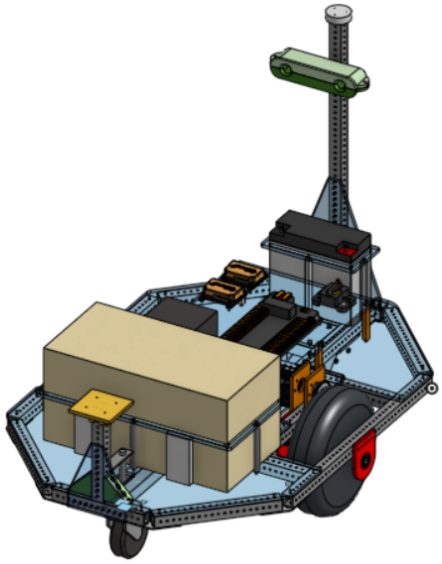
RPI Robotics Club - "Tin Shady"



Intelligent Ground Vehicle Competition AutoNav 2026

Team Lead: Zane Brotherton (brothz@rpi.edu)

Faculty Advisor: Prof. Kimberly Oakes (oakesk2@rpi.edu)

Last modified: May 15, 2026

Autonomy	Simulation	Hardware	Electrical	Mechanical
Gavin Lesko	**Mathew Davis**	**Tag Ciccone**	**Jerry Norton**	**Bonden Tucker**
Ashvin Ganesan	Vincent Borello	Titan Spellos	Raj Vishnu	Benjamin Eyre
Sasha Murokh	Bryce Haddock		James Qiu	Richard Weng
Camden Laursen-Carr				
Pradeep Giri				
Rosanna Lu				
Vincenzo Rapacciuolo				

Contents

1 | System and Subsystem Requirements

1.1 | Engineering Requirement Analysis

Requirements were derived by decomposing the given ruleset objective constraints and strategic constraints. Objective constraints refer to mandatory items, such as safety rules, or other rules given a strict heuristic and not left up to interpretation, such as vehicle dimensions, e-stop behavior, payload retention, speed limits, lane following, obstacle avoidance, and waypoint navigation. Strategic constraints are our extrapolations from those rules, and all open-ended design questions posed within the document; essentially, how our robot could be designed for the best competitive chance possible. These were then allocated to mechanical, safety, electrical, perception, driving logic, and KPI subsystems. Each requirement was assigned a measurable target so that design decisions could be validated through bench tests, simulation, and outdoor course testing.

1.2 | Requirements

The table of requirements is enumerated below, with any applicable key performance indicators listed with each requirement.

A) The perception system must detect left and right lane boundaries during autonomous operation.
   - 95% usable lane detections at 10 Hz

B) The perception system shall detect obstacles in the robot's path early enough for avoidance.
   - Detect obstacles >= 2.0 meters away with a 95% detection rate

C) The driving logic shall generate goal points that keep the robot centered between detected lane boundaries.
   - Average centerline error <0.5m and 0 full boundary crossings

D) The driving logic shall adjust the robot's path to avoid detected obstacles while continuing toward the next goal or waypoint.
   - 0 collisions, >0.5m clearance from obstacles

E) The physical frame of the robot shall be greater than a minimum size requirement of two feet wide by three feet long.

F) The robot shall be able to transport a 20-pound payload with the approximate shape and size of a 16in x 8in x 8in cinder block.

G) The robot shall have a wireless emergency stop system that must effectively bring the robot to an immediate stop, activated over a minimum of 100 feet.
   - Emergency stop signal reception >= 100 feet

H) The robot shall be sufficiently protected from harsh outdoor conditions.

I) The robot shall have sufficient failsafes and breakers for overcurrent and overheating.
   - Breaker shutoff in 5s when current through a motor channel >= 80 Amps

J) The robot battery shall provide sufficient usable energy for at least one 6-minute AutoNav run under expected competition loading.

- Robot maintains capability to drive when driven at full speed for over 6 minutes.

K) The robot shall be able to travel in a straight line above the given minimum speed limit.

- Complete first 44 ft above 1 mph.

L) The robot shall be able to travel through the course without disqualification.

- Complete a representative 500 ft course within 6 minutes without collision or boundary exit.

2 | Mechanical Design

2.1 | Overview and Design Philosophy

As a rookie team, our goal for the robot was to keep it simple and robust, while allowing room for changes to the robot as we learn and discover. To get an idea of what makes a good IGVC robot, we looked at previous years for common winning trends between teams. We identified that being small and maneuverable were key elements in being successful. We modeled our entire robot in CAD before the manufacturing process in order to allow for fluid changes and continuity between revisions. We used OnShape in order for easy collaboration and organization through the cloud. Our robot is 24.849"x37.099" which fills requirement E on our requirement table, as well as satisfying the rules. In CAD, we divide the robot up into subsystems each with their own bounding boxes. These boxes allow for clear space restrictions while collaborating, ensuring that no two people create conflicting subsystems and improves our overall design efficiency.

2.2 | Bill of Materials

Below is an abridged bill of materials that includes all major mechanical costs.

Component	Quantity	Total Cost (USD)	Club Cost (USD)
Aluminum	$\approx 350in$	$121	$121
Polycarbonate	$\approx 3000in^2$	$110	0
Hardware and Fasteners	1	$50	$50
Gearbox Assembly	2	$230	$230
Motors	2	$85	$85
Motor Controllers	2	$336	$336
Wheels	4	$71	$71

2.3 | Chassis and Frame

The chassis is made out of aluminum in order to be lightweight and still maintain a rigid structure. We made the decision to cut the corners off our robot in order to reduce our footprint while turning, which we think will give us an advantage while running the course. We use a mix of pre-punched aluminum tubing and t-slot tubing to give our robot plenty of room for adjustments while still retaining precise and strong joints. Polycarbonate was used to create platforms on top of the aluminum frame for us to mount electronics, sensors, and more. We chose this material because it is a familiar material, we already had it

on hand and would allow us to work within our limited budget. The payload is retained through multiple aluminum standoffs and a polycarbonate ring that ensures we have 5 points of contact at all times, and satisfies requirement F on the requirement table.

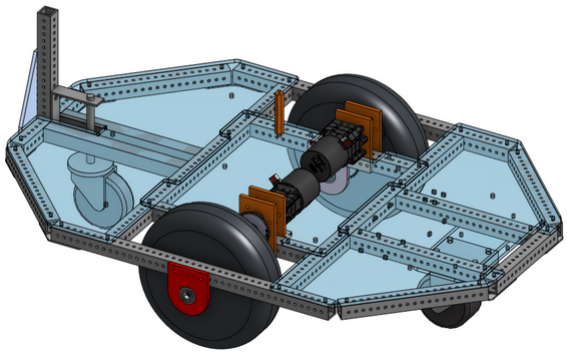


Figure 2.1: Enter Caption

2.4 | Drive System

We are using a differential drive system because it keeps our drive system simple and robust with very few moving parts. Our wheels are powered by 2 brushless motors on a 25:1 gear reduction and connected via sprocket and chain. This system allows our 2 main wheels to bear most of the weight which is best for our weight distribution, especially when going across obstacles like the ramp. The chain allows us to put our motors above the frame which helps to keep them protected from obstacles, while still delivering power to our wheels.

2.5 | Weatherproofing and Structural Robustness

To ensure our structure was robust, we used sets of gussets and screws for easy maintenance. By checking joints periodically, we can ensure that we are always maintaining a rigid frame, and quickly replace damaged parts. This allowed us to keep our cost down, and allowed us to work with more familiar techniques as opposed to other joining methods like welding. A variety of 3D printed mounts for sensor and electronics ensure that we get secure mounting beyond what is provided by the supplier and is tailored to our needs.

3 | Safety

3.1 | Transport and Storage Safety

When the robot is not in use and in storage, it is kept in a secure and dry location where it cannot move. The battery and main breaker are disconnected and the battery is stored in a dry and dark place. During transport, to prevent the robot from rolling and causing injuries, it is secured on wooden blocks and strapped down tightly to prevent any movement.

3.2 | Emergency Stop System

Tin Shady uses both a mechanical emergency stop and a wireless emergency stop. The mechanical e-stop is mounted at the rear center of the vehicle within the required height range and is positioned so that it

can be reached safely from behind the robot. The button is wired into the drivetrain disable path so that pressing it disables motor output through the ODrive controllers.
The wireless e-stop uses an HC-12 radio link connected to an ESP32-based safety controller. This safety controller is separate from the Jetson, ROS 2 autonomy stack, and Nav2 control software. When a stop command is received, or if the wireless signal is lost, the safety controller drives a GPIO disable signal to the ODrive motor controllers. The ODrives then disable motor output and use their electrical braking behavior to bring the drivetrain to a stop. This does not cut battery power to the robot, but it disables commanded drivetrain motion independently of the primary autonomy computer.
This architecture was selected to keep the e-stop behavior separate from high-level autonomy software. A failure of ROS 2, Nav2, perception, or the Jetson should not prevent the dedicated safety controller from disabling drivetrain output. Final validation will confirm wireless range, signal-loss behavior, and stopping behavior during low-speed drivetrain tests.

3.3 | Safety Light

A strip of addressable LEDs wound around a cylinder are used for our indicator light. During normal powered operation, the light displays a solid state. When the robot enters autonomous mode, the light changes to a flashing state to clearly indicate that the vehicle may move under autonomous control. When autonomous operation ends, the light returns to solid. This behavior directly supports competition safety requirements and also gives team members a quick visual indication of the robot's control state during testing.

3.4 | Sharp Edges, Exposed Components, Loose Connections

Careful sanding, polishing and deburring was performed on the robot to remove any and all sharp edges. The majority of exposed components on the robot are rated IP67. Water sensitive components are shielded by a 3D printed enclosure, printed in PETG and finished with an acrylic coating. Wiring was scrutinized for any loose or incorrect connections before any field tests or connecting to power.

3.5 | Requirement Adherence

Safety requirements were addressed by designing the robot around hardware-enforced stopping, visible operating-state indication, and conservative handling of exposed components. Requirement G is addressed through the wireless e-stop architecture, where the receiver output is wired into the drivetrain stop path rather than being interpreted only by software. The target wireless range is at least 100 ft, with full physical validation pending final assembly. The mechanical e-stop is designed as a hardware push-to-stop input located at the rear of the robot, allowing the drivetrain to be disabled without depending on the autonomy computer.
Requirement H is addressed through material selection, component placement, and shielding. The robot uses aluminum and polycarbonate structure, elevated drivetrain components, IP-rated exposed components where available, and PETG enclosures for water-sensitive electronics. Since the robot is still in final construction, the most recent actual validation is design-level inspection rather than full outdoor rain testing. This indicates that the safety design has been incorporated into the vehicle architecture, but final compliance must still be verified through pre-competition inspection and testing.
Requirement I is addressed through the use of a main fuse, power distribution protection, regulated low-voltage electronics rails, and motor-controller current limiting. The target behavior is that drivetrain overcurrent or unsafe motor-controller behavior disables the affected power path before wiring or electronics are damaged. Final measured breaker/fuse behavior has not yet been validated on the complete robot. The main remaining safety task is to test the completed robot under power and confirm that the fuse,

regulators, ODrive limits, and e-stop behavior work as expected with the full drivetrain and compute load installed.

4 | Electrical Design

4.1 | Electrical Architecture Overview

Tin Shady's electrical system is designed around a 12V power distribution system that powers all computation, sensing, and motor control components. The figure below shows the full electrical block diagram including power, communication, and safety communication pathways.

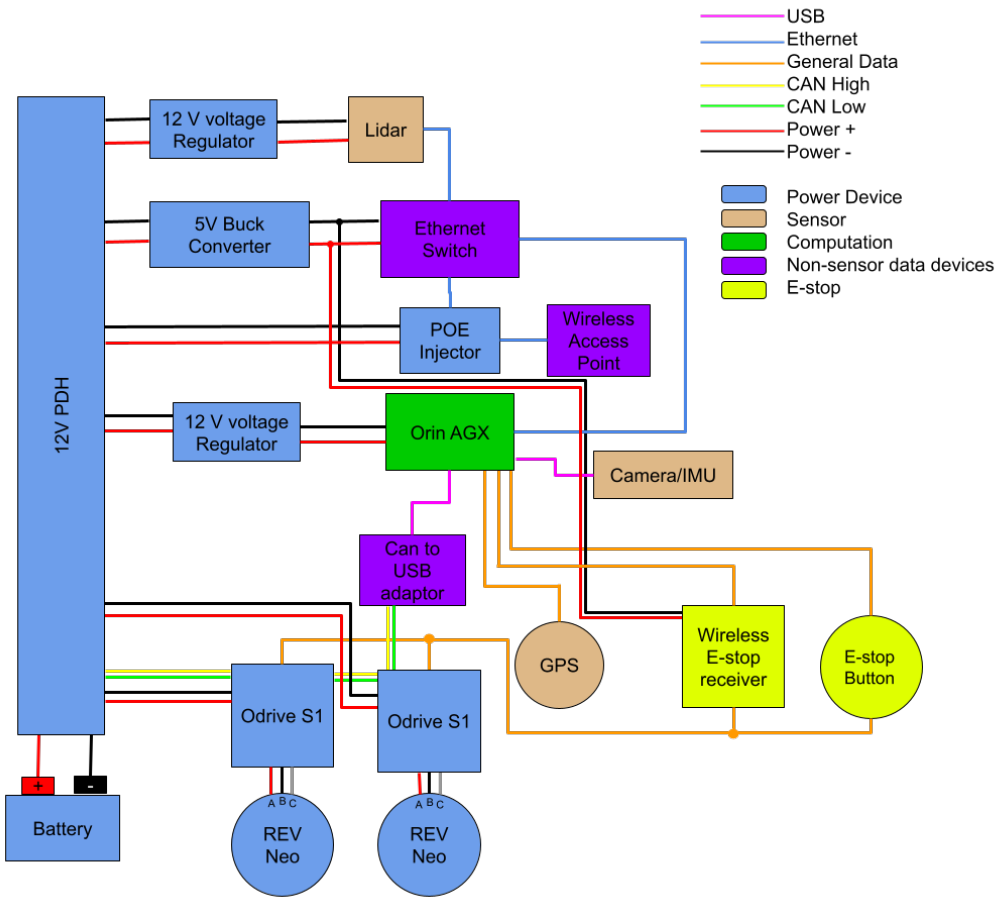


Figure 4.1: Full electrical block diagram

4.2 | Power Distribution

The power distribution system is designed to separate high-current drivetrain loads from low-power sensors and computation devices. Power comes from the 12V battery which is routed through a 120A fuse and then into the power distribution hub (PDH). The PDH supplies raw battery voltage to the ODrive S1 motor controllers, for high-current drivetrain operation. To support the low-current sensitive electronics, the PDH also feeds multiple regulators. A 5V buck converter provides stable low-voltage power for logic computation. In addition, dedicated 12V voltage regulators supply consistent power

to subsystems such as sensors and computation. By isolating regulated electronics from the drivetrain power, the system reduces voltage sag and electrical noise caused by motor load spikes. This improves the overall system's stability and prevents unexpected failures in computation and sensing components. The 120A fuse was selected as the primary upstream protection device for the 12V distribution system. This provides protection for the main battery output while still allowing short-duration drivetrain current spikes during acceleration or turning. Low-voltage electronics are isolated from drivetrain load transients through dedicated regulators, reducing the chance that motor current spikes reset the Jetson, sensors, or networking hardware. The expected runtime target is at least one complete 6-minute AutoNav run under competition loading. Because the vehicle is still in final integration, full-system runtime has not yet been measured; final validation will be performed by running the drivetrain, compute system, sensors, and safety light simultaneously under representative autonomous operation. The battery capacity is 18 Ah at 12V, corresponding to approximately 216 Wh of nominal stored energy.

4.3 | Compute Platform

Due to its preexisting availability on RPI's campus and exceptional computing power, we elected to use a NVIDIA Jetson Orin AGX 64GB Developer Kit as the robot's primary computer. This allowed memory and CPU usage to not be a bottleneck for our algorithmic design, and for GPU utilization to be possible for the ZED camera's depth calculations. Along with being specifically designed for edge computing applications, this computer provides a diverse GPIO header, supporting a large variety of communication formats.

4.4 | Sensors

To know its surroundings Tin Shady uses a number of sensors. The S2E LiDAR paired with a ZED 2i camera provides obstacle detection and environmental mapping through ethernet and USB communication respectively. With GPS supplying localization data, and the ZED's internal IMU, provide both visual and inertial measurements for navigation and state estimation. These sensors all interface directly with the Orin. The sensor suite was selected to balance reliability, cost, and integration complexity for a rookie AutoNav platform. The LiDAR provides geometry-based obstacle detection that is less sensitive to lighting changes, while the ZED 2i provides RGB and depth data for lane, pothole, and visual-feature detection. GPS provides coarse global localization for waypoint navigation, and inertial data from the ZED assists with short-term motion estimation. This combination allows the autonomy system to avoid relying on a single sensor modality: camera data is useful for lane markings and potholes, while LiDAR is useful for obstacles and local free-space estimation.

4.5 | Motor Control

Tin Shady uses two ODrive S1 motor controllers to drive the left and right drivetrain motors. The differential-drive layout allows the robot to command forward motion and turning by independently controlling the left and right wheel speeds. Each ODrive receives power from the high-current 12V distribution path and receives control commands from the onboard compute system through the robot's communication bus. This layout keeps the drivetrain electrically simple while still allowing closed-loop velocity control and software-defined motion commands.

Motor control was designed with safety and competition speed limits in mind. The controllers provide the low-level interface between autonomy commands and motor output, while the robot's higher-level driving logic limits commanded velocity based on lane confidence, obstacle proximity, turning radius, and ramp traversal. The emergency-stop path is designed to override normal motor control and disable drivetrain output without depending on the autonomous software stack. This separation between normal control

and emergency shutdown is important because a software planner failure should not prevent the robot from stopping.

4.6 | Requirement Adherence

The electrical design primarily supports Requirements I and J, while also enabling the perception, driving logic, and safety requirements. Requirement I is addressed through the 120A main fuse, protected power distribution, regulated low-voltage rails, and motor-controller current limiting. The design target is that drivetrain faults or overcurrent conditions are isolated before they damage wiring, regulators, or compute hardware. The most recent actual value for full-system fault response is not yet available because the robot is still in final assembly. This indicates that the electrical protection strategy has been designed into the architecture, but must still be validated with controlled power-on and drivetrain tests.
Requirement J is addressed by sizing the power system for at least one 6-minute AutoNav run under expected competition loading. The drivetrain is the dominant electrical load, while the Jetson, LiDAR, camera, GPS, networking equipment, and safety light form a smaller but continuous compute and sensing load. The target result is that the robot remains powered, responsive, and capable of driving for longer than the competition run time. Full-system runtime has not yet been measured, so final validation will be performed after drivetrain assembly by powering all major loads simultaneously and logging battery voltage during representative autonomous operation.
The electrical system also supports Requirement G by keeping the e-stop path separate from normal autonomy software. The wireless receiver and physical e-stop button are represented as safety-critical inputs in the electrical architecture. Their purpose is to remove or disable drivetrain output even if the Jetson, ROS 2 stack, or navigation software is not operating correctly. This reduces the risk that a software failure becomes an uncontrolled-motion hazard.

5 | Perception

5.1 | Software Architecture

To develop our autonomous software, we focused primarily on using existing open source frameworks, tools, and packages. This includes ROS2, Nav2 navigation package, OpenCV for detecting lanes and potholes, and RTAB-Map for mapping and localization.

5.2 | Lane Detection

The lane detection node processes synchronized RGB and depth images from the onboard camera. Each RGB frame is converted to grayscale, smoothed with a Gaussian blur, and processed with Canny edge detection to extract strong visual edges. A probabilistic Hough Line Transform is then used to identify line segments corresponding to visible lane-boundary candidates. The detected line pixels are matched with depth values from the aligned depth image and projected into 3D using the camera intrinsics. These points are converted into the robot's local coordinate frame, split into left and right boundary paths based on lateral position, sorted by forward distance, and published as `/left_boundary` and `/right_boundary` for use by the line pathing system.

5.3 | Obstacle Detection

Obstacle detection uses LiDAR and depth-camera data to identify non-ground objects within the local drivable corridor. LiDAR scans are filtered to remove returns from the robot body and points outside the expected obstacle height/range. Remaining points are clustered in the robot frame and converted

into obstacle markers or costmap obstacles. Depth-camera data provides supplemental detection for obstacles that may be poorly represented in 2D LiDAR, while LiDAR provides reliable range data in variable lighting. Detected obstacles are published into the local costmap for Nav2 and are also available to the driving logic as obstacle positions relative to the robot.

5.4 | Localization and Mapping

RTAB-Map and local costmaps are used for short-horizon situational awareness, not for memorizing the competition course between runs. The robot builds a local representation of nearby lane boundaries, obstacles, and free space during each run. GPS is used only for the provided navigation waypoints and coarse global progress, while local lane and obstacle perception governs immediate motion.

5.5 | Requirement Adherence

6 | Driving Logic

6.1 | Behavior Overview

Tin Shady's driving logic is organized as a layered autonomy stack rather than a single monolithic controller. Perception nodes generate lane boundaries, obstacle locations, and local costmap information. A goal-selection layer converts this information into short-horizon navigation goals. Nav2 then generates velocity commands to move toward those goals while respecting obstacle constraints. This structure was chosen so that individual behaviors, such as lane following, obstacle avoidance, waypoint approach, and ramp traversal, can be tested separately before being integrated into a full AutoNav run.

6.2 | Lane Following

After the lane detection system identifies the left and right course boundaries, a ROS 2 goal selection node converts those boundary paths into a local navigation target for the robot. The node subscribes to `/left_boundary` and `/right_boundary`, each published as a `nav_msgs/Path`, and periodically selects a lookahead point from both boundaries. The x-y coordinates of the selected left and right points are averaged to estimate the center of the drivable lane. This midpoint is then published as a PoseStamped goal on `/goal_pose`, allowing the navigation stack to guide the robot toward the center of the lane. To provide a meaningful heading for the goal, the node also calculates a second midpoint slightly farther along the boundary paths. The direction from the current midpoint to this future midpoint is converted into a yaw angle, then transformed into a ROS quaternion for the goal orientation. This allows the generated goal to point in the direction of the lane rather than simply marking a position.

6.3 | Obstacle Avoidance

Obstacle avoidance is handled through the robot's local costmap and navigation controller. Obstacles detected by LiDAR and depth sensing are inserted into the local costmap as occupied regions. When an obstacle blocks the nominal centerline path, the planner selects a feasible path around the occupied region while remaining inside the perceived lane boundaries. The intended behavior is to bias the robot toward the side of the lane with greater free space, while maintaining a target clearance of at least 0.5 m from obstacles. If the planner cannot identify a safe path, the robot should stop rather than continue into an uncertain region.

6.4 | Waypoint Navigation

Waypoint navigation is used for competition-provided navigation goals, especially in No Man's Land and near the ramp entrance. GPS is fused with other odometry and state-estimation sources to estimate the robot's position relative to these global goals. The waypoint-following layer provides a high-level goal to Nav2, while local lane, LiDAR, depth, and costmap information continue to govern immediate motion decisions.
This prevents the robot from blindly driving toward a GPS coordinate when obstacles, barrels, boundaries, or the ramp require local path adjustment. In practice, waypoint navigation is treated as a coarse routing behavior layered above the same local planning stack used for lane following and obstacle avoidance.

6.5 | Ramp Handling

The AutoNav course contains a single ramp feature, so Tin Shady treats ramp traversal as a special constrained behavior rather than as a general repeated obstacle type. When approaching the ramp-associated waypoint region, the robot should reduce speed, align its heading with the ramp direction, and avoid aggressive steering commands while climbing or descending. This reduces the risk of slipping, tipping, losing localization quality, or incorrectly interpreting the ramp surface as an obstacle.
Because the ramp is expected to be encountered only once during a run, the ramp behavior is primarily a transition mode: approach the ramp using waypoint and local planner guidance, traverse it conservatively, then return to normal lane following and obstacle avoidance after exiting the ramp region.

6.6 | Special Obstacle Handling

Special obstacles require conservative handling because they may not resemble ordinary barrels. Simulated potholes are treated as no-go regions when detected as solid white circular markings. Center obstacles and chicanes are handled by selecting the side of the lane with greater free-space clearance. If the robot enters a dead-end or trap-like region where no valid local path is available, the safest intended behavior is to stop and wait for recovery or e-stop rather than risk a boundary crossing or collision. This conservative strategy prioritizes avoiding end-of-run penalties over forcing uncertain motion.

6.7 | Speed Management

Speed management is based on both competition constraints and perception confidence. The drivetrain is designed to remain below the 5 mph maximum speed limit, while autonomous motion targets a lower operating speed to provide enough time for perception and planning. The robot should slow down during sharp turns, obstacle avoidance, ramp traversal, poor lane confidence, or uncertain costmap conditions. During straight, high-confidence lane following, the target speed should remain comfortably above the 1 mph minimum speed requirement.

6.8 | Requirement Adherence

The driving-logic requirements are currently validated at the design and software-architecture level, with full physical validation pending final robot integration. The lane-following design directly supports Requirement C by generating midpoint goals between detected lane boundaries. Obstacle avoidance supports Requirement D by using local costmap obstacles to modify the planned path around detected hazards. Waypoint navigation supports qualification and No Man's Land traversal by allowing GPS goals to guide the robot while preserving local obstacle avoidance. Final validation will be performed through staged outdoor tests beginning with straight-line speed compliance, followed by lane following, obstacle avoidance, waypoint approach, and representative full-course attempts.

7 | Key Performance Indicators

At the time of submission, Tin Shady is still in final manufacturing and integration, so several full-vehicle measurements are not yet available. Rather than reporting unverified results, this section lists the main values we plan to measure during testing. These KPIs were chosen because they map directly to qualification and run-ending failure modes: speed, lane keeping, obstacle avoidance, e-stop behavior, payload retention, and ramp traversal.

7.1 | KPI Selection & Table

The following KPIs were selected to connect the robot's design requirements to measurable competition performance:

1. Speed compliance. The robot shall complete the initial 44 ft qualification segment while averaging greater than 1 mph. This KPI is tied to the speed-management requirement and is required because vehicles moving too slowly may be disqualified or stopped for holding up traffic. Additionally, The robot shall remain below the 5 mph competition speed limit through a hardware speed limit and conservative software velocity commands. This is checked using ROS topic rates and visual inspection of lane outputs.

2. Lane-boundary detection rate. The perception system shall publish usable left and right lane-boundary estimates at a target rate of 10 Hz during autonomous operation. This is checked during lane-following tests on a marked outdoor course.

3. Lane-center tracking error. During representative lane-following tests, the robot shall maintain an average centerline error below 0.5 m when both lane boundaries are visible. This is also checked during lane-following tests on a marked outdoor course.

4. Obstacle detection range and clearance. The perception system shall detect representative barrels or course obstacles at a target distance of at least 2.0 m. This KPI is tied to obstacle avoidance, since the robot must identify obstacles early enough to plan around them. During representative obstacle-avoidance maneuvers, the robot shall maintain at least 0.5 m clearance from detected obstacles. This KPI reduces the risk of collision, sideswipe penalties, and end-of-run events.

5. Wireless emergency-stop range. The wireless emergency stop shall actuate the drivetrain stop path from at least 100 ft away. This is checked during wireless e-stop range and signal-loss tests.

6. Autonomous course completion time. The robot shall be capable of completing a representative 500 ft AutoNav-style course within the 6 minute run limit. This is checked during the final integrated run.

7.2 | KPI Results & Analysis

At the time of report submission, full-vehicle KPI validation has not yet been completed because the robot is still in final manufacturing and integration. The team's immediate testing plan is to validate the KPIs in dependency order: first confirming e-stop behavior, drivetrain speed limits, and payload retention; then validating sensor data streams and perception update rates; then testing lane following and obstacle avoidance on a shortened outdoor course; and finally attempting a full representative AutoNav run. This staged approach prioritizes qualification-critical safety and mobility requirements before higher-level course-performance optimization.

8 | Analysis of Complete Vehicle

8.1 | Integration Lessons Learned

The largest integration lesson from Tin Shady was schedule-related. The robot was designed to be mechanically simple, but manufacturing and assembly still took longer than expected. This compressed the time available for full-system testing and forced the team to prioritize qualification-critical work first: drivetrain motion, e-stop behavior, sensor bringup, and basic autonomy.

A second lesson was that packaging matters even on a small robot. The payload bay, electronics deck, drivetrain, camera mast, LiDAR, and wiring all compete for space. CAD bounding boxes helped catch some conflicts early, but final assembly still required coordination between mechanical access, wiring serviceability, and sensor placement.

8.2 | Hardware Failures and Mitigations

Since the vehicle is still in final construction, the team has focused on preventing likely hardware failure modes before full testing begins. Expected risks include loose fasteners, chain tension changes, sensor mount vibration, drivetrain current spikes, and wiring strain during motion. Mitigations include using gussets and repeated fastener checks, mounting drivetrain components above the lower frame where possible, separating high-current and regulated electronics, and routing wiring to reduce strain and accidental disconnection. These mitigations will be verified during initial drivetrain and outdoor tests.

8.3 | Software Testing Process

Software testing is staged to reduce the risk of debugging the entire autonomy stack at once. Individual ROS 2 nodes are first tested independently using known inputs, logged data, or simulated sensor messages. Perception outputs are inspected visually in RViz or debug displays before being used for motion. Driving logic is then tested with simplified goals before enabling full autonomous navigation. This process is intended to catch topic, frame, unit, and configuration errors before the robot is moving at speed.

8.4 | Simulation Testing

Simulation is used to test software structure before the physical robot is fully available. Simulated testing is most useful for verifying ROS 2 launch files, topic connections, navigation configuration, lane-goal generation, and basic planner behavior. However, simulation cannot fully replace outdoor testing because real lighting, asphalt texture, sensor noise, wheel slip, vibration, and mechanical tolerances strongly affect AutoNav performance. For this reason, simulation is treated as a debugging and integration tool rather than proof of final competition readiness.

8.5 | Physical Testing

Physical testing is planned in a staged sequence after final assembly. The first tests will verify battery isolation, breaker behavior, e-stop function, safety-light behavior, and drivetrain motion while the robot is lifted or restrained. The second stage will test low-speed teleoperated motion, speed limiting, and payload retention. The third stage will validate sensor data streams and perception outputs outdoors. The final stage will combine lane following, obstacle avoidance, waypoint navigation, and ramp handling on a representative AutoNav-style course.

9 | Cyber Security Analysis

9.1 | Context

Although Tin Shady is a competition prototype, cybersecurity is still relevant because the robot contains wireless devices, networked sensors, ROS 2 communication, and command pathways capable of affecting physical motion. The most important security goal is preventing unauthorized or accidental commands from causing unsafe drivetrain behavior. For this reason, the safety-critical emergency stop path is designed to remain hardware-based rather than dependent on the autonomous computer.

9.2 | Identified Vulnerabilities

- Vulnerability 1: Unauthorized ROS 2 commands. If an attacker or misconfigured device joined the robot network, it could potentially publish unsafe velocity commands or interfere with autonomy topics. A production version of this system would mitigate this with network isolation, allowlisted devices, authenticated communication, and restricted command topics.
- Vulnerability 2: Wireless network exposure. The robot uses wireless communication for development, monitoring, or emergency-stop-related functions. Poorly secured wireless access could allow unauthorized access or interference. A production system would use strong authentication, disabled default credentials, limited open ports, and firewall rules restricting access to only required services.
- Vulnerability 3: Sensor spoofing or data corruption. Because the autonomy stack depends on camera, LiDAR, GPS, and IMU data, corrupted or spoofed sensor inputs could cause incorrect planning decisions. A production system would mitigate this using sensor-fusion consistency checks, sanity bounds on sensor data, watchdogs for stale topics, and fallback behaviors that stop the robot when perception confidence is too low.
- Vulnerability 4: E-stop interference. The wireless e-stop is safety critical, so loss of signal, interference, or spoofing must fail toward safety. The competition robot addresses this by keeping the actual stop action hardware-based. A production system would further harden this path with authenticated pairing, signal-health monitoring, and a default-safe behavior when communication is lost.