UNIVERSITY OF BRIDGEPORT

Faculty Advisor: Dr. Ahmed El Sayed

Team Captain: Jun Zhang (zhangjun@bridgeport.edu)

Date Submitted: May 15, 2025

STATEMENT OF INTEGRITY:

I certify that the design and engineering of TrUBot by the current listed student team has been significant and equivalent to what would be awarded credit in an independent study course at the University of Bridgeport.

Ahmed ElSayed

Team Members:

Andrew Iorio (aiorio@my.bridgeport.edu)

Scott Goodman (scgoodma@my.bridgeport.edu)

Anass Saoudi (asaoudi@my.bridgeport.edu)

Ali Hamadeen (alihamad@my.bridgeport.edu)

Rudra Mitra (rmitra@my.bridgeport.edu)

Lee Donovan (donovanl@student.goodwin.edu)

Table of Contents

1. Introduction

TrUBot is an autonomous ground vehicle designed and developed to compete in the Intelligent Ground Vehicle Competition (IGVC). The robot embodies a robust fusion of real-time perception, decision-making, and motion control systems, engineered to navigate complex outdoor environments with high reliability and adaptability.

This marks the first time researchers from the LACASA (Laboratory of Advanced Control, Autonomous Systems, and Automation) at the University of Bridgeport will participate in the IGVC. TrUBot represents the lab's commitment to advancing research in intelligent robotics and real-world autonomous mobility. The project serves both as a technical milestone and a platform for applying state-of-the-art algorithms and systems integration in a competitive, outdoor robotics environment.

2. Organization

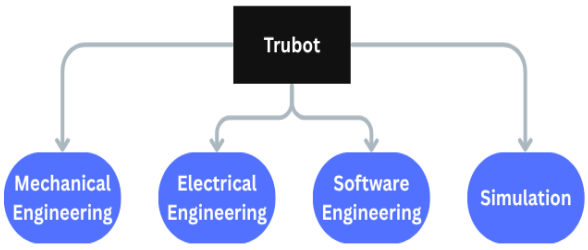


Figure 1 Organization

The TruBot team at the University of Bridgeport is composed of students from a variety of academic disciplines. Despite the team's small size, comprising only six members, they initiated the project in early February. The TruBot team encompasses mechanical engineers, software engineers, electrical engineers, security engineers, and simulation engineers. Given the constrained timeline, each member collaborated across disciplines to provide support to other teams in a peer-review capacity, thereby ensuring that each component of the design was completed to the highest standard. This systematic approach guaranteed that at least one individual reviewed each phase of the system life cycle for implementation, with an additional reviewer or tester involved in the process. Consequently, the team successfully ensured that all project aspects were completed within a three-month timeframe.

3. Design Process

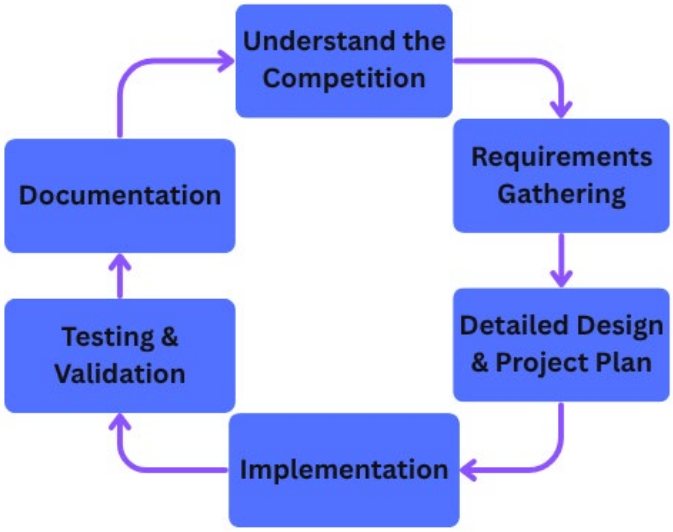


Figure 2. Design Process Workflow

3. 1 Autonomous Robotic Body Design

The initial design plan has not changed much but the addition of other humans joining the fabrication team refined every detail of the building. The pointed front of the X Y axis of the chassis was intended for a very quick obstacle avoidance maneuver. The two wheel design with the wheels as far back as would allow and the payload and batteries centrally located also remained from the original design. Staying away from multiple moving or combined parts was achieved by c and c design for the base of the robot. Once the single piece of steel was laser cut, the work of the build really began. The original design of the wheels was to have the axle supported on both sides like a wheel barrel to keep the weight of the vehicle off the axle. We achieved this goal by placing a pillow block with ball bearings and a bore hole equal in diameter to the axle and the wheels bore holes. The pillow blocks are placed on either side of each wheel and bolted to the chassis. After attempting several different width and diameter wheels we ended up with a seven-inch diameter by a one- and three-quarter inch wide plastic wheels. The difficulty of finding a wheel with an eight-millimeter bore hole to match the diameter of the motor axle. It was very difficult to find the correct wheels without a ball bearing surrounding the bore hole. This difficulty was overcome by plugging the half inch bore hole of the wheel that worked and re drilling the center at eight millimeters.

The very first obstacle that was presented was the connection between the motor and the encoder as well as the wheel. The idea the team was working with initially was to connect the motor and encoder by chain. This proved extremely problematic, and as a result a gear box was considered and planned. After much deliberation and redrafting this also was rejected in favor of a through design by using a motor that has an embedded encoder with the wheel connected directly to the shaft of the motor/encoder. Payload delivery is the mission and without dropping it as a priority a fully enclosed storage compartment was the original design. One team member took this concept to the next level by building a steel mesh cage that is bolted closed. The floor of the chassis is covered in non-conductive material that will protect the possibility of shorts.

Motor covers were designed and built as well as a steel arch to house the stop button and the operation lights. A cage was constructed to represent the body that will be filled out by plastic coverings.

4. Hardware

TrUBot is built on a robust and modular hardware platform designed for reliable operation in outdoor environments, meeting the performance and safety requirements of IGVC 2025. The hardware architecture balances powerful computation, precise motor control, and real-time sensing in a compact footprint. Below is a detailed description of the main hardware components:

4.1 Drive System

- **Motors:** Two 12V high-torque brushed DC motors (100 kg/cm torque) with embedded quadrature encoders are used for differential drive control.
- **Motor Controller:** A SmartDrive Duo dual-channel motor driver manages motor power and direction using PWM and direction signals from the microcontroller. It supports feedback from the motor encoders for speed estimation.

4.2 Sensing and Perception

- **LiDAR:** The robot is equipped with an RPLiDAR A1 360° laser scanner to detect obstacles and perform real-time environment mapping.
- **Depth Camera:** An Intel RealSense D435i depth camera provides RGB-D perception and includes an integrated IMU (accelerometer and gyroscope), which is used for orientation estimation and sensor fusion.
- **IMU:** The embedded IMU from the D435i is fused with encoder feedback on the ROS 2 side for improved localization and odometry.
- **GPS Module:** A high-precision GNSS receiver is used to determine global position. The GPS data is fused with other localization inputs and plays a critical role in mission-level planning, such as defining and tracking global goal coordinates during the Navigation Challenge of the IGVC.

4.3 Computation and Control

- **Microcontroller:** An **Arduino Portenta H7** mounted on its official **Portenta Breakout Board** handles low-level control, including:
  - Motor speed control.
  - Encoder feedback processing.
  - Light tower control via relay.
  - RC receiver input.
  - Publishing velocity and encoder data via **micro-ROS**.
- **Onboard Computer:** An **Acemagic M2A mini-PC** runs ROS 2 (Humble), managing high-level perception, planning, and decision-making. It connects to the Portenta H7 over USB using micro-ROS serial communication.

4.4 Power Distribution

- **Batteries**: Two 12V sealed lead-acid batteries provide power for motors and electronics. Each subsystem is protected and powered through a central Power Distribution Board.

- **DC-DC Converters**:
  - A 12V to 5V step-down converter powers the Portenta H7.
  - A 12V to 19V boost converter supplies regulated power to the onboard computer.

4.5 User Interface and Safety

- **Start Button:** A physical push button is mounted on the chassis to manually activate the robot.
- **Remote Kill Switch:** A Flysky FS-i6S radio transmitter and receiver allow operators to remotely stop the robot during testing or competition runs.
- **Emergency Stop:** A large emergency stop button disconnects the motor power in critical situations.
- **Status Indicator:** A multi-color light tower controlled by the Portenta via a relay module indicates system status:
  - Green: Autonomous Mode.
  - Red: Emergency Stop.
  - Yellow: Remote Control Mode.

5. Electrical

5.1 Power Supply Overview

The robot's electronic and power system is designed to ensure reliable and isolated operation of both computational and actuation subsystems. The architecture employs **two independent 12V batteries**, each dedicated to a specific subsystem:

- **Battery 1:** Supplies power to the onboard computer.
- **Battery 2:** Supplies power to the drive motors via a dual-channel motor controller.

This separation provides both **power isolation** and **electrical noise reduction**, improving system stability and reliability during high-load operation.

5.2 ACEMAGIC Computer Power

The onboard computer responsible for high-level processing, perception, and control tasks is the **ACEMAGIC mini PC**, which requires a **19V input** for operation. To adapt the 12V battery output to this requirement, a **DC-DC boost converter (12V to 19V)** is used.

- **Input:** 12V from Battery 1.
- **Output:** 19V regulated to power ACEMAGIC.
- **Purpose:** Ensures stable operation of the computational unit regardless of battery voltage drops during runtime.

This design isolates the computing unit from motor-induced voltage spikes, common in mobile robotic platforms.

5.3 Motor Power and Control

The robot uses **two 12V DC motors** for differential drive locomotion. These motors are directly powered by **Battery 2** and are controlled using a **SmartDriveDuo-30 motor controller**, which supports bidirectional speed and direction control.

- **Input Power:** 12V from Battery 2.
- **Control Signals:** Serial or PWM commands from the ACEMAGIC mini PC.
- **Motor Driver:** SmartDriveDuo-30 (supports 30 amp per channel).

The motor controller receives **desired velocity commands** from the ACEMAGIC via a microcontroller (e.g., Arduino Portenta or Due running micro-ROS). These commands are typically based on /cmd_vel messages published from the robot's ROS 2 control stack. The controller then drives the motors with appropriate PWM signals to match the desired speed.

5.4 Safety Devices and Integration

To ensure safe operation during development, testing, and autonomous missions, the robot incorporates multiple **safety mechanisms**, both electronic and mechanical. These systems are designed to provide clear operational status feedback and allow for immediate intervention in case of emergencies.

Safety Light Tower

A **multi-color safety light tower** is installed on the robot to indicate its operational state visually:

- **Green Light (Solid):** The robot is powered on and in manual or standby mode.
- **Yellow Light (Flashing):** The robot is actively running in **autonomous mode**.
- **Red Light (Flashing):** Indicates a fault, emergency stop, or manual intervention required.

This real-time visual feedback allows nearby operators to quickly assess the robot's current mode and respond accordingly.

5.5 Mechanical Emergency Stop Button

In addition to the remote E-Stop, the robot is equipped with a physical emergency stop button mounted on its chassis. Pressing this button:

- Instantly cuts off power to the motor controller or disables motor output.
- It is designed to be easily accessible and robust, even with gloves.
- Redundant with the RC stop for enhanced safety.

E-Stop via RC (Remote Emergency Stop)

A wireless emergency stop (E-Stop) is implemented using an RC (radio control) system. If the operator detects unsafe behavior or an imminent collision, they can immediately cut off the motor commands via RC.

- Mechanism: Interrupts command signals sent from the PC to the motor controller.
- Range: The FlySky FS i6S has an impressive range, offering up to 1 kilometer (0.62 miles) in optimal conditions.

This provides reliable and remote safety overriding during the competition or in uncontrolled environments.

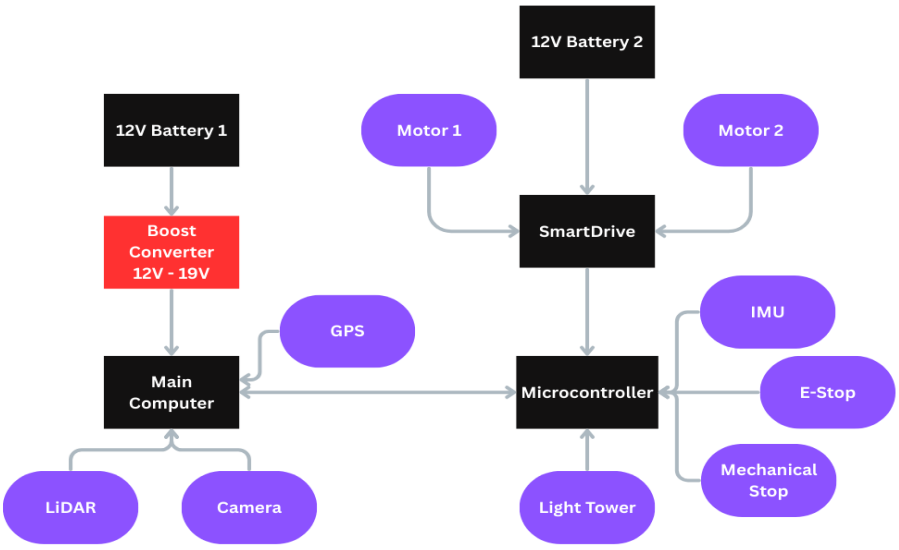


Figure 3. Schematic Layout

6. Software

6.1 Lane following

Lane following was implemented using the raw RGB camera image. The detection logic resides in the detect_lane() function within the Fusion-2.py script. This function accepts the image as a required input and includes eight optional hyperparameters that can be tuned for different environments. The lane detection process begins by thresholding the image in the HSV color space to isolate white and yellow lane markings. The resulting binary mask undergoes morphological operations and Gaussian blurring to reduce noise and close small gaps. Canny edge detection is then applied to extract lane edges, followed by the Hough Line Transform (cv2.HoughLinesP) to detect continuous lane segments.

This custom pipeline was chosen over the YOLOP (You Only Look Once for Panoptic Driving Perception) based approach due to its improved performance in preserving lane continuity, especially around curves and intersections. YOLOP tended to leave gaps in the lane mask, whereas this method produced more consistent and reliable line detection in the team's Webots simulation environment.

(The green lines indicate the lanes highlighted below for the lane following.)

Figure 4. Lanes and Path Planning

6.2 Mapping and localization

The mapping consists of four sensors and three key components. The four sensors include an IMU (Inertial measurement unit) for position and orientation, a GPS (Global Positioning System) for waypoint location, a LiDAR (Light Detection and Ranging) for local object detection, and a camera for lane, object, and cost detection. The three components are localization of the current x, y, and theta of the vehicle, goal point detection, and mapping, which utilizes RRT* to create the waypoints in the map.

Localization forms the core logic within a class named RobotLocalization. The code combines several references taken from https://www. youtube. com/watch? v = J 77 kNrfYKoE and https://medium. com/@ zillur- rahman/how-to- use- the- ros- robot- localization- package- 534 fe04014d3. This class utilizes the left and right motors, as well as the IMU, to estimate the autonomous car's position using differential drive and encoder readings. For each loop, the message is provided on the Fusion node; however, it can also be implemented on the Odometry node if desired. The sample for the team's Webots simulation is "X: 478.93 Y: 719.24 heading: 0.83". From the perspective of the camera, this corresponds to pixel positions 478 and 719, with a heading at an angle along the x- axis of 1 degree. The localization will publish the x, y, and theta, representing the direction the car is heading.

The second and subsequent key component of map generation is goal generation. The objective of goal generation is to identify a point, either near or distant, that the mapping should strive to achieve. Within the Fusion-2.py code, a method named get_next_goal_with_lane is referred to. It can utilize server key components from the lane, LiDAR, and camera maps. As depicted in the image below, which illustrates object detection (object detection will be further discussed in Section 6.3) and lane detection, the lane detection algorithm was able to identify lanes using the Hough lines P to detect the yellow and white lines, allowing for the detection of lanes around corners and "S" curves. The LiDAR is used to help navigate obstacles near the vehicle, such as cones and barriers.

The final component is path planning using the Rapid-exploring Random Tree Start (RRT *). The planning logic implement code is implemented in the RRT _ Star. py, which was adapted https://github.com/AtsushiSakai/PythonRobotics/blob/master/PathPlanning/RRTStar/rrt_star.py. RRT* was selected due to its unique adaptability and ability to handle both static and dynamic obstacles. However, during testing, the code was modified and improved to enhance its performance. During testing, the map from the Webots simulation is shown below.

This is a sample taken from the starting point to the goal point, as a green polyline connecting the robot's start and goal points in figure 4 above.

6.3 Object Detection

It's extremely light design and real-time inference capability on embedded systems led us to choose YOLO11-Nano (YOLO11n), which is perfect for the traffic cones detection. Our main goal was to add traffic cones—objects missing from its original COCO-trained model—to YOLO11n's perceptual repertoire so that our robot could consistently identify cones. We therefore obtained a committed traffic-cone dataset from Roboflow consisting of more than 2,000 YOLO-formatted images with normalized bounding-box annotations. At the same time, we put together a YOLO-formatted COCO128 subset, including the original 80 COCO classes. All cone annotations (originally marked as class 0) were remapped to class index 80 before merging, and our class list was expanded to 81 entries; file names were prefixed with "coco_" or "cone_" to prevent conflicts. Then we created a YAML configuration file defining train, validation, Number of classes 81, and the full class names by combining image and label directories into single training and validation folders. Using transfer learning, we retrained YOLO11n on this combined dataset for 100 epochs at 640×640 with 16 batch size. The retrained model detected traffic-cone class. Figure 1 shows the traffic-cone sample distribution per training batch; Figure 2 shows qualitative inference findings on IGVC course video validating effective cone detection.

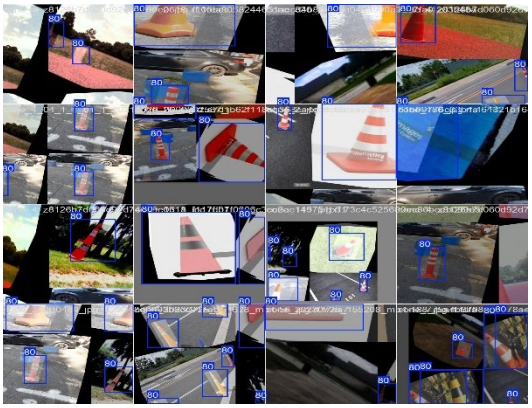


Figure 5. Traffic cone Training Batch

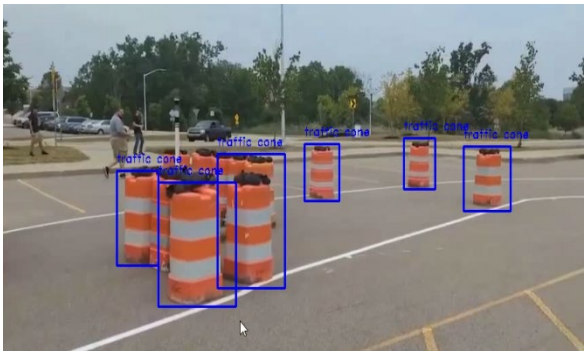


Figure 6. Traffic cone detection

7. Simulation

7.1 Overview

this chapter represents the process of constructing a high-fidelity virtual environment in Webots simulation for the IGVC 2025 competition held in Parking Lot 37 as we can see it in figure.

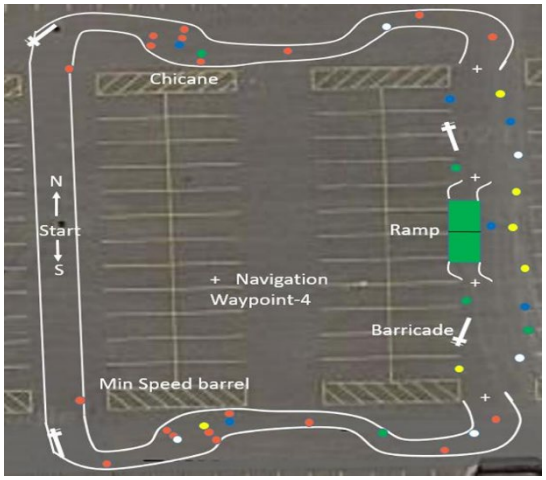


Figure 7. IGVC 25 Auto-Nav Course

7.2 Webots Simulation

Webots is an open source and multi-platform desktop application used to simulate robots. It provides a complete development environment to model, program and simulate robots.

7.2.1 Webots Features

We used Webots R2025a because it has the following features :

Direct interaction with OpenStreetMap allows genuine geographic locations to be converted into 3D map, therefore preserving accurate topology and layout. As we can see in the following figure, we used OpenStreetMap to take the real map for the competition. After cropping the area, the saved OSM file that can convert into Webots world using World Generation.

python importer.py --input=IGVC.osm --output=IGVC.wbt

to produce IGVC.wbt, representing Oakland parking lot 37.

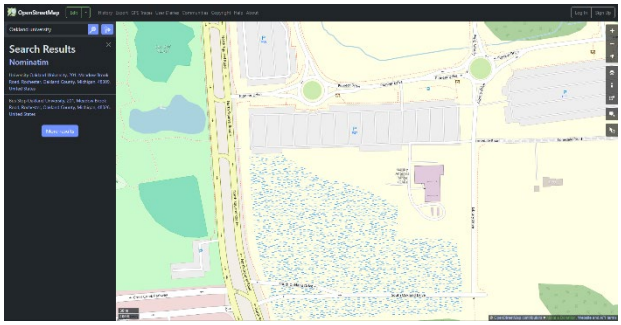


Figure 8. OpenStreetMap (parking lot 37)

The Flexible Scene Graph provides a node-based architecture (PROTO and Solid nodes) for adding, removing, and modifying scene components including lanes, cones, obstacles, and ramps without starting over from scratch.

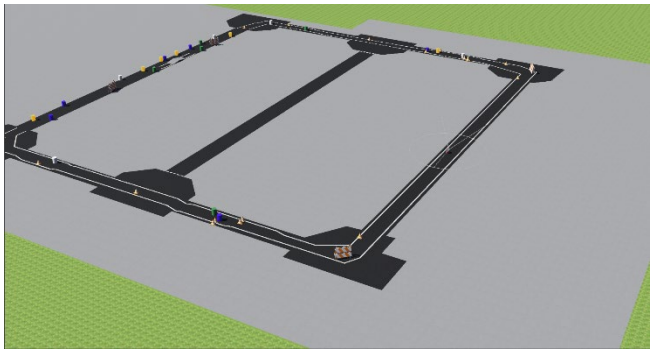


Figure 9. IGVC course on webots simulation

High fidelity integrated physics engine (ODE) and a strong sensor library **(camera, LIDAR, IMU, GPS)** provide realistic interactions and data collecting.

This 3D simulation was very helpful for testing the object detection and lane following models before testing it in the real world course.

We can utilize standard **ROS 2** topics, services, and transforms directly within the simulator thanks to its smooth integration with ROS 2. utilizing the webots_ros2 library.

7.2.2 Course & Robot Customization

To make our simulation the same as the real competition course with our robot, we add the following：

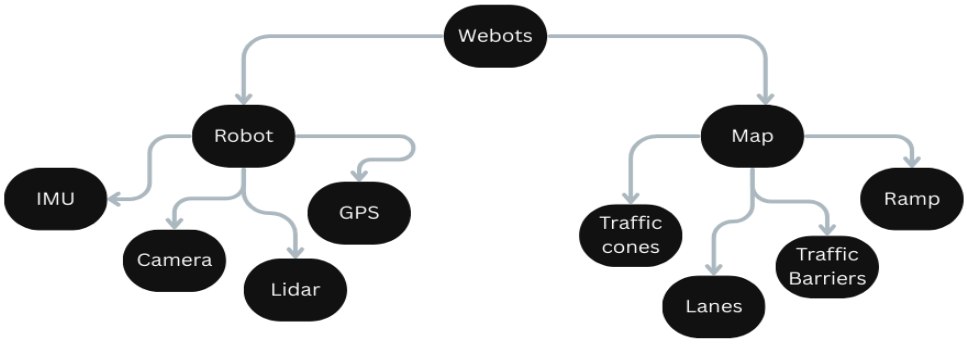


Figure 10. Simulation workflow

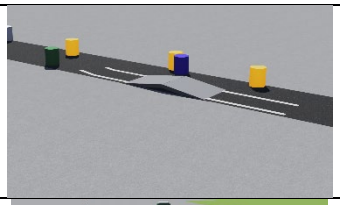
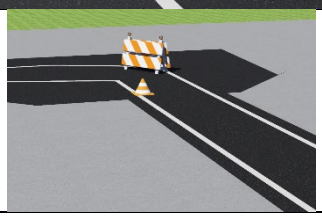
Ramp	
Lane	
Traffic Cones	
Traffic Barrier	

Table 1. Added Parts of Simulation

8. Autonomous Navigation Task

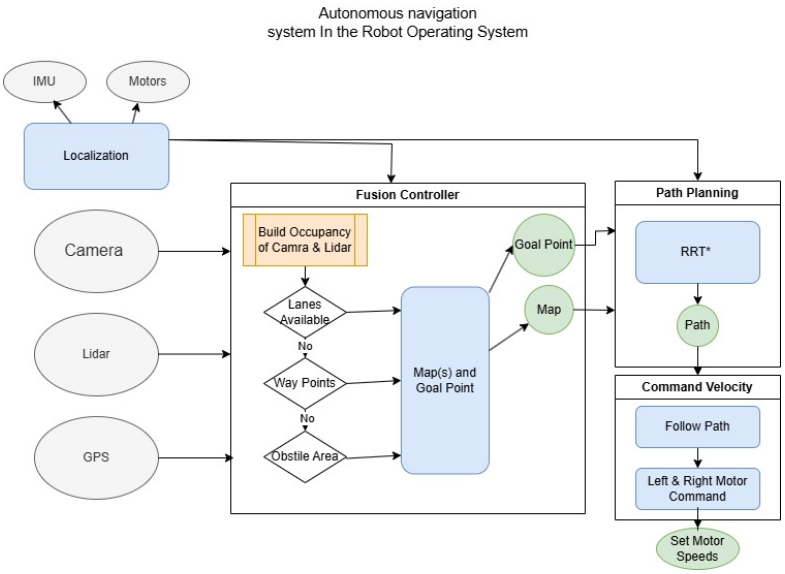


Figure 11. Navigation Flow

The vehicle's autonomous navigation system, based on the Robot Operating System (ROS2), is regulated by the Fusion Controller node, which oversees data integration from various sources, including localization, LiDAR, cameras, GPS, path planning, and control velocity. The localization system, as detailed in Section 6.2, acquires the x and y coordinates, as well as the angle at which the vehicle is heading, through information obtained from the motors and the Inertial Measurement Unit (IMU). Subsequently, the process node determines, based on established thresholds, whether to utilize the camera or LiDAR system. The camera is employed in scenarios where lanes are present, and only a single obstacle is detected; conversely, LiDAR is utilized when an obstacle is identified within a range of four to five meters. Based on the camera input and the established thresholds, the system assesses the availability of lanes and ascertains whether waypoints or obstacles necessitate navigation. These inputs identify the goal points, after which the start and goal points are provided to the RRT* algorithm for the establishment of the optimal path. Ultimately, the speed commands are generated by the control velocity controller, which determines the appropriate speed (ranging from one to five miles per hour) to dispatch to each motor. Each motor operates independently; therefore, the control velocity must account for the differential slip of each tire for every command issued.

9. Cybersecurity Analysis

In autonomous racing, ensuring the security and reliability of robots is crucial. The risk of software disruption by rival teams is a significant concern. To address this, a risk management framework is applied to identify vulnerabilities, model potential threats, and assess their impact. Thus, a comprehensive approach is taken via applying the NIST Risk Management Framework (RMF) and NIST AI Risk Management Framework (AI RMF), threat modeling with STRIDE, a model of threats that used to help reason and find threats to a system, and selecting controls aligned to high-impact risks. Rigorous implementation and penetration testing in the pit area will ensure that even a determined rival team cannot compromise safety or performance.

9.1 Attack Surface & Vulnerability Assessment

- FlySky FS i6S RC (2405.5–2475.0 MHz, according to https://fcc.report/FCC-ID/N4ZFLYSKYI6S/2911062.pdf): RF jamming, replay, spoofing of E-Stop or control channels. Potential to loss of emergency control or malicious robot commands.
- micro-ROS: Open ROS topics over Wi-Fi / 5 GHz backhaul, unauthenticated message injection, MITM.
- ROS Nodes & YOLO11-Nano: Unpatched libraries, insecure ROS parameter server, buffer overflows. Potential to Code execution, sensor-data manipulation.
- Motor Controller Interface (SmartDriveDuo-30): Insecure serial/UART commands, lack of authentication
- GPS: Spoofing or jamming, resulting in incorrect waypoint data.
- LiDAR & Camera (YOLOP): Sensor spoofing (laser reflectors), adversarial images (camouflage cones)
- USB / JTAG on Arduino Portenta/Due: Unlocked bootloader, firmware reflashing
- Onboard PC (ACEMAGIC M2A): OS vulnerabilities, exposed SSH/RDP, missing disk encryption

9.2 Risk Management and Threat Modeling

The NIST Risk Management Framework (RMF) is a set of guidelines that helps organizations manage and reduce risks to their information systems and data. The process involves seven key steps: prepare, categorize, select, implement, assess, authorize, and monitor. These steps provide a structured approach to identifying and mitigating risks, ensuring the security and integrity of information systems and data.

Threat	STRIDE Category	Likelihood	Impact	Risk Level
RF Jamming / Spoofing of RC/E-Stop	Tampering, Denial	Medium	High	High
Unauthorized ROS Command Injection	Spoofing, Elevation	Medium	High	High
GPS Spoofing	Repudiation, Tamper	Low–Med	Medium	Medium
Firmware Reflash via USB/JTAG	Tampering	Low	High	High
Adversarial Camera Input (YOLOP)	Elevation, Tamper	Low	Medium	Medium

Table 2. Threat Modeling

The NIST AI RMF helps AI risk governance across four core functions—Govern, Map, Measure, and Manage—to ensure trustworthy, robust, and secure unmanned or autonomous vehicles.

To mitigate cybersecurity risks of robots, several control measures can be implemented. For RF link protection, encrypted RC and E-stop signals can be used, such as integrating AES-256 radio modules or utilizing FlySky firmware with rolling codes. Additionally, RF-jamming detection can be enabled through a spectrum sensor that triggers a fallback to a manual kill switch. Secure ROS communications can be achieved by enabling SROS2, which encrypts topics and enforces node certificates, as well as implementing namespace segmentation to restrict critical motor

commands to authenticated nodes. Host hardening measures include secure boot and firmware signing, disk encryption, and host firewalls. Physical port controls can be implemented by disabling unused ports and logging any connections. Furthermore, sensor integrity can be ensured through GPS anti-spoofing using multi-constellation GNSS with signal authentication, and camera/LiDAR filtering can be validated with sanity checks to prevent false detections. Furthermore, by layering AI RMF on the top of conventional RMF, which can ensure the computing stack not only resists traditional cyber-attacks but also remains reliable and safe under adversarial AI/ML threats.

9.3 RF Penetration Test

Target: Autonomous Racing Robot (FlySky FS i6S RC link @ 2405.5–2475 MHz & micro-ROS Wi-Fi)

Scope & Objectives: Assess security of the 2.4 GHz RC/E-Stop link and robot's Wi-Fi-based micro-ROS communications. Identify vulnerabilities to jamming, replay, spoofing, and unauthorized command injection. Validate detection and mitigation mechanisms under real-world pit-area conditions. DOI: 10.14722/vehiclesec.2023.23037

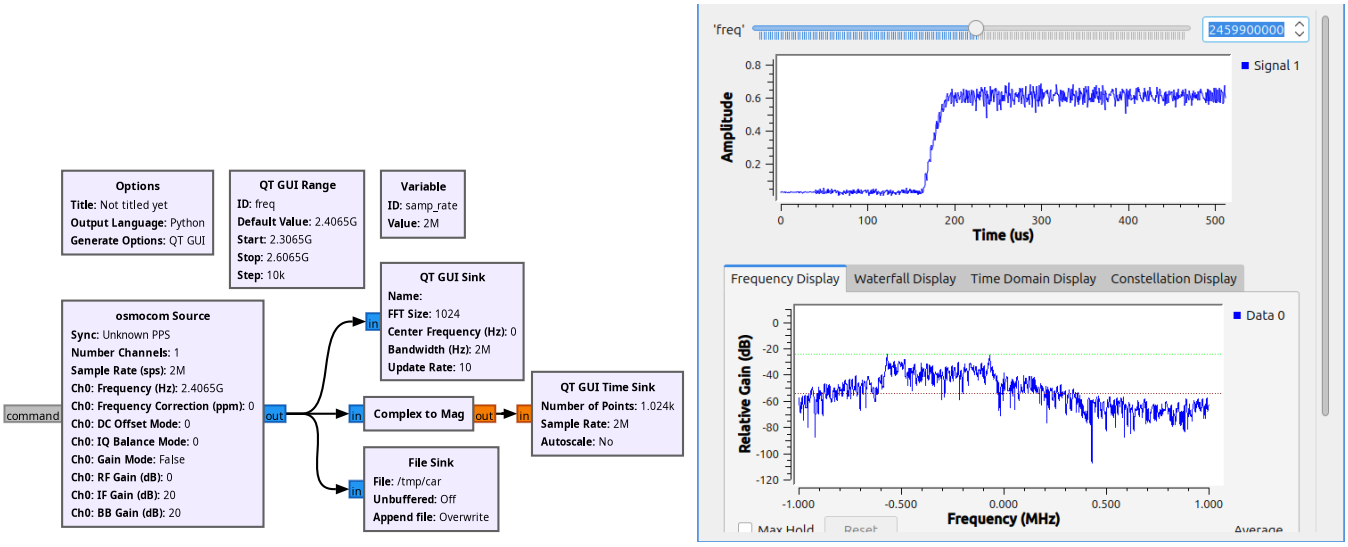


Figure 12. GNU Radio Companion Diagrams used with HackRF One

Environment & Tools:

- HackRF One: Wideband TX/RX SDR for jamming, spectrum analysis, packet capture as shown in Figure 12
- DragonOS: Kali-based preconfigured SDR distro with spectrum-analysis and decoding utilities
- GNU Radio Companion: Custom flowgraphs for real-time modulation/demodulation and signal injection

Expectations & Findings:

- **Reconnaissance**: Passively monitored the robot's RF emissions in the pit area over 30 MHz–10 GHz using HackRF One + GNU Radio. Identified control channel center frequency (~2439 MHz), bandwidth (~2 MHz), and typical packet timing.
- **Sweep Jamming**: Generated continuous-wave and swept-tone jamming signals across 2425–2455 MHz. Measured operational degradation: control latency, packet loss rate, and range reduction.
- **Packet Replay**: Recorded valid RC packets with HackRF One. Replayed at varied power levels and timing offsets to test acceptance by motor controller. Crafted modified command sequences to attempt unauthorized steering/braking signals.

- **Protocol Injection**: Reverse-engineered packet structure in GNU Radio—frame delimiters, parity, rolling-code fields. Attempted live injection of malformed/unauthenticated frames to observe robot behavior and error handling.

The RF penetration test using HackRF One, DragonOS, and GNU Radio revealed critical vulnerabilities to jam and replay attacks on the FS i6S link. By adopting encrypted/rolling-code radios, jamming detection, and FHSS, the robot's wireless safety and control channels can be significantly hardened against adversarial interference.

10. Performance Assessment

TrUBot has undergone a series of performance validation tests to ensure its readiness for the IGVC 2025 competition. These tests focused on evaluating speed, maneuverability, operational endurance, and system reliability under realistic operating conditions.

During mobility trials, **TrUBot successfully exceeded the IGVC 2025 minimum speed requirement of 1 mph**, demonstrating smooth and stable motion across various terrain types. The robot maintained consistent control at low speeds, a critical requirement for precise navigation during lane following and obstacle avoidance tasks. Although the exact maximum speed is yet to be determined, initial results indicate that the platform can operate well beyond the minimum threshold.

Battery endurance was tested with a full payload of **20 pounds**, simulating competition-level mission conditions. TrUBot sustained uninterrupted operation for approximately **40 minutes**, which comfortably satisfies the expected task durations within the IGVC environment. This performance highlights the effectiveness of its power distribution strategy and overall energy efficiency.

Further testing confirmed the robot's ability to maintain stable localization, accurate command following from /cmd_vel, and consistent sensor data publishing (IMU, LiDAR, and wheel encoders), validating the robustness of the integrated ROS 2 and micro-ROS systems.

Overall, TrUBot has demonstrated reliable performance in both autonomous control and teleoperation modes, laying a strong foundation for real-time decision-making and long-duration field deployment during the competition.