

Marvin



Rocker Robotics Team President

Josiah Huntington | Josiah.Huntington@mines.sdsmt.edu

Rocker Robotics Team Advisor

Rohan Loveland | rohan.loveland@sdsmt.edu

Members

Autonomy & Controls	Mechanical
Landon Lamoreaux † - Senior, CSC Bennet Outland - Senior/Graduate, ME/CSC/CSE Josiah Huntington - Senior, CENG Michael Klingelhofer - Junior, CSC Devon Hamm - Freshman, CENG Aden Muzzey - Freshman, CSC	Jacob Decker † - Sophomore, ME Kaylee Herndon - Junior, ME Heath Buer - Senior, ME/EE Steven Duong - Junior, ME Derek Matthies - Sophomore, ME Ethan Miller - Sophomore, ME Sam Ryckman - Graduate, ME/CSC/CSE* Zane Wilson - Sophomore, ME Daron Graf - Freshman, ME
Electrical	
Alexis Englund † - Sophomore, CENG Chase Reinertson - Senior, EE Sierra Rodewald - Junior, EE	

† Team Lead

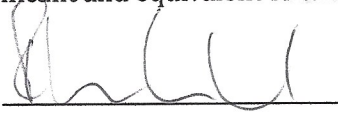
CENG: B.S. Computer Engineering | CSC: B.S. Computer Science | CSE*: M.S. Computer Science and Engineering | EE: B.S. Electrical Engineering | EE*: M.S. Electrical Engineering | ME: B.S. Mechanical Engineering

List of member email addresses: landon.lamoreaux@mines.sdsmt.edu, bennet.outland@mines.sdsmt.edu,, alexis.englund@mines.sdsmt.edu, josiah.huntington@mines.sdsmt.edu, michael.klingelhofer@mines.sdsmt.edu, devon.hamm@mines.sdsmt.edu, aden.muzzey@mines.sdsmt.edu, jacob.decker@mines.sdsmt.edu, kaylee.herndon@mines.sdsmt.edu, heath.buer@mines.sdsmt.edu, steven.duong@mines.sdsmt.edu, derek.matthies@mines.sdsmt.edu, ethan.miller@mines.sdsmt.edu, samuel.ryckman@mines.sdsmt.edu, zane.wilson@mines.sdsmt.edu, daron.graf@mines.sdsmt.edu, chase.reinertson@mines.sdsmt.edu, sierra.rodewald@mines.sdsmt.edu.

Rocker Robotics – Marvin

I, Rohan Loveland, certify that the design and engineering of this vehicle by the current student team have been significant and equivalent to what might be awarded credit in a senior design course.

Signature: _____



Date: _____

5/15/2024

Contents

Team Introduction	1
Summary	1
Acknowledgements	1
Design Assumptions and Process	2
Mechanical	2
Autonomy and Controls	2
Electrical	2
Innovations	3
Mechanical	3
Overview	3
Frame	3
Steering	4
Outer Shell	5
Electrical	6
Computers	6
Sensors	7
Motors	7
Power Connections	7
Autonomy and Controls	8
Paradigm	8
LIDAR	8
Obstacle Detection & Map Generation	9
State Estimation	9
Path Planning	10
Robot Control	11
Simulation	12
Potential Failure Points	12
Predicted Performance	13
Cost Breakdown	14
References	15

Team Introduction

Rocker Robotics is an extracurricular robotics team from the South Dakota School of Mines and Technology. The goal of the team is to teach robotics concepts to its members, bring robotics and STEM outreach to our community, and compete in a robotics competition each year. Its members have a range of robotics experience, majors, and backgrounds. Josiah Huntington is the president, Landon Lameroux is the autonomy and controls team lead, Jacob Decker is the mechanical team lead, and Alexis Englund is the electrical team lead.

Summary

Our robot is named Marvin. It uses a limited-swerve drive system. Across all members of the team, we spent approximately 5600 person-hours on this project. We are using computer vision to identify the white-lane lines and potholes. LIDAR and ultrasonic sensors are used to identify 3D barriers in our path, and a Global Positioning System (GPS) to estimate our location on the course. We create a map of obstacles that we see in the world and then use the A* (A-star) path planning algorithm to create a path that avoids all these obstacles.

The processing is split between an Nvidia Jetson Orin Nano and a Raspberry Pi Pico microcontroller. LIDAR, computer vision, and path planning are all handled by the Jetson, while the simpler sensors are connected to the Raspberry Pi Pico via a custom printed circuit board (PCB). The E-stop is an isolated board that controls the state of a relay that can interrupt the power connection to our motor controllers. This relay can be actuated either by the E-stop button on the robot or the button on either remote using a wireless connection. The robot is capable of four-wheel drive using four NEO brushless DC motors and two brushed motors for steering. A standard Ackerman steering is used, the same format found in most cars. With the added functionality of the left and right front wheels able to turn independently of each other. The design also included the development of a frame to support the steering mechanism and the outer shell of the robot.

Acknowledgments

We would like to thank Dr. Rohan Loveland, our advisor, for all his help throughout this year. We would also like to thank the South Dakota School of Mines and Technology and the Electrical Engineering, Computer Engineering, and Computer Science (EECS) Department for providing us with funding and lab space in which to design, build, and test our robot. Additionally, we thank the team's umbrella organization, South Dakota Mines CAMP, for helping us with leadership training and team administration. For assisting in design reviews, we would like to thank Dr. Rohan Loveland, Dr. Larry Simonson, Dr. Thomas Montoya, Dr. Jason Ash, Dr. Ryan Koontz, Ms. Amy DiReinzo, Mr. Lowell Kolb, and Mr. Pratik Sinai Kunkolienker.

Design Assumptions and Process

Design assumptions for this robot began with its size. The robot must fit within the constraints set by the competition rules [1]. Past this, we targeted a few major design features: the robot should be car-like, have a turning radius significantly tighter than the tightest turn described in the rules, be relatively robust, and not be so tightly integrated that it's difficult to build and maintain. The team placed a large emphasis on working around materials already available in their lab. Notable parts of the robot that are in line with this include the LIDAR, most of the frame, and the wheels and tires.

Mechanical

At the beginning of the school year, our decision-making was guided by decision matrices. After working through multiple group discussions based on these tools, we settled on the final architecture for the robot: a rigid chassis with a central pivot, limited-swerve steering, independently driven wheels, and a body in the style of a small van. We started with an existing chassis for testing and later designed a final chassis using aluminum T-slot extrusion and waterjetted aluminum plates.

Autonomy and Controls

As for sensors, computers, and software stacks, much of the decision-making was based on what the members of the Autonomy and Controls subteam were already comfortable with. This led to us using an Nvidia Jetson Nano running ROS2 [2] on Ubuntu Linux, and the Arduino framework on our Raspberry Pi Pico microcontroller. Sensor choices were guided by considering what a typical autonomous car might have, leading us to use a combination of a LIDAR, stereo cameras, a GPS, and an IMU.

Electrical

The primary goal for the electrical team this year was to expand on the boards completed in the previous year. We spent time this semester improving on the designs and redesigning the boards. Most of the electronics in the robot have been converted to printed circuit boards (PCB) utilizing SMD/THT components on our boards.

Innovations

As a team, we aimed to innovate in the areas of computer science, electrical engineering, and mechanical engineering. Some of the software innovations presented in this project include sensor processing methods and a novel waypointing method. We created a novel algorithm for detecting lane lines and converting them into an obstacle point cloud in three dimensions. The line-obstacle data is then used to generate a valid driving region for the robot. From this point, a ray is cast through the objects, and the optimal target waypoint is determined after filtering. Both of these methods will be further explained in the Autonomy and Controls section of the report.

In terms of electrical engineering, we designed a printed circuit board to handle the interface between sensors and our computers. Additionally, we developed our E-stop board and software, so we can safely stop the robot from a distance in accordance with the competition rules. The E-stop board allows us to disconnect the motors from the battery power while always keeping power to the other electronics to avoid potential filesystem corruption if our computers were to lose power. The E-stop board is electrically isolated from the power of the rest of the robot. It uses 915 MHz radio modules that allow for two-way communication with multiple remotes to allow for multiple ways to turn the robot off if needed.

On the mechanical side, the steering architecture chosen is a four-wheel drive system, using a restricted swerve module drive for steering. The main benefit of this drive system is that the robot can execute tight turns at a low complexity. The steering and drive systems are attached to an aluminum frame split by a center pivot point, which helps to support all mechanical and electrical components used within this design and maintain ground contact for all four wheels. To provide both water resistance to electronic components and further our robot's car-like appearance, a wood shell was created to rest on top of the chassis.

Mechanical

Overview

For this project the mechanical subteam was responsible for the prototyping, designing, and manufacturing of the chassis and related components; including several subsystems such as the frame, drivetrain, and outer shell. The key idea we applied to our decisions was simplicity; we created a list of the features we wanted to improve from our previous design and brainstormed ways to achieve these features without adding unnecessary complications to our designs. The desire for simplicity became a driving factor in all of our decision-making processes.

Frame

The key improvement we made to the frame was the addition of a center pivot point. A failure point in our design last year was a warp in our frame which lifted a wheel off the ground, propagating a further chain of component strength failures. To avoid that, we decided to build the frame around a center pivot point which would ensure all four wheels would maintain ground contact. As seen in Figure (1) below, the 1.5" diameter steel rod acts

as our pivot axis. The front end of the robot with the steering elements affixes the rod, while the back end has an inset pair of bearings that smooths the rotation. An important note to mention at this point is that all of the components in this section have been designed with a considerable factor of safety to ensure that this section does not become another failure point. Building from that, the rest of the backbone of the frame consists of a spine of 3" x 1 1/2" T-slot with a 1 1/2" x 1 1/2" aluminum T-slot mounted perpendicular to the spine at each end of this center beam as can be seen in Figure (1). This creates a proper support structure for all electronic components and the payload and adds attachment points for drivetrain components. Aluminum plates and brackets are used throughout the frame to strengthen and increase rigidity in the chassis, as these create cross braces within the frame. These components are also used to aid in mounting the drivetrain systems and shell. The T-slot material was chosen as it is strong and there was an abundance readily accessible to our team. An interesting challenge to this iteration of the robot is its weight. Since these parts were designed to be excessively strong, they are also fairly heavy. We have, however, increased the torque of our driving motors (see below) and changed the materials of the components that failed last year to compensate for a heavier frame.

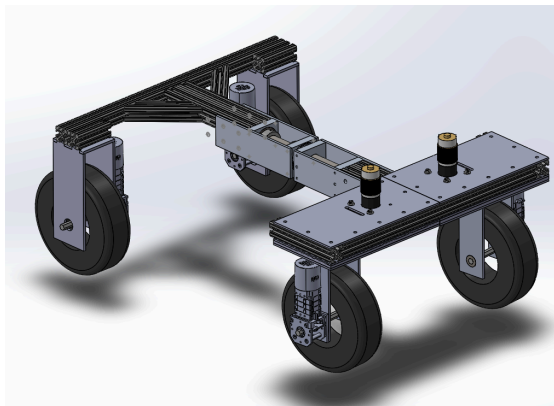


Figure 1: CAD Model of the chassis

Steering

The most unique part of our robot is the steering system. Overall, we chose a setup very similar to a modified Ackermann drivetrain involving a pair of steered wheels. During the process of choosing a drivetrain style, we worked through multiple options looking for the best mix of benefits and costs. In addition, we considered what we had learned from last year with an inefficient chain-based design. The process led us back to using a system that mimics Ackerman-style steering but uses a pair of independently driven, restricted swerve modules. These modules use a steering motor to turn a bracket that holds the wheel and its driving motor. Though these could technically act as swerve modules, the driving motor acts as our limiting factor as its wires would get twisted if it were to make a full rotation. Instead, we chose to limit the rotation to nearly 180 degrees, which provides a short turning radius without the complexity of a full swerve module. We have both a physical limit and a software-based limit to ensure these do not over-rotate.

We also decided that to halve the time it took to get the robot fully assembled and running we would only equip the front wheels with turning capabilities. These fixed-back wheels have minimal chances of failing, providing us with at least some direct propulsion in a worst-case scenario. Though this removes the possibility of a zero-point turn, it simplified our drivetrain and reduced the parts costs, and assembly complexity drastically without sacrificing much of our turning radius. As an aside, this pattern also gives us a design that mimics how full-size vehicles are driven, which we felt aligned with our goal of having a car-like appearance.

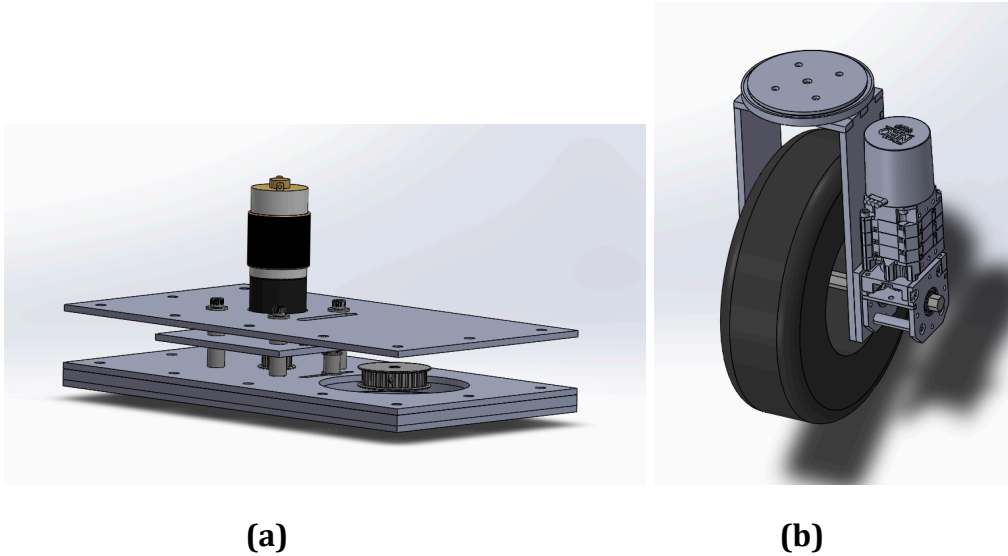


Figure 2: The two main steering assembly parts.

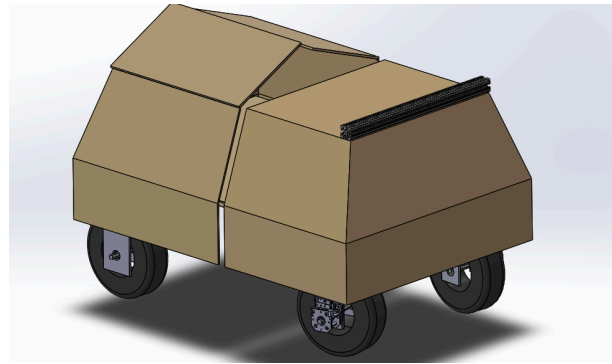
(a) Belt Assembly. **(b)** Driving Assembly

Outer Shell

The shell serves a variety of functions, including mounting points for various sensors, and providing aesthetic appeal. The design of the shell is based on the Mystery Machine from the iconic Scooby-Doo series, which was chosen due to the relative simplicity of the shell shape. The manufacturing method of the shell includes utilizing both laser cutting and 3D printing. The outer panels were made in separate wood pieces, while the brackets that hold them are made using Polylactic Acid (PLA). There is also a small frame made using an extruded aluminum t-slot, which helps to support the weight of the sensors that are placed on top. The shell also provides some weather resistance, this is aided by a commercially sourced electronics box, which provides further water resistance.



(a)



(b)

Figure 3: (a) Example of the Mystery Machine that the design was based on. (b) “Shell” of the robot, along with other shell components.

Electrical

Computers

The intensive processing is primarily done on a Nvidia Jetson Orin Nano including image processing and path planning calculations. It also deals with LIDAR and camera data taking in and sending commands to the Logic Board which houses a Raspberry Pi Pico as the on-board computer. The Logic Board handles processing for less intensive systems, such as the Inertial Measurement Unit (IMU) and GPS to off load from the Orin. As well as providing broken out debugger pins and plenty of 5V GPIO pins for any needed use. This board also contains an integrated controller area network (CAN) module to connect to our motor controllers. The Orin is powered from a 19V power supply and the Logic Board is powered from a 5V power supply, both supplies are connected directly to the battery.

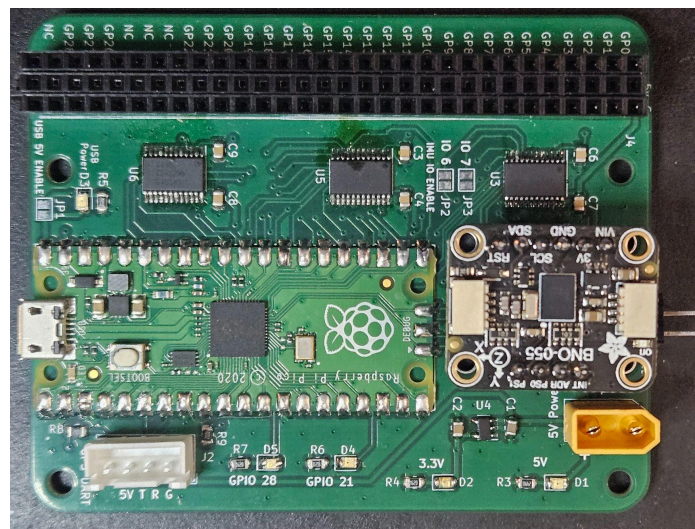


Figure 4: Logic Board

Sensors

Primary object detection is performed by a Hokuyo UTM-30LX LIDAR sensor. This unit has a field of view of 270 degrees with a detection range of 0.1m to 30m. Line and pothole detection is performed by a pair of Intel RealSense Depth Camera D435s. These will be used to identify and locate the lines and potholes of the track. Additional location information will be obtained from an Adafruit Ultimate GPS. A BNO055 inertial measurement unit (IMU) will be used to determine the orientation of the vehicle. Ultrasonic sensors are also placed at the front of the robot to serve as emergency obstacle avoidance sensors in case the robot gets too close to an object that was missed by the LIDAR.

Motors

We are using 4 Neo brushless DC motors as the main drive system. These motors were chosen in part because they are very powerful for their price. Marvin's steering is achieved using two brushed DC motors. There is a Spark Max motor controller for each of the motors. The Spark Max motor controllers have built-in voltage regulation for the motors, as well as compatibility with CAN for easy communication. These motor controllers are also rated for continuous currents of up to 60A, which is a much higher current than we expect to see.

Power Connections

Our robot uses a 4-cell, 20Ah LiPo battery to supply approximately 14.8V power to the motor controllers. The connection between this battery and the motor controllers is set up so that it can be interrupted by a relay for the emergency stop system, as detailed in the E-Stop subsection. A second smaller 4-cell LiPo battery will provide power to the sensors and electronics. This battery will have direct connections to our LIDAR, beacons, 19V power supply, and motor controllers. The beacon state is controlled by the E-Stop to account for the current state of the robot. We have also designed a custom led light driver board to handle a more visual representation of the robot state. The colors will change depending on the state the robot is currently in with each color code depending on the input from various other boards. We are using an off-the-shelf DC Buck Boost Module to set the voltage from the battery for the Nvidia Jetson. The Logic Board routes power to our remaining sensors and provides communication connections to the electronics not directly connected to our more powerful computers.

E-Stop

As required by the rules for this competition, we have designed an emergency stop system for the robot. This system is an isolated circuit from the rest of the electronics. It is based on a dual-purpose custom PCB that, depending on configuration, can be used either as a transmitter or receiver. There is a solder jumper on the back to dictate the board configuration. The receiver configuration will be located on the robot itself and will be controlling the state of a relay that connects our motors and motor controllers to the battery. This battery connection is separate from the battery connection to our main computers and sensors, allowing us to kill power to all of the drive systems while keeping power to the electronics that would be more sensitive to sudden power loss as data is being transferred. The receiver board will open the relay contacts if either the E-stop button

onboard the robot is pressed, or it receives a command from the remote E-stop. The remote E-stop board uses the same PCB in the transmit configuration, which will communicate with the receiver board using LoRa RFM69HCW packet radio modules. Pressing the E-stop button on the remote sends a status update to the receiver board which will then open the relay contacts the same way that the E-stop button on the robot does. Both the transmitter and receiver are also constantly sending keep alive pings to blink a light on both boards. This ensures that the boards are communicating and gives a visual to double check communications are still alive.

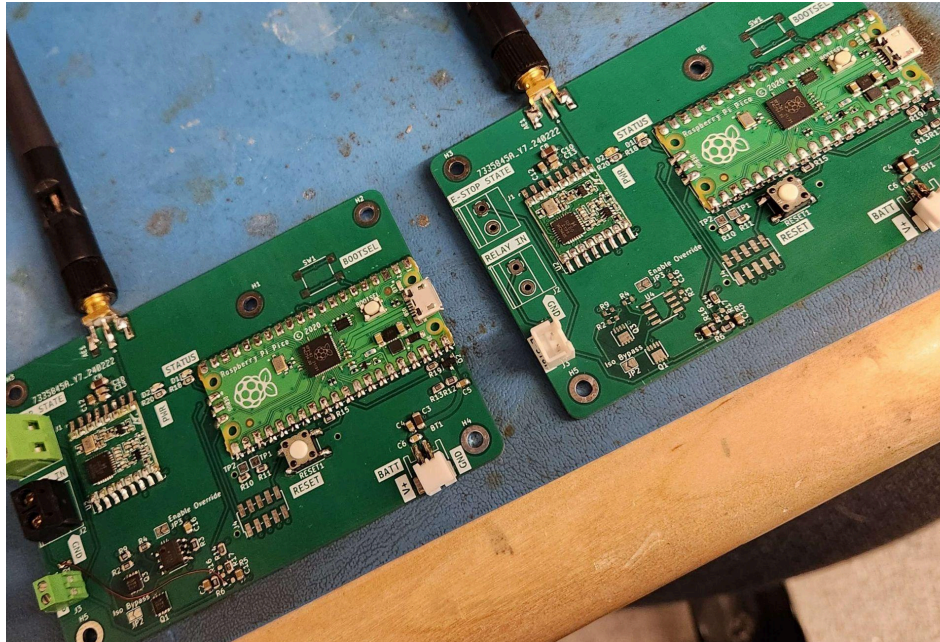


Figure 5: E-Stop Boards

Autonomy and Controls

Paradigm

This year, our robot's autonomy and control uses the same concepts as last year, but with further testing and reworking to ensure robustness. It is based on the concept of treating all course obstacles (line, barrels, potholes, etc) identically. This allows for a simple, robust, and general framework for navigating through the course. In cases where there are not any lines, the obstacles will still be considered while the GPS coordinates act as waypoints to guide the robot. This allows for standardization of the outputs from all obstacle-related sensors. All of the sensor interfacing and communication will be done through ROS2 [2].

LIDAR

Marvin uses a 2D LIDAR unit to detect the location of obstacles in relation to the robot. It is positioned to be high enough to avoid detecting the ramp, but low enough to detect any of the course barrels. It then sends these obstacle locations to the map generation algorithm as described below.

Obstacle Detection & Map Generation

The robot uses two Intel Realsense D435 cameras for lane following and pothole avoidance. These cameras provide both an RGB stream and a depth stream. Both streams are read in and aligned to each other. Figure 3 shows the algorithm used to convert the color image into a point cloud that can be used by the path planning algorithm. The color image is converted into a grayscale image and then adaptive thresholding is used to find where any white lines are by looking at each pixel and its neighbors to see if they are significantly different, if they are, it is assumed to be a white line, and is kept in the image. The image is then eroded to remove any noise and the top half is masked out to reduce the sample complexity since the ground is where the obstacles are. The resulting image contains lane lines and any potholes. Combining this with the depth image allows us to create a point cloud of all the points that need to be avoided. This algorithm is repeated for the second camera. The two-point clouds are then transformed to be in the global coordinate frame. These point clouds are combined with the point cloud from the LIDAR to create one point cloud of all the obstacles. All these points are mapped to an image to create a picture of everything around the robot. The path planning algorithm uses this point cloud to navigate around the course.

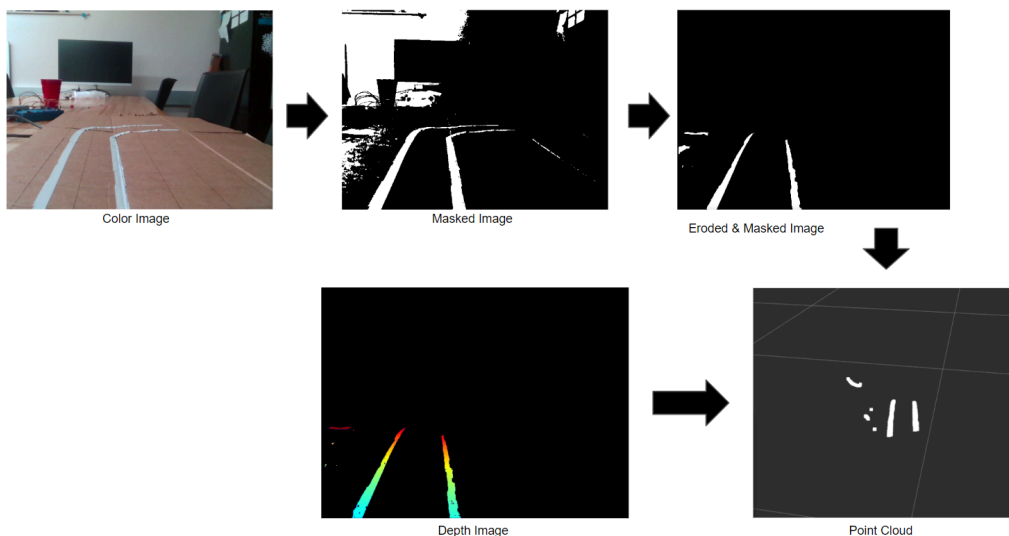


Figure 6: This is a visualization of each of the image manipulation stages discussed. The depth image here only contains the points from the eroded masked image.

State Estimation

An important aspect of many of these sub-algorithms is knowing the location of the robot. To accomplish this we are using an Extended Kalman Filter (EKF) for state sensor fusion. An EKF is derived from the optimal, Gaussian, linear state filter, a Kalman filter, where the linear filter is used to approximate nonlinear behavior for small timesteps. Given estimations of the state and the expected error of the sensors used, a robust approximation of the actual state can be determined.

To determine the robot's location a combination of a GPS and a 9-axis IMU are used. We are using a Bosch Sensortec BNO055 IMU, which nicely calculates an absolute orientation estimate. With the data from these sensors, we can get accurate estimates of the actual position and heading of the robot.

Path Planning

Using our state estimation, we can determine where the obstacles are in relation to the robot. We then start determining what path the robot will need to take to traverse the course. To determine the optimal path, three different elements must be determined. First, the sub-state must be found that describes where the robot can and cannot be. Second, the target waypoint that the robot travels towards is set. Lastly, the path to move between the current robot state and the target waypoint is determined. Each of these can be considered individually.

To determine where the robot is allowed to drive, we look at the local state and find all obstacles that are within a set radius from the robot. This is done so we may focus only on the obstacles that will have a relevant impact on the direction of the robot. From this point, we take the local state that has been created and compress it to a smaller state represented by a $n \times n$ gridspace to save on computational expense. With this gridspace defined, we have a loose collection of obstacles spread throughout as seen in Figure (5a). Issues can arise with the invariability of sensor noise, resulting in obstacles not being detected or spuriously detected. To counter this, we can apply a dilation kernel to fill in any gaps where obstacles should have been detected. If there were any small areas of spuriously identified obstacles, an erosion kernel can remove the extraneous obstacles. Ultimately, we are then left with a cleaned region with the dilated obstacles being returned to their original size. The results of applying the two kernels can be seen in Figure (5b). Under the assumption that the robot is currently between the lines, we apply a flood fill algorithm to define the region between the lines where the robot is allowed to traverse. This can be seen in Figure (5c). This was accomplished using the OpenCV library [3].

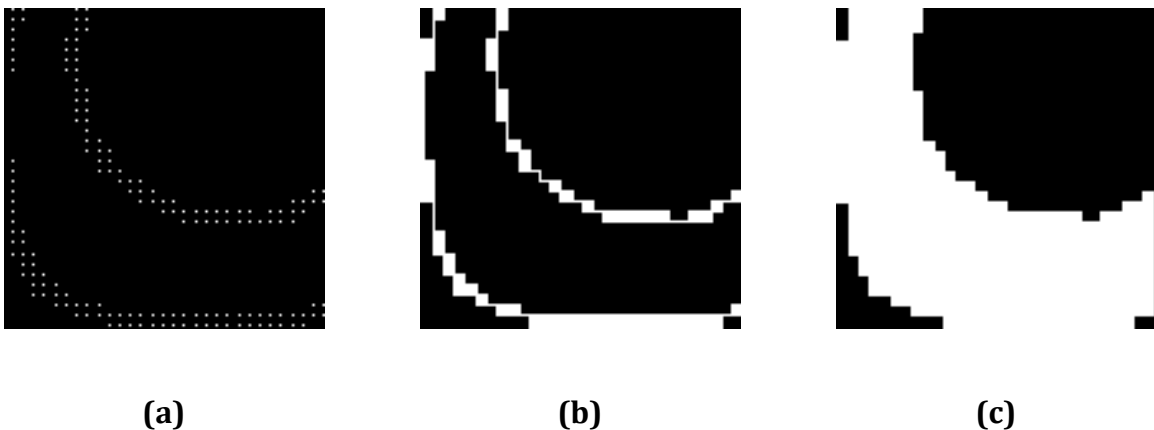


Figure 7: Different steps in the processing of the algorithm to determine the driveable region. **(a)** Raw obstacle data from sensors. **(b)** Obstacle representation after dilation and erosion. **(c)** Drivable region determined post-floodfill.

Given that a traversable region is defined, we can start locating a waypoint for the robot to navigate towards. To accomplish this, we use a “T-beam” with a fixed-width web and flange where the size of the flange is much greater than the size of the web. This is done through the following:

- Let F be the set of points from some $(-l, 0)$ to $(l, 0)$ varying the x component
- Collect the current orientation of the robot, θ
- Rotate flange by $\theta + \frac{\pi}{2}$
- Shift the central point in the flange to the robot through additive broadcasting
- Calculate the location of the end of the web using the given distance and the current orientation of the robot
- Shift the central point in F to the tip of the web through additive broadcasting

From this point, we want to determine a target point that is not near obstacles and would not result in needing to take sharp turns unnecessarily. To do this, we apply a Gaussian filter to the grid elements along the flange to smoothen the discontinuous steps between obstacle and non-obstacle regions. This results in a set of hills and valleys where the hills represent the favorable drivable regions since they are devoid of obstacles while adding a buffering region near the obstacles. To mitigate sharp turning, a bandpass-like filter is applied that increases the favorability of smaller turns, and outside the central region is subjected to a decrementing linear trend. A linear search algorithm is then used to determine the maximum along the now-filtered flange elements. Through this process, we can extract a better target point by taking into account the local obstacles and the turning angle.

With a target point defined, we can define the starting position to be the currently estimated state of the robot. To plan a path between the starting and target waypoint, we use A^* due to its superior performance in terms of its percent optimal performance per calculation time [4]. We create the grid space so that the driveable areas have minimal weights to move between nodes while non-traversable areas have high node weights. We extract a path from the A^* algorithm created for the smaller $n \times n$ grid space. The path is now transformed from the compressed gridspace to the local region. While this does cause some discretization error the error is negligible due to the small regions considered versus the size of the compressed grid. As we step through the path, we can determine the steering angles needed.

Robot Control

The control of the robot can be separated into three main categories: steering, desired velocity, and stopping. To steer the robot, a target is read from the path planning algorithm. Steering motors are then turned and the real angle is read by potentiometers connected to the front wheels. A PID loop is then set to maintain the desired angle. The desired velocity is likewise read in from the path planning algorithm according to the density of objects in the local region. It is then read back by encoders on all the drive motors and each is set according to an Ackerman style kinematic model. We also explored different methodologies for developing a remote E-stop. We are currently using an RFM69_HCW and Raspberry Pi Pico for sending and receiving E-stop commands.

Simulation

We used both physical and software simulation environments to validate our algorithms. Specifically, in the cases of the GPS, orientation sensors, and depth cameras, physical simulation environments were created for real-world validation. In the case of path planning and state estimation, we used an in-house simulator that was developed to be lightweight and versatile. A sample output can be seen in Figure (6). Through both of these environments, we were able to test and validate the algorithms created for this project.

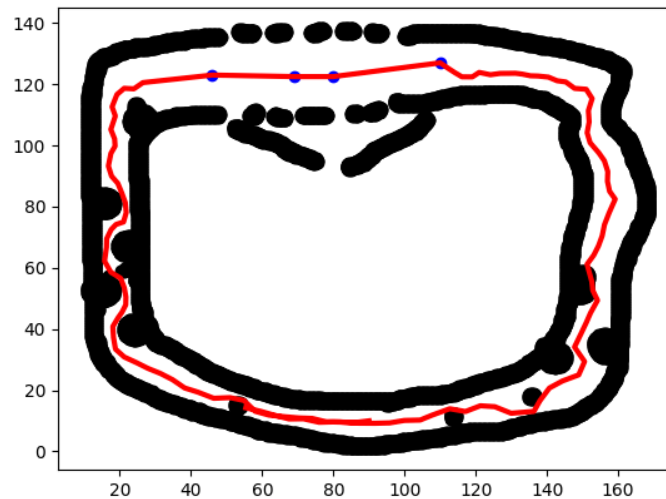


Figure 8: Example figure from the simulator used demonstrating the robot consistently traversing the course by the placement of internal waypoints. These waypoints have been connected to demonstrate the path, but the A* generated path will ultimately be used. Note that the blue dots are simulated GPS waypoints.

Potential Failure Points

There is a potential for failure points in the autonomy, electrical, and mechanical systems. In terms of autonomy, the computer vision system always has the possibility of identifying the lines improperly. Changing lighting conditions throughout the run and shadows over the line could cause our robot to not see some of the lines. Autotuning to the lighting conditions has been implemented, but rapid changes may cause a momentary loss of obstacle data from the camera.

In the electrical system, the main two points of failure are vibrations and water entrance. Due to the lack of a suspension and the slightly bumpy terrain, frequent vibrations are expected. Vibration of the wires over a long period may slowly disconnect various connectors. To combat this, electrical connections are all either soldered in place or attached by snug connectors. The robot is not designed to be waterproof, but rather simply water resistant. An excessive amount of water on the robot may cause some water damage

and might cause shorts. This has been mitigated, but cannot be guaranteed to be perfect. Should either of these cases occur, spare components and connections will be brought along to the competition. Tape may also be used to provide additional temporary sealing for problem areas. One of the other areas for concern is that radio frequency (RF) components tend to be very sensitive. To combat potential for powering our RF circuitry without antennas we designed them to utilize SMA connectors so antennas are more securely connected. These antennas will not easily come loose from vibrations or other outside factors. They however could still be a failure point if someone were to take the antenna off, then power on the radio. Further, one of our other main intentions is to keep board costs low and as the specific risk is low we have not included reverse polarity protection on our boards in the form of protection diodes. We instead utilized XT30 connectors which are directional and will help prevent reverse polarity on our Logic Board. Our E-Stop boards are fitted with a battery connector that is also directionally connected with an interlocking JST connector. Though, this does not stop someone from accidentally entering the battery into the E-Stop incorrectly via the contact strips. We have included symbols which are directional to help combat this.

In the mechanical subsystem, there are a few points of failure that could occur. One possible failure point for the chassis is at the wheels and axles. As the entire weight of the robot rests on the two components, there is a chance that either of them could break if the robot is overloaded, or the shafts for the wheel assemblies could bend. To work around this, we used aluminum and 3D-printed parts to help reduce the overall weight of the chassis and distributed weight thoughtfully throughout the chassis, allowing each wheel to receive less of the load. Repeated sharp steering angles will increase the stress through the steering mechanism which may cause failure if used for long periods. We resolve this issue through our control of the system by disincentivizing sharp steering angles when the path that the robot takes is planned. Another concern is vibrations in the robot loosening bolts, to prevent this a thread locking compound and lock nuts were used to keep bolts tight and in place along with personal checking components between runs. Design reviews were also conducted with advice from faculty to ensure a robust design in the more critical sections of the robot.

Predicted Performance

We predict that we will be able to perform better than we did in the previous year. As this is the second iteration of this robot, we have improved several of the components. However, there are still several areas that we may be able to improve on from this iteration of the robot. Our robot is predicted to be able to identify obstacles reasonably well and be able to create a path that avoids all of the identified obstacles. In simulation, the robot has been able to successfully traverse the course. Based on worst-case power consumption, we expect the robot's battery life to be at least two hours. Based on how far down the drive motors are, this robot should be able to climb the ramp with no issues. Lane lines will be detected five to ten feet ahead of the robot. Obstacles detected by the LIDAR will be seen at 20 to 30 feet away from the robot.

Cost Breakdown

Product Name	Unit Cost	Quantity Used
Hokuyo UTM-30LX LIDAR*	\$4,675	1
Intel Realsense D435*	\$314	2
Nvidia Jetson Orin Nano	\$500	1
Turnigy 20000mAh Battery*	\$142	1
Rev Robotics Spark Max*	\$90	6
Grantury Junction Box*	\$63.49	1
6061 Aluminum U Channel - 3ft	\$42.64	1
Rev Robotics Ultra 90° Gearbox	\$42	4
McMaster-Carr Metal Gear Rack	\$41.61	2
Raspberry Pi Pico*	\$4	4
Rev Robotics NEO DC motor	\$48	4
MGN12H 300mm Linear Rail	\$28	2
1kg 3D Printed Material	\$25	4
Rev Robotics 5mm Hex Shaft, 400mm length	\$21.50	2
3 JSN-SR04T Ultrasonic Distance Sensor*	\$20	2
QCQIANG E-Stop Button*	\$18.29	1
McMaster-Carr Metal Gear 20° angle	\$13.68	2
Rev Robotics UltraPlanetary Cartridge 4:1	\$11.50	12
10" Pneumatic Tire and Wheel*	\$8.49	4
Rev Robotics 5mm Hex Bearing Block	\$4	4
1.5" x 3" T-Slot Extrusion - Cost is Per Inch*	\$1.02	33
1.5" x 1.5" T-Slot Extrusion - Cost is Per Inch*	\$0.57	96
2' x 2' x ¼" Wooden Panels	\$8.99	12
Total Cost: \$7307.22		

* Denotes items the team had on hand and did not need to purchase

References

- [1] IGVC. (2022, October). The 30th Annual Intelligent Ground Vehicle Competition (IGVC) Self-Drive.
- [2] Stanford Artificial Intelligence Laboratory et al. (2018). *Robotic Operating System*. Retrieved from <https://www.ros.org>
- [3] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.
- [4] Fareh, R., Baziyad, M., Rahman, M., Rabie, T., & Bettayeb, M. (2020). Investigating Reduced Path Planning Strategy for Differential Wheeled Mobile Robot. *Robotica*, 38(2), 235-255. doi:10.1017/S0263574719000572