EVOLUTION IN ELECTRICAL AND ELECTRONIC ENGINEERING
e-ISSN: 2756-8458

EEEE

Vol. 7 No. 1 (2026) 33-38
https://publisher.uthm.edu.my/periodicals/index.php/eeee

Development of a Mobile Robot Based on ROS2 Architecture

Intan Nur Azreena Jamaluddin[1], Hisyam Abdul Rahman[1]*

[1] *Faculty of Electrical and Electronic Engineering*
Universiti Tun Hussein Onn Malaysia, Batu Pahat, Johor, 86400, MALAYSIA

*Corresponding Author: arhisyam@uthm.edu.my
DOI: https://doi.org/10.30880/eeee.2026.07.01.005

Article Info

Received: 12 February 2026
Accepted: 9 April 2026
Available online: 30 April 2026

Keywords

Autonomous Mobile Robot, ROS 2, SLAM, LiDAR, Navigation 2.

Abstract

The primary aims of this work were to create an autonomous path planner for an unknown environment and to implement Simultaneous Localisation and Mapping (SLAM), both achieved through the development and construction of a differential-drive mobile robot. Its approach is step-by-step, starting with kinematic and sensor validation of a Unified Robot Description Format (URDF) model in a Gazebo simulation environment, followed by physical hardware integration. The hardware is designed around the Raspberry Pi 4 Model B as the central processing unit, with the Arduino Nano connected via a serial bridge for low-level motor control. The sensor is the RPLIDAR A1, which perceives the environment. It is built on the ROS 2 Humble Hawksbill release, the SLAM Toolbox to create a 2D occupancy grid, and Navigation 2 (Nav2) to plan both global and local paths. The experimental results demonstrate that the system generates high-quality static maps and autonomously navigates while avoiding dynamic obstacles. Lastly, this low-cost, easily copiable platform offers a handy testbed and has a high likelihood of producing educational robotics and applied ROS 2 research.

1. Introduction

Autonomous Mobile Robots (AMRs) are making industries more adaptable with flexible infrastructure-free automation. Unlike traditional Automated Guided Vehicles (AGVs), which use fixed magnetic tapes or wires, AMRs use built-in sensors and processors to navigate unstructured environments without human guidance. These robots are primarily used in industrial environments for material handling and surveillance, improving operational efficiency and enabling adaptation to changing workflows.

Despite their usefulness, their widespread use in academic research is constrained by the prohibitive cost of these proprietary platforms. Many institutions are financially constrained from hands-on experimentation and from democratising advanced robotics knowledge. In addition, navigating dynamic environments is a great technical challenge. Standard navigation stacks often assume their world is static, i.e., that the movement of objects (humans or other machinery) can lead to localisation drift and potential collisions.

To address these issues, this research develops an open-source AMR platform that balances cost and performance. The system takes advantage of the ROS 2 Humble system, which provides a decentralised architecture based on the Data Distribution Service (DDS) for reliable communication of real-time data between nodes. Consequently, the specific goals of this research study are to develop an indoor mobile robot (on a Raspberry Pi platform and the ROS 2 framework) and, later, to implement SLAM techniques for real-time mapping and localisation in indoor environments.

2. Methodology

The URDF model is then validated in the Gazebo simulation environment to determine the robot's kinematic behaviour, sensor configuration, and control logic, and to physically implement it. Once a successful simulation is completed, the physical chassis assembly is performed. The Raspberry Pi 4B is the central processing unit, connected to an RPLIDAR A1 sensor and wheel encoders via an Arduino Nano microcontroller using a serial bridge communication protocol. The software architecture includes ROS2Control for low-level hardware abstraction, SLAM Toolbox for simultaneous localisation and mapping, and Nav2 for autonomous route planning.

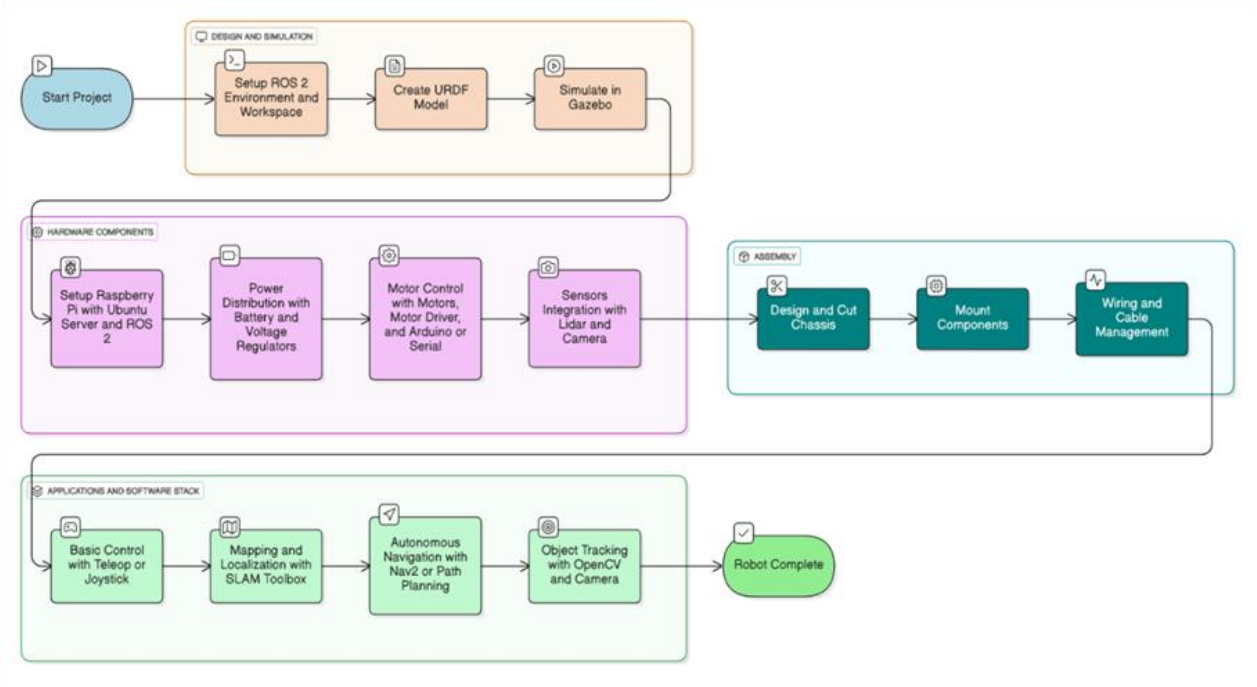


Fig. 1 *The overall project flow*

The Fig. 1 shows the extensive development process used in this project for designing and implementing an autonomous mobile robot based on ROS 2. The project starts with the design and simulation phase, in which the ROS 2 environment and workspaces are created, and the robot model is developed by using Unified Robot Description Format (URDF). The URDF model is validated in both the Gazebo simulation environment and in the sensor and control logic configuration for the robot before the physical implementation. Once it has been confirmed that the simulation result meets the required standards, the procedure proceeds to the stage involving hardware components.

This includes the setup of the Raspberry Pi, where Ubuntu Server and ROS 2 were used as the main processing unit. This phase includes the supplementary elements of the power distribution phase, including further steps such as integrating motor control with a motor driver and a microcontroller, as well as installing sensors, such as LiDAR and a camera. The following stage focuses on assembling the robot's components, including constructing the core, installing components on the core, and organising the wiring to ensure optimal operation. The end phase consists of applications and software, beginning with basic remote control, using the SLAM toolbox for mapping and localisation, then self-driving with the Nav2 stack, and finally object tracking with OpenCV. The successful completion of these stages enables the development of autonomous, functional mobile robots that can move within and interact with their environment.

The detailed block diagram of the autonomous mobile robot developed as a project is shown in Fig. 2. The power subsystem, the computing unit, the low-level control unit, and the actuation and sensing units are the main four subsystems of the system. The power subsystem consists of an 11.1V LiPo battery that powers the entire system. A buck converter reduces the battery voltage to 5V, which then supplies a safe amount of power to the Raspberry Pi 4B and all active sensors. Meanwhile, the DC motors have their own power line supplied to the motor driver. The computing unit is around the Raspberry Pi 4B, serving as the primary controller running Ubuntu and ROS 2. The information obtained from the LiDAR sensor and camera module will be processed by the Raspberry Pi to enable perception, mapping, localisation, and navigation.

Evolution in Electrical and Electronic Engineering Vol. 7 No. 1 (2026) p. 33-38

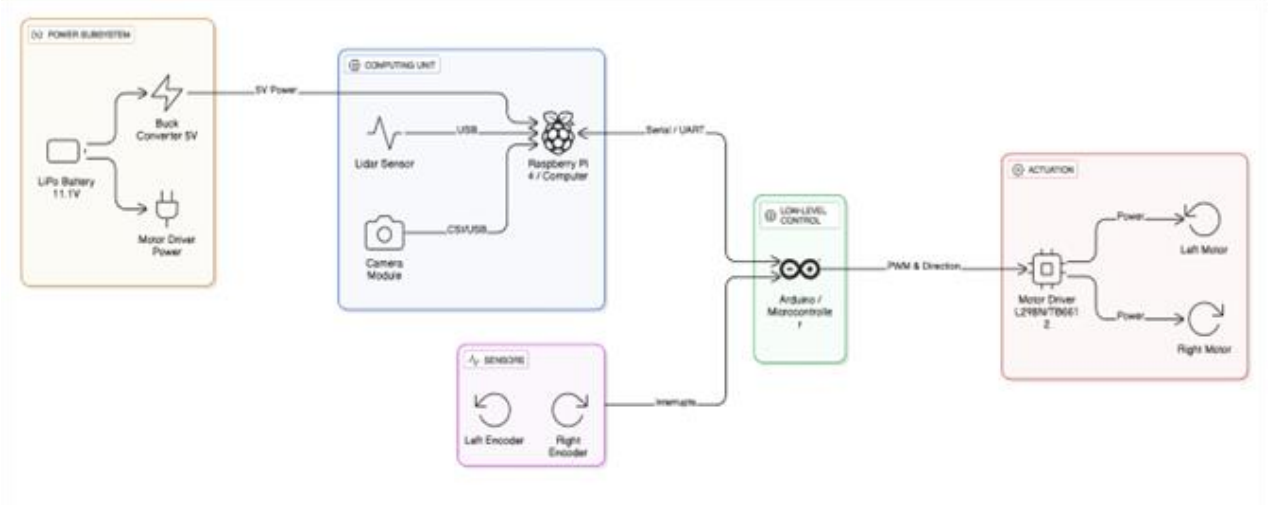


Fig. 2 *Block Diagram of the system flow*

Velocity commands are generated from the navigation output and sent to the Arduino Nano via serial (UART) communication. A low-level control unit based on an Arduino Nano microcontroller controls the motors in real time and takes encoder feedback. The Arduino receives the high-level velocity from the Raspberry Pi and uses it to send a pulse-width-modulated signal with the correct direction. Signs from the left and right motor encoders are read via hardware interrupts to detect wheel speed and collect odometry data. The actuation unit consists of an L298N motor drive unit connected to both encoder motors, the left and right. The motor driver functions well as an amplifier, taking the control signals from the Arduino and amplifying them so the motors can achieve differential-drive motion. This structured approach separates higher-level decision-making from lower-level motor controls to improve system reliability and real-time performance.

Fig. 3 shows that, at the perception layer, sensor data is processed in the SLAM Toolbox to perform Simultaneous Localisation and Mapping. Utilising both LiDAR scan matching and odometry feedback, the SLAM algorithm essentially constructs a map of the environment by forming an occupancy grid and simultaneously estimating the robot's pose (position and orientation) within it. The dual output, consisting of a map and the robot's pose, is of great importance because it enables the robot to understand both the environment's structure and its position within it. Simultaneously, object detection modules can be used in the processing of visual information to detect obstacles or features that improve environmental awareness, in addition to static mapping. Once a map is available, the system switches to the planning layer, which can be used for localisation and navigation decisions.

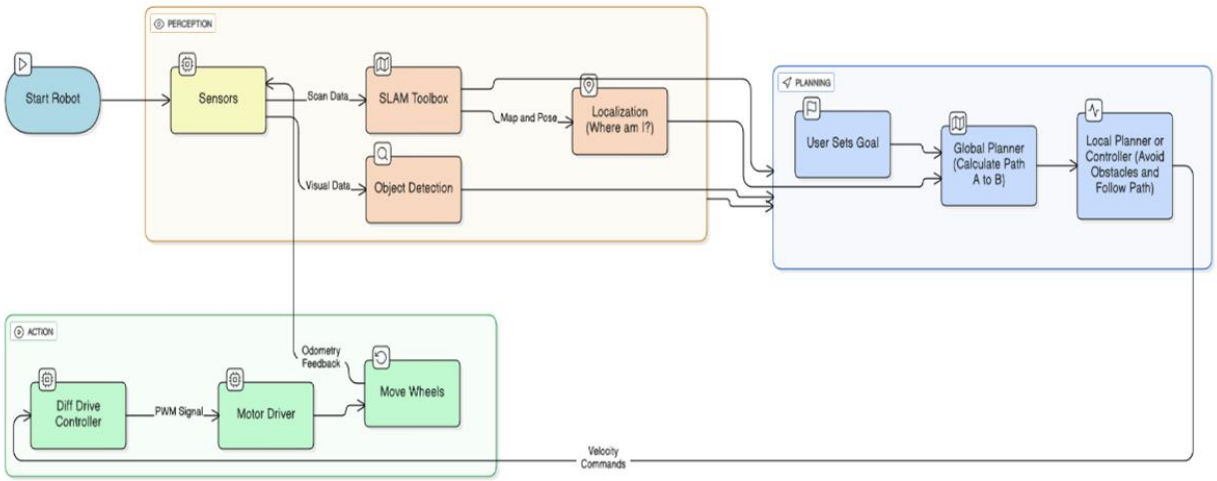


Fig. 3 *The process flow of mapping*

The localisation module deals with the problem of determining where the robot is in the world with respect to the map that was created, the basic question of "Where am I?" Utilising this information, the operator can establish a navigation objective. The global planner then calculates the most efficient route to the intended destination from its current location, using the static map as a reference. This proposed route provides a strategic

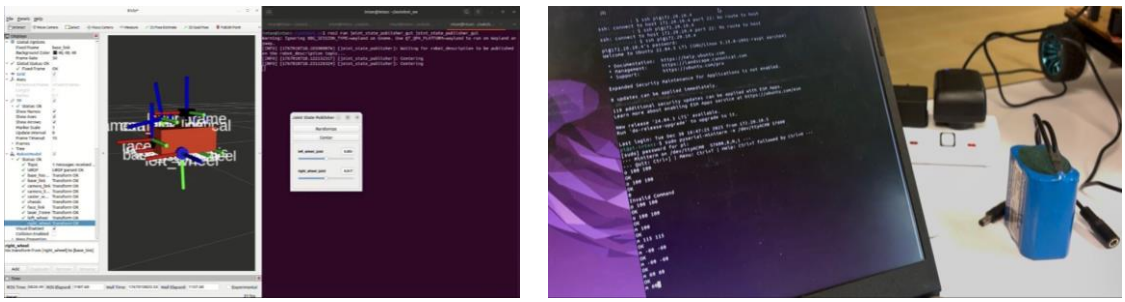
overview, avoids identified barriers, and minimises necessary travel expenses. execution of the planned motion is carried out by the action layer through closed-loop control.

The local planner or controller is responsible for converting the global path into velocity commands and issuing them in real time, while actively avoiding obstacles using sensors. The velocity commands are sent to the differential drive controller, which converts them into PWM signals for the motor driver. The motors are used to engage the wheels as required, and odometry feedback is regularly provided to the perception layer, thereby completing the feedback loop. This iterative process allows the robot to generate a map of its location, identify safe paths, and move itself in an unknown environment.

3. Result and Discussion

3.1 Analysis of Verification of Kinematics and Sensors

As preparation for autonomous trials, the robot's Unified Robot Description Format (URDF) model was validated in a Gazebo simulation. The check of the physical hardware integration was obtained with teleoperation tests. The encoder polarity was then changed to ensure that rotation of the forward wheel matched positive tick increments, meeting the right-hand rule requirements for ROS 2 odometry. Fig. 4 demonstrates the preliminary validation procedures conducted prior to autonomous navigation experiments, including verification of the robot's URDF kinematic model in RViz and testing of serial communication, motor control, and encoder polarity consistency for ROS 2 odometry integration.

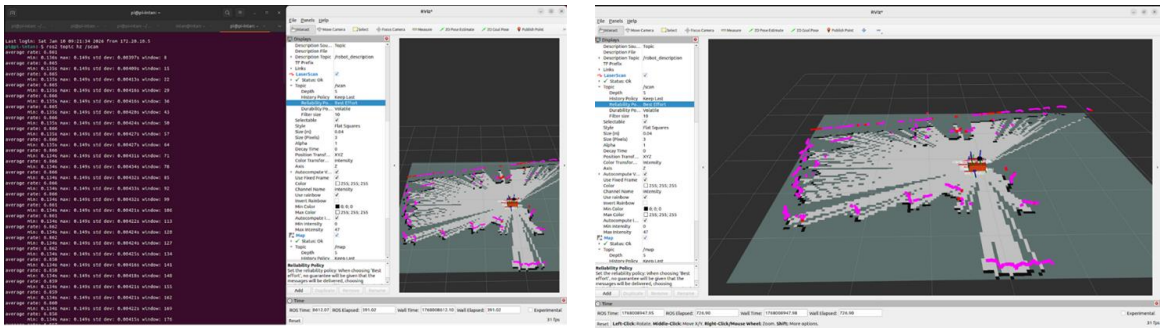


(a) **(b)**

Fig. 4 *(a) URDF Model Verification and Joint Articulation Test in Rviz; (b) Serial Interface Communication used for Motor and Encoder Testing*

3.2 Analysis of Precision of SLAM Mapping

The SLAM Toolbox was tested in a controlled laboratory environment and produced a high-resolution 2D occupancy grid map. To assess the interface's reliability, communication via the serial bridge between the Raspberry Pi 4B and the Arduino Nano was tested, with an average command latency of approximately 15 ms. The encoder feedback was estimated to be 95 per cent accurate in a continuous test distance of 5 meters. Fig. 5 shows a robot equipped with LiDAR moving around an environment while the ROS SLAM Toolbox continuously builds a live navigational map and estimates the robot's position for autonomous movement.



(a) **(b)**

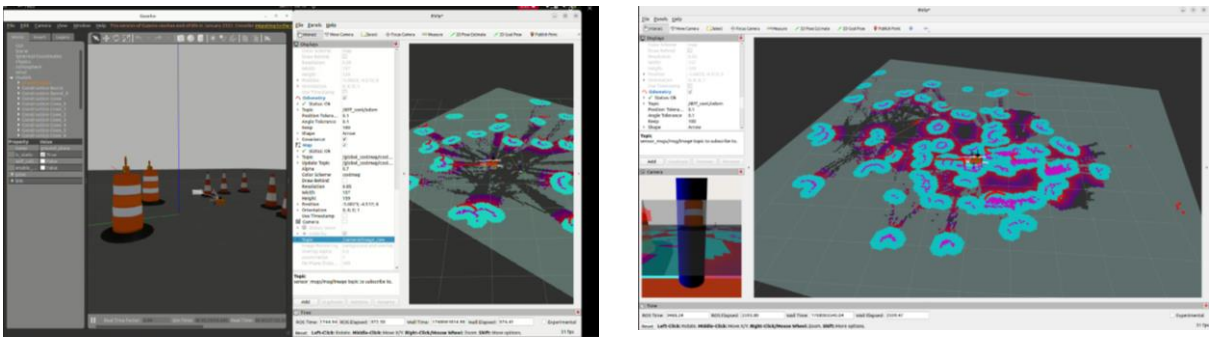
Fig. 5 *(a) The process of mapping robot with LiDAR; (b) SLAM Toolbox Real-time generation of a global costmap*

3.3 Independent Navigation

When using the Nav2 stack for autonomous navigation, the robot successfully planned and executed paths from a starting point to a specified final orientation. The system achieved an estimated path efficiency of 85% and a total

Evolution in Electrical and Electronic Engineering Vol. 7 No. 1 (2026) p. 33-38

goal arrival success rate of 90 in a series of 10 independent navigation trials. The DWB local planner developed safety barriers around obstacles using inflation layers, and the robot dynamically changed its path to avoid collisions whilst following its global path. As shown in Fig. 6, the successful integration of simulated environment modelling, robot perception, and real-time local navigation planning within the Nav2 autonomous navigation framework.



(a) **(b)**

Fig. 6 *(a) The coherence of both the simulated physical environment and internal perception model of the robot; (b) The generated local costmap*

4. Conclusion

The findings of this paper suggest that the proposed architecture is implementable to support autonomous indoor navigation. It turned out that the Raspberry Pi 4B could be the heart of the computation and, in fact, meet the computational needs of the ROS 2 Humble ecosystem. The pulse width modulation (PWM) generation and encoder reading were connected to an Arduino microcontroller. This solution provided a stable serial bridge architecture, ensuring that real-time requirements were met without placing excessive load on the underlying host operating system. This bi-directional architecture has been very useful for user-friendly mobile robotics.

Secondly, the project has found that the SLAM Toolbox is very helpful for developing appropriate 2D maps in a stationary setting using LiDAR data alone. The robot was able to navigate an unfamiliar-sized room, track its route, and construct a map that reflected the physical layout of the test room. The second was the development of the Adaptive Monte Carlo Localisation (AMCL) algorithm, which enabled the robot to determine its position in this map with very high accuracy. Therefore, the sensor-fusion technique used to combine wheel odometry and laser-scanning data was worth the effort.

Thirdly, the Nav2 stack architecture was a stable point-to-point autonomous navigation system. The findings indicate that the costmap parameters, particularly the obstacle inflation radius, are significant for the safety of the navigation process. The trial-and-error approach enabled the robot to navigate narrow corridors and avoid obstacles, demonstrating that the DWB (Dynamic Window Approach) local planner is highly effective for differential-drive robots of this size (30 cm x 21 cm). The elements of navigation are also exploited to the maximum to achieve the lowest intelligent elements of the project.

Lastly, the manual approach to bypassing the algorithms and a convenient debugging tool was the default Xbox 360 controller. It also ensured that the motor drivers and motor settings were correctly polarised before enabling the autonomous algorithms. The fact that the teleoperation and autonomous modes can be easily switched is a shining example of the power of the state machine logic in the software stack. The project has already demonstrated a working electronic prototype, showing that complex autonomous behaviours are possible using readily available off-the-shelf components.

The DWB local planner performed fairly well in this differential-drive robot case, but future research can compare its performance with other algorithms, such as the Timed Elastic Band (TEB) local planner, to understand how trajectories can be optimised in more constrained or complex environments. Since then, the system has been constrained by the fact that it can only utilise 2D LiDAR, which may not be useful in high-reflectivity conditions; 3D vision cameras can significantly improve perception. Finally, it is a low-cost platform that can be easily duplicated and is highly important to learning institutions. This will allow students and researchers to gain hands-on experience with advanced ROS 2 integration, bridging the theory-to-practice gap in robotics without the prohibitive cost of commercial industrial platforms.

Acknowledgement

The authors would like to thank the Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, for its support.

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: study conception and design: Intan Nur Azreena Jamaluddin, Hisyam Abdul Rahman; data collection: Intan Nur Azreena Jamaluddin; analysis and interpretation of results: Intan Nur Azreena Jamaluddin; draft manuscript preparation: Intan Nur Azreena Jamaluddin. All authors reviewed the results and approved the final version of the manuscript.

References

[1] X. Zhao and T. Chidambareswaran, "Autonomous Mobile Robots in Manufacturing Operations," in IEEE International Conference on Automation Science and Engineering, IEEE Computer Society, 2023. doi: 10.1109/CASE56687.2023.10260631.

[2] L. Nwankwo, C. Fritze, K. Bartsch, and E. Rueckert, "ROMR: A ROS-based open-source mobile robot", doi: 10.17605/OSF.IO/K83X7.

[3] Z. Xie, P. Xin, and P. Dames, "Towards Safe Navigation Through Crowded Dynamic Environments," in IEEE International Conference on Intelligent Robots and Systems, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 4934–4940. doi: 10.1109/IROS51168.2021.9636102.

[4] S. Macenski and I. Jambrecic, "SLAM Toolbox: SLAM for the dynamic world," J. Open Source Softw., vol. 6, no. 61, p. 2783, May 2021, doi: 10.21105/joss.02783.

[5] S. Macenski, F. Martin, R. White, and J. G. Clavero, "The marathon 2: A navigation system," in IEEE International Conference on Intelligent Robots and Systems, Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 2718–2725. doi: 10.1109/IROS45743.2020.9341207.

[6] T. Kronauer, J. Pohlmann, M. Matthe, T. Smejkal, and G. Fettweis, "Latency Analysis of ROS2 Multi-Node Systems," in IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/MFI52462.2021.9591166.