University of Toronto

UTRA Autonomous Rover Team (ART): Timbot

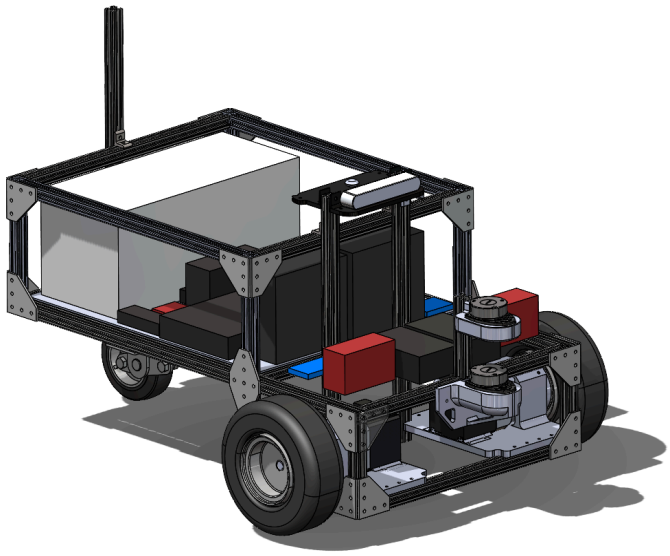


Figure 1: Cad model of Timbot

Date Submitted: May 15, 2026

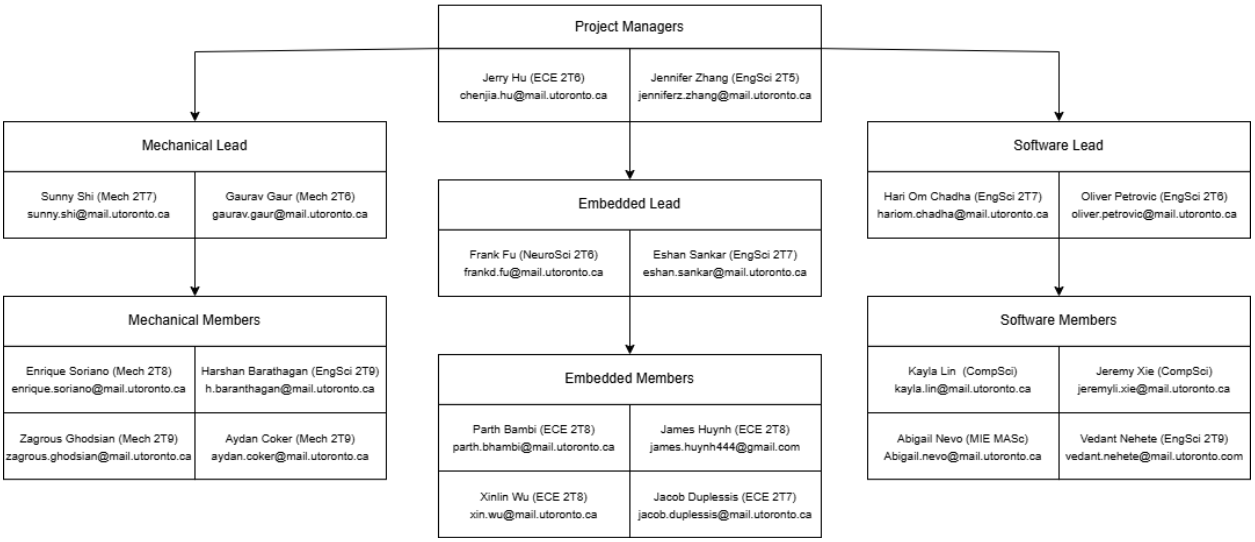
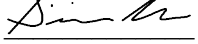


Figure 2. Team Leads and Members

I, Professor Colic, certify that the design and engineering of the vehicle (Timbot) by the current student team has been significant and equivalent to what might be awarded credit in a senior design course.

Professor's Signature: 

Targeted challenge: Auto Nav

1. Conduct of design process, team identification and team organization

1.1 Introduction

Founded in 2008 by students passionate about robotics, the University of Toronto Robotics Association's (UTRA) Autonomous Rover Team (ART) has been a consistent competitor in the Intelligent Ground Vehicle Competition (IGVC) since 2022. We believe in continuous improvement, using each competition's results to refine our designs and passing that knowledge down to future members. This year, we implemented major hardware and software upgrades to our previous rover, Espresso, culminating in our newest model: Timbot.

1.2 Organization

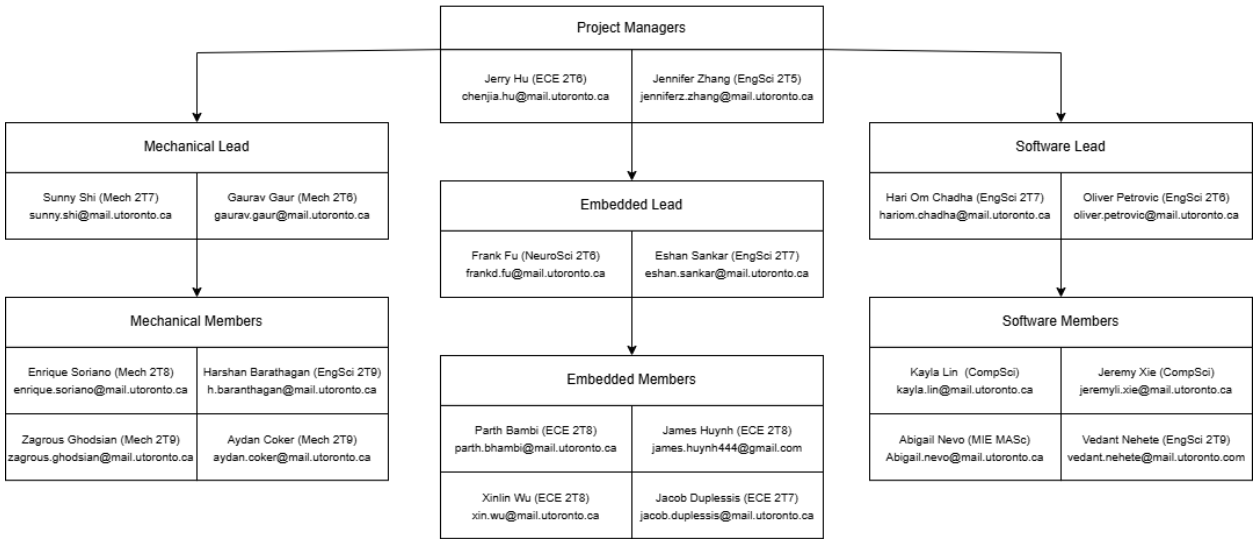


Figure 3. Team Lead and Members

The design team is organized into three subteams: Embedded (Active general members: 5), Mechanical (Active general members: 4), and Software (Active general members: 6). To effectively distribute the workload and maintain clear project focus, multiple team leads are appointed. The leads take responsibility in mentoring the new members, clarifying and designing milestones, and coordinating the members' effort. Project managers maintain regular communication with subteam leads and other UTRA executives to ensure timely task completion and to coordinate support whenever challenges arise. Throughout the academic year, beginning from September to May, the team devotes at least 4 hours per week working on the rover.

1.3 Design Assumptions and Design Process

Members were split into smaller groups within their subteam. Each member would work on a specialized task which the group is responsible for. During the fall, the subteam leads are mainly onboarding new members and assigning some preliminary tasks to familiarize themselves with the newly taught skills. After onboarding, the members are assigned to more complicated tasks that are more vital to the functionality of the rover.

Some design considerations and assumptions our team employs:

- **Money**: The team's tight budget should be allocated across travel expenses for the competition, team social events, and essential components to enhance the rover's functionality. Since this is a design team, the team prioritizes improving the rover ahead of other categories. To minimize costs, we preserved and reused as many parts from previous years as possible. While this strategy significantly reduced the rover's construction costs, the team still faced challenges in meeting the demands of this year's plans with the budget due unforeseen additional expenses required to improve the rover.

- **Resource**: We are limited in members that are licensed to use our school's machine shop, and even more limited in availability to use it. As such, designs are influenced heavily by a small subset of tools all members can use. We avoid lathes, mills, CNCs, and welding wherever possible.
- **Time**: Our team is entirely extra-curricular. We lose a lot of members during midterm and final exam season, and have further member loss after the school year ends.
- **Competition Rules**: We assumed that the competition course and rules have remained majorly unchanged.
- **Human Resources:** The main purpose of our team is education, so we elected to incorporate general members with little or no prior knowledge in robotics. Many of them are in their first or second year of university, yet have shown huge passion when they were applying to be part of the team. The team conducted extensive onboarding as the goal is to feed and grow each members' passion and knowledge in robotics.

Our design decisions process starts with identifying an issue, then the relevant subteams produce candidate solutions that are brought to a leads' meeting. At this meeting, candidate solutions are proposed and other subteam leads are able to comment on them, specifically considering how each design will affect the subteams. If a candidate solution appears to satisfy constraints, a team member is assigned to the design, and a prototype is made. This typically means a CAD model for mechanical systems, circuitry built on breadboards for embedded systems, and prototype code tested in simulation for software systems. If the design appears to work, the member in charge follows through while working with leads to implement a final design version.

2. System and subsystem requirements

2.1 System Engineering Process

To identify critical system and subsystem requirements, the team employed a top-down, iterative Systems Engineering approach. The process began with a thorough analysis of the Intelligent Ground Vehicle Competition (IGVC) rulebook to extract top-level constraints (e.g., safety mandates, course dimensions, and obstacle types). These top-level constraints were then transferred into subsystem requirements across the mechanical, electrical, and software subteams. Important decisions on rover components such as using the ZED Stereo Camera and 12V LiFePO4 batteries are made according to these constraints. Afterwards, testing metrics were refined through Gazebo simulations and hardware-in-the-loop testing. Continuous performance analysis during work sessions allowed the team to adjust target values, ensuring all modules work cohesively toward the AutoNav challenge objectives.

2.2 Subsystem requirements

2.2.1 Mechanical Design

1. Motor Mount Structural Durability

- Driver: High structural integrity under dynamic loading is required to prevent the mechanical fatigue and shear failures previously observed in the 3D-printed mounts.
- Measurement: Number of shear failures or stress fractures observed in the upgraded 8-gauge sheet metal motor mounts during post-integration testing under extended dynamic load.
- Target Value: 0 structural failures

2. Drivetrain Coupler Reliability

- Driver: Reliable torque transfer between the motors and the drive wheels is required to prevent drivetrain slippage and maintain accurate autonomous navigation performance.
- Measurement: Number of observed wheel coupler slippage events or loss of drivetrain engagement during acceleration, turning and prolonged operation testing.
- Target Value: 0 significant drivetrain slippage events

3. Structural Weatherproofing and Panel Integrity

- Driver: Environmental resilience requirements dictate protection against liquid ingress and the strict elimination of panel flex and structural cracking during manufacturing, handling, and operation.

- Measurement: Visual inspection of the upgraded 6mm acrylic panels, rubber gaskets, and silicone caulking for fractures or leaks during handling and outdoor testing.
- Target Value: 0 panel fractures and zero liquid ingress under operating conditions.

2.2.2 Safety

1. Remote Emergency Stop Reliability

- Driver: IGVC safety protocol requires the ability to disable the vehicle remotely in case of erratic behavior.
- Measurement: Maximum distance at which the remote-controlled relay shutoff successfully actuates the rover's halt state.
- Target Value: < 20 meters with 100% actuation success rate.

2. Voltage Protection

- Driver: Protection against functional voltage of 3.3V for ESP32.
- Measurement: The peak voltage during rover operation
- Target Value: < 3.3V in overall rover operation

2.2.3 Electrical/Electronic Components

1. Power Initialization and BMS Stability

- Driver: The need to ensure reliable system startup without triggering the battery management system (BMS) overcurrent protection or blowing the main safety fuse.
- Measurement: Peak inrush current measured across the motor power circuit during a cold initialization sequence using the soft-start switch.
- Target Value: < 8.0A peak current.

2. Battery Duration

- Driver: The requirement for continuous operation of rover through the course without depleting the battery
- Measurement: The time before rover stops functioning due to depletion of battery
- Target Value: > 1h operation time..

2.3 AutoNav Challenge Requirements

2.3.1 Perception

1. Comprehensive Obstacle Detection

- Driver: The AutoNav course features physical obstacles (e.g., barrels) that must be avoided.
- Measurement: The field of view (FOV) and effective detection range of the dual Slamtec RPLIDAR A1 setup.
- Target Value: 270° FOV with a reliable point-cloud detection range of 0.15m to 8m.

2. Robust Lane Tracking

- Driver: The vehicle must remain within the painted boundary lines of the IGVC course.
- Measurement: The frame processing rate and detection accuracy of the YOLOv8 object detection model (or classical filtering) running on the primary laptop via the ZED Stereo Camera.
- Target Value: > 15 Frames Per Second (FPS) processing speed with > 90% confidence in lane boundary identification under variable outdoor lighting.

2.3.2 Driving Logic

1. Multi-Sensor Localization Accuracy

- Driver: High-precision location tracking is necessary for the SLAM ‘cartographer’ package to generate accurate maps and transform frames.
- Measurement: Positional error variance when fusing Wheel Encoders, IMU (PhidgetSpatial), GPS (Columbus P-7), and visual odometry through the robot_localization ROS package.
- Target Value: < 0.1 meters of drift over a 10-meter travel distance.

2. Dynamic Trajectory Generation

- Driver: The move_base package must rapidly update path plans to avoid sudden obstacles or newly mapped dead ends.
- Measurement: Frequency at which the navigation stack generates a new, collision-free trajectory.
- Target Value: Path regeneration rate of > 10 Hz.

2.3.3 Key Performance Indicators (KPIs)

1. Vision-Based Lane Boundary Retention

- Driver: The AutoNav course is strictly bounded by continuous and dashed white lines. To prevent out-of-bounds penalties, the perception pipelines must consistently track these lines despite harsh outdoor lighting or shadows on the asphalt.
- Measurement: The percentage of operational time the vision stack successfully publishes valid lane boundaries to the ROS 2 navigation stack without dropping the track during a continuous run.
- Target Value: > 95% continuous lane tracking uptime

2. End Goal Proximity Accuracy

- Driver: Successfully completing an AutoNav run requires the rover to halt precisely at the assigned destination. If the navigation stack's goal tolerance parameters are configured too loosely, or if the control loop struggles with fine motor adjustments during the final approach phase, the rover will register a "Goal Reached" state prematurely. This leaves the vehicle stranded outside the acceptable scoring radius of the target.
- Measurement: The 2D Euclidean distance between the rover's final localized position (derived from the map frame) and the exact commanded goal coordinates.
- Target Value: < 0.3 meters from the commanded end goal center.

3. Mechanical Design

3.1 Overview

This year’s rover design focused on improving the reliability, durability, and modularity of critical mechanical subsystems based on observations from previous testing and operation. While the overall chassis and drivetrain architecture from the previous iteration continued to provide sufficient structural performance, several components showed signs of wear, vibration-induced instability, and drivetrain inefficiencies during prolonged use. As a result, the mechanical subteam concentrated on reinforcing existing assemblies, reducing vibration, improving wheel power transmission, and enhancing sensor mounting stability.

The primary mechanical upgrades included redesigning the motor mount assembly to improve structural rigidity and reduce vibration through the addition of damping elements, improving the wheel mount and wheel coupler system to mitigate slippage during operation, replacing the worn ventilation fan to maintain reliable thermal management,

designing and manufacturing a new 3D-printed sensor and lighting platform, and adjusting the camera mounting angle to improve the rover's field of view and environmental perception.

These changes aimed to improve both the rover's operational reliability during autonomous navigation and the maintainability of the system during testing and competition.

3.2 Frame Structure & Housing

The rover structure consists of a rigid aluminum t-slot frame with acrylic housing panels enclosing the electrical and computing subsystems. The t-slot chassis was selected due to its modularity, lightweight construction, and ease of maintenance. The modular frame allows rapid subsystem installation, adjustment, and replacement during testing and competition preparation.

The acrylic side panels provide environmental protection for internal electronics while maintaining accessibility for troubleshooting and maintenance. The enclosed housing also helps shield internal components from dust, debris, and light moisture encountered during outdoor operation.

The overall frame structure from the previous rover iteration was retained because testing demonstrated sufficient structural integrity under expected operating conditions. No major chassis failures or excessive frame deformation were observed during recent testing cycles.

3.3 Main Mechanical Components

3.3.1 Motor Mount Redesign

During previous testing cycles, excessive vibration and localized instability were observed in the motor mounting assembly during rover operation on uneven terrain. These vibrations negatively affected drivetrain smoothness and introduced additional mechanical stress into surrounding components. To address this issue, the motor mount system was redesigned with a stronger and more rigid support structure.

In addition, vibration damping elements were incorporated into the mounting assembly to reduce the transmission of motor-induced vibrations into the chassis. The dampers helped absorb high-frequency oscillations generated during acceleration and terrain traversal, improving overall drivetrain stability and reducing mechanical fatigue on connected components. The updated design also improved alignment consistency between the motor and drivetrain components, contributing to smoother power transmission and more reliable operation.

3.3.2 Wheel Mount & Coupler Design

Previous rover testing identified intermittent slippage between the wheel couplers and drivetrain shaft interfaces, particularly during rapid acceleration and turning maneuvers. This slippage reduced drivetrain efficiency and negatively impacted motion accuracy during autonomous navigation.

To mitigate this issue, the wheel mount and wheel coupler assemblies were redesigned and reinforced to provide a more secure mechanical connection between the motors and drive wheels. The updated coupler configuration improved torque transfer reliability while reducing rotational play within the drivetrain system. The wheel mounting interfaces were also strengthened to improve structural robustness during extended operation over uneven terrain.

These modifications resulted in improved drivetrain responsiveness, reduced wheel slip, and more consistent rover movement during testing.

3.3.3 Ventilation & Thermal Management

Reliable thermal management remained an important consideration due to the enclosed electronics compartment and prolonged system operation during outdoor testing. Inspection of the rover revealed that the existing ventilation fan had experienced wear from previous operational use, reducing airflow effectiveness. To maintain adequate cooling

performance, the worn fan was replaced with a new unit of equivalent specifications. The updated ventilation system restored proper airflow throughout the electronics enclosure, helping regulate internal temperatures and improve the reliability of onboard electrical components during extended operation.

3.3.4 Sensor, Camera, Lighting Mount

To improve modularity and organization of external sensing components, a new 3D-printed mounting platform was designed and manufactured for the rover's sensors and lighting systems. The platform consolidated multiple external components into a single integrated structure, simplifying installation and maintenance procedures. The use of additive manufacturing allowed the geometry of the platform to be customized to match the required sensor layout while minimizing weight. The design also improved cable routing and reduced unwanted movement of mounted components during rover operation. The resulting assembly provided a more stable and organized sensing platform for autonomous navigation tasks.

The camera mounting assembly was modified to optimize the camera viewing angle for autonomous navigation and obstacle detection. Previous testing indicated that the original mounting orientation limited the effective field of view in certain operating conditions. The revised camera mount adjusted the camera angle to improve forward visibility and environmental coverage while maintaining structural stability during rover movement. This modification improved the quality of visual data captured by the rover and enhanced the effectiveness of perception-related subsystems during testing.

3.3.5 Weatherproofing

To improve on waterproofing, one of the encompassing changes was replacing the 3mm acrylic panels with thicker, 6mm panels of the same material. Throughout the manufacturing and handling of the previous iteration of Espresso, the 3mm panels cracked easily, creating gaps for liquid to seep into. Doubling the thickness of the panels was done since it was the largest thickness that would still fit with the existing dimensions of the t-slots used for the chassis. Furthermore, the panels were lined with rubber gaskets and silicone caulking to increase the waterproofing. A secondary measure, a lightweight covering (the poncho), was placed over the rover to act as another barrier.

3.3.6 Chassis Design

The rover's structure consists of a 36" by 22" ladder frame, with an upper and lower subassembly. The upper assembly houses the payload and all electronics except the sensors and motors. The lower sub-assembly houses the drive end.

Last year, the 3D printed motor mounts experienced a shear failure during regular use. This failure was attributed to mechanical fatigue. To mitigate this, a new design is implemented. The new design includes an updated motor mount constructed of 8-gauge sheet metal, with spare robust 3d-printed alternatives to bring to competition. With these spares and new parts, the effects of mechanical fatigue should be minimal.

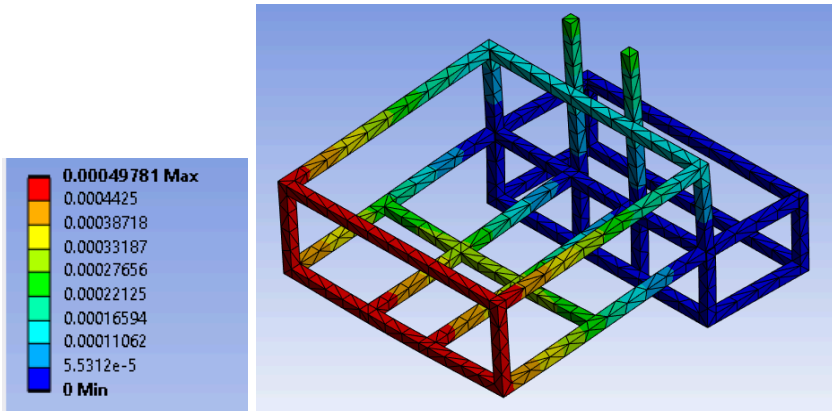


Figure 4 : Ansys static load simulation on the body of the rover.

3.3.7 Drive and suspension

The rover relies on a differential drive system conforming to a unicycle model. The two drive wheels are in the front, with a freely rotating caster wheel in the rear. Each drive wheel is driven by a 300W DC motor, providing ~16 Nm of torque. This allows the rover to easily reach the maximum allowable speed of 5 mph, climb the ramp, and do so with design room to spare. Since the t-slot chassis does not mitigate the effects of vibration, anti-vibration measures include suspension coils in the caster wheel and rubber tires with low pressure. These anti-vibration measures have proven to be sufficient from testing results on asphalt surfaces, which were conducted to mimic the competition.

3.4 Requirement-driven mechanical design & performance evaluation

The mechanical design of the Timbot rover prioritizes operational reliability, structural integrity, and stable perception. To ensure robust terrain mobility, we implemented a differential drive system powered by 300W BLDC motors, and redesigned the wheel couplers to completely eliminate previous drivetrain slippage. Structurally, upgrading from 3D-printed to 8-gauge sheet metal motor mounts successfully mitigated past mechanical fatigue and shear failures. Additionally, newly adjusted, vibration-damped sensor and camera platforms guarantee stable, optimized fields of view during chassis movement. Field testing confirms these upgrades meet all target metrics: the vehicle consistently achieves the 5 mph speed limi. Furthermore, both the 8-gauge mounts and 6mm chassis panels have demonstrated zero fractures or shear failures under extended load, confirming the system is highly durable and well-equipped for competition rigors.

4. Safety

4.1 Mechanical and wireless E-stop systems

The mechanical E-stop is a large red pushbutton switch located 2ft off the ground, positioned at the center of the rover's rear. The wireless E-stop is an antenna relay that is only controlled by a remote controller, with 20m of range. Both E-stops directly disconnect components from the 24V power via a hardware high-side switch without any software control. This guarantees the vehicle can halt even during a software freeze.

4.2 Requirement-driven design & performance evaluation

To ensure operational safety, we developed a dedicated ESP32-based fault monitor utilizing FreeRTOS to manage overcurrent, overvoltage, and temperature protections by deterministically disconnecting power when thresholds are exceeded. As described in section 5.3, the system protects the 24V dual-LiFePO4 battery circuit by using an analog optoisolator to cut power if voltage exceeds functional voltage of 3.3V for ESP32, effectively preventing regulator or battery surges. To protect the motor controllers, an ACS712 sensor triggers a cutoff if the current exceeds 7A, backed by an 8A fail-safe fuse. Additionally, a custom soft-start circuit caps inrush current at 3.8A to prevent false shutoffs during normal startup. Field testing successfully validated all these automated protections, alongside a reliable hardware-level E-stop that achieved a 100% actuation success rate at a 20-meter range, ensuring immediate and guaranteed halting on the course.

5. Electronic and Power Design

5.1 Overview

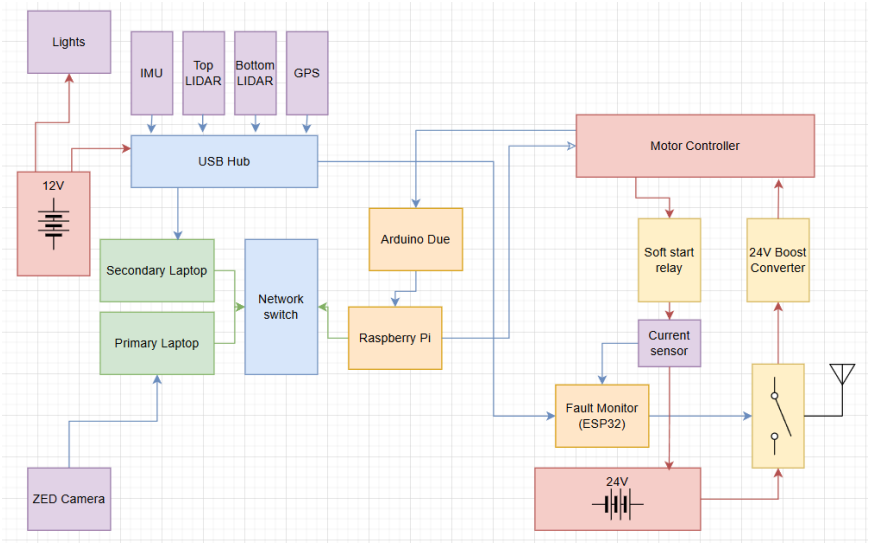


Figure 5: High-level overview of the electronics systems.

The overall electronic system is divided into two main circuits - the sensory/compute circuit and the motor power circuit. The sensory/compute circuit contains all 'intelligent' parts of the electronics suite, including the laptops, Raspberry Pi, microcontrollers, and all sensors; a single 12V LiFePO4 battery powers them. The motor power circuit consists of two 12V LiFePO4 batteries, the motors, the boost voltage regulators, and safety components including fuses, emergency stops, and relays. This design intends to have as much isolation between the two systems as possible, allowing for either system to be tested independently, and safeguarding the sensor/compute circuit from interactions with high amperage. The only point of connection between the two circuits is the RPi, which communicates with the motor controllers through optocouplers (achieving galvanic isolation).

5.2 Description of the significant power and electronic components

The power circuit consists of six PC817 optocouplers to facilitate communication with the sensor/compute circuit, an ACS712 current sensor, and several LM60CIZ temperature sensors.

The motors contain Hall effect sensors, which generate a square-wave signal at a frequency proportional to the motors' rotational speed. These feedback signals are used to estimate the rover's velocity and displacement. However, the signals are very delicate and prone to noise, so we use optocouplers and an RC filter to isolate and denoise the signals.

The sensor/compute circuit consists of a laptop, a Raspberry Pi (RPi), an Arduino Due, and an ESP32. The sensors (i.e. camera, LiDARs, IMU, and GPS) are connected to the laptops through a USB hub powered by a separate 12V LiFePO4 battery to extend runtime. The laptop and Raspberry Pi constitute the computing system on which all ROS nodes are run, connected to each other via an ethernet cable. The Raspberry Pi serves as the bridge between the laptops' high-level ROS nodes and the motor control circuit; the RPi communicates speed controls to the BLD750 motor controllers through optocouplers on a custom-designed printed circuit board (PCB). An Arduino Due is connected to the RPi and solely serves to read Hall effect feedback from the motors via the motor controllers, and communicate that information to the RPi via USB. Our use of a dedicated microcontroller ensures complete accuracy and signal integrity of wheel odometry.

The fault monitor PCB consists of an ESP32 reading temperature, voltage, current, temperature, and alarm status to automatically shut off the rover if the predetermined safe operating conditions for temperature, voltage, and current are violated. The low-level control of this system is accomplished by an ESP32, which takes in readings, outputs logs, and

directly disconnects the rover from power. The power circuit is connected with ESP32 through an analog optoisolator to prevent exceeding the limit of 3.3V of ESP32. For our embedded code, we leverage FreeRTOS to achieve readings, shut-offs, and status logging (in both hardware and software) within extremely short and consistent deadlines.

5.3 Power distribution system

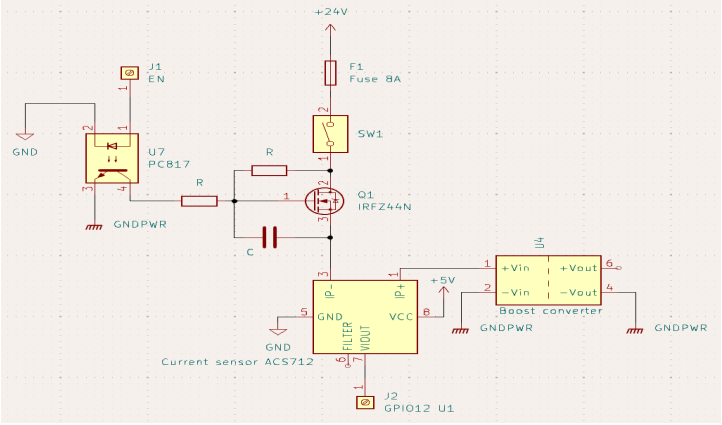


Figure 6: Motor drive circuit

The power circuit is driven by two 12V 8AH LiFePO4 batteries in series, which together provide 24V to the motors and motor controllers. The motors are rated at 300W, but the speeds at which we run them only require a maximum of 6A and an average of 2.5A, resulting in a runtime of 1.3-3.2 hours, depending on rover speed. Both batteries recharge at a rate of 144W, resulting in a recharge time of 1.3 hours.

The primary concern with these batteries is that their low capacity causes their internal battery management systems (BMS), developed by the manufacturer, to shut off the batteries during rover startup due to the high inrush current. This is mitigated through the use of a soft start switch, which limits the rover's initial current by routing power through a power resistor during its startup sequence, before switching to a short during normal operation.

Safety features for the power circuit include a mechanical push-button E-stop and a remote relay E-stop (see Section 4.1), a 24V boost converter, 8A fuse, and a fault monitor (see Section 5.2).

5.4 Requirement-driven electrical design & performance evaluation

The electronic and power distribution systems were designed to ensure circuit isolation, power stability, and real-time safety. To protect low-voltage logic hardware, PC817 optocouplers provide complete galvanic isolation between the 12V compute and 24V motor circuits. A custom soft-start circuit stabilizes power delivery during initialization, throttling the high inrush currents that previously triggered battery management system (BMS) shutoffs. Furthermore, a dedicated ESP32 fault monitor utilizing FreeRTOS guarantees deterministic safety intervention, immediately disconnecting high-current power if temperature, voltage, or current thresholds are exceeded.

Recent testing confirms the system operates highly efficiently and within design parameters. The soft-start network successfully caps peak inrush current at approximately 3.8A, remaining well below the 8A main fuse rating and eliminating startup power failures. Additionally, field testing at a nominal 2.5A draw demonstrated a sustained runtime of 3.1 hours. This comfortably exceeds the 1-hour requirement and aligns perfectly with the theoretical maximums of the dual LiFePO4 configuration.

6. Perception

6.1 Overview and Challenge Scope

The UTRA Autonomous Rover Team is exclusively targeting the **AutoNav Challenge**. Therefore, our perception system is highly specialized for identifying continuous lane boundaries, potholes, and static physical obstacles (such as barrels and barricades) in an off-road environment. The system utilizes a ZED stereo camera for machine vision tasks and a dual Slamtec RPLIDAR A1 setup for physical obstacle detection.

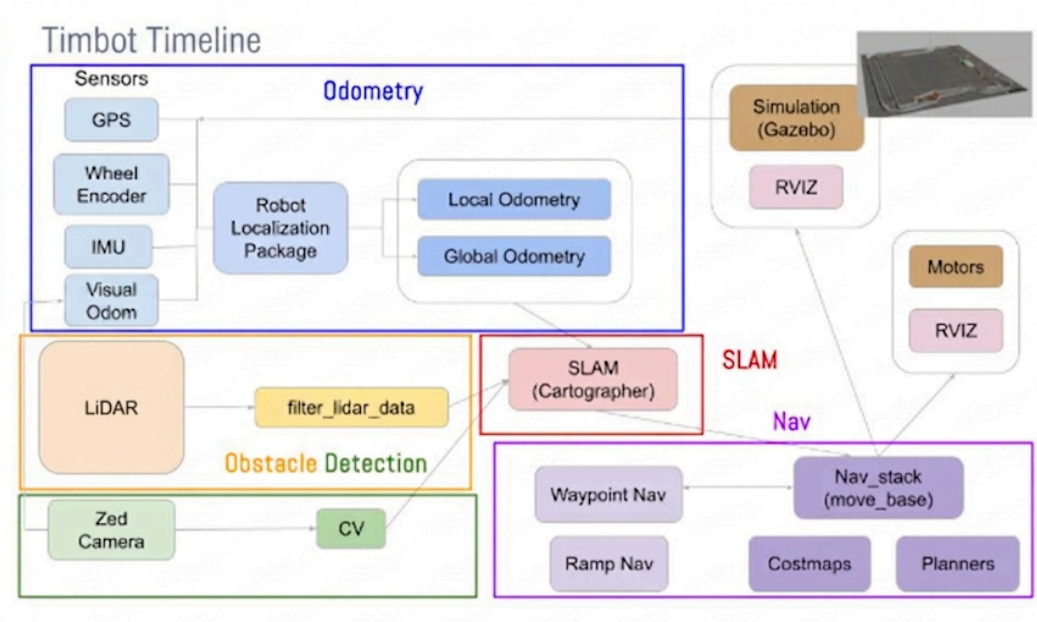


Figure 7: Software pipeline diagram

6.2 Course feature and items identification

6.2.1 Lane, Pylon, and Pothole Detection

To encounter various situations and ensure we always have robust and fast lane detection and tracking according to the listed requirement, we implemented both classical and deep learning approaches for lane detection.

Our classical approach mainly relied on feature detection and filtering techniques. First, the image is converted to greyscale. A rough mask is retrieved by filtering pixels within a saturation and value range near white. The connected components of the mask are retrieved and filtered by minimum area. A mask of the barrels is retrieved by a similar filter for the color orange. This mask is then inflated around each barrel and subtracted from the white lane mask. Since shape is not inherently considered, the classical model can detect both lanes and potholes. As a result, we reached a 2ms inference speed and robust detection.

Our deep learning approach mainly utilizes the YOLO26 segmentation model. The model was first trained on renders of objects in IsaacSim. It was then further fine tuned using a manually labelled dataset of 305 pictures, from footage of previous competition tracks. The results are more accurate near the camera than further away. This resulted in an inference speed of 5ms.

Simulation in Gazebo posed a challenge, because the lane lines lacked texture and the map was low resolution. While the ML approach performed decently on real competition footage, it often failed to detect anything in sim or had large holes. This motivated us to use the classical approach for lane and pothole detection.

A 2D mask outlining the section of the robot in front of the camera was applied (not shown below), to prevent erroneous detections of the robot.

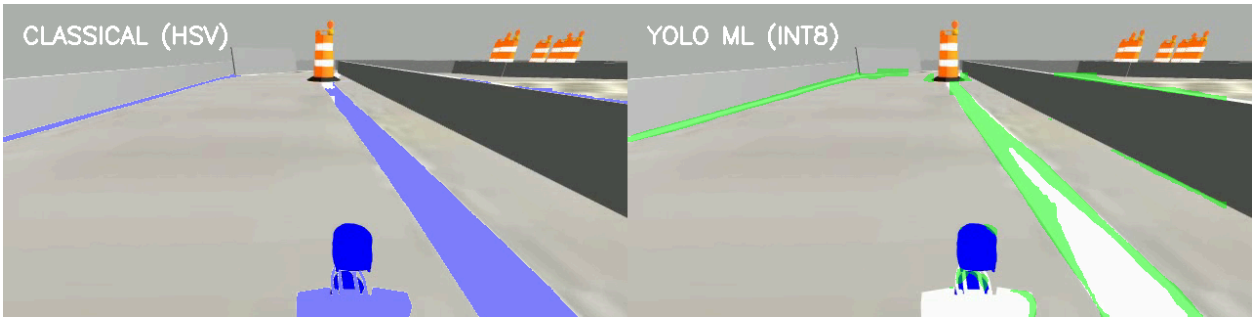


Figure 8: Lane detection result on sim footage for both approaches

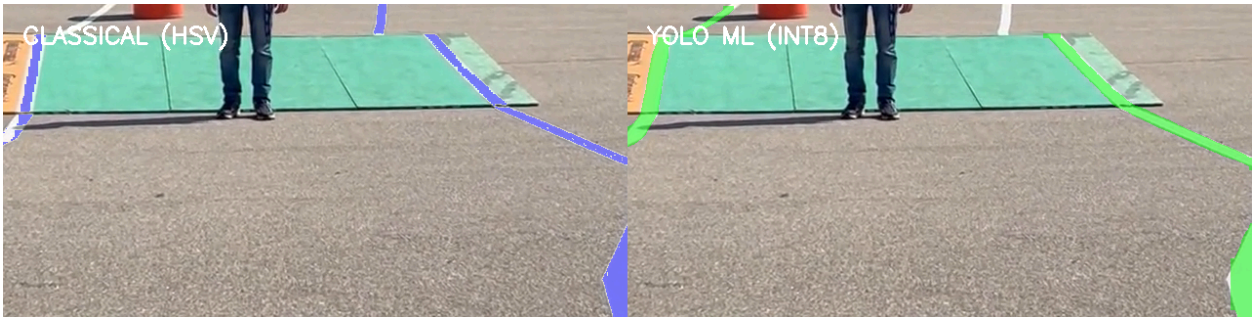


Figure 9: Lane detection result on IGVC 2025 footage for both approaches

6.2.2 CV Integration into ROS

The integration of CV pipeline detection is straightforward. The lanes and potholes are labelled using different numbers. Through bitwise OR of all the detection masks, a map containing all detections is obtained. Then, the depth reading from the ZED camera and image_geometry library is used to convert the 2D map into a 3D map.

6.2.3 Detection of obstacles with physical volume

To detect physical obstacles like barrels and barriers, we rely on a Slamtec RPLIDAR A1M8 for its high accuracy and speed. 3D LiDARs were once considered, but they are beyond our budget, and computer vision is less accurate. Due to the relative sparsity of the data, it was also computationally feasible for us to avoid filtering the point cloud, so we directly added points detected by our lidar to the map. This assumes that anything the lidar detects is an obstacle. We handle ramp detection separately to avoid recognizing it as an obstacle. This reduced computational expense.

In addition to LiDAR-based obstacle detection, we use a ZED point-cloud processing pipeline to identify obstacle-relevant regions from the camera-provided point cloud. The pipeline applies spatial filtering and clustering to separate likely physical obstacles from irrelevant ground and background points, producing a dedicated obstacle point-cloud output for perception.

6.2.4 Ramp detection

We implemented a ZED point-cloud-based ramp detection method. The camera-provided point cloud is filtered to a local region in front of the rover and organized into an elevation grid. The algorithm identifies connected terrain cells with a consistent positive forward slope while rejecting isolated height spikes and low-support regions. Accepted ramp-like regions are then published as a dedicated ramp point-cloud output, allowing sloped terrain to be distinguished from ordinary obstacles.

6.3 Representation of course and mapping

The map is generated using the SLAM package ‘Cartographer’ [1], which provides loop-closure, helping the robot identify previously seen locations. Mapping works in two main components: local and global SLAM. Local SLAM refers to building smaller ‘local’ maps, called submaps, by placing the current LiDAR scan based on odometry information and adjusting its position to the best fit within the submap. Global SLAM operates in the background to

optimally connect different submaps via loop closure. Submaps are initially joined together based on odometry information. When submaps match, constraints are added that tweak the relative positions of the submaps to determine the optimal connections. The final map is published as an occupancy grid.

6.4 Requirement-driven perception pipeline design & performance evaluation

The perception pipeline is designed to meet the strict requirements for robust, accurate, and high-speed obstacle detection and mapping. To ensure reliable lane detection across varying conditions, we implemented both classical computer vision and deep learning approaches. Both methods achieve over 90% detection accuracy; the classical technique optimizes processing speed, while the deep learning model provides enhanced robustness against environmental variations. Additionally, our 2D LiDAR system demonstrates strong performance, offering an approximate 270-degree field of view (FOV) and detecting obstacles with a margin of error of less than 10 cm.

7. Driving Logic

7.1 Lane following and obstacle avoidance logic

7.1.1 Odometry and Localization

Tracking the robot's current position is done by fusing data from the IMU, GPS, Hall sensors (Wheel Encoder), ZED depth image, and LiDAR scans using the robot_localization package [6] and the SLAM package 'Cartographer' [1]. Local odometry is generated by fusing IMU and Hall sensor data through an extended Kalman filter (EKF). 'Cartographer' uses local odometry along with GPS data, LiDAR scans, and depth image point clouds from the ZED camera to estimate the robot's current position in the map. In addition, GPS data, is entered into the navsat_transform_node (provided by robot_localization), along with an initial orientation and heading given by the IMU, to generate a transform from UTM (Universal Transverse Mercator) coordinates to Cartesian coordinates [6]. As 'Cartographer' does not provide covariances with its estimated pose, a covariance matrix is attached by the "pose_relay" node and passed into another, 'global' EKF to smooth the output and determine global odometry for the nav_transform_node to transform GPS into global waypoints.

7.1.2 Ramp Navigation Mode

In order to tackle the challenge of navigating over a ramp, we decided to take a relatively simple approach, rather than overcomplicate the problem.

We implemented a ZED point-cloud-based ramp detection method, which identifies connected sloped terrain regions and publishes them as a dedicated ramp point-cloud output.

The lane lines on the ramp keep the rover from driving off of the ramp, and the waypoints before and after the ramp are enough to guide the rover in the right direction. This way, we can navigate over the ramp within normal operation, without any specialized additional features.

7.1.3 Goal Selection and Path Planning

To set goals, we enter the GPS coordinates into an ordered JSON file, with a parameter for going clockwise or counter-clockwise (depending on if we need to go the opposite way around the course) and the amount of laps to complete. Then, we send the transformed pose to a Nav2 goal. Nav2 handles the path generation based on the costmap_2d, which reads directly from the Cartographer map (Figure 11), as well as the . Afterwards, we query our GPS to see if we reached the point.

For path planning, we rely on 4 key packages: Nav2 [2], costmap_2d [3], planner_server [4], and controller_server[5]. Nav2 serves as our high-level interface and is used to coordinate the costmap and path planning algorithm into movement commands. The costmap_2d package, reading from the SLAM map, creates a cost-based representation. "planner_server" and "controller_server" use this costmap to compute a global and local path that avoids obstacles (higher cost). The planner_server and controller_server use multiple plugins to generate the global plan and velocity commands respectively. Specifically, for the global_planner we use the NavFn planner, which uses an expanded A-star

search for the global plan, and the controller planner uses the DWB controller, which creates velocity commands to follow the plan outlined by the global planner using a dynamic window approach. As new obstacles are detected, they are added to the SLAM map, which updates the costmap and subsequently the path plan. The team also has a failsafe path planner option described by a behaviour_tree, which outlines actions that the rover should take when in a recovery state.

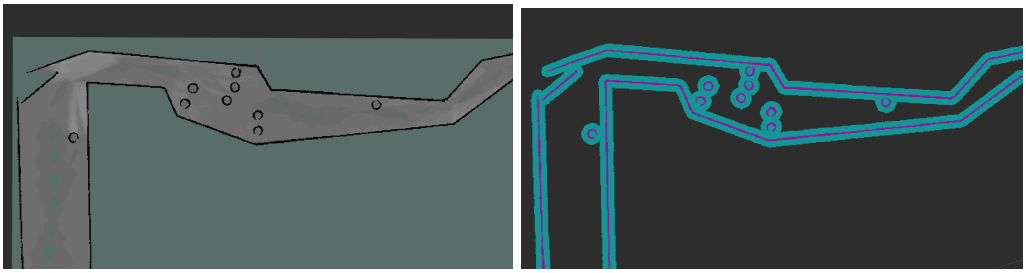


Figure 10: Map generated using Cartographer (left) vs costmap generated by costmap_2d (right)

7.2 Use of perception system and motion generation

The command velocity (cmd_vel) of the rover mainly comes from two sources - the navigation velocity (nav_vel) computed by Nav2 in auto-navigation mode and the velocity the team manually set through teleoperation (key_vel). The Twist_mux package enabled the rover to choose which velocity to use as the cmd_vel under different operating modes. The motor control node subscribes to two sets of ROS2 topics: the desired velocity commands for the left and right wheels (attained from cmd_vel) and the odometry stack's measurements of the actual velocity for the left and right wheels. The difference of these values (for both wheels) is the error signal, which is processed by a Proportional-Integral-Derivative (PID) control algorithm. PID control multiplies the the error by a constant (proportional), finds the average between the previous and new errors (integral), and finds the rate of change between the previous and new errors (derivative); the weighted sum of these terms (weighted by a set of manually-tuned coefficients) is sent as a GPIO signal to corresponding wheel's velocity input in the motor control circuit mentioned in Section 5.2. This method of motor control embodies a significant improvement over our previous method and conforms to the standard of PID control in the automation industry.

8. Key Performance Indicators

To ensure the rover can stay within the course boundaries, our target is to achieve greater than 95% continuous lane tracking uptime throughout the rover's operation. To measure this, we set up an artificial track using white tape to simulate the real white paint used for lane detection on the competition course. We then kept the rover navigating through this track and logged how consistently the vision stack published valid lane boundaries to the ROS 2 navigation stack. In the end, we successfully achieved higher than 95% lane tracking consistency over the total running time, demonstrating that the perception pipeline can effectively handle variations in outdoor lighting and shadows.

To ensure the rover successfully completes an AutoNav run without prematurely registering a "Goal Reached" state and halting outside the scoring radius, our goal is to achieve an end goal proximity accuracy of less than 0.3 meters from the commanded target center. To measure this number, we commanded the rover to navigate autonomously to precise coordinates in our test area, relying on the local planner and control loop to handle the fine motor adjustments during the final approach phase. We then calculated the 2D Euclidean distance between the rover's final localized position—derived from the map frame—and the exact commanded goal coordinates at the precise moment the rover stopped. In the end, we consistently achieved final stopping distances well within our target threshold, proving that our navigation goal tolerances and physical control systems effectively guide the vehicle to an exact and reliable halt.

9. Analysis of Complete Vehicle

9.1 Lessons Learned

From a software perspective, operations proceeded very smoothly this year. We initiated the development cycle by successfully migrating to ROS2 and validating our entire stack in simulation. Another major success was the adoption of a modular workflow; assigning dedicated teams to specific modules fostered independence and significantly streamlined the debugging process. However, while the embedded and mechanical systems were ultimately successful, hardware delivery delays created a bottleneck for software deployment and physical testing. Because hardware-software integration consistently introduces unforeseen challenges, field testing needed to commence earlier. Moving forward, we must prioritize maintaining an operational testbed rover throughout the project lifecycle to facilitate continuous, real-world software debugging.

9.2 Top Hardware Failures

9.2.1 Embedded

Failure Components	How are they addressed and prevent future failure
High inrush current causes undervoltage and heating	Fault monitor pcb is dedicated to such situations and voltage regulators will help prevent this from happening.
Components/connections physically make contact with each other and cause short circuits.	Via WAGO connectors, grommets, and electrical tape, we have effectively built our electrical harness to ensure that this does not happen. If failures still occur, the rover will be swiftly stopped, and we shall unit-test electrical components with digital multimeters to determine their functionality.

Table 1. Potential Embedded Hardware Failures and how to address them

9.2.2 Mechanical

Failure Components	How are they addressed and prevent future failure
Pole supporting emergency stop keeps wobbling	Reprinted supporting parts and fixed the pole. Spare parts are kept to ensure rapid fix for future failure
Vibration causes the wheel to loosen and wobble during trial runs	Redesigned more robust wheel couplers and eliminated the wobbling issue

Table 2. Potential Mechanical Hardware Failure and how to address them

9.3 Software Testing, Tracking, and Version Control

Git, via Github, was used to maintain version control and to keep a record of code development. Members would make branches of the main codebase to develop specific components or to assess and fix specific bugs. This allowed developers to test their code as needed, without disrupting the work of others, or the main branch. After the issue has been resolved or the new element has been properly implemented, a pull request would be made on Github. This then starts a code review session where peers would assess the changes for accuracy, quality, and style. Only once reviewed and accepted would the branch be merged, ensuring stability of the main branch in the codebase. The software leads are the only ones capable of merging branches into main without issuing a pull request, all other merges *must* go through a pull request. Each member was encouraged to create their own branch to write their own contributions in, and create a pull request to the main branch when they were finished with the branch.

Because of ROS2's inherent modular nature, individual members could test code by building their packages and leveraging the gazebo simulation to test for any errors, without needing to wait on other members to implement their packages. To further enable independent work, members were assigned packages to manage (as well as an individual git branch), and were in charge of carrying out tasks for their respective packages. Once the members deemed that

their changes were error free, the pull request process described above was then undertaken to ensure that the code is error free. This way, integration testing with the rest of the packages can be done

9.4 Virtual Simulation Testing Environment

Gazebo and RViz were both used in our virtual simulation environment to test and visualize robotic behaviour in a controlled environment. The details and specifications of our robot were set up in a URDF file, including the locations of sensors and the dimensions of our robot. Additionally, we replicated possible IGVC courses using the course map, known waypoints, knowledge from past competitions, and the constraints detailed in the rulebook. Using this, Gazebo provided accurate simulations of the sensor data and the robot's movement that would result from the generated environments we provided. RViz enhanced this simulation by providing a visualization of the robot's perception of the environment and the robot's planned path. Debugging and fine-tuning the robot's behaviour during development was made much easier by RViz's ability to monitor the outputs of sensor data.

9.5 Physical Testing to Date

The current CV lane detection and ROS integration, as well as mapping, were tested using 2 strips of white tape approximately 5 ft apart and 7 ft in length. We stuck the tapes to the asphalt, then we pointed the camera at them at different heights and tilts and holistically measured the inference. The results were promising, with the lane projections appearing parallel to each other. This test was performed a couple times in different locations and under different lighting conditions. Previous attempts at testing involved a cardboard target with white strips, observed indoors. Due to the lighting, the ZED camera's autocalibration resulted in underexposed images, which our classical CV pipeline struggled with. Relaxing white sensitivity was not enough to fix the issue. Manual calibration settings were added for testing indoors.

Sensor testing involved two phases: a stationary test and a "return to start" test, initially conducted in simulation and then on the actual rover. The stationary test served as a basic assessment of odometry drift speed and noise levels. In contrast, the "return to start" test constituted a mini path designed for the rover to navigate, ultimately returning it precisely to its initial position and orientation. Any deviation in positional values indicates potential issues with sensor readings. These tests encompassed rotations, turning motions, and both vertical and horizontal movements, ensuring a comprehensive evaluation of all possible maneuvers. Results highlighted rapid accumulation of drift during simultaneous movement and turning, accompanied by highly noisy acceleration and velocity readings throughout the path. These findings lead us to develop new algorithms aimed at mitigating these issues. In addition, testing of ZED cameras through similar methods aforementioned brought to light the issue of the ZED SDK sometimes not detecting the ZED camera due to the camera USB cable being damaged, which would effectively restrict any sensor information coming from the camera (RGB-D and visual odometry). To remedy this, we forewent the SDK by using an open source wrapper that takes vision and depth data directly from the camera.

Real-world testing of odometry reliability is conducted in a similar style to simulation. The rover is taken outdoors and operated via teleoperation to observe sensor outputs in a controlled environment. Such testing enabled us to detect and fix unexpected issues affecting sensor accuracy, such as those caused by the rover's inherently tilted surface where the sensors are mounted (i.e IMU, Camera, etc.).

Specifically, we tuned the covariances sent to the EKF for the IMU & GPS. This was done by recording data streams from each sensor while it was kept stationary for 120 seconds and then calculating the variance for each measured dimension. For the IMU, we calculated the variance for angular velocity and linear acceleration in the x, y, and z axes. We constructed a covariance matrix using these measured variances as the diagonals. For the GPS, we similarly kept it still for 30min and calculated covariances for latitude, longitude and altitude. The measured covariance matrix is sent to the EKF rather than the default covariance matrix. This will allow the EKF to weigh the data from each sensor appropriately, resulting in better sensor fusion, and more accurate odometry.

The navigation and overall integration testing was done by giving the rover GPS navigation goals, and having to navigate around obstacles that we set in front of it, as well stay within lane lines that taped to the ground. This allowed us to fix any sim-to-real problems, most notably being tuning the navigation planner to work in reality and tuning our CV algorithm to work on the ZED camera rather than HD video data.

Mechanical testing focused primarily on drivetrain reliability, vibration reduction, sensor mount stability, and overall structural robustness during outdoor rover operation. Motor mount testing was conducted by driving the rover over asphalt, uneven terrain, and inclined surfaces to evaluate vibration and structural stability under dynamic loading generated by accelerating, decelerating and turning the rover. Repeated outdoor testing showed significantly reduced vibration transfer in visibly reduced oscillatory motion and no observable structural failures or fastener loosening during acceleration and terrain traversal.

Wheel coupler testing was performed to evaluate drivetrain slippage during acceleration, turning, and incline traversal. Alignment markings were placed on drivetrain interfaces to monitor relative movement between the motor shaft, coupler, and wheel hub. The updated coupler assembly showed no measurable displacement during repeated testing, indicating improved torque transfer reliability.

Sensor platform and camera mount testing focused on stability and field-of-view optimization during rover movement. The updated 3D-printed sensor mount demonstrated improved rigidity with minimal vibration-induced movement observed during outdoor testing. In addition, the adjusted camera mount angle improved forward environmental visibility while maintaining stable positioning throughout rover operation.

10. Cybersecurity Analysis

10.1 About the NIST RMF

The NIST RMF provides a structured approach that our team can use to mitigate cyber threats associated with the rover in competition [7]; such threats might undermine our team's ability to succeed and may have the potential to compromise intellectual property [8]. RMF's aspects include preparing the organization to implement the RMF, identifying, describing, and categorizing potential threats (based on factors such as severity), selecting appropriate risk management controls against them (and ensuring that team leads know about them), implementing the controls and assessing their efficacy, authorizing how the system should be used (if at all) given the risks, and monitoring the risk-control interaction over time [8] [9]. RMF emphasizes the analysis of threats and implementation of controls from a system level [9]; this is relevant to us, as many of our risk management strategies operate at the interfaces of both the embedded and software subsystems, and also include team-wide standards/practices.

10.2 Identifying Team-Specific Cyberthreats and Proposing Controls Against Them with the NIST RMF

One potential high-impact threat is the hacking of the onboard Raspberry Pi (RPi), which runs and contains essential ROS nodes. To categorize this threat's impact, our rover would likely stop functioning and our codebase could be observed by anyone. However, we have prepared against this threat by implementing Ethernet-based SSH connections as the only point of access to the RPi, and we may consider further controls such as password-protecting or encrypting sensitive files. As our team is interdisciplinary and tightly-knit, we can easily authorize any teammate to give the rover a test-run and ensure its functionality, to assess whether our controls have been effective. Another high-impact threat is the potential hacking of the team's laptop, which could not only compromise critical software but also result in privacy breaches. To reduce this risk, we implement private local hotspots to which the laptop can be connected and make regular backups of the codebase, allowing our teammates to wipe their laptops of malicious interference and restore their data. Furthermore, our team's use of GitHub as a platform for the rover's codebase allows our team to monitor whether any unauthorized code was pushed to our codebase and identify who may have done so. The final risk is RF- or magnet-based interference, which can be categorized as low-risk in the pit, but it could severely impact critical sensors like the IMU, Hall effect sensors, or GPS during a run. Since implementing physical shielding could disrupt sensor functionality, we should select and implement software-based monitoring

strategies, such as echoing relevant ROS2 topics and logging sensor behavior. These logs could be assessed with machine learning methods to detect anomalies indicative of cyber tampering.

References:

[1] “Cartographer Ros integration,” Cartographer ROS Integration - Cartographer ROS documentation, https://google-cartographer-ros.readthedocs.io/en/latest/. [Accessed May 13, 2025]

[2] “move_base,” ros.org, http://wiki.ros.org/move_base. [Accessed May 13, 2025]

[3] “costmap_2d,” ros.org, http://wiki.ros.org/costmap_2d. [Accessed May 13, 2025].

[4] "global_planner," ros.org, http://wiki.ros.org/global_planner. [Accessed May 13, 2025]

[5] “dwa_local_planner,” https://wiki.ros.org/dwa_local_planner. [Accessed May 13, 2025]

[6] T. Moore, "robot_localization 2.7.4 documentation," 2016. [Online]. Available: http://docs.ros.org/en/noetic/api/robot_localization/html/index.html. [Accessed May 13 2025].

[7] “NIST Cybersecurity Framework”, Wikipedia, n.d. [Online]. Available: https://en.wikipedia.org/wiki/NIST_Cybersecurity_Framework. [Accessed May 13 2025].

[8] National Institute of Standards and Technology, “NIST Risk Management Framework”, NIST Computer Security Resource Center (CSRC), March 11 2025. [Online]. Available: https://csrc.nist.gov/projects/risk-management/about-rmf. [Accessed May 13 2025].

[9] National Institute of Standards and Technology, “Risk Management Framework for Information Systems and Organizations” (Revision 2), NIST, n.d. Available: https://doi.org/10.6028/NIST.SP.800-37r2. [Accessed May 13 2025].