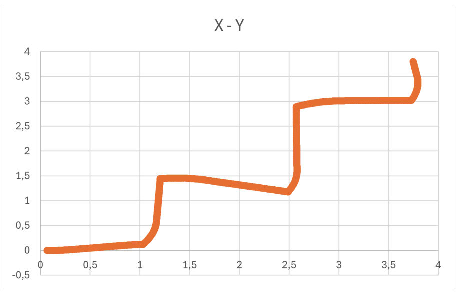

Figure 16. X-Y position plot showing the robot's trajectory.

When this recorded trajectory (orange line) is overlaid onto the simulation environment, as shown in Figure 17, we can verify the robot's successful navigation from point A to point B while following the traffic signs.

The comparison between the expected path (yellow dashed line in Figure 3) and the actual trajectory demonstrates that the robot successfully executed all required turns and maintained proper alignment during straight segments. The transitions observed in both representations validate the effectiveness of the SignTrackingLogic FMU in interpreting visual signals and generating appropriate control commands.

6 Conclusions and Future Work

In this work, we propose a cost-effective, modular, and interoperable methodology for developing CPSs using FMI

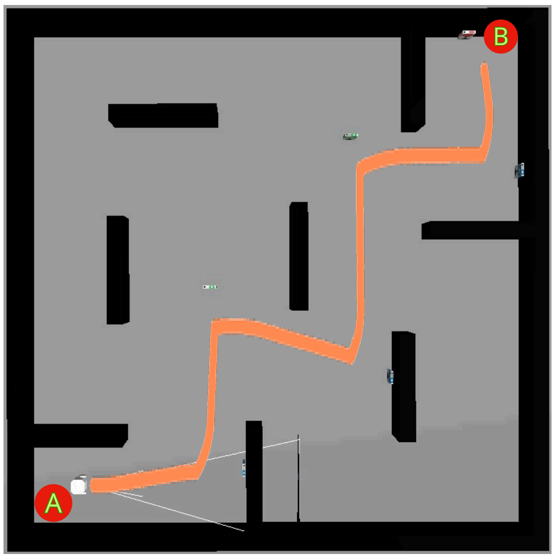

Figure 17. Actual trajectory (orange line) overlaid on the simulation environment, demonstrating precise path following and successful navigation through the course.

2.0 standard. We tested our solution with an autonomous robot navigation use case, integrating FMUs and ONNX models to establish the robot's behavior. The experimental results demonstrate that the system maintains precision while ensuring component independence. Specifically, our tests show accurate odometry tracking, reliable vision-based detection, and control behavior evidenced by consistent velocity patterns and effective trajectory adjustments.

The modular design, which wrappers FMUs and ONNX models as independent ROS2 nodes, offers significant benefits in terms of scalability, maintainability, and interoperability. The implementation demonstrates robust performance in both simulation and real-world environments, with the FMU-based control loop and ONNX inference running effectively within the computational constraints of the embedded platform. The ROS2 middleware proves efficient in handling the communication between components, maintaining system stability and responsiveness throughout operation.

The successful transition from simulation to physical implementation validates the practical applicability of our approach. As shown in Figure 18, while the real environment didn't exactly match the virtual scenario's measurements, the system demonstrated robust adaptability. This flexibility stems from the algorithm's fundamental reliance on traffic sign recognition and following, enabling effective operation in any environment where the standardized signs are present.

By relying on open and interoperable standards, our approach allows easy module replacement and simplifies system updates. This openness not only removes dependencies on proprietary tools but also fosters collaboration among robotic system developers. Moreover, our methodology significantly reduces development time and costs by

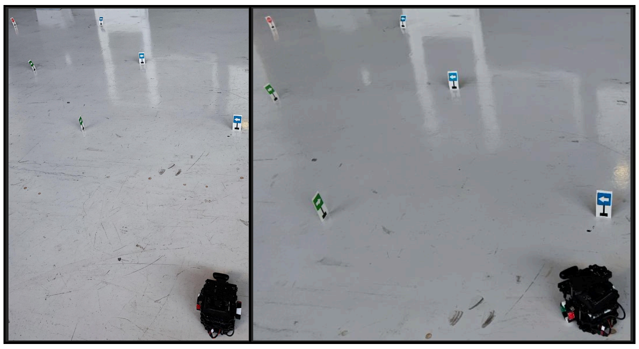
Figure 18. Implementation in physical robot demonstrating system adaptability in real-world conditions

allowing seamless code reuse between the simulation environment and the final hardware deployment. The same ROS2-based modules used for testing and validation in simulation can be directly integrated into the target robotic platform, eliminating the need for reimplementation.

Regarding future research, we identified the following aspects:

- **Advanced AI algorithms:** Development of efficient convolutional neural networks through dynamic neural architecture search and adaptive pruning techniques, focusing on minimizing computational overhead while preserving detection accuracy in vision-based navigation tasks.

- **Use of Zenoh as communication middleware:** Integration of the Zenoh protocol to enhance real-time data distribution in distributed CPS deployments. Zenoh's inherent support for time-sensitive networking and decentralized pub/sub patterns could reduce end-to-end latency while improving interoperability across heterogeneous hardware.

Overall, the current results suggest that our architecture is well-suited to accommodate such extensions while maintaining its core reliability and performance. This work opens new paths for integrating simulation, control, and AI-driven perception, contributing to the advancement of autonomous robotic systems.

References

Amorim, Tiago (2019-01). "Strategies and Best Practices for Model-Based Systems Engineering Adoption in Embedded Systems Industry". In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. URL: https://www.academia.edu/103032710/Strategies_and_Best_Practices_for_Model_Based_Systems_Engineering_Adoption_in_Embedded_Systems_Industry (visited on 2025-04-04).

Bardaro, Gianluca et al. (2017-07). "Using Modelica for advanced Multi-Body modelling in 3D graphical robotic simulators". en. In: pp. 887–894. DOI: 10.3384/ecp17132887. (Visited on 2025-04-13).

Corallo, Angelo et al. (2022). "Evaluating the Maturity of MBE Application". en. In: *Product Lifecycle Management. Green and Blue Technologies to Support Smart and Sustainable Organizations*. Ed. by Osiris Canciglieri Junior et al. Cham: Springer International Publishing, pp. 401–415. ISBN: 978-3-030-94335-6. DOI: 10.1007/978-3-030-94335-6_29.

Cornille, Tobias (2024). *Software Tools For Robotics Landscape (2024)*. Segments.ai. Accessed: Mar. 04, 2025. URL: https://segments.ai/blog/software-tools-for-robotics-landscape/.

Haessler, Philipp, Ferran Giones, and Alexander Brem (2023-03). "The who and how of commercializing emerging technologies: A technology-focused review". In: *Technovation* 121, p. 102637. ISSN: 0166-4972. DOI: 10.1016/j.technovation.2022.102637. (Visited on 2025-04-04).

Holtmann, Jörg, Grischa Liebel, and Jan-Philipp Steghöfer (2024-04). "Processes, methods, and tools in model-based engineering—A qualitative multiple-case study". In: *Journal of Systems and Software* 210, p. 111943. ISSN: 0164-1212. DOI: 10.1016/j.jss.2023.111943. (Visited on 2025-04-01).

El-Hussieny, H. (2024-07). "Real-time deep learning-based model predictive control of a 3-DOF biped robot leg". In: *Scientific Reports* 14.1, p. 16243. DOI: 10.1038/s41598-024-66104-y.

Idoko et al. (2024-11). "Mathematical modeling and simulations using software like MATLAB, COMSOL and Python". In: *Magna Scientia Advanced Research and Reviews* 12, pp. 62–095. DOI: 10.30574/msarr.2024.12.2.0181.

Javaid, Mohd et al. (2023). "An integrated outlook of Cyber–Physical Systems for Industry 4.0: Topical practices, architecture, and applications". In: *Green Technologies and Sustainability* 1.1, p. 100001. ISSN: 2949-7361. DOI: https://doi.org/10.1016/j.grets.2022.100001.

Khandoker, Azad et al. (2022). "Towards a logical framework for ideal MBSE tool selection based on discipline specific requirements". In: *Journal of Systems and Software* 189, p. 111306. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2022.111306.

Lange, Ralph et al. (2020-08). "Integrating the Functional Mock-Up Interface with ROS and Gazebo". In: pp. 187–231. ISBN: 978-3-030-45955-0. DOI: 10.1007/978-3-030-45956-7_7.

Licardo, Josip Tomo, Mihael Domjan, and Tihomir Orehovački (2024). "Intelligent Robotics—A Systematic Review of Emerging Technologies and Trends". In: *Electronics* 13.3. ISSN: 2079-9292. DOI: 10.3390/electronics13030542.

MathWorks (2024). *Sign-Following Robot with ROS 2 in Simulink*. URL: https://es.mathworks.com/help/ros/ug/sign-following-robot-using-ros2-simulink.html (visited on 2025-03-03).

Modelica Association (2025). *Functional Mock-up Interface*. Accessed: Mar. 07, 2025. URL: https://fmi-standard.org/.

Moshood, Taofeeq D. et al. (2024). "Infrastructure digital twin technology: A new paradigm for future construction industry". In: *Technology in Society* 77, p. 102519. ISSN: 0160-791X. DOI: https://doi.org/10.1016/j.techsoc.2024.102519.

Nevliudov, Igor, Oleksandr Tsymbal, and Artem Bronnikov (2021-12). "IMPROVEMENT OF ROBOTIC SYSTEMS BASED ON VISUAL CONTROL". In: *Innovative Technologies and Scientific Solutions for Industries*, pp. 65–74. DOI: 10.30837/ITSSI.2021.18.065.

ONNX community (2025). *Open Neural Network Exchange (ONNX)*. [Accessed 20-03-2025]. URL: https://onnx.ai/.

Open Robotics (2025a). *Gazebo*. Accessed: Mar. 07, 2025. URL: https://gazebosim.org/home.

Open Robotics (2025b). *ROS2 Humble*. URL: https://docs.ros.org/en/humble/index.html (visited on 2025-03-25).

Oudart, D. et al. (2020). “A Model based Toolchain for the Cosimulation of Cyber-physical Systems with FMI”. In: *Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development*. Valletta, Malta, pp. 15–25. DOI: 10.5220/0008875400150025.

Parwej, Firoj (2024-05). “The Implementation of NGCPS to Integrate AI in Order to Get the Accurate Decision Making”. In: *2024 4th International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pp. 1300–1305. DOI: 10 . 1109 / ICACITE60783 . 2024 . 10616671. (Visited on 2025-04-13).

Patel, Vatsal, Minas Liarokapis, and Aaron Dollar (2022-01). “Open Robot Hardware: Progress, Benefits, Challenges, and Best Practices”. In: *IEEE Robotics and Automation Magazine* PP, pp. 2–29. DOI: 10.1109/MRA.2022.3225725.

Patrício, Leonel, Leonilde Varela, and Zilda Silveira (2024). “Integration of Artificial Intelligence and Robotic Process Automation: Literature Review and Proposal for a Sustainable Model”. In: *Applied Sciences* 14.21. ISSN: 2076-3417. DOI: 10.3390/app14219648.

Ray, Douglas and Jose Ramirez-Marquez (2020-01). “A framework for probabilistic model-based engineering and data synthesis”. In: *Reliability Engineering & System Safety* 193, p. 106679. ISSN: 0951-8320. DOI: 10 . 1016 / j . ress . 2019 . 106679. (Visited on 2025-04-13).

Robles, Julia, Cristian Martín, and Manuel Díaz (2023). “OpenTwins: An open-source framework for the development of next-gen compositional digital twins”. In: *Computers in Industry* 152, p. 104007. ISSN: 0166-3615. DOI: https://doi.org/10.1016/j.compind.2023.104007.

Santos, Adriano et al. (2024-11). “Integration of Artificial Vision and Image Processing into a Pick and Place Collaborative Robotic System”. In: *Journal of Intelligent and Robotic Systems* 110. DOI: 10.1007/s10846-024-02195-z.

Sekar, P. K. M. and J. S. Baras (2022-07). “Model-Based Systems Engineering Simulation Framework for Robot Grasping”. In: *INCOSE International Symposium*. Vol. 32. S2, pp. 82–89. DOI: 10.1002/iis2.12898.

Shahsavari, Sajad et al. (2021-06). *MCX An Open-Source Framework For Digital Twins*. DOI: 10.7148/2021-0119.

Soori, Mohsen, Behrooz Arezoo, and Fooad Karimi Ghaleh Jough (2024-01). “Intelligent Robotic Systems in Industry 4.0, A Review”. In: *Journal of Advanced Manufacturing Science and Technology*, p. 2023. DOI: 10 . 51393 / j . jamst . 2024007.

Xu, Haowen et al. (2024-10). “Leveraging generative AI for urban digital twins: a scoping review on the autonomous generation of urban data, scenarios, designs, and 3D city models for smart city advancement”. In: *Urban Informatics* 3.1, p. 29. ISSN: 2731-6963. DOI: 10.1007/s44212-024-00060-w.

Yue, L. (2024-08). *Design of Real-Time Dynamic Motion Compensation in Vision-Guided Robots for Biopsy Needle Alignment*. URL: https://essay.utwente.nl/103243/1/LI_MA_EEMCS.pdf.