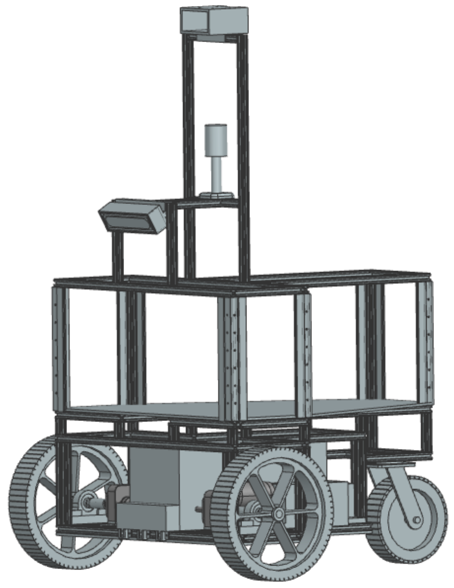


"Blue"

5/15/2023

Team Lead

Kyle Tomczik| ktomczik@ltu.edu+

Design Lead

Yeen Lee | ylee1@ltu.edu

Electrical Lead

Marc Duckworth | mduckwort@ltu.edu

Fabrication Lead

Erica Morrissey | emorrisse@ltu.edu

Programming Lead

Brenden Priess| bpreiss@ltu.edu

Members

Noah Combs | ncombs@ltu.edu
Hannah Comilla| hcomilla@ltu.edu
Jacob Devereaux | jdevereau@ltu.edu
Sydney Schmid | sschmid@ltu.edu
Joshua Standifer | jstandife@ltu.edu
Victor Queinnec | vqueinnec@ltu.edu

Faculty Advisor

Giscard Kfoury, Ph.D. | gkfoury@ltu.edu

"I certify that the design and engineering of "Blue" by the 2022/2023 Lawrence Technological University Robotics Team has been significant and equivalent to what might be awarded credit in a senior design course".

Signature: Date:

[illegible]

Prof. Giscard Kfoury, Ph.D. May 15, 2023

1. Introduction

Senior robotics students from Lawrence Technological University (LTU) have been actively involved in the Intelligent Ground Vehicle Competition (IGVC), a yearly event that aims to challenge student lead teams to design and build a fully autonomous ground vehicle from scratch that is capable of navigating an outdoor environment. This competition presents an exceptional opportunity for students to apply their knowledge and personal skills in robotics engineering and programming to a real-world scenario. Our vehicle, dubbed '*Blue*,' has been designed to participate in the 2023 IGVC competition. The cost of building '*Blue*' is given in Table 1:

Item	Retail	Team Cost
Computer	$1133.00	Donation
HMI	$1357.10	Donation
GPS	$5000.00	Donation
ZED Camera	$350.00	Donation
Safety Light	$15.00	$15.00
Emergency Stop	$320.00	Donation
Arduino Mega	$48.99	$48.99
Arduino Uno	$24.99	Donation
RC Receiver & Controller	$50.00	$50.00
Wheels	$300.00	$300.00
Frame	$345.00	$345.00
Motors	$384.00	Donation
Motor Controller	$180.00	$180.00
Electrical Components	$325.68	$325.68
Weather Proofing	$303.75	$303.75
3D Filament	$104.00	$104.00
IMU	$35.00	Donation
Total	$10,266.51	$1,662.42

Table 1. Cost of Blue

2. Team Organization

Our IGVC team consists of undergraduate students overseen by Dr. Giscard Kfoury and Dr. Gaurav Singh. We have four sub-teams, each with its own lead: fabrication, design, electrical, and programming. In order to ensure that each member is fully engaged, they must be on at least two sub-teams. Each sub-team reports progress to the team captain weekly, who reports to the advisors. Regular meetings and multiple methods of communication ensure that progress is monitored and any potential issues are addressed. This structure enables the team to work effectively and efficiently towards creating a successful autonomous ground vehicle for the competition.

Member	Design	Electrical	Fabrication	Programming	Hours
Noah Combs			x	x	95+
Hannah Comilla	x		x		90+
Jacob Devereaux		x	x		180+
Marc Duckworth		Lead	x		185+
Yeen Lee	Lead	x			140+
Erica Morissey		x	Lead		85+
Brenden Preiss		x		Lead	120+
Sydney Schmid	x			x	65+
Joshua Standifer	x			x	130+
Kyle Tomczik	x	x	x	x	100+
Victor Queinnec	x			x	90+

Table 2. Team Member Contribution

2.1 Design Process

The goal for our team is to complete the IGVC 2023 course without failures. In order to accomplish this, our team would need to implement a fitting model. Our team used the waterfall development model for fabrication, electrical, design, and programming. The figure below depicts our development process step by step.

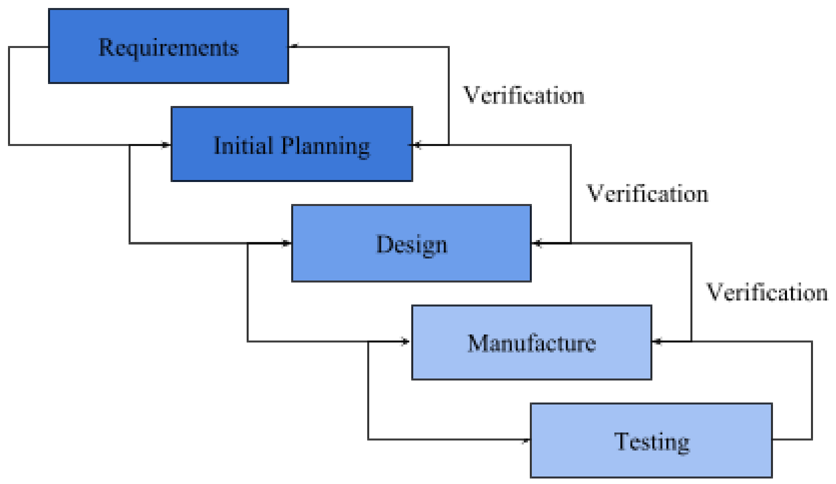


Figure 1. Design Process

Using this method, we quickly identified any potential problems in our design and performed regular verification checks to ensure that progress was moving smoothly.

3. Innovations

Our team's participation in the competition allowed us to showcase our engineering skills and knowledge. It also allowed us to include several new innovations in our robot. In order to distinguish ourselves from other teams, we incorporated new designs and fabrication solutions. These included extruded aluminum T-slot, multiple 3D printing techniques, and water cooling.

Listed below are the innovations' advantages and disadvantages.

3.1 Concepts

3.1.1 Water cooling

Previous teams discovered issues with the robot overheating during runs. This problem could have devastating effects on our computer and also compromise the entire system. To tackle this issue head-on, we implemented a practical solution: a water cooler designed for the CPU. This innovative cooling system enables us to efficiently draw heat away from the primary processing unit, thereby maintaining optimal operating temperatures for the computer. With this addition, we have lowered the risk of significant damage to our computer and system.

3.1.2 3D Printing

3D printing was a major innovation for our team. It allowed us to create prototypes rapidly and significantly accelerated the timeline. It enabled us to prototype and test the custom-designed components with high accuracy and precision. Utilizing multiple different 3D printing filament types, we were able to tailor the materials' properties to suit the design. For example, Stainless Steel PLA was used to fabricate stronger standoffs and the initial motor mounts. Thermoplastic Polyurethane (TPU) was being tested for its properties as a vibration dampener; however, we found a more promising solution that better suited our needs. The disadvantage of using 3D printing was that for the specific materials that we wanted to print with, it required a direct drive system and a hardened steel nozzle which only one of our printers was capable of. Furthermore, TPU proved ineffective as it did not bond with other materials.

4. Mechanical Design

4.1 Overview

The mechanical design was based on the rules specifying the robot's size, speed, payload, and navigational requirements. It must be a ground vehicle with dimensions between 3-7' in length, 2-4' in width, and no taller than 6'. It must be able to carry a 20 lb cinder block with an approximate size of 18" x 8" x 8". The rules state that the run will be terminated if the payload falls out. The robot must also not exceed five mph and include its own onboard power source, indicator light, and E-stop. The E-stop will need to be located in the center rear of the vehicle at least 2' up but no more than 4' high. A wireless E-stop is also required and must be effective from a minimum distance of 100 feet, hardware-based, and not controlled through software. Activation of either E-stop must bring the vehicle to a complete stop. With these rules in mind, the team began the selection of various components as well as initiated the design process accordingly. The graphics below depict the CAD model on which we have based our robot.

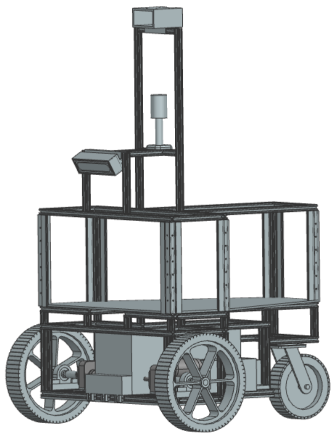


Figure 2. CAD Model

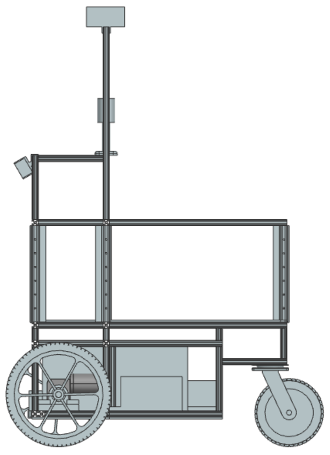


Figure 3. Side View

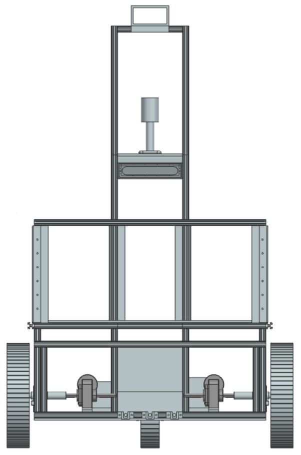


Figure 4. Front View

4.2 Frame Structure

The team decided that the frame would consist of three layers:

- Bottom Layer: The bottom layer of the robot will be carrying most of the weight, including the motors, batteries, and payload. Placing the weight at the lowest point, keeping in mind the center of gravity, allows for a more stable and rigid robot when calculating acting forces. The payload, a cinder block estimated to be 18” x 8” x 8”, requires a minimum height of 8” for the bottom layer. Therefore, the team selected 12.5” utility wheels rated for handling higher loads to ensure clearance and turning radius. These wheels fit the kinematic equations and came with a key for better motor mounting.
- Middle Layer: The middle layer consists of the electrical components on a large aluminum plate that houses the motherboard, Arduino, motor controller, fuse boxes, step-down converter, and the wires coming from the sensors mounted above. The wires from the bottom plate are fed through drilled holes on this plate.
- Top Layer: The top layer is the designated place for our sensors with a flexible design to accommodate all possible sensor arrangements. The finalized sensor array consists of the ZED Camera and the AtlasLink GPS. The ZED camera is to be mounted 36” in the air from ground height and has a 30-degree offset for object detection and lane following. The GPS will be mounted at the highest point of our robot to ensure clear visibility to receive accurate data. A lightstack will also be installed 12” from the base to have a 360-degree field of view.

4.3 Drive Train

Our team researched several drivetrains, including crab drive, Ackerman steering, and tank drive. Ultimately, our team decided that the best drivetrain for the robot would be a three-wheeled differential drive. The differential drive is based on two driving wheels; if they both move forward, the vehicle moves forward; if the right wheel moves forward and the left moves backward, the vehicle turns left and vice versa. The team chose this drivetrain for a few reasons. The first reason was that we had the most experience with three-wheeled differential drive robots, having already built one in a previous project. Secondly, the kinematics, controls, and turning radius equations were the easiest to understand and calculate. Lastly, the three-wheeled differential design is the competition's most successful design.

4.4 Weatherproofing

For the ‘Intro to Capstone Projects' class, the 2024 IGVC students were tasked with designing and fabricating a weatherproofing and cooling system for the robot as it is required to compete in the presence of light rain. This means that the electrical components must be housed and protected from water. The weatherproofing consists of quarter-inch panels of blue acrylic connected by M-5 bolts that pair with our extruded aluminum frame. The front access panel is removable and held in place with two magnets. The magnets allow the electrical team to access the components quickly and comfortably. The team has cut 2.5” x 2.5” holes for ventilating the middle layer. To cover the ventilation hole, the 2024 team designed a 3D-printed vent that allows heat to flow out and keeps water from leaking in. The team also decided to implement cable glands for the cables running down from our sensors. The HMI, ZED Camera, GPS, and lightstack will have their cables routed through the previously mentioned cable glands. The figures below show the completed weatherproofing around our robot.

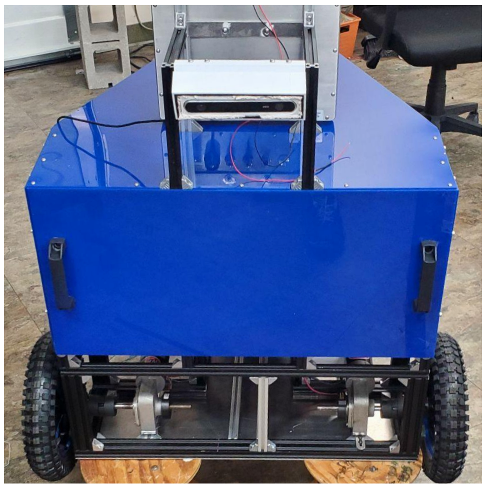


Figure 5. Front View of Waterproofing

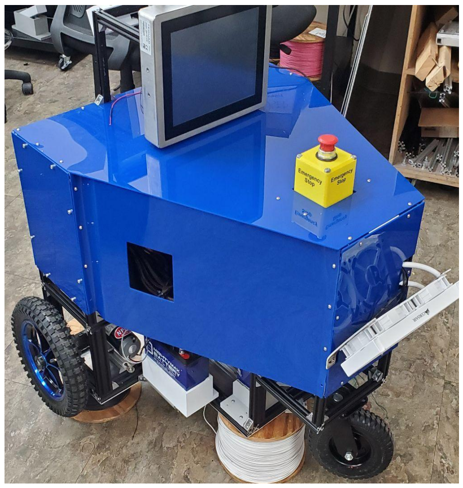


Figure 6. Side View of Waterproofing

5. Electrical Design

5.1 Overview

Our system prioritizes safety, reliability, and quality of life. All components were tested separately in order to ensure proper functionality. Furthermore, despite some voltage drop, they were adequately spaced apart for clean and easy wire management. Our vehicle has three 12V, 22 Ah lead-acid batteries. The first battery powers the drivetrain directly from a powered switch to the motor controller. We use automotive relays to ensure added safety of our motors, E-stop, and motor controller. Additionally, our motors have encoders to confirm the robot’s position accurately. The other two batteries are connected in parallel to a switch, then to a fuse box that powers the computation system. Most components are powered by the 12V fuse box attached to a 12V to 5V step-down converter for lower voltage components. The ZED camera is the only exception and is powered by the computer via USB 3.0.

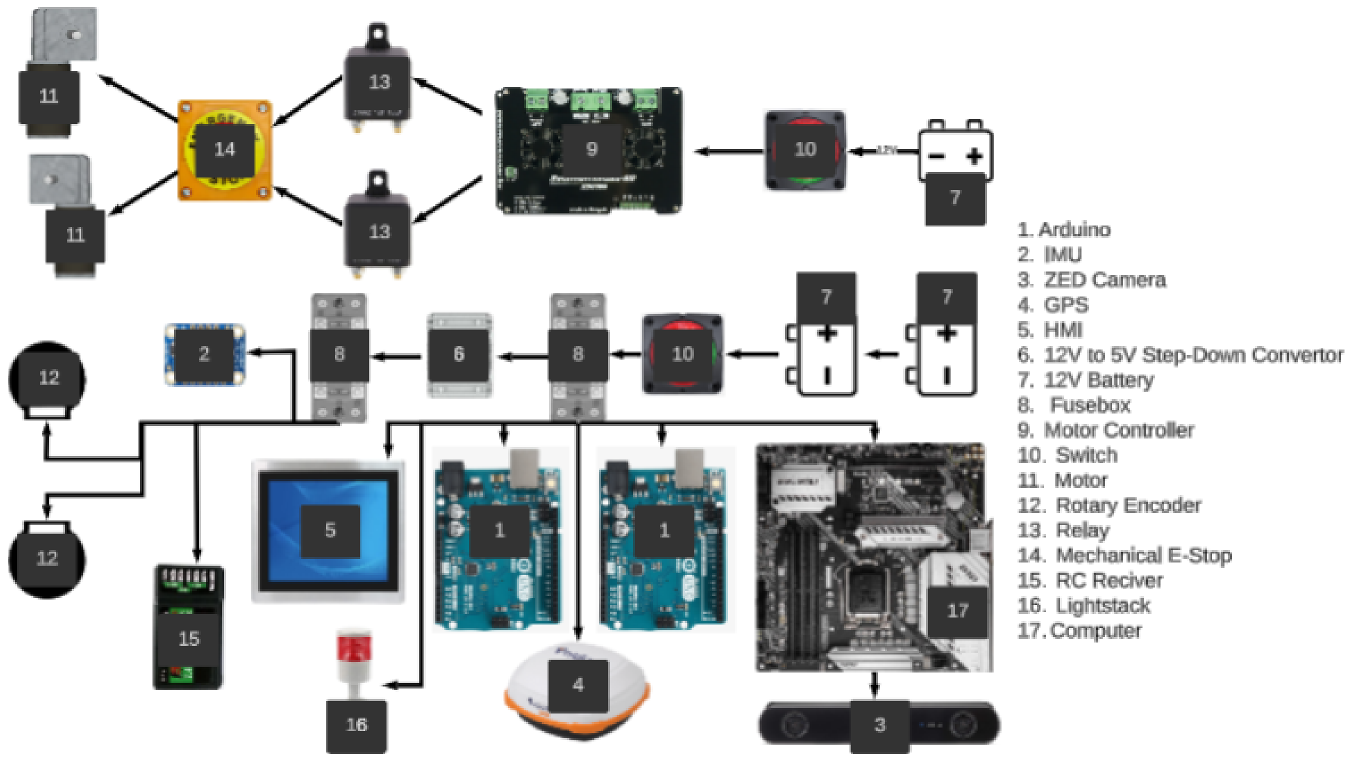


Figure 7. Simplified Power Schematic

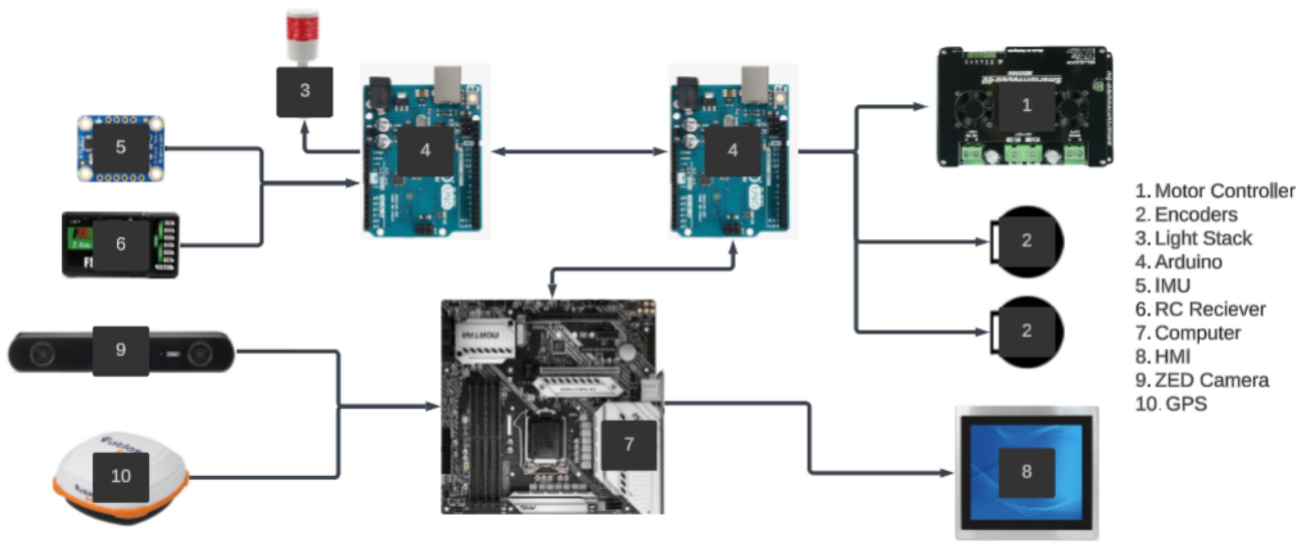


Figure 8. Simplified Data Schematic

5.2 Power Distribution

Component	Voltage (V)	Power Consumption (W)
Safety Light	12	0.5
HMI	12	8
Computer	12	70
GPS	12	4.9
Arduino (x2)	12	2
Motors (x2)	12	250
Encoders	5	1
IMU	5	0.25
ZED Camera	5	1.9

Table 3. Power Requirements of Electrical Components

Note: The power consumption values of each component were confirmed by measuring with either a multimeter or a hall sensor. These measurements were taken to ensure accuracy and were not solely based on the values listed on the data sheet.

Based on the values above, the drivetrain can run continuously for 63 minutes. Additionally, our computation system can run for six hours with the extra power it provides. We did this to reduce the number of times we turned the system on and off or had to replace the batteries.

5.3 Sensors

5.3.1 ZED Camera

This camera is functional, passed down by previous teams, and happened to meet our requirements. These requirements include object detection and point cloud map and can be powered via the computer or 5V. The camera uses two lenses to capture two frames, similar to human vision. It has a resolution of 1080p with a refresh rate of up to 100 fps.

5.3.2 Atlas Link GPS

The A326 AtlasLink Smart Antenna will be used for the waypoint navigation section of the course. The GPS has a location accuracy of 0.16 meters. In addition, the GPS has an IP67 waterproof rating, making it suitable for the outdoor environment. The 2022 team donated this GPS, and we decided to use it as it satisfies our need for GPS navigation and reduces the overall cost of the robot.

5.3.3 IMU

We selected the Adafruit BNO055 9-DOF IMU for the vehicle's compass navigation. Once again, this was passed down by previous years and functions correctly. It works with the GPS and Arduino to give the vehicle an accurate heading and direction.

5.4 Motor Controller

Selecting the proper motor controller for our system took the longest to research by far since we could not use previous teams' motor drivers. We decided to go with the SmartDriveDuo-60 MDDS60 motor controller since our other option, the RoboClaw 2x60A Motor Controller, was out of stock at the time of research. These two had similar, if not the exact specifications we were looking for, and all other motor controllers still needed to meet at least one of our requirements.

Important Specs	Motor Specs	Motor Controller
Peak Current (A)	94	100
Voltage	12	7-45
Input Modes	PWM, RC, Serial	RC, Analog, PWM, Simplified Serial, Packetized Serial
Motor Compatibility	Brushed DC	Brushed DC
Channels	Dual Channel	Dual Channel
Communication Compatibility	Arduino	Arduino

Table 4. Motor Controller Justification

As mentioned previously, our controller met our specifications perfectly, and as a plus, we have direct contact with the engineer responsible for its design.

5.5 Arduino

The Raspberry Pi would have been a more practical choice for our project because it is a microprocessor, not a microcontroller, and its clock speed and memory. However, our team had much more experience with Arduino. Our motor controller was only compatible with Arduino, so we decided to go with an Arduino Uno R3 and an Arduino Mega 2560 R3. Having two Arduinos that communicate with each other will help with the lower processing done by a single Arduino.

5.6 Computer

The robot runs on the onboard computer system mounted on our electrical plate. The computer has an Intel i5 10400F core, a GeForce GT 1030 graphics card, 16 Gb of RAM, and 500 Gb of memory. We use the Robotic Operating System (ROS) on Ubuntu Linux 20.04. Previous teams successfully used the exact specifications for their computer and only ran into a few issues. Our computer performed exceptionally well in rigorous testing, including simulated ROS courses, proving it is reliable and efficient.

5.7 HMI

We are reusing our Teguar TSD-45-12 Waterproof Monitor. It has a weatherproof rating of IP66/IP69K. This is a sunlight-readable touchscreen monitor with an operating voltage of 9-36V. This display will allow us to debug the robot effectively on the course instead of returning to the base.

5.8 Safety

5.8.1 Safety Light

Our safety light is red and mounted on the top of our robot, making it visible from 360 degrees around the robot. The safety light we purchased had a physical switch that changed it from solid to blinking. In order to get the lightstack to blink, an Arduino-compatible relay was added.

5.8.2 Mechanical E-stop

The mechanical E-stop is physically located on the vehicle's rear and easily accessible. It is wired between the normally-open relays and the motors themselves, and when the button is pressed, the connection to the motors is lost, and the robot will stop. All the other components on our robot are unaffected by this E-stop.

5.8.3 Wireless E-stop

The wireless E-stop will be implemented with a wireless toggle switch. This switch is placed in between the motor controllers and the motors to allow for a safe stop of the motors, even if the robot is not reachable.

5.8.4 Additional Safety

On top of the required safety implementation, we added fuse boxes and in-line fuses to our system to protect our hardware. We have several fuses of different ratings, adequately chosen for each piece of hardware. We have blown some fuses, which have protected our sensors and hardware. The only thing we damaged was our 12V to 5V step-down converter, which was not fused properly since it is directly connected to our 12V fuse box.

6. Software Design

6.1 Overview

The software for our robot was developed using the Robotic Operating System (ROS) on Ubuntu Linux 20.04. We used Python 3 in ROS and a C++-based language in the Arduino IDE. Our strategy involved implementing nodes in ROS for the robot’s primary functions, such as lane detection, object detection and avoidance, and GPS navigation. These nodes converge into a central navigation node that controls when each node is activated/deactivated.

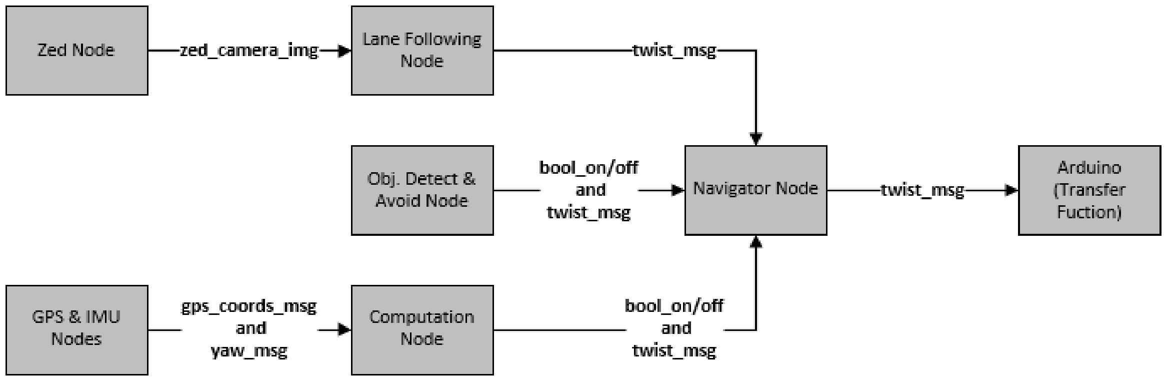


Figure 9. Software Architecture Diagram

6.2 Lane Detection & Navigation

The robot uses computer vision to detect and navigate the course's lane lines. It identifies the lane lines by implementing image processing techniques and algorithms. This section will summarize the image processing sequence we implemented.

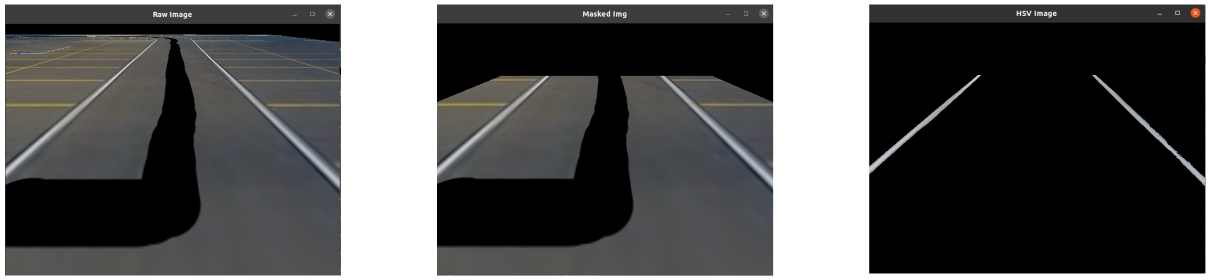


Figure 10. Left to Right: Raw Source Image, Masked Source Image, HSV Filtered Image

The three images in the figure above were captured during tests performed in a simulation environment. The leftmost image shows the raw source image captured by the ZED camera, and the middle image shows the source image after image masking is applied. Image masking allows the robot to focus on specific areas of interest to eliminate unnecessary noise within the source image. After masking, HSV filtering is applied to isolate the lane lines within the masked image, shown in the rightmost image.

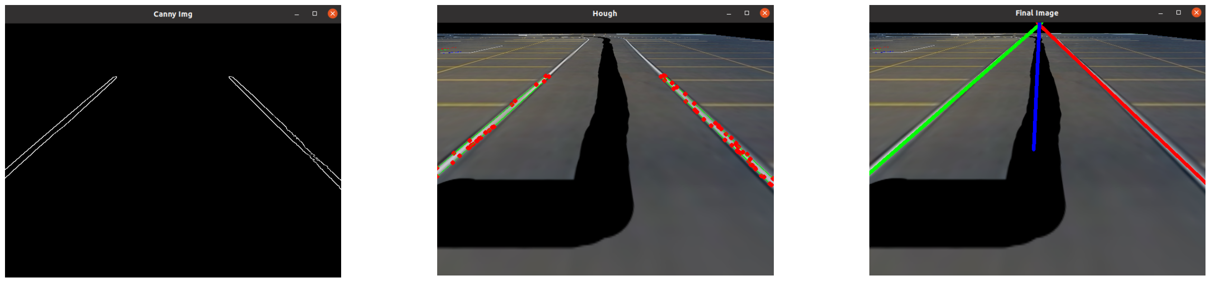
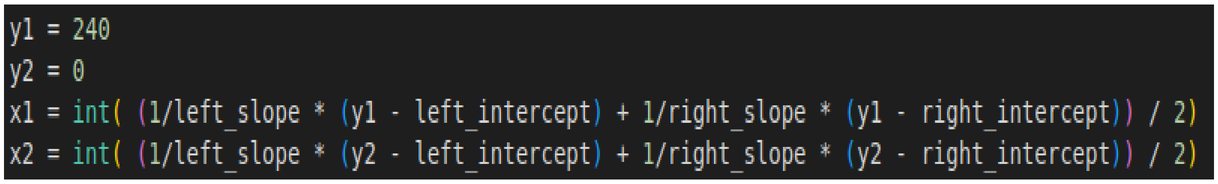


Figure 11. Left to Right: Canny Image, Hough Transform Image, Final Image

In Figure 11 above, the leftmost image illustrates the HSV-filtered image with Canny Edge Detection applied. This algorithm lets us detect the lane line edges by leveraging edge gradients and hysteresis thresholding to connect corresponding pixels. Subsequently, the Hough Line

Transform (HLT) algorithm fits lines through the detected edges. By applying HLT to the Canny image, we can identify the lines that represent the lane line edges. This algorithm operates in the polar coordinate space, where each pixel corresponds to a line in the image. The validity of each line is determined by assessing the number of points that fall on the pixel. Greater point density indicates higher confidence in identifying actual lines.

The rightmost image in Figure 11 represents the final image utilized by the robot for lane-based navigation. In order to obtain this image, the HLT algorithm's output is sorted into separate arrays, one for the left lane and one for the right lane, based on slope, length, and position. A single left and right lane line (represented by the image's green and red lines) is determined by averaging their slopes and y-intercepts. The centerline is then determined by using the slope-intercept equation and averaging x-values from the topmost and bottommost points of the left and right lines to find two points between them that can be used to draw a centerline. The robot uses the centerline to navigate by calculating the percent error between the actual image center and the topmost x-value ('x1' in Figure 12). The calculated error determines the robot's steering (left for negative error, right for positive error).



```
y1 = 240
y2 = 0
x1 = int( (1/left_slope * (y1 - left_intercept) + 1/right_slope * (y1 - right_intercept)) / 2)
x2 = int( (1/left_slope * (y2 - left_intercept) + 1/right_slope * (y2 - right_intercept)) / 2)
```

Figure 12. Code Block Demonstrating Finding a Centerline Via Averaging

6.3 Object Detection & Avoidance

The Object Avoidance node uses the ZED camera's built-in functionality to generate a point cloud map that is stored as the given frame. By analyzing the distance of each pixel from the camera's point cloud map, the node identifies the middle and closest objects' distances using an algorithm. To avoid noise from external objects, the node waits until the middle object is within 1.5m and disregards anything beyond 5m. Once the middle object is within range, the node publishes a flag and determines how the robot should avoid the object based on the closest obstacle. The node calculates the free space using the rule of at least 5ft clearance from an obstacle. If the closest object is on the right, the node sends a message telling the robot to turn left and vice versa. The node selects the direction with better clearance if the closest object is the middle one. If there is a processing error, the robot defaults to driving straight. The routes are set to end just past the last, closest object to prevent detection and enable quick discovery of lane centroids by the lane following node.

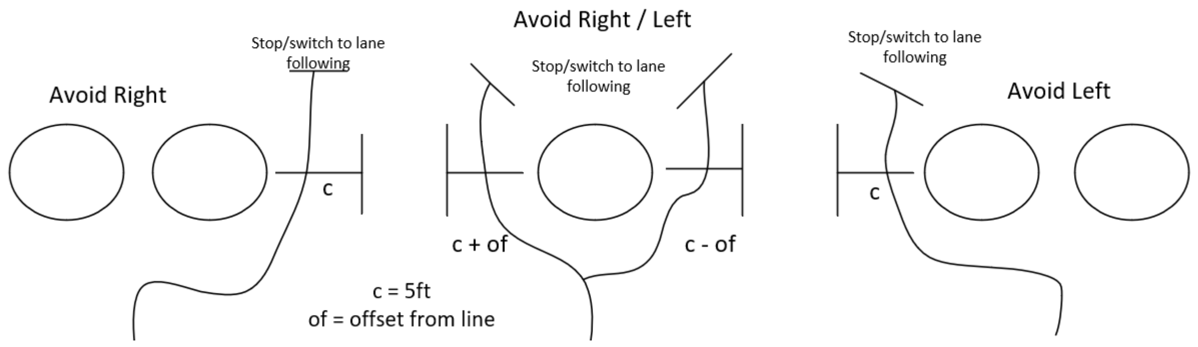

Avoid Right / Left
Stop/switch to lane following
Stop/switch to lane following
Stop/switch to lane following
Avoid Right
Avoid Left
c
c
c + of
c - of
c = 5ft
of = offset from line

Figure 13. Diagram Showing the Robot's Avoidance Routes for Different Object Placements.

6.4 GPS Navigation

Our ROS program will use the four GPS coordinates given before the competition to determine the direction and distance the robot needs to complete the waypoint navigation section of the course. Our GPS, the AtlasLink GNSS Smart Antenna, provides live GPS positioning of the robot with an accuracy of about 16 centimeters. We are using the ROS nmea_navsat_driver to parse the NMEA messages from the GPS and read the coordinates in ROS. The latitude and longitude of the robot's position are sent over the '/fix' topic in ROS. Relying solely on GPS coordinates to navigate through the course's waypoint navigation portion is insufficient. We need to know the robot's heading to move in the right direction. Therefore, we use an IMU (Inertial Measurement Unit) to determine its orientation and maintain accurate navigation. The 9-axis IMU gives us absolute orientation relative to the north, making the calculation easy to navigate between GPS waypoints. The IMU is connected to an Arduino UNO that sends the robot's heading (yaw angle) to ROS. The ROS code receives and publishes the angle over the '/yaw' topic.

With both the GPS and yaw messages now available, we created the computation node in ROS to combine the data from both sensors. The computation node first calculates the distance between the current position of the robot and the next waypoint coordinate; this is done using the haversine formula. Then it computes the bearing angle to go from the current point to the next point using trigonometry. The bearing angle is defined as the angle measured in the clockwise direction from the north line with the robot's current position as the segment's origin. Comparing the bearing angle to the current heading of the robot, the computation node will send the proper messages (twist message in ROS) to the navigator node.

6.5 Navigation Node

The navigator node is a listening state machine that determines which nodes should fire at specific times based on a set of if-else statements. The node has multiple subscribers and publishers to collect data from other nodes and send drive commands to the Arduino. The process of the navigator node begins with subscribing to the computation node and the object avoidance node. Both nodes publish flags to the navigator node only when they detect an obstacle, or the real-time GPS coordinates match the first waypoint. The computation node flag takes priority, indicating when the robot has reached No Man's Land. With the flag set to 'True', the computation node sends drive commands to the robot to move from the given waypoint to the next until it reaches waypoint 4. The object avoidance node takes priority when its flag is ‘True’, and it sends a driving route that the robot must complete before continuing with other nodes. If both flags are ‘False’, the navigator node publishes the twist command the lane following node sent as a default.

6.6 Kinematics & Motor Control

A differential drive robot was chosen to simplify the control. Using the velocities of the wheels, the forward velocity and angular velocity of the robot can be computed:

$$v = \frac{R}{2}(v_r + v_l) \, , \, \omega = \frac{R}{L}(v_r - v_l)$$

R= Wheel Radius, L = Distance Between Wheels

From these equations we obtain equations for the angular velocity of the motors given the forward and angular velocities:

$$\omega_r = \frac{2v+\omega L}{2R^2} \, , \, \omega_l = \frac{2v-\omega L}{2R^2}$$

Utilizing these equations, the desired angular velocities for each of the wheels is derived. The motors are connected to the MDDS60 motor controller. This controller can take a PWM signal as an input and drive the motors at a proportional speed. The MDDS60 receives a PWM from an Arduino dedicated to the control of the motors. The Arduino has a few main jobs:

1. Receiving the desired angular velocities from the central controller.
2. Use signals from the encoders to determine the angular velocity of each motor.
    a. Trigger an interrupt every time the ‘A’ channel rises. This counts the number of triggers.
    b. Determine the angular velocity using the number of triggers and the time step.
    c. Apply a low pass filter to the velocity data to reduce the high-frequency noise.
3. Apply a PID controller based on the error of the desired angular velocity minus the actual angular velocity.
4. Scaling the control effort in terms of PWM from 0-255.
5. Sending this PWM value to the motor controller.

The loop of the Arduino takes about 15 ms to complete on average. Thus, we implemented a 20 ms time step on the loop.

7. Performance

7.1 Ramp Climbing Ability

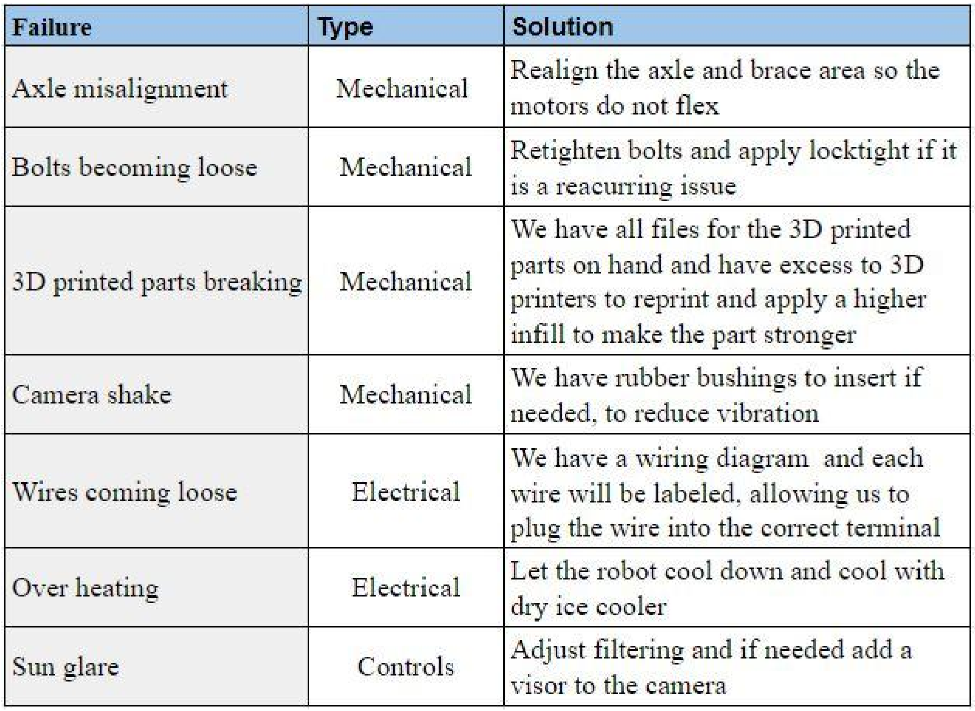
Our first task was ensuring our motors could produce the necessary torque to climb the ramp. Using prior knowledge from our physics classes, we decided to use formulas we were familiar with to split up the forces in the x and y direction for a rolling object going up a ramp. We were able to calculate that we needed 10.1 Nm of torque, with an estimated robot weight of 150 lbs and a wheel radius of 6.25 inches to complete the task. Our motors’ stall torque is 19.88 Nm which meets our safety factor of 1.5 that we initially had in mind going into this project. In addition to this, we tested our robot on an incline of 20 degrees and were successfully able to climb it.

7.2 Speed

The motors we chose to go with are high-torque motors which are more efficient at lower speeds. These were ideal since the speed limit is five mph on the course. The robot's speed has been tested and it can move at least double the required speed.

8. Failure Points

Table 5 shows the failures that can occur and our solutions to these problems. We have encountered these problems while testing, or problems that we anticipate happening during competition.



Failure	Type	Solution
Axle misalignment	Mechanical	Realign the axle and brace area so the motors do not flex
Bolts becoming loose	Mechanical	Retighten bolts and apply locktight if it is a reacurring issue
3D printed parts breaking	Mechanical	We have all files for the 3D printed parts on hand and have excess to 3D printers to reprint and apply a higher infill to make the part stronger
Camera shake	Mechanical	We have rubber bushings to insert if needed, to reduce vibration
Wires coming loose	Electrical	We have a wiring diagram and each wire will be labeled, allowing us to plug the wire into the correct terminal
Over heating	Electrical	Let the robot cool down and cool with dry ice cooler
Sun glare	Controls	Adjust filtering and if needed add a visor to the camera

Table 5. Failure Points

9. Conclusion

Our team's goal of successfully designing and building an autonomous ground vehicle capable of navigating and completing the IGVC course is close to being achieved. Through multiple iterations of testing and optimization, we have gained valuable knowledge and experience in project management, teamwork, and an opportunity to apply our skills to a real-world challenge. We are confident and proud that our robot *'Blue'* will get us the best results that show our dedication and hard work.

References

"The 30th Annual Intelligent Ground Vehicle Competition." *Official Competition Rules*, www.igvc.org/rules.htm. Accessed 25 Mar. 2023.