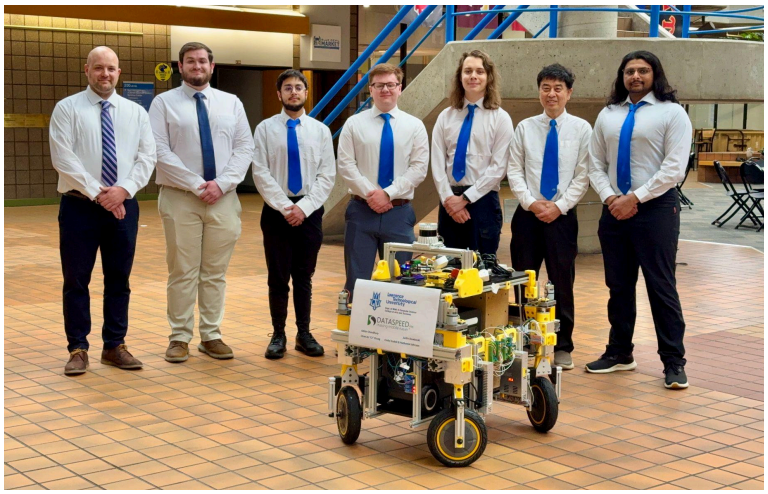
IGVC 2025 Design Report

Lawrence Technological University

Team: rACTor
Robot: Schoolbus

Date: May 15, 2025

Team Captain	Ryan Kaddis	rkaddis@ltu.edu
Members	Devson Butani	dbutani@ltu.edu
	Milan Jostes	mjostes@ltu.edu
	Travis Bowman	tbowman@ltu.edu
	Pranav Malik	pmalik@ltu.edu
Faculty Advisors	Nicholas Paul	npaul@ltu.edu
	Chan-Jin “CJ” Chung	cchung@ltu.edu



Faculty Advisor Statement

I, CJ Chung and Nicholas Paul of the Department of Math and Computer Science at Lawrence Technological University, certify that the design and development on the Schoolbus research platform by the individuals on the design team is significant and is either for-credit or equivalent to what might be awarded credit in a senior design course.

2. Conduct Of Design Process, Team Identification And Team Organization

2.1 Team Introduction and Organization

Team rACTor (pronounced “re·ac·tor”, homage to the LTU’s ACTor platform) consists of undergrad students and graduate students. The team was officially formed on January 14, 2025, and met every Tuesday evening for task updates, group work, and discussion. Each member contributed at least 5 hours every week to designing, fabricating, programming, documenting, and maintaining this year’s competition robot, Schoolbus. Tasks were assigned to specific members of the team according to their proficiencies and prior experience.

Name	Degree Program	Primary Responsibilities	Hours
Ryan Kaddis	M.S. Computer Science	Team Captain	600
Devson Butani	B.S. Mechanical Engineering M.S. Computer Science	Electro-Mechanical Architecture and Rapid Prototyping, Component and Software Pipeline Specification Development	600
Milan Jostes	B.S. Computer Science	Nav2, SLAM, LIO, and Simulation Environment	300
Travis Bowman	B.S. Robotics Engineering, B.S. Computer Science	Teleoperation, E-Stop, Kinematics and Fabrication	200
Pranav Malik	B.S Robotics Engineering	Object Detection and Classification	100

Table 1: Team Members and Responsibilities

2.2 Design Assumptions and Process

As a results-oriented team, we employ an Agile workflow process and actively promote collaboration and quality deliverables over hard deadlines. On our Tuesday meetings, we discuss the prior week’s accomplishments and setbacks as well as plans for the coming week. In addition to weekly meetings, we devote Fridays to working sessions, where members can work together to develop and test new ideas in an environment that supports innovation.

Our design process aims to capitalize each member’s time spent, regardless of the status of the project as a whole. This year will be our team’s first year designing and constructing a robot from the ground up, so we leveraged each member’s time and abilities. Our design process over the months preceding the competition are as follows:

1. **Robot Design:** Identify competition requirements and plan a suitable vehicle: including materials, sensors, actuators, software, and funding.
2. **Assembly:** Obtain and construct necessary parts for robot functionality. Core software functions, like the drive-by-wire system and sensor handlers, are prepared prior to hardware assembly such that the robot is functional upon assembly.
3. **Competition Preparation:** Design solutions required for success at the competition. This includes robot safety, waterproofing, maintenance, and software development like teleoperation, autonomous navigation, and user experience.
4. **Testing:** Functions are thoroughly tested in a setting similar to that at competition. Methodologies and results are recorded and well documented.

3. System Architecture of our Vehicle

Schoolbus is a custom-made autonomous robotic platform jointly sponsored by Dataspeed, and individual donations. The system is equipped with a 3D Lidar, an RTK GNSS, a camera sensor, multiple IMUs, and an all-new omnidirectional drivetrain. The sensors and actuators interface with a Raspberry Pi 5, which communicates with a GPU laptop over ethernet on a local network. The robot receives power from an internal portable power station. Significant hardware improvements have been made over the ACTor platform of previous years, including major upgrades in LiDAR, GNSS, and computer hardware. Figure 2 shows a high level component interface diagram.

Safety is placed at the forefront of everything we do for the vehicle. Failure states are tracked at every layer in Schoolbus, and appropriate responses are taken. A BRB (Big Red Button) and a radio controller enable a user to trigger an emergency stop on a hardware level. Schoolbus actively monitors its internals for faults, at multiple layers, and is capable of triggering a stop in the event of a sensor, actuator, software, or network fault.

Significant software innovations have been made for this year's competition. The Schoolbus software stack uses ROS2 Jazzy in Ubuntu 24.04. In order to make our platform more extensible to various tasks, we use ROS2 Navigation.

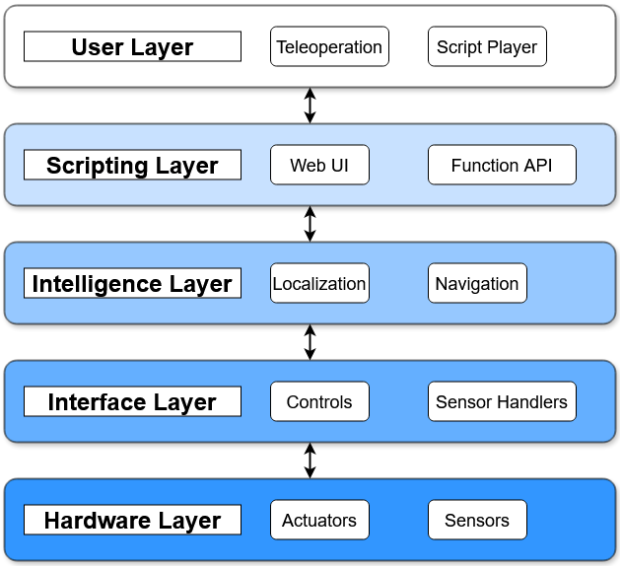


Figure 2: High Level Component Layers

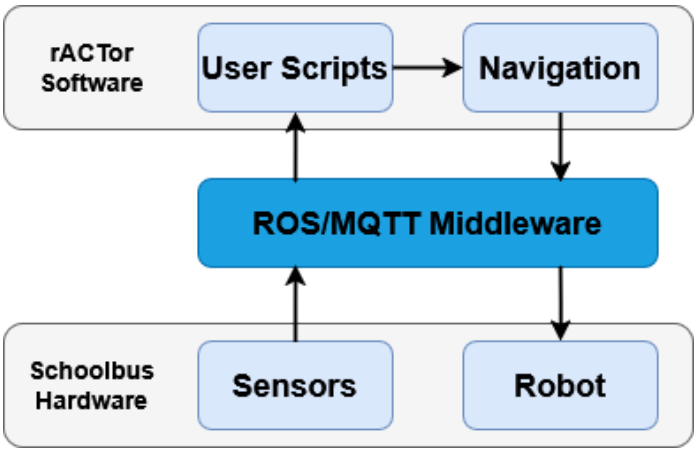


Figure 3: High Level Operating Architecture

ROS2 Navigation enables sensor fusion techniques for Simultaneous Localization and Mapping (SLAM) and route planning in difficult environments. Our omnidirectional controller allows two modes of maneuvering, "Double-Ackermann" and "Fixed Heading". For the specific IGVC tasks, Python scripts are used to easily interface with our sensor and navigation software. These scripts contain simple, high-level code that are designed to complete individual IGVC requirements. A web UI allows for users on the robot's local network to access sensor information, load and execute scripts, and manage robot controls.

4. Effective Innovations In Vehicle Design

There are many differences from other bots and improvements over the ACTor platform that sets Schoolbus apart. Some of the major differences include:

- **All-New Robot Design.** Unlike the Polaris GEM e2 electric car ACTor was built on, Schoolbus is custom-built by the members of team rACTor. Schoolbus is built to be light, highly-manoeuvrable, and agile. An aluminum body balances structural stability and weight. 3D-printed PETG enables parts like mounts, joints, and covers to be manufactured with sub-millimeter precision. The entire bot takes up only 38x32x40in, allowing for movement in tight spaces.
- **Omnidirectional Four-Wheel Steering (4WS) System.** Each of the four wheels on Schoolbus is individually controllable in both linear and angular motion. This allows for unique movement options over conventional robot models, including rotation-in-place, fixed-heading translation, and Ackermann-like control of both the front and rear wheels, which we call "Double-Ackermann" control. In combination with the bot's small form factor, Schoolbus is able to navigate challenging environments and easily accomplish tasks that a traditional robot would struggle to achieve.
- **Middleware Improvements Using ROS2 Jazzy in Ubuntu 24.04.** We have upgraded our ROS implementation to ROS2 Jazzy over our previous ROS Noetic, keeping in line with industry standards. ROS2 is not only more widely supported than ROS1, but is also far more secure. Upgrading our software solutions to ROS2 allows for more programmer-friendly software development and easier code maintenance for future projects and competitions. Ubuntu 24.04 is a significant upgrade over the previously-used Ubuntu 20.04 in terms of support, security, and user experience.
- **Professional Navigation Stack with ROS2 Navigation.** ROS2 Navigation (Nav2) is a professional platform for robot localization and navigation, enabling robots of various configurations to maneuver complex environments. Nav2 is not only well-suited for the challenges presented by IGVC, but is also capable of handling additional future projects. By implementing Nav2 in our control stack, we can ensure that Schoolbus is a capable robotic platform, whether at competition or conducting research.
- **State-of-the-Art Object Detection in YOLOv12.** Our object detection methods now use the new YOLOv12 weights, released in February 2025. The new weights provide a significant advantage over our previous YOLOv8 model in inference speed, resource consumption, and accuracy. Our models will use weights from custom dataset training (such as tires and potholes) alongside the Common Object dataset already prebuilt into the software.
- **Custom Wheel Control Stack.** Our custom-made wheel layout and control structure enable highly precise and responsive control of the robot, both during teleoperation and autonomous control. All software responsible for managing wheel actuators is written and optimized by team rACTor, allowing us to personalize our control scheme to our specific requirements.

5. Description Of Mechanical Design

The mechanical design of the rACTor is guided by a hardware philosophy focused on rapid prototyping, symmetry, and simplicity. Our primary motivation for implementing four-wheel steering (4WS) was to explore a new approach for the team and to unify the steering control strategies required for both AutoNav and Self-Drive courses. While Self-Drive typically relies on Ackermann steering, this configuration would make navigating around barrels on the AutoNav course difficult. Alternatively, a differential drive system would depend too heavily on wheel slip during lane following in Self-Drive. By adopting 4WS, we achieve a balanced solution: it maintains the benefits of Ackermann steering while Double-Ackermann (steering on both front and rear axles) delivers full maneuverability for complex navigation tasks.

Using standardized, readily available commercial off-the-shelf (COTS) components significantly reduced costs and development time. This approach allowed for rapid design iterations and early identification of component clearance and structural design. SolidWorks (CAD) enabled designing a constrained frame geometry and 3D printed mounting for most components. The design process was kept simple and iterative; starting from the ground up, adding components only for functionality and ensuring they were well constrained.

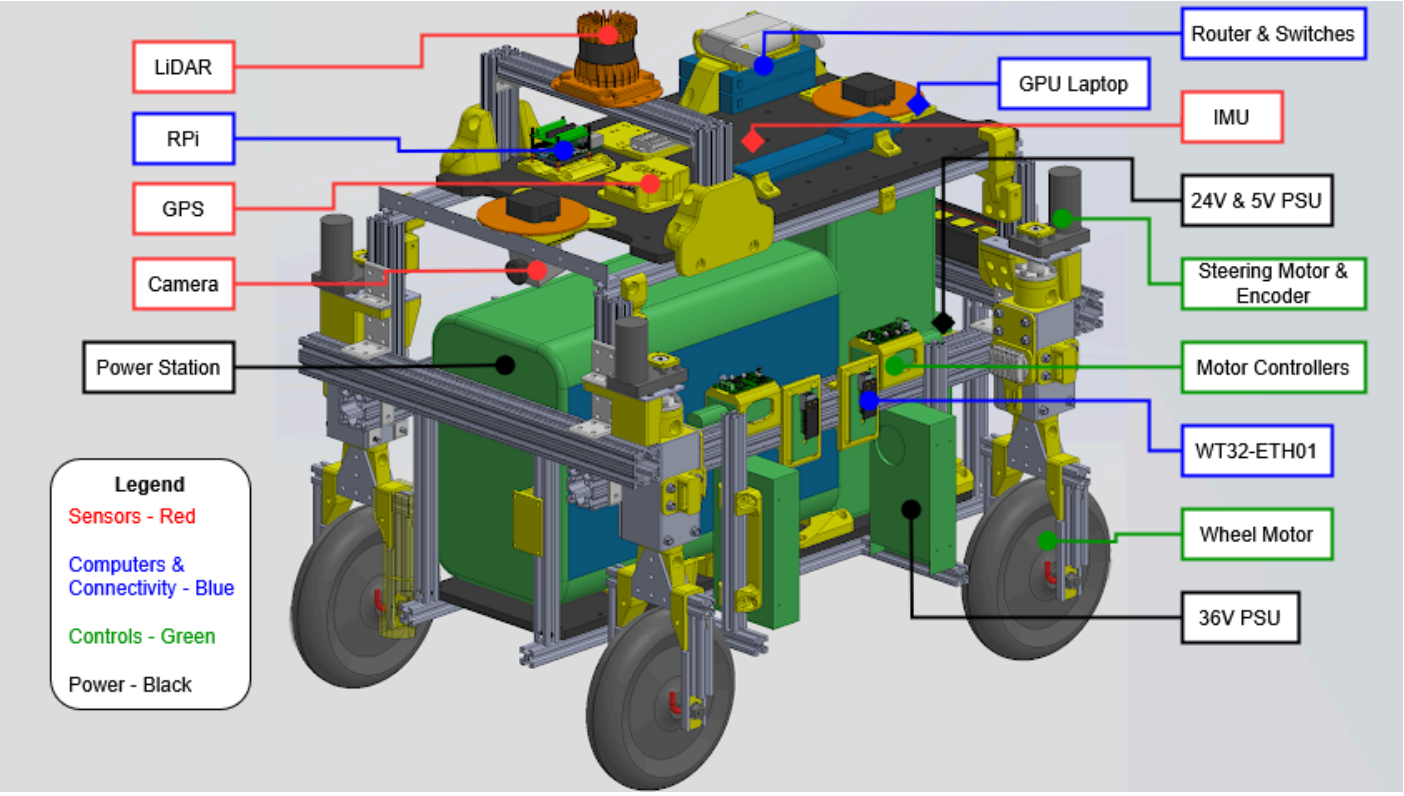


Figure 4: Schoolbus Mechanical Design and Component Layout

Our main focus was the unique steering system and the power station as core components driving the frame construction. The overall structure follows the "exact constraint design" principle, ensuring the robot is neither over-constrained nor under-constrained. The frame design prioritizes structural integrity and symmetrical design to simplify fabrication and balance weight distribution. Utilizing standardized fasteners (primarily ¼-20 and M6), ensures simple assembly and quick maintenance timelines.

Aluminum Extrusion Framework: Cut-to-order and repurposed aluminum extrusions form the backbone of the robot's structure. Different profile sizes were selected based on their torsional rigidity, load capacity, availability, and ease of assembly. This modular approach allowed for flexible mounting options and simplified the integration of additional components as the design evolved. The T-slotted aluminum extrusion system provided unprecedented tensile strength, vibration resistance, and load-bearing stability while enabling easy maintenance and part replacement.

Material Combinations: The design incorporates a strategic mix of materials to optimize performance while managing costs and lead times:

- Aluminum and steel for primary load-bearing structures
- Wood for static, non-critical structural elements
- 3D printed PETG for complex geometries and connector parts, leveraging PETG's high tensile strength and resilience

Drive System: The rACTor utilizes a Segway Ninebot Max G30P Electric Scooter Hub Motor (500W BLDC). This integrated wheel-motor combination provides powerful and smooth movement at lower speeds while maintaining quiet operation on paved surfaces. The integrated tire design simplifies the overall mechanical assembly while ensuring reliable traction.

Steering Mechanism: The symmetrical steering assembly features an aluminum square tube with 3D printed inserts housing steel ball bearings. A smaller square aluminum extrusion acts as the shaft which slides into the bearings using a 3D-printed collar. This design creates a robust yet smooth steering action, with loads transferred from the shaft to the frame via steel bolts. The square profile prevents unwanted rotation while allowing necessary vertical movement.

Suspension: Though the steering column design supports the addition of a suspension system, we opted not to implement one given the competition scope and budget constraints. Instead, Noise, Vibration, and Harshness (NVH) management for electronics is accomplished through soft mounts for vibration-sensitive components,

thread lockers and washers for fasteners to prevent loosening, and strategic component placement to minimize the effects of vibration. This approach balances performance requirements with design simplicity while maintaining the option to add suspension in future iterations if necessary.

Weatherproofing: The outdoor-rated hub motor features built-in weatherproofing, while non-waterproof electronics are positioned higher in the frame to avoid water exposure. The frame design incorporates natural drainage paths and protective covers for sensitive components. PETG's excellent chemical resistance and UV light protection capabilities make it ideal for outdoor-exposed 3D-printed components. The external housing geometry is designed to direct water away from sensitive components and connection points. All external sensors feature waterproof housings or are installed with appropriate protective covers to ensure reliable operation in various weather conditions.

6. Description Of Electrical And Power Design

The electrical and power design of the rACTor prioritizes reliability, flexibility, and safety. The system architecture features a centralized power source with distributed power management for various subsystems. The electronics design also incorporates COTS components where appropriate, complemented by custom mounting solutions to meet specific performance requirements.

Power Source: The rACTor's core is an Anker SOLIX F2000 Portable Power Station featuring a 2kWh LiFePO4 (Lithium Iron Phosphate) battery that supports all types of power needs at 2kW max draw; pure sine wave 120VAC via 4×NEMA 5-20 and 1×NEMA TT-30, 12VDC via a 2xSAE J563 auxiliary outlets, and 2xUSB-A and 3xUSB-C PD ports.

This power station was selected for its high energy density, extended cycle life, and enhanced safety characteristics compared to traditional lithium-ion batteries. The LiFePO4 chemistry provides excellent thermal stability, reducing fire risks during operation and charging. All the main electronic components work on AC power allowing them to be run outside the robot while the power station fast charges 0 to 80% in just 1.4 hours. Moreover, the capability of directly adding 1kW of Solar Panels opens up the possibility of setting up a charging dock that enables 100% renewable energy powered deployment. Circuit protection is handled by the power station as well as connected surge protectors for additional safety.

Distribution: The power station is connected to individual components via AC-DC, DC-DC converters and direct AC plugins depending on the component's power requirements. In terms of power, Schoolbus can be physically separated into two parts; the driver suite (responsible for autonomous driving) senses and thinks while the robot base (solely responsible for motion and power storage) acts. This unique feature allows independent maintenance and development of each part. For example, by simply removing the top driver suite and plugging it directly into a wall outlet, it allows members to work on software development and testing without requiring the robot base to be connected.

Robot Base: A dedicated 350W BLDC motor controller (commonly found as ebike controllers) drives each hub motor and a separate 250W brushed DC controller operates the steering motor for each corner. These components are selected due to budget constraints and can be easily upgraded later on. For each corner, both of these motor controllers are driven via an ESP32 based board (WT32-ETH01) connected over Ethernet. Keeping all the analog signals to very short cable lengths allows individual corners to operate independently. This strategy was chosen to kickstart the firmware development concurrently with frame design and other component selection.

Driver Suite: A standalone set of sensors are directly connected to the Raspberry Pi 5 and a laptop with discrete GPU via USB and ethernet switch. The following sensors are installed:

- **Ouster OS1 LiDAR**
  - Provides full-surround 3D point cloud data with high resolution and range.
- **RouteCAM_CU22_IP67 IP Camera**
  - PoE camera sensor that captures high resolution images with low latency.
- **Yahboom CMP10A Inertial Measurement Unit (IMU)**
  - Tracks robot orientation and motion using gyroscopes (yaw), accelerometers (roll and pitch), and a magnetometer.
- **ArduSimple simpleRTK2B GNSS**
  - 2x Ublox ZED-F9P multiband receiver supplies a GPS fix with RTK accuracy.

ESTOP: Wireless estop using RC controller and receiver (Turnigy i6S TX and iA6B RX) is connected in series with mechanical estop buttons which drive active-on relays directly connecting the motor controller brake line. Breaking the loop anywhere applies instant braking force via the BLDC motor controllers.

7. Description of Software System

Schoolbus is built on the principle of modular programming, where software solutions are split into smaller sections that function independently. Programming in modules allows multiple developers to work seamlessly and in parallel. Modules can also be tested individually, enabling developers to complete and validate sections of code with no deadlocks.

ROS2 supports the modular approach to software development by presenting a messaging scheme that nodes can use to broadcast and retrieve information. Rather than having nodes communicate with each other directly, they instead share and receive information over “topics”. This allows nodes to use data available on the ROS network without having to rely on another node. ROS2 also allows for real-time messaging between nodes running on different devices.

The entire software stack is split across three different devices, each responsible for a different layer of robot controls and software intelligence. On the lowest layer, WT32-ETH01 microcontrollers control the actuators for each wheel on the Schoolbus. On the sensor layer, a Raspberry Pi 5 receives data from the many sensors on board and publishes them to the ROS network. The Raspberry Pi also manages robot movement schemes and teleoperation. On the intelligence layer, a GPU laptop does advanced tasks like localization, navigation, object detection, and user interaction.

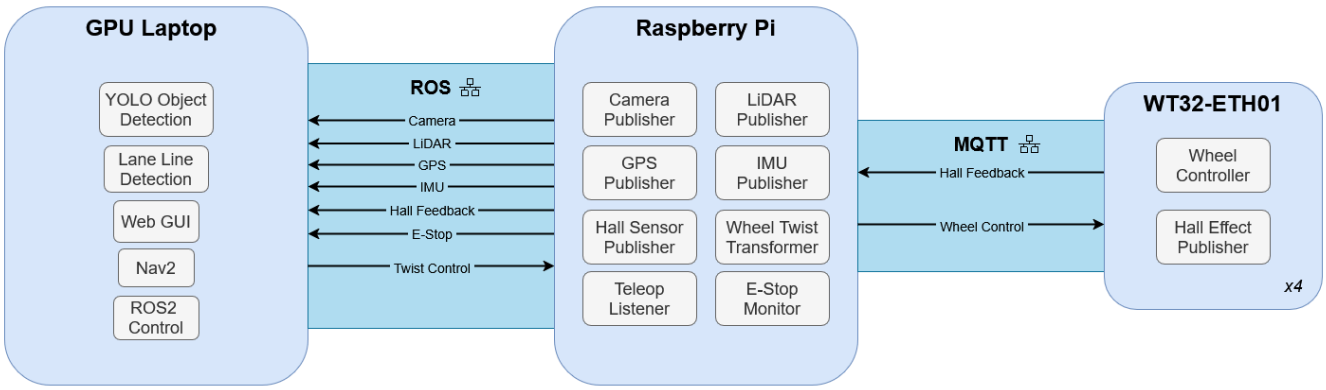


Figure 5. Software and Communication Architecture.

7.1 Perception and Localization

The Schoolbus platform is equipped with an impressive sensor suite to support perception and localization tasks. The suite includes a 360-degree LiDAR for high-resolution spatial mapping, a 2D camera for visual perception and object detection, an inertial measurement unit (IMU) for estimating orientation and acceleration, and a GPS module for global positioning. Together, these sensors enable the system to perceive its surroundings and accurately localize itself within its environment.

7.2 Mapping and Navigation with Nav2

Nav2 is a successor of the ROS Navigation Stack. It was developed to allow robots and autonomous vehicles to avoid obstacles and move through complex environments. Modern warehouse robots utilize Nav2 in order to navigate around prefabricated floor plans, using this predetermined map, the bot is able to determine its location. These robots use a predefined global map, which best approximates permanent walls in the environment, and a local map, which dynamically uses the robot's sensors to create a map. It then utilizes the maps and odometry data to plot routes from its current location to a specified goal, as well as detect and avoid obstacles that may appear in its path. Using LiDAR (Light Detection and Ranging) as well as SLAM (Simultaneous Localization and Mapping), the robot can identify its current location by comparing the identified walls of its local map with that of the global map. By sending sensor data, as well as the maps to the controller server, it can create a costmap that can be sent to a planner server. This planner server then can instruct the robot on an optimal path to its desired end point.

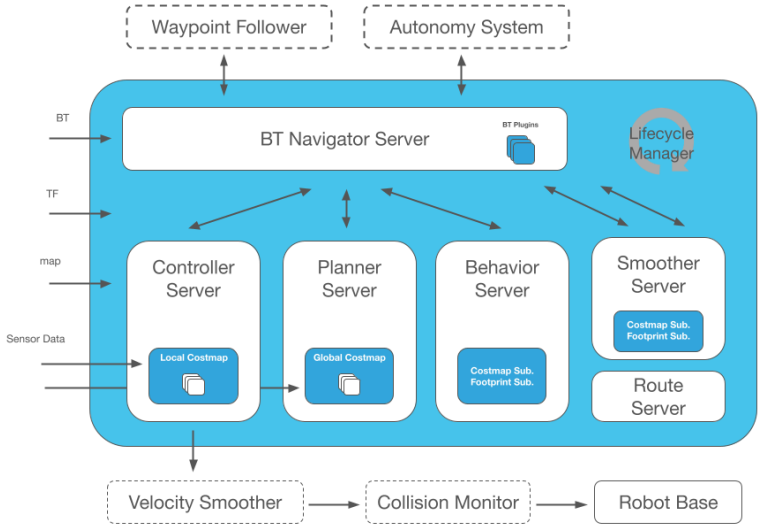


Figure 6: Nav2 Process Diagram

The robot used in this project will be placed in an unfamiliar environment, so the use of a predefined global map is impossible. Therefore the robot must estimate its current position and dynamically create a global map through the updates to its local map. Because the robot does not make use of a predefined map, it must rely entirely on its odometry and sensor data. IMU data and GPS coordinates lay the foundation for basic odometry, however, drift in the data aggregates into larger problems. In order to combat this problem, the use of LIO (Lidar Inertial Odometry) is used to supplement the data. This system utilizes the point cloud from the LiDAR to approximate its movement by computing the difference between its previous distance to known walls. This allows the robot to maintain a much more stable and accurate odometry, eliminating the drift. This odometry data is then used alongside the costmap to plan a route and send instructions to the robot and effectively navigate around obstacles.

Alongside obstacles, lane lines are also plotted in the costmap. Since lane lines cannot be detected with LiDAR, we use the camera to detect lane lines, and use the position of the camera to translate detected lines to the costmap. Doing this, the route planner is able to plan a path that does not cross over a line. For lane changes and

other safety-related situations, the lane lines can be temporarily ignored until it is safe to continue following the lane.

7.3 Teleoperation

Teleoperation is the process of remotely controlling a robotic system from a distance, typically with a human operator. In this system, the operator sends movement commands over a wireless communication protocol, such as the iBUS protocol. The control interface consists of a Turnigy Flysky transmitter, which provides eight primary radio channels for motion control and an additional dedicated channel for remote emergency stop (e-stop) - a critical safety requirement.

Although the robotic system is primarily autonomous, teleoperation is essential during transportation to and from competition courses. Additionally, it plays a crucial role in evaluation and testing, where manual control enables fine-tuning and troubleshooting. These operational requirements drive the need for a robust and reliable teleoperation system.

In teleoperation, the user interfaces with the Turnigy FlySky transmitter, which provides four directional control channels and four selectable switch channels. The transmitter communicates using iBUS, a digital, low-latency, bidirectional protocol that efficiently transmits multiple control channels over a single wire at a fixed baud rate of 115200, reducing system complexity. The system's teleoperation receiver, a TGY-iA6B, is integrated into the Raspberry Pi 5 hardware stack, where it receives the iBUS signal and relays the data to the robot's ROS2 software architecture. At this stage, the E-Stop signal is directly routed to an emergency stop relay, which immediately cuts power to the robot for safety without the need of the microcontroller.

7.4 Drive-by-Wire Control

Both Nav2 and teleoperation produce a general scheme for how the robot should move as their output. This control scheme needs to be translated into individual instructions for the four wheels. This is accomplished through the Wheel Twist Transformer, which converts general movement messages into wheel instructions using trigonometry. As previously mentioned, Schoolbus has two primary movement modes, labeled as "Double Ackermann" mode and "Fixed Heading" mode.

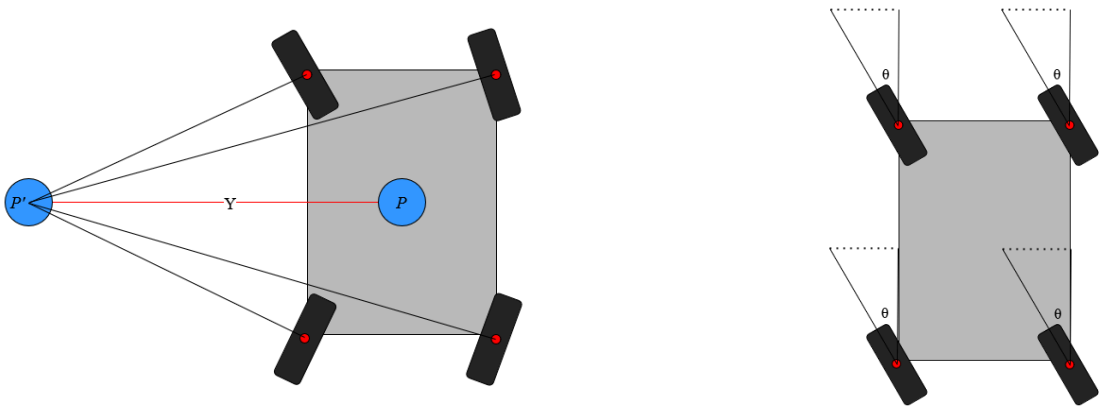


Figure 7. Double Ackermann (Left) and Fixed Heading (Right) Steering Diagrams.

Double Ackermann

The “Double Ackermann” control mode follows the traditional Ackermann method of steering; however, all four wheels participate in steering. In Double Ackermann mode, the road angle for each wheel is calculated with the following algorithm:

1. Assume a Point P in the center of the robot.
2. Obtain a yaw value Y from a general control message.
3. Generate a Point P’ that is Y units away from Point P. Point P’ will be the center of the robot’s angular path, with radius Y.
4. Rotate each wheel such that the road angle of the wheel is normal to a vector to Point P’.

Fixed Heading

The “Fixed Heading” control mode allows the robot to translate over the road surface without changing the robot’s heading. In Fixed Heading mode, the road angle for each wheel is calculated with the following algorithm:

1. Obtain a road angle value θ from a general control message.
2. Rotate each wheel such that the road angle of the wheel is θ.

Wheel Operations

Each wheel is equipped with a WT32-ETH01 microcontroller, which is responsible for handling basic operations local to the wheel. The wheel microcontroller is responsible for three major functions: speed control, Hall Effect interpretation, and steering control. The microcontrollers use Hall Effect sensors inside the wheel to maintain accurate speed control while managing the throttle and brake channels on the motor controller. The microcontroller also reads from an encoder on the steering motor to maintain the proper wheel angle.

Communication with the wheels is done through MQTT, a publisher/subscriber messaging framework, like ROS; but more lightweight. Even though it is lacking many of the features offered by ROS, MQTT is still valuable to low-power devices, like Arduino, ESP32, and other microcontrollers like the boards used on the wheels.

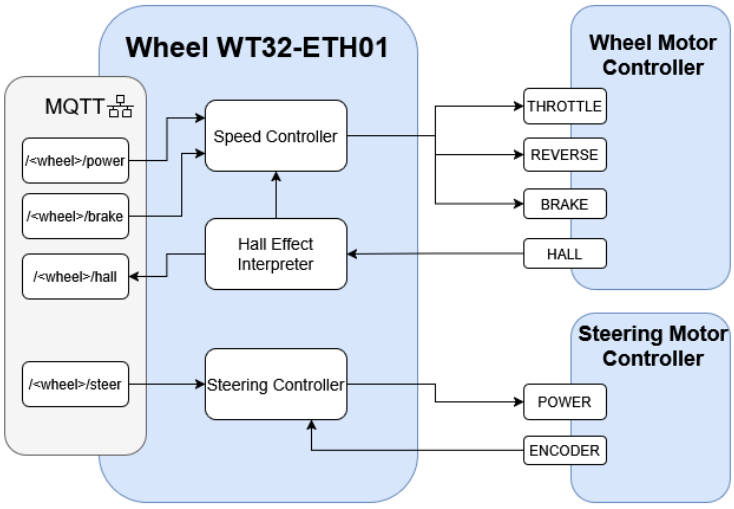


Figure 8: WT32-ETH01 Process and IO Map

7.5 Object Detection and Classification

7.5.1 YOLO Models

Ultraltytics’s YOLO v12 is the latest model as of February 2025 and trades off higher accuracy in all tasks for slower training times compared to previous models. We decided to utilize the v12’s medium model for

object detection, having a higher mAP (Mean Accuracy Precision, a measure of the number of correct/accurate predictions) than v11 while only taking 3% more inference time during training.

7.5.2 Training & Classification

There are 4 classes (3 obstacles and 1 sign) that will trigger an event and change the movement of the bot.

- **Obstacles** - If any of these are seen, move around the obstacle(s) if space is available
  - Potholes
  - Tires
  - Pedestrians
- **Signs**
  - Stop signs - If this is seen, the robot must come to a stop at the white line

We created a dataset from photos taken directly from the camera mounted on Schoolbus and trained the medium version of YOLO v12 on it. The following images are examples from testing on the tire obstacle dataset.

Figure 9: Images Batch of v12 Model Successfully Identifying When a Tire is Present

7.6 Scripted Route System

Our simplified route system enables us to use the Schoolbus platform with a Python API for the robot's functions and simple web interface to allow viewing ROS topics in real-time. It can also select and run Python scripts designed for specific routes. The API simplifies the designing routes to the level of verbal directions given to a driver in an unknown area. For example, a route can start with following the lane up to a GPS waypoint then turn right at the stop sign to continue following the lane until the next waypoint. Nav2 handles obstacle avoidance checks which run in parallel where the vehicle either stops if not enough room is available or changes the lane if another path is found.

Using Python as the base language, our route scripts are compiled right before execution allowing for easy debugging or alterations to routes on the fly. This is one of the biggest time-saving features of our software stack. Cutting down on debugging time while testing in the field has had immediate benefits for the development timeline.

8. Cyber Security Analysis

The NIST Risk Management Framework (RMF) is a 7-step methodology for assessing and managing information security in data-intensive systems. These steps include preparing the organization, categorizing data by impact, selecting and implementing security controls, assessing effectiveness, authorizing operation, and continuously monitoring for risk. These steps are required for FISMA (Federal Information Security Modernization Act)

compliance and form the basis of the cybersecurity process we adopted for securing our autonomous vehicle system.

As our platform transitioned to ROS2, which offers enhanced security features, we now benefit from support for secure DDS (Data Distribution Service) communication, which includes built-in options for authentication, encryption, and access control between nodes. This shift is critical for protecting inter-process communication in distributed robotic systems.

We also utilize MQTT for lightweight message transport across the drive-train subsystem. To secure MQTT communication, we implement the following measures:

- Client authentication using unique credentials.
- Broker-side IP whitelisting and blacklisting to control which devices can connect.
- Rate limiting and denial-of-service (DoS) protection to mitigate potential flooding attacks or misuse.

To further protect the system:

- WPA2 security is enabled on the local network, and MAC address whitelisting restricts device access.
- Wired connections are used for all critical control systems to mitigate the effects of 2.4 GHz and 5 GHz jamming.
- Driver override is enforced by design: any manual input disengages autonomous control.
- Emergency stop (E-stop) logic defaults to *engaged* on communication loss or process failure.

These layers of defense work together to mitigate potential cyber-physical attacks and ensure the safe operation of the vehicle. Table 3 outlines the most likely threats and our corresponding risk mitigations.

Attack	**Threat Level**	**Risk**	**Defense**	**Response**
Local Network Breach	Medium	ROS Access, Web GUI Access	WPA2 Security, MAC Address Whitelisting	Activate hardware E-Stop, MAC Address Blacklisting
Source Code Tampering/Removal	Low	Unexpected Vehicle Behavior	Secured Device, Controlled Access	Recover code from version controlled online repository
Remote Connection to Main Computer	Medium	Process Tampering, File Tampering	Restricted SSH Access, Password Protection	Recover code, SSH Blacklisting
Remote Connection to E-Stop Monitor	Low	E-Stop Process Tampering	Restricted SSH Access, Password Protection	E-Stop turns on when the monitor is disabled or loses power
Wireless Signal Jamming	Low	Wireless Network Malfunction/Disable	Wired Connection for Critical Devices	Remote E-Stop will activate, Use exclusively wired connections

Table 3: Cyber Security Threats and their Responses

9. Analysis of Complete Vehicle

9.1 Construction and Integration

Initial testing confirms that the frame design is both robust and sturdy, capable of withstanding rough driving conditions. However, these tests also highlighted the need for a small suspension system when operating off-road or on poorly maintained pavement. The symmetrical chassis design proved advantageous, reducing the number of replacement parts required and simplifying both manufacturing and maintenance. Constructing the robot as two main modules-the motion base and the driver suite-enabled parallel integration timelines and minimized delays due to component sourcing. This approach allowed some team members to concentrate on

developing the 4WS robot base and its firmware, while others focused on validating individual components of the driver suite.

A recurring challenge for the team has been the limited number of members with expertise in electrical and mechanical design. To address this, we leveraged the power station's AC power, and ethernet switches, which significantly streamlined the integration of software and hardware. As a result, assembling the electronics for the driver suite required only reference to product manuals, eliminating the need for specialized electrical or hardware knowledge and enabling full system integration with minimal difficulty.

9.2 Robot Safety and Reliability

In order to provide adequate safety for the team members and other pedestrians, Schoolbus has a number of safety features implemented. Whenever the robot is active, a safety light flashes, indicating that it is on and possibly mobile. Schoolbus has a set firmware speed limit of 5 MPH, ensuring that the robot will not cause significant damage in the event of a collision. Steering limits stop the robot from pulling out cables or wires by ensuring that it only ever turns within its given range of freedom. Finally, the robot utilizes a physical E-Stop line that prevents the robot from moving as soon as the E-Stop button is pressed, the wireless E-Stop is activated, or connection is lost with critical devices. All of these features are meant to keep the people, and the robot, safe from harm.

Even though Schoolbus hardware is built to be robust, we have come across hardware failures that are addressed using the table below.

Failure Points	Resolution
Camera is not found	Unplug and plug the ethernet cable back in, restart the router and ping the assigned IP
Laptop crashes	Hold the power button for 10s to shut it off. Then make sure the charger is connected and powered on. Turn it back on and re-launch the software
Ethernet connection	Make sure the cable is pushed in. Listen for an audible click as the cable latches in. Ping the IP of the device in question for further network troubleshooting
ESTOP Braking Failure	Restart the motor controllers and the estop loop. Sequentially verify every estop activation point in the loop (BRB, Wireless, relays, brake wire)
Raspberry Pi Malfunction	Restart. If SSH does not work, remove the pi from the vehicle and diagnose it outside. Worst case, re-flash using the backup image and validate it

Table 4: Hardware Failures and their Respective Resolutions

9.4 Robot Control

As detailed in Section 6.4, Schoolbus has two primary control modes, Double-Ackermann and Fixed Heading. These modes are available and used in both autonomous and teleoperation modes. Nearly all of our autonomous strategy uses Double-Ackermann, as Ackermann is a very common control scheme, and is natively supported by Nav2. Fixed Heading mode is used in select circumstances where it would provide a significant advantage, such as in parallel parking.

9.5 Software Development and Version Control

All software development done for rACTor is publicly available on our Github organization. We use the proven principles of version control present in Git/Github to properly maintain software and enable collaborative programming.

We also make disk backups of important devices like our Raspberry Pi, so that they can be restored in the event of a major failure. Additionally, we make install scripts that set up developer devices for rACTor, so that new devices and members can be easily onboarded.

9.6 Simulation Environment and Testing

The simulation environment used for the development of the robot is based on the Gazebo Simulator, an open source simulator that allows developers to create and test robots in virtual environments. Used alongside ROS2 Jazzy, Gazebo provides a realistic simulation of a robot's behavior, as well as simulated data from sensors. These simulations allow the testing of software such as Nav2 and SLAM with realistic data, in various environments.

Gazebo has the ability to utilize various sensors through the use of a URDF file. The URDF file is used to define the robot's physical model, including the placement of the sensors. This makes the data received more closely imitate that of a real robot, allowing for more effective simulation, and the fine tuning of algorithms. The primary goal of testing was to ensure that Nav2 was capable of navigating complex environments, as well as to optimize the sensor placement on the actual robot.

9.7 Physical Testing and Performance

The table below details the physical tests conducted on Schoolbus. This table is up-to-date as of May 15, 2025.

Physical Test		**Status**
Controls	Linear Movement	Functional, Tested
	Steering	Functional, Tested
	Brakes	Installed for Front Wheels, Testing
	Double-Ackermann	Functional, Tested
	Fixed Heading	Functional, Tested
	Remote Control	Functional, Tested, <10ms response time
Sensors	RouteCAM Camera	Functional, Tested, Available in ROS Network
	Ouster OS1 LiDAR	Functional, Tested, Available in ROS Network
	simpleRTK2B GNSS	Functional, Tested, Available in ROS Network
	Yahboom CMP10A IMU	Functional, Tested, Available in ROS Network
Safety	E-STOP BRBs	Functional, Tested
	Remote E-STOP Switch	Functional, Tested
	Speed Limit	Functional, Firmware Limited to 2 m/s (4.47 mph)

Table 5: Physical Tests and their Statuses

10. Unique Software, Sensors, and Controls

The Schoolbus platform and most rACTor software is designed to be as modular and generalized as possible. As such, most hardware and software is not specifically designed for IGVC. Competition specific content includes:

10.1 AutoNav

- **Nav2.** The navigation stack, including LiDAR Inertial Odometry, line detection, and route planning.
- **Waypoint Navigation.** GPS based navigation and route planning to a target location.

10.2 Self-Drive

- **YOLO Object Detection.** The tire, pedestrian, and stop sign detection models, and software for detecting these objects within an image.
- **Competition Scripts.** The high-level scripted routes designed to complete specific IGVC-related tasks.

11. Initial Performance Assessments

How is Schoolbus performing to date?

Using our course at LTU, we can test functions required for the competition ahead of time. The tables below detail the status of each test.

Self-Drive Tasks	
Qualification Tests **Complete**- Q.1: E-Stop Manual **Complete**- Q.2: E-Stop Wireless **Testing**- Q.3: Lane Keeping (Go Straight) **Complete**- Q.4: Left Turn **Complete**- Q.2: Q.5: Right Turn *Machine Vision Tests* **Complete**- FI.1: White Line Detection **Complete**- FI.2: Static Pedestrian Detection (Vision) **Complete**- FI.3: Tire Detection *Traffic Sign Tests* **Incomplete**- FII.1: Stop Sign Detection *Intersection Tests* **Incomplete**- FIII.1: Lane Keeping **Incomplete**- FIII.2: Left Turn **Incomplete**- FIII.3: Right Turn	*Parking Tests* **Complete**- FIV.1: Parking. Pull Out **Complete**- FIV.2: Parking. Pull In **Complete**- FIV.3: Parking. Parallel *VRU Tests* **Incomplete**- FV.1: Unobstructed STATIC pedestrian detection **Incomplete**- FV.2: Obstructed DYNAMIC pedestrian detection **Incomplete**- FV.3: STATIC pedestrian detection. Lane changing **Incomplete**- FV.4: Obstacle detection. Lane change *Curve Road Tests* **Incomplete**- FVI.1: Lane Keeping **Incomplete**- FVI.2: Lane Changing *Other Tests* **Testing**- FVII.1: Pothole Detection **Incomplete**- FVII.2: Merging *Main Course* **Incomplete**- Simple Main

AutoNav Tasks	
Qualification Tests **Complete**- Robot Dimensions **Complete**- E-Stop Manual **Complete**- E-Stop Wireless **Testing**- Safety Light **Complete**- Speed Control **Testing**- Lane Following **Incomplete**- Obstacle Avoidance **Incomplete**- Waypoint Navigation	*Course Tests* **Testing**- Autonomous Control **Complete**- Ramps/Grades **Incomplete**- Obstacle Routing **Complete**- GPS Localization

Table 6: Status of IGVC Functions