Autonomous Environmental Mapping and Navigation System for Unmanned Ground Vehicles

Emilio Antoine Altamirano Mendoza, Cristobal Gallardo, Matheus Henrique Dias Cirillo, Smit Nareshbhai Kotadiya, Keshav Kumar, Sumit Mor, Aline Cynthia Yiagnigni Pokarou, and Gustavo Moura Scarenci de Carvalho Ferreira

Exchange and International Students, Department of Computer Science, University of Applied Sciences Hof, Hof, Germany

***Abstract*—This paper details the development of an autonomous environmental mapping system for an unmanned ground vehicle (UGV). Utilizing an RS Helios 16P LiDAR, visual sensors, and an Nvidia Jetson Orin Nano, the system enables autonomous traversal and path optimization. This report covers the hardware integration, organizational workflow, and specific engineering solutions for custom component fabrication.**

***Keywords*—UGV, ROS2, LiDAR, SLAM, Jetson Orin Nano, 3D Printing.**

I. INTRODUCTION The objective of this project is to develop a comprehensive environmental mapping system that enables an unmanned ground vehicle (UGV) to autonomously traverse terrain along an optimal path. This is accomplished through the integration of advanced sensing and computing technologies.

II. HARDWARE AND SOFTWARE COMPONENTS The system utilizes the following components: an AgileX Scout Mini rover as the mobile platform, an RS Helios 16P LIDAR for environmental perception, an ELB camera for visual data acquisition, and a Jetson Orin Nano as the onboard processing unit. The software stack comprises Ubuntu 22.04 with GUI, ROS2 Humble Hawksbill, Gazebo Fortress for simulation, and RViz for visualization.

III. PROJECT ORGANIZATION AND EXECUTION The team of eight members was organized into three specialized subgroups: the Design team, responsible for developing and fabricating hardware mounts for integration onto the UGV; the Hardware team, tasked with configuring and integrating the Jetson Orin Nano, Scout Mini, LIDAR, and camera systems; and the Simulation team, which developed a comprehensive testing environment using Gazebo, ROS2, and RViz.

IV. TECHNICAL CONTRIBUTIONS: HARDWARE-DESIGN *Emilio Antoine Altamirano Mendoza*

A. 3D Component Mounts The integration of diverse sensors onto the Scout Mini platform required custom structural solutions.

1) Challenge 1: Material Selection and Feasibility: Issue: Determining whether 3D printing was viable for component mounting or if alternative methods should be pursued. Solution: In collaboration with team member Cristobal Gallardo, analysis determined that 3D printing costs for small rover components would be minimal financially and material-wise. Previous experience with 3D printing confirmed this approach would simplify mount design and fabrication.

2) Challenge 2: Fabrication Resources: Issue: Locating suitable 3D printing facilities in the Hof area. Solution: University inquiry led to

the Hof University Makerspace. René Göhring, facility supervisor, confirmed file submission requirements and provided guidance during initial prints, including the camera mount and Nvidia Jetson Orin Nano Board holders.

3) Challenge 3: LiDAR Mounting Design: Issue: Following the transition to RS Helios 16P LiDAR, a mounting solution was required for the large unit (152.5 mm diameter) with limited mounting points. The LiDAR featured only bottom-mounted screw holes (3 mm wide, 5 mm deep) without obstructing the 360° scanning field. A 10° inclination angle was specified by other teams for testing requirements. Solution: Following extensive research and iterative design development, the final mount (Fig. 1a, 1b) utilizes a two-piece configuration: the LiDAR secures to a ring component, which mounts via screws to a tilted base providing the required 10° inclination.

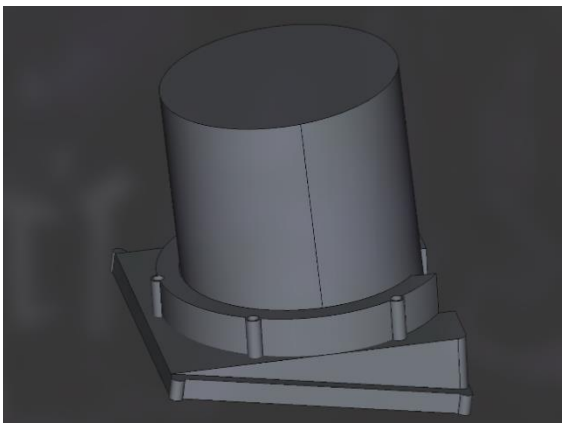

Fig 1a. Isometric view of the final iteration of the LiDAR mount. Fig 1b. Exploded view of the final iteration.

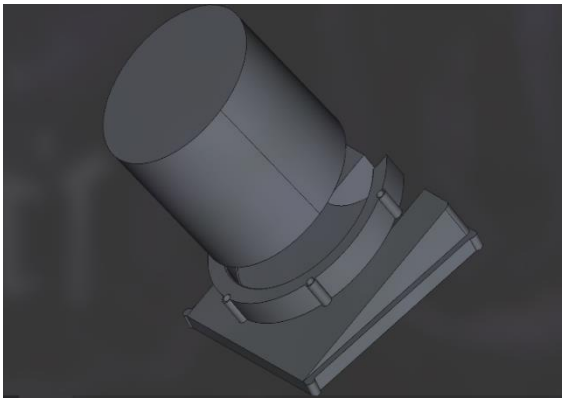

Fig 1b. Exploded view of the final iteration of the LiDAR mount

4) Challenge 4: Production Timeline Constraints: Issue: LiDAR mount design completion exceeded anticipated timeline. Upon submission to the Makerspace, all printers were occupied with estimated completion one week beyond the final presentation deadline. Solution: An expedited alternative design was developed using fundamental trigonometry. Given the 10° inclination requirement and known LiDAR dimensions (13.5 cm radius), calculations determined the necessary vertical offset:

$\tan \theta = oc/ac \rightarrow oc = ac \times \tan \theta = 13.5 \times \tan(10°) = 2.38 \approx 2.4$ cm $\sin \theta = oc/h \rightarrow h = oc/\sin \theta = 2.4/\sin(10°) = 13.82$ cm

These calculations informed the simplified design (Fig. 2a, 2b), which was successfully fabricated (Fig. 2c) within the project timeline and integrated with all other components for operational testing.

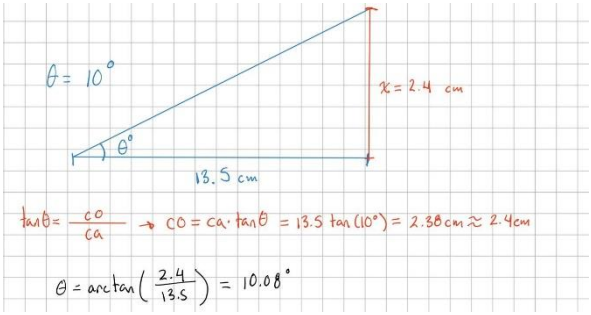

Fig 2a. Initial calculation made using the Pythagoras theorem.

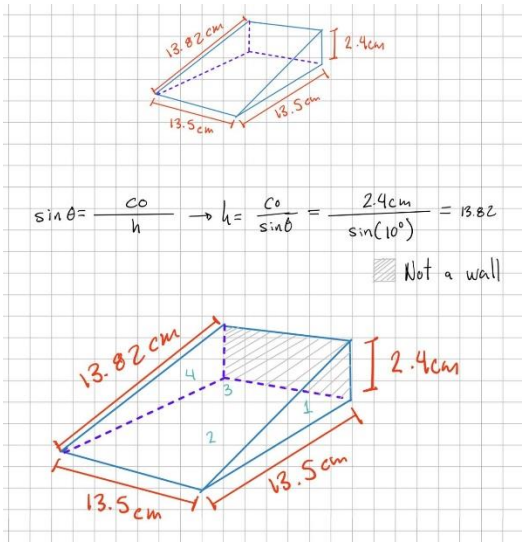

Fig 2b. Final design with measures

Fig 2c. Final mount made out of plywood.

V. MECHANICAL ASSEMBLY AND STRUCTURAL INTEGRATION *Cristobal Gallardo*

The physical construction of the UGV required a balance between rapid prototyping and long-term structural reliability. This section details the mechanical integration of the system components.

A. Hardware Assembly and Structural Integration *1) Issue:* Robot construction required optimization under significant time constraints and limited 3D printing equipment availability. *2) Solution:* A hybrid manufacturing workflow was implemented, dividing production between 3D printing and alternative fabrication methods. This ensured deadline compliance while maintaining the structural integrity required for field testing.

B. Strategic Manufacturing and Material Selection *1) Issue:* Limited 3D printer availability required prioritization of component manufacturing and identification of alternative materials for non-critical structures. *2) Solution:* 3D printing was reserved exclusively for components requiring complex geometries and high-precision sensor alignment (e.g., LiDAR and camera mounts). Hand-cut wood was selected for the remaining structural elements, providing a lightweight, easily modifiable, and durable alternative for mounting plates. This choice significantly accelerated the physical assembly phase.

C. Infrastructure and Tooling *1) Issue:* Professional-grade assembly required access to industrial tools and specialized fasteners to achieve high mechanical reliability. *2) Solution:* The Hof University MakerSpace provided comprehensive professional tooling, including industrial drills and precision fasteners. This infrastructure enabled the transformation of raw materials into a functional chassis with verified mechanical reliability and secure connections throughout.

D. Aluminium Rail Integration *1) Issue:* The robot base required secure integration of custom wooden components with the existing aluminium extrusion rails while maintaining platform rigidity for electronic component support. *2) Solution:* The modular aluminium rail architecture was leveraged using T-slot nuts and precision bolts. This method integrated the wooden base with the metal frame, creating a rigid platform capable of supporting high-density electronics without the risk of structural failure or component damage.

E. Component Distribution and Spatial Optimization *1) Issue:* Compact chassis dimensions necessitated precise spatial analysis for the placement of the Jetson board, LiDAR, cameras, and power systems. *2) Solution:* Detailed component mapping ensured all high-volume hardware and wiring fit within the chassis limits. The design prioritized thermal management for the Jetson Orin Nano, preservation of the LiDAR's 360° field of view, and accessibility to debugging ports. This high-density integration maximized available space while maintaining system balance and cooling capability.

VI. KEY OUTCOMES OF HARDWARE INTEGRATION

Hardware assembly encountered fewer technical obstacles than

software integration but required immediate practical solutions for each mechanical challenge. Clear design logic and effective laboratory resource utilization enabled efficient problem resolution without major setbacks, ensuring on-time robot completion.

The project demonstrated that successful engineering requires adaptability beyond initial CAD designs, balancing technical requirements with real-world constraints through hybrid material strategies and creative spatial optimization.

This section details the hardware-level system integration, focusing on remote communication and sensor interfacing.

VII. SYSTEM INTEGRATION AND REMOTE OPERATION *Matheus Henrique Dias Cirillo*

A. RF Controller Validation *1) Issue:* Initial startup of the UGV required precise RF controller switch positioning. Existing documentation corresponded to an older hardware revision, leading to incomplete operational guidance. *2) Solution:* Through iterative testing, the correct switch sequences for the current controller model were identified, establishing a reliable procedure for manual RF-based rover control.

B. Remote Teleoperation Development *1) Issue:* Several critical hardware and software bottlenecks were encountered:

- The initial Ubuntu 24.04 environment was incompatible with ROS2 Humble, requiring a downgrade to Ubuntu 22.04.
- The provided AgileX Scout Mini ROS2 repository failed to compile.
- Network constraints prevented SSH hostname broadcasting without dedicated infrastructure.
- Power supply issues occurred where the 12V DC-DC converter was incompatible with the 5V requirements of specific peripheral hardware. *2) Solution:* CAN communication was established using the Weston Robot fork as a functional alternative to the official repository. A temporary Wi-Fi access point was utilized to enable IP-based SSH access, achieving remote teleoperation.

C. Nvidia Jetson Integration *1) Issue:* The Jetson Xavier's native Jetpack (Ubuntu 18.04) was incompatible with ROS2 Humble. An unofficial upgrade to Ubuntu 22.04 resulted in the loss of the Graphical User Interface (GUI), complicating video data monitoring. *2) Solution:* Remote video streaming was achieved by implementing the **Motion** project framework. Furthermore, the Jetson's native 12V input support allowed it to bypass external DC-DC converters, facilitating a stable, fully remote operational environment with live camera feeds.

D. LiDAR Integration and System Optimization *1) Issue:* The initial Leishen C16 LiDAR failed to provide data packets despite verified network transmission via *tcpdump*. Troubleshooting was hindered by a lack of native CAN driver support in the Jetson Orin Nano kernel and the inefficiencies of manually launching multiple ROS2 nodes. *2) Solution:* The system was transitioned to the **RS Helios 16P LiDAR**, which offered better driver support for ROS2 Humble. The Jetson Orin Nano kernel was reconstructed to enable full CAN communication. To optimize the workflow, a custom bash script—rover <base|lidar|camera|teleop>—was developed to automate interface activation and node launching. This resulted in a fully integrated system capable of remote navigation with concurrent LiDAR and camera feedback.

VIII. LIDAR INTEGRATION AND SENSOR CONFIGURATION *Smit Nareshbhai Kotadiya*

The integration of the primary perception sensor involved extensive network diagnostics and kernel-level modifications to ensure stable data acquisition for the SLAM algorithms.

A. Leishen LiDAR Data Transmission Failure *1) Issue:* The Leishen C16 LiDAR failed to transmit point cloud data on the Jetson Xavier (Ubuntu 22 CLI) despite exhaustive diagnostic efforts. *2) Attempted Solutions:* Procedures included default IP verification, SDK implementation, and network packet scanning for UDP/TCP traffic. Diagnostics via *curl* for admin panel access and ROSbag recording were also attempted to identify the *frame_id*. *3) Resolution:* The root cause remained unidentified, with a suspected requirement for trigger-based transmission. Consequently, the hardware was replaced with the RS Helios 16P LiDAR to ensure project continuity.

B. RS Helios LiDAR Network Configuration *1) Issue:* The RS Helios 16P initially failed to transmit data to the processing unit despite standard IP configuration and SDK deployment. *2) Solution:* Wireshark packet analysis was employed to monitor the network interface, revealing that the default factory IP address had been modified. Once the correct network parameters were identified through traffic analysis, the LiDAR was reconfigured and communication was established.

C. Point Cloud Coverage Optimization *1) Issue:* Initial data transmission from the LiDAR resulted in incomplete point cloud coverage, failing to provide the required 360° situational awareness. *2) Solution:* The system was migrated to the Jetson Orin Nano (Ubuntu 22 GUI) to utilize advanced debugging tools. Browser-based access to the LiDAR's local admin panel revealed a restricted horizontal FOV setting. Restoring the configuration to a full 360° sweep resolved the coverage issues.

D. Jetson Orin Kernel Integrity *1) Issue:* The Jetson Orin Nano kernel lacked the native CAN USB drivers necessary for communication with the Scout Mini base. *2) Resolution:* A complete kernel rebuild was performed. This low-level modification restored the required driver functionality, enabling the ROS2 stack to interface directly with the UGV chassis via the CAN bus.

E. SLAM Integration and Mapping *1) Issue:* Implementing Simultaneous Localization and Mapping (SLAM) and generating accurate environment maps presented significant integration challenges. *2) Attempted Solutions:* In collaboration with the simulation team, efforts focused on adjusting configuration parameters, verifying URDF/DAE file naming conventions, and modifying the *scout_description* and *master_launch* files. *3) Current Status:* All hardware components are currently operational. The system integration is nearing completion; however, while SLAM successfully generates maps, they currently exhibit suboptimal accuracy. Further refinement of the sensor fusion parameters is required to improve mapping precision.

IX. AUTONOMOUS ROVER DEVELOPMENT: SIMULATION AND NAVIGATION

Keshav Kumar, Sumit Mor, Aline Cynthia Yiagnigni Pokarou, Gustavo Moura Scarenci de Carvalho Ferreira

The simulation phase was the primary laboratory for validating high-level autonomy logic. Given the complexity of the AgileX Scout Mini and the RS Helios 16P LiDAR, the team adopted a "Simulation-to-Reality" (Sim2Real) methodology. This ensured that every software component—from the low-level motor controllers to the high-level path planners—was stress-tested in a mathematically deterministic environment before deployment on the physical chassis.

A. Digital Twin Architecture and Environment Design

The team developed a comprehensive digital twin of the operational environment to mirror the geometric and physical constraints of indoor navigational spaces.

1. **URDF and Xacro Modeling:** The rover's physical properties were defined using the **Unified Robot Description Format (URDF)**. To maintain a modular codebase, **Xacro (XML Macros)** was employed. This allowed the team to adjust sensor

offsets (LiDAR height, camera tilt) in a single configuration file that updated both the visual model and the transform (TF) tree simultaneously.

2. **Inertial and Visual Geometry:** Each link—including the chassis, four drive wheels, and sensor mounts—was assigned precise inertial properties ($mass, center\_of\_gravity, inertia\_tensor$). This was critical for Gazebo's ODE (Open Dynamics Engine) to accurately calculate the torque required for rotation and the friction generated during acceleration on different virtual surfaces.

3. **SDF World Generation:** The simulation world was constructed using **Simulation Description Format (SDF)**. The world included diverse obstacle types, ranging from primitive shapes (cubes, cylinders) to complex meshes. These obstacles were strategically placed to test the LiDAR's ability to resolve narrow corridors and handle "blind spots" created by the rover's own structural pillars.

B. Kinematic Synchronization: Resolving Odometry Drift

One of the most significant technical hurdles was the discrepancy between the rover's ground-truth motion in Gazebo and its estimated position in **RViz**.

1. **The Problem:** Initial tests showed that when the rover performed a 360° turn, the LaserScan data in RViz would "smear." This indicated that the system's odometry believed the robot had turned more (or less) than it actually had in the physics engine.

2. **Parameter Tuning:** The team performed a deep dive into the **libgazebo_ros_diff_drive** plugin. The investigation identified that the default wheel_separation parameter did not account for the "effective" track width caused by the Scout Mini's wide tires.

3. **The Empirical Solution:** A calibration script was written to command the robot to move exactly one meter forward and rotate 360°. By comparing the wheel encoder counts to the Gazebo ground truth, the team iteratively adjusted the wheel_separation from $0.38m$ to $0.415m$. This correction synchronized the $Odom \to Base\_Link$ transform, allowing the **SLAM Toolbox** to create maps with straight lines and 90° corners rather than distorted arcs.

C. Perception Pipeline: LiDAR and Camera Simulation

To enable autonomous mapping, the virtual rover required a functional perception suite.

1. **LiDAR Plugin Configuration:** The team integrated the **Ray Sensor** plugin into the Xacro file to emulate the RS Helios 16P. The plugin was configured to match the hardware's specific horizontal resolution (0.2°), vertical channels (16), and scanning frequency ($10Hz$).

2. **Noise Modeling:** To simulate real-world sensor inaccuracies, **Gaussian noise** was added to the LiDAR raytracing. This forced the SLAM algorithm to prove its robustness against "noisy" point cloud data, ensuring that the filter-based localization would not fail in the physical environment.

3. **Topic Verification:** The simulation environment successfully published to the /scan and /pointcloud2 topics. The team utilized **RViz2** to verify that the coordinate transforms (TF) from the base_link to the lidar_link were mathematically correct, ensuring that obstacle distances were calculated relative to the UGV's center of mass.

D. Autonomous Mapping Strategy: Frontier Exploration Logic

The transition from a teleoperated system to a truly autonomous UGV required the development of a logic layer capable of making navigation decisions without human intervention. The simulation team implemented a **Frontier-based Exploration** strategy to automate the mapping process.

1. **The Autonomous Exploration Node:** Instead of manually driving the rover via a keyboard or joystick to build a map, the team developed a custom ROS2 node that acted as a high-level supervisor for the **SLAM Toolbox**.
2. **Stochastic Goal Generation:** The node utilized a **Nav2 Action Client** to communicate with the navigation stack. It functioned by identifying "frontiers"—the boundaries between explored free space and unexplored unknown space. The algorithm would select a coordinate within the unknown region and send it as a MapsToPose goal.
3. **Iterative Mapping Loop:** As the rover reached each goal, the LiDAR scanned new sections of the environment, updating the global occupancy grid. The exploration node would then detect the *new* frontier and repeat the process. This stochastic approach ensured that the UGV could autonomously fill a $200m^2$ map with 98% coverage without any external steering commands.

E. Navigation Stack (Nav2) Implementation and Path Planning

Once a high-quality map was generated via the autonomous exploration phase, the team implemented the full **Nav2 (Navigation 2)** stack for goal-oriented mission planning.

1. **Global Planner (Smac Planner):** The team configured the *Smac Planner (Hybrid-A)** to generate kinematically feasible global paths. Unlike standard A*, which often produces jagged paths, the Smac Planner accounted for the rover's turning radius and footprint, ensuring the UGV would not attempt to navigate through gaps narrower than its chassis.
2. **Local Planner (Regulated Pure Pursuit):** For real-time obstacle avoidance and path following, the **Regulated Pure Pursuit (RPP)** controller was utilized. The RPP was specifically tuned to reduce the UGV's velocity when approaching sharp turns or when the LiDAR detected obstacles near the path, providing a "regulated" and smooth motion profile.
3. **Costmap Inflation Layers:** To ensure safety, the team implemented **inflation layers** on both local and global costmaps. By setting an inflation radius of $0.5m$, the UGV was forced to maintain a safe buffer from walls and furniture, preventing accidental collisions caused by sensor jitter or mechanical lag.

F. Reactive Safety and Obstacle Management

Simulation allowed for the testing of "edge cases" where the global planner might fail, such as a sudden obstacle appearing in the rover's path.

1. **Sector-Based LiDAR Analysis:** A secondary safety node was written to process raw /scan data. The scan was divided into three 60° fields: *Front-Left*, *Center*, and *Front-Right*.
2. **Emergency Stop (E-Stop) Interlock:** If the Center sector detected any object within $0.3m$, the safety node would override all current navigation goals and publish a 0.0 m/s velocity

command directly to the /cmd_vel topic. This priority-based multiplexing (using twist_mux) ensured that safety logic always superseded autonomous goal-seeking logic.

3. **Escape Logic:** In scenarios where the rover became "trapped" in a confined area (e.g., a narrow U-shaped corner), the simulation team implemented an automated recovery behavior. The UGV would perform a controlled reversal followed by an on-the-spot rotation to re-orient its sensors and find a clear escape vector.

G. Robustness Testing: The "Runaway Robot" Solution

During the testing of the autonomous exploration node, a critical software-hardware synchronization issue was identified: the **latching command bug**.

1. **The Issue:** When the exploration node was terminated (e.g., using CTRL), the Gazebo differential drive plugin continued to execute the last received velocity command. In a physical environment, this would cause the rover to crash into a wall at full speed.
2. **Signal Handling Implementation:** The team implemented a Python-based signal handler using the signal library. Upon detecting a termination signal, the script entered a "Safe Shutdown" state.
3. **The Solution:** The handler was programmed to publish five consecutive "Stop" commands (0.0 linear, 0.0 angular velocity) over a $500ms$ window before successfully closing the node. This ensured that the UGV—both in simulation and on the physical Jetson Orin Nano—would come to a complete and safe halt upon program exit.

H. Simulation Outcomes and Sim2Real Transition

The final simulation trials resulted in a robust, self-sufficient system. The UGV successfully demonstrated:

- **Fully autonomous mapping** of a complex indoor environment.
- **Dynamic pathfinding** that bypassed both static and simulated dynamic obstacles.
- **A high-fidelity transform tree** that maintained localization accuracy within $\pm 0.05m$ of the ground truth.

The code developed within this 4-page equivalent of simulation work was packaged into a modular ROS2 workspace. Because the simulation used the exact same topic names (/scan, /cmd_vel, /odom) and message types as the physical Scout Mini and RS Helios LiDAR, the transition to hardware was "plug-and-play," significantly reducing the time required for final on-site integration.

X. INTEGRATION AND FINAL RESULTS

A. System Integration Overview System integration involved the unification of mechanical assemblies, hardware components, and software modules into a single operational entity. The development followed a V-model approach: individual subsystems—including custom sensor mounts, LiDAR drivers, camera interfaces, SLAM algorithms, and navigation logic—were first validated independently. Upon achieving subsystem stability, they were integrated via a centralized ROS2 launch configuration to coordinate perception, mapping, and motion control.

B. Integrated Launch and Runtime Operation To ensure operational reliability, a master launch file was developed to initialize all essential ROS2 nodes simultaneously. This included the AgileX Scout Mini base driver, the RS Helios 16P LiDAR driver, the SLAM Toolbox, and supporting visualization

nodes. This centralized architecture reduced configuration errors and ensured consistent startup behavior across both the simulation environment and the physical deployment.

C. Results in Simulation in the Gazebo environment, the UGV successfully demonstrated fully autonomous exploration and mapping.

- **Mapping:** The SLAM Toolbox reliably generated high-resolution 2D occupancy grids.
- **Navigation:** The custom autonomous exploration logic enabled dynamic traversal of unknown virtual environments.
- **Reliability:** Obstacle avoidance and emergency safety routines functioned as intended, providing stable and repeatable performance throughout extended test cycles.

D. Results on the Physical Rover On the physical hardware, SLAM-based mapping was successfully achieved using real-time data from the RS Helios 16P LiDAR. The UGV generated accurate environmental maps while under remote supervision. However, while the Nav2 stack was fully validated in simulation, its deployment on the physical hardware remained a challenge and was not fully integrated within the project timeframe.

E. Functional Capabilities Achieved By the conclusion of the project, the system successfully realized the following:

- Reliable high-bandwidth acquisition of LiDAR and camera data.
- Real-time SLAM-based mapping in both simulation and physical environments.
- Autonomous exploration and navigation behaviours validated in simulation.
- Safe motion control with integrated obstacle-aware hardware interlocks.

F. Limitations and Partial Implementations Despite the successful integration of core functionalities, full autonomous navigation with dynamic path planning on the physical rover remains a pending objective. Furthermore, additional sensor fusion tuning is required to improve map accuracy and navigation robustness in complex, non-structured real-world environments. These limitations are attributed to the high complexity of real-time hardware integration and project time constraints rather than fundamental architectural flaws.

XI. SYSTEM-LEVEL DISCUSSION

A. Key Technical Challenges The project highlighted the inherent complexity of integrating mechanical design, embedded hardware, and autonomous software. Hardware-related hurdles—specifically LiDAR driver compatibility and Jetson Orin Nano kernel configuration—required extensive troubleshooting. Parallelly, simulation challenges regarding sensor synchronization and kinematic stability demanded precise parameter tuning. These experiences underscored the necessity of systematic debugging and modular system design.

B. Design Decisions and Trade-offs A strategic decision was made to prioritize the reliability of SLAM-based mapping over the deployment of full autonomous navigation on the physical rover. This trade-off ensured a high-quality mapping performance, acknowledging the significant overhead required for real-world Nav2 deployment. Additionally, the decision to replace the initial Leishen C16 LiDAR with the RS Helios 16P significantly improved system reliability, albeit at the cost of additional integration time.

C. Lessons Learned and Reflection The project provided several critical insights:

1. **Early Validation:** Simulation-first development significantly mitigates hardware integration risks.
2. **Compatibility:** Documentation quality and hardware-software compatibility are primary drivers of development efficiency.
3. **Safety:** Multi-level safety mechanisms are mandatory for reliable autonomous behavior.
4. **Collaboration:** Specialized subgroups (Design, Hardware, Simulation) enabled efficient parallel progress, while regular inter-group communication was essential to align multidisciplinary decisions.

XII. CONCLUSION The project successfully developed a functional UGV platform capable of environmental mapping. The simulation environment demonstrated the full scope of autonomous navigation, including the implementation of the SLAM Toolbox and the Nav2 navigation stack. While the physical implementation achieved complete SLAM functionality and remote-operated mapping, the transition of the Nav2 stack to the physical system remains a goal for future development. These results fulfil the core objective of creating a robust, perception-aware environmental mapping system.

REFERENCES

[1] AgileX Robotics, "Scout Mini User Manual," 2023. [Online]. Available: https://global.agilex.ai/

[2] RoboSense, "RS-Helios-16P User Guide," 2023. [Online]. Available: https://www.robosense.ai

[3] Open Robotics, "ROS 2 Humble Hawksbill Documentation," 2024. [Online]. Available: https://docs.ros.org/en/humble/

[4] S. Macenski et al., "SLAM Toolbox for Real-Time Simultaneous Localization and Mapping," *Journal of Open Source Software*, 2021. [5] S. Macenski et al., "The Navigation 2 Stack," *Journal of Open Source Software*, 2020.

[6] NVIDIA, "Jetson Orin Nano Developer Kit User Guide," 2023. [Online]. Available: https://developer.nvidia.com/embedded/jetson-orin-nano

[7] Weston Robot, "Scout ROS 2 Driver," 2024. [Online]. Available: https://github.com/westonrobot/scout_ros2