University of Toronto

UTRA Autonomous Rover Team (ART): Espresso

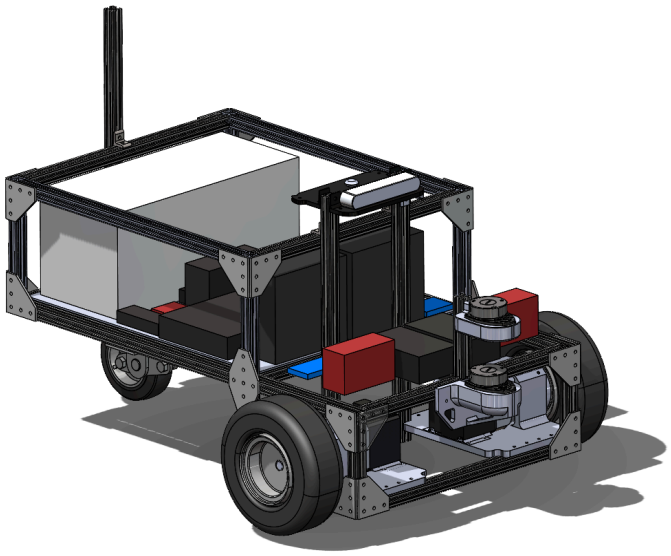

Figure 1: Cad model of Espresso

Date Submitted: May 15, 2025

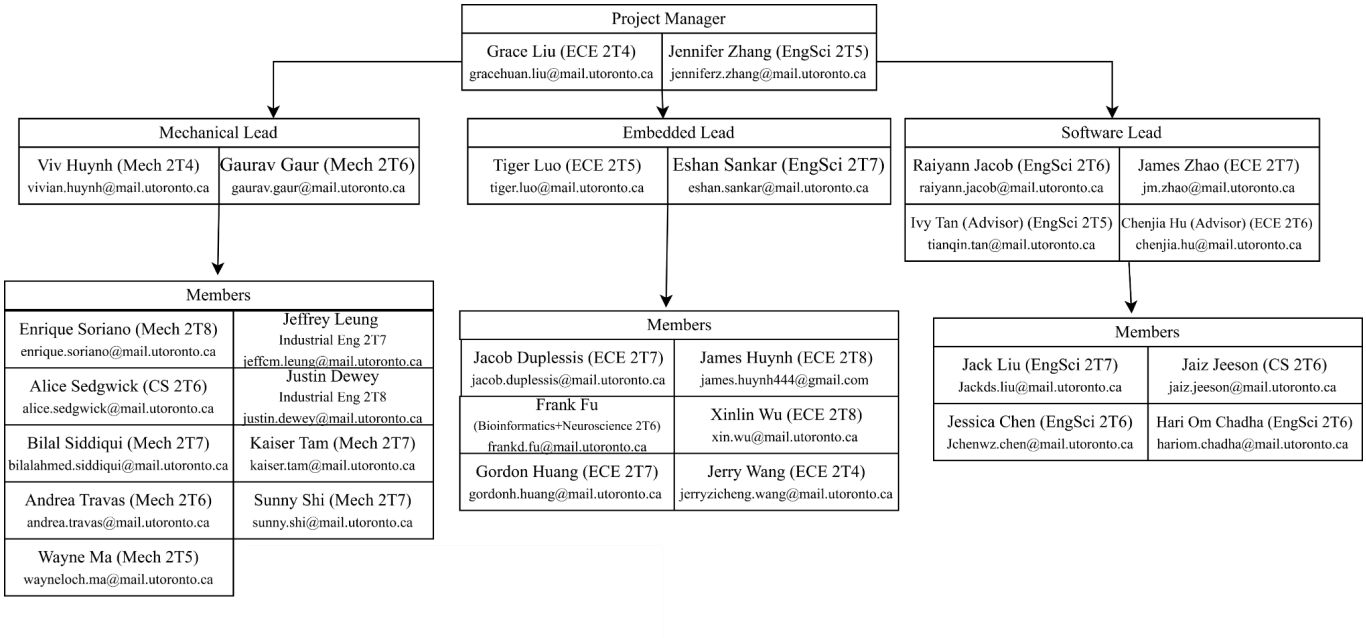

Figure 2. Team Leads and Members

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

Professor's Signature: _____

Prof. Sinisa Colic
Assistant Professor, Teaching Stream, Mechanical Engineering colic@mie.utoronto.ca
Tel: 416-978-5435

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

1.1 Introduction

The University of Toronto Robotics Association's (UTRA) Autonomous Rover Team (ART) was founded in 2008 by a group of students passionate about robotics. Since 2022, the team has consistently participated in the Intelligent Ground Vehicle Competition (IGVC). The team uses each year's results to reflect on and improve the rover's design. These insights are passed on to future team members to ensure continuous development. This year, building on the existing rover, *Espresso*, we introduced significant hardware and software upgrades to enhance its overall functionality.

1.2 Organization

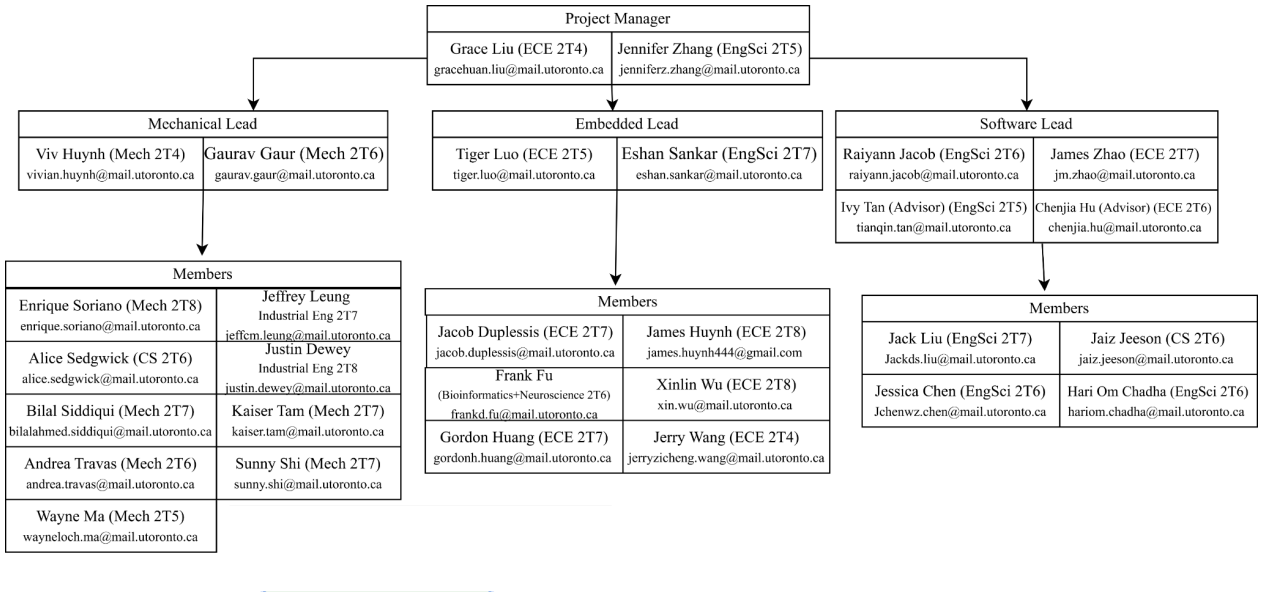

Figure 3. Team Lead and Members

The design team is organized into three subteams: Embedded (Active general members: 6), Mechanical (Active general members: 9), and Software (Active general members: 4). Prior to 2024, the Software team was split into Computer Vision and ROS subteams, which have since been merged into a singular software team to facilitate better collaboration. To streamline onboarding and ensure each subteam has a clear focus, multiple team leads are appointed. These leads work collaboratively to guide new members and provide each subteam with defined design goals to address in each work session. Project managers maintain regular communication with subteam leads and other UTRA executives to ensure timely task completion and to coordinate support whenever challenges arise. Members who contribute significantly to the direction and design of the rover, but aren't able to be appointed leads due to extenuating circumstances are designated as advisors. 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 those tasks are finished, 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 architecture of the rover

2.1.Significant Mechanical, Power, and Electronic Components

Electronic	Primary laptop: ASUS TUF F15 (i7-12650H, RTX 3070) Secondary laptop: Flexible, any ubuntu system with repository installed Microcontrollers: Arduino Uno, Arduino Due, ESP32, Raspberry Pi 3 Motor controllers: BLD-750 (x2) Lidar: Slamtec RPLIDAR A1 (x2) Camera: ZED 2i Stereo Camera IMU: PhidgetSpatial Precision 3/3/3 GPS: Columbus P-7 GNSS Receiver Network switch: TP-Link TL-SG105
Power	Batteries: 12V LiFePO4 (x3) Regulators: DC-DC Boost Converters rated 30A (x2) Relays: 30VDC 10A rated Motors: Motors: D5BLD300-24A-30S (x2) (300W DC)
Mechanical	Chassis: 1" T-slot 6061 aluminum and compatible fasteners Gearbox: 16:1 planetary gearbox (x2) Housing: 6mm thick acrylic panels

Table 1. Significant Components

2.2 Safety Device

Manual safety devices involve a manual emergency stop and a remotely controlled relay shutoff. Remote shut-off possesses a range of >20m, and manual shut-off is placed 3ft above the ground. Automatic safety devices include fuses, heat sensors, current sensors, voltage sensors, and an automatic microcontroller shutoff. The microcontroller

shutoff monitors current and voltage at the motor controllers, measures temperature within the rover enclosure, and the state of the ROS stack. If any unusual or extreme conditions are sensed or error messages are received from ROS, power is cut off to all systems, causing the rover to halt. Once the issue is resolved, the system can be restarted manually.

2.3 Significant software module

We leverage Robot Operating System (ROS) to facilitate sensor communication and rover behavior. Significant software components include using the robot_localization package to filter odometry sensor data from the Hall Effect Sensor, Wheel Encoders, IMU, visual odometry from a ZED stereo camera and GPS. Obstacle Detection relies on a 2D LiDAR for physical obstacles (i.e. barrels), while the lanes are tracked using the ZED stereo camera with the option of utilizing either classical or deep learning machine learning model. The classical approach utilized feature detection and image filtering techniques. The deep learning approach mainly uses the YOLOv8 object detection model.

Mapping is handled using the SLAM package 'cartographer', which takes in obstacle and odometry data and outputs a map and transform frames. Finally, our navigation is handled by the Navigation Stack, which uses the move_base package to interface with the path planning packages and generate trajectories and associated movement commands.

2.4 How these Items are Connected and Interact in the Vehicle

The primary laptop is responsible for running intensive Computer Vision (CV) models and autonav, while the secondary laptop runs mapping and sensor processing software. These two laptops then communicate over ethernet on the ROS server. ROS receives sensor data, which is provided to the secondary laptop either through direct usb communication with the sensors or through the Raspberry Pi. In turn, the Pi is connected to the motor controllers and reads feedback from the motors to ROS. The sensor and computer systems are able to interact and control the motor systems through optocouplers to prevent the power circuit damaging any electronics.

Upon receiving stable sensor inputs, the dual laptops carry out corresponding software tasks mentioned in the prior paragraph. Procedures regarding robot localization, mapping, and navigation in ROS are discussed in Section 2.3.

On the mechanical side, the chassis provides a platform on which all components can be securely positioned on the rover (using the 3d printed mounts, metal mounts, or velcro strap) and easily accessed when needed. The chassis also provides waterproofing and protection to the components with an acrylic and metal structure. During testing, assembly files of the designed chassis were used to simulate the performance of the Rover more accurately in Gazebo.

3. Effective innovations in your vehicle design

3.1 Detection of Ramp using Two LiDAR Units

Ramp detection is done via two LiDAR units to detect a difference in distance from the robot's point of view. The expectation is that a ramp is the only obstacle that will generate a large enough distance difference to be detected by our software, given that the LiDARs are at two different heights. Thus, the cost of an expensive 3D LiDAR unit can be avoided by using two cheaper 2D units. This is described in greater detail in section 6.2.4.

3.2 Ramp Navigation using Series of Waypoints

We developed an operating mode to traverse the ramp in a controlled manner. When the ramp is within visible range of the rover, the regular navigation is interrupted and enters ramp navigation. We ensure the rover moves straight by setting a series of waypoints spaced along the ramp, thus limiting the undesired path variability (i.e. big turns) that could force the rover to fall off. Discussed in more detail in section 6.6.

3.3 Electronic Subsystem Isolation

A large portion of electrical systems are isolated from one another to ensure that the failure of one does not mean the failure of all. This was designed for robustness and testability in mind. Electronics are designed in a way that can work in isolation from the others. For instance, the system controlling automatic shutoff does not require ROS, and the ROS system does not require the rover to be powered to test sensors and mapping. The isolation of these subsystems

includes electrically isolating the two major circuits, the power supply circuit and the control circuit, as the power supplies used for each are different.

3.4 Robust Fault Detection and Handling

To prepare for the testing phase of our rover, given that we overhauled our electrical and sensor subsystems (which could increase the risk of unintended behaviour such as overheating, undervoltage, or overcurrent), we developed a robust fault monitoring system. It reads from temperature, current, voltage, and alarm sensors, and shuts off power to the rover when a fault occurs. We used a real-time operating system, FreeRTOS, to ensure it operates quickly and consistently. This system minimizes the risk to both our rover and our team members in the event of a fault.

3.5 Deep Learning and Classical Model

To further improve inference speed and reduce computational power consumption, we quantized the model to INT8 format. This optimization reduced the inference time from 131.5 ms to 44.6 ms while maintaining a mean average precision of 95.2%. In addition to lowering resource usage, we observed that the inference callback outpaced the bandwidth capacity of ROS topics, causing a bottleneck. To address this, we reduced the camera frame rate to limit the number of images being transmitted.

3.6 Dual laptop system

A recent innovation has been the inclusion of a secondary laptop to split the computational load in half. Only a fraction of the software system requires the hardware on the primary laptop, specifically the Nvidia graphics cards used for ML models. The ML models are computationally expensive and slow down the ROS pipeline. However, with the introduction of a secondary laptop communicating with the primary over ethernet, the software systems can be sped up significantly by allowing the secondary laptop to run ROS nodes that don't require a graphics card. This system is also flexible enough to allow any computer containing the project's software to be used as the secondary laptop, so an older laptop was added to the rover at zero cost, allowing for budget conservation.

4. Mechanical Design

4.1 Overview

This year's rover was an upgrade of the previous rover, Espresso. Espresso's chassis consisted of 1-inch t-slot aluminum extrusions and acrylic panels, creating a rigid and lightweight structure. Reported from previous year's theoretical and practical testing results, the overall structural integrity of the frame was more than sufficient. However, the rover had significant issues in other aspects. Gaps in the acrylic panels allowed small but non-negligible amounts of water to seep in, potentially damaging the electronics. The motor mount also had a shearing failure during normal operation. Similarly, the mounting of external parts (LIDAR, e-stop, laptop, IMU, etc.) was unstable and inconvenient for the team to access and run tests on. Finally, during testing, it was noted that the width of the rover allowed for about one inch of clearance, if not less, through parts of the course. Therefore, the mechanical design focus this year was on improving the waterproofing, reinforcing mounting points, and reducing the overall width of the rover.

Housing

The housing consists of acrylic panels that fit into the gaps of the t-slot frame. Acrylic was chosen since it is lightweight, relatively cheap, and transparent to allow for ease of troubleshooting. Plywood was chosen as a bottom base piece since it is easier to modify/construct than acrylic, so the internal layout of the rover can be changed easily. The base is also non-conductive, reducing the chances of accidental shorts.

Structure Design

The major structure change this year was reducing the overall width of the rover by 4" in total. Reducing the width creates a larger amount of clearance when navigating the course. Such reduction is the maximum extent to which the width could be reduced while still maintaining a sufficient working area and designated payload area. Another significant structural change was replacing the 3D printed PETG motor mounts with a new welded steel design.

Furthermore, the worn-out aluminum motor couplers were replaced with new, carbon fibre reinforced nylon couplers. The previous couplers were slipping when the rover was running, causing precision and accuracy issues in the driving.

In addition, minor structural changes were made to improve functionality and ease of working. A laptop drawer was installed, allowing easy access and proper storage of the laptops. Moreover, the lidar, emergency stop, and IMU were mounted more rigidly, increasing the accuracy of sensor measurements and stability of the mounted components. Finally, the camera tower was shifted upwards to raise the camera height by 6", increasing the field of view.

4.2 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.

4.3 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.

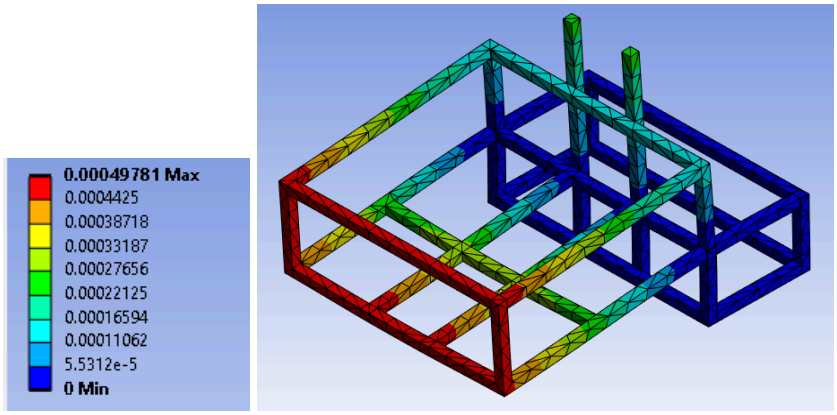

Figure 4 : Ansys static load simulation on the body of the rover.

4.4 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.

5. Electronic and Power Design

5.1 Overview

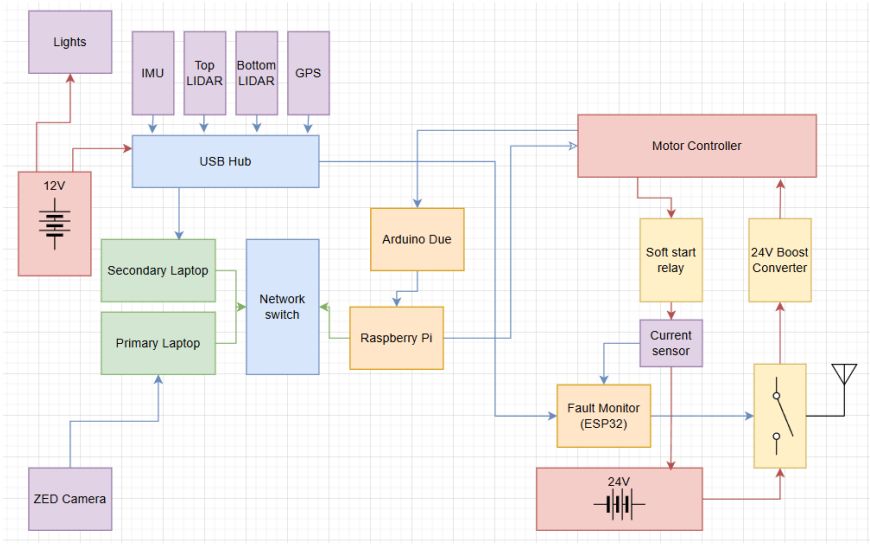

Figure 5: High-level overview of the electronics systems.

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. There are only two points of connection between the two circuits. The first is the RPi, which communicates with the motor controllers through optocouplers (achieving galvanic isolation). The second is the fault monitor subsystem, which only shares a ground connection to the power circuit

5.2 Description of the significant power and electronic components

All components are described in Section 2.1 under the 'Electronic' and 'Power' section of the table. Additionally, the power circuit also consists of six PC817 optocouplers to facilitate communication with the sensor/compute circuit, an ACS712 current sensor, and several LM60CIZ temperature sensors.

5.3 Power distribution system

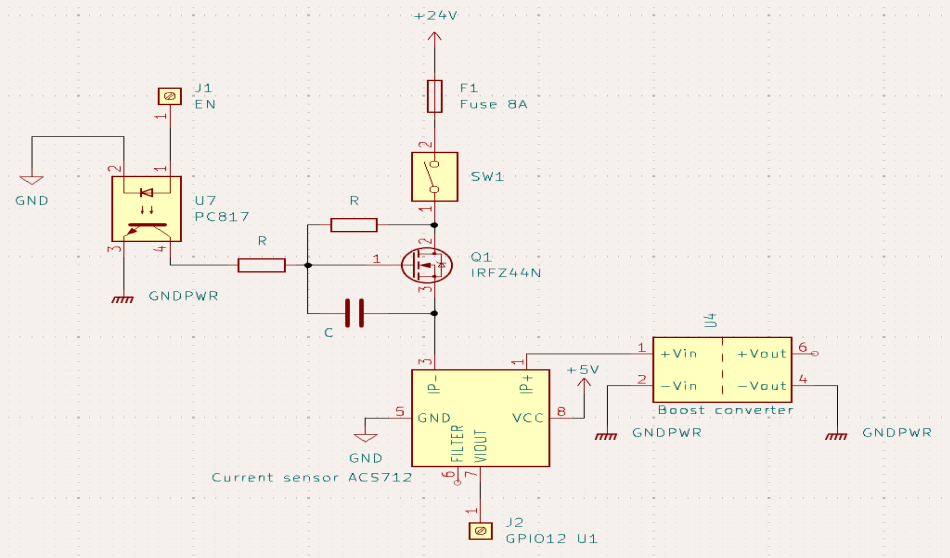

Figure 6: Motor drive circuit

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 5.5), a 24V boost converter, 8A fuse, and a fault monitor (see Section 5.4).

5.4 Electronics suite (computers, sensors, motor controllers)

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 the primary and secondary laptops, 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 two laptops and Raspberry Pi constitute the computing system on which all ROS nodes are run, with the three connected to each other via an ethernet network switch. 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 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. 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.

5.5 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 estop is an antenna relay with 20m of range. Both E-stops directly disconnect components from the 24V power via a hardware high-side switch.

6. Software System

6.1 Overview

The overall software system consists of four main components: odometry, obstacle detection, SLAM, and navigation (shown in Figure 5). The odometry module integrates data from sensors to provide both local and global positioning. For obstacle detection, we use LiDAR to identify 3D objects and a ZED camera to detect 2D features like potholes and lane markings. The outputs from the odometry and obstacle detection modules are passed to Cartographer for SLAM processing. Finally, the navigation module utilizes the SLAM results to generate a costmap and plan the optimal route.

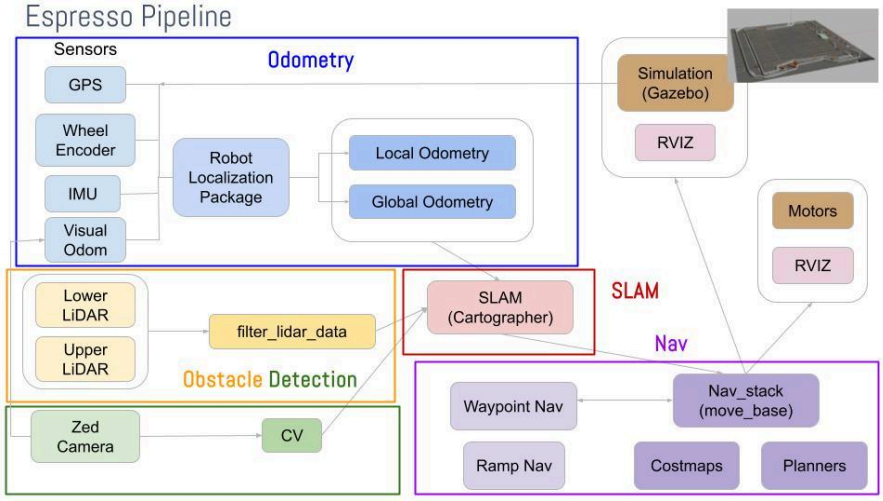

Figure 7: Software pipeline diagram

6.2 Object Detection and Avoidance

6.2.1 Lane, Pylon, and Pothole Detection

To encounter various situations, we implemented both classical and deep learning approaches for lane detection.

Our classical approach (Figure 6) mainly relied on feature detection and filtering techniques. First, the image is converted to grayscale and Gaussian blurred. Then, the Canny edge detector is used to extract the lane mark edges. The obtained edge map is further filtered by only picking the white contour to get an accurate lane detection mask. As a result, we reached a 25ms inference speed and near-perfect detection.

Our deep learning approach (Figure 7) mainly utilizes the YOLOv8 object detection model. A dataset of 1174 pictures from last year's competition track is manually labelled in the segmentation mask format. The results are more accurate near the camera than further away. To further increase the inference speed, we tried parameter pruning. Testing showed that pruning 5% of the parameters gives an inference speed of 9ms with roughly the same accuracy.

Figure 8: Lane detection result from our classical approach (blue lines)

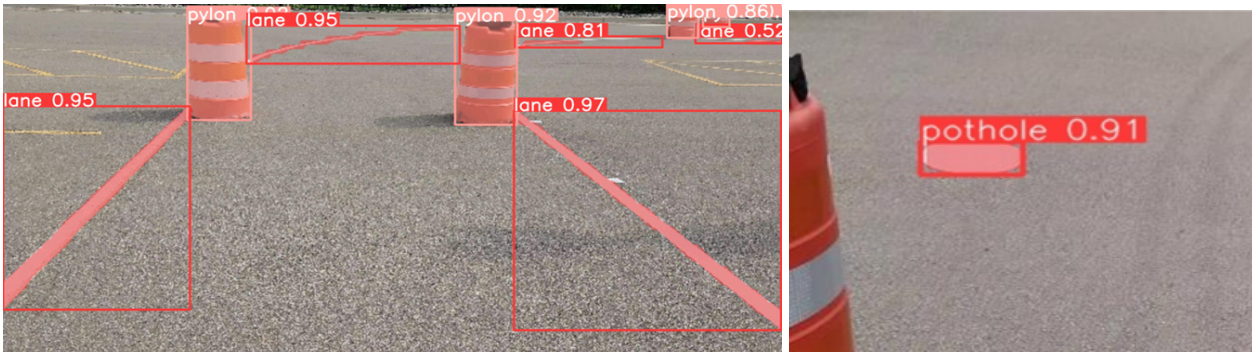

Figure 9: Lane, barrel, and pothole detection result from YOLOv8

6.2.2 Optimizing CPU load

We significantly improve the computer vision algorithms at the model and pipeline levels. At the model level, this consisted of implementing model pruning and quantization. We quantized the model weights to INT8, which allowed us to maintain very similar accuracy/confidence scores while reducing inference time by almost 3x the original rate.

The team further reduced the computer vision node's frame rate to 5 frames per second by implementing additional logic for frame skipping and queue size. Thus, high accuracy was kept while reducing the model's CPU usage by 60%.

6.2.3 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.

6.2.4 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.

6.3 Map Generation

The map is generated using the SLAM package ‘Cartographer’ [1], which provides loop-closure that aids in returning the rover to its starting position. Mapping works by successively building submaps (local SLAM) using sensor data and connecting them consistently with background threads for loop closure (global SLAM). Each submap is built using obstacle data obtained from obstacle detection (camera and LiDAR), with new submaps being placed according to odometry information.

To detect the ramp, we mount a second LiDAR above the first and measure the distance difference resulting from the incline of the ramp and publish a new LaserScan message that removes the ramp from our detection data so the rover does not avoid the ramp. We computed the optimal distance threshold to trigger filtering using the fact that the ramp does not exceed a certain incline and trial and error.

6.4 Goal Selection and Path Planning

To set goals, we enter the GPS coordinates into an ordered JSON file, with heading (North vs South, in case we must complete the course backwards) and laps to complete. Then, we send the transformed pose to a move_base goal. Move_base handles the path generation based on the costmap_2d, which reads directly from the Cartographer map (Figure 11). Afterwards, we query our GPS to see if we reached the point.

For path planning, we rely on 4 key packages: move_base [2], costmap_2d [3], global_planner[4], and dwa_local_planner [5]. Move_base 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. “Global_planner” and “dwa_local_planner” use this costmap to compute a global and local path that avoids obstacles (higher cost). 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 to use “navnf” or “base_local_planner” as a replacement for the global or local planner packages.

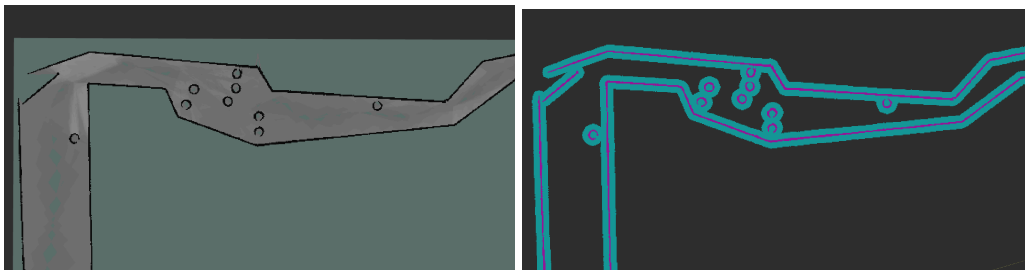

Figure 10: Map generated using Cartographer (left) vs costmap generated by costmap_2d (right)

6.5 Odometry and Localization

To track the robot’s current position, odometry is generated by fusing data from the Hall sensors, IMU, GPS data and/or visual odometry using the robot_localization package [6]. Visual odometry is generated using a stereo odometry node from the rtabmap_odom package, using ZED camera data [7]. The EKF nodes implement extended Kalman filter state estimation to output global and local odometry [6]. Before fusing GPS data, it is entered into the navsat_transform_node with an initial orientation and heading given by the IMU to generate a transform from UTM (Universal Transverse Mercator) coordinates into Cartesian [6].

6.6 Ramp Navigation Mode

Relying on the given waypoints would not be enough to ensure the planned trajectory results in a straight line across the ramp (and not dangerously near the edge where it could fall off) due factors like the inclination of the slope, we decided to implement a ramp navigation mode that interrupts the regular path planning to safely get across the ramp.

The dual LiDAR system described in Section 6.3 is used to detect proximity to a ramp and trigger an interrupt. The navigation program then plans a waypoint at the center and perpendicular to the ramp entrance. Once the rover is in place, a series of waypoints at 0.5-meter increments are placed along the length of the ramp, guiding the rover to traverse the ramp without veering off. Once the ramp is completed, the navigation program returns to the original list of GPS waypoints and continues on the course.

This strategy was to address the following problems: simply sending a ‘go straight’ command once at the ramp introduces a risk of running off if the rover becomes the slightest bit crooked; entering the ramp at an angle or close to the edge may result in rotational compensation on the ramp that may overshoot, since the wheels are at different heights; the small increments constrain the path trajectory to a straight line, and path deviations are less likely to occur.

6.7 How Vehicle Motion is Generated and Monitored from the Trajectory

The command velocity (cmd_vel) of the rover mainly comes from two sources - the navigation velocity (nav_vel) computed by move_base 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 ROS 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 5.4. 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.

7. Analyzing Cybersecurity for our Team with the NIST Risk Management Framework (RMF)

7.1 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 [8]; such threats might undermine our team's ability to succeed and may have the potential to compromise intellectual property [9]. 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 [9] [10]. RMF emphasizes the analysis of threats and implementation of controls from a system level [10]; 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.

7.2 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 ROS topics and logging sensor behavior. These logs could be *assessed* with machine learning methods to detect anomalies indicative of cyber tampering.

8. Analysis of Complete Vehicle

8.1 Lessons Learned

On the software side, the team had embarked on multiple bold endeavours to improve on existing features. These attempts include: using the ZED camera and ramp detection algorithms, such as the RANSAC algorithm, to detect the ramp, instead of the dual-lidar system, and developing a customized reinforcement learning ROS package for global and local path planning. However, due to insufficient knowledge accumulation in respective fields, these attempts are still under development. Luckily, our existing software system is robust and acts as our failsafe, though these endeavours are not realized. The main lesson the team had learned from this is: to always develop a minimal viable product before developing more interesting and challenging add-on features.

The embedded subteam began this year's design cycle on top of the foundational embedded system that had been created during the previous design cycle, which was functioning but had significant room for improvement. Learning from software-related issues during previous competitions, the embedded subteam was more deliberate in their

endeavours. The subteam leads educated new members on rover design and future project ideas based on the old, barebones design, and conducted rigorous testing (e.g. teleoperation, monitoring the CPU load of ROS nodes, etc.) on the old rover. The results from these tests helped justify the ambitious projects (designing and implementing a motor control PCB and a revamped fault monitor, etc.). Furthermore, the barebones embedded components acted as a fallback plan in case the new designs were not performing as intended, so that other subteams could test the rover without the embedded subteam causing deadlock. Overall, the embedded subteam's approach enabled the successful implementation of large-scale changes and various quality-of-life fixes for the rover.

On the mechanical side, the team finished a design cycle that included a full mechanical redesign. While most of it went smoothly, there were some points of failure. At the 2024 competition, issues surfaced regarding the efficacy of the waterproofing and the failure of the motor mounts. With the knowledge and experience of the leads, the new systems have been manufactured with better tolerances and more stringent failure criteria. The main lessons learned were the importance of testing and validation, and how to better optimize the time management of a design cycle.

8.2 Top Hardware Failures

8.2.1 Embedded

Failure Components	How will they be addressed if they occur during competition
Rover/laptop(s) run out of battery during a trial.	By being proactive and charging the batteries and laptops when in the pit, the likelihood of this will be minimized.
Rover draws too much current (which could result in overheating or undervoltage).	We can prevent these situations by having the rover adhere to speed limits (set in software) and ensure that the mechanical subsystem is well-lubricated, so that the motors do not exert more effort than necessary. If any of these issues occur, the fault monitor would immediately notice and shut off the rover to avoid further damage.
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 2. Potential Embedded Hardware Failures and how to address them

8.2.2 Mechanical

Failure Components	How will they be addressed if they occur during competition
3D printed motor mount shears due to mechanical failure	Motor mount has been redesigned, but spares will be brought along and be used to replace broken components if need be
Waterproofing failure	Rover will be constantly monitored for moisture inside the upper chassis. It will be cleaned and dried, and waterproofing will be enhanced with temporary measures before going back out.
Vibration causes the wheel to loosen and wobble during trial runs	The system will be monitored after every trial run, and the wheels will be tightened as needed

Table 3. Potential Mechanical Hardware Failure and how to address them

8.3 Safety, Reliability, and Durability

Software needs to ensure the reliability of odometry. The main breakthroughs with our Odometry & Localization system this year have come through the form of validating, verifying, and improving results from our sensor fusion.

While odometry from wheel encoders, ZED Camera, and GPS was quite reliable (concluded after trials of testing), however, odometry from our IMU was found to be noisy and had significant drift over time. A solution we implemented was a signal processing filter, a Butterworth Filter, which helped smooth out our IMU data, guaranteeing that clean IMU inputs are being fused in our odometry.

Our sensor fusion is Extended Kalman Filter-based (EKF) using the robot_localization package. Each sensor has 15 different state variables that can be fused in the EKF. Our team had tested how different combinations of each sensor's state variables contribute to the final fused result from the EKF. Evaluating the result from testing the rover's maneuver in real life, we validated the most accurate combination. Such a combination is used as our final odometry input for robot localization. For example, using both position and linear velocity from the wheel encoder, instead of just the linear velocity, leads to better results in the long run.

8.4 Analysis of Predicted Performance

Speed: The rover's speed is limited in software, in the global and local planner packages' parameters, to stay within the competition restriction of not surpassing 5 miles per hour.

Ramp Climbing Ability: The 300W motors provide enough torque for the rover to climb ramps of 15º inclines during testing, though doing so causes a higher current draw and consumes battery life quite fast. However, as we generally run the rover at slow, gradual speeds, the batteries can definitely provide the necessary power to ascend any ramps.

Reaction times: The rover's predicted reaction time (tested in both simulation and real life) is around 2-3 seconds when it recognizes an obstacle and begins to navigate around it.

GPS usage: GPS is fused with our global odometry; we have tested its accuracy separately and the fused global odometry's accuracy in real life. Both are reliable. Discussion on how it is used is in Sections 6.4 and 6.5.

Battery Life: Batteries are capable of lasting around 1.3-3.2 hours when the rover operates at normal speeds and experiences an average current draw of 2.5A. Battery lifespan is decreased when the rover runs faster or climbs ramps. Batteries take approximately 1.3 hours to recharge.

Distance at which obstacles are detected: The rover is able to identify the obstacles and include them in mapping as soon as they have entered the detectable range of CV or Lidar.

How vehicles deal with complex obstacles: Our rover is designed to compete in auto-navigation and will only face the ramp as a complex obstacle. The algorithm developed to deal with this is discussed in section 6.6. Its performance is steady and solid in simulation; real-life testing is still undergoing.

Addressing vehicle failure points and modes: The team identifies and addresses failure points through a combination of proactive monitoring, iterative testing, and modular design. Our robust fault monitor continuously monitors and outputs the key metrics of temperature, voltage, and current, using an ESP32 microcontroller with FreeRTOS to ensure quick and consistent response times. If anomalies are detected, the system automatically triggers a hardware shutdown to prevent further damage. On the mechanical side, known weak points from previous iterations (shearing of motor mounts and waterproofing issues) were redesigned with improved materials and tolerances. Extra critical components have been made available as replacements during the competition, should any failures occur. Additionally, virtual and real-world testing help uncover vulnerabilities early. This systematic approach allows the team to mitigate risks and enhance reliability across both hardware and software systems.

Accuracy of arriving goals in navigation: Goal tolerance parameters have been configured in the software planner to allow the rover to recognize nearby points as acceptable substitutes for the exact goal location. This prevents the rover from attempting to reach an unattainable precise position due to physical limitations in its structure. By accepting a small range of positional tolerance, the rover can complete its navigation tasks more efficiently and reliably.

Comparison of predictions with actual testing: The rover's mechanical, embedded, and software performance are predicted to be very good. This year's redesign of the chassis led to significant improvements in mitigating vibrations and mechanical fatigue, with the rover being capable of completing the course. The embedded subteam has verified subsystem functionality via unit testing on subsystems such as the e-stops and fault monitor. Furthermore, throughout the development cycle, the embedded subteam has verified functionality independently from other subteams, through the use of digital multimeters, oscilloscopes, and debugging via the microcontrollers' serial interfaces. Software's

simulation results match the real-life testing. The team is currently engaged in integration-level debugging on the physical rover, where real-world testing is revealing unexpected issues not encountered in the separated testing.

8.5 Software Testing, Tracking, and Version Control

Git, via Github, is our platform for version control, enabling collaborative and organized software development. GitHub Issues is used to create a central location for tracking and discussion when a problem or bug is found. To work on the solution independently and maintain the stability of the main codebase while having the freedom to test as needed, developers then build a dedicated branch for that particular issue. After the branch's issue has been fixed, a pull request is made, which starts a code review session where peers can assess the changes for accuracy, quality, and style. To preserve a traceable development history, the branch is merged into the main branch upon acceptance.

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.

8.6 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. As the main physics-based simulator, Gazebo enabled us to accurately mimic surroundings, sensor data, and robot dynamics, allowing us to conduct realistic tests of our robot's movement, interactions, and obstacle avoidance. By offering a potent visualization interface for the robot's perception and planning processes, RViz enhanced Gazebo. Debugging and fine-tuning the robot's behaviour during development was made much easier by RViz's ability to monitor sensor outputs in real-time.

8.7 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 2 times in different locations and under different lighting conditions. Full tests of the entire pipeline are scheduled for the next two weeks.

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. Tests described in section 8.5 are then conducted. 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.

The rover is also tested in auto-navigation mode, leading to the discovery of problems with the lidars picking up phantom obstacles and the rover boxing itself with inerasable costmaps. The former led us to construct a stiffer lidar mount, and the latter is still being debugged.

To ensure the mechanical robustness of the rover, extensive physical testing was conducted on several critical components. The redesigned motor mounts underwent load testing to verify their ability to withstand operational stresses without experiencing shearing failures. This involved subjecting the mounts to forces mimicking real-world driving conditions to validate their structural integrity. In parallel, the rover was operated on asphalt surfaces to simulate competition terrain and assess the effectiveness of its anti-vibration measures, such as low-pressure rubber

tires and suspension in the rear caster wheel. Additionally, the waterproofing subsystems were practically tested by exposing the rover to wet environments and monitoring for internal moisture intrusion.

9.0 Unique Design for AutoNav

9.1 Ramp Detection

The design specifically for the AutoNav challenge is our dual lidar system, which is discussed in detail in Section 6.6.

10. Initial Performance Assessments

10.1 Lane and Pothole Detection

Our algorithm is mainly tested using videos of the track taken from previous years. The GPU used is Nvidia T4 in Google Colab. All detection accuracy mentioned is measuring the IOU of the detection and ground truth. In terms of our classical detection approach, we reach a near 100% detection accuracy in terms of IOU and an inference speed of 25ms. For our deep learning algorithm with INT8 quantization, the model reached 94.4% and 96.1% detection accuracy for lane and pothole, respectively. The average inference speed for running both lane and pylon detection is 44.6ms. Lane detection was tested physically to validate the algorithm, with results as mentioned in section 7.5.1.

10.2 Software Testing

Testing of mapping and navigation has primarily been in simulation (mentioned in section 8.3) and has performed satisfactorily. Map generation of obstacles detected using the LiDAR is consistently accurate to the Gazebo world, with delays of <1s between initial detection via sensor and its placement on the map.

Both classical and deep learning models of CV are integrated with ROS: the rover treats the detected lane lines as obstacles and forms corresponding costmaps around them. While the lanes appear on the map in clear lines, they are not perfectly incident to the real lines.

The goal-setting system in Section 6.4 is able to accurately place a goal based on GPS input. The path planning also satisfactorily avoids all obstacles, and with time, the rover will reach the goal. The completion time is currently undesirable and will be improved in the future. Finally, tests with the ramp navigation mode have successfully allowed the rover to traverse the ramp approximately 85% of the time in simulation, with the rover remaining within 10° of the length of the ramp. This comes at the cost of additional time taken, and further optimization may be made.

10.3 Mechanical testing

The load tests of the motor mount revealed that shear failure, similar to what was experienced last year, is not a significant concern. With the previous design being 3D printed, the layer lines concentrated shear forces causing a failure in the design. With the new steel design, shearing due to layer lines is no longer an issue. In addition, vibration tests confirmed that levels were within acceptable limits. Enhancements such as thicker acrylic panels, rubber gaskets, silicone caulking, and an external waterproof covering (poncho) were validated through these trials, ensuring the electronics remained dry and secure under damp conditions.

10.4 Embedded Testing

Speed and run tests showed the motors were capable of accelerating up to 1.2m/s before batteries were no longer able to provide the required power and shut off automatically. At this point, over 8A were being drawn from the batteries, thus hardware limiters were set to 6A in the fault monitor to prevent overdraw of batteries. During high speed tests, the rover was capable of running for around 4 minutes at maximum speed before automatic shutdown was activated due to high temperatures. Odometry tests for wheel encoders showed at minimum 5% drift in position tracking. Computer load and speed tests showed the primary laptop using over 80% of CPU resources on average, but was able to maintain a real-time to ROS-time of one-to-one.

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] M. Labbe, "rtabmap_odom," Open Robotics, April 19 2023. [Online]. Available: http://wiki.ros.org/rtabmap_odom#stereo_odometry. [Accessed May 13 2025].

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

[9] 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].

[10] 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].