Intelligent Ground Vehicle Competition 2025

# Manipal Academy of Higher Education

## Project MANAS

# Design Report

**STEVE**

**I certify that the development of the vehicle, *STEVE*, described in this report, has been equivalent to the work in a senior design course. The students of *Project MANAS* have prepared this report under my guidance.**


**Ashalatha Nayak**
Professor
Department of Computer Science & Engineering

Date: 15.05.2025

Team Members

| Artificial Intelligence | Sensing and Automation | Mechanical |
|---|---|---|
| Asmit Paul | Ashwin Modey | Aadya Jha |
| Janak Shah | Davasam Kartikeya | Aditi Jain |
| Mrigaank Mouli | Kaustubh Pandey | Anvith Jasti |
| Om Kumar | Kartikeya Singh | Farshad Lavangia |
| Tanmay Singh Bains | Sachin Sushil (Team Captain) | Mahua Singh |
| Vivek Singh | Sudeep Sharma | Mohit Nambiar |

# 1. TEAM ORGANIZATION & DESIGN PROCESS

## 1.1. *Introduction*

*Project MANAS*, the official AI and robotics team of *Manipal Academy of Higher Education (MAHE)*, proudly presents *STEVE*, our entry to the *Intelligent Ground Vehicle Competition (IGVC)*, 2025. We spent the last year developing a new autonomous platform, based on the lessons and knowledge we gained from our previous iterations. Each member of our interdisciplinary team contributed greatly to ensure that we create something truly novel. *STEVE* is the culmination of these efforts, representing a holistic upgrade to our UGV.

## 1.2. *Organization*

*Project MANAS* comprises 42 undergraduate members pursuing various fields of engineering. The team is organized into four subsystems:

- *Artificial Intelligence* - Develops the software stack of the UGV.
- *Sensing and Automation* - Responsible for electronics and power distribution, actuator control, and sensor suites.
- *Mechanical* - Designs and manufactures mechanical components of the UGV.
- *Management* - Oversees finances, logistics, PR, and sponsor outreach of the team.

The team is managed by a group of senior Board members, who comprise subsystem heads, the Team Manager, the Technical Head, and the R&D Head. Senior members of the team guide and supervise its operation. Through a system of regular meetings, and shared resources, the team is able to collaborate with each other effectively.

## 1.3. *Design Process & Assumptions*

To effectively manage and coordinate tasks leading up to the development of *STEVE*, an approach adapted from the *Integrated Product and Process Development (IPPD)* lifecycle was adopted. This approach was well-suited to the cross-disciplinary nature of the team, allowing for the concurrent development of individual components of the UGV.

Following extensive research, the design process of the UGV was divided into subsystems, with the design individualized into isolated components. Using this approach, the development of the IGVC 2025 bot was structured into parallel design and process planning tracks. This initial design was refined through continuous iterations and effective team communication, ultimately resulting in detailed designs. Once each component was completed, it was integrated into a complete prototype.

The design and implementation were carried out over a span of 8 months, with each team member averaging approximately 560 hours of work. A total expenditure of 7500 USD was incurred for the acquisition, maintenance, and replacement of components.

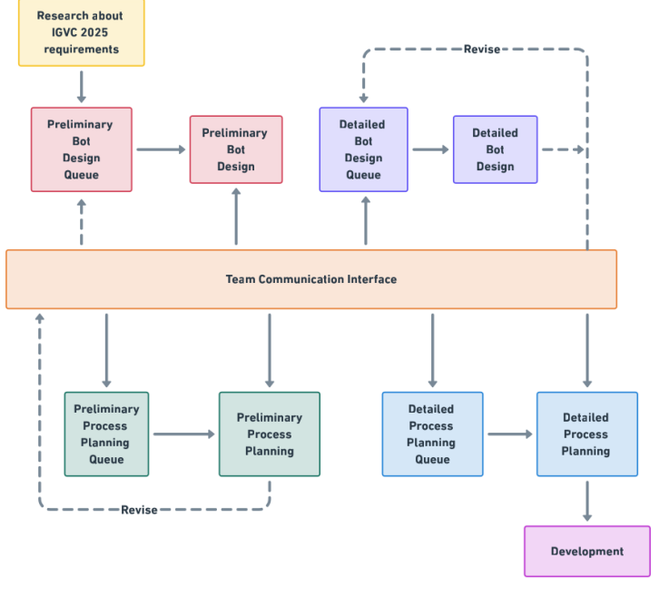



**Figure 1.** Design Process

# 2. SYSTEM ARCHITECTURE

## 2.1. *Significant Mechanical, Power, & Electronic Components*

*STEVE*'s mechanical and electronics suite has been selected based on simulations and analysis to ensure that all components perform adequately in their operational range. All components have a significant margin of safety, which ensures reliable bot operation.

| Category | Component | Model |
|---|---|---|
| 1. Processors | Computer | Asus TUF A15 |
| | Communication Module | STM32F401CCU6 |
| | Drive Controller | STM32G491RET6 |
| 2. Peripherals | E-Stop Radio | nRF24L01 Transceiver Module |
| | Status Lights | Adafruit Neopixel Strip |
| 3. Sensors | LiDAR | Ouster OS1 |
| | Stereo Camera | Stereolabs ZED 2i |
| | GPS | Hiwonder GPS unit |
| | IMU | Hiwonder 10-axis IMU |
| 4. Power | Battery | 6S 30,000 mAh Tattu Battery |
| | Inverter | RS PRO 400W Inverter |
| 5. Motors | BDC Motors | Rhino 24V 60RPM IG52 Motor |

**Table 1.** Significant Components

### 2.2. ***Safety Features***

*STEVE* incorporates multiple safety mechanisms to ensure adequate operation of the bot within the competition environment.

The bot includes a comprehensive E-Stop system that can be triggered mechanically or through a remote switch. Additionally, *STEVE* incorporates a safety indicator light mounted on the chassis of the bot. The safety light indicates the mode of operation of the bot. A solid light indicates the bot is in manual mode, and a blinking light indicates that the bot is operating autonomously.

In addition to these competition-specified safety features, we have included our own set of custom safety features to ensure smooth operation of the bot.

- If the Driver Controller detects abnormal operation of the motors for an extended period of time, it will halt the operation of the motors. Additionally, any loss of communication will also trigger this mechanism.
- The custom-built Power Distribution Board (PDB) features 30A fuses, which in the event of overcurrent draw will automatically shut down power to all critical components.
- Current and voltage sense resistors on the PDB also allow the power state of the bot to be monitored, shutting down power in case abnormal spikes are detected.

### 2.3. ***Significant Software Modules***

Our software stack has been divided into various modules, based on their functionality and purpose:

- **Perception -** Responsible for detection of lanes, potholes and obstacles for the Auto-Nav challenge, and detection of pedestrians. The module is a mix of Deep Learning and traditional algorithmic approaches to obstacle detection.
- **Localization -** Fusing odometry from multiple sources(LiDAR, wheel encoders) to accurately estimate the robot's pose, while ensuring it is immune to noise.
- **Mapping -** A dynamic representation of the bot's surroundings as an occupancy grid for effective path planning.
- **Goal Selection -** Determines intermediate navigation goals based on the current map and dynamically adapts the selection approach based on the bot's location and the obstacles detected. A control node manages the switching between different strategies.
- **Navigation -** Utilizes motion and path planning techniques to compute the most efficient trajectory to the target, using the selected goals as references.

### 2.4. ***Integration of UGV subsystems***

The UGV system begins with sensors that gather environmental data, connected directly to a laptop running dedicated software to collect, filter, and publish this data to ROS topics. Based on the sensor input, a map is generated, and the Goal Generator analyzes this map to continuously publish new goal positions. The Planner then calculates a path to these goals and sends movement commands (`cmd_vel`). These commands are converted into RPM values using the robot's physical parameters and sent to the control units via USB. The microcontroller translates these RPM values into PWM signals to control the motor drivers, which in turn power the motors. It also reads encoder feedback and uses a PID controller to ensure the motors maintain the desired speed accurately.

## 3. EFFECTIVE INNOVATIONS

### 3.1. ***Mechanical***

#### 3.1.1. *Caster Suspension Linkage*

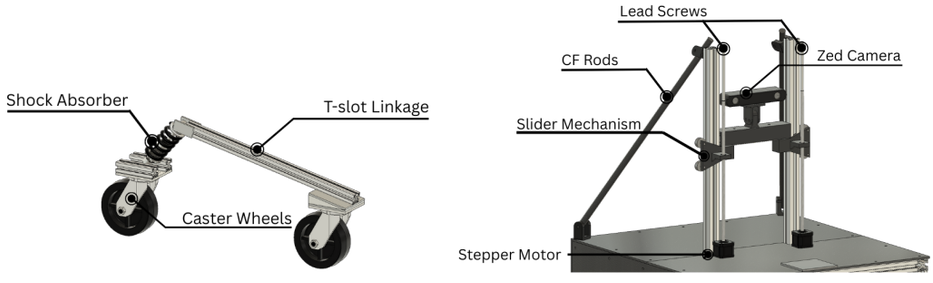
One of the major shortcomings of our previous bot, *NOVA*, was its inability to effectively climb ramps, which significantly limited its operational versatility. This year, the issue was addressed head-on by implementing a new suspension system with a

TPU-based vibration damping mechanism and a front-rear caster linkage. This upgrade has greatly enhanced the bot's ability to traverse uneven terrain and tackle inclined surfaces with stability and control, effectively overcoming one of the most critical limitations of the earlier design.

A T-slot extends across the chassis, linking the caster wheels with a shock absorber at the rear caster. The T-slot running down the middle of the frame is supported through a pivot mechanism, allowing it to rotate. This mechanism provides for increased travel for the front caster when it is met with uneven terrain, with the linkage forcing the casters to maintain contact with the ground at all times. On the other hand, we have worked on making electronics and other hardware mounts out of TPU so as to isolate them from any impulses or high-frequency vibrations.

#### 3.1.2. *Dynamic ZED-Camera Movement System*

With the team's participation in both AutoNav and Self-Drive this year, the field of view needed by the Software Stack is different based on the conditions that the bot will be facing. One of the major reasons for designing a mechanism that allows the ZED-Camera to change its height based on its surroundings is to provide the most optimal field of view (FOV) for mapping its environment. This feature in turn allows for increased performance in our runs, providing an optimal viewing angle at all times. The extending mechanism utilizes lead screws and a stepper motor, allowing for smooth and precise control over the camera position, resulting in a total of 350 mm of travel.




(a) Caster Suspension (b) ZED Mount

**Figure 2.** Mechanical Innovations

#### 3.1.3. *Modular External Paneling*

Carbon fibre was chosen as the material for the external paneling for our bot due to its high strength-to-weight ratio, stiffness, structural rigidity, and resistance to corrosion. This decision allowed us to cut down 10kg of weight and increase its performance in adverse weather conditions.

Magnetic Couplings are used for the panels present at the most accessed regions, such as the payload bay, electronics compartment, and laptop compartment, allowing for easy access whenever required.

### 3.2. ***Electronics***

#### 3.2.1. *Modular Control Architecture*

This year, in order to align with our goal of a modular design, the hardware control functions have been split into two units, i.e, the Drive Controller and the Communication Module. With this split comes improvements in our communication methods, namely, the availability of CAN-FD and RS-485 buses on board. CAN-FD in particular is a notable solution in the automotive industry, owing to its high reliability, speed, and ease of application. These incorporations are significant in improving our bot's architecture to allow for more complex controller setups.

#### 3.2.2. *Static Code Analysis of Firmware*

All the controller firmware has been written from the ground up using the Rust language and open-source toolkits. Built-in features of the Rust language allow static code checks to be performed before the firmware is deployed, ensuring that no memory-misuse bugs occur and allowing for high-performance and reliable behavior.

### 3.3. ***Software***

#### 3.3.1. *Quadtree-based Submap Management*

Storing multiple revisions of the global map introduces significant overhead, especially in memory-constrained systems or when operating in real-time. To address this, a Quadtree-based map representation is employed. The hierarchical structure of Quadtrees enables efficient spatial partitioning, allowing regions with uniform occupancy to be represented compactly while preserving higher resolution in areas with complex features. This reduces both the memory footprint and inter-node communication overhead, making it well-suited for scalable multi-resolution mapping in robotic systems.

#### 3.3.2. *Filtered Depth Image Mapping with Polygonal FOV Estimation*

Instead of processing all points, a mask is applied to the RGB image to extract only relevant ones. Using corner points from the corresponding masked depth image, a polygonal boundary is defined on the occupancy grid. Obstacle points within this region are then iterated over, avoiding full-grid traversal.

#### 3.3.3. *Map Slicing*

To manage time efficiency of algorithms, occupancy grids of the environment are sliced to a local area surrounding the bot. In addition, the maps are further revised to extract features such as nearest lanes and fill in gaps via interpolation. Multiple maps are combined with each other to give modules the relevant information.

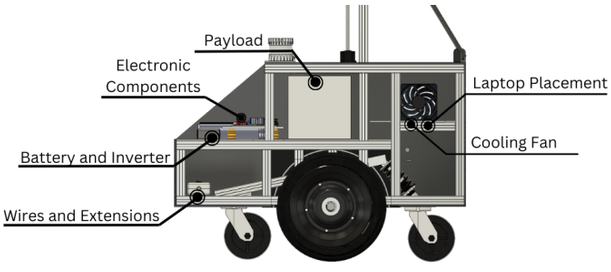
## 4. DESCRIPTION OF MECHANICAL DESIGN

### 4.1. ***Design Overview***

The mechanical subsystem forms the backbone of *STEVE*'s platform, enabling robust structural support for all other subsystems. The design and analysis workflows are performed in Fusion 360 and Ansys to simulate stress, vibration, and thermal loads. Components are fabricated using a combination of CNC waterjet cutting, radial drilling, FDM 3D printing, and a Mitre saw. *STEVE*'s compact footprint and calculated torque outputs ensure effective navigation of the competition obstacle course.

### 4.2. ***Structure***

#### 4.2.1. *Chassis*




(a) Chassis Design

| Parameter | Value |
|---|---|
| Mass | 65 kg |
| Length | 3.5 ft |
| Width | 3 ft |
| Height | 3.9 ft |
| Max Torque | 16 N-m |

(b) Chassis Parameters

**Figure 3.** Overview of the chassis.

*STEVE*'s frame is constructed using 30mm×30mm aluminum T-slot profiles, known for their modularity, strength, and lightweight nature. The use of aluminum corner brackets ensures high rigidity while maintaining design adaptability. The T-slot arrangement has been redesigned to centralize weight distribution and accommodate additional modules like the extending ZED-Camera mount and suspension linkage.

#### 4.2.2. *Compact Footprint*

*STEVE*'s compact 3ft×2ft footprint optimizes maneuverability, enabling efficient navigation in constrained environments such as warehouses and healthcare facilities. The reduced form factor minimizes turning radius and facilitates more predictable dynamics, thereby improving sensor calibration accuracy and enhancing the performance of localization algorithms.

#### 4.2.3. *Vibration Control*

The updated chassis now incorporates a reinforced carbon fiber mount composed of two slanting 3K 200GSM 2×2 twill rods for the ZED-Camera. This reinforced mounting architecture increases overall stiffness, reduces structural resonance, and improves image stability. The angled support geometry offers enhanced vibration damping, particularly during acceleration and directional changes, leading to more stable imaging by the ZED-Camera.

### 4.3. ***Drivetrain***

*STEVE* employs a compact drivetrain configuration with centrally located drive wheels and caster wheels at the front and rear for enhanced stability. The motor shaft is supported by dual bearings and connected to the wheels via M8 bolts arranged in orthogonal planes, ensuring force distribution across two shear surfaces. The central axis layout allows precise, predictable turns by eliminating body swing, reducing navigation errors, and improving sensor accuracy through minimized inertial disturbance.

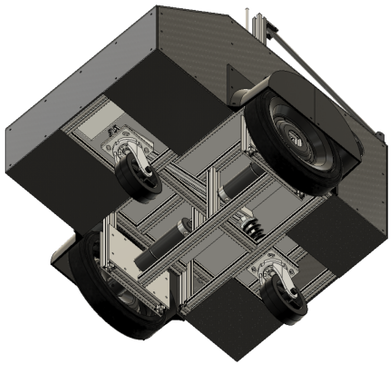


**Figure 4.** Drive Train

### 4.4. ***Suspension***

Given the relatively smooth asphalt surface of the course, a full fledged active suspension system was initially deemed unnecessary. However, test runs revealed intermittent loss of traction, particularly on slopes and during transitions between level surfaces and ramps. To mitigate this, we introduced a passive mechanical compensation system using a central pivot and shock absorber assembly that balances load distribution between the front and rear caster wheels. This setup ensures consistent ground

contact by passively adapting to minor elevation changes, improving drivetrain efficiency and preserving directional stability. Additionally, the integrated dampeners help reduce structural vibrations, contributing to smoother motion and protecting sensitive onboard electronics.

### 4.5. *Weatherproofing*

This year, in order to expand the conditions in which the bot can perform, gaskets have been utilized throughout the chassis and body panels to effectively seal critical components from external elements such as dust and water. Additionally, wheel coverings were also incorporated to ensure that the motor and drivetrain remain protected during operation. With these enhancements, the bot now meets the IP55 protection standard, providing reliable resistance against dust ingress and low-pressure water jets – making it well-suited for a variety of outdoor environments.

**Figure 5.** Wheel Hub

## 5. DESCRIPTION OF ELECTRONICS & POWER DESIGN

### 5.1. *Overview*

To achieve the goal of future-proofing and extensibility, the electronics architecture of *STEVE* has been designed to be as modular as possible. This was a drawback we encountered with our previous iteration, *NOVA*, where the system was highly specialized and features could not be added easily. We achieve this flexibility by isolating essential system functions. The responsibility of motor control is delegated to the *Drive Controller* Unit. All other sensors and radio communication are handled by the *Communication Module*.

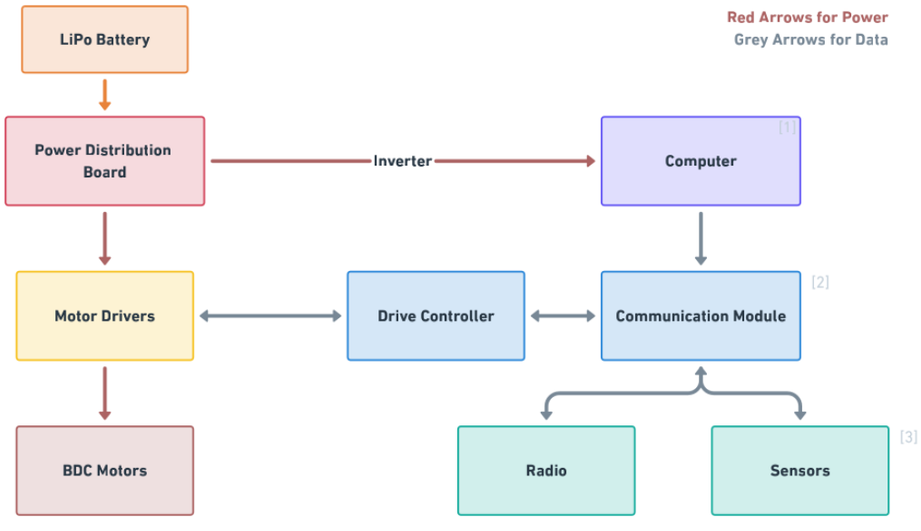



**Figure 6.** Overall Electronics Architecture

### 5.2. *Power Architecture*

The system's DC power is supplied by a 6S lithium-polymer battery rated at 30,000 mAh. The DC power is converted into AC via a commercially available 400 W inverter converting 24 V DC into 230 V AC. The total current draw is monitored via a Shunt Resistor and a Current Sense Amplifier that sends an analog signal to the ADC of the Communication Module.

Based on conservative estimates, the setup consumes roughly 500 W under typical loads and up to 700 W at full capacity – equivalent to currents of 22 A and 32 A at 24 V, respectively. The power to the BDC motors is routed through a Solid-State Relay (SSR), which facilitates an immediate power isolation through the E-stop in case of an emergency.

On the power distribution board, a 24 V to 5 V buck converter provides constant 5 V supply. A 30 A automotive fuse is used to safeguards the Cytron drivers in case of a sudden power surge —automotive fuses being both fast-acting and simple to replace.

### 5.3. *Sensor & Control Units*

#### 5.3.1. *Dedicated Sensor Units*

To enable robust environmental perception and accurate localization, our system incorporates four dedicated sensor units. Each of these units serves a specific role in the overall sensing and control architecture:

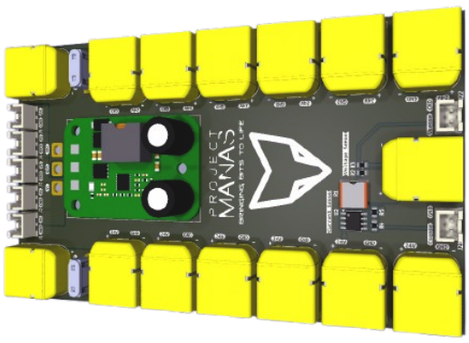



**Figure 7.** PDB Design

| | |
|---|---|
| **Battery Type** | Lithium Polymer |
| **Voltage Rating** | 6S (22.2V to 25.2V) |
| **Capacity** | 30,000 mAh |
| **Discharge Current** | 25C |
| **Nominal Runtime** | 60 min |
| **Full-load Runtime** | 40 min |

**Table 2.** Battery Specifications

1. **Ouster OS-1 LiDAR -** The Ouster OS-1 is a 3D LiDAR with a 120 m range and 45° vertical FOV. We used it for obstacle detection and Direct LiDAR Odometry (DLO).
2. **ZED 2i Stereo Camera -** The ZED 2i is a high-performance stereo camera with a 110°×70° field of view and up to 20 m depth sensing. We used it for lane and object detection to aid autonomous decision-making.
3. **Hiwonder IMU Module -** The Hiwonder IMU is a 10-axis sensor with 200 Hz output, used for magnetic heading (ENU) in NavSat Transform and motion data for EKF to enhance localization.
4. **Hiwonder GPS Positioning Module -** The Hiwonder GPS module offers < 1m accuracy and multi-constellation support. It is used for global positioning and waypoint-based navigation.

#### 5.3.2. *Control Units*

1. **Asus TUF A15 laptop -** The main computer consists of an AMD Ryzen 7 7735HS processor and an Nvidia RTX 4050 GPU. It runs the core ROS system and handles all high-level control and processing tasks.
2. **Communication Module -** The communication module connects to the laptop via USB (CDC) and manages communication along with additional onboard functions. The module forwards command velocities and direction to the Drive Controller over the RS-485 bus.
3. **Drive Controller -** The Drive Controller MCU interfaces directly with the motor drivers, issuing PWM signals to drive each wheel, and monitoring its RPM to process wheel odometry.
4. **Cytron MDDS30 -** The BDC driver is responsible for controlling the left and right BDC motors. It provides multiple safety features, such as protection against stray PWM signals before initialization.

### 5.4. ***Communication Module***

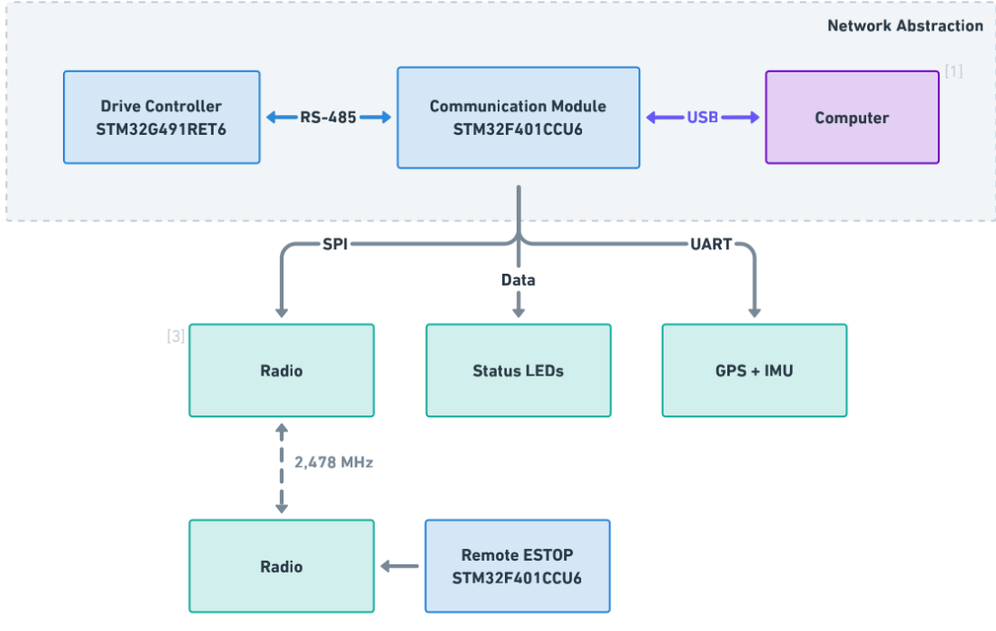



**Figure 8.** Communication Flow

The communication module handles all peripheral functionality of *STEVE*. It is responsible for connecting the Drive Controller to the computer, receiving data from GPS+IMU units, and controlling the safety lights. In addition, the module fully handles the remote E-Stop functionality via the NRF24L01 radio transceiver attached on board. Details of the E-Stop implementation are described in Section 5.6.

### 5.5. *Drive Controller*

The Drive Controller is implemented as an independent, self-sufficient control unit. The sole responsibility of the drive controller is to listen to `cmd_vel` fed into one of its input streams, and control all the on-board motors to exhibit the requested velocity.

Each motor has an independent adaptive PID, allowing for precise tuning across a wide range of speeds. The BDCs selected are Rhino 24V 60RPM IG52 Extra Heavy Duty Planetary Geared DC motor. Combined with our 14-inch wheels, this gives the bot a maximum linear velocity of 1 m/s ($\sim$ 2.2 mph). The BDCs have a stall torque output of 37.24 Nm, capable of handling the weight and payload on the bot. The control system hierarchy is described in the diagram shown below.

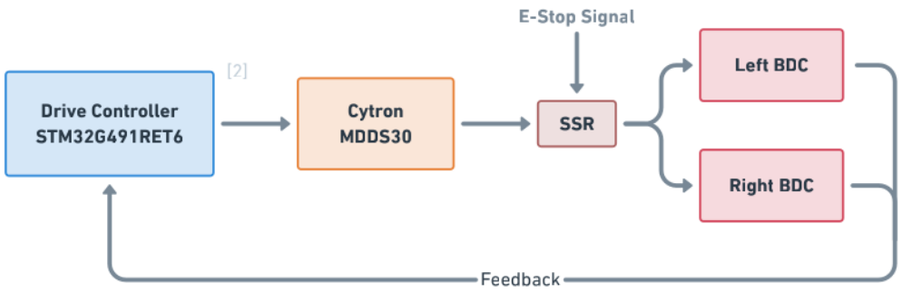
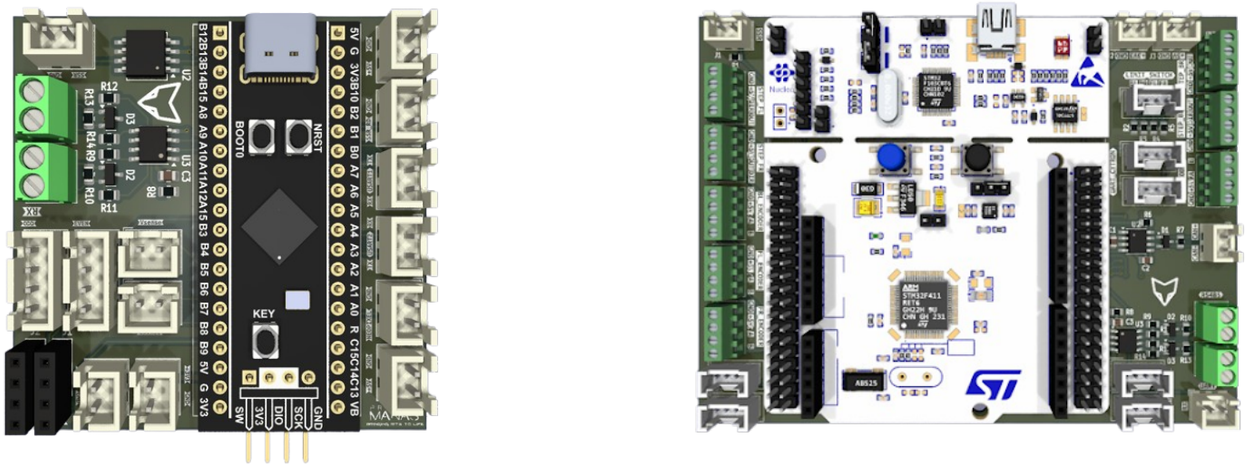



**Figure 9.** Drive Control Loop



(a) Communications Board

(b) Drive Controller Board

**Figure 10.** Controller Boards

### 5.6. *E-Stop*

The E-Stop requirements of the competition are implemented by a hardware-based solid-state relay (SSR) and are managed by the *Communication Module*. Therefore, there are two ways of triggering the E-Stop:

- Remote
- Mechanically

The Remote E-Stop is triggered when the E-Stop sequence is received on the nRF24L01 radio module. This causes the *Communication Module* to trip the SSR electronically. Additionally, a mechanical switch can also physically trip the SSR. When the SSR is tripped, all power to the motors is cut, stopping the bot.

## 6. DESCRIPTION OF SOFTWARE SYSTEM

### 6.1. *Perception*

#### 6.1.1. *Lane Filtering*

Lane detection begins by converting sensor images from RGB to HSV color space, which separates chromatic information from intensity and improves segmentation under varying lighting. A range-based filter is then applied in the HSV space to extract

lane-colored regions. Polynomial line-fitting refines the segmented areas by modeling the lanes as continuous curves and suppressing noise. The result is a binary mask that accurately captures the lane boundaries for navigation.

#### 6.1.2. *Obstacle Detection*

2D Laser Scan from the LiDAR is used to detect barrels for AutoNav (Refer to Section 7.2). YOLO is used in combination with a height-based filtering algorithm to detect the various obstacles mentioned in Self-Drive (Refer to Section 8.2).

### 6.2. ***Sensor Fusion and Localization***

Precise location is attained through combining data from several sensors which vary in range, frequency, and accuracy. This enables our system to have a reliable estimate of the vehicle's pose and motion accurately both on a local and global scale.

#### 6.2.1. *Direct LiDAR Odometry (DLO)*

Direct LiDAR Odometry is used to compute the vehicle's motion directly from dense 3D point clouds generated by the LiDAR sensor. This method aligns successive LiDAR scans to estimate relative movement over time. While DLO offers high accuracy and strong performance in unstructured environments, it operates at a relatively lower frequency compared to other sensors. Therefore, its output is used to stabilize and correct the drift from faster but noisier sources using Dual EKF.

#### 6.2.2. *Dual EKF Architecture*

A Dual Extended Kalman Filter (EKF) setup is used to enhance the robustness of state estimation:

- **The Local EKF** processes high-frequency inputs such as wheel encoder data, and IMU readings. This filter maintains a continuously updated estimate of the robot's position and velocity in real-time.
- **The Global EKF** fuses low-frequency but globally accurate data such as GPS coordinates and DLO-based odometry. This filter compensates for long-term drift in the local EKF by periodically correcting its estimate with global references.

This two-layer EKF architecture ensures that the system maintains both responsiveness and long-term accuracy.

| | GPS | LiDAR | IMU | | Wheel Odometry |
|---|---|---|---|---|---|
| Datapoints | $x, y$ | $x, y, \text{yaw}, v_x, v_y, v_{yaw}$ | $yaw$ | $v_x, v_y, a_x, a_y$ | $x, y, v_x, v_y$ |
| Type of Data | Latitude/Longitude | Odometry | IMU | IMU | Odometry |
| Scope | Global | Global | Local | Global | Local |

**Table 3.** Comparison of Sensor Data Characteristics

#### 6.2.3. *Localization*

To enhance positioning accuracy in known environments, the system integrates map-based localisation techniques. Specifically, we utilize Adaptive Monte Carlo Localization (AMCL), a probabilistic algorithm that estimates the vehicle's pose on a pre-saved map using particle filtering. By continuously comparing real-time sensor observations—such as LiDAR scans—with the static map, AMCL identifies the most likely position and orientation of the vehicle. In case no pre-saved map is recieved, the above mentioned Dual EKF module is used for localization

This method significantly improves localisation stability, especially in areas with weak or unavailable GPS signals. It also reduces computational overhead, as it eliminates the need for constant re-mapping. When fused with the global and local EKF outputs, AMCL further enhances the accuracy and reliability of the pose estimate, enabling safer and more confident navigation.

### 6.3. ***Mapping***

To enable efficient and accurate real-time mapping for navigation, a depth-based occupancy grid approach is adopted and enhanced with selective probabilistic filtering. Depth images from a ZED2i stereo camera are processed with a lane mask to isolate relevant navigational features while discarding background clutter. Selected depth pixels are projected into 3D space using the camera's intrinsic parameters and mapped onto 2D occupancy grid coordinates. An instantaneous occupancy grid is generated by marking the corresponding grid cells as occupied, and a log-odds framework refines occupancy probabilities over time through multiple observations over time. Moreover, a rolling window mechanism is implemented, shifting the mapped region as the robot moves. This limits memory usage. Updates are restricted to the visible field of view, ensuring the system remains responsive for real-time operation.

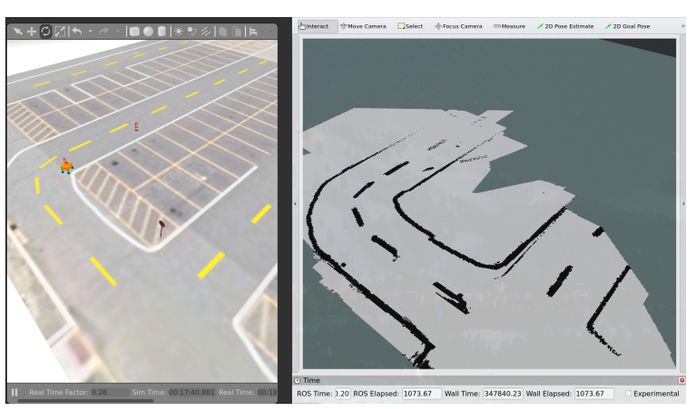



**Figure 11.** Mapping

### 6.4. *Lane Following*

To ensure stable lane following, goals are generated directly from the lane map without relying on the robot's orientation. The nearest white and yellow lane points are first identified using breadth-first search. A hill-climbing search is performed along each lane, selecting points that minimize angular deviation relative to the last two goals, promoting smooth forward motion. The candidate goal is computed as the midpoint of the optimized points. This is accepted if it satisfies the angular heuristic. The goal's orientation is set from the average of recent position estimates to maintain heading stability.

The first two goals (the start position and a point 1.5 meters ahead) are initialized, with subsequent goals computed dynamically using the described method. By focusing on lane geometry and minimizing orientation dependence, the technique ensures reliable center-lane tracking even during turns or partial occlusions, supporting robust and consistent forward movement.

### 6.5. *Path and Motion Planning*

The move_base package, provided by Robot Operating System (ROS), is used for motion and path planning, accompanied with a ROS bridge to maintain compatibility with ROS 2. The local waypoint generated by the goal-generation process serves as the input for this system.

The A* algorithm, provided by the NavFN plugin in move_base, is used for path planning. A* is well-suited for this purpose due to its high performance and speed, efficiently finding the shortest path to the target while avoiding obstacles based on the map data.

For motion planning, the TEB (Timed Elastic Band) planner is employed, and offers several key advantages:

- **Time-Optimal Trajectories -** TEB considers the dynamic constraints of the robot, generating time-optimal paths that respect velocity and acceleration limits.
- **Dynamic Obstacle Avoidance -** It effectively avoids moving obstacles in real-time by continuously updating the trajectory based on the robot's surroundings.
- **Efficient Path Planning -** The planner uses optimization-based techniques to generate smooth and feasible paths, avoiding unnecessary computational overhead.

By combining the strengths of A* for path planning and the TEB planner for motion planning, our system ensures fast, efficient, and collision-free navigation in complex and dynamic environments.

## 7. AUTO-NAV

### 7.1. *Overview*

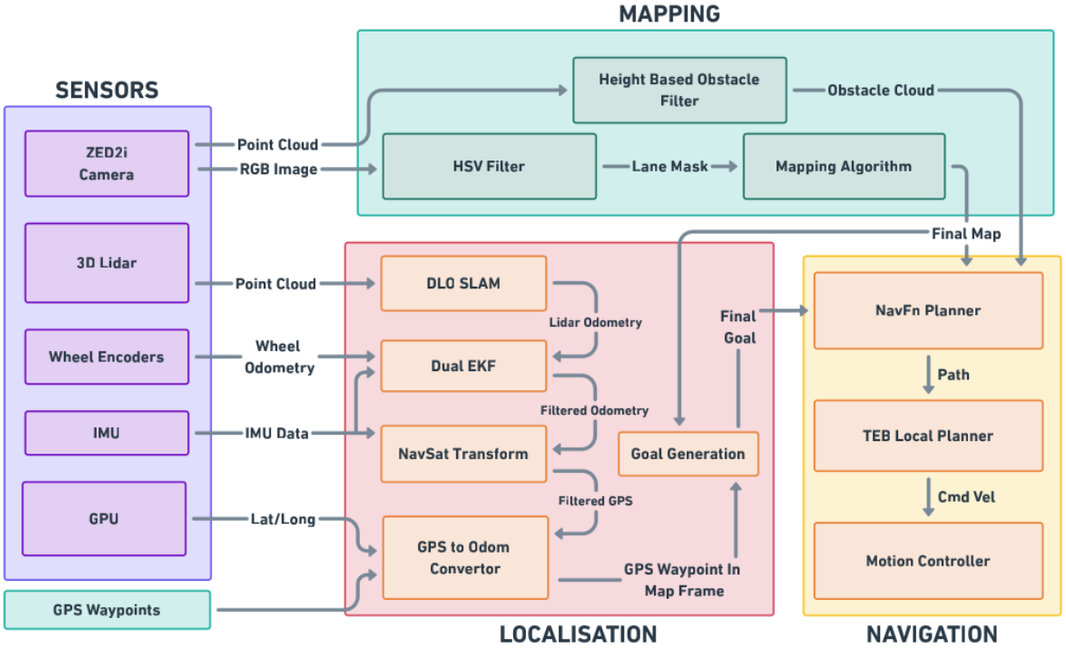



**Figure 12.** Auto-Nav Architecture

### 7.2. *Perception*

The incoming laser scan from the OS1 LiDAR, fused with odometry data, is used to detect barrels by analyzing the 3D point cloud for barrels. Using the robot's estimated position from odometry, the detected barrels are accurately localized on the global map. To prevent collisions, we inflate an area around each detected barrel on the global map. This inflated costmap increases the cost around the obstacles which forces the path planner to avoid paths that come too close to the barrels, ensuring safe navigation around obstacles.

### 7.3. *Goal Calculation*

#### 7.3.1. *Lane Following*

Using the approach described earlier in Section 6.4, we compute goals for lane following, given occupancy grid constructed from white lanes detected.

#### 7.3.2. *GPS Following*

Once the bot enters no-man's land and reach the first GPS waypoint, the goal calculator automatically switches to only publish the GPS coordinates provided. We apply Nav-Sat transforms which produce an odometry output with the position of the GPS in the map frame, and are published as goals to the motion planner.

## 8. SELF-DRIVE

### 8.1. *Overview*

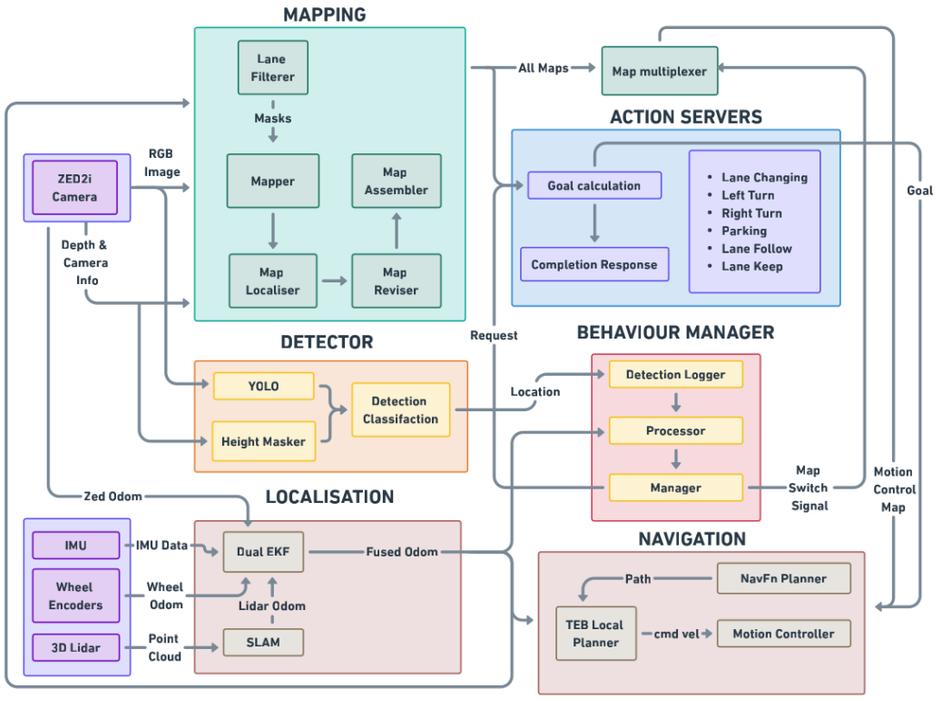



**Figure 13.** Self Drive Architecture

### 8.2. *Object Detection and Avoidance*

The obstacle detection framework combines deep learning-based object recognition with depth-based geometric validation to ensure accurate and reliable identification of environmental hazards.

Primary detection is performed using YOLO (You Only Look Once) object detection models, which provide real-time identification of key obstacles, with the relevant mAP50-90 (mean average precision) metrics listed out below:

| **Object** | **Train** | **Test** |
|---|---|---|
| Stop Sign (YOLO + OCR) | 0.92 | 0.88 |
| Tyre (YOLO) | 0.70 | 0.69 |
| Pedestrian (YOLO) | 0.68 | 0.64 |
| Traffic Drum (YOLO) | 0.75 | 0.71 |

**Table 4.** Model Accuracy

While YOLO provides high-confidence object classification, additional validation steps are necessary to maintain robustness, particularly in complex or cluttered environments. To this end, a height-based mask is applied to the depth image. This mask leverages prior knowledge of the expected physical dimensions of each obstacle type, enabling the system to filter out erroneous detections.

Building upon the perception capabilities, the system implements a deliberate and responsive obstacle avoidance mechanism. Following validation, the global coordinate position of each confirmed obstacle is computed and recorded. Upon detecting an obstacle at a new position, an appropriate response is triggered: a request is issued to the action server to execute a predefined avoidance maneuver. The system awaits confirmation of action completion before resuming its navigation tasks, ensuring that all behaviors are executed in a controlled and asynchronized manner.

## 8.3. *System Dashboard and Diagnostics GUI*

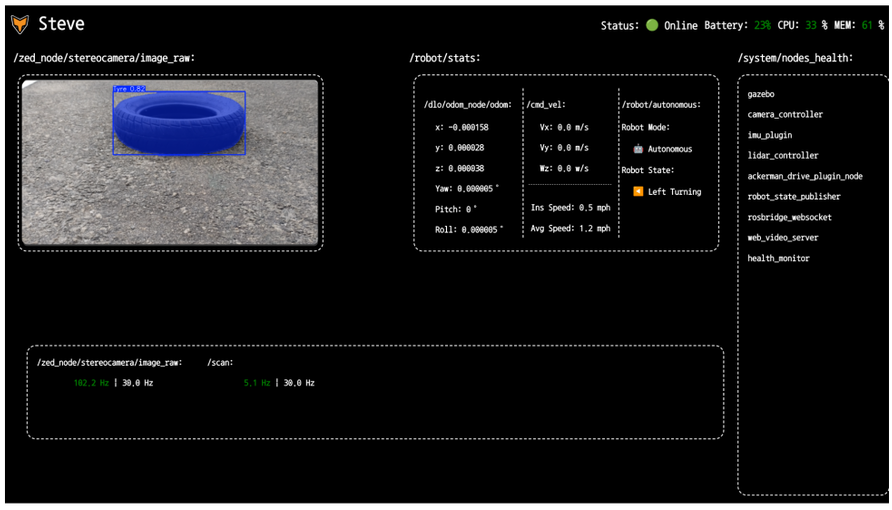



**Figure 14.** System Dashboard and GUI

We have developed a System Dashboard GUI as part of our software stack to support testing, debugging, and real-time monitoring during development and field runs. The dashboard provides a user-friendly interface to observe important system information while the robot is operating.

The GUI includes features like a live camera feed with detections, ROS topic frequency and health status, active ROS nodes, display current robot position, and velocity commands , along with many more useful tools. This dashboard has been extremely helpful during integration and testing by allowing us to quickly spot and resolve issues in the system.

## 8.4. *Mapping*

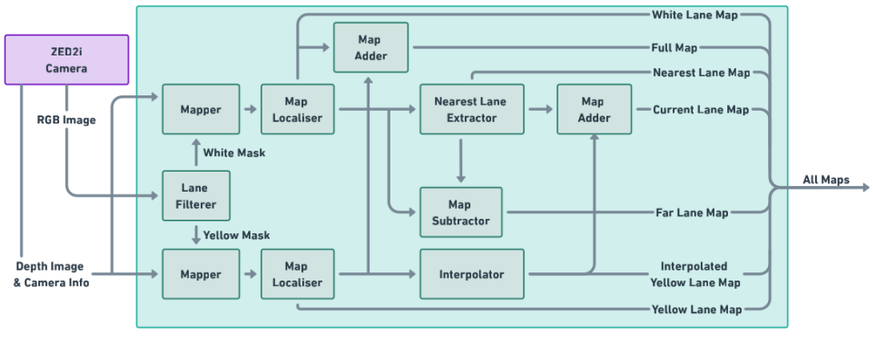



**Figure 15.** Mapping Architecture

To support different navigation tasks, we generate specialized lane maps instead of using a single, general-purpose map. These maps are tailored to highlight specific features like lane type, nearby segments, or continuity.

The process begins with the map-constructor, which uses a depth-based approach described earlier(Refer Section 6.3) to create separate occupancy grids for white and yellow lanes. The map-localiser then restricts the size of these maps to keep them lightweight and suitable for real-time processing. To handle missing or incomplete lane data, the map-reviser fills in gaps through interpolation and identifies the nearest lanes. Finally, the multi-map assembler combines or subtracts these maps to produce task-specific variants required by goal computation and navigation algorithms.

## 8.5. *Goal Calculation*

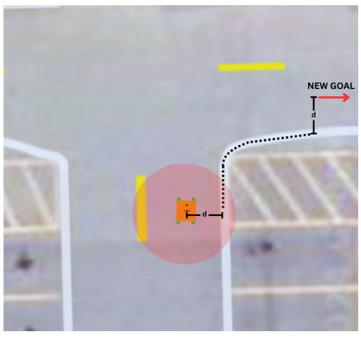
### 8.5.1. *Lane Following*

For lane following, we extract the nearest white lane markings and the yellow middle lane markings, to establish the current lane (left or right) that the robot is following on the occupancy grid. On this map, we employ the same algorithm explained in Section 6.4 to compute the goal.

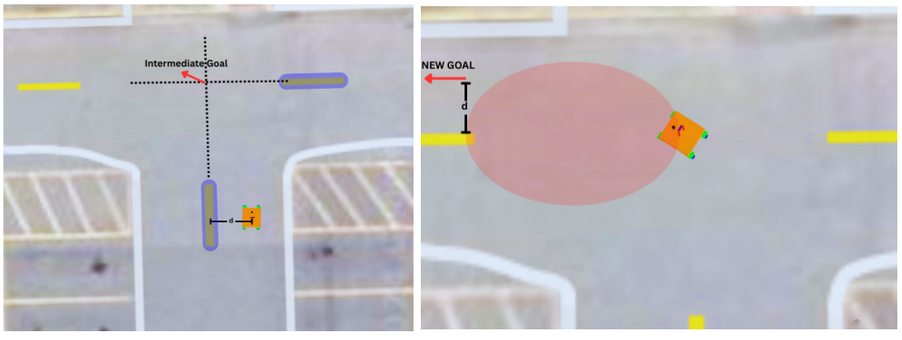
### 8.5.2. *Intersection Actions*

- **Right Turn -** Refer to Figure 16a. Breadth-First Search is run on the map to find the nearest point on the white lane and measure the offset distance. Using this, the algorithm follows the white lane across the intersection and sets a goal at the same offset, oriented perpendicular to the current lane's direction.

- **Left Turn -** Refer to Figure 16b. The algorithm uses DBSCAN to identify the two main lane clusters at the intersection and compute their centers. It then finds the intersection point between the line connecting these centers and the robot's heading, setting it as the intermediate goal. A Breadth-First Search locates the left lane, and the initial offset from the yellow lane is applied to determine the final goal, oriented perpendicular to the original lane's heading.
- **Lane Keeping -** The algorithm uses Breadth-First Search to find the nearest points on the yellow and white lanes and calculates their midpoint as the lane center. The goal's orientation aligns with the current lane's heading. A navigation goal is then set at a fixed distance ahead along this direction and is updated as the bot moves forward, keeping it centered in the lane.




(a) Right Turn




(b) Left Turn

**Figure 16.** Intersection Actions

#### 8.5.3. *Parking*

We utilise similar approaches to accomplish all three modes of parking:

- **Pull In -** The surrounding area is scanned for the parking area, and it's trajectory is accordingly planned for entry. If obstacle-free, goals along this path are published and followed.
- **Pull Out -** As the bot exits the parking area, it's angular velocity increases proportional to the inverse tangent of 1/distance to smoothly align with the lane. The bot then halts 3 feet from a detected traffic drum.
- **Parallel -** Occurs similar to Pull In, with the final goal's orientation being matched to the bot's initial orientation, ensuring a smooth, parallel entry into the parking spot.

#### 8.5.4. *Lane Changing*

The robot receives a processed binary map (representing lane markings) and applies the DBSCAN clustering algorithm to group nearby yellow pixels into clusters, effectively segmenting the yellow lane marking. Using the robot's current position within its lane, the algorithm performs Breadth-First Search inside the cluster to find the furthest point. Using geometrical calculations, it applies an offset, extrapolating the furthest point on the lane marking to determine a new goal position in the adjacent lane.

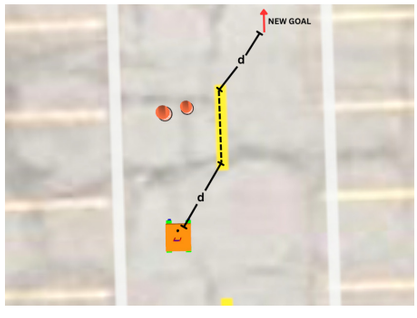



**Figure 17.** Lane Change

## 9. CYBER SECURITY ANALYSIS USING RMF

### 9.1. *NIST Risk Management Framework (RMF)*

The NIST RMF outlines a 7-step approach that organizations can adopt to cost-effectively implement and maintain their cybersecurity policies. This flexible framework ensures continuous monitoring and improved security posture. The steps include:

- **Prepare -** In this initial phase, the organization establishes readiness for RMF by defining roles and creating a strategic plan.
- **Categorize -** The system is classified according to control requirements. Impact levels—based on confidentiality, integrity, and availability—are assessed and finalized.
- **Select -** With the impact levels and risk tolerance defined, appropriate controls are chosen, and a plan for implementation and oversight is developed.
- **Implement -** The selected security controls are integrated into the information system.
- **Assess-** An independent evaluation is conducted to verify that the controls function as intended and meet established requirements.

- **Authorize -** A designated senior official, known as the Authorizing Official (AO), reviews the risk assessment to determine if the system's risk posture is acceptable, granting formal authorization accordingly.
- **Monitor-** The system undergoes continuous monitoring to detect and respond to potential security threats.

### 9.2. *Model the threats and analyze their impact*

| System | Threat Description | Confidential | Availability | Integrity | Overall |
|---|---|---|---|---|---|
| GPS Waypoints | If the GPS is altered the bot may go in wrong direction | low | moderate | high | high |
| Odom | If the odometry is altered bot may not know its relative position | low | high | high | high |
| ROS Commands | If velocity commands are intercepted the bot can be controlled | low | high | high | high |
| Version control credentials | Getting access to version control credentials can be used to change bot code | high | low | high | high |
| Wireless E-stop command | Rival team may send a wireless estop command to sabotage the run | high | moderate | low | moderate |
| Bag files | Bag files are only used for debugging, unlikely to be a threat | low | low | low | low |
| Onboard Computer | The onboard computer acts like the central node which can be used to control everything | high | high | high | high |
| Power Distribution Board | The main board responsible for relaying the commands to the parts | high | high | high | high |
| Printed Circuit Board | The board which receives sensor board data, if modified can be used to send wrong sensor data | high | high | high | high |
| Detector | If the detector produces false positives or negatives, it may cause incorrect interpretation of the environment | high | low | high | high |
| Behaviour Manager | If the behaviour manager is compromised, the bot may take incorrect actions despite correct inputs | low | high | high | high |

### 9.3. *Cyber Controls and descriptions of their implementations*

| Goal | Implementation | Efficacy |
|---|---|---|
| **AC-1 (Access Control)** | | |
| Protecting code from being modified by a rival team | Github is used to manage version control for better security and access control | High: Delegating version control to Github reduces risks the team has to take |
| **AC-4 (Information Flow Enforcement)** | | |
| Protecting bot from receiving commands from rival team | Only the onboard computer may be connected to the internet and local parts are part of a local network which is not accessible from the internet so only the parts on bot can communicate with each other | High: Stops most wireless attacks targeted at onboard parts |
| Receiving wireless ESTOP signal from rival team | The ESTOP signal will be encrypted using RSA with a timestamp to stop man-in-middle or someone else sending an estop command | Medium: Prevents "man in the middle" attacks |
| **SC-41 (Port and I/O Device Access)** | | |
| Stopping rival team from adding devices | Extra ports are blocked, and a custom circuit board is employed to exclusively permit essential devices. | High: Stops rival team from adding malicious devices |

| Goal | Implementation | Efficacy |
|---|---|---|
| **AC-10 (Concurrent Session Control)** | | |
| Stopping rival team from connecting to central node | The central node only allows a set number of devices to connect to it | High: Stops rival team from connecting to central node |
| **IA-5 (Identification and Authentication)** | | |
| To prevent unauthorized users from gaining access to the onboard computer | Each user has to enter a root password and a separate login password, only accepted if there is a minimum password length, at least 1 digit, at least 1 uppercase character | Moderate: Bears the brunt of the most rudimentary entry level attacks |

## 10. ANALYSIS OF COMPLETE VEHICLE

### 10.1. *Lessons learned during construction and system Integration*

Throughout the build and design process, our aim was to determine the shortcomings of our previous design and improve on those areas in a significant manner. This led us to the development of the suspension linkage across the two casters, overcoming the difficulties we faced while traversing the ramp last year.

One key lesson we learned was the importance of choosing the right software early. We started with ROS1 but subsequently switched to ROS2. This gave us better performance, more reliable communication, and was easier to scale. It also has a more active open-source community, which was highly beneficial. This taught us how important it is to use software that can grow with the project.

### 10.2. *Potential Hardware Failures and their Mitigation*

| **Failure Points** | Causes | Solutions |
|---|---|---|
| **ZED2i Camera Vibration Instability** | Loose movement from the lead screw and vibrations from rough terrain | Addition of inclined carbon fiber rods to form a triangulated brace for enhanced rigidity and minimizing lateral deflection |
| **Connection Integrity Issues** | Increased number of wires and components | Designed a PCB and use of clip-on connectors |
| **SSR Failure at low temperatures** | Operation in suboptimal temperatures | Introduction of gate driver to drive the SSR |
| **Overvoltage** | Excessive current draw | Installation of fuses and setting appropriate current bounds in the motor drivers |
| **Suspension Mount Fracture** | Concentrated stress at chassis attachment points while traversing uneven terrain | Use of compliant mounts and load-spreading brackets to reduce stress concentration |

### 10.3. *SIL Virtual Environment Testing*

Testing in the real world is hard and time-consuming, and problems are often difficult to debug. To save time and avoid damage, we first test in simulation using Gazebo. Our virtual robot matches the real one in size, weight, and sensor placements. The simulated course mimics the real competition conditions with roads, lanes, obstacles, and a ramp. Each run has random lane paths and obstacle positions to test different cases.

**Figure 18.** Software In the Loop

### 10.4. *Software Testing, Bug Tracking, and Version Control Process*

We used GitHub for bug tracking and version control, organizing our development through protected branching strategies. The main branch was reserved for deployment on the robot and kept secure, while feature-specific sub-branches were created for each component of the Self-Drive and AutoNav competitions. Dependency management was handled through virtual environments to ensure reproducibility. For debugging and data analysis, we utilized GDB and ROS bags extensively, enabling step-by-step debugging, logging, and replaying of robot behavior to gain deeper insights into system performance. These comprehensive tests were designed to guarantee consistent performance in all expected real-world conditions.

| Failure Points | Causes | Solutions |
|---|---|---|
| **Lane Inconsistency** | Due to outliers and limitations in camera inaccuracy, lanes can be captured inconsistently in the costmap which can lead to an incomplete map. | Goal generation (refer to Section 6.4) generates intermediate waypoints. |
| **False detection by YOLO** | Deep learning models can never give us 100% accuracy, so false detections can occur. | A height-based filtering system is used to validate the detection so it matches the actual object dimensions. |
| **No possible path to goal** | Inaccurate costmap due to distorted image data and phantom obstacles being mapped. | Navigation Recovery Mode: perform a 360° scan to remap the local surroundings and clear any incorrectly mapped obstacles. |
| **Sensor Data Inaccuracy** | Accumulated error from GPS drift and odometry. | Implemented Dual EKF (refer to Section 6.2.2) to fuse GPS and IMU for more robust pose estimation. |

**Table 5.** Perception Failure Points, Their Causes, and Our Solutions

### 10.5. *Physical Testing to Date*

- **Mechanical:** Mechanical testing of STEVE emphasized structural performance and vibration mitigation under dynamic loading. A new suspension system was introduced to improve ground contact and shock absorption across uneven terrain. This system was directly mounted to the chassis and tested for deflection and damping response, showing a notable improvement in ride stability and impact handling.
- **Electronics:** Most parts of the electronics system, including communication lines and the power distribution board, have been tested and are working well. The motors and motor drivers have been fully assembled on the robot and tested, ensuring smooth functionality. Safety systems like Emergency Stop (E-Stop) and safety lights have also been tested and are working as expected. We are continuing to make improvements to the electronics stack to make it more efficient and reliable.
- **Software:** To make sure the software works reliably, we tested it in conditions similar to the ones in the competition, including thorough checks of each part of the system.
  - Mapping and obstacle detection were tested both during the day and night to ensure they work well in different lighting.
  - The navigation system was put through various conditions, including lanes that were partially visible and obstacles of different shapes and sizes.
  - GPS navigation was also tested to ensure it followed the planned routes accurately. Finally, detection models were tested in tough situations, including cases where lanes were partially blocked.

## 11. INITIAL PERFORMANCE ASSESSMENT

*STEVE* was rigorously tested on makeshift AutoNav and Self-Drive courses, with the AutoNav course averaging a run time of 4 minutes per attempt. *STEVE* was able to reliably complete the courses and function tests, including dynamic obstacles and ramps.

| | |
|---|---|
| **Runs Performed** | 152 |
| **Average Course Time** | 4 min 02 seconds |
| **Average Speed** | 0.6 m/s |
| **Max Speed** | 1 m/s |
| **Max Acceleration** | 1.4 $m/s^2$ |
| **Battery Life** | 40 min |
| **Waypoint Accuracy** | 0.1 meters tolerance of the goal point |
| **Distance at Which Obstacles Are Detected** | 120 meter LiDAR range for obstacles, 20 meter ZED range |
| **Planner Frequency** | 5 Hz path update rate |