TENNESSEE TECH UNIVERSITY

Team: Slow-Mho 5/16/2025

Team Leader

- Evan Kvalvik — ekvalvik42@tntech.edu

Faculty Advisors / CC

- Mr. Rentschler – memrentschler@tntech.edu
- Dr. HyungJin Yoon — hyoon@tntech.edu
- Dr. Byron Pardue — bpardue@tntech.edu
- Dr. Van Neste – cvanneste@tntech.edu

Team Members

- Keegan Adreon — kcadreon42@tntech.edu
- Ian Cronin — imcronin42@tntech.edu
- Alan Arvidson — aparvidson42@tntech.edu
- Patrick Stultz — pnstultz42@tntech.edu
- Christopher Sullivan — cwsullivan43@tntech.edu
- Mary Bickel – mgbickel42@tntech.edu
- Sarah Boyce – srboyce42@tntech.edu
- Seth Eddins – steddis42@tntech.edu

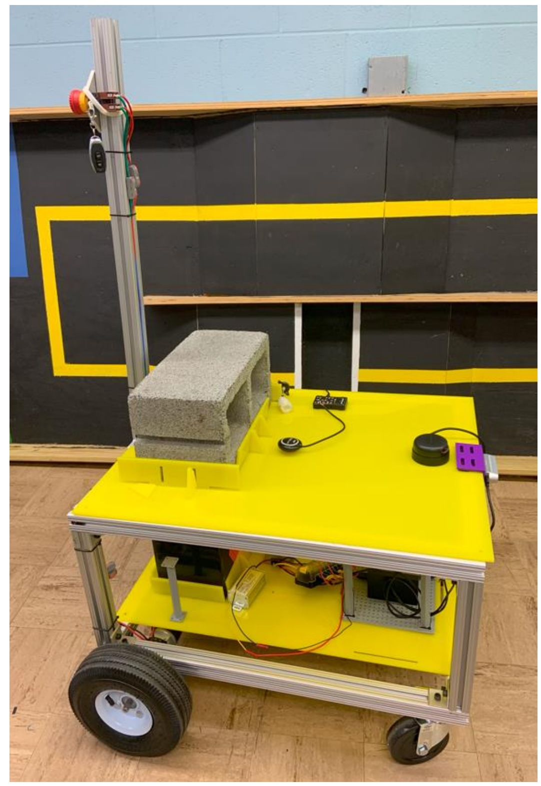


FIGURE 1: SLOW-MHO – AN AUTONOMOUS VEHICLE

Statement of Integrity

I certify that the design and engineering of Slow-Mho – the autonomous vehicle - has been undertaken by the team listed above and that the efforts have met the demands of a senior level design course.

Signature: ____________________ 5/16/2025

Signature: ____________________ 5/16/2025

Table of Contents

Contents

1. Abstract

This project presents the design, construction, and testing of an autonomous ground vehicle developed for the Intelligent Ground Vehicle Competition (IGVC). The system integrates real-time perception, path planning, and autonomous navigation using a suite of sensors including LiDAR, GPS, IMU, and a camera. Emphasis was placed on modular hardware, ROS 2-based software architecture, and efficient debugging practices. Key challenges included sensor fusion, USB stability, and maintaining robust navigation in outdoor environments. Performance metrics such as obstacle detection range, waypoint accuracy, and battery life are evaluated and compared to predictions. Lessons learned in hardware reliability, safety design, and simulation testing informed improvements throughout the development cycle.

2. CONDUCT OF DESIGN PROCESS

2.1 Introduction

Slow-Mho was developed as a multidisciplinary collaboration between Electrical and Mechanical Engineering students to design and prototype an autonomous mobility platform. The project required integrating hardware, embedded software, and mechanical systems, aligning with real-world engineering challenges faced in autonomous vehicle development.

2.2 Organization

The project was carried out by one Electrical Engineering (EE) team and two Mechanical Engineering (ME) teams. The ME teams were both responsible for chassis development. One ME team pursued a short-term solution by adapting an existing wheelchair frame for early testing and development. The second ME team took on a longer-term goal, designing and constructing a custom chassis from the ground up for final integration.

Name	**Major**	**Fields**
Gabriel Gibson	ME	Mechanical
Ian Cronin	ME	Mechanical
Alan Arvidson	ME	Mechanical
Keegan Adreon	ME	Mechanical
Peyton Pope	ME	Mechanical
Josiah Reese	ME	Mechanical
Patrick Stulz	ME	Mechanical
J. W. Beasley	ME	Mechanical
Seth Eddins	CS	Software

Mary Bickel	EE	Software, Electrical
Evan Kvalvik	EE	Software, Electrical
Sarah Boyce-Howard	EE	Electrical
Christopher Sullivan	EE	Electrical

Estimated Total Person-Hours:
Over the course of two semesters, with seven active student contributors and ongoing weekly efforts (averaging 8–12 hours per week per student), we estimate approximately **1,200–1,500 total person-hours** were dedicated to the project.

2.3 Design Assumptions and Design Process

Several assumptions guided our early design efforts:

- The system must operate on uneven outdoor terrain (IGVC-style course), necessitating robust drive and control systems.
- ROS 2 would be the primary software framework for middleware and autonomy features.
- Power systems would be battery-operated, with runtime goals of at least one hour.
- All components must be modular and easily replaceable to facilitate multiple chassis and iterative testing.
- All IGVC requirements would be adhered to.

The design process followed a hybrid V-model and Agile methodology. High-level functional requirements were defined early in the fall semester. Sub teams then performed individual subsystem research, feasibility analysis, and iterative prototyping. Regular sprint-style reviews helped synchronize efforts, track progress, and reallocate resources dynamically based on emerging challenges.

Hardware and software integration was performed progressively, with multiple test phases culminating in a fully integrated outdoor test. Each phase included performance evaluation, bug fixes, and feature updates driven by team feedback and real-world constraints.

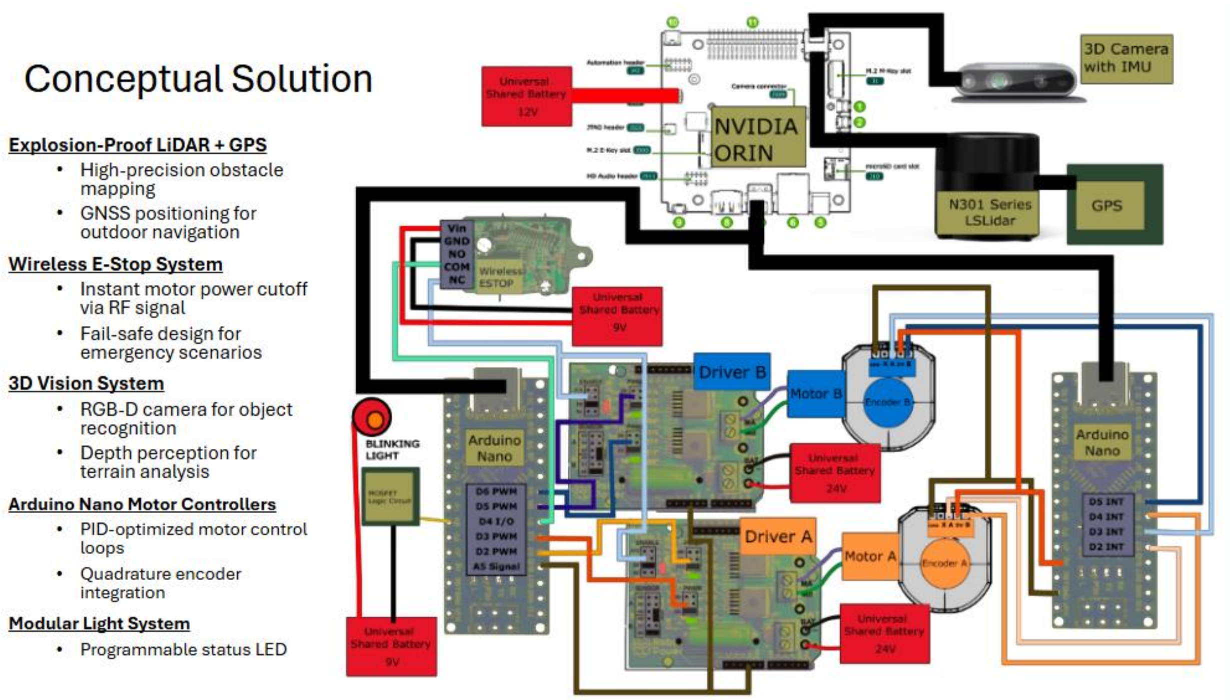


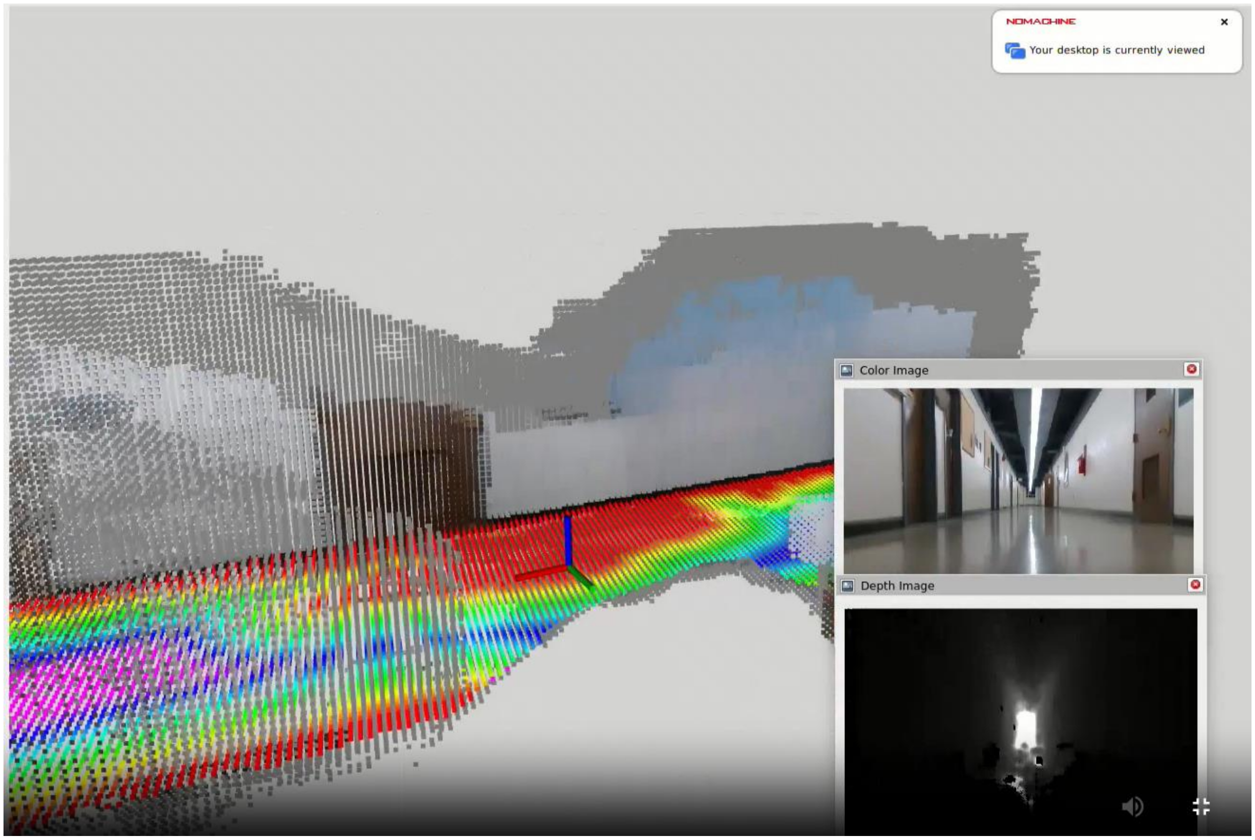
FIGURE 2 THE EARLY STAGES OF THE CONCEPTUAL SOLUTION TO CREATING SLOW-MHO

3. SYSTEM ARCHITECTURE & INNOVATION

The *Slow-Mho* vehicle is built around a modular system architecture that integrates mechanical, electrical, and software components using a ROS 2-based framework. Mechanically, two chassis solutions were developed. One mechanical engineering team retrofitted an existing powered wheelchair to create a temporary platform for early testing, while the second team designed and fabricated a custom aluminum chassis tailored for long-term use. The final chassis was optimized for sensor placement, component mounting, and structural rigidity suited for outdoor autonomous navigation.

The power system centers on a 51.2 V lithium-ion battery with an onboard battery management system. Power is distributed through a custom power distribution board (PDB) that regulates voltage and current to all components, including the motor controllers and computer hardware. For motion, the vehicle uses two MegaMoto Plus motor controllers driven by an Arduino Nano, which receives high-level velocity commands from the Jetson AGX Orin. An emergency stop switch is wired directly into the motor power lines, providing a fail-safe shutdown mechanism in case of unexpected behavior.

The main computation unit is the NVIDIA Jetson AGX Orin, which runs ROS 2 Humble and all the high-level autonomy software. This includes real-time object detection using YOLOv8, 3D environment mapping with NVBlox, and motion planning through the Nav2 stack. The Orin receives perception data from the Intel RealSense depth camera and Leishen 2D LiDAR. A Pixhawk GPS module provides outdoor geolocation data, further supporting the navigation stack. Custom software nodes handle motor command generation, sensor fusion, and interprocess communication between all subsystems.



1. **FIGURE 3 NAVIGATING BROWN HALL WITH ROS2 NAVIGATION**

Subsystems are connected through ROS 2 topics and services. Sensor data flows into the Jetson, where it is processed and passed through the perception and navigation stacks. The resulting control commands are then sent to the Arduino Nano over serial. The Arduino interprets these commands and outputs the appropriate PWM signals to the motor controllers, enabling the robot to execute its planned motion.

Several innovations were introduced over the course of development. One key innovation was the use of a dual-chassis approach. By deploying a temporary wheelchair-based chassis early in the project, the team was able to begin electrical integration and software testing months before the final mechanical frame was ready. This decoupling of the development timeline allowed parallel progress and helped mitigate project risk. Another innovation was the development of a modular motor control interface within ROS 2 that supported both manual and autonomous modes. This flexibility allowed for fast testing, simplified debugging, and safer transitions between development stages. Finally, the integration of NVBlox with the RealSense camera enabled 3D mapping capabilities that far exceeded the spatial awareness typically offered by 2D LiDAR alone. This decision was made after comparing mapping frameworks and recognizing NVBlox's advantages for volumetric occupancy mapping in outdoor environments.

These innovations emerged organically through iterative testing, team discussions, and strategic tradeoffs. As system bottlenecks and testing needs evolved, the team remained agile—adjusting designs and toolchains to stay aligned with project goals and performance requirements.

4. DESCRIPTION OF MECHANICAL DESIGN

The mechanical design of *Slow-Mho* was developed in two distinct phases to accommodate both short-term testing and long-term deployment. Initially, the team repurposed a motorized wheelchair as a temporary chassis. This ready-made platform provided a reliable drivetrain and basic frame structure, allowing the electrical and software teams to begin development and integration while the final chassis was still in progress. Although the wheelchair's internal components were not modified, its frame was adapted with mounting plates and custom brackets to support onboard electronics and sensors.

In parallel, a second mechanical engineering team worked on the design and fabrication of a custom long-term chassis. This process began with a detailed requirements analysis, followed by multiple design iterations in Computer-Aided Design (CAD) software. Autodesk Inventor was used to model the frame, estimate material stresses, and verify clearances for key components such as batteries, motor controllers, and compute units. The final design was constructed using aluminum square tubing for its strength-to-weight ratio, corrosion resistance, and ease of machining.

Key structural decisions included isolating the electronics in a central compartment to protect them from mechanical vibrations and routing power and signal lines through protected channels along the frame. While the wheelchair platform featured built-in suspension elements, the custom chassis was rigid by design, as initial tests suggested minimal benefit from active suspension due to the vehicle's relatively low speed and predictable course environment. Any minor vibration was mitigated through the use of rubber grommets and vibration-dampening pads beneath sensitive electronics.

Weatherproofing was approached pragmatically. While *Slow-Mho* is not intended for use in heavy rain, several passive protection strategies were implemented. All critical electronics were mounted within sealed plastic enclosures or weather-resistant cases, with gaskets and cable glands to prevent moisture ingress. Ventilation holes were added only where necessary and positioned to reduce the likelihood of splash exposure. Exposed wiring was enclosed in split loom tubing and tied down to prevent movement during travel.

Throughout the design process, the mechanical team worked closely with electrical and software sub teams to ensure all components could be securely and accessibly mounted. Mounting points for the camera, LiDAR, GPS, and E-Stop were all integrated into the CAD model and confirmed through physical mock-ups before final fabrication. The result was a robust and maintainable structure that could support field testing and future component upgrades.

5. Electronics and Power

The power subsystem for *Slow-Mho* was designed to provide reliable and regulated electrical energy to all components during operation. A key function of this subsystem is to ensure the vehicle can complete the competition track without power failure, which required selecting a battery with enough capacity and voltage headroom, as well as appropriate voltage converters to deliver clean power to each subsystem. To this end, the team selected a 51.2V, 25Ah LiFePO4 battery, which offers over 1,280Wh of capacity—more than enough to support the system's maximum estimated power draw of 540W for well over two hours. This battery was also chosen for its timely availability, relatively low weight (23.6 lbs), and compact form factor, making it suitable for integration within the mechanical constraints of the vehicle.

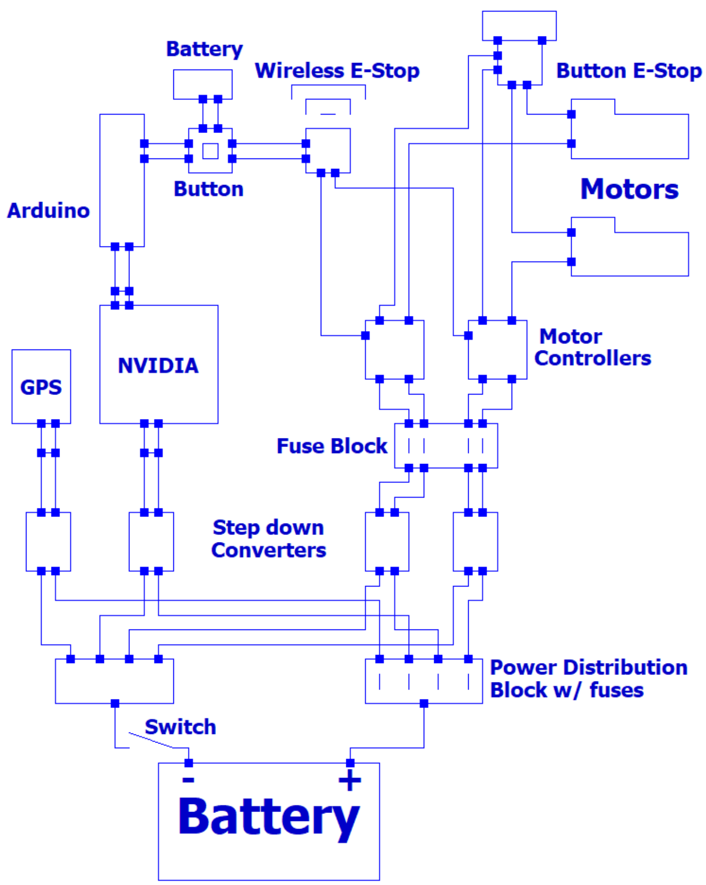


FIGURE 4 POWER SUBSYSTEM CONFIGURATION

The DC-DC power regulation system is built around two types of converters. The vehicle's main compute and control units operate on 12V, so step-down converters were used to transform the 51.2V battery voltage to 12V at both high and low current levels. Specifically, two 12V/20A converters power the motor controllers, while one 12V/10A converter supports the NVIDIA Jetson AGX Orin and its peripherals. The Arduino Nano is powered directly via the Jetson's USB port. The converters were chosen for their input voltage tolerance (up to 60V) and current capacity, ensuring headroom and stability under varying loads. Additionally, the subsystem includes a physical E-stop button and in-line fuses on the return paths from the motors to meet safety requirements.

To ensure safe and efficient power delivery, wire gauges were carefully selected based on expected current. Team 4 utilized 6 AWG wiring for battery connections, 10 AWG for up to 20A circuits, 16 AWG for circuits below 10A, and 20 AWG for signal-level connections, such as between the Arduino and motor drivers. All wires were insulated to provide protection from environmental exposure and unintentional shorting during handling. Terminal blocks, fuse holders, and fork terminals were used to securely connect all components and simplify assembly and troubleshooting. Together, the battery, converters, and wiring form a robust and modular power delivery system that interfaces cleanly with the mechanical and software subsystems, supporting stable and safe vehicle operation.

6. Description of software system

The software system for *Slow-Mho* was designed using a modular, ROS 2-based architecture running on the NVIDIA Jetson AGX Orin. Its core function is to perceive the environment, classify objects, plan safe trajectories, and execute those trajectories to complete the competition course. Sensor data is collected from multiple modalities: a RealSense depth camera, a 2D LiDAR, GPS via Pixhawk, and wheel encoders. These raw data streams are processed by specialized nodes for object detection, localization, and mapping.

Sensor fusion is performed through synchronized timestamping and ROS 2 message filters, allowing the system to correlate depth data from the RealSense camera with 2D scans from the LiDAR and geolocation data from the GPS. This enables *Slow-Mho* to generate a more complete understanding of its surroundings—referred to as the **current scene**. The RealSense camera is also used to run a YOLOv8 object detection model in real time, which identifies cones and other IGVC-relevant obstacles. This classification data is incorporated into NVBlox, a volumetric mapping system that uses depth images and odometry to incrementally build a 3D map of the environment—effectively combining current and past scene data into a **world model**. This allows the robot to reason not only about what it sees now, but what it has seen before and avoid obstacles accordingly.

FIGURE 5 IMAGE RECOGNITION WITH YOLOV8

Navigation is handled through the ROS 2 Navigation Stack (Nav2), which ingests this world model, localizes the robot, and generates global and local paths using a hybrid approach of GPS waypoints and obstacle-aware path planning. When operating in GPS-following mode, the robot uses GPS coordinates as goals and fuses them with odometry to localize itself on a coarse map. In lane-following mode, the system instead relies on visual cues—like cones or boundaries—detected by YOLOv8 to generate a trajectory through NVblox and Nav2. Mode switching is triggered based on mission state: GPS mode is used for long-range travel between course segments, while lane-following activates when the system detects navigation boundaries, such as colored cones or white circle “pothole simulators.”

The planned trajectory is followed by computing target velocities for the drive motors and publishing motion commands to a custom motor control node. This node translates the high-level trajectory into PWM commands sent via serial to an Arduino Nano, which drives the MegaMoto motor controllers. Feedback from encoders and system monitoring (e.g., watchdog timers, heartbeat messages) ensure that motion adheres to the plan and safety overrides can be triggered if necessary.

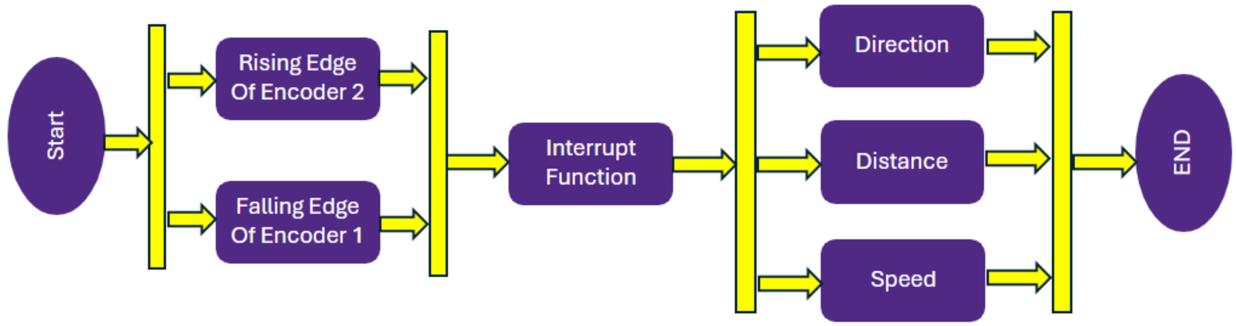


FIGURE 6 ENCODER TEST DATA

In addition to standard functionality, *Slow-Mho* incorporates creative design choices to improve robustness. For instance, a fallback manual mode allows for testing using keyboard inputs, and a modular control interface permits easy swapping of autonomy modules without changing the rest of the system. Overall, the software system combines state-of-the-art perception and control tools in a scalable, fault-tolerant framework suited for real-world autonomous navigation.

7. Cyber Security Analysis using RMF

To understand and mitigate cybersecurity risks for *Slow-Mho*, we applied the NIST Risk Management Framework (RMF), which outlines a structured, seven step approach: Prepare, Categorize, Select, Implement, Assess, Authorize, and Monitor. This framework ensures that cybersecurity is treated as a system-wide concern rather than a last-minute add-on.

Threat modeling revealed the most significant risk as malicious actors—such as rival teams—disrupting software via USB access or wireless interference while competing. Such attacks could involve overwriting configuration files, injecting false commands, or disabling safety systems. The impact could be catastrophic: loss of control, inability to start, or unpredictable behavior resulting in disqualification.

To counter the mist likely threat, we selected the following cybersecurity controls:

- Physical Access Controls: USB ports will be closely monitored.
- User Authentication: Password-protected access on the Jetson Orin.
- Coded Remote Cut-off: The remote cutoff trigger provides a basic level of security.
- Lidar/GPS spoofing: While Lidar or GPS spoofing is possible, those threats are in their early stages of development. This team has no defense against such an attack, but the likelihood of that kind of attack has been deemed very unlikely.

Future improvements include adopting secure ROS 2 DDS transport layers, intrusion detection agents, and prevention of spoofing.

8. Analysis of Complete Vehicle

Lessons Learned During Construction and System Integration

One important lesson we learned is to never delete seemingly useless files, as they often turn out to be critical later. It's also wise to avoid troubleshooting hardware and software problems at the same time, since isolating issues is much easier when focusing on one domain. We found it's essential to record working configurations while everything is still functioning, so there's a reference point when things break. If you're stuck, chances are someone else has encountered the same problem—so do some research before reinventing the wheel. We also realized that one semester is not enough time to fully learn programming and robotics, especially when building a system from scratch. Finally, we learned the hard way that anything that happens in the container stays in the container—containerized environments can mask problems or changes unless carefully documented.

Failures, Effects and Strategies of Mitigation

The following is a table of some documented issues that arose either before or after getting a specific component to function and how the team overcame the failure.

Failure	Effect	Mitigation Strategy
CH340 USB serial disconnects	Loss of motor control	Switched to shielded cables, verified kernel module installation
Power rail voltage drops	System resets	Added decoupling capacitors, switched to BECs with voltage regulation
GPS loss or noise	Navigation drift	Implemented dead reckoning with IMU fallback and added watchdog for GPS health
RealSense camera instability	Loss of depth mapping	Upgraded USB cables, ran diagnostics in RViz, added fallback logic

Budget

Component	Quantity	Cost to Team
51.2V/25Ah Battery	1	$269.99
Battery Charger	1	$35.00
BSS Fuse Block	1	$31.55
BSS Fuses	4	$13.64
ANL Fuse Block	1	$19.99
ANL Fuses	1	$4.99
12V/20A Converters	2	$37.98
12V/10A Converters	2	$24.66
Wire to Power Jacks	1	$6.19
Terminal Blocks	1	$11.49
Wire Terminals	1	$13.59
Nvidia Orin Jetson Nano	1	$0
Nvidia SSD	1	$44.60
Arduino Nano	1	$24.12
Encoders	2	$0
Encoder Cables	2	$21.00
12V Motors	2	$99.50
Motor Gearboxes	2	$242.00
Motor Drivers	2	$159.54
3D Camera	1	$295.42
2D LiDAR	1	$0
Physical E-stop	1	$32.00
Wireless E-stop	2	$23.98
Light	1	$12.23
Wires of Various Gauges	4	$0
Total		$1,423.46

References

[1] A. Tantos, “H-Bridge Secrets,” *Modular Circuits*. [Online]. Available: https://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridge_drivers/
[2] Jameco Electronics, “Product Datasheet: 2204023MAN,” *Jameco Electronics*. [Online]. Available: https://www.jameco.com/Jameco/Products/ProdDS/2204023MAN.pdf
[3] Dayton, “DC Gearmotor, RPM 150, 24VDC, 5LAG5,” *Zoro*. [Online]. Available: https://www.zoro.com/dayton-dc-gearmotor-rpm-150-24vdc-5lag5/i/G0897757/
[4] Tutorials Point, “Rolling Friction and Rolling Resistance,” *Tutorials Point*. [Online]. Available: https://www.tutorialspoint.com/rolling-friction-and-rolling-resistance
[5] Unitree Robotics, “Official Website,” *Unitree Robotics*. [Online]. Available: https://www.unitree.com/
[6] Robotics Backend, “Raspberry Pi (master) Arduino Uno (slave) SPI communication with WiringPi,” *Robotics Backend*. [Online]. Available: https://roboticsbackend.com/raspberry-pi-master-arduino-uno-slave-spi-communication-with-wiringpi/
[7] Newbiely, “Raspberry Pi - DC Motor,” *Newbiely Tutorials for Beginners*. [Online]. Available: https://newbiely.com/tutorials/raspberry-pi/raspberry-pi-dc-motor

[8] Arduino Stack Exchange, "Arduino Nano use SPI pins as I/O pins? and still be able to do ISP?," *Arduino Stack Exchange*. [Online]. Available: https://arduino.stackexchange.com/questions/73433/arduino-nano-use-spi-pins-as-i-o-pins-and-still-be-able-to-do-isp
[9] Hackatronic, "Raspberry Pi 5 Pinout, Specifications, Pricing: A Complete Guide," *Hackatronic*. [Online]. Available: https://www.hackatronic.com/raspberry-pi-5-pinout-specifications-pricing-a-complete-guide/
[10] ThingsDAQ, "Encoder with Raspberry Pi," *ThingsDAQ*. [Online]. Available: https://thingsdaq.org/2022/03/09/encoder-with-raspberry-pi/
[11] Jameco Electronics, "Official Website," *Jameco Electronics*. [Online]. Available: https://www.jameco.com/
[12] Dynapar, "Quadrature Encoders - The Ultimate Guide," *Dynapar*. [Online]. Available: https://www.dynapar.com/technology/encoder_basics/quadrature_encoder/
[13] Total Phase, "I2C vs SPI Protocol: Differences and Similarities," *Total Phase Blog*, Jul. 2021. [Online]. Available: https://www.totalphase.com/blog/2021/07/i2c-vs-spi-protocol-analyzers-differences-and-similarities/
[14] Same Sky Devices, "AMT10 Series Modular Incremental Encoders," *Same Sky Devices*. [Online]. Available: https://www.sameskydevices.com/product/motion-and-control/rotary-encoders/incremental/modular/amt10-series
[15] F. Anwar, "Comparison of the Scalability and Performance of ROS1 and ROS2 Systems," *LinkedIn*, Jan. 2023. [Online]. Available: https://www.linkedin.com/pulse/comparison-scalability-performance-ros1-ros2-systems-fouad-anwar/
[16] ROS 2 Design, "Design Changes from ROS 1 to ROS 2," *ROS 2 Design*. [Online]. Available: https://design.ros2.org/articles/changes.html
[17] DDS Foundation, "What is DDS?," *DDS Foundation*. [Online]. Available: https://www.dds-foundation.org/what-is-dds-3
[18] University of Michigan Robotics, "ROB599-F19: Class 18," *University of Michigan Robotics*. [Online]. Available: https://robotics.umich.edu/academics/courses/online-courses/rob599-f19/class18
[19] ROS 2 Design, "ROS with ZeroMQ," *ROS 2 Design*. [Online]. Available: https://design.ros2.org/articles/ros_with_zeromq.html
[20] IGVC, "2025 Official Rules," *Intelligent Ground Vehicle Competition*. [Online]. Available: https://igvc.secs.oakland.edu/2025rules.pdf
[21] IGVC, "Reports Archive," *Intelligent Ground Vehicle Competition*. [Online]. Available: https://igvc.secs.oakland.edu/reports.htm
[22] Raspberry Pi Forums, "Topic: 361746," *Raspberry Pi Forums*. [Online]. Available: https://forums.raspberrypi.com/viewtopic.php?t=361746
[23] ROS 2 Documentation, "Understanding ROS2 Topics," *ROS 2 Documentation*. [Online]. Available: https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding-ROS2-Topics.html
[24] ROS 2 Documentation, "Sync vs Async," *ROS 2 Documentation*. [Online]. Available: https://docs.ros.org/en/foxy/How-To-Guides/Sync-Vs-Async.html
[25] PowerStream, "Wire Size Calculator," *PowerStream*. [Online]. Available: https://www.powerstream.com/Wire_Size.htm
[26] Mean Well, "SCWN06 Series Datasheet," *Mean Well*. [Online]. Available: https://www.meanwell-web.com/content/files/pdfs/productPdfs/MW/SCWN06/SCWN06,DCWN06-spec.pdf
[27] OSHA, "Standard Interpretations: 2013-06-04," *Occupational Safety and Health Administration*. [Online]. Available: https://www.osha.gov/laws-regs/standardinterpretations/2013-06-04-0
[28] Adafruit, "Product 4517," *Adafruit*. [Online]. Available: https://www.adafruit.com/product/4517
[29] Raspberry Pi, "Raspberry Pi M.2 HAT Plus Schematics," *Raspberry Pi*. [Online]. Available: https://datasheets.raspberrypi.com/m2-hat-plus/raspberry-pi-m2-hat-plus-schematics.pdf
[30] Arduino, "Arduino Nano," *Arduino*. [Online]. Available: https://store.arduino.cc/products/arduino-nano
[31] Cerrowire, "Applications Charts," *Cerrowire*. [Online]. Available: https://www.cerrowire.com/products/resources/tables-calculators/applications-charts/
[32] AltE Store, "Wire Sizing Chart for 12V, 24V, and 48V DC Systems," *AltE Store*. [Online]. Available: https://www.altestore.com/pages/wire-sizing-chart-for-12v-24v-and-48v-dc-systems
[33] East Coast Overland Adventures, "VES Distribution," *East Coast Overland Adventures*. [Online]. Available: https://eastcoastoverlandadventures.com/2022/05/ves-distribution.html
[34] Enerdrive, "Select a Fuse and Fuse Holder for Your DC Product Installation," *Enerdrive*. [Online]. Available: https://enerdrive.com.au/2018/02/06/select-a-fuse-and-fuse-holder-for-your-dc-product-installation/