Intelligent Ground Vehicle Competition 2026

Manipal Academy of Higher Education

Project Manas

Design Report

ALYX

Self-Drive & AutoNav

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

Dr.Ashalatha Nayak
Professor, School of Computer Engineering
Email : asha.nayak@manipal.edu
Phone no.: +91 94817 53598

Date: May 14, 2026

Team Members

Sensing and Automation	Artificial Intelligence	Mechanical
Akshat Pankaj Kakade Aryan Pagaria Astin Dsouza Shubh Kesarwani Soham Saxena Taman Raja Kochi	Aditya Singh Waldia Asavari Kaushal Chirag Rao KV Hariharan VS Jasmine Pannu Michelle Andrade Onas Chaturvedi (Team Captain) Sparsh Agarwal	Anant Shankar Gupta Anup Kamath Bedobrata Saikia Bhuvan Chand Rayani Preet Shah Yash Ralhan

First Point of Contact : Onas | onaschaturvedi@gmail.com | +91 78190 11071

Second Point of Contact : Akshat | akshat.kakade@gmail.com | +91 91759 51663

Contents

1. SYSTEM AND SUBSYSTEM REQUIREMENTS

1.1 System Engineering Process

To manage development of ALYX, an approach adapted from the Integrated Product and Process Development (IPPD) lifecycle was adopted, well-suited to the team's cross-disciplinary nature. After understanding competition requirements, the team followed an iterative process of reviewing progress across each subsystem. Once subsystem goals were met, cross-subsystem integration began, followed by continuous testing and refinement to achieve optimal performance.

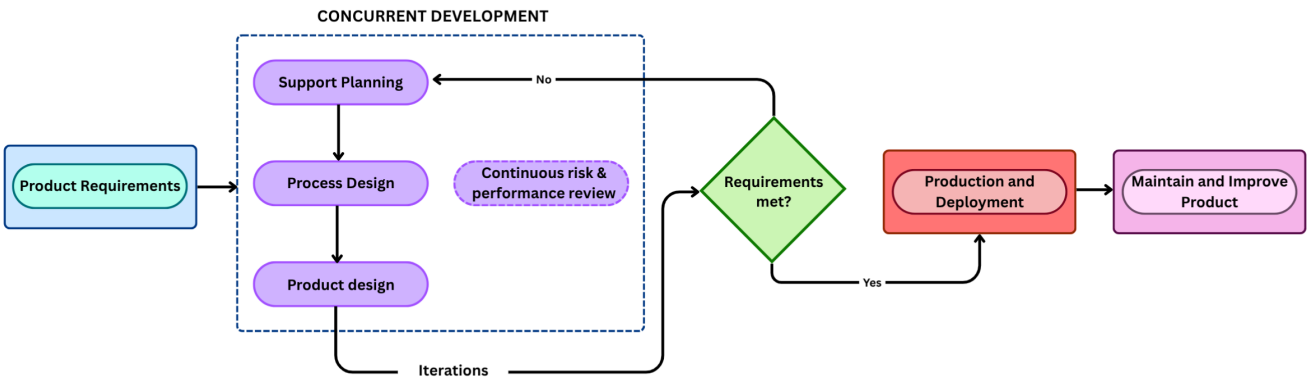

Figure 1: IPPD design process for ALYX development.

1.2 Hardware & Safety Requirements

1.2.1 Mechanical Requirements

Omnidirectional Maneuverability: The vehicle must decouple its travel direction from its orientation to navigate tight obstacles without multi-point turns.

- **Target:** Achieve a true zero turning radius and seamless lateral translation (strafing) capability.
- **Verification:** Field-test pivot and 90° lateral maneuvers to measure center deviation, ensuring minimal motor housing binding under high torque.

Passive Suspension: The system must provide a sturdy and reliable suspension capable of handling surface irregularities and protecting the chassis from shocks.

- **Target:** TPU tires should provide sufficient passive flexibility to absorb small surface irregularities and reduce shock transfer to the chassis.
- **Verification:** Drive vehicle over a measured surface bump and record chassis vibration, confirming reduction in shock amplitude compared to wheel-level input.

1.2.2 Electrical, Electronic, and Embedded Requirements

Deterministic Control Loop: Low-level motor control must operate independently of the main computer to provide a guaranteed, real-time response.

- **Target:** The control loop must execute at a minimum frequency of 100 Hz, with a timing jitter of less than 1 μs.
- **Verification:** Profile timer interrupts using internal CPU cycle counters to confirm timing consistency.

Precision Velocity Tracking: The robot must accurately track commanded linear and angular velocities to ensure physical motion matches the published velocity commands.

- **Target:** Steady-state linear and angular velocity errors must not exceed 0.05 m/s and 0.05 rad/s during operation.
- **Verification:** Continuously compare commanded linear and angular velocities against robot velocity estimates derived from onboard LiDAR-based odometry and IMU feedback.

1.2.3 Safety Requirements

Wired and Wireless E-Stop Response Time:

- **Target:** Time from wired trigger to full motor power cutoff $<$ 100 ms and time from wireless trigger to full motor power cutoff $<$ 250 ms.
- **Verification:** A logic analyzer was used to measure the delay between E-stop activation and motor power cutoff.

Wireless E-Stop Range:

- **Target:** Operational range of wireless E-Stop link $\geq$ 100 m.
- **Verification:** Wireless E-stop functionality was tested by triggering E-stop commands at multiple distances up to 150 m and verifying successful motor power cutoff.

Post E-Stop State:

- **Target:** All drive commands zeroed, and all steering modules returned to zero orientation.
- **Verification:** Post E-stop behavior was validated by triggering E-stop during operations and verifying zero drive output commands and steering module zero return behavior.

1.3 AutoNav Software Requirements

1.3.1 Perception

Reliable Lane Segmentation: The perception system must robustly segment lanes despite temporal variations, varied lighting, and dirty surfaces.

- **Target:** Design a lane segmentation solution that achieves inference speeds of less than 50 ms and an IoU greater than 0.8.
- **Verification:** Evaluated inference speed and accuracy under variable real-world conditions.

Efficient Mapping: The internal map representation must successfully mitigate sensor noise while incurring minimal computational overhead.

- **Target:** Achieve mapping of the environment with accurate positioning and persistence of obstacles at high speeds and update latency under 50 ms.
- **Verification:** Measured mapping speed and accuracy during real-world testing on a dummy course with obstacles such as barrels and boxes.

1.3.2 Driving Logic

Consistent Course Progress: The goal generation module must continuously generate intermediate goals to maintain forward momentum, even in scenarios where lanes are occluded.

- **Target:** The vehicle must maintain an average velocity above the mandatory 1.0 mph (0.45 m/s) minimum, even through sharp turns or lane occlusions.
- **Verification:** Measured average autonomous velocity on a dummy course featuring tight turns and lane occlusions.

Agile Traversal and Obstacle Clearance: The planner must generate safe, high-speed trajectories that respect vehicle dimensions and maintain strict obstacle buffers.

- **Target:** Ensure no collisions and maintain a distance greater than 0.2m from obstacles throughout the course. Maintain an average speed greater than 1.2 m/s.
- **Verification:** Conducted extensive field-testing of the vehicle on various dummy courses and derived key metrics from acquired data.

1.3.3 Key Performance Indicators

Course Completion Efficiency: The system must ensure continuous forward progress to complete the course as rapidly as possible so as to ensure competitive performance in the AutoNav challenge.

- **Target:** Complete a similar course to the one described in the rules under 3 minutes while maintaining an optimal average speed.
- **Verification:** Recorded course completion times in a dummy course and compared to the target time.

Safe Navigation: The robot must strictly avoid boundary violations, lane crossings, and obstacle collisions.

- **Target:** No lane or course boundary violations and no collisions with obstacles by maintaining a safe margin from them.
- **Verification:** Conducted extensive field-testing of collision avoidance and course navigation systems by constructing dummy scenarios that mimic course obstacle layouts.

1.4 Self-Drive Software Requirements

1.4.1 Perception

Reliable Object Detection: The perception pipeline must accurately detect, classify and track both dynamic and static obstacles (tires, barrels, mannequins).

- **Target:** Maintain false-positive and false-negative detection rates at or below 5%.
- **Verification:** Evaluated detection metrics across varied test scenarios, comparing processed LiDAR and YOLO based outputs against physical ground truth data.

Urban Infrastructure Recognition: The system must recognize intersections, potholes, and read traffic sign text via OCR to obey road rules.

- **Target:** Achieve >90% accuracy for STOP sign detection and text verification, and 95% accuracy when mapping intersections and potholes across varying distances.
- **Verification:** Conducted field tests with physical stop signs, artificial potholes, and lane markings, measuring the true positive detection rate and map placement accuracy.

1.4.2 Driving Logic

Stable Lane Following and Obstacle Avoidance: The driving system shall maintain stable lane tracking using the lane centerline extracted from the lane map obtained by the ZED2i camera.

- **Target:** Enabled stable and safe lane transitions with consistent obstacle avoidance and minimal path deviation, achieving a 90% success rate.
- **Verification:** Tested on tracks with static obstacles to measure path deviation, obstacle clearance, transition smoothness, and the success rate of rejoining the lane.

Complex Maneuver Execution: The vehicle must autonomously coordinate precise trajectories for intersections and multi-step parking sequences via Nav2.

- **Target:** Execute intersection and parking splines sequences autonomously, achieving final parking placement within $\pm$0.05m error from the designated slot.
- **Verification:** Performed repeated trials of turning and parking sequences in dummy urban environments, measuring final pose error against expected ideal coordinates.

1.4.3 Key Performance Indicators

System Latency: For structured urban environments, the vehicle must rapidly process and respond to static and dynamic hazards, such as STOP signs or sudden lane blockages.

- **Target:** Achieve a latency of $\leq$150 ms from perception to control signal output.
- **Verification:** Measured the latency from the exact moment a hazard is perceived by the sensors to the publication of the deceleration command.

Safe halt distance: The vehicle must possess the mechanical braking authority to completely stop within a strict safety buffer.

- **Target:** Come to a complete halt within a $\leq$1.0 meter stopping distance from a cruising speed of 1.5 - 3 mph.
- **Verification:** Conducted automated braking tests, physically measuring the final stopping distance and cross-referencing with odometry logs.

2. MECHANICAL DESIGN

2.1 Design Overview

The mechanical design of ALYX emphasizes maneuverability, structural integrity, and modularity, implemented using T-slot profiles that allow rapid hardware reconfiguration. The platform adopts a holonomic drive system, enabling omnidirectional motion with minimal turning constraints. To improve navigation in narrow spaces, the vehicle width was reduced by integrating in-hub motors within the wheel assemblies, resulting in a compact drivetrain and increased internal volume for electronics and mounting.

To ensure that the final design was optimized for weight distribution and torque capacity, we incorporated Fusion 360 for detailed modeling and calculations, ANSYS for structural testing under maximum loads, and MATLAB for performance verification and analytical checks.

Figure 2: Structural overview of ALYX

The main frame employs 30 × 30 mm aluminum T-slot profiles, connected at the joints using aluminum corner brackets. The frame allows us to minimize vehicle width by integrating in-hub motors within the wheels. The ZED2i camera mount is reinforced with full-height 30×30 mm profiles and carbon fiber bracing to minimize vibrations. The lower holonomic assembly uses 20 × 20 mm profiles, chosen to suit the in-hub wheel geometry and reduce overall weight.

2.2 Wheels

ALYX has a three-layered 3D-printed wheel design optimized for low mass and structural robustness. The hollow wheel core accommodates in-hub motors, eliminating external mounts and reducing the overall width of the vehicle.

The wheel is made up of three functional zones:

Inner Rim: Made from mild steel for structural integrity and load-bearing performance.

Intermediate Layer: Printed in PLA for lightweight support and to maintain the wheel's precise shape.

Outer Layer: A flexible TPU layer, chosen for traction on various terrains.

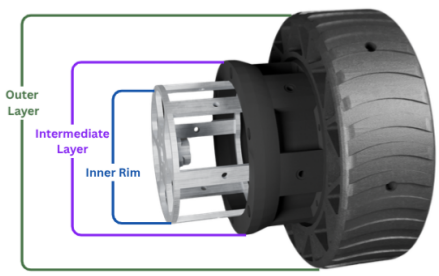

Figure 3: Functional zones of wheel

The TPU outer layer compresses under load, functioning as a passive suspension system. This material-efficient approach, instead of traditional heavy wheel assemblies, decreases mechanical strain on the motors and extends battery life.

2.3 Drive System

ALYX uses a holonomic drive system enabling accurate omnidirectional control. Each wheel unit combines a stepper motor for orientation control and a BLDC motor for propulsion. This configuration enhances maneuverability without requiring a large turning radius. The system delivers up to 24 N·m torque at a total mass of 70 kg (Table 1).

Parameter	Value
Length	3 feet
Width	2 feet
Height	5.1 feet
Max torque	24 N-m
Mass with payload	70 kg

Table 1: Specifications

Figure 4: Drive Bracket

2.4 Covering and Weatherproofing

ALYX uses acrylic panels as its outer enclosure, offering a lightweight and rigid structure with efficient fabrication and integration. The material delivers reliable protection and supports a compact, well-organized design, and was chosen based on the rigorous evaluation in Table 2.

Material	Acrylic	PVC	Polycarbonate	HDPE	Aluminum Sheet	CFRP
Lightweight	✓	✓	✓	✓		✓
Rigidity	✓		✓		✓	✓
Impact Resistance			✓	✓	✓	✓
Surface Finish / Appearance	✓		✓		✓	✓
Electrical Insulation	✓	✓	✓	✓		
Availability	✓	✓	✓	✓	✓	
Cost Effectiveness	✓	✓		✓		

Table 2: Material Comparison

2.5 Requirement Validation

A holonomic wheel arrangement was adopted to achieve **Omnidirectional Maneuverability** with a true zero turning radius, allowing the vehicle to move in any direction and rotate on the spot, without multi-point turns. TPU tires were selected for their **Passive Suspension** properties, naturally absorbing surface irregularities and reducing shock transfer to the chassis.

Metric	Target Value	Measured Value	Status
Zero turning radius deviation	$<$5 cm from center point	1–2 cm	Passed
TPU tire shock absorption	Absorb surface irregularities and reduce shock transfer	Verified	Satisfied

Table 3: Mechanical Requirement Validation

3. SAFETY

3.1 Transportation, Parking, and Charging Safety

The bot can be set in IDLE mode during transportation and post-parking. This locks the motor positions and disables the bot's movement. Safe battery charging practices are ensured by charging the battery off-the-bot using a dedicated LiPo balance charger to ensure safe operation and proper cell balancing. The onboard computing device is powered through a USB Type-C charging board compliant with USB Power Delivery 3.0.

3.2 Operational Safety and E-STOP

3.2.1 Emergency Stop and Safety Response

The safety architecture implements two independent emergency stop mechanisms.

- **Wired E-Stop:** Directly interrupts motor power through a hardware-level, SSR-based cutoff circuit.
- **Wireless E-Stop:** A dedicated safety microcontroller continuously monitors the independent wireless communication channel. Upon receiving a kill command or detecting communication loss, the controller forces the robot into an ESTOP state and immediately cuts drive motor power. This operation is independent of the high-level commands and control software.

In addition to hardware shutdown, the embedded controller executes a controlled recovery behavior in which all drive motor commands are immediately zeroed, and the power to the drive motors is disabled while steering modules remain temporarily energized to execute a controlled return-to-zero procedure, and all pending motion commands are cleared. This ensures a deterministic post-stop system state and safe re-initialization.

3.2.2 Mode Enforcement and Safety Authority

The dedicated safety microcontroller acts as the safety layer, overseeing the operational mode of the primary motion controller. This safety board reads physical mode switches and enforces states accordingly. In case of an invalid input, the bot is transitioned into an IDLE mode to ensure safety. The E-Stop is always given priority over physical mode switch states.
This separation ensures that safety is not dependent on high-level autonomy software and that modes can be changed dynamically during runtime without requiring restarts or separate setups.

3.2.3 Safety Indication System

A programmable LED array provides visual feedback of system state. Solid colors indicate non-autonomous system states, blinking patterns indicate autonomous operation modes and a solid red indication represents an active ESTOP condition.

3.3 Requirement Validation

The dedicated safety controller and the independent hardware cutoff circuit contribute to reducing the **Wired E-Stop Response Time** and **Wireless E-Stop Response Time** requirements. This is achieved by ensuring quick motor cutoff through the utilization of an SSR.
Reliable NRF communication satisfies the **Wireless E-Stop Operational Range** requirement, while post-stop motor zeroing and steering reset logic fulfill the **Post E-Stop System State** requirement.

Metric	Target Value	Measured Value	Status
Wired E-Stop response time	100 ms to full motor power cutoff	$<$30 ms	Passed
Wireless E-Stop response time	250 ms to full motor power cutoff	$<$150 ms	Passed
Wireless E-Stop operational range	$\geq$100 m	150 m	Passed
Post E-Stop system state	All drive commands zeroed and steering modules at zero orientation	Verified	Passed

Table 4: Emergency Stop System Validation

4. ELECTRICAL/ELECTRONIC DESIGN

4.1 Overview of Electrical Architecture

The electrical architecture is designed as a layered system separating high-level autonomy, real-time control, and safety enforcement. An onboard laptop handles perception and high-level decision-making. The drive controller is responsible for real-time motion control and actuation, while the safety board handles safety enforcement and mode control. A centralized power distribution board supplies regulated multi-rail power, and the sensor suite includes LiDAR, camera, encoders, GPS, and IMU. This architecture ensures that failures in high-level software do not propagate to low-level, safety-critical systems.

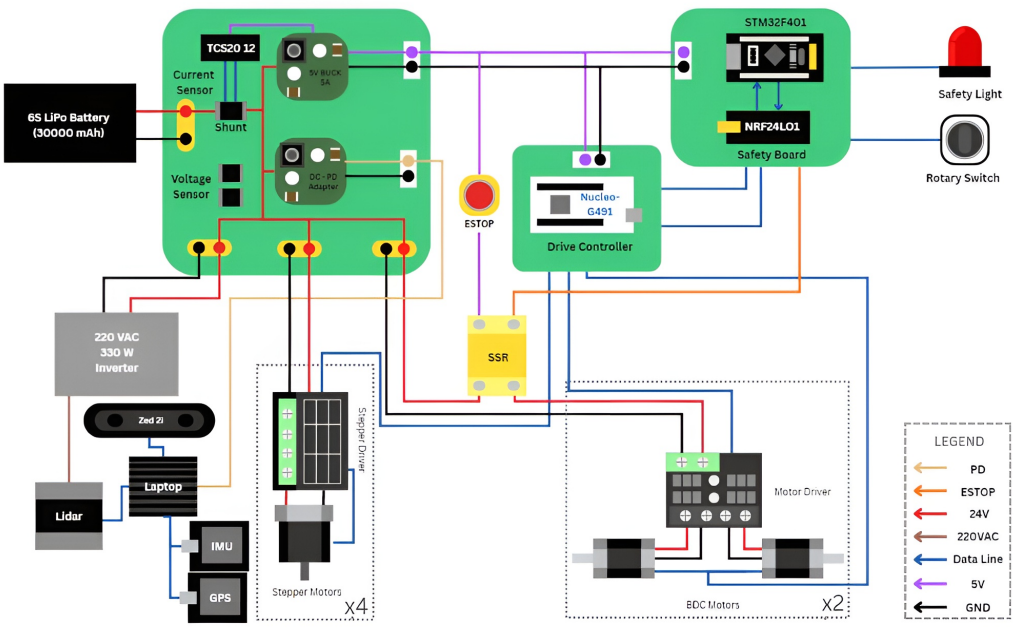

Figure 5: Electrical Architecture Diagram

Component	Model	Remarks
Battery + PDB	Tattu 6S 30000mAh 25C + Custom PDB	Provides 24V, 19V, 12V, 5V rails
Compute Device	Mini PC / Laptop	High-level perception and autonomy
Motor Control Board	NUCLEO (STM32G491RE)	Real-time motion control
Safety Board	Black Pill (STM32F401CCU6)	Safety enforcement, mode control, auxiliary peripherals
Brushed Motor Driver	Cytron MDD20A	Drives BDC motors using PWM input
Brushed Motors	GoBilda Saturn 5303 (260 RPM)	Provides rotation and movement
Stepper Motor Drivers	StepperOnline CL57T-V41	Drives stepper motors (closed-loop control, PWM input)

Stepper Motors	NEMA 24 Closed Loop Stepper (4.0 Nm) with a 5:1 planetary gearbox	Used for steering wheel direction
Sensors	• Ouster OS1 LiDAR • Magnetic (Hall Effect) Encoders • HiWonder IMU • HiWonder GPS Module • Zed2i Stereo Camera	Real-time perception and sensing
Wireless Communication Module	NRF24L01 2.4GHz PA/LNA transceiver with external antenna	Wireless E-stop

Table 5: Electrical and Electronic Components Overview

4.2 Communication Architecture

The system uses multiple communication channels:

- Computer to Drive Controller via UART with DMA for motion commands.
- Safety Board to Motor Control Board via UART with interrupt for mode and safety control.
- NRF wireless channel from wireless E-stop module to the safety board.

Incoming motion commands are buffered. This allows communication handling and command execution to operate independently.

4.3 Drive Control System

Accurate drive control is achieved through closed-loop RPM control with encoder feedback. A PID controller is employed with deadband filtering, integral decay, and slew rate limiting to suppress oscillations under varying load conditions. Encoder measurements are filtered using an exponentially weighted moving average to improve control accuracy.

Steering is implemented using independent stepper motors for each wheel, featuring absolute angle control, constraint-aware motion planning, timer-driven pulse generation, and responsive command execution. This enables precise and smooth steering transitions. Acceleration smoothing is also applied to stepper motion to provide controlled ramp-up, preventing mechanical shock and improving trajectory smoothness.

To ensure accurate zeroing, each stepper motor has a corresponding limit switch positioned at a known offset. During the homing sequence, each stepper rotates up to 360° anti-clockwise until its respective limit switch is triggered, allowing it to be zeroed precisely.

During normal operation, steering motion is constrained to prevent the limit switches from being triggered. The direction of rotation is selected based on the shortest-path arc to minimize actuation time, determined by both the target angle and the current travel direction of the BDC. This ensures that all valid steering movement remain under 90°. A timer-interrupt-driven control loop enforces consistent execution at a fixed frequency, eliminating the influence of external delays.

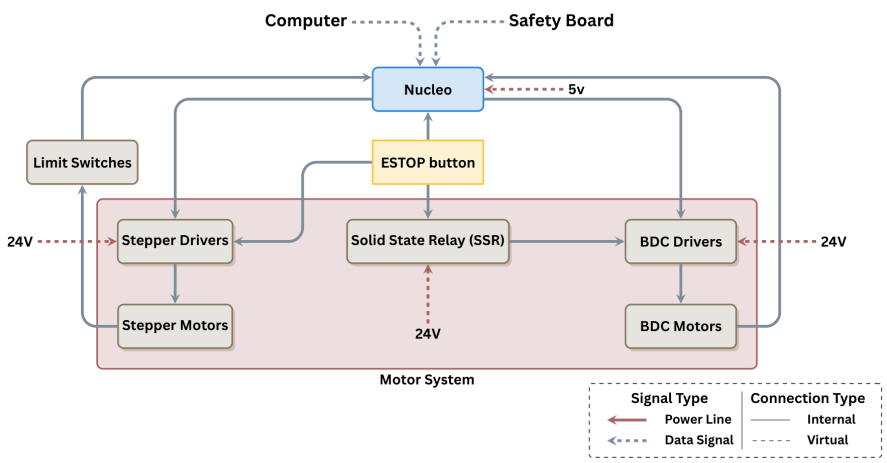

Figure 6: Drive Control System

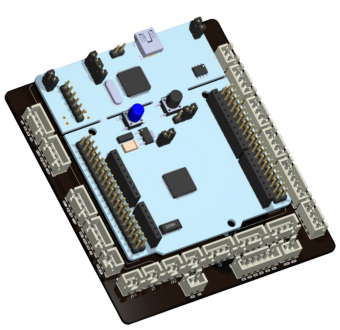

Figure 7: Motor Control PCB

4.4 Safety Board/System

The safety board is implemented on the STM32F401CCU6 microcontroller. The board operates independently of the main compute device and the motor control system to ensure smooth handling of safety-critical functions. The safety board is responsible for wireless E-Stop, safety lights, and monitoring of the rotary switch for mode selections and transitions. Loss of communication or invalid mode states default the system into IDLE mode for a safe stop condition.

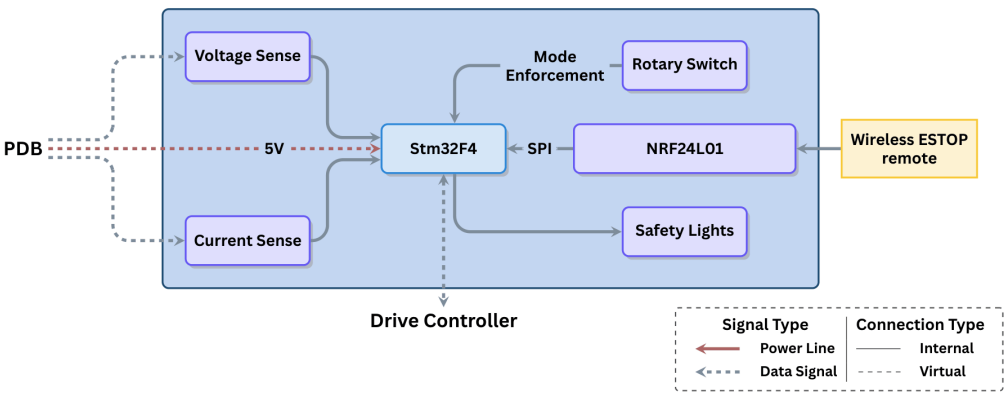

Figure 8: Safety Board Diagram

4.5 Mode Architecture

The vehicle operates using a mode-based architecture to ensure that particular actions are only permitted under certain conditions and that transitions between states are controlled and safe. Below are the defined modes and their use cases.

Mode	Description	Key Behaviour	Entry Condition	Exit Condition
ESTOP	Highest priority state with all motion disabled	• Motor power cut via hardware cutoff • Drive commands forced to zero • Stepper motion preempted • Steering aligned to zero • Command queues cleared	• E-Stop trigger • Communication loss • Invalid instructions • Invalid mode selection	• Manual reset • Valid mode selection
HOMING	Initialization mode used to obtain absolute steering reference	• Limit switch based zeroing • Offset correction • Independent calibration of all 4 modules	• System startup • Reset after fault	Completion of homing for all modules→IDLE
IDLE	System is powered, initialized, and homed, but not executing motion	No motion commands executed	Successful homing	Transition to TELEOP, AUTONAV, or SELF-DRIVE
TELEOP	Manual control mode	Executes user commands via joystick/keyboard interface	Valid mode selection	• Mode switch • E-Stop trigger • Command loss
AUTONAV	Autonomous navigation	Executes autonomously generated navigation commands	Valid mode selection post completed homing	• Mode switch • E-Stop trigger • Command loss
SELF-DRIVE	Advanced autonomous mode	Handles self-drive scenarios (parking, pedestrians, signs, etc.)	Valid mode selection post completed homing	• Mode switch • E-Stop trigger • Command loss

Table 6: System Operating Modes

4.6 Power System and Performance

The system uses a power distribution board supplying multiple voltage rails derived from a 6S LiPo battery. An onboard 24 V DC to 220 V AC inverter is used to power the LiDAR. A USB-C PD power management module powers and charges the computing device.

- Nominal current consumption: ∼34 A
- Peak current capability: ∼100 A (within PDB limits)
- Recharge time: ∼ 40 minutes (20% to 100%)
- Runtime: 45 - 90 minutes

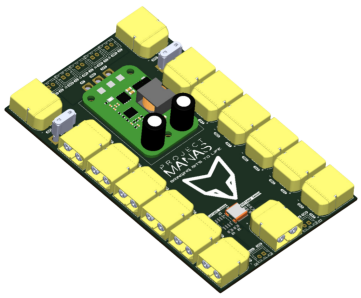

Figure 9: ALYX power distribution board

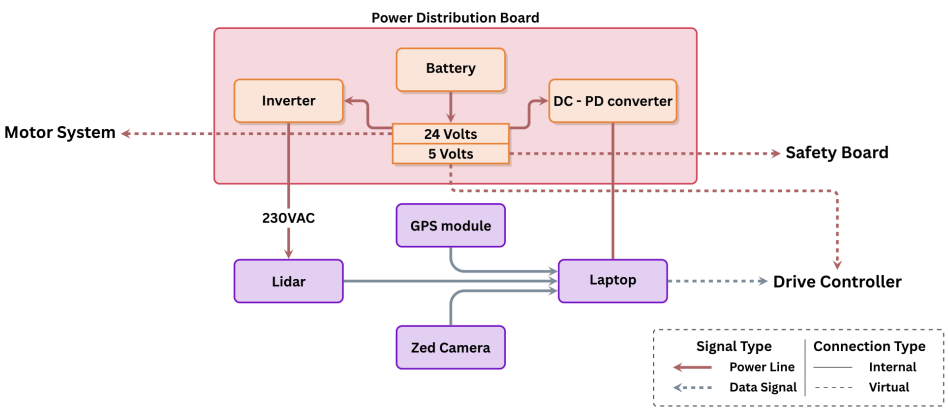

Figure 10: Power Distribution on ALYX

4.7 Key Design Decisions

4.7.1 Separation of Safety and Control

Instead of allowing the laptop to control all subsystems, safety functions are offloaded to the Safety Board. This ensures that the emergency stop remains functional even if high-level systems fail. The mode control is enforced at the hardware level through a rotary switch.

4.7.2 Multi-Rail Power Distribution

The system uses multiple regulated voltage rails:

- 24 V for high-power loads such as brushed DC motors and stepper motors
- 220 VAC Inverter for LiDAR
- 5 V for logic, sensors, and control electronics

This reduces noise coupling between subsystems and prevents cascading failures.

4.7.3 SSR-Based Power Isolation

Motor power is controlled through solid-state relays (SSR), allowing immediate shutdown during emergency stop and electrical isolation between power and control subsystems.

4.7.4 Dedicated Real-Time Controller

The Motor Control microcontroller is responsible for driving the wheels and the steering modules using PWM and Step signal generation. A closed-loop system has been implemented for the wheels for rapid and reliable control. The microcontroller also handles the homing (zero-calibration) for the steer modules.

4.8 Requirement Validation

The dedicated drive controller architecture, along with DMA-based communication, reduces interrupt overhead and assists in fulfilling the **Control Loop Frequency** requirement by maintaining a stable 100 Hz control loop with low timing jitter. Encoder-based closed-loop control and filtered sensor feedback

help satisfy the **Drive Velocity Error** requirement by achieving velocity and angular velocity errors well below the specified limits.

Metric	Target Value	Measured Value	Status
Control loop frequency	$\geq$100 Hz, $< 1\,\mu s$ jitter	100 Hz, ~ $480\,ns$ jitter (peak)	Passed
Drive Velocity error	$\leq$0.05 m/s, $\leq$0.05 rad/s	$<$0.04 m/s, $<$0.01 rad/s	Passed

Table 7: Electrical, Control, and Motion Performance Validation

5. PERCEPTION

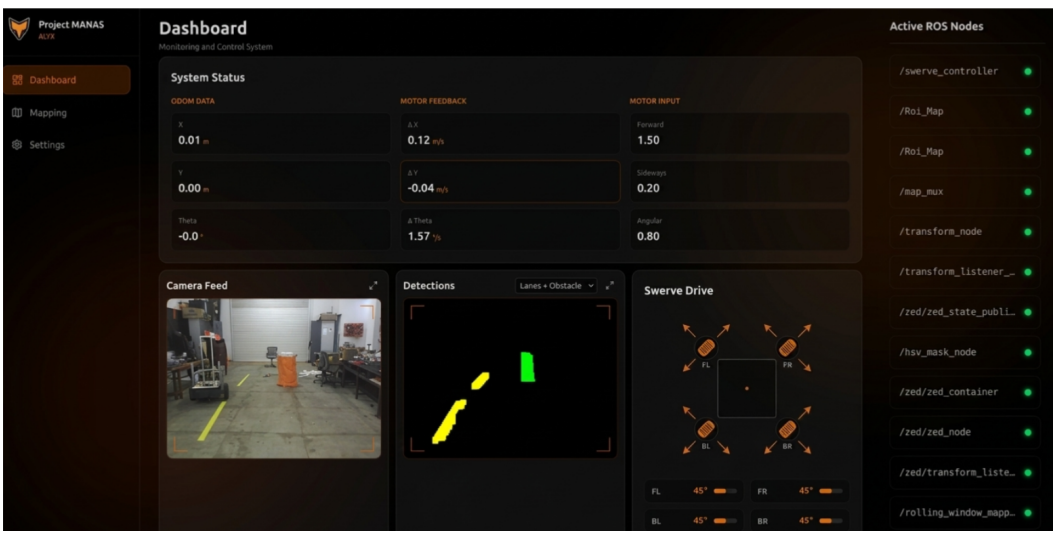

Figure 11: System Dashboard & GUI

5.1 Sensor Data Processing and Internal Representation

The course is internally modeled as a dynamic, probabilistic log-odds occupancy grid. As the robot moves, ZED2i 3D point clouds are transformed into the local `base_link` frame, while a segmentation model produces lane masks from RGB images.

These lane detections are projected onto a 2D grid using Bayesian log-odds updates instead of binary mapping. Each cell's probabilistic occupancy increases with lane detections and decreases when free, reducing the impact of noise and false positives.

For efficiency, the map operates as a rolling window centered on the robot. The grid shifts using fast bitwise operations as the vehicle moves, preventing memory growth and keeping the map lightweight and low-latency.

5.2 AutoNav Perception Pipeline

The AutoNav perception stack heavily relies on the ZED2i camera and Ouster OS1 LiDAR to accurately identify course features.

5.2.1 Lane Identification

The RGB images from the ZED2i camera are passed through a custom Deep Learning pipeline to segment accurate lane masks. To tackle temporal and environmental variations, the model is trained on a diverse set of captures annotated using SAM3 (Segment Anything Model) with strict manual oversight.

A U-Net architecture with a MobileNetV2 encoder is used for efficient, real-time inference on the onboard device. The network outputs a binary lane mask, which is fused with the transformed ZED2i point cloud to project lanes onto the occupancy grid.

5.2.2 Obstacle Detection

Obstacle detection uses Ouster OS1 LiDAR point clouds. Distance and height thresholding on the point cloud isolates barrels and vertical objects, avoiding deep learning methods while reducing latency and compute. Detections are overlaid on the previously obtained lane map, and both obstacles and lanes are inflated by the robot's physical radius to add a safety buffer before passing to the navigation module.

5.3 Self-Drive Perception Pipeline

The Self-Drive pipeline extends the perception framework to handle structured urban scenarios by leveraging LiDAR-driven geometry and further occupancy-grid processing.

5.3.1 Object Detection and Classification

Obstacle detection is performed using LiDAR point clouds in conjunction with a YOLO model trained on publicly available images and fine tuned using carefully curated images to classify tires, barrels, and mannequins. Ground points are removed from the LiDAR data to account for sloped terrain, after which the point cloud is clustered into obstacles to estimate their locations. These coordinates are then cross-referenced with corresponding bounding box coordinates from the ZED2i's RGB-D images, enabling YOLO-based obstacle classification.

5.3.2 Object Localization

The obstacle detector provides the 3D location of the obstacle. Perpendicular boundaries generated from the line fitted to the yellow centerline are then used to determine whether the obstacle lies within the lane boundaries and belongs to the same lane as the robot.

5.3.3 Intersection Detection

Intersections are detected by identifying a distinct perpendicular line on the lane map of yellow and white lanes. The system scans specific sections of the map to extract features unique to these lines.

5.3.4 Parking Slot Detection

Pull-in and parallel parking slots are detected by identifying pairs of white lane lines that are parallel to each other and are perpendicular to the robot's heading. Pull-out slots are detected by identifying white lane lines that are parallel to each other and have the robot present in between them.

5.3.5 Pothole Detection

Potholes are detected by identifying clusters of high-intensity pixels on the lane map that meet specific criteria for size, shape, and circularity.

5.3.6 Sign Detection and Rules of the Road

For stop signs, a custom Optical Character Recognition (OCR) model detects and reads sign text, triggering the driving logic to execute a controlled stop when a "STOP" sign is recognized. Detected obstacles are further filtered using lane information to ensure only obstacles within the drivable corridor manipulate the planner.

5.4 Requirement Validation

The use of an efficient, robust deep-learning-based lane masking solution directly addresses our **Reliable Lane Segmentation** requirement, ensuring reliable perception under unpredictable lighting conditions. The probabilistic log-odds mapping satisfies the **Efficient Mapping** requirement by providing a lightweight, rapidly updatable structure that mitigates sensor noise.

Metric	**Target Value**	**Measured Value**	**Status**
Model inference speed	$\leq$50 ms	34 ms	Passed
Lane segmentation accuracy(IoU)	$\geq$0.8	0.87	Passed
Map update latency	$\leq$50 ms	38 ms	Passed

Table 8: AutoNav Perception Performance

The combination of LiDAR geometric clustering and YOLO-based visual validation meets the **Object Tracking** and **Reliable Occupancy Grids** requirements, aiming to keep false-positive and false-negative rates $\leq 5\%$. The dedicated OCR pipeline fulfills the **Sign Recognition** requirement.

Metric	Target Value	Measured Value	Status
False positive rate	$\leq 5\%$	3.2%	Passed
False negative rate	$\leq 5\%$	4.1%	Passed
Object classification accuracy	$\geq$90%	92%	Passed
STOP sign OCR accuracy	$\geq$90%	95% for <= 5 m and 80% for > 5 m	Satisfactory
State estimation accuracy	$\geq$95%	93%	Satisfactory

Table 9: Self-Drive Perception Performance

6. DRIVING LOGIC

6.1 Control & Kinematics

The robot employs a custom ROS2 node that implements the kinematic controller of the holonomic drive. This module acts as the interface between high-level motion commands generated by the navigation stack and the low-level motor actuation handled by the embedded system. The controller receives the desired robot velocities from the high-level navigation controller via the `/cmd_vel` topic (`geometry_msgs/Twist`). Using these velocities, the controller performs inverse kinematic transformations for a four-wheel holonomic configuration. Based on the robot's geometric parameters, such as its wheelbase and track width, it computes for each wheel module the required wheel rotational velocity (RPM) and steering angle. These computations enable holonomic motion, allowing simultaneous translation and rotation. The computed wheel speeds and steering angles are sent to the motor control system, which drives the wheel motors and steering mechanisms.

6.2 Navigation and Localization Strategy

Replacing the previous Direct LiDAR Odometry (DLO) framework, the current localization stack utilizes Direct LiDAR-Inertial Odometry (DLIO) to tightly couple dense 3D LiDAR point cloud(s) with IMU data, which efficiently deskews motion distortion caused by irregular terrain. DLIO provides a higher-quality initial odometry estimate, reducing drift compared to DLO, while the Dual EKF fuses additional sensor data and improves state consistency.

6.3 AutoNav Driving Logic

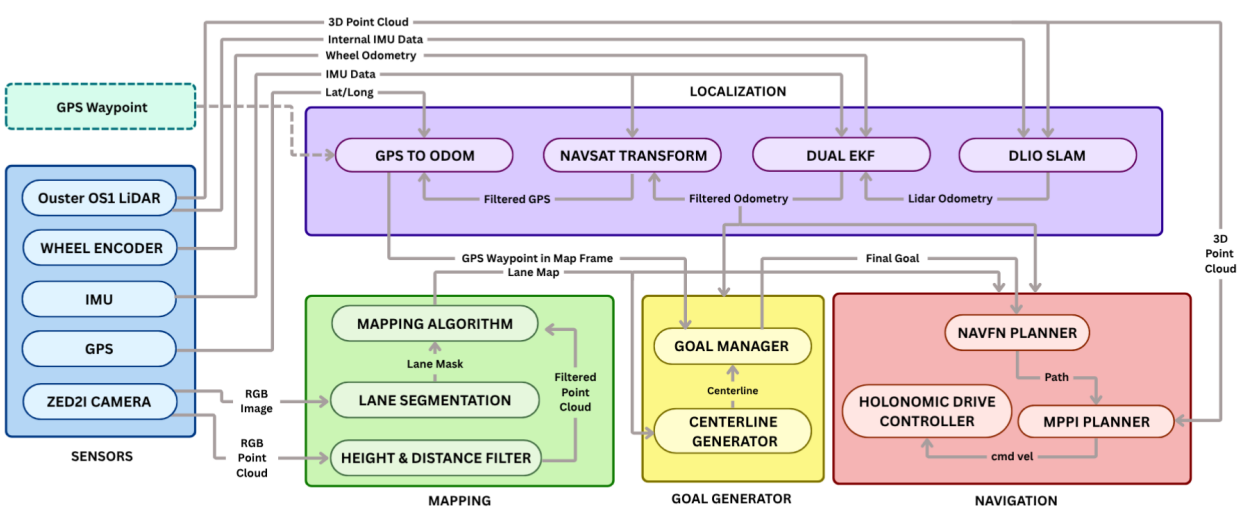

Figure 12: Overview of AutoNav Stack

6.3.1 Centerline Generation

The occupancy grid map is utilized by a custom ray marching method for intermediate goal generation. This algorithm generates a Signed Distance Field (SDF) of the map and extracts a drivable centerline using beam search by propagating rays forward through the map according to a custom heuristic function.

6.3.2 Goal Management and Waypoint Navigation

The system uses a dynamic state machine to switch between two navigation modes: Lane following and GPS waypoint navigation. During normal operation, the goal manager generates intermediate targets from the lane centerline. When lane markings are lost in “no-man’s land,” the system automatically switches to GPS mode, where the robot follows a sequence of global waypoints to continue forward motion until visual lanes are detected again.

For GPS-based navigation, a custom ROS2 C++ node converts raw GPS data (`sensor_msgs/msg/NavSatFix`) into odometry (`nav_msgs/msg/Odometry`). It also applies a dynamic yaw correction to align the GPS reference frame with the robot’s local map frame. This conversion allows the robot to treat GPS positions as usable odometry, enabling consistent tracking relative to global waypoints.

6.3.3 Path and Motion Planning

NavFN calculates the shortest global paths while the MPPI Local Planner handles dynamic constraints and obstacle avoidance maneuvering within the generated local trajectories.

6.4 Self-Drive Driving Logic

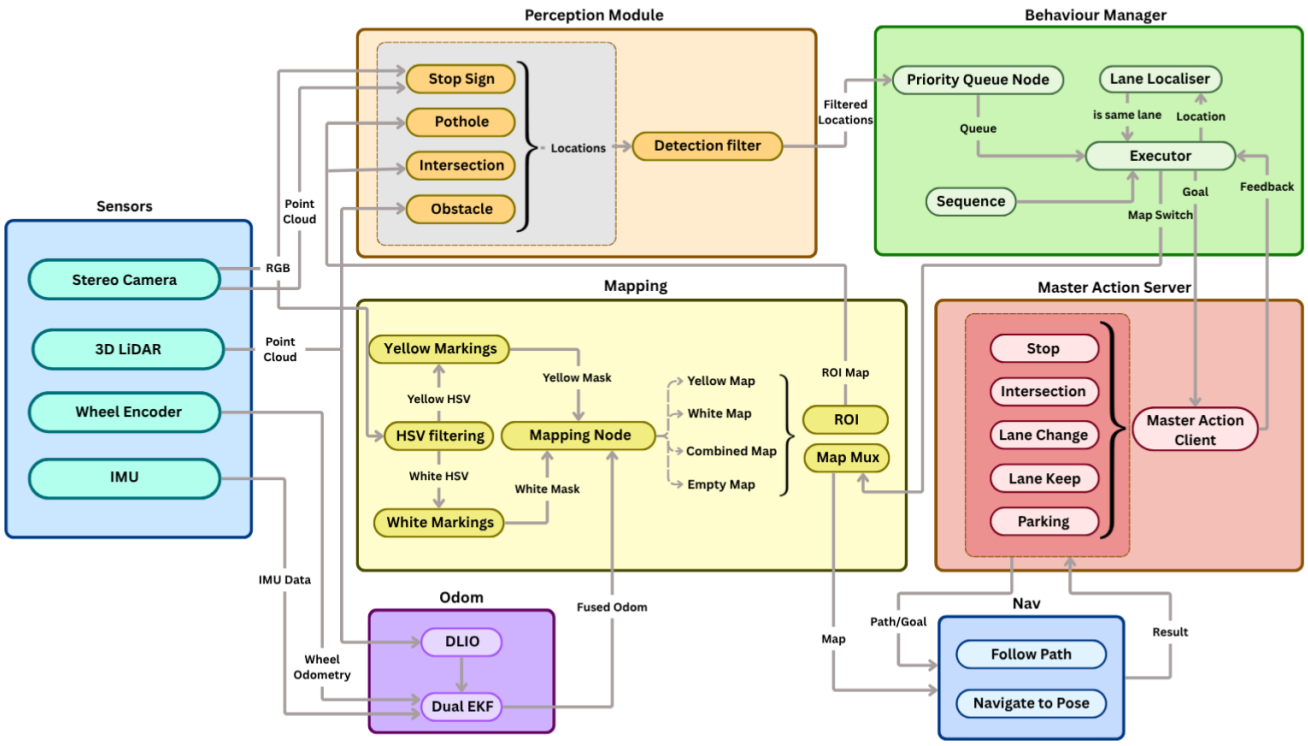

Figure 13: Overview of Self-Drive Stack

6.4.1 Behaviour Manager

The self-driving behavior manager determines vehicle actions using detections from the perception modules, along with their relative positions. To improve stability, detections are first processed through a filtered object localization module. Detected entities are then assigned priority levels, with pedestrians and mannequins having the highest priority, followed by stop signs and intersections, and then parking slots, barrels, tires, and potholes. Only objects within the robot’s current lane are considered.

Using obstacle priority and distance, the behavior manager maintains a priority queue of active detections. The highest-priority object is continuously evaluated against predefined action thresholds that account for braking distance and controller response time. If no object crosses its threshold, the robot continues lane following; otherwise, the corresponding maneuver, such as stopping, lane changing, intersection handling, or parking, is executed.

The behavior manager also serves as the system’s primary fail-safe by monitoring communication with action servers and reissuing failed actions until they succeed or the triggering percept is no longer detected.

6.4.2 Intersection Actions

Right turn: Refer to figure 14. The map is skeletonized to remove any noise, and yellow lanes are identified using RANSAC. The intersection of the two lines is the intersection point P3. An offset is taken from the yellow lanes and the intersection of those lines is P4. Using the equation of the yellow line, P5 is calculated. Using the vehicle's current position as the start, a path is generated using splines that passes through P4 and P5.
Left turn: Similar to the right turn, here the offset line parallel to the further yellow lane is taken. Giving points P1, P2 and P4. Using the bot's position and these three points, a path is generated.

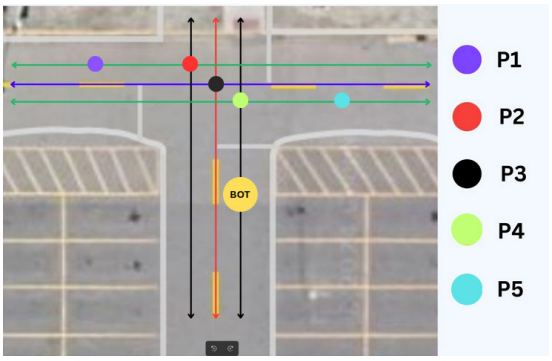

Figure 14: Intersection Actions

Straight: Using the same approach, here we use the bot's position and P4 to generate a straight path into the other side of the intersection.

6.4.3 Parking

Pull-In: After the pull-in slot is detected, a waypoint, P1, is published at the entry of the slot, and a waypoint, P2, is published at the center of the slot. A path is interpolated between these waypoints and published. The slot center waypoint's orientation is perpendicular to the bot's initial heading - the perpendicularity direction is decided based on whether the parking is a right turn or left turn pull-in. This ensures correct final alignment with the bot facing into the slot.

Pull-Out: The robot drives straight along its initial heading till it exits the slot at point P1 and then turns right or left depending on whether the test is a right or left pull-out. After this turn, the robot keeps driving straight ahead till a barrel is detected, at which point the stop action is triggered, and the bot stops at point P2.

Parallel: After the slot is detected, waypoints are published at the slot entry (point P1) and center (point P2), and a path is interpolated between them. The robot drives straight till it is ahead of the slot, and then reverses into the slot by following the path calculated. The slot center waypoint's orientation is the same as the initial heading of the robot.

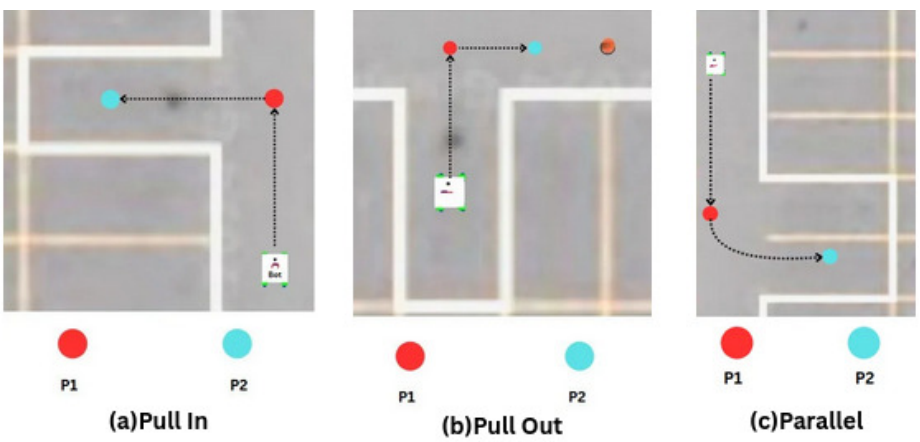

Figure 15: Parking Actions

6.4.4 Lane Following

From the filtered lane pixels, small linear functions are repeatedly fitted over consecutive sections of the detected lane markings to approximate the overall road centerline. This estimated centerline is then laterally shifted to generate adjacent lane boundaries and lane trajectories, allowing the system to track the robot's current lane. Waypoints are subsequently sampled along the selected lane trajectory and published to the Nav2 stack for path planning and execution.

6.4.5 Obstacle Avoidance

For lane-change maneuvers, the system uses precomputed lane offsets representing parallel trajectories adjacent to the current lane. A target point is selected at a fixed lookahead distance along the desired adjacent lane and sent to the navigation stack as a goal. Once the robot reaches this target, control is returned to the lane-following module, and the new lane becomes the active trajectory.

6.5 Requirement Validation

The implementation of reliable lane centerline generation directly addresses the **Consistent Course Progress** requirement by ensuring the vehicle maintains forward momentum even in heavily occluded scenarios. Furthermore, our well-tuned and tested MPPI planner helps satisfy the **Agile Traversal and Obstacle Clearance** requirement, successfully maintaining a safe buffer from obstacles during high-speed navigation.

Metric	Target Value	Measured Value	Status
Minimum average speed	$\geq$0.45 m/s	0.56 m/s	Passed
Average speed	$\geq$1.2 m/s	1.34 m/s	Passed
Obstacle clearance	$\geq$0.2 m	0.28 m	Passed

Table 10: AutoNav Driving Logic Performance

In urban scenarios, the **Stable Lane Following and Obstacle Avoidance** requirement is met via a proactive algorithm that smoothly steers the robot into pre-calculated adjacent lanes to bypass hazards (barrels, tires, pedestrians) without stopping. Additionally, the **Complex Maneuver Execution** requirement is satisfied using waypoint-driven logic to accurately park and navigate intersections.

Metric	Target Value	Measured Value	Status
Lane transition success rate	$\geq$95%	96%	Passed
Maximum path deviation	$\pm$0.2 m	$\pm$0.24 m	Satisfactory
Parking success rate	$\geq$90%	91%	Passed
Final parking pose error	$\pm$0.05m	$\pm$0.1 m	Satisfactory
Intersection maneuver success	$\geq$90%	94%	Passed

Table 11: Self-Drive Driving Logic Performance

7. KEY PERFORMANCE INDICATORS

7.1 AutoNav Challenge KPIs

The following KPIs are selected to assess the efficiency and reliability of the autonomous navigation stack throughout the AutoNav Challenge.

- **Course Completion Time** measures the elapsed time from the start of autonomous operation to reaching the goal using system timestamps, targeting minimal traversal time while operating under the 5 mph speed limit.
- **Average Velocity** measures the mean speed sustained over the course via motor feedback, ensuring consistent operation near the maximum allowed speed without compromising safety.
- **Obstacle Collision and Lane Overflow** tracks any contact with obstacles or excursions beyond course boundaries, with a target of zero collisions and zero boundary violations per run.
- **Waypoint Localization Error** measures the residual distance between the vehicle's stopping position and each target GPS waypoint, reflecting the combined accuracy of the onboard odometry and GPS sensor in resolving the goal location.
- **System Failure Rate** measures the number of times the vehicle enters an unrecoverable state requiring human intervention per run, targeting zero failures to ensure sustained autonomous operation throughout the course.

7.2 Self-Drive Challenge KPIs

The following KPIs are selected to assess both efficiency and safety, ensuring the vehicle maintains forward progress while handling real-world driving scenarios.

- **Average Speed** measures the vehicle's velocity over the entire run via odometry, ensuring it consistently exceeds the minimum speed stipulated in the rules.
- **Response Distance** defines thresholds for stopping, slowing, and turning towards scenarios such as stop signs, static and dynamic obstacles, and intersections, ensuring the vehicle responds safely within the available space.
- **Lane Keeping Consistency** measures the percentage of run time the vehicle remains within lane boundaries, capturing sustained path adherence over the full course.

- **False Positive Rate** measures the frequency of incorrectly detected obstacles in the absence of a valid obstacle, targeting minimal spurious responses that impede forward progress.
- **Mission Completion Rate** measures the percentage of the course navigated without human intervention, a software fault, or an emergency stop, indicating overall system robustness.

These KPIs collectively assess both efficiency and safety, ensuring the vehicle maintains forward progress while handling real-world driving scenarios.

7.3 Requirement Validation

Metric	Target Value	Measured Value	Status
Course Completion Time	$\leq$ 180 s	168 s	Passed
Average Velocity	$\geq$1.2 m/s	1.35 m/s	Passed
Obstacle Collisions	0	0	Passed
Lane/Course Boundary Violations	0	0	Passed
Waypoint Localisation Error	$\leq$ 1.5 m	1.23 m	Passed
System Failure Rate	$\leq$ 5%	2%	Passed

Table 12: AutoNav Key Performance Indicators

Metric	Target Value	Measured Value	Status
Average Speed	$\leq$1.8 m/s	2.0 m/s	Satisfactory
Response Distance	$\leq$0.1 m	0.04 m	Passed
Lane Keeping Consistency	$\geq$95%	100%	Passed
False Positive Rate	$\leq$20%	5%	Passed
Mission Complete Rate	$\geq$80%	92%	Passed

Table 13: Self-Drive Key Performance Metrics

8. ANALYSIS OF COMPLETE VEHICLE

8.1 Lessons learned during construction and system integration

Over the course of 10 months, our team encountered various issues, resolving which led to a robust and reliable system. On the hardware side, major issues were encountered with wheel stability due to custom-made components and with flange-clamp alignment, as the clamp held the entire lower assembly (which houses the drive motor and wheels). This was resolved using a machining fixture and a CNC-machined flange clamp, respectively.

On the software side, the biggest challenge was sim-to-real transfer and sensor stack integration. This included issues with obstacle detection at high speeds, precise timing of behavior manager triggers given strict latency constraints, and MPPI parameter tuning. Additionally, the initial versions of the software stack were created under the assumption that the world was a perfectly flat plane. Through iterative testing and refinement, this was modified to account for the sloped terrain as expected in the real world.

8.2 Mechanical and Electrical Failures and their Mitigation

- ZED2i camera vibrations were mitigated with carbon fiber rod bracing.
- Steering angle inaccuracies were solved by replacing software-based pulse counting with hardware-timed pulse generation.
- Stepper motor stalling under steering load was resolved using a 5:1 planetary gearbox.

8.3 Simulation Testing

The holonomic drive controller was tested in the Gazebo simulation environment before deployment on the physical robot. Real-world mechanical constraints such as robot mass, wheel dimensions, and limit switch placement were considered to closely replicate actual operating conditions. Different motion scenarios were tested to verify wheel coordination, velocity tracking, and overall controller stability before implementation on the real bot.

Gazebo also served as the primary evaluation platform for the AutoNav and Self-Drive software stack. The simulation fully replicates the robot's dynamics and sensor stack providing a safe and controlled virtual testbed. Development was managed through GitHub, where the team adopted a branch-based workflow—isolating features and fixes into dedicated branches to minimize interference between workstreams and streamline bug identification. ROS bags, which are used to store sensor messages data, robot state, and command topics, were extensively used to recreate bugs and resolve issues. Low-level debugging was carried out using GDB, allowing precise inspection of runtime behavior.

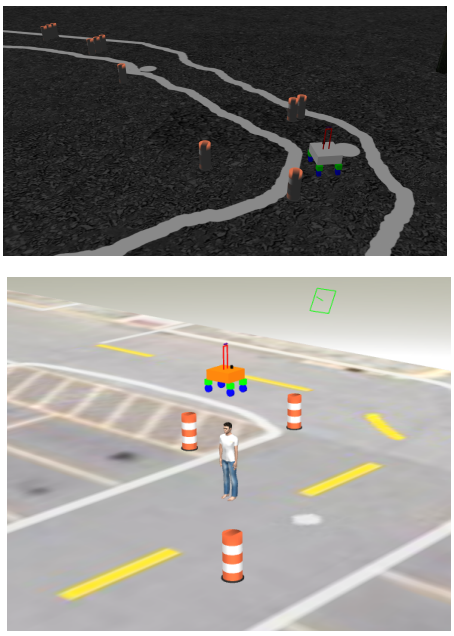

Figure 16: Simulation Environments

8.4 Physical Testing to Date

Physical testing was conducted to validate ALYX's key systems. Stepper motors were tested to confirm they provide the torque required to turn the wheel assemblies, and BDC motors were validated for sufficient RPM and torque to drive the bot smoothly. TPU tires were verified to withstand the full operational load without failure. Microcontroller communication and steering module pulse generation were tested using a logic analyzer, confirming signal reliability and precise pulse output.

9. CYBER SECURITY ANALYSIS

Modern robotic and human-driven vehicles are interconnected via the network to large amounts of software from multiple vendors, and are an inviting target for cyber-attacks. The table below identifies three key vulnerabilities present in ALYX, scored across the CIA triad using a Low / Moderate / High impact scale.

Module	Threat Description	Confidentiality	Integrity	Availability
Remote E-Stop	The E-Stop link operates on a fixed 2.4 GHz channel with no message authentication. A replay attack could trigger an unintended halt, while a sustained jammer could prevent a genuine kill command from reaching the safety board.	Low	High	High
ROS 2 DDS Middleware	DDS Security is disabled by default, so any node on the same `ROS_DOMAIN_ID` can publish to `/cmd_vel` without authentication. A rogue actor on the same network could inject motion commands that bypass the Safety Board entirely.	Moderate	High	High
Software Supply Chain	Third-party packages installed via `pip` and `rosdep` (Nav2, Patchwork++, etc.) are not version-pinned or scanned for CVEs. A compromised dependency running on the compute device has a direct path to influencing vehicle motion.	High	High	Moderate

Table 14: Analysis of cyber vulnerabilities

9.1 Hardening Before Series Production

9.1.1 Remote E-Stop

The NRF24L01 link currently has no protection against replayed or suppressed packets. Each transmitted packet should carry an HMAC and a monotonically incrementing sequence number so that the safety board can reject replayed or out-of-order messages. A heartbeat timeout should also be enforced: if no valid packet arrives within a set window, the system must default to ESTOP rather than continue running. To reduce jamming risk, frequency-hopping across the module's available channels should be used. For a production deployment, switching to a safety-rated wireless protocol with built-in redundancy would be the appropriate long-term step.

9.1.2 ROS 2 DDS Middleware

With DDS Security off, the `/cmd_vel` topic that drives ALYX's kinematic controller is open to anyone on the same network. SROS2 should be enabled, with per-node identity certificates and permissions files that explicitly restrict which nodes can publish or subscribe to motion-critical topics like `/cmd_vel` and `/goal_pose`. The `ROS_DOMAIN_ID` should be bound to a private network interface rather than a shared one. As an additional check, the UART bridge to the Motor Control Board should validate that incoming velocity commands fall within physically plausible limits.

9.1.3 Software Supply Chain

With no version pinning or vulnerability scanning in place, a compromised upstream package could silently affect the autonomy stack. All `pip` dependencies should be pinned with SHA-256 hashes in `requirements.txt`, and `rosdep` packages should be pulled from a controlled internal mirror rather than directly from upstream. A CVE scanner like `pip-audit` should run in CI and block any build with a dependency scoring above a set CVSS threshold. The onboard ROS 2 workspace should run from a read-only filesystem image, updated only through a signed release process.