Enhancing Robotics Education with ROS2: Curriculum Design and Teaching Methods

Dr. Jose James, Lawrence Technological University

Dr. Jose James is an Assistant Professor in the A. Leon Linton Department of Mechanical, Robotics, and Industrial Engineering at Lawrence Technological University. His research is centered at the intersection of human-robot interaction, haptics, virtual reality (VR), and artificial intelligence (AI). A key focus of his work is pioneering innovative teaching methods in robotics education, where he actively develops curricula that integrate these cutting-edge technologies. Notably, he has designed and introduced new, specialized courses in both Haptics and Virtual Reality and ROS2 for the university's undergraduate and graduate robotics programs. Holding a Ph.D. in Haptics and Robotics Engineering, Dr. James also brings valuable industry experience in developing VR and haptic-based simulators for motor skill training. He is committed to bridging the gap between academic theory and industry needs, preparing the next generation of engineers for successful careers in the rapidly evolving field of robotics.

Dr. James A. Mynderse, Lawrence Technological University

James A. Mynderse, PhD is a Professor in the A. Leon Linton Department of Mechanical, Robotics, and Industrial Engineering at Lawrence Technological University. He serves as director for the BS in Robotics Engineering and MS in Mechatronics and Robotics Engineering programs.

Enhancing Robotics Education with ROS2: Curriculum Design and Teaching Methods

Abstract

This paper presents the development and integration of a new "Robot Operating System 2 (ROS2)" course into the College of Engineering curriculum, replacing the outdated ROS1 course for BS Robotics Engineering and MS Mechatronics and Robotics Engineering programs at Dept. of Mechanical, Robotics & Industrial Engineering in the Lawrence Technological University. The transition was motivated by the approaching End of Life (EOL) of ROS1 Noetic in May 2025 and the enhanced capabilities of ROS2 in real-time performance, security, and modularity. The course covers fundamental topics such as ROS2 architecture, ROS2 programming in Python and C++, and advanced topics in robot control, navigation, and simulation. It also includes practical knowledge on integrating with simulation tools like Gazebo and TurtleBot, and interfacing with MATLAB and Simulink. The curriculum emphasizes hands-on labs and projects, utilizing both simulations and real-world robotic systems to provide practical experience. This approach allows students to apply theoretical concepts to real-world scenarios, enhancing their problem-solving skills and technical proficiency. The course's interdisciplinary nature fosters collaboration between mechanical engineering, computer science, and electrical engineering students, mirroring industry practices. By integrating ROS2 into the curriculum, the program enhances industry relevance and career preparation, develops essential technical skills, and accelerates learning and project development. Students gain experience in creating scalable robotics software, integrating diverse components, and developing real-time systems for critical applications. This comprehensive approach to robotics education not only enhances employability but also prepares students to tackle complex challenges in the rapidly evolving field of robotics, positioning them for success in their future careers and enabling them to contribute to advancements in the field.

1. The Transition to ROS2 in Robotics Education

Robotics education has evolved rapidly over the last decade, catalyzed by breakthroughs in autonomy, sensing, and intelligent systems. At the heart of this evolution is the Robot Operating System (ROS), an open-source middleware framework that provides the hardware abstraction and distributed computing necessary for complex system development [1]. While ROS is technically a suite of tools and libraries rather than a traditional OS, its graph-based architecture - where independent "nodes" communicate via message passing - has become the industry standard for scalable robotics [2].

The scheduled End of Life (EOL) for ROS1 Noetic in May 2025 marks a critical turning point for robotics education, making the transition to ROS2 an urgent necessity for modern engineering programs [1]. This next-generation framework was specifically engineered to overcome the architectural limitations of its predecessor, such as the single point of failure inherent in the ROS Master node [3]. By integrating Data Distribution Service (DDS) at its core, ROS2 provides the deterministic, real-time performance required for safety-critical applications and ensures robust security protocols for data encryption and authentication [4]. Furthermore, it simplifies complex operations through native multi-robot communication - a feature lacking in the original ROS design - and offers a significantly smaller footprint, providing optimized support for the resource-constrained embedded systems that drive contemporary edge robotics [1,4].

1.1. Curriculum Design and Tools

In response to the evolving industry landscape, the Dept. of Mechanical, Robotics and Industrial Engineering at Lawrence Technological University has launched a comprehensive ROS2 curriculum designed to replace its legacy ROS1 predecessor. This updated syllabus prioritizes proficiency in professional-grade visualization and debugging utilities - specifically Rviz2, rqt, and ros2cli - while leveraging the Gazebo environment for high-fidelity simulation. To bridge the gap between theoretical frameworks and practical application, the course integrates several advanced instructional strategies. Students engage in cross-platform development by interfacing ROS2 with MATLAB and Simulink, and they participate in hardware-in-the-loop (HIL) exercises that facilitate a seamless transition from TurtleBot simulations to physical robotic deployments [5]. Furthermore, the curriculum is rooted in interdisciplinary collaboration, featuring project structures that mirror professional environments by requiring a synergy of mechanical, electrical, and computer engineering disciplines to solve complex robotic challenges [6-8].

1.2. Student Demographics and Prerequisites

A significant challenge in the curriculum design is managing the disparate backgrounds of the student body, as the course must cater to two distinct cohorts with varying technical strengths. The first group consists of BS Robotics Engineering seniors who enter the course with a robust programming foundation in C, C++, and C#, supported by prior coursework in data structures, discrete math, and embedded systems. In contrast, the MS Mechatronics and Robotics Engineering students often transition from traditional Mechanical Engineering backgrounds; while they possess deep domain knowledge in hardware and physical systems, they frequently have limited experience with high-level languages like Python or C++. To address this gap, the curriculum is strategically structured to accommodate these differing levels of software proficiency, providing the necessary scaffolding for graduate students while maintaining the technical rigor required to ensure all students achieve industry-standard competency in ROS2. This tiered pedagogical approach ensures that students with mechanical backgrounds can bridge the software gap through modular learning paths without hindering the progress of those with advanced computer science skills [9].

1.3. Impact on Career Readiness

Mastering ROS2 provides students with a significant competitive advantage in the job market. From autonomous vehicles and drones to AI-driven manufacturing, industry demand for ROS2-literate engineers is surging. By focusing on scalable software architecture and real-time system integration, this course does more than teach a tool; it prepares students to solve complex, high-stakes challenges in the rapidly shifting robotics landscape. This curriculum directly aligns with the current industrial shift toward "Industry 4.0," where proficiency in middleware frameworks like ROS2 is considered a core competency for systems integration and the deployment of autonomous mobile robots in smart factories [10]. By positioning students at the forefront of this technological shift, the program enables them to contribute immediately to advancements in the field and secures their path toward leadership roles in robotics R&D [11,12].

2. Course Learning Objectives, Curriculum Design, and Development

The ROS2 three credit hour course was designed for senior undergraduate and graduate students in robotics, mechanical engineering, mechatronics, and computer science. No formal prerequisites were required for the first offering, though future iterations will incorporate programming and controls prerequisites. The first offering of this updated Robot Operating System 2 (ROS2) course

was in Fall 2024. Previous offering was ROS1. The course development commenced with a defined set of broad learning objectives, detailed below. A comprehensive list of section-specific learning objectives is supplied to students in the course syllabus. The course objectives for the ROS2 course aim to provide students with a comprehensive understanding of robotics software development and integration.

By the end of the course, students should be able to:

- Explain the architecture and communication mechanisms of ROS2
- Develop ROS2 nodes in Python and C++
- Integrate sensors and actuators into ROS2-based systems
- Utilize ROS2 tools such as rviz, rqt, and ros2cli
- Implement robot navigation and control using ROS2 packages
- Interface ROS2 with simulation tools such as Gazebo and MATLAB/Simulink
- Design and execute a robotics project using ROS2

Table 1: Objectives of the course

ROS2 Fundamentals	Intermediate ROS2 Development	Advanced Robotics Applications
Introduction to ROS and ROS1 vs. ROS2 differences	Launch files and parameter servers	Multi-robot communication
ROS2 architecture (DDS, nodes, topics, services, actions)	TF2 coordinate frames	Real-time control
Workspaces, packages, and build systems (Colcon)	Sensor integration (LiDAR, IMU, cameras)	MATLAB/Simulink integration
ROS2 command-line tools	Actuator control	MoveIt for manipulation
Writing publisher/subscriber nodes in Python and C++	Gazebo simulation & TurtleBot3 navigation stack	ROS2 security features

The course follows a lecture-and-lab model, balancing theory with practice, preparing students for research and industry applications. Weekly, students attend a 75-minute lecture on ROS2 architecture followed by a 75-minute lab for simulation and robot testing. Assessments include a course project. To accommodate professional domestic and international students, sessions are held in the evenings. As detailed in Table 1, the curriculum is partitioned into three modules: ROS2 Fundamentals, Intermediate Development, and Advanced Applications, guiding students from basic messaging to complex autonomous behaviors [13].

2.1. ROS2 Fundamentals

The introductory module, ROS2 Fundamentals, establishes a robust theoretical and practical foundation by examining the architectural shift from legacy ROS1 to the modern, DDS-based framework of ROS2. Students explore the core computational graph, mastering the lifecycle of nodes and the primary communication modalities, including asynchronous topics, synchronous services, and preemptible actions. Technical training emphasizes the Colcon build system and workspace management alongside the ros2cli command-line suite for real-time system introspection. In the laboratory, students apply these concepts by developing their first publisher

and subscriber nodes in both Python and C++, using rviz to visualize live data streams and ensure proper communication across the computational graph.

2.2. Intermediate ROS2 Development

Building upon these foundations, the Intermediate ROS2 Development module shifts the focus toward system integration and the practical development of mobile robot platforms. This stage introduces the use of launch files for complex multi-node orchestration and the parameter server for dynamic configuration management. A significant portion of the coursework is dedicated to the TF2 transform library, which is essential for managing the spatial relationships between robot coordinate frames during sensor fusion. Students learn to integrate LiDAR, IMU, and camera data within the Gazebo simulator, creating a high-fidelity digital twin of their environment. This module culminates in the deployment of the TurtleBot3 navigation stack, where students implement SLAM for autonomous mapping and trajectory planning.

2.3. Advanced Topics

The final module, Advanced Robotics Applications, prepares students for specialized industry challenges by introducing cutting-edge technologies within the ROS2 ecosystem. Students explore multi-robot communication, leveraging native discovery protocols to coordinate swarm behaviors and agent-to-agent interaction. The curriculum also covers critical security features through SROS2, focusing on encryption and access control for robotic networks. To bridge the gap between high-level design and hardware execution, the module incorporates MATLAB and Simulink integration for control system modeling and MoveIt 2 for advanced robotic arm manipulation. By the end of this module, students are capable of configuring real-time control loops and managing sophisticated hardware interfaces, positioning them at the forefront of modern robotics engineering.

3. Student Activities

To ensure comprehensive mastery of the ROS2 ecosystem, the curriculum is built around three core pillars of student activity: technical homework assignments, academic reading, and a cumulative final project. These activities are designed to transition students from basic syntax to high-level system architecture.

3.1. Homework Assignments

Practical assignments serve as the primary mechanism for reinforcing lecture concepts, moving from foundational communication to complex spatial reasoning. These tasks include:

- *Node Development and Communication:* Students write custom ROS2 nodes in C++ or Python, implementing various Quality of Service (QoS) settings to observe their impact on data reliability and latency.
- *System Orchestration:* Writing complex launch files to manage multi-node lifecycles and parameter configurations across distributed systems.
- *Spatial Transformations:* Implementing TF2 transforms to handle coordinate frame relationships, a critical skill for sensor fusion and robot localization.
- *Sensor and Environment Integration:* Utilizing Gazebo to integrate simulated LiDAR, IMU, and camera data, allowing students to test perception algorithms in a risk-free digital twin environment.

- *Diagnostics and Debugging:* Utilizing ros2cli and rqt_graph to troubleshoot broken communication links and optimize node performance.

3.2. Research and Reading Assignments

To connect classroom learning with current advancements in the field, students are required to analyze a peer-reviewed research paper published within the last decade. This assignment focuses on robotics middleware, autonomy, or intelligent systems, culminating in a presentation that highlights:

- *Problem Identification:* Defining the specific gap in robotics research the authors addressed.
- *Methodological Rigor:* Evaluating the hardware and software stack used, with an emphasis on the role of ROS or similar middleware.
- *Empirical Results:* Summarizing the findings and their statistical or practical significance.
- *ROS2 Contextualization:* Discussing how the research could be implemented or improved using modern ROS2 features like DDS or SROS2.

3.3. Final Project: System Integration

The capstone of the course is a team-based final project where students design and deploy a complete ROS2-based robotic system. These projects mirror industry R&D cycles and often focus on:

- *Autonomous Navigation:* Deploying SLAM (Simultaneous Localization and Mapping) and the Nav2 stack on TurtleBot3 platforms.
- *Multi-Robot Coordination:* Leveraging the native DDS capabilities of ROS2 to enable swarm behaviors or collaborative task execution.
- *Manipulation and Control:* Utilizing MoveIt 2 for trajectory planning and inverse kinematics on robotic arm platforms.
- *Social Robotics:* Developing human-robot interaction (HRI) demos that incorporate computer vision and natural language processing.

The project lifecycle is strictly managed through formal milestones, including a technical proposal, a mid-semester progress report to identify bottlenecks, a live demonstration, and a comprehensive final report detailing the software architecture and performance metrics.

4. Implementation and Early Results

The inaugural implementation of the ROS2 curriculum yielded significant insights into student engagement and the practical application of middleware in an academic setting. By transitioning from a purely theoretical framework to an integrated, hands-on model, the course succeeded in fostering a high degree of technical proficiency. Observations from the first semester indicate that the students did not only master the syntax of ROS2 but also developed a sophisticated understanding of distributed system architecture, which has directly translated into improved performance in both academic and extracurricular engineering environments.

The first cohort comprised a total of 10 students (7 undergraduate seniors and 3 graduate students). In total, the course supported 7 domestic and 3 international students, with a gender distribution

of 9 males and 1 female. This demographic variety allowed for rich interdisciplinary collaboration, particularly during the final project phase where students were encouraged to form teams that balanced software expertise with mechanical design strengths.

4.1. Evaluation Metrics and Academic Performance

To quantify the impact of the new curriculum, the instructor conducted comprehensive pre- and post-course surveys alongside traditional grading metrics. The pre-course survey revealed a significant baseline gap: while 85% of students had heard of ROS, only 50% were aware of the critical ROS2 transition from ROS1. Furthermore, only 10% felt comfortable navigating a Linux terminal or writing a basic publisher node.

By the conclusion of the semester, the post-course survey indicated a dramatic shift, with over 90% of students reporting high confidence in debugging ROS2 systems and implementing complex communication patterns. Academic performance reflected this growth, with the class achieving a mean grade of 94.82%. A granular analysis of these grades showed that students demonstrated particular strength in simulation-based tasks, achieving high marks in environment mapping and path planning within Gazebo. Data indicates a 15% performance delta between simulation and physical deployment tasks. Students achieved near-perfect scores (96%) in Gazebo-based navigation, but encountered a "reality gap" when addressing challenges in latency management and sensor calibration [14].

Additional data suggests that the interdisciplinary team structure played a vital role; teams composed of both CS-heavy and ME-heavy backgrounds scored 8% higher on average in the final project than homogeneous teams. This suggests that the "Fundamentals to Advanced" modular approach effectively scaffolded the learning process, allowing students to leverage their respective strengths while overcoming initial programming hurdles [5]. Furthermore, the integration of Quality of Service (QoS) settings in the final modules was cited by 75% of students as the most challenging yet rewarding technical concept, essential for stabilizing communication in real-world, lossy wireless environments [15].

4.2. Impact on Student Competitions: Formula Student and BAJA

The practical utility of ROS2 was most visible in its immediate adoption by the Lawrence Technological University’s competitive engineering teams. Students involved in the Formula Student (Autonomous) competition leveraged the course's Nav2 and SLAM modules to overhaul their vehicle’s perception stack, replacing legacy ROS1 drivers with more efficient ROS2 nodes. This transition resulted in a measurable reduction in sensor-to-actuator latency, allowing the vehicle to navigate track boundaries with greater precision at higher speeds.

Similarly, the curriculum's impact extended to the Intelligent Ground Vehicle Competition (IGVC) and BAJA SAE teams [16,17]. IGVC participants utilized the multi-node scalability of ROS2 to integrate LiDAR and stereo vision for complex obstacle avoidance in unstructured environments. Meanwhile, the BAJA SAE team utilized ROS2 to develop a ruggedized telemetry system, by deploying custom nodes on an onboard microcontroller (micro-ROS), demonstrating the framework's versatility in harsh, off-road conditions [18]. These competition-based applications proved that ROS2 training provided students with a distinct competitive advantage, enabling them to implement industry-standard tools in high-pressure, real-world scenarios.

4.3. Final Project Case Studies and Integration

The capstone of the course required student teams to design and implement a fully autonomous ROS2-based system, resulting in several impressive demonstrations of technical mastery. One notable project involved an autonomous warehouse swarm where multiple TurtleBot3 units were programmed to perform coordinated patrolling and obstacle avoidance without the need for a centralized controller, showcasing the power of the ROS2 discovery protocol. Another high-performing team developed a robotic arm manipulation system using MoveIt 2, which integrated an RGB-D camera to perform real-time object detection and sorting based on color and geometry. Each project was required to pass through a formal lifecycle, starting with a technical proposal and a mid-semester progress report, culminating in a live demonstration and a final written report. These projects demonstrated that students could successfully navigate the complexities of hardware-in-the-loop development while maintaining clean, modular codebases.

4.4. Challenges and Lessons Learned

Despite the overall success of the course, several challenges emerged that will inform future iterations of the curriculum. The most significant hurdle was the steep learning curve associated with the Linux environment and C++ build dependencies, which often consumed valuable lab time that could have been spent on robotics logic. Hardware availability also proved to be a constraint, as the high enrollment numbers occasionally led to bottlenecks during the testing phase of the final projects. Furthermore, the time required to troubleshoot initial ROS2 installations on varied personal laptops suggested a need for more standardized development environments. To address these issues, future offerings will include a pre-semester programming bootcamp to normalize the student body's software skills and an expanded library of simulation-based labs to ensure that students can progress through the curriculum even when physical hardware is in high demand.

5. Analysis of Student Final Projects

The culmination of the ROS2 curriculum was a high-stakes final project where student teams applied their cumulative knowledge to solve practical robotics problems. These projects demonstrated the students' ability to navigate the complexities of system architecture, sensor integration, and real-time control [19].

5.1. Autonomous Maze-Solver with TurtleBot3

The first student project involved the development of an autonomous navigation system designed to solve complex maze environments using the TurtleBot3 "Waffle" platform. To establish a rigorous testing ground, the student utilized Gazebo’s world generation tools to convert a standard png maze image into a high-fidelity 3D simulation environment. The software architecture relied on the integration of the SLAM (Simultaneous Localization and Mapping) toolbox and Rviz for real-time visualization, allowing the robot to build a persistent occupancy grid as it explored unknown territory as shown in Figure 1. This foundational setup ensured that the robot could maintain an accurate estimate of its pose while identifying the structural boundaries of the maze.

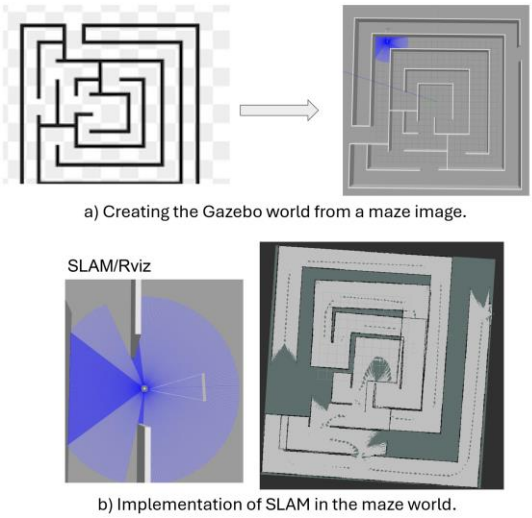


Figure 1: Autonomous Maze-Solver with TurtleBot3

For the motion control logic, the student implemented a Python-based reactive controller that processed LiDAR sensor data to perform deterministic obstacle avoidance. By segmenting the laser scan ranges into specific angular sectors— left (60°-120°), right (240°-300°), and forward (350°-10°)—the script dynamically adjusted the robot's linear and angular velocities. When the forward distance dropped below a 0.5-meter threshold, the robot was programmed to halt its forward progress and execute a pivot rotation toward the direction with the greatest clearance. This project successfully demonstrated the transition from raw sensor perception to closed-loop actuation, resulting in a system capable of traversing non-linear environments with minimal localization drift.

5.2. IGVC-Inspired Autonomous Obstacle Avoidance

The second student project focused on developing a ROS2-based autonomous navigation stack specifically designed for integration into the Lawrence Technological University's Intelligent Ground Vehicle Competition (IGVC) platform [16]. The primary objective was to engineer a system capable of navigating a simulated outdoor environment while successfully identifying and bypassing both static and dynamic obstacles as shown in Figure 2. To achieve this, the team implemented a dual-layered navigation architecture consisting of a global path planner and a local reactive controller. The global planner utilized a static occupancy grid (/map) to generate optimized waypoints, while the local controller continuously processed high-frequency LiDAR (/scan) data to adjust the robot's velocity commands (/cmd_vel) in real-time, ensuring safe passage through narrow corridors and around moving hazards. The Rviz dashboard displays the robot's global path planning results and real-time obstacle detection signatures during a fully autonomous mission.

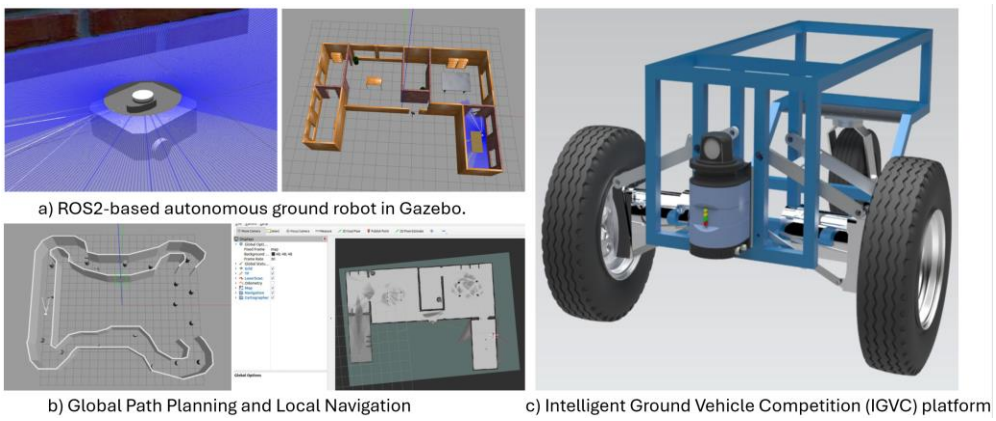


Figure 2: IGVC-Inspired Autonomous Obstacle Avoidance System.

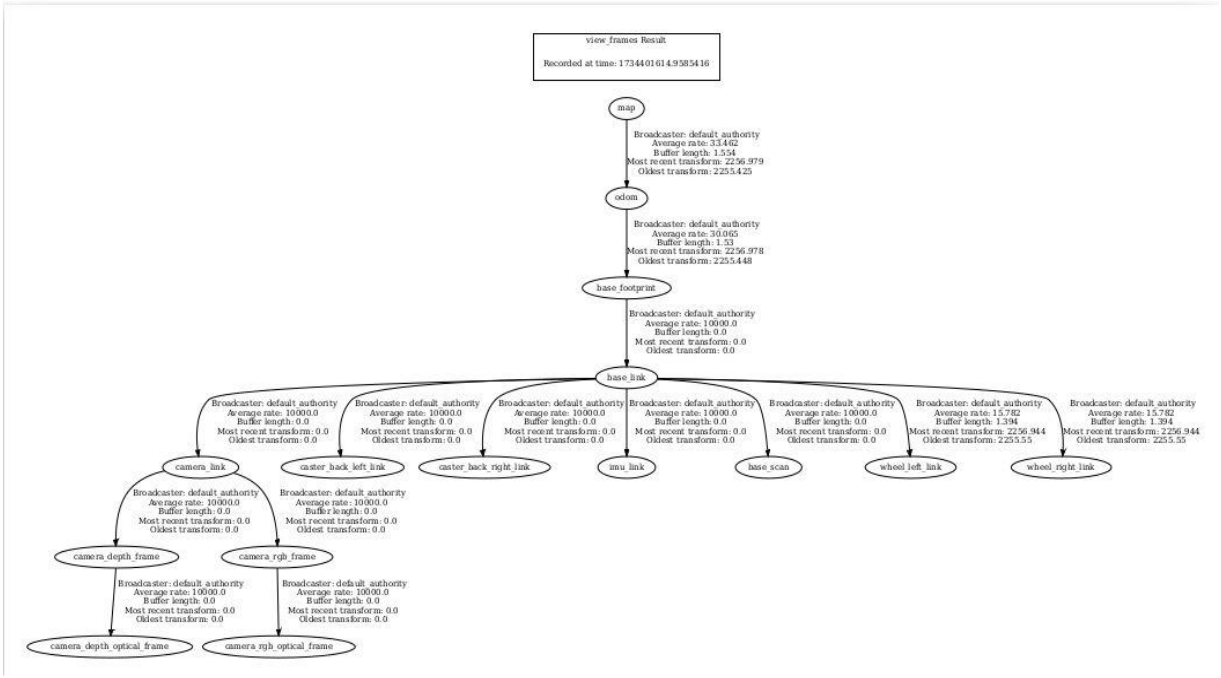


Figure 3: The ROS2 Transform (TF) tree showing the hierarchical relationship between the map, odometry, and robot base frames required for accurate localization.

A critical component of this project was the management of complex spatial relationships through the ROS2 Transform (TF) tree. By maintaining accurate coordinate frames between the "map," "odom," and "base_link," the students were able to achieve robust localization and deterministic state-switching. In the final demonstration, the system effectively prioritized safety-critical data; if an obstacle was detected within a predefined 0.5-meter buffer, the local controller would override the global trajectory to execute an immediate avoidance maneuver. Despite challenges related to virtual machine performance and map resolution tuning, the project successfully validated a scalable navigation framework that the team intends to transition to physical GPS-based outdoor hardware in future iterations. Figure 3 showing the TF transform tree and the Rviz global plan overlayed with local LiDAR point clouds.

5.3. AMR Warehouse Logistics Simulation

Focusing on industrial applications, this project developed an Autonomous Mobile Robot (AMR) tailored for warehouse environments as shown in Figure 4. The student created a comprehensive URDF model from scratch, defining visual, collision, and inertial properties for a differential drive chassis equipped with LiDAR and camera sensors. The software stack leveraged the slam_toolbox for asynchronous mapping and the Nav2 stack for autonomous goal reaching. The project demonstrated a professional-grade simulation pipeline, including joystick integration for manual teleoperation and successful autonomous navigation through narrow warehouse aisles.

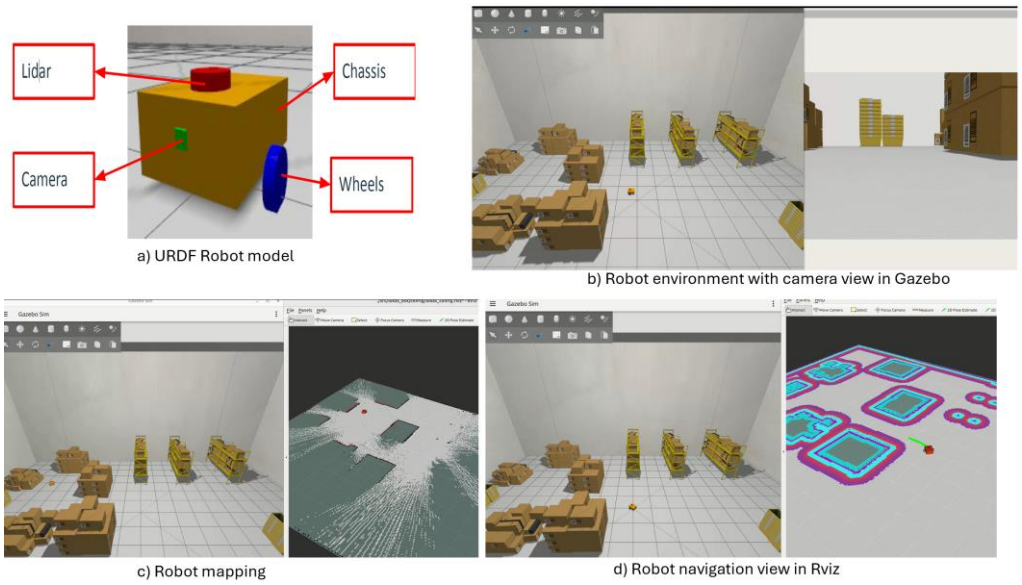


Figure 4: Autonomous Mobile Robot (AMR) based warehouse logistics simulation

5.4. Turtle Sim Proportional Chasing Logic

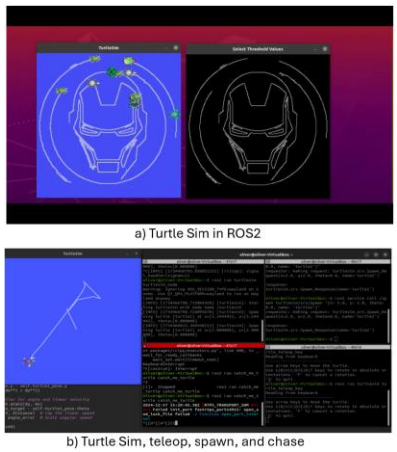


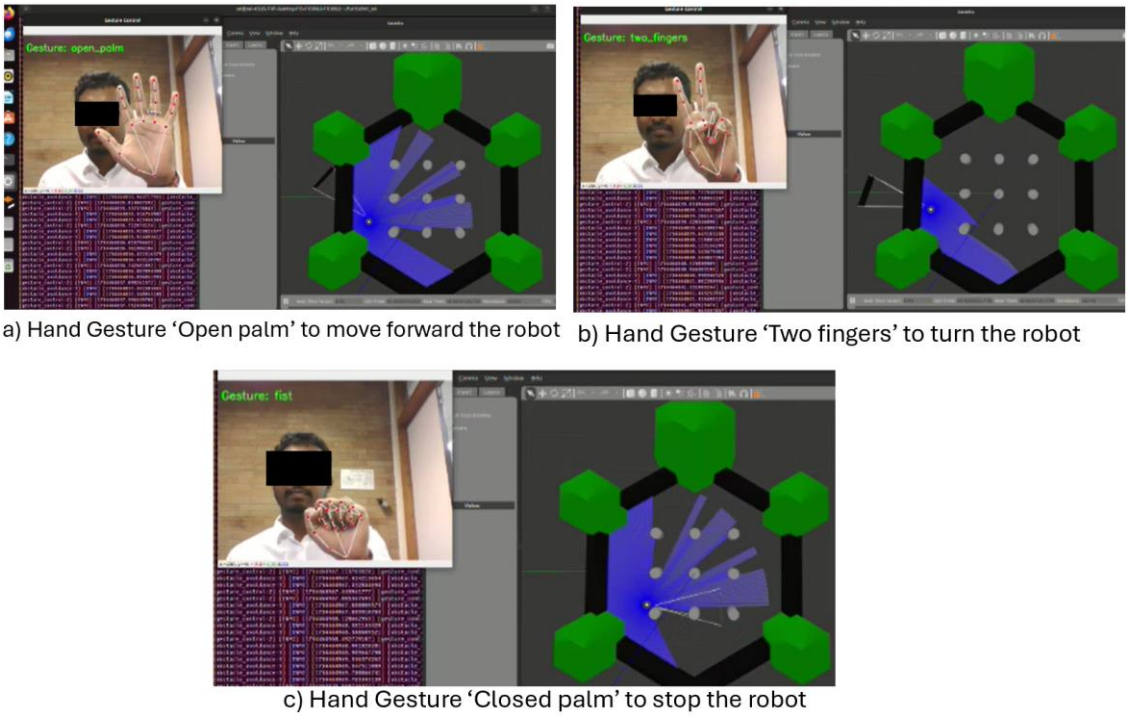
Figure 4: Turtle Sim Chase and draw simulation

In this multi-agent simulation project, students developed a "chaser" turtle that autonomously pursued a "target" turtle controlled via teleoperation. The logic relied on subscribing to both turtles'

pose messages and calculating the Euclidean distance and heading error. A proportional (P) controller was then implemented to adjust the chaser's linear and angular speeds dynamically. This project highlighted the importance of message synchronization and the practical application of control theory within the ROS2 framework.

5.5. Hand-Guided Autonomy: Gesture-Controlled TurtleBot Navigation

This highly innovative project integrated human-computer interaction (HCI) with traditional robotics by using a camera-based gesture recognition system to drive a TurtleBot. Utilizing OpenCV and Mediapipe, the students mapped hand landmarks to specific ROS2 commands: an 'Open palm' for forward motion, a 'Closed palm' to stop, and Two fingers' to turn the robot as shown in Figure 5. Crucially, they implemented an "Autonomy Override" node that used LiDAR data to ignore user commands if a collision was imminent. Despite initial challenges with latency and gesture misrecognition, the team optimized the system to three distinct gestures, achieving a responsive and safe hand-guided navigation system.



a) Hand Gesture 'Open palm' to move forward the robot

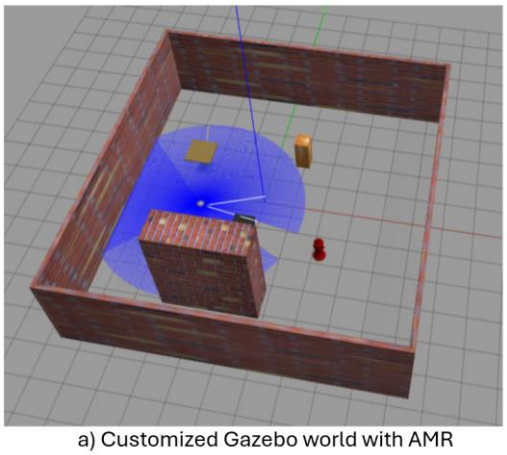
b) Hand Gesture 'Two fingers' to turn the robot

c) Hand Gesture 'Closed palm' to stop the robot

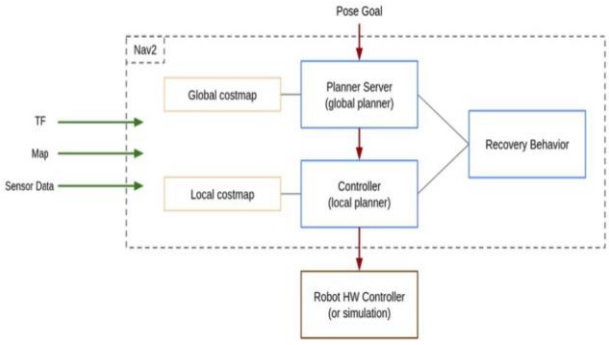
Figure 5: Hand-Guided Autonomy: Gesture-Controlled TurtleBot Navigation in Gazebo using ROS2 and OpenCV

5.6. Comprehensive AMR Mapping and Localization

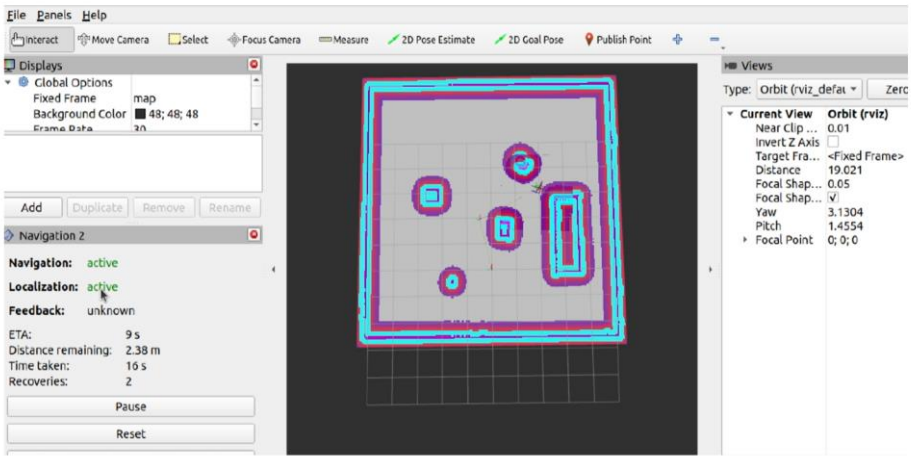
This student's project focused on the complete "Map-to-Goal" pipeline for an autonomous mobile robot as shown in Figure 6. The student built a custom navigation package that included configured YAML files for the Nav2 controller and planner servers. By utilizing the *libgazebo_ros_diff_drive* plugin, the student ensured that the simulated odometry accurately reflected the robot's physical movement, which is critical for minimizing localization drift during SLAM. The student successfully mapped a complex environment using the Cartographer node and demonstrated "Waypoint Navigation," where the robot could be assigned multiple sequential goals through the Rviz interface.



a) Customized Gazebo world with AMR



b) AMR navigation stack block diagram



c) Navigation and simulation using Rviz

Figure 6: Comprehensive AMR Mapping and Localization

6. Conclusion and Future Work

The successful transition from a legacy ROS1 curriculum to a modern ROS2 framework at the Dept. of Mechanical, Robotics & Industrial Engineering in the Lawrence Technological University represents a vital step in aligning robotics education with the current demands of industry and high-level research. By moving away from an aging middleware and adopting a framework built for real-time performance and multi-agent coordination, the department has provided students with a toolkit that is directly applicable to cutting-edge sectors such as autonomous driving, smart manufacturing, and service robotics. The implementation of this course confirmed that while the technical barriers to entry are higher in ROS2, the resulting proficiency allows students to build more robust, scalable, and secure robotic systems than was possible under previous pedagogical models.

The first offering of the ROS2 course demonstrated a profound impact on student engagement and professional development. Students reported not only an increased confidence in robotics programming but also a marked improvement in their ability to function within interdisciplinary teams, a skill that mirrors the collaborative nature of the modern engineering workforce.

Furthermore, the tangible success seen in student competitions, such as Formula Student and BAJA SAE, highlights the immediate value of this curriculum in enhancing the Lawrence Technological University's competitive standing. Industry partners have already noted that graduates with documented ROS2 experience possess a significant advantage in the job market, as they can contribute to sophisticated R&D projects with minimal onboarding.

Looking ahead, several improvements are planned to address the challenges identified during this initial rollout. To mitigate the steep learning curve for students with limited software backgrounds, the department will introduce a pre-semester programming bootcamp focused on Linux systems and C++ fundamentals. Additionally, the curriculum will be bolstered by an expanded suite of simulation-based labs and increased hardware resources to ensure all students have ample opportunity for physical system testing. By continuously refining these teaching methods and expanding the integration of advanced topics like SROS2 and MoveIt 2, the program aims to remain at the forefront of robotics education, preparing the next generation of engineers to lead advancements in this rapidly evolving field.

7. References

[1] Al-Batati, A. S., Koubaa, A., & Abdelkader, M. (2024). ROS 2 in a nutshell: A survey. MDPI AG. https://doi.org/10.20944/preprints202410.1204.v2

[2] A. Bonci, F. Gaudeni, Maria Cristina Giannini, and S. Longhi, "Robot Operating System 2 (ROS2)-Based Frameworks for Increasing Robot Autonomy: A Survey," Applied sciences, vol. 13, no. 23, pp. 12796–12796, Nov. 2023, doi: https://doi.org/10.3390/app132312796.

[3] Dey, E., Walczak, M., Anwar, M. S., Roy, N., Freeman, J., Gregory, T., Suri, N., & Busart, C. (2023). A novel ROS2 QoS policy-enabled synchronizing middleware for co-simulation of heterogeneous multi-robot systems. 2023 32nd International Conference on Computer Communications and Networks (ICCCN), 1–10.https://doi.org/10.1109/icccn58024.2023.10230109

[4] Jo, Y. H., Cho, S. Y., & Choi, B. W. (2023). Towards a ROS2-based software architecture for service robots. Bulletin of Electrical Engineering and Informatics, 12(5), 3027–3038.https://doi.org/10.11591/eei.v12i5.5590

[5] Luo, J., Jifri, S. M., Al-Saadi, A. S., & Al-Kindi, M. S. (2023). Integrated Teaching of ROS 2 and MATLAB/Simulink for Robotics Education: A Practical Approach. *International Journal of Mechanical Engineering Education*, *51*(4), 312–328.https://doi.org/10.1177/03064190231189422

[6] Macenski, S., Foote, T., Gerkey, B., Lalancette, L., and Woodall, W. "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, Vol. 7, No. 66, 2022.

[7] Cañas, J. M., Perdices, E., García-Pérez, L., & Fernández-Conde, J. (2020). A ROS-based open tool for intelligent robotics education. Applied Sciences, 10(21), 7419.

[8] Monteiro, A. F., Miranda-Pinto, M., Osório, A. J., & Araújo, C. (2019). Curricular integration of computational thinking, programming and robotics in basic education: A proposal for teacher training.

[9] Cabrera-Ponce, M. J., Morales, A., & Rodriguez-Gomez, G. (2022). Teaching robotics and ROS2 to students with diverse backgrounds: A project-based approach. *Frontiers in Robotics and AI*, *9*, 895471. https://doi.org/10.3389/frobt.2022.895471

[10] Koubaa, A. (2023). Robot Operating System (ROS): The Complete Reference (Volume 7). Springer Nature. https://doi.org/10.1007/978-3-031-20125-7

[11] Tosello, E., Castaman, N., & Menegatti, E. (2019). Using robotics to train students for Industry 4.0. IFAC-PapersOnLine, 52(9), 153-158.

[12] Vicente, F. R., Zapatera Llinares, A., & Montes Sanchez, N. (2021). Curriculum analysis and design, implementation, and validation of a STEAM project through educational robotics in primary education. Computer Applications in Engineering Education, 29(1), 160-174.

[13] Caiazza, G., Esposito, A., & Petrillo, A. (2023). A Hands-On Approach for Teaching ROS 2 in Robotics Engineering: From Theory to Practice. *Sensors*, *23*(15), 6821. https://doi.org/10.3390/s23156821

[14] García-Vicent, V., Villalonga, A., & Rosillo, N. (2023). Bridging the Gap in Robotics Education: A Multi-Disciplinary ROS 2 Framework for Graduate and Undergraduate Students. *Journal of Intelligent & Robotic Systems*, *108*(3), 45. https://doi.org/10.1007/s10846-023-01892-z

[15] Maruyama, Y., Kato, S., & Azumi, T. (2022). Exploring the performance of ROS 2 as a real-time distributed system. *Proceedings of the 2022 IEEE 28th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 154–167. https://doi.org/10.1109/RTAS54340.2022.00021

[16] The annual Intelligent Ground vehicle Competition (IGVC) sponsored by Michigan Economic Development Corp(MEDC). http://www.igvc.org/

[17] SAE Baja intercollegiate engineering challenge. https://www.bajasae.net/

[18] Abele, A., Almansa, J. P., & Koubaa, A. (2022). Micro-ROS: Bringing ROS 2 to Microcontrollers for Industrial and Competition Robotics. *Journal of Software Engineering for Robotics*, *13*(1), 1–15. https://doi.org/10.6092/JOSER_2022_V13_I1_P1

[19] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. "ROS: an open-source Robot Operating System," *ICRA workshop on open source software*, Vol. 3, No. 3.2, 2009.