UNIVERSITY OF THE DISTRICT OF COLUMBIA
SCHOOL OF ENGINEERING AND APPLIED SCIENCES

Department of Electrical and Computer Engineering

Autonomous Ground Vehicle for IGVC AutoNav

FIREBIRD

Team Captain/Members:

***Marvin Cheguen Mendez**, 240-505-9926, marbymendez81@gmail.com*

Chukwuma Duru, 267-336-4047, chumduru@gmail.com

Faculty Advisor:

***Dr. Ososanya**, 202-274-5837, eososanya@udc.edu*

Target Challenge: *AutoNav Challenge*

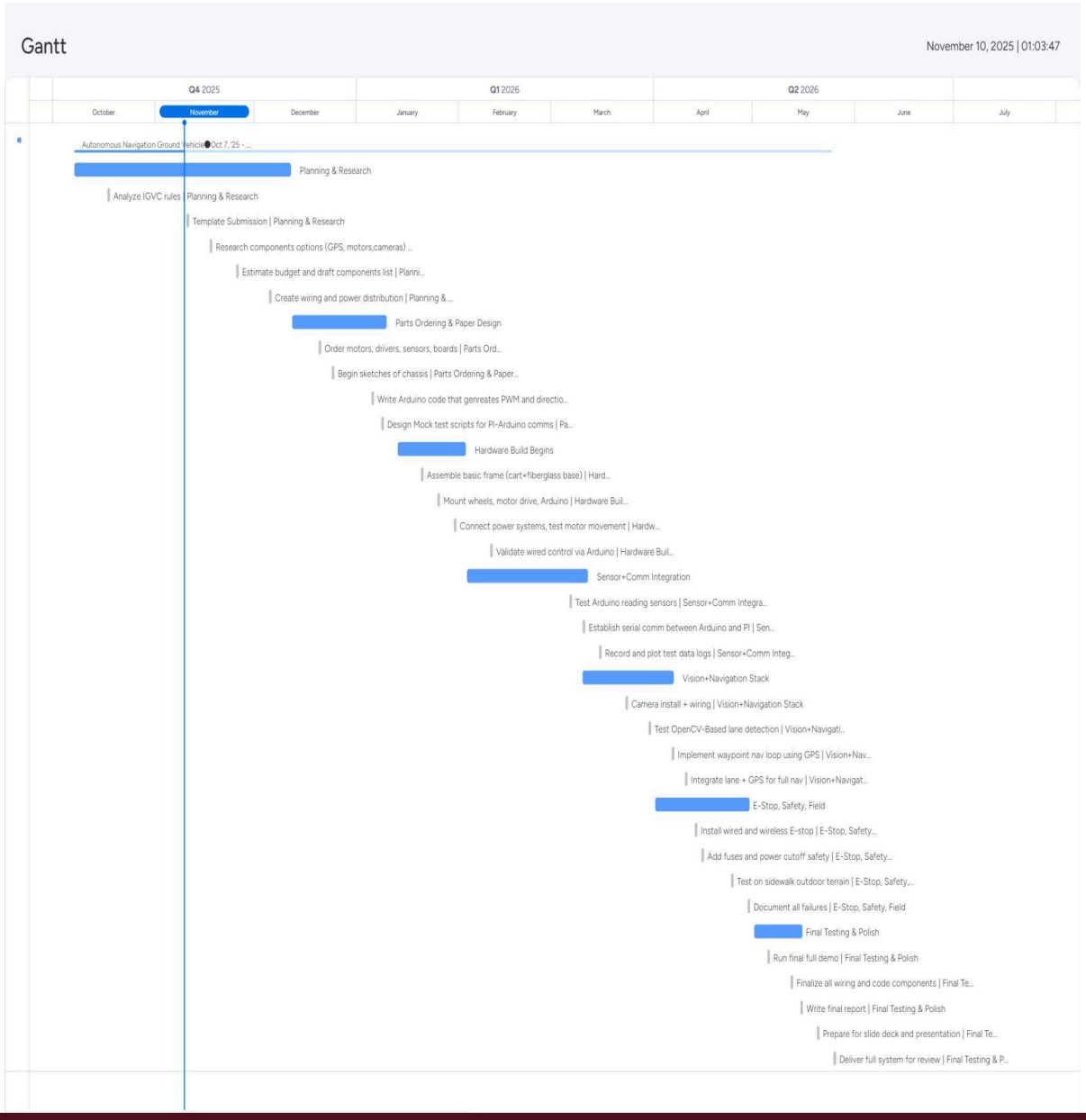
1. System and Subsystem Requirements

1.1 System Engineering Process

The vehicle was developed as a competition-focused autonomous ground vehicle for the IGVC AutoNav Challenge. The design process began by reviewing the AutoNav course requirements, qualification requirements, vehicle size limits, speed limits, safety requirements, payload requirement, and design-report scoring categories. These rules were converted into measurable engineering requirements for the mechanical system, safety system, electrical/electronic system, perception system, driving logic, and key performance indicators.

The system was divided into six major subsystems: mechanical chassis and drivetrain, electrical power distribution, embedded control, sensor perception, autonomous driving logic, and safety. Each subsystem was selected and tested based on whether it supported the vehicle's ability to qualify and operate on the AutoNav course. The main competition goal is not just to build a robot that moves, but to build a vehicle that can perceive lane boundaries, detect obstacles, follow a safe path, operate within the speed limits, carry the required payload, and stop immediately when commanded by mechanical or wireless emergency stop systems.

The design followed an iterative process. First, major competition constraints were identified. Second, components were selected based on cost, availability, simplicity, and ability to meet the requirements. Third, each subsystem was tested individually. Finally, the system was brought together through integrated testing so that sensor data from the Raspberry Pi could be converted into motion commands for the Arduino and Cytron motor driver.



1.2 Design Concept and Autonomous Algorithm Approach

The FIREBIRD vehicle was designed around a simple and reliable autonomous navigation concept: use lightweight perception, conservative decision-making, and hardware-level safety to complete AutoNav qualification and prepare for outdoor course operation. Instead of relying on a high-cost commercial robotic base or GPU-heavy autonomy system, the vehicle uses a low-cost differential-drive platform controlled by a Raspberry Pi and Arduino Mega. This concept supports the main AutoNav needs: detecting lane boundaries, identifying obstacles, maintaining low-speed motion, supporting GPS waypoint navigation, and stopping safely when required.

The autonomous algorithm approach follows a layered structure. The Raspberry Pi operates as the high-level decision processor, receiving input from the camera, LiDAR, GPS, and IMU. The camera supports lane-detection logic using OpenCV, while the LiDAR provides obstacle-distance information divided into front, left, and right detection zones. GPS data supports waypoint awareness, and IMU data supports heading feedback. These sensor inputs are converted into simple motion decisions such as forward, left correction, right correction, obstacle stop, or waypoint correction. The Arduino Mega then executes these commands through PWM and direction signals sent to the Cytron motor driver.

This approach was selected because it matches the current stage of the vehicle and the available computing resources. Classical Computer vision and zone-based LiDAR logic are easier to validate within the final two-week testing period than a deep-learning or full-SLAM system. The goal is to first achieve reliable AutoNav qualification behavior, then improve full-course performance through outdoor testing, tuning, and additional sensor fusion.

1.3 Requirement Summary

Requirement Area	Requirement	Rule / Design Driver	Target Value	Current / Most Recent Measured Value	Status
Mechanical	Vehicle length within IGVC range	AutoNav vehicle size limit	3–7 ft	3 ft length	Meets design target
Mechanical	Vehicle width within IGVC range	AutoNav vehicle size limit	2–4 ft	2 ft width	Meets design target
Mechanical	Payload support	IGVC payload requirement	20 lb	Chassis designed with payload platform	20 lb payload test
Safety	Mechanical E-stop	IGVC hardware E-stop requirement	Immediate motor cutoff	Wired E-stop path designed and integrated	Hardware pah interated
Safety	Wireless E-stop	IGVC wireless E-stop requirement	≥100 ft range	Wireless E-stop included in final safety plan	Final range test needed
Safety	Safety light	IGVC visible power/autonomy light requirement	Solid on power, flashing in autonomous mode	Included in final checklist	Pending
Electrical	Separate motor and logic power	Reliability and noise reduction	Protected branches	Battery, fuses, terminal blocks, buck converters assembled	Meets design target
Electrical	Stable 5V/logic power	Raspberry Pi and sensor reliability	Stable regulated output	Buck converter system assembled and tested	Meets design target
Electrical	Motor driver capacity	Drive motors require high current control	≥30A class motor driver	Cytron MDDS30 installed	Meets design target
Perception	Lane detection	AutoNav lane following	Detect visible white lane boundaries	OpenCV line detection tested	Pending
Perception	Obstacle detection	AutoNav obstacle avoidance	Detect obstacle before collision	RPLiDAR scan data and obstacle response tested	Operational, needs tuning

Driving Logic	Minimum speed compliance	AutoNav qualification	>1 mph	Low-speed motion demonstrated	Meets design target
Driving Logic	Maximum speed compliance	AutoNav safety limit	<5 mph hardware/software governed	Low-speed command system used	Meets design target
Driving Logic	Waypoint support	AutoNav waypoint navigation	Navigate to 2 m waypoint area	GPS data acquired	Pending
KPI	Qualification readiness	IGVC qualification criteria	Pass lane, obstacle, speed, E-stop, waypoint checks	Subsystems validated; full course not yet complete	Pending

The current vehicle has demonstrated the foundation required for AutoNav competition readiness: sensor acquisition, embedded communication, motor actuation, power distribution, drivetrain movement, and emergency-stop architecture.The current vehicle has demonstrated the foundation required for AutoNav competition readiness: sensor acquisition, embedded communication, motor actuation, power distribution, drivetrain movement, and emergency-stop architecture. Over the next two weeks, the remaining work will focus on final tuning, qualification-style speed testing, wireless E-stop validation, payload testing, safety-light validation, outdoor lane-following, LiDAR stop-command tuning, and integrated waypoint navigation.

2. Mechanical Design

2.1 Frame and Chassis Structure

The mechanical platform uses a modular structural-channel chassis instead of a fully custom welded or machined frame. This approach was selected because it allowed rapid fabrication, adjustment, and repair during the build process. The frame supports the drivetrain, electronics enclosure, battery, sensors, payload area, and emergency stop hardware while remaining simple enough to modify as the integration process continues.

The chassis was designed around the AutoNav vehicle size limits. The vehicle must remain wide enough for stability and payload support, but not so large that it becomes difficult to maneuver through lane boundaries, obstacles, turns, and narrow course sections. The open-frame layout also improves access to wiring, motor mounts, battery placement, and drivetrain alignment.

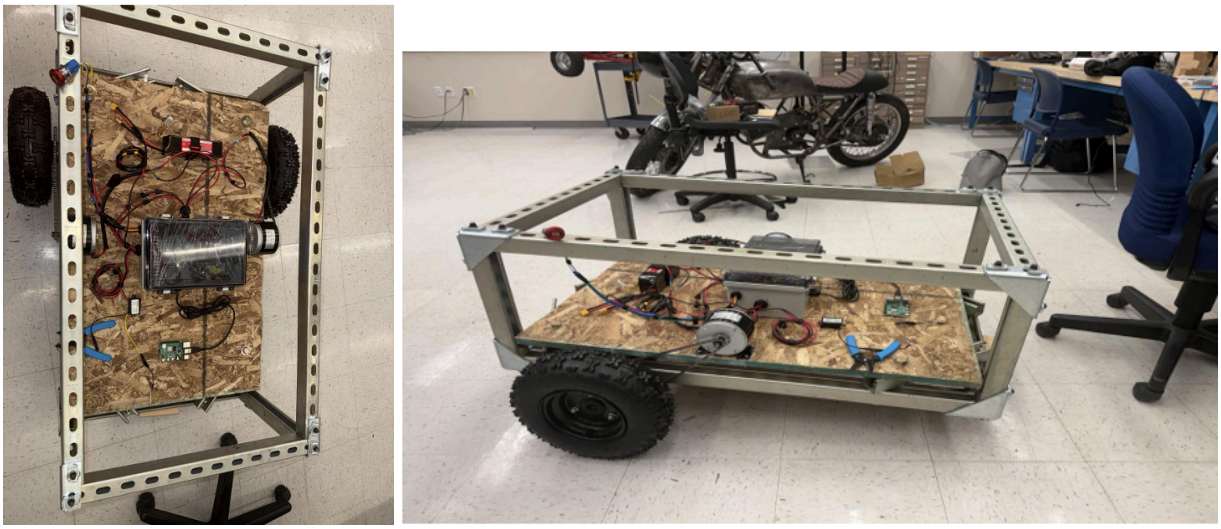


Figure 2: View chassis layout showing frame dimensions, wheel positions, caster position, battery location, electronics enclosure, payload area, LiDAR/camera location, and E-stop location.

2.2 Drivetrain Configuration

The vehicle uses a differential-drive system with two independently powered drive wheels and a caster support. Differential drive was selected because it reduces steering complexity and allows the robot to turn by varying left and right wheel speeds. This is practical for a student-built AutoNav vehicle because it avoids the added fabrication requirements of a steering rack, servo steering linkage, or Ackermann-style steering system.

Power is transmitted through a chain-and-sprocket drivetrain. This choice provides better torque transfer than a belt system and allows the motors to drive the wheels through a simple mechanical layout. The drivetrain includes DC motors, sprockets, chain, shafts, wheel hubs, bearings, and motor mounts.

2.3 Mechanical Design Decisions

Several mechanical decisions were driven by competition requirements and practical build constraints:

- The chassis was built using structural channel because it is adjustable, available, and strong enough for prototype-level testing.
- The battery and heavier electronics are planned low on the frame to improve stability.
- The drivetrain was kept mechanically simple so that alignment issues could be adjusted during testing.
- The platform includes enough space for the 20 lb payload and the main electronics enclosure.
- The frame is open enough for serviceability, which matters during competition when quick repairs and inspections may be needed.

The most significant mechanical challenge has been drivetrain alignment. Chain tension, sprocket alignment, shaft placement, and bearing position all affect motion accuracy. During testing, movement drift was observed, especially during reverse operation. This suggests that final tuning should focus on chain tension, wheel alignment, motor response matching, and eventually encoder feedback.

2.4 Requirement Comparison

Mechanical Requirement	Target	Current Value / Evidence	Interpretation
Length	3–7 ft	3ft	Vehicle is sized for competition rules.
Width	2–4 ft	2ft	Width supports stability while remaining within rules.
Payload capacity	20 lb	Meets design target	Structure appears suitable but should be formally tested.
Serviceability	Components reachable	Open-frame layout and enclosure access	Good for debugging and competition inspection.
Drivetrain operation	Forward, reverse, left, right, stop	Floor movement demonstrated	Basic mobility works; final tuning remains.

3. Safety

3.1 Safety Philosophy

Safety was treated as a primary competition requirement rather than an optional feature. Because the vehicle is powered by a high-energy LiPo battery and uses DC motors capable of moving the platform under load, the safety design must stop the vehicle even if the software freezes, the Raspberry Pi stops responding, or serial communication fails.

The safety design focuses on hardware-level motor shutdown, fuse protection, wiring organization, and visible operational indication. The intended safety behavior is simple: if unsafe behavior occurs, motor power must be removed quickly and the vehicle must stop.

3.2 Mechanical and Wireless E-Stop

The mechanical E-stop is designed as a hardware-level stop path for the motor system. Instead of relying only on a software stop command, the E-stop interrupts motor power through the relay or contactor path. This is important because IGVC expects emergency stop systems to be hardware based and capable of bringing the vehicle to a quick and complete stop.

The wireless E-stop is included as a required final safety item. During competition, the wireless E-stop is held by the judges, so the system must be reliable and effective at the required range. Final testing must verify the range, response time, and actual motor cutoff behavior.

3.3 Transport, Parking, and Charging Safety

While being transported or parked, the vehicle should remain powered off, with battery leads disconnected or isolated when possible. The LiPo battery should be secured to prevent movement and should not be charged while loosely mounted or near exposed wiring. Charging should be performed using the balance charger and supervised procedures. Exposed terminals, loose connectors, and unsecured wiring must be corrected before full competition operation.

During operation, the safety light must show vehicle state. The final design should use a solid light when vehicle power is on and a flashing light when autonomous mode is active. This helps judges, operators, and bystanders understand whether the vehicle is only powered or actively operating autonomously.

3.4 Requirement Comparison

Safety Requirement	Target	Current Value / Evidence	Interpretation
Mechanical E-stop	Hardware motor cutoff	E-stop wiring and relay path integrated	Meets design target
Wireless E-stop	≥100 ft range	Wireless E-stop planned/included	Pending
Safety light	Solid powered, flashing autonomous	Included in final checklist	Pending
Fuse protection	Protect motor and logic circuits	Fuse protection assembled	System has electrical protection foundation.
Safe charging	Controlled LiPo charging	Balance charger available	Use supervised charging procedure.

4. Electrical and Electronic Design

4.1 Electrical Architecture

The electrical system is built around a 22.2V LiPo battery that supplies power to both the motor branch and the logic/sensor branch. The power system uses fuse protection, terminal blocks, regulated buck converters, and separated wiring paths. The high-current motor path powers the Cytron MDDS30 motor driver and the drive motors. The logic path powers the Raspberry Pi, Arduino Mega, and sensors through regulated voltage outputs.

This separation is important because motor current spikes can create electrical noise and voltage drops. By separating motor power from low-voltage logic power, the system reduces the risk that motor behavior will reset or disrupt the Raspberry Pi and sensors.

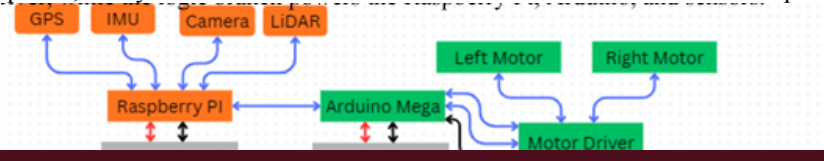


Figure 5: Electrical block diagram showing LiPo battery, master fuse, terminal distribution, buck converters, Raspberry Pi, Arduino Mega, LiDAR, GPS, IMU, camera, motor driver, motors, and E-stop path.

4.2 Embedded Control Components

The Raspberry Pi functions as the high-level processor. It handles camera processing, LiDAR reading, GPS data, IMU data, and navigation decision logic. The Arduino Mega functions as the low-level motor controller. It receives serial commands from the Raspberry Pi and converts them into PWM and direction signals for the motor driver.

The Cytron MDDS30 motor driver controls the two DC motors. This driver was selected because it provides a strong current rating for a student-built vehicle and supports simple control from the Arduino. The command-based structure makes early testing easier because the Raspberry Pi can send clear movement commands such as forward, left, right, stop, and speed-level changes.

4.3 Power System Performance

The power system has been assembled and tested through subsystem-level and integrated operation. Battery power, terminal distribution, buck conversion, motor driver operation, and controller wiring have all been validated. During floor testing, startup current spikes caused fuse failures when the motors accelerated under load. To reduce this issue, software-based soft-start ramping was added to the Arduino motor-control logic.

This result shows that the power system is functional but requires final fuse coordination and controlled acceleration. For competition operation, the vehicle should avoid sudden full-current starts and should use conservative acceleration profiles.

4.4 Requirement Comparison

Electrical Requirement	Target	Current Value / Evidence	Interpretation
Battery source	Onboard power	22.2V LiPo battery used	Meets onboard power requirement.
Motor driver	High-current dual motor control	Cytron MDDS30 installed/tested	Suitable for two-motor differential drive.
Logic regulation	Stable low-voltage power	Buck converters assembled/tested	Supports Pi, Arduino, and sensors.
Noise reduction	Separate motor/logic wiring	Separate branches used	Improves reliability.
Fuse protection	Protect against overcurrent	Fuses installed; startup failures observed	Soft-start added; final fuse sizing required.

5. Perception

5.1 Sensor Suite Overview

The AutoNav perception system uses camera vision, LiDAR, GPS, and IMU data. Each sensor supports a specific part of the navigation problem. The camera supports lane boundary detection. The LiDAR supports obstacle detection and distance awareness. The GPS supports waypoint navigation. The IMU supports heading and orientation feedback.

This combination was selected because AutoNav requires the vehicle to remain inside lane boundaries, avoid obstacles, and navigate toward GPS waypoints. A single sensor would not be reliable enough for all of these tasks. The multi-sensor approach provides a more complete view of the course while staying within the project's budget and computing limits.

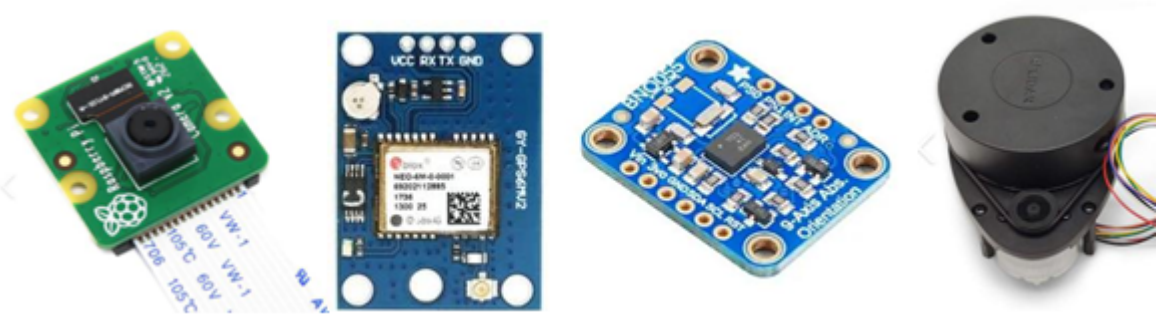


Figure 7: Sensor suite photo showing camera, RPLiDAR A1, U-Blox NEO-M8N GPS, and BNO055 IMU

5.2 Lane Detection

Lane detection is performed using the Raspberry Pi camera and OpenCV. The processing pipeline uses grayscale conversion, edge detection, a region of interest, and line detection to identify visible course boundaries. The output is converted into directional guidance, such as staying centered, correcting left, correcting right, or stopping if lines are lost.

The current line detection works best in controlled lighting. Outdoor lighting changes, shadows, glare, worn tape, and camera vibration can reduce accuracy. For the competition version, the camera should be tested under outdoor conditions similar to the AutoNav course, and threshold values should be tuned for white lane markings on asphalt.

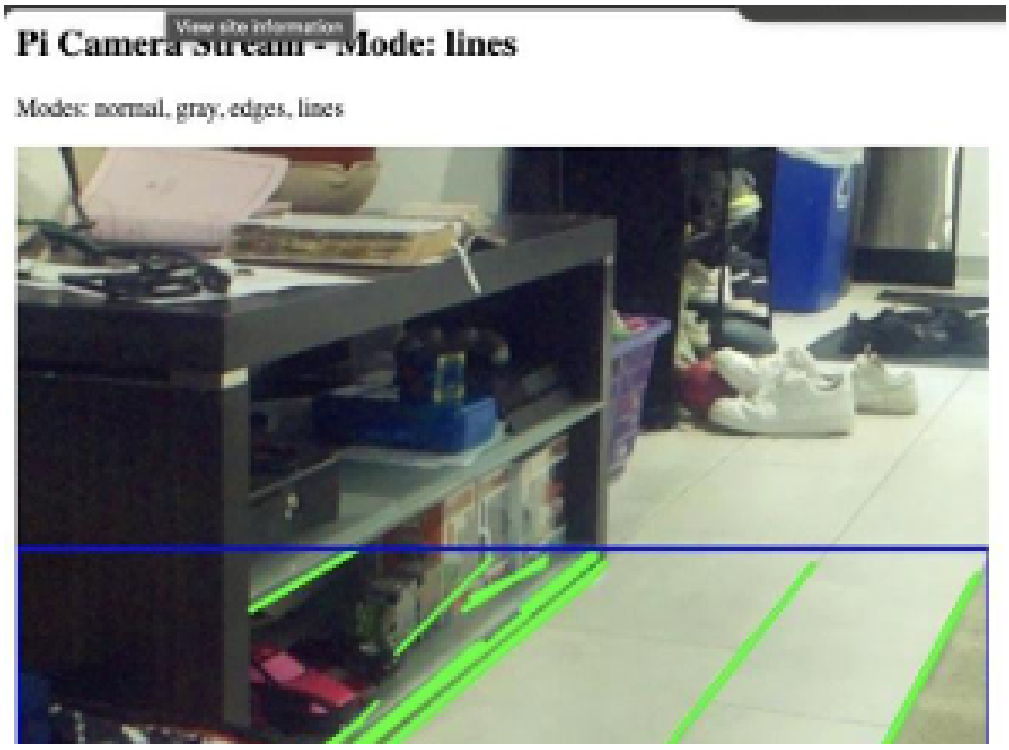


Figure 8: OpenCV lane detection screenshot showing camera frame beside processed edge/line output.

5.3 Obstacle Detection

Obstacle detection is performed with the RPLiDAR. The scan is divided into front, left, and right regions. If an object is detected in the front safety zone, the vehicle can stop or select a turning response based on which side has more clearance. This is directly tied to AutoNav course requirements because obstacles may appear at random locations and must be avoided without touching or displacing them.

LiDAR testing confirmed that scan data can be read and interpreted. The remaining challenge is stabilizing the real-time obstacle-response loop during continuous vehicle motion. In some integrated tests, the LiDAR generated stop decisions, but the drivetrain continued moving. This indicates that final debugging should focus on command priority, serial timing, stop-command handling, and ensuring the Arduino immediately overrides motion when stop commands are received.

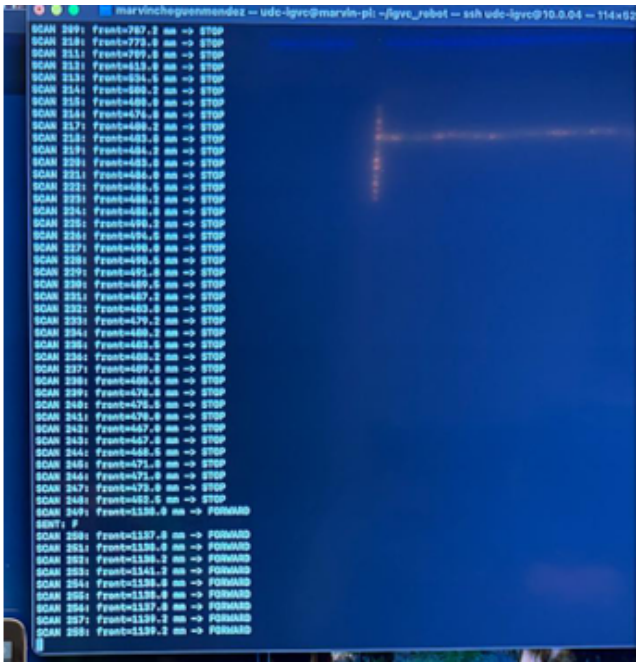


Figure 9: LiDAR scan output showing obstacle readings divided into front, left, and right detection zones.

5.4 Internal Course Representation

The vehicle uses a lightweight internal representation rather than a complex map. This is appropriate because AutoNav rules do not allow course memorization or mapping for a fixed course layout. The representation is updated in real time and contains:

- Lane-center estimate from camera processing
- Front obstacle distance from LiDAR
- Left and right clearance from LiDAR
- GPS position and waypoint bearing
- IMU heading estimate
- Current movement command
- Safety state

This representation is refreshed as the vehicle moves. The driving logic uses the most recent values to decide whether to continue forward, adjust steering, stop, turn around an obstacle, or move toward a waypoint.

5.5 Requirement Comparison

Perception Requirement	Target	Current Value / Evidence	Interpretation

Lane detection	Detect lane boundaries	OpenCV line detection tested	Works in controlled lighting; needs outdoor tuning.
Obstacle detection	Detect obstacles before collision	RPLiDAR scan data validated	Functional; stop logic needs final integration tuning.
GPS waypoint support	Provide position data	GPS data acquired	Supports future waypoint behavior.
Heading feedback	Provide orientation data	IMU output validated	Supports directional correction.

6. Driving Logic

6.1 AutoNav Driving Strategy

The AutoNav driving strategy is designed around conservative, low-speed autonomous operation. The vehicle does not attempt aggressive navigation. Instead, it prioritizes staying inside lane boundaries, avoiding obstacles, maintaining safe speed, and stopping when confidence is low.

The Raspberry Pi acts as the decision layer. It reads camera, LiDAR, GPS, and IMU data, then sends simple motion commands to the Arduino. The Arduino acts as the execution layer, converting those commands into motor driver signals.

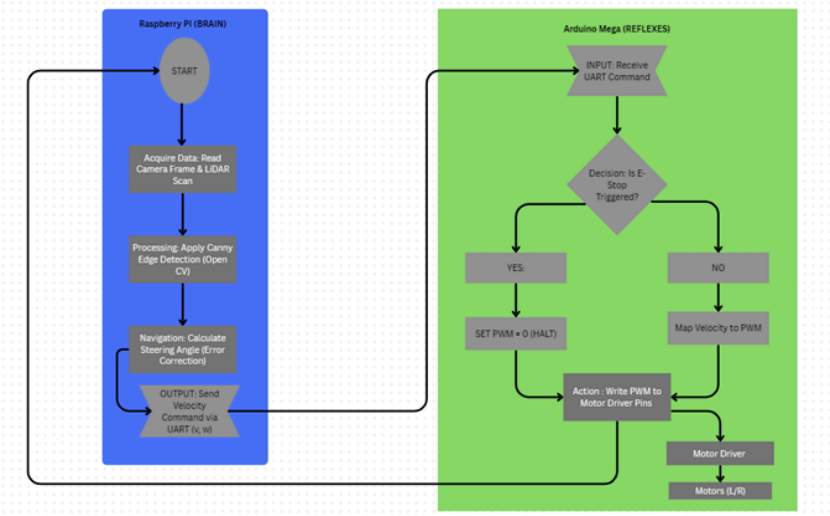


Figure 10: Driving logic flowchart: read sensors → check E-stop/safety → check obstacle → check lane position → check waypoint bearing → send motor command → monitor response.

6.2 Lane Following

Lane following uses camera output to estimate whether the vehicle is centered between lane boundaries. If the detected lane center shifts left or right, the Raspberry Pi sends correction commands to the Arduino. With differential drive, corrections are made by changing relative motor speeds or by using discrete left/right commands during early testing.

The initial command-based method is simple and reliable for debugging. The competition-ready version should improve this behavior by smoothing commands and adding proportional steering corrections.

6.3 Obstacle Avoidance

Obstacle avoidance uses LiDAR distance zones. If an obstacle is detected in the front zone, the vehicle stops or slows. The system compares left and right clearance to decide the safer avoidance direction. After turning away from the obstacle, the vehicle should return to lane-following behavior.

This behavior is especially important because AutoNav obstacles can be randomly placed and may include barrels, natural obstacles, manmade obstacles, and simulated potholes. The vehicle must avoid touching or displacing obstacles because collisions result in penalties or end-of-run conditions.

6.4 GPS Waypoint Navigation

GPS waypoint support is included for the AutoNav qualification and competition requirement. The GPS module provides position data, and the IMU provides heading support. The intended waypoint logic calculates the bearing from the current GPS location to the target waypoint, compares it with the current heading, and generates a correction command.

The current system has acquired GPS data, but final waypoint navigation still requires outdoor integrated testing. The target is to navigate toward a two-meter waypoint region while still using LiDAR and camera data to avoid obstacles and remain inside course boundaries.

6.5 Ramp, Switchbacks, Dead Ends, and Potholes

For the AutoNav course, the vehicle must be prepared for ramps, repeated barrels, switchbacks, center islands, dead ends, traps, and pothole-like white circles. The current strategy handles these features through conservative sensor-based behavior:

- Ramps: maintain slow speed and use chassis stability plus IMU feedback to monitor orientation.
- Switchbacks: rely on lane detection and low-speed turning.
- Center islands/dead ends: use LiDAR clearance to identify blocked paths and choose open space.
- Potholes: treat white circular regions as avoidable course hazards during camera processing in future work.
- Obstacles: use LiDAR zones to stop, turn, or route around objects.

This approach is not fully optimized yet, but it gives the vehicle a realistic competition path: qualify first, then improve course distance and stability.

6.6 Requirement Comparison

Driving Logic Requirement	Target	Current Value / Evidence	Interpretation
Minimum speed	>1 mph	Motion demonstrated; formal 44-ft test needed	Meets design target
Maximum speed	<5 mph	Low-speed commands used	Meets design target
Lane following	Detect/follow lanes	Camera line detection tested	Pending
Obstacle avoidance	Detect/avoid obstacles	LiDAR stop logic tested	Pending
Waypoint navigation	Navigate to 2 m waypoint	GPS output validated	Pending

7. Key Performance Indicators

The key performance indicators were selected based on what matters for AutoNav qualification and competition performance. The first goal is qualification readiness. After qualification, the next goal is improving distance traveled, reducing penalties, and increasing reliability.

KPI	Measurement Method	Target	Current Status	Meaning
E-stop response	Trigger wired/wireless stop while moving	Immediate motor cutoff	Wired path integrated; wireless pending	Required for safe operation.
44-ft speed test	Time vehicle over 44 ft	>1 mph and <5 mph	Needs formal test	Required for qualification.
Lane detection reliability	Count successful detections over repeated runs	≥80% controlled-course detection	Controlled test successful	Outdoor tuning needed.
Obstacle detection	Place obstacle at known distances	Detect before collision zone	LiDAR scan validated	Command response needs

distance				tuning.
Command latency	Time from Pi command to motor action	Low enough for safe movement	Serial control works	Needs measured latency test.
Runtime	Continuous operation on battery	Enough for qualification/testing sessions	Battery system assembled	Full runtime test needed.
Payload stability	Drive with 20 lb payload	Payload remains secured	Payload area designed	Formal load test needed.
Course readiness	Qualification-style checklist	Pass E-stop, speed, lane, obstacle, waypoint	In progress	Main readiness indicator.

8. Analysis of Complete Vehicle

8.1 Lessons Learned During Integration

The complete vehicle confirmed that autonomous robot development is mainly a system integration challenge. Individual components can work separately, but the hardest part is making the drivetrain, power system, sensors, controllers, and safety hardware work together on one moving platform.

A major lesson was that mechanical alignment directly affects electrical and software behavior. If one chain is tighter than the other, or if one wheel responds differently, the software command may be correct but the robot may still drift. Another lesson was that power startup behavior matters. Sudden motor acceleration caused current spikes and fuse issues, which required a soft-start solution in software.

The testing process also showed why competition readiness should be measured with real driving tests instead of only bench tests. The camera, LiDAR, GPS, IMU, Arduino, and motor driver can all produce correct outputs individually, but the full system still needs tuning when the robot is moving under load.

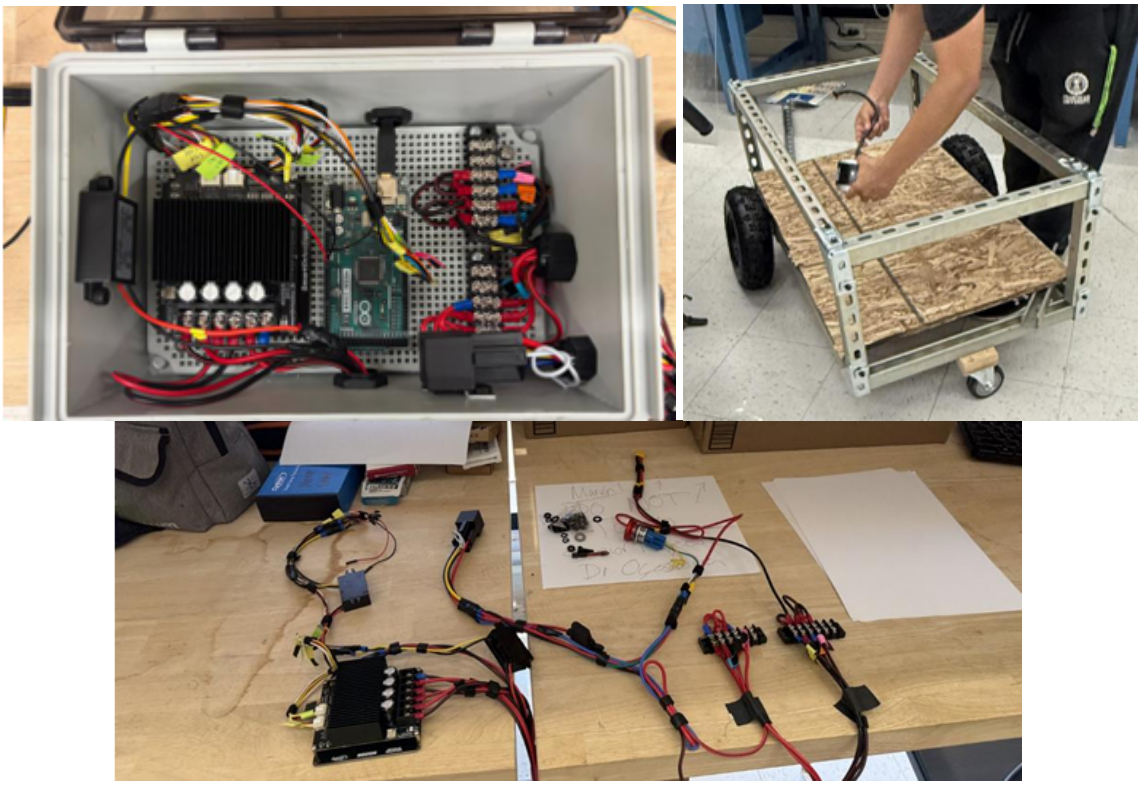


Figure 12: Integration photos showing the electronics enclosure, battery, drivetrain, chassis structure, and E-stop hardware used during final system assembly.

8.2 Component Failures and Corrective Actions

The most important issue observed during testing was fuse failure caused by motor startup current spikes. The corrective action was to reduce sudden acceleration by adding soft-start ramping in the Arduino motor control code. This improved electrical stability and reduced the likelihood of abrupt current demand.

Another issue was drivetrain drift. The likely causes were sprocket alignment, chain tension, wheel alignment, and unmatched motor response. Corrective actions include realigning the drivetrain, adjusting chain tension, checking bearing placement, and adding encoder feedback in future development.

A third issue was LiDAR stop-command reliability during integrated motion. The LiDAR successfully generated obstacle readings and stop decisions, but final command priority needs improvement so stop commands always override movement commands.

8.3 Software Testing and Version Control

Software was tested through individual scripts before full integration. Separate tests were used for camera processing, LiDAR scanning, GPS output, IMU output, serial communication, and motor driver control. This modular approach made it easier to isolate problems. The next version should use structured version control with clear commits for sensor tests, motor-control updates, safety logic, and integrated autonomy code.

8.4 Simulation-Based Testing

A full Gazebo or ROS simulation was not completed during this build cycle. Instead, the project used physical subsystem testing and controlled indoor testing. Future work should include simulation of lane following, obstacle avoidance, and waypoint behavior before running the vehicle outdoors. Simulation would reduce risk and allow faster testing of navigation logic.

8.5 Physical Testing

Physical testing completed to date includes:

- Raspberry Pi setup and sensor script testing
- Arduino serial communication testing
- Motor driver response testing
- Forward, reverse, left, right, and stop movement commands
- Power distribution and voltage regulation testing
- Camera-based line detection testing
- IMU output testing
- GPS output testing
- LiDAR scan and obstacle detection testing
- Floor movement under vehicle load

The vehicle is therefore not just a paper design. It is a built and partially integrated autonomous platform entering its final two-week validation period. The next step is formal AutoNav qualification-style testing, including E-stop validation, speed verification, payload testing, outdoor lane-following, obstacle-response tuning, and waypoint navigation.

9. Cybersecurity Analysis

Although this vehicle is a student-built prototype, it still contains cybersecurity risks because it uses embedded controllers, wireless safety hardware, software scripts, serial communication, and potentially network access through the Raspberry Pi.

9.1 Vulnerability 1: Raspberry Pi Network Access

The Raspberry Pi may be accessed through Wi-Fi, hotspot, SSH, or local network connections during development. If this access is left unsecured, an unauthorized user could connect to the vehicle, modify scripts, interrupt operation, or trigger unsafe behavior.

Hardening Step: Disable unnecessary network services before competition, use strong passwords, avoid open Wi-Fi networks, restrict SSH access, and run only required services during testing.

9.2 Vulnerability 2: Unprotected Serial Command Interface

The Raspberry Pi sends movement commands to the Arduino through serial communication. If unexpected or corrupted commands are sent, the Arduino may continue movement or fail to stop when expected.

Hardening Step: Add command validation, timeouts, heartbeat checks, and a default stop state. If commands are not received within a short time window, the Arduino should stop the motors automatically.

9.3 Vulnerability 3: Wireless E-Stop Reliability and Interference

The wireless E-stop is a critical safety device. Wireless systems can experience interference, range limitations, receiver issues, or battery failure.

Hardening Step: Test wireless E-stop range before every run, use a fail-safe receiver configuration, verify battery charge, and ensure that wireless loss results in motor cutoff rather than continued motion.

9.4 Vulnerability 4: Software Configuration Drift

During testing, quick code changes can cause inconsistent behavior if old scripts, untested updates, or incorrect versions are run by mistake.

Hardening Step: Use version control, label tested scripts, maintain a competition branch, and document the exact startup procedure used during qualification.

10. Conclusion

This report presents the design and current readiness of a low-cost autonomous ground vehicle developed for the IGVC AutoNav Challenge. The vehicle is built around a modular structural-channel chassis, differential-drive drivetrain, Raspberry Pi high-level processor, Arduino Mega low-level controller, Cytron MDDS30 motor driver, 22.2V LiPo power system, RPLiDAR obstacle detection, camera-based lane detection, GPS waypoint support, IMU heading feedback, and hardware-level emergency stop architecture.

The system has demonstrated the core functions needed for continued AutoNav development. The Raspberry Pi and Arduino communicate successfully, the motor driver responds to movement commands, the drivetrain moves the vehicle under load, the camera detects lane markings in controlled conditions, the LiDAR produces scan data for obstacle detection, and GPS/IMU sensors provide usable navigation information.

The vehicle has reached the final qualification-preparation stage, where the remaining work is focused and measurable. The highest-priority tasks before AutoNav qualification are mounted E-stop validation, wireless E-stop range testing, safety light installation, 44-ft speed verification, 20-lb payload testing, outdoor lane-following validation, LiDAR stop-command tuning, and integrated waypoint navigation.

The strongest part of the design is that it creates a complete, student-built, expandable foundation for UDC's IGVC participation. Instead of relying on a commercial robotic base, the project demonstrates actual mechanical fabrication, electrical power design, embedded control, sensor integration, and autonomous behavior development. With the planned two-week final tuning and qualification-style testing period, the platform is positioned to continue toward IGVC AutoNav readiness while also providing future UDC teams with a practical base for improvement.

References

[1] Intelligent Ground Vehicle Competition, official competition rules and design guidance.

[2] Raspberry Pi Foundation, Raspberry Pi hardware and camera documentation.

[3] Arduino, Arduino Mega 2560 hardware and software documentation.

[4] Cytron Technologies, MDDS30 Dual Channel DC Motor Driver documentation.

[5] Slamtec, RPLiDAR A1 product documentation.

[6] Adafruit Industries, BNO055 Absolute Orientation Sensor documentation.

[7] U-Blox, NEO-M8N GPS module documentation.

[8] OpenCV, image processing and Hough line transform documentation.