1/10 scale Autonomous Vehicle using ROS

1st Nabil Hassan
Mechatronics Department
German University in Cairo
Cairo, Egypt
nabil.hassan@student.guc.edu.eg

2nd Yassin Eissa
Mechatronics Department
German University in Cairo
Cairo, Egypt
yassin.eissa@student.guc.edu.eg

3rd Mahmoud Assem
Mechatronics Department
German University in Cairo
Cairo, Egypt
mahmoud.elkattan@student.guc.edu.eg

4th Mohamed Mohy
Mechatronics Department
German University in Cairo
Cairo, Egypt
mohamed.mohyeldin@student.guc.edu.eg

5th Ahmed Akkad
Mechatronics Department
German University in Cairo
Cairo, Egypt
ahmed.alakkad@student.guc.edu.eg

***Abstract*—This paper presents the development and implementation of a low-cost, small-scale autonomous ground vehicle designed for structured environments. The system adopts a modular architecture composed of a Raspberry Pi as the onboard computer for high-level processing and an Arduino Nano as the low-level controller for motor actuation and sensor interfacing. The vehicle integrates an Inertial Measurement Unit (IMU) for heading correction and wheel encoders for real-time speed estimation. A Stanley controller is utilized for lateral path tracking, while a PID controller governs longitudinal velocity. The software stack is built using the Robot Operating System (ROS), enabling efficient communication between system components and real-time decision-making. The proposed platform demonstrates reliable autonomous navigation capabilities and serves as an accessible and scalable solution for research and educational applications in the field of autonomous systems.**

***Index Terms*—Autonomous ground vehicle, ROS, Raspberry Pi, Arduino Nano, Stanley controller, PID control, IMU, wheel encoders, embedded systems, mobile robotics, path tracking.**

I. Introduction

Autonomous vehicles are a rapidly evolving field at the intersection of robotics, artificial intelligence, and control systems. Their development promises to transform the way humans transport goods and themselves, offering significant advantages in safety, efficiency, scalability, and sustainability. While full-scale autonomous vehicles are the focus of major industrial efforts and commercial deployments, scaled-down autonomous platforms have emerged as powerful tools for research and education. Among these, 1/10 scale autonomous cars have gained considerable traction due to their affordability, safety, ease of experimentation, and strong community support.

A particularly prominent framework in this domain is the F1Tenth Autonomous Racing Platform, which provides a standardized, reproducible, and open-source platform for developing and benchmarking autonomous driving algorithms at 1/10 scale. Built on top of the Robot Operating System (ROS), the F1Tenth platform offers modular tools and software packages for perception, planning, and control, enabling researchers and students to implement real-world autonomous systems in a simulated or physical environment. The platform is supported by an international community and regularly features in autonomous racing competitions, fostering innovation and collaboration.

The development of autonomous vehicles, even at reduced scale, involves solving several complex tasks: localization, mapping, obstacle detection, trajectory planning, and control. These tasks must operate concurrently under real-time constraints and interact through a tightly integrated software architecture. ROS facilitates this integration by providing communication tools, simulation environments like Gazebo, and driver libraries for interfacing with sensors and actuators. As a middleware, ROS enables distributed development and modular testing, which are essential for the layered design of autonomous systems.

In this work, we present the design and implementation of a 1/10 scale autonomous vehicle built upon the principles of the F1Tenth platform and ROS architecture. The system was developed to perform autonomous lane-following and obstacle avoidance using real-time sensor feedback and closed-loop control. A Raspberry Pi 4 was used as the onboard high-level computer, running ROS nodes responsible for perception, planning, and control. Meanwhile, an Arduino Nano served as the low-level controller to directly manage motor speed and steering commands.

The localization system employed encoder data for speed estimation and an Inertial Measurement Unit (IMU) for heading correction, fused to provide accurate pose estimation in the absence of GPS. For lateral control, we implemented the Stanley Controller, a well-known algorithm used in path tracking for autonomous vehicles due to its simplicity and robustness to lateral errors. For longitudinal control, a classic PID controller was used to maintain desired velocity profiles. Together, these controllers enabled smooth, stable, and responsive vehicle behavior during navigation.

The platform was tested in both simulated environments and real-world indoor tracks, demonstrating its ability to track waypoints, navigate turns, and maintain lane stability in a reproducible manner. By adopting a system-level engineering approach, our implementation provides a practical foundation

for further experimentation in areas such as real-time SLAM, deep learning-based perception, or multi-agent vehicle coordination.

This project contributes to the growing body of work that leverages 1/10 scale vehicles not only as educational tools but also as scalable prototypes for real-world autonomous driving systems. By aligning with the F1Tenth initiative, our system encourages reproducibility, benchmarking, and collaborative development, making it an ideal platform for future research and innovation in the field of autonomous systems.

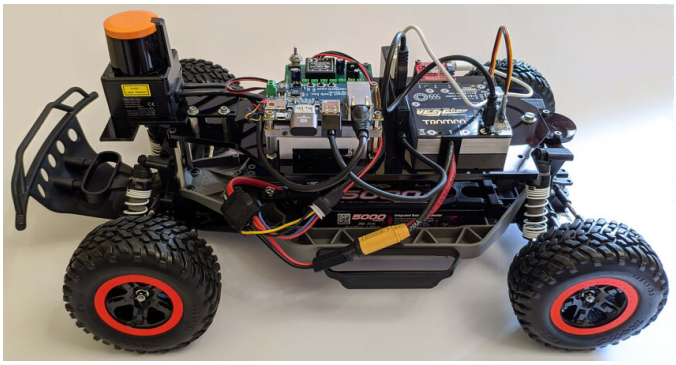

Fig. 1. F1Tenth Autonomous Car

II. Literature Review

Autonomous vehicles (AVs) rely on a combination of perception, localization, planning, and control modules to operate safely and efficiently. Recent research has explored innovations in each of these areas, with a particular emphasis on integrated system design, sensor fusion, and real-time decision-making. This section reviews recent contributions in control, localization, navigation, and integrated system architectures relevant to autonomous driving, including insights from the F1TENTH autonomous racing platform.

A. Control Modules in Autonomous Vehicles

The control system is responsible for executing planned trajectories while maintaining stability and safety. Modern approaches increasingly integrate perception and control to optimize performance in dynamic environments.

- **DeepIPC: Deeply Integrated Perception and Control for an Autonomous Vehicle in Real Environments** [1] proposes an end-to-end model that tightly couples perception and control. The system utilizes RGB-D images for semantic segmentation and generates a bird's-eye view (BEV) representation. Fused with GNSS and angular velocity data, the system predicts navigational waypoints, demonstrating high accuracy and robustness in complex, real-world scenarios.
- **One Stack to Rule Them All: To Drive Automated Vehicles, and Reach for the 4th Level** [2] presents a modular automated driving stack with components for localization, perception, planning, control, and safety. The framework emphasizes scalability and rapid prototyping, and it has been successfully deployed in urban passenger transport scenarios.
- **Real-Time Control Strategies for High-Speed Autonomous Racing** [3] demonstrates the F1TENTH platform's capabilities in developing and testing control algorithms under extreme conditions. The study compares model predictive control (MPC) and pure pursuit controllers for aggressive cornering at speeds exceeding 8 m/s with 1:10 scale vehicles.

B. Localization Modules in Autonomous Vehicles

Precise localization is essential for reliable autonomous navigation, especially in GNSS-denied or cluttered environments.

- **CV2X-LOCA: Roadside Unit-Enabled Cooperative Localization Framework for Autonomous Vehicles** [4] introduces a cooperative localization method using cellular vehicle-to-everything (C-V2X) communication. The system achieves lane-level accuracy in GNSS-challenged environments by exploiting channel state information from nearby roadside units.
- **A Survey on Localization for Autonomous Vehicles** [5] provides a comprehensive overview of state-of-the-art localization techniques, analyzing sensor-based, map-based, and learning-based methods. It highlights the trade-offs in terms of accuracy, robustness, and computational requirements, and underscores the growing importance of sensor fusion.

C. Navigation Modules in Autonomous Vehicles

Navigation modules generate feasible and safe trajectories while adapting to dynamic surroundings. Recent efforts focus on combining classical algorithms with AI-based learning methods.

- **Navigation of Autonomous Light Vehicles Using an Optimal Trajectory Planning Algorithm** [6] presents a trajectory planning framework that accounts for energy efficiency, localization, mapping, and obstacle avoidance. The algorithm improves path optimality for lightweight AV platforms.
- **Hierarchical End-to-End Autonomous Navigation Through Few-Shot Learning** [7] explores a hybrid framework that integrates few-shot learning with classical motion planning. The hierarchical structure enables AVs to generalize to new environments with minimal retraining, offering a promising direction for scalable navigation systems.
- **Minimum-time Trajectory Planning for Autonomous Racing** [8] implements a time-optimal racing line generation method on the F1TENTH platform. The approach combines Bézier curve optimization with vehicle dynamics models, achieving lap times within 97% of human expert drivers in scaled environments.

D. Integrated Architectures for Autonomous Vehicles

Seamless integration of localization, control, and navigation modules is critical for real-time autonomy. Modular and

scalable system designs enable deployment across diverse platforms.

- **System, Design, and Experimental Validation of Autonomous Vehicle Architectures** [**?**] details a distributed architecture with modular components for perception, planning, and control. The system emphasizes sensor fusion for robust localization and has been validated in multiple operational scenarios.
- **One Stack to Rule Them All** [2], revisited here for its integrated design, further demonstrates how modular architectures can support real-world autonomy, including safety-critical applications and rapid experimental iterations.
- **The F1TENTH Autonomous Racing Ecosystem** [9] presents an open-source hardware/software platform for developing autonomous racing algorithms. The system integrates ROS 2 with custom vehicle dynamics models, enabling real-time performance at 60 Hz update rates while maintaining sub-centimeter positioning accuracy through LiDAR-based localization.

II. Methodology

The development of our 1/10 scale autonomous vehicle was approached through several key phases, beginning with hardware fabrication and basic system setup, progressing through simulation and control architecture design, and culminating in real-world testing and system integration using the Robot Operating System (ROS) framework.

A. Hardware Fabrication and Initial Setup

The project commenced with the physical fabrication of a 1/10 scale vehicle chassis, which was designed and 3D-printed in-house. The drivetrain consisted of a high-torque 1100KV Brushless DC (BLDC) motor controlled via an Electronic Speed Controller (ESC), and a servo motor for steering actuation. Both actuators were interfaced with an Arduino Nano microcontroller for low-level Pulse Width Modulation (PWM) control. A Raspberry Pi 4B running Debian Buster was installed as the high-level computing unit to interface with ROS and run control and localization nodes.

Concurrently, the software environment was prepared. The F1Tenth-inspired vehicle model was initially tested in the *Gazebo* simulator using an *Audi R8* model. A "Hello World" ROS node was executed on both the Raspberry Pi and the host Ubuntu machine to verify the correct setup and inter-device communication.

B. Basic Control Nodes and Motor Testing

Following successful system initialization, two ROS nodes were developed and tested on the Ubuntu machine, which hosted the Gazebo environment:

- **Open Loop Response Node (OLR)**: This node commanded the vehicle to move in a straight line at constant speed without feedback.
- **Teleoperation Node**: This node enabled keyboard-based manual control of the vehicle's speed and steering.

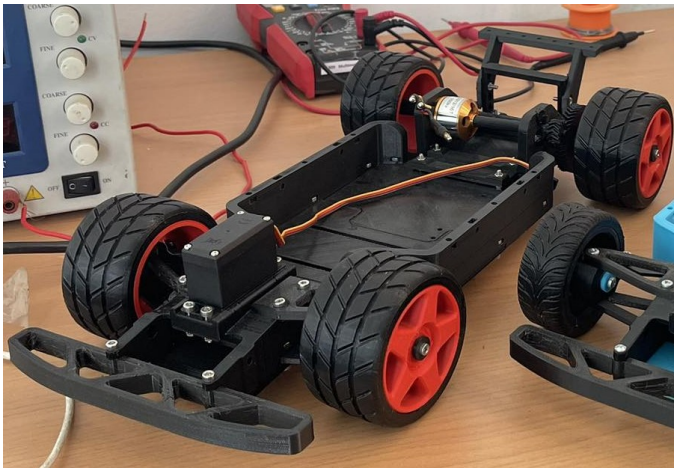

Fig. 2. The 1/10 3D printed Model

These nodes were integrated into a single `roslaunch` file that initialized the Audi R8 Gazebo model, OLR, and teleoperation simultaneously, enabling coordinated testing of the simulated environment. In parallel, motor testing was conducted on the physical hardware. The Arduino Nano was programmed to send PWM signals to the ESC and servo motor, allowing manual control and validation of actuator response.

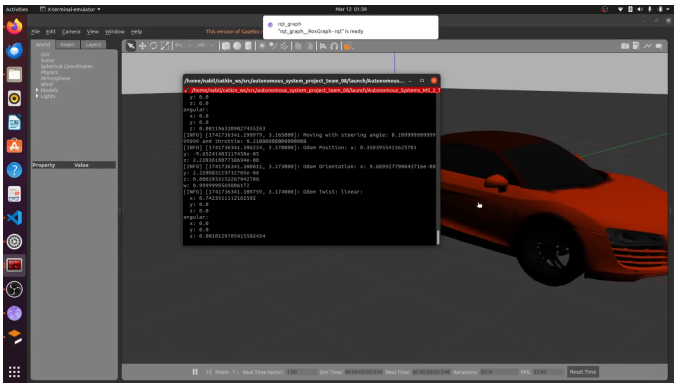

Fig. 3. Audi R8 in Gazebo

C. Closed-Loop Control Development

Subsequent work focused on designing and implementing closed-loop control systems. Two control ROS nodes were developed:

- **Speed Controller Node**: A feedback controller that regulated the vehicle's linear velocity using encoder readings.
- **Lateral Controller Node**: A PID-based controller that maintained the vehicle's lane position using a simple error-based heading correction scheme.

These control nodes were integrated into a new `roslaunch` file alongside the Gazebo vehicle model. Vehicle start and target positions were set using `rosparam`. The control algorithms were validated in simulation across

various target positions within an empty Gazebo world to cover the entire vehicle workspace.

An updated teleoperation node from Milestone 2 (MS_2) was also re-integrated, now capable of sending desired speed and steering commands to the Arduino via USB serial communication. On the Arduino side, a proportional controller was implemented to regulate PWM signals sent to the actuators based on the target commands. To achieve accurate feedback control, encoders were added to both the BLDC motor and the steering mechanism for real-time velocity and angle measurement.

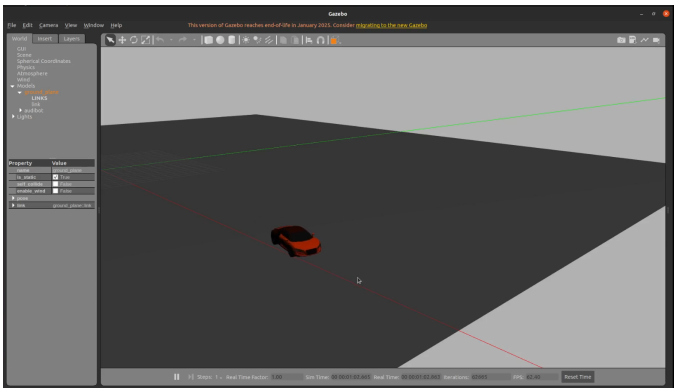

Fig. 4. Lateral & Longitudinal Control in Gazebo

D. Planning and Map Integration

A scaled track was designed with two lanes and known obstacle positions. The map was recreated in the Gazebo simulator, with all obstacles placed accurately. Due to dimensional mismatch (track designed for 1:14–1:18 scale), appropriate scaling was applied to ensure compatibility with the Audi bot Audi R8 in gazebo

A dedicated **Planning ROS Node** was developed to:

- Analyze the environment and obstacle positions.
- Generate a lane-following strategy with dynamic lane changes to avoid collisions.
- Plan a velocity profile appropriate for the path and send target commands to the Control ROS node.

A new launch file was created to initialize the Gazebo environment, vehicle model, controller, and planner. Initial vehicle pose, velocity, and planning parameters were set using `rosparam`.

Hardware-wise, essential sensors were purchased and installed to allow full measurement of system inputs (velocity and steering angle) and states (heading and position). Arduino firmware was updated to acquire and transmit sensor readings to the Raspberry Pi via serial communication for further processing.

E. Localization and System Integration

To enable real-time feedback and noise-resilient estimation, a **Localization ROS Node** was implemented. This node:

- Subscribed to raw sensory data from the Arduino.

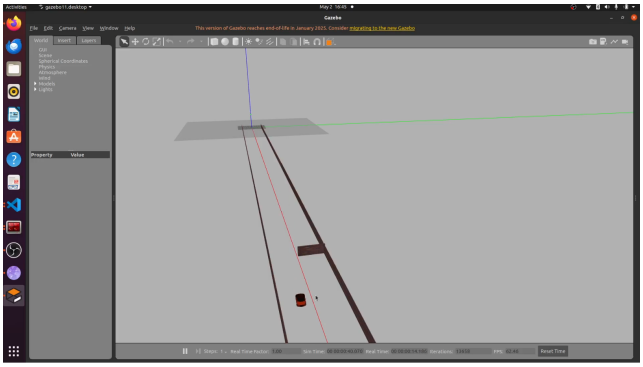

Fig. 5. Obstacle Avoidance Path Planning

- Injected synthetic Gaussian white noise based on the real sensors' specifications.
- Filtered the noisy signals using appropriate filters.
- Published the filtered state estimations to the Control node.

This Localization node was integrated into a consolidated `roslaunch` file along with the Gazebo model, planner, and controller. Gaussian noise parameters and initial conditions were specified using `rosparam`.

A custom **PCB** was designed and fabricated to consolidate all electrical wiring beneath the vehicle chassis, ensuring structural integrity and clean layout for testing.

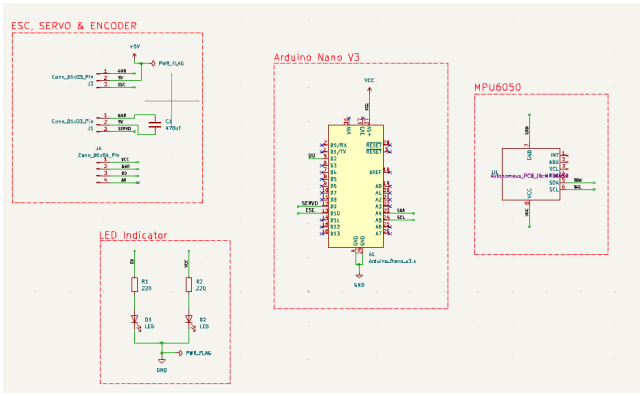

Fig. 6. The Schematic

In this phase, a Localization ROS node was designed and implemented to estimate the vehicle's state in a manner that closely simulates real-world behavior. The node subscribes to essential dynamic parameters of the vehicle — such as velocity, steering angle, and heading — which are typically obtained from onboard sensors in practical scenarios.

To emulate realistic sensor performance within the simulation environment, white Gaussian noise was intentionally added to the measurement signals. The standard deviation of this noise was chosen based on typical values found in commercial sensor specifications. This step allowed us to evaluate the control algorithms under more realistic and

challenging conditions, accounting for the presence of measurement uncertainty.

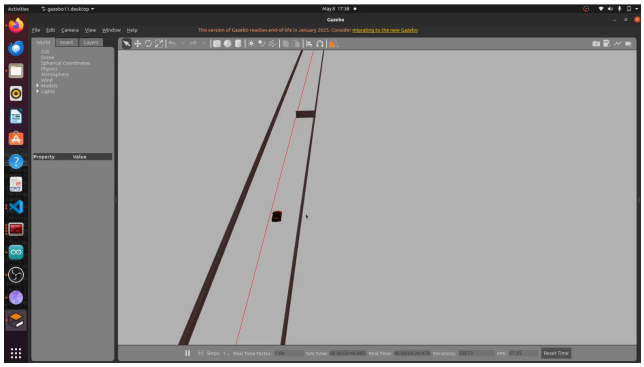

Fig. 7. The filtered response using kalman filter

To recover accurate state information from the noisy inputs, a Kalman Filter was implemented within the Localization node. This filter efficiently fuses the noisy sensor data with a motion prediction model of the vehicle, providing more reliable state estimates. These filtered estimates were then published to the Control ROS node, which depended on clean and stable data for effective execution of control commands.

The addition of the Kalman Filter affected the dynamic behavior of the vehicle due to the inherent smoothing and slight delay in the filtered signals. Consequently, the Stanley controller used for lateral path tracking had to be re-tuned. Specifically, its gain parameters were adjusted to compensate for the changes in signal dynamics and ensure that the vehicle could still follow the desired path accurately. This calibration was essential for maintaining precise lane-keeping and obstacle avoidance behavior within the simulated environment.

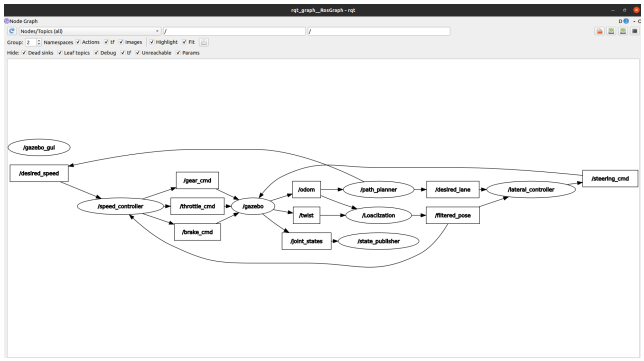

Fig. 8. The final rqt-graph

III. Results

A. Simulation Results

1) Gazebo: The simulation environment was set up in Gazebo using a model of an Audi R8 model, within a custom-built world that included both straight paths and obstacle-populated lanes. This allowed for a comprehensive evaluation of the performance of the vehicle's perception, control, and planning nodes under different driving conditions.

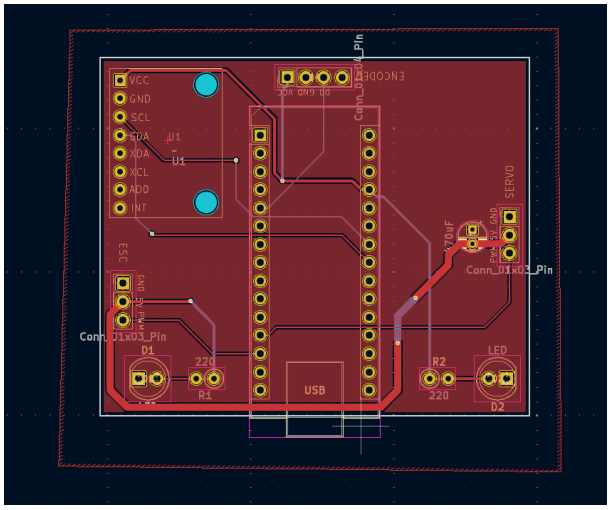

Fig. 9. The designed PCB

In the straight-line driving scenario, the vehicle exhibited smooth and steady motion with minimal deviation from the desired path, maintaining orientation within ±3 degrees of the reference heading. The open-loop response validated the correct actuation and model configuration within the Gazebo environment, while the closed-loop control (velocity PID and Stanley lateral controller) provided precise trajectory tracking.

The Kalman Filter used in the localization node effectively suppressed simulated sensor noise (white Gaussian), providing clean and accurate estimates of the vehicle's position and heading. This ensured that the control system received reliable feedback, contributing to stable and responsive vehicle motion throughout the simulation runs.

In the obstacle avoidance scenario, the planning node successfully generated a velocity profile and lane trajectory that enabled the car to detect, plan around, and avoid multiple static obstacles placed along the track. The coordination between the planning and control nodes was seamless, resulting in smooth lane changes and adaptive speed control without any collision or instability.

These simulation results confirmed the correctness, robustness, and consistency of the ROS-based architecture, validating the integration of control, planning, and localization subsystems in a virtual environment before hardware deployment.

2) Hardware Results: Following extensive validation in simulation, the developed autonomous vehicle system was deployed and tested on the physical 1/10 scale car platform. The objective was to evaluate the real-time performance of the control and planning algorithms under real-world conditions using actual sensors and actuators.

In the straight-line test, the car demonstrated highly stable motion with minimal steering drift. The measured heading deviation remained within ±3 degrees, consistent with the simulation results. This confirmed the effectiveness of both the speed controller and the lateral Stanley controller in maintaining the desired trajectory on the physical platform.

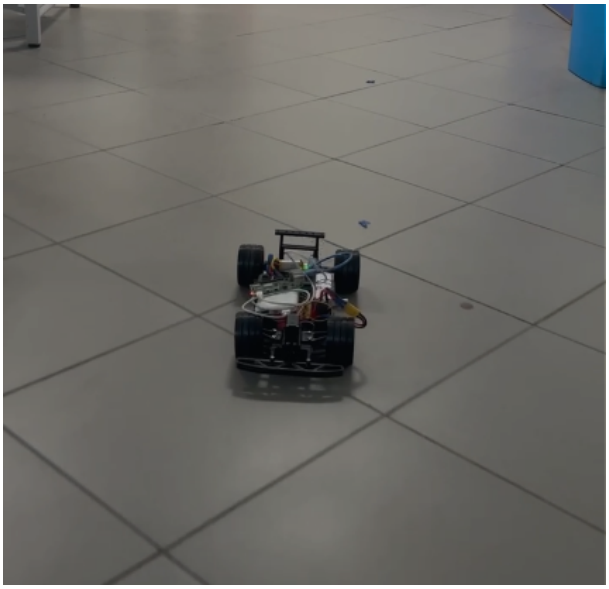

Fig. 10. The Straight line scenario

For obstacle avoidance, the autonomous car successfully executed smooth lane changes in response to the pre-mapped obstacle layout. The Planning ROS node, running onboard, generated suitable trajectories in real-time that were executed accurately by the control system. The system reacted promptly to upcoming obstacles by adjusting both the speed and lane position, replicating the simulation performance with high fidelity.

While some controller gain adjustments were necessary during hardware tuning—particularly for the Stanley controller due to latency and response non-idealities—the overall performance remained robust and consistent with expectations from simulation.

Overall, the hardware results confirmed that the integrated ROS system operated successfully, achieving autonomous navigation, stable control, and reliable obstacle avoidance under real-world conditions.

IV. Conclusion & Future Work

A. Conclusion

This project successfully demonstrated the development, simulation, and real-world testing of a 1/10 scale autonomous vehicle using the Robot Operating System (ROS). Through a systematic and modular approach, we implemented various ROS nodes that handled open-loop and closed-loop control, teleoperation, trajectory planning, and localization.

Simulation results confirmed the correctness of our design, as the vehicle was able to maintain a straight trajectory with a deviation of less than 3 degrees and to perform obstacle avoidance maneuvers effectively. These promising results were validated in the hardware environment, where the vehicle exhibited steady control and reliable response to commands.

The use of a Kalman filter in the localization node significantly reduced the effect of sensor noise, resulting in smoother and more reliable estimation of the vehicle's state. This enabled the controllers to function more accurately and ensured system stability across different scenarios. All nodes were integrated into a single launch file, allowing full simulation of the vehicle behavior in Gazebo, and later transitioning to partial hardware deployment.

Fig. 11. The Obstacle Avoidance scenario

Overall, the project validated the feasibility of using ROS for real-time control and planning in scaled autonomous vehicles, contributing to the broader body of research in F1Tenth-scale robotic platforms.

B. Future Work

Although the core functionality was achieved, several areas can be explored to enhance system robustness and scalability:

- **Sensor Fusion with Additional Modalities:** Future iterations can integrate sensors such as LiDAR or stereo cameras to improve environmental awareness and dynamic obstacle detection.
- **Real-Time SLAM and Mapping:** Implementing SLAM techniques would allow the vehicle to operate in unknown or changing environments without relying on predefined maps.
- **Advanced Path Planning:** The current planning can be enhanced by incorporating algorithms like A*, RRT*, or hybrid A*, enabling more efficient and intelligent trajectory generation.
- **Adaptive and Nonlinear Control:** Replacing simple PID and Stanley controllers with more advanced techniques

such as Model Predictive Control (MPC) or adaptive control could provide better handling under variable conditions.

- **Cloud or Edge-based Processing:** Offloading intensive computations to edge or cloud platforms can improve system performance and reduce the load on the onboard computer.
- **Real-World Scalability Testing:** Expanding the testing domain to larger tracks with moving obstacles or real-time interactive elements would further challenge and validate the system.
- **Machine Learning Integration:** End-to-end learning or reinforcement learning can be applied to train the vehicle to handle more complex behaviors and decision-making.

This project lays the groundwork for scalable and extensible autonomous vehicle platforms and opens up new opportunities for integrating more advanced AI and robotics algorithms in the future.

REFERENCES

[1] O. Natan and J. Miura, "Deepipc: Deeply integrated perception and control for an autonomous vehicle in real environments," *arXiv preprint arXiv:2207.09934*, 2022.

[2] S. Ochs, J. Doll, D. Grimm, *et al.*, "One stack to rule them all: To drive automated vehicles, and reach for the 4th level," *arXiv preprint arXiv:2404.02645*, 2024.

[3] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "Real-time control strategies for high-speed autonomous racing," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1785–1791, 2021.

[4] Z. Huang, S. Chen, Y. Pian, *et al.*, "Cv2x-loca: Roadside unit-enabled cooperative localization framework for autonomous vehicles," *arXiv preprint arXiv:2304.00676*, 2023.

[5] A. names, "A survey on localization for autonomous vehicles," *IEEE Xplore*, 2022.

[6] Valera, F. Valero, M. Vallés, *et al.*, "Navigation of autonomous light vehicles using an optimal trajectory planning algorithm," *Sustainability*, vol. 13, no. 3, p. 1233, 2021.

[7] A. names, "Hierarchical end-to-end autonomous navigation through few-shot learning," *arXiv preprint arXiv:2409.14633*, 2023.

[8] J. Brunn, M. O'Kelly, and R. Mangharam, "Minimum-time trajectory planning for autonomous racing," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 451–465, 2022.

[9] J. Pestana, M. O'Kelly, H. Zheng, and R. Mangharam, "The f1tenth autonomous racing ecosystem: From research to education," in *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, pp. 1–12, 2023.