Intelligent Ground Vehicle Competition 2026

Wayne State University

Warrior Robotics

Design Report

WARRIOR 1

AutoNav & Self Drive

Team Captain

Salem Al-Hadrusi, eb2357@wayne.edu, 248-422-8872

Project Czar

Daniel Lord, hq9142@wayne.edu, 217-891-4808

I certify that the development of the vehicle, Warrior 1, described in this report is equivalent to the work in a senior design course. The students of this team have prepared this report under my guidance.

Abhilash Pandya, Professor, apandya@wayne.edu**,** 734-552-8452

Yanbing Mao, Assistant Professor, hm9062@wayne.edu

Department of Electrical and Computer Engineering Date: 15/5/26

Table of Contents

1. SYSTEM AND SUBSYSTEM REQUIREMENTS 1

1.1 Introduction 1

1.2 System Identification and Requirements 1

2. MECHANICAL DESIGN 3

3. SAFETY 5

3.1 Overview 5

3.2 Emergency Stop Architecture 6

3.3 Relay Selection and Shutdown Reliability 6

3.4 Power Isolation and Safe Enable Logic 6

3.5 Wireless Shutdown and Status Indication 6

4. Electrical/Electronic Design 7

4.1 Overview 7

4.2 Swerve-Module Power Architecture 7

4.3 Compute and Sensor Power Architecture 8

4.4 Power Distribution Capacity, Runtime, and Recharge 8

4.5 Wiring, Connectors, and Communication Hardware 9

4.6 Cooling and Electrical Fault Isolation 9

4.7 Protection Strategy and Measured Electrical Results 9

5. PERCEPTION 10

5.1 Overview 10

5.2 Data Collection and Annotation 10

5.3 Multitask Model Architecture 11

5.4 Training and Evaluation 11

6. DRIVING LOGIC 12

6.1 Overview 12

6.2 Lane Following, Obstacle Avoidance, and Costmap Construction 12

6.3. GPS Waypoint Navigation and Localization 13

6.4. Path Planning and Motion Generation 13

6.5. AutoNav Driving Logic 14

6.6. Simulation, Digital Twin Testing, and Recovery Behavior 14

7.1 Key Performance Indicators 15

8. Analysis of Complete Vehicle 16

8.1. Integration Lessons and Development Findings 16

8.2. Simulation, Physical Testing, and Remaining Vehicle-Level Risks 17
9. CYBER SECURITY 17

1. SYSTEM AND SUBSYSTEM REQUIREMENTS

1.1 Introduction

Warrior 1 is Wayne State University's newly developed autonomous ground vehicle for the 2026 Intelligent Ground Vehicle Competition. The platform was designed as a modular research and competition vehicle capable of supporting both the AutoNav and Self Drive challenges. Its development focuses on reliable outdoor autonomy, including perception, localization, planning, control, safety, and onboard computation.

Gokul Kunmaran	Mehreen Farooqi	Ali Mustafa	Ryan Ross
Rowell Delos	Lucas Frazier	Siva Jaideep	Jean Dilloway
Hadi Mhanna	Sam Mead	Kalaivani	Augustin Garcia
Adrian Tlatelpa	Mohamad Saab	Brian Dang	
David Majano	Jordan Davis	Syeda Ahmed	
Josh El	K'Ron Boyd	Remi Flanz	
Elias Trevino	Yihao Cai	Samuel Redman	

Table 1. Wayne State Robotics Team

Rather than adapting a previous vehicle, Warrior 1 was developed as a new platform to improve maneuverability, software modularity, and perception capability for competition-style environments. The design emphasizes rule compliance, field robustness, and future expandability as the team (Table 1) prepares the vehicle for increasingly complex autonomous navigation tasks.

1.2 System Identification and Requirements

The system engineering process began by reviewing the IGVC 2026 rulebook and identifying the primary vehicle functions required for competition. These functions included autonomous operation, lane keeping, obstacle avoidance, GPS waypoint navigation, payload support, speed compliance, onboard computation, and safe emergency shutdown.

The team then translated these vehicle-level needs into subsystem requirements for mechanical design, electrical power, safety, perception, localization, planning, control, and computing. In addition to rule compliance, the team considered expected field conditions such as changing outdoor lighting, uneven pavement, ramps, cluttered obstacle fields, variable course layouts, glare, occlusion, and GPS uncertainty.

These requirements flow down guided the major architecture decisions for Warrior 1. A three-module swerve-drive platform was selected to improve maneuverability, a ROS 2-based autonomy stack was selected for modular software integration, and a learned perception system was developed to identify lanes, obstacles, potholes, pedestrians, and signs in real time. Each major design choice was therefore tied to either a competition requirement, an expected operating condition, or a vehicle-level performance objective.

1.3 System and Subsystem Requirements

Rule / Design Driver	Requirement	Subsystem	Target / Verification
Vehicle size limits	The vehicle shall satisfy IGVC dimensional constraints.	Mechanical	3–7 ft long, 2–4 ft wide, and ≤6 ft tall. Verified by inspection.
Payload requirement	The vehicle shall securely carry the required payload during operation.	Mechanical	20 lb payload mounted without shifting. Verified by static load test.
Emergency stop requirement	The vehicle shall support reliable emergency shutdown.	Safety / Electrical	Mechanical and wireless E-stop disable propulsion. Verified during safety testing.
Autonomous operation	The vehicle shall operate without external control during competition runs.	Software / Computer	All perception, planning, and control run onboard. Verified through autonomous trials.
Lane and obstacle navigation	The vehicle shall detect course boundaries and avoid obstacles.	Perception / Planning	Lane and obstacle outputs support ROS 2 navigation. Verified through course testing.
GPS waypoint navigation	The vehicle shall navigate through assigned GPS waypoints.	Localization / Planning	Waypoints are reached during outdoor trials. Verified through waypoint testing.
Speed compliance	The vehicle shall operate within the required competition speed range.	Control / Performance	Speed remains within rule limits during autonomous operation. Verified by timed runs.
Self Drive behavior	The vehicle shall respond to pedestrians, signs, potholes, and road obstacles.	Self Drive Perception / Logic	Stable detections produce correct stop, yield, avoid, or continue behaviors. Verified through scripted scenarios.

Outdoor robustness	The vehicle shall operate under realistic outdoor course conditions.	Full System	Performance remains stable under lighting changes, ramps, glare, occlusion, and GPS noise. Verified through repeated field testing.

Table 2. System and Subsystem Requirements

Table 1 summarizes the primary system and subsystem requirements used to guide the design of Warrior 1. Each requirement is connected to a rule-derived need or competition design driver, assigned to a responsible subsystem, and paired with a measurable target or verification method.

2. MECHANICAL DESIGN

2.1 Overview

Fig 2.1 *Warrior 1 three-wheel swerve-drive mechanical layout*

Warrior 1 was designed as a compact, highly maneuverable autonomous ground vehicle for outdoor IGVC operation. The platform uses a three-module swerve-drive configuration arranged in a triangular geometry. This layout provides omnidirectional motion, stable ground contact, and reduced drivetrain complexity compared with a four-module swerve architecture.

The vehicle has an approximate measured footprint of **47.8 in wide by 44.0 in long**, with a total height of approximately **49.1 in** including the sensor mast. These dimensions keep the robot within the required IGVC vehicle envelope while providing enough width for lateral stability and enough internal volume for batteries, computing hardware, power electronics, sensors, and the required payload.

2.2. Frame, Structure and Packing

The chassis is built primarily from **1 in × 1 in 80/20 aluminum extrusion** and **0.125 in aluminum plates**. This structure was chosen because it provides a strong, modular, and serviceable frame while allowing rapid design changes during testing. The triangular footprint places one swerve module near each corner of the load path, improving balance during turning and reducing unnecessary weight from a fourth drive module.

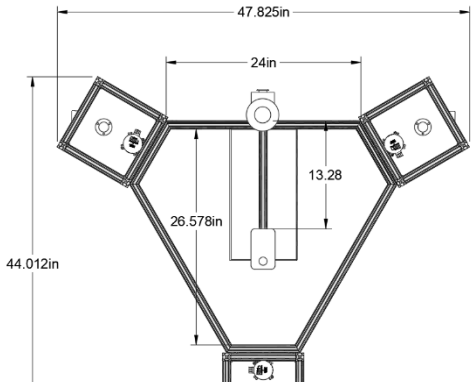

Fig 2.2 *Dimensioned top view of Warrior 1 mechanical layout.*

The robot uses a two-level packaging layout. Batteries, payload mass, and primary power hardware are placed low in the chassis to reduce the center of gravity. Lighter electronics and control hardware are mounted higher for service access. The sensor mast raises the Insta360

camera and LiDAR to improve field of view for perception and localization. The current design uses a rigid chassis without active suspension; stability is achieved through the low-mounted mass distribution, triangular wheel placement, and wide footprint.

2.3. Drive System, Electrical Power System and Computer Controlled Motion

Warrior 1 uses **three ready-made swerve-drive modules** as the primary drive system. Since these modules were acquired as integrated drivetrain assemblies, their internal mechanical construction is not described in detail. Each module provides independent wheel driving and steering, allowing the robot to translate, rotate, and maneuver around obstacles without requiring a traditional Ackermann steering layout.

The electrical power system is divided into separate drive and auxiliary power domains. The drive power system supplies the swerve modules and motor-control hardware, while the auxiliary power system supports onboard computation, sensors, safety electronics, and communication hardware. Separating these domains helps reduce the effect of high-current drivetrain loads on sensitive compute and perception components.

Computer-controlled motion is enabled through the onboard autonomy stack. High-level navigation commands are generated in ROS 2 and converted into module-level steering and drive commands. The control system allows the robot to follow planned paths, avoid obstacles, rotate in place, and execute low-speed autonomous maneuvers required for AutoNav and Self Drive operation.

2.4. Requirement Driven significant Mechanical Components

Component	Type	Subsystem	Function in Vehicle
1 in × 1 in 80/20 aluminum extrusion	Structural stock	Frame / chassis	Forms the main modular chassis structure.
0.125 in custom aluminum plates and brackets	Custom fabricated	Frame / mounting	Provides mounting surfaces for drivetrain, batteries, electronics, and sensor mast.
Three swerve-drive assemblies	Team-integrated assembly	Drive system	Enables independent wheel steering and drive for omnidirectional motion.
NEO Brushless Motor V1.1	Ready-made / COTS	Drive system	Provides drive or steering actuation for each swerve module.

Spark MAX motor controller	Ready-made / COTS	Motion control	Controls NEO brushless motors using commands from the onboard control system.
MAX Planetary Gearbox 15:1	Ready-made / COTS	Drive transmission	Provides gear reduction for increased torque and controllable low-speed motion.
45-tooth sprocket, 16-tooth hub sprocket, and #35 roller chain	Ready-made / COTS	Power transmission	Transfers motor torque through the drivetrain.
Rear hub assembly	Ready-made / COTS	Wheel interface	Supports wheel mounting and rotational load transfer.
CV axle	Ready-made / COTS	Drive system	Transfers torque through the wheel assembly while supporting drivetrain packaging.
Hoverboard wheel components	Repurposed commercial component	Wheel system	Provides wheel hardware for ground contact and vehicle motion.
Mechanical connectors and fasteners	Ready-made / COTS	Frame / assembly	Connects structural members and supports modular assembly.

Table 3. Major Mechanical and Motion Components

3. SAFETY

3.1 Overview

Warrior 1 uses a hardware-based safety architecture designed to stop vehicle motion without depending on software. The safety system includes a rear-mounted mechanical E-stop, a wireless E-stop, relay-isolated traction power, separate main and auxiliary power domains, and a visible vehicle-status light. When either E-stop is activated, the control voltage to the traction relays is removed, which disconnects motor power and prevents the robot from continuing to move. The onboard computer can remain powered after a shutdown, allowing the team to review logs, sensor status, and system health before re-enabling the vehicle.

3.2 Emergency Stop Architecture

During **transport and parked conditions**, Warrior 1 is kept in a non-driving state by disabling the traction power path. The drive system is not enabled unless the main power system, auxiliary power system, and relay-control circuit are intentionally activated. This prevents unintended wheel motion while the robot is being moved, staged, serviced, or stored.

During **charging**, the vehicle is treated as a stationary system. The traction enable circuit remains off so the swerve modules cannot receive drive power. The separation between drive power and auxiliary power allows the team to manage charging, diagnostics, and system checks without enabling motion. This reduces risk to team members working around the robot and prevents accidental actuation during maintenance.

3.3 Relay Selection and Shutdown Reliability

Warrior 1 uses solid-state relays rated for 60 A in the traction shutdown system. These relays were selected to improve reliability under high-current conditions and reduce the risk of contact welding compared with conventional mechanical relay contacts. Their role is safety-critical: they act as the electrical interruption point between the enabled control state and power delivery to the drive motors. The system also includes a remote disconnect unit that isolates wheel power without shutting down the onboard computing systems. This allows the vehicle to enter a safe non-moving state while preserving software logs, sensor status, and other diagnostic information needed for recovery and troubleshooting.

3.4 Power Isolation and Safe Enable Logic

The robot uses separate main and auxiliary power systems, and both must be enabled for full vehicle operation. One switch powers the onboard computing systems, while a second switch provides the 5 V control voltage used for relay activation and motor power control. This arrangement electrically separates the computing system from the traction enable circuit and keeps the critical shutdown path hardware-based rather than software-dependent. Because wheel power depends on relay-control voltage, removing that control voltage immediately disables traction. At the same time, onboard computing can remain active, allowing the team to verify system status before the robot is re-enabled. This architecture supports both safer operation and more efficient post-stop diagnostics.

3.5 Wireless Shutdown and Status Indication

The **wireless E-stop** remains functional at distances greater than 100 ft, providing judges and operators with an independent way to shut down vehicle motion during autonomous operation. This remote shutdown capability extends the hardware safety system beyond the physical rear-mounted E-stop and allows intervention without approaching a moving robot. Warrior 1 also includes a visible status indicator light to communicate operating state to nearby personnel. The light remains solid whenever vehicle power is enabled and flashes during autonomous mode. This gives observers an immediate indication of whether the robot is merely powered or actively operating autonomously.

4. Electrical/Electronic Design

4.1 Overview

Warrior 1 uses a **distributed electrical architecture** designed to separate high-current drive loads from compute and sensor electronics while keeping the system modular and serviceable. The robot is powered by four 36 V, 20 Ah batteries. Three batteries independently support the three swerve-drive modules, while the fourth battery supplies the compute, sensing, communication, and auxiliary electronics. This layout reduces the chance that traction-motor voltage drops or current spikes will disturb the Jetson computer, LiDAR, cameras, or other low-power electronics. The electrical design is organized around four functions: swerve-module power delivery, auxiliary compute and sensor power, modular wiring and communication, and fault-tolerant power distribution. Custom PCBs, regulated DC-DC converters, XT60 connectors, modular panel connections, and separate voltage rails were selected to reduce wiring complexity and simplify service during testing and competition operation.

4.2 Swerve-Module Power Architecture

Each of the three swerve modules is electrically independent and powered by its own 36 V battery. A module contains one 36 V traction motor for wheel propulsion, and one 12 V steering motor for wheel rotation. A custom PCB was developed for each swerve module to manage local power distribution. The board accepts 36 V battery input, routes 36 V directly to the traction motor, and uses a buck converter to step the voltage down to 12 V for the steering motor. This localized conversion strategy reduces long low-voltage wire runs across the chassis and keeps each module self-contained.

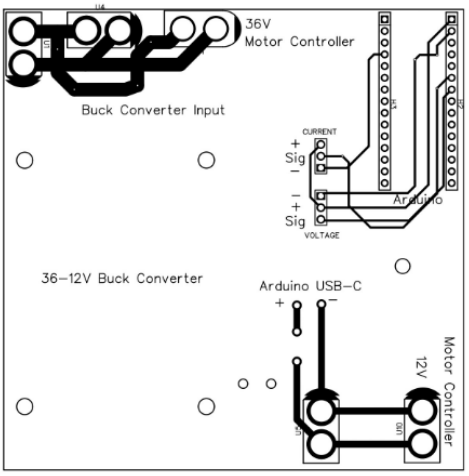

Fig 4.1 *Layout of the Custom Swerve PCB*

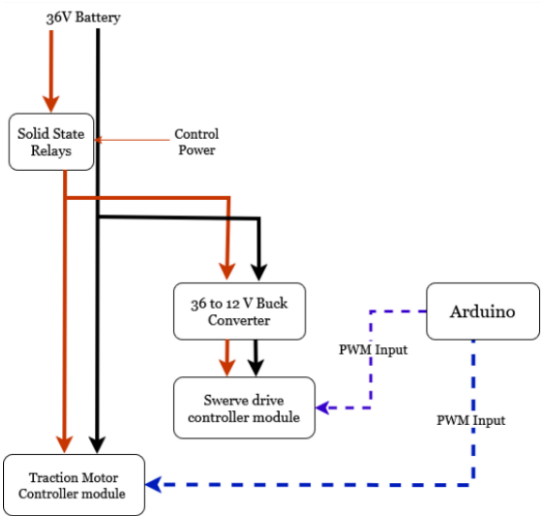

Fig 4.2 *Power Distribution for Swerve Module*

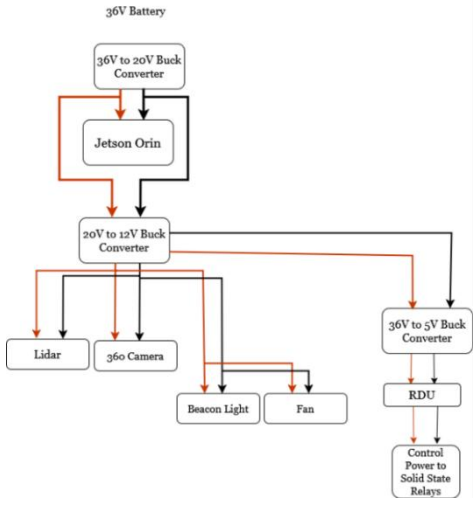

Fig 4.3 *Auxiliary Power System*

Custom PCBs were also used to improve wiring cleanliness and reduce electrical clutter inside the robot. By integrating DC-DC conversion, motor power distribution, and relay-control-related functions into compact local assemblies, the electrical system becomes easier to troubleshoot and individual modules become faster to replace during maintenance.

4.3 Compute and Sensor Power Architecture

The fourth 36 V, 20 Ah battery powers the robot's compute and sensing hardware. Rather than distributing raw battery voltage directly to every auxiliary device, the system uses several DC-DC converters to create regulated voltage rails matched to subsystem requirements. The first converter reduces 36 V to 20 V for the NVIDIA Jetson computer. Additional conversion stages produce 12 V, 9 V, and 5 V rails for supporting electronics and communication devices. The 12 V rail powers the LiDAR, cameras, USB hub, GPS, cooling fans, and mini PC. The 5 V rail supports relay-control electronics, the remote-disconnect electronics, and other low-power control devices. A regulated 9 V supply is used for the Ethernet hub. Separating the compute battery from the traction batteries improves system stability and reduces the possibility that motor-side noise will affect the robot's sensing or onboard processing.

The internal electronics layout reflects this **multi-rail design**. The power-conversion area contains four buck-converter stages: 36 V to 20 V, 20 V to 9 V, 20 V to 12 V, and 12 V to 5 V. These rails then feed the fan connections, Wago distribution points, relay circuits, remote-start wiring, USB hub, and sensor/compute loads throughout the robot.

4.4 Power Distribution Capacity, Runtime, and Recharge

Each battery is rated at 36 V and 20 Ah, providing approximately 720 Wh of stored energy. With four batteries installed, the robot carries approximately 2.88 kWh of nominal total stored battery energy, distributed across the three drive modules and the auxiliary electronics system. The electrical design estimates an operating **runtime** of approximately 2.5 to 3 hours under normal conditions. Actual runtime depends on terrain, steering activity, traction demand, and how aggressively the vehicle accelerates or changes direction. The batteries are charged using external smart chargers, with a full recharge time of approximately 3 to 5 hours depending on discharge level and charger current settings.

Subsystem	Voltage	Nominal Current	Peak Current	Estimated Power
Traction motor, hoverboard wheel	36 V	10 A	16 A	360 W nominal, 576 W peak
Steering Motor, NEO	12 V	15 A	105 A stall	180 W nominal, 1260 W stall
Jetson Orin	20 V	2 A	3 A	40 W nominal, 60 W peak
LiDAR	12 V	0.5 A	1 A	6 W nominal, 12 W peak
Ethernet Hub	9 V	0.3 A	0.6 A	2.7 nominal, 5.4 W peak
Insta360 Camera	5 V	2 A	3 A	10 nominal, 15 W peak
PC Fans	12 V	0.2 A each	0.5 A each	2.4 nominal, 6 W peak

TABLE 4. Subsystem and its Requirements

4.5 Wiring, Connectors, and Communication Hardware

Warrior 1's electrical system uses XT60 connectors for battery connections and panel-mounted connections to simplify maintenance and battery replacement. To make battery changes more consistent, the battery-facing connection convention was standardized so that battery-related interfaces use matching XT60 orientation across the robot. A four-outlet charging block was also created so all batteries can be charged without being removed from the vehicle. Wiring was organized to reduce clutter and minimize accidental contact. Insulated crimp connectors were used to avoid unnecessary solder splices, while fork and ring terminals were used at relay connection points. Wiring was routed along the edge of the robot where it was practical to preserve central space for electronics and improve accessibility. Fan-control wiring is routed through Wago connectors to the 12 V rail. The robot uses both Ethernet and USB communication to connect its computer platform with cameras, LiDAR, and other networked electronics. A dedicated Ethernet hub powered by a regulated 9 V supply provides the onboard network backbone, while USB hubs provide additional connectivity for peripherals and sensing devices.

4.6 Cooling and Electrical Fault Isolation

Cooling fans are installed inside the chassis to maintain airflow around the Jetson computer, DC-DC converters, and other heat-generating electronics. This is important because the electrical design contains multiple voltage-conversion stages and high-current subsystems operating within a compact vehicle enclosure.

The electrical design identifies several likely **failure points**:

- loose high-current connectors
- DC-DC converter overheating
- relay failure
- vibration-related wire damage
- excessive current draw during aggressive steering or acceleration

To reduce these risks, the system uses modular connectors, secured wiring, separate high-current and low-current routing where possible, cooling fans, and distributed batteries that isolate faults to individual subsystems. If one traction subsystem fails, the compute and sensing systems remain independently powered for **diagnostics and recovery**.

4.7 Protection Strategy and Measured Electrical Results

Warrior 1's electrical protection strategy combines battery management system protection inside each battery, XT60 high-current connectors, regulated DC-DC conversion, relay-controlled traction shutdown, solid-state relays, and manual disconnect switches. These measures are intended to reduce fault propagation, preserve electrical stability, and maintain serviceability under repeated testing conditions. Repeated driving and steering tests were used to verify stable voltage regulation, relay operation, and emergency shutdown functionality. Testing showed stable operation of the compute and sensing systems during steering and traction maneuvers. The distributed power architecture reduced electrical noise between the motors and sensor systems, cooling fans maintained stable operating temperatures around the Jetson computer and power electronics, and the regulated 20 V, 12 V, 9 V, and 5 V rails remained stable during normal operation.

5. PERCEPTION

5.1 Overview

Warrior 1's perception subsystem converts equirectangular imagery from the Insta360 X4 into scene information required for autonomous navigation. The system is designed around two primary tasks: lane segmentation and object detection. Lane segmentation identifies the left and right course boundaries needed for lane keeping, while object detection identifies relevant obstacles and traffic cues such as barrels, cones, pedestrians, and stop signs. AutoNav requires robust lane segmentation as well as object detection to remain in lanes and avoid discrete objects. Self drive extends the requirements of AutoNav and has an emphasis on object detection for pedestrians and signage on roads. These outputs which consist of masks, point clouds, and cost maps are passed to the downstream autonomy stack for navigation, obstacle avoidance, and behavior control.

A central design constraint was the use of panoramic equirectangular imagery rather than a conventional forward-facing camera. Early testing showed that standard lane-detection approaches and off-the-shelf segmentation pipelines did not transfer reliably to this geometry, especially under outdoor lighting variation and wide-angle distortion. Instead of reprojecting the imagery into perspective views and risking loss of context, the team chose to preserve equirectangular geometry throughout preprocessing, annotation, lane-label generation, training, and evaluation. This reduced train-deployment mismatch and kept the perception pipeline aligned with the robot's true sensing configuration.

5.2 Data Collection and Annotation

Object data was manually annotated using LabelMe (Fig. 5.1) for competition-relevant classes including barrels, cones, pedestrians, stop signs, and sign-like distractors. Lane supervision required a different approach. Classical geometric methods such as Hough-style processing were evaluated first, but they produced unstable masks and excessive background leakage on the team's scenes. To make lane-label generation scalable, the team adopted a pseudo-labeling workflow that combines CLRNet and SAM 2. CLRNet provides coarse lane structure, and SAM 2 refines those prompts into denser pixel-level masks. Because pseudo-label quality is not uniform, the pipeline uses confidence-aware filtering and weighting so that lower-quality masks contribute less during training. This made large-scale lane supervision practical without relying entirely on manual segmentation.

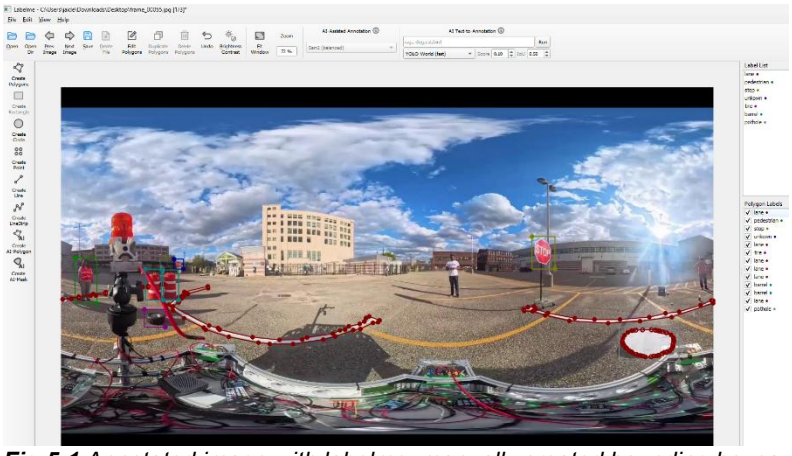

Fig 5.1 *Annotated image with labelme; manually created bounding boxes*

5.3 Multitask Model Architecture

The perception model uses a multitask architecture with a shared backbone, a Feature Pyramid Network (FPN), and two task-specific heads: a segmentation head for lane structure and pothole regions, and a detection head for discrete objects such as barrels, cones, pedestrians, and stop signs. This design was selected over task-specific models because the shared backbone improved both generalization and runtime efficiency during experimentation. By reusing low- and mid-level visual features across related tasks, the model reduces the computational cost of training and deployment while preserving strong task performance. Several backbone candidates were evaluated, including ResNet-18, ResNet-50, ConvNeXt-Base, Swin-B, and an HRNet-style option. Based on the current repository comparison results, ResNet-18 remains the most practical baseline because it provides the best tradeoff between segmentation performance and runtime cost on available hardware.

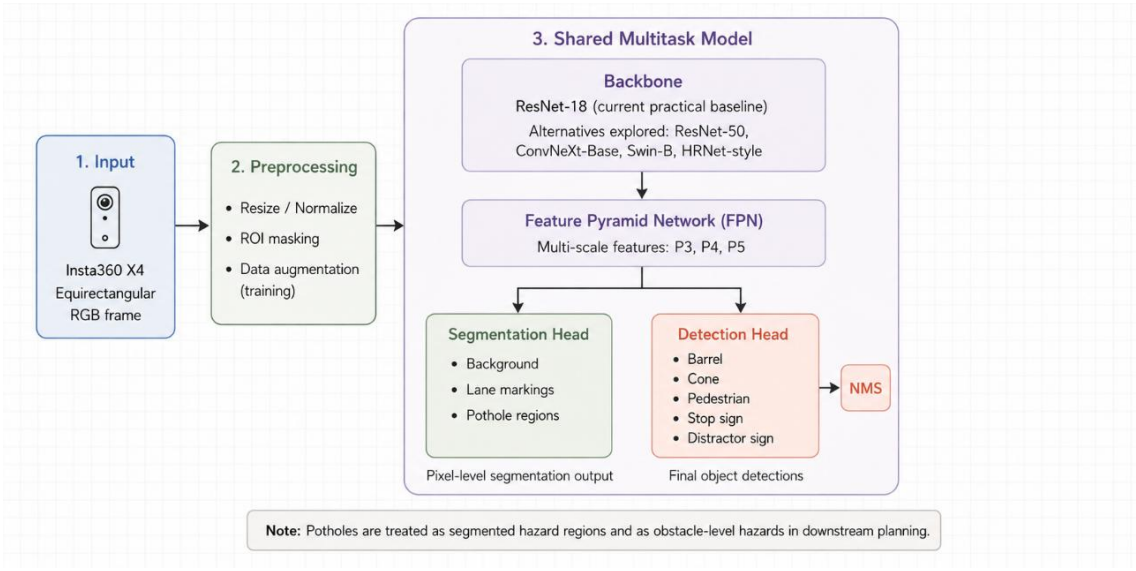

Fig 5.2 *Model Architecture Flow*

5.4 Training and Evaluation

Training is performed in PyTorch using AdamW with a cosine annealing learning-rate schedule. The data pipeline is manifest-driven and supports images, detection boxes, labels, segmentation masks, ROI masks, and segmentation-confidence values within a unified training format. To improve robustness under field conditions, the dataset is augmented using Albumentations which synthesizes data by altering brightness and contrast variation, hue-saturation shifts, blur, affine transforms, and related perturbations. These augmentations are intended to expose the model to the lighting variation, motion effects, and appearance changes expected in outdoor competition settings without departing from plausible operating conditions.

Run	**Backbone**	**Best Seg IoU**	**Best Seg Dice**	**Latency (ms)**	**Comment**
phase_A_resnet18_det_seg	ResNet-18	0.7967	0.8711	150.03	Best practical baseline

_convnext_base_det_seg	ConvNeXt-Base	0.7767	0.8489	158.30	Similar latency, lower IoU
_swin_b_det_seg	Swin-B	0.7902	0.8665	1498.36	Too slow in current form
hrnet_w32_det_seg_exploratory	HRNet-style fallback	0.8298	0.8923	1007.63	Best IoU, but high latency

TABLE 5. Runs and their Evaluations

6. DRIVING LOGIC

6.1 Overview

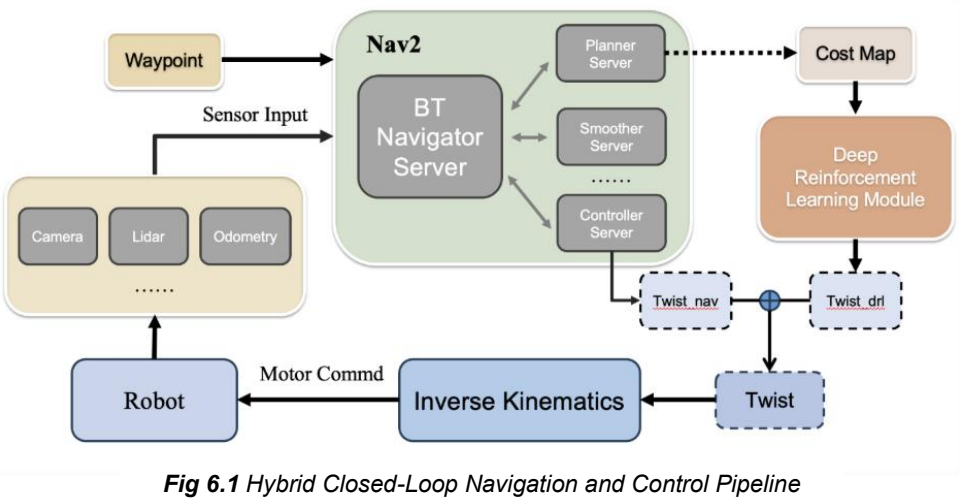

Fig 6.1 *Hybrid Closed-Loop Navigation and Control Pipeline*

Warrior 1's driving logic is built around a ROS 2 navigation stack that combines perception outputs, LiDAR occupancy data, GPS waypoint guidance, and fused localization to generate motion commands for the robot. The system is designed to follow lane boundaries, avoid obstacles, navigate between GPS waypoints, and recover from navigation errors without requiring direct operator intervention. The current driving logic documentation is designed such that the navigation stack, costmap structure, GPS waypoint process, path planning, motion control, and simulation workflow are defined.

6.2 Lane Following, Obstacle Avoidance, and Costmap Construction

The navigation stack treats both physical obstacles and detected lane boundaries as regions the robot should avoid. LiDAR scan data is passed into a SLAM toolbox node, which populates an occupancy grid with free and occupied space. This occupancy grid forms the obstacle layer of the navigation costmap. An inflation layer then expands obstacle regions to account for the robot footprint and increases the cost of nearby cells, creating a safer buffer around detected hazards.

Lane detections produced by the perception system are incorporated into the same obstacle layer. Since the robot should not make contact with or cross the lane boundaries, they are handled in navigation similarly to LiDAR-detected obstacles. The local costmaps produced from these layers are then fused into the broader **global costmap** used for path planning. This approach allows one navigation representation to support both obstacle avoidance and lane following. Rather than treating these as unrelated behaviors, the planner receives a unified map of where the robot can and cannot safely drive.

6.3. GPS Waypoint Navigation and Localization

GPS is used in two related ways. First, it contributes to the robot's localization estimate. Second, it provides the waypoint targets that guide navigation through the course. Waypoint navigation is handled through Nav2's logged waypoint follower, which reads a sequence of GPS waypoints from a YAML file and converts them into goal poses for the navigation stack. Once the robot reaches a waypoint within the specified tolerance, the next waypoint is passed forward until the final goal has been reached.

The localization system uses data from motor encoders, IMU accelerometers and gyroscopes, LiDAR, and GPS. State information is tracked through three coordinate frames: (1) Base: fixed to the robot, (2) Odom: used for continuous local motion tracking, (3) Map: used as the world-level reference frame. The separation between the odom and map frames is important because navigation requires continuous input, while GPS measurements are discontinuous and update less smoothly than odometry.

Sensor fusion is performed using an **Extended Kalman Filter** framework. The IMU contributes linear acceleration and angular velocity information. Encoder odometry supplies iterative motion estimates based on measured wheel behavior and robot geometry. GPS produces an additional odometry estimate in the global frame. Together, these inputs support estimation of the robot's linear and angular position, velocity, and acceleration.

The transform structure is generated through three nodes from the robot_localization package: two EKF nodes and one navsat transform node. The navsat transform node converts GPS information from UTM coordinates into the robot's map frame. The map EKF fuses IMU, encoder odometry, and GPS odometry to estimate the transform from map to base, while the odom EKF fuses IMU and encoder odometry to maintain continuous local tracking. These transforms provide the coordinate relationships required by the navigation stack. The SLAM configuration is intentionally adjusted, so it does not generate a conflicting map transform. It runs without a static layer to avoid interference with the GPS-derived map transform used by the localization system.

6.4. Path Planning and Motion Generation

Once the robot has a localized position and a goal pose, the navigation stack plans a route using the A* algorithm. The planner uses the costmap's cell-by-cell occupancy values to determine feasible travel space. In the documented representation, a value of 0 corresponds to free space, while a value near 254 represents a confirmed obstacle. Intermediate values form a gradient

around obstacles based on inflation and proximity cost. A* evaluates both distance to the goal and occupancy cost to find the lowest-cost sequence of cells between the robot pose and the goal pose. Cells above a selected cost threshold are rejected during planning so that the route does not allow any part of the robot footprint to collide with a mapped obstacle or boundary.

The resulting path is converted into robot motion using regulated pure pursuit. The controller selects a lookahead point 0.1 m ahead of the vehicle, draws an arc from the current robot position to that point, and generates a corresponding **cmd_vel command**. The process repeats continuously as the robot progresses along the path. Regulated pure pursuit was selected over standard pure pursuit because it also accounts for path curvature and potential collision conditions, adjusting velocity in response to local path demands. The navigation stack therefore follows a complete feedback loop:

Sensor inputs → localization and costmaps → A* path planning → regulated pure pursuit control → command velocity → robot actuation.

6.5. AutoNav Driving Logic

For AutoNav, the driving logic is intended to combine lane-following, obstacle avoidance, and GPS waypoint navigation into a single course-driving strategy. Detected lane boundaries and physical obstacles are inserted into the costmap as non-traversable or high-cost regions. GPS waypoints establish the broader direction of travel, while the local planner determines the safe path through the currently perceived environment.

This design directly supports common AutoNav conditions such as curved course segments, obstacles located near one side of the lane, and path adjustments around blocked regions. The system is also intended to address more difficult course elements such as ramps, switchbacks, center-island dead ends, traps, and potholes. The underlying navigation framework is prepared to handle these through perception-fed costmaps, global replanning, and recovery behaviors, although final course-specific validation remains to be documented.

6.6. Simulation, Digital Twin Testing, and Recovery Behavior

The full navigation stack was simulated using a TurtleBot3 model on a custom **Gazebo** track and was also exercised in real life on a mini IGVC-style track. The simulation used a simplified sensor set consisting of 2D LiDAR, IMU, and spoofed GPS coordinates. This approach allowed

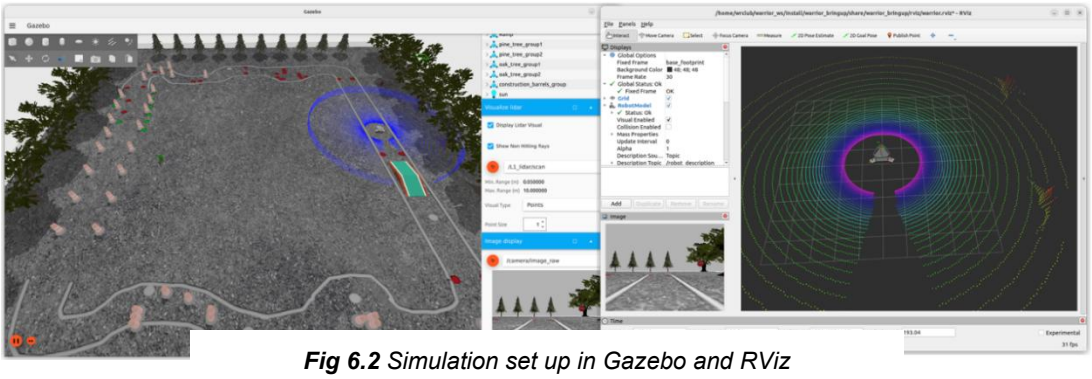

Fig 6.2 *Simulation set up in Gazebo and RViz*

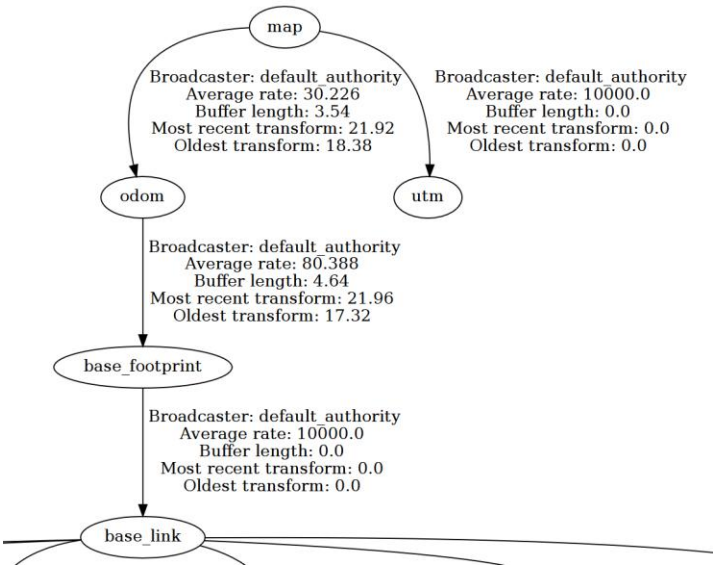

Fig 6.3 *Robot transform tree used by Navigation stack*

the same waypoint-following logic to be tested across different robotic platforms and prioritized modular development of the navigation stack before full vehicle integration.

Behavior trees are used to manage both standard navigation and recovery logic. The base navigation behavior governs when new paths are generated and when costmaps should be cleared to respond to potential sensor error. If a navigation error occurs, the recovery layer can perform several corrective actions: wait, refresh costmaps, spin to gather new sensor data, back up, and resume normal path planning once the error condition has been handled. This recovery structure is important because it gives the robot a defined response when local planning becomes uncertain or when the costmap contains stale or conflicting information. Rather than stopping permanently after a planning failure, the system attempts to restore usable state information and continue the course.

7.1 Key Performance Indicators

Warrior 1's key performance indicators were selected to measure whether the robot is progressing toward competition readiness for both AutoNav and Self Drive. The KPIs focus on the capabilities most connected to successful **course performance**: legal autonomous speed, lane following, obstacle avoidance, waypoint navigation, Self Drive event handling, real-time perception, and reliable electrical/safety operation.

Category	KPI Description	Technical Target / Spec	Calculated / Measured Value
Endurance	Battery Capacity	36V @ 80 Ah (Total System)	2,880 Wh (Total Energy)
Endurance	Propulsion Runtime	10A draw per module	~120 Minutes (Continuous)
Steering	360° Revolution Time	NEO BLDC (42:1 Ratio)	0.44 Seconds
Steering	Steering Torque	42:1 Reduction	109.2 Nm (Per Module)
Compute	VRAM Allocation	Jetson AGX Orin (32 GB)	~17 GB (NAV + Perception AI)
Compute	Processing Overhead	200 TOPS (INT8)	< 45% CPU/GPU Load
Velocity	Combined Drive Torque	3x 350W Hub Motors	18 Nm (Total at Wheels)
Velocity	Acceleration (0-5 mph)	1,050W Peak Power	~1.4 Seconds

Perception	Sensor Fusion Field	360° LiDAR / 4K Camera	0% Blind Spots (Static)
Safety	Wireless E-Stop Range	Hardware Kill-Switch	> 100 Feet

TABLE 6. Categories and their Targets/Measured Values

The current KPI status shows strong progress in subsystem design and early validation, particularly in electrical stability, safety shutdown, localization testing, and navigation architecture. The largest remaining readiness gaps are final perception benchmarking, outdoor GPS waypoint testing, full AutoNav course testing, and completed Self Drive behavior testing.

8. Analysis of Complete Vehicle

8.1. Integration Lessons and Development Findings

Warrior 1's development has shown that subsystem design and full-vehicle readiness are separate milestones. Meaningful progress has been made in perception, localization, navigation, electrical design, safety, and mechanical analysis, but the major remaining challenge is proving that these systems perform together during final autonomous testing.

Several concrete lessons have already emerged. In perception development, early automated methods were not reliable enough for competition-specific scenes. One initial approach confused the soup sign with the stop sign, and early Hough-transform-style lane processing produced masks that included too much background rather than isolating lane markings cleanly. These issues pushed the team toward manual object annotation, structured dataset generation, and the CLRNet + SAM 2 pseudo-labeling pipeline for lane segmentation.

The localization and navigation teams also created practical development substitutes where full outdoor testing was not yet available. **AprilTag-based GPS spoofing** allowed the localization pipeline and waypoint logic to be tested indoors, while the driving logic was exercised in **Gazebo** and on a mini IGVC-style course to validate path planning, regulated pure pursuit, and recovery behavior before full-scale deployment.

Subsystem	Issue Observed	Root Cause	Corrective Action	Current Status
Perception	Early lane-detection attempts included excessive background instead of clean lane masks.	Hough-transform-style methods did not generalize well to panoramic IGVC imagery.	Adopted a CLRNet + SAM 2 pseudo-labeling pipeline for lane segmentation.	New labeling pipeline implemented; outdoor validation pending.
Localization	GPS-style navigation testing was limited indoors.	Reliable outdoor GPS testing was not always available during development.	Created an AprilTag-based GPS spoofing system for indoor localization and waypoint testing.	Indoor testing completed; outdoor GPS validation still needed.
Driving Logic	Full-scale navigation	Final robot integration lagged	Tested waypoint following, planning,	Navigation architecture

	behavior could not be verified early in development.	software development.	and recovery logic in Gazebo and on a mini IGVC-style track.	exercised; further Warrior 1 testing still needed.

TABLE 7. Subsystem Issues and Resolutions

8.2. Simulation, Physical Testing, and Remaining Vehicle-Level Risks

Simulation and substitute testing environments reduced integration risk by allowing critical autonomy concepts to be tested before the final robot was fully validated. Costmap-based planning, A* path generation, regulated pure pursuit, waypoint following, and behavior-tree recovery logic were all exercised in controlled environments. However, those results still need to be confirmed on the final vehicle, where full-scale drivetrain behavior, real sensor noise, terrain effects, and outdoor GPS conditions will affect performance.

The same distinction applies to localization and perception. AprilTag GPS spoofing confirms that the team can publish GPS-style robot position information indoors and use it for waypoint-development work, but it does not replace outdoor GPS validation. Likewise, the perception pipeline is technically well defined, but its final competition-level accuracy remains incomplete until the model is retrained and benchmarked on representative robot-mounted outdoor data.

9. CYBER SECURITY

9.1 Risk Assessment Approach

Warrior 1's cybersecurity analysis focuses on three attack surfaces that could directly affect autonomous behavior: localization inputs, ROS 2 communication, and onboard compute/software access. These areas were selected because the robot depends on GPS-guided waypoint navigation, sensor-driven costmaps, ROS 2 message exchange, and Jetson-based perception and control processing. A compromise in any of these systems could alter the robot's understanding of its position, its environment, or the commands used to generate motion.

9.2 Key Vulnerabilities and Impacts

The most important cybersecurity risks for Warrior 1 are those that could interfere with its autonomous decision-making. The vehicle depends on localization data to determine where it is, ROS 2 communication to move information between sensing and navigation modules, and onboard software files to control perception and motion behavior. If any of these elements are altered, blocked, or replaced with false information, the robot may respond incorrectly even if its mechanical and electrical systems remain fully functional.

Three vulnerabilities are especially relevant to the current architecture. First, spoofed or corrupted GPS-position data could mislead waypoint navigation and shift the robot's estimated location. Second, unauthorized or abnormal ROS 2 message traffic could disrupt the transfer of perception outputs, costmap updates, or motion commands. Third, unauthorized access to the Jetson computer, model files, launch files, or navigation parameters could alter how the robot interprets

its environment and generates driving behavior. The table below summarizes these vulnerabilities, their possible effects on Warrior 1, and the hardening direction needed to reduce each risk.

Area	Vulnerability	Impact	Mitigation
Localization	Spoofed or corrupted GPS-position input	False position data could mislead waypoint navigation, shift the robot's estimated location, or cause the planner to pursue an incorrect path.	Cross-check GPS against odometry and IMU trends, reject impossible position jumps, and trigger fallback behavior when localization sources disagree.
ROS2/ Navigation	Unauthorized or abnormal ROS 2 message traffic	Injected, stale, or flooded messages could corrupt costmaps, perception outputs, waypoint processing, or motion commands used by the navigation stack.	Restrict which nodes may publish critical topics, monitor message timing/rates, reject stale data, and apply secure ROS 2 communication policies before wider deployment.
Computer/ Software	Unauthorized access to Jetson software, model files, or configuration settings	Modified perception models, launch files, or navigation parameters could change how the robot detects lanes, interprets obstacles, or generates driving commands.	Restrict which nodes may publish critical topics, monitor message timing/rates, reject stale data, and apply secure ROS 2 communication policies before wider deployment.

TABLE 8. Vulnerabilities and their Mitigation