

University:

North Dakota State University (NDSU)

Team:

Autonomous Bison

Date Submitted:

5/15/2019

Team Lead:

Andrew Jones andrew.jones.4@ndsu.edu

Team Members:

Rodi Murad	rodi.murad@ndsu.edu
Avijeet Tomer	avijeet.tomer@ndsu.edu
Daniel Anderson	daniel.anderson.3@ndsu.edu
Ali Rahim Taleqani	ali.rahimtaleqani@ndsu.edu
Qianqian Yao	qianqian.yao@ndsu.edu

Faculty Advisor:

Dr. Jeremy Straub



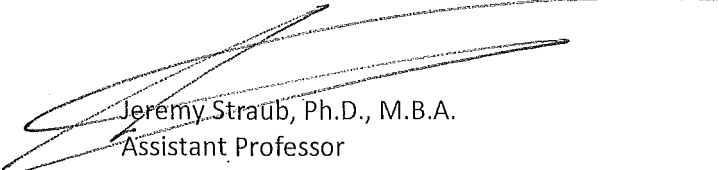
NDSU NORTH DAKOTA STATE UNIVERSITY

5/14/2019

To Whom It May Concern:

I've been asked to provide a statement regarding the originality of the NDSU Autonomous Bison team's vehicle design. The team has undertaken significant work, from both a design and development perspective to create this vehicle. The design, while again based on and recycling a chassis from a wheel chair is unique and designed by the team members. The recycled chassis presented an opportunity for the team as well as a number of challenges that they had to adapt to and was an ecologically friendly way of robot creation. The team integrated this chassis and a variety of hardware and software components and developed new original software as part of their solution. This work, in my opinion, meets or exceeds the requirements of most senior design courses at most colleges and universities. Please don't hesitate to contact me if I can provide any additional information or clarification regarding the foregoing.

Sincerely,



Jeremy Straub, Ph.D., M.B.A.
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE

NDSU Dept 2740 | PO Box 6050 | Fargo ND 58108-6050 | 701.231.8562 | Fax 701.231.8255 | <http://cs.ndsu.edu>

NDSU is an EO/AA university.

2. Design Process and Team Organization

2.1 Introduction

The Autonomous Bison team participating in this year's Intelligent Ground Vehicle Competition is comprised of students from North Dakota State University (NDSU). This marks the second year of participation in the IGVC for many team members, and the first year of participation for others. The design process has been a great learning experience for the team, and we are excited to participate in this year's competition.

2.2 Team Organization

Andrew Jones	Team Lead/Software/Electrical
Rodi Murad	Software
Avijeet Tomer	Software
Daniel Anderson	Software/Mechanical/Electrical
Ali Rahim Taleqani	Software
Qianqian Yao	Software/Mechanical

2.3 Design Process

The main goal for the design process was to completely overhaul our software system from the previous year. With the majority of our team members having a computer science background, this focus on software seemed to be the best objective. The software is written in Python, and interfaces with ROS. It is further described in Section 6.

On the mechanical side, the goal was to improve the wheelchair frame from the previous year. The electronics mounting configuration was the focus of the redesign. In particular, the decision to replace the laptop (which was previously used) with a Nvidia AGX Xavier module allowed for a more contained design (which improved weather resistance). The mechanical structure is further described in Section 4.

3. Innovations

One of the design decisions is to use the Nvidia AGX Xavier, which is a new product designed with autonomous vehicles in mind. It has a compact form factor, and has relatively low power consumption (~30 watt). The power consumption in particular gives it a distinct advantage over a laptop, which may draw over 60 watts. The Xavier has a 512-core Volta GPU, which is compatible with Cuda and the ZED camera.

Another innovation, garnered from what other teams have done, is the use of the ZED stereo camera. The ZED camera's capabilities (depth perception, positional tracking, and RGBA color values) allow us to have a single point for generating data that can be used for both obstacle detection and odometry.

Lastly, our overall software redesign aims to minimize the usage of the Robot Operating System (ROS) due to the performance drop when using larger data types such as point clouds. The software itself is mostly composed of custom code, written in Python. Although, ROS is used in the program, and the ROS Launch API being natively written in python allows launch files to be started directly from the code.

4. Mechanical Design

4.1 Overview

The base chassis used for our vehicle is a donated Jazzy Select 6 power chair. This electric wheelchair model met the vehicle dimension, speed, and maneuverability requirements for the competition. The following sections describe the frame structure, suspension, and weather proofing design details.

4.2 Frame Structure

Two 12-volt AGM batteries are housed in a battery enclosure towards the rear of the vehicle. This enclosure was part of the original wheelchair design and was only slightly modified to accommodate the component mounting layers above this area. The electronics are mounted on a layer directly above the battery enclosure. For testing purposes, a laptop can be mounted above the electronics layer.

The payload will be secured on the wheelchair footrest by a ratchet strap. This placement location helps offset the weight of the batteries and achieve a lower center of gravity. Furthermore, rubber shock absorbers are utilized to help reduce shock load.

The Zed camera is situated on a stainless-steel tube mast, roughly 4.5 feet from the ground. This height offset is intended to provide a field of view above the common traffic barrel obstacle height.

4.3 Suspension

The drive train consists of two front caster wheels, two mid drive wheels, and two rear caster wheels. The frame has Independent left and right rockers for the front and middle wheels. The rear wheels are laterally connected. The front rockers are particularly advantageous for transitioning to and from ramp inclines, as it helps the mid drive wheels maintain a level of traction. No spring devices are used due to relatively low speeds and the existing design of the wheelchair. Furthermore, the lack of springs reduces complexity and modes of failure.



4.4 Weatherproofing

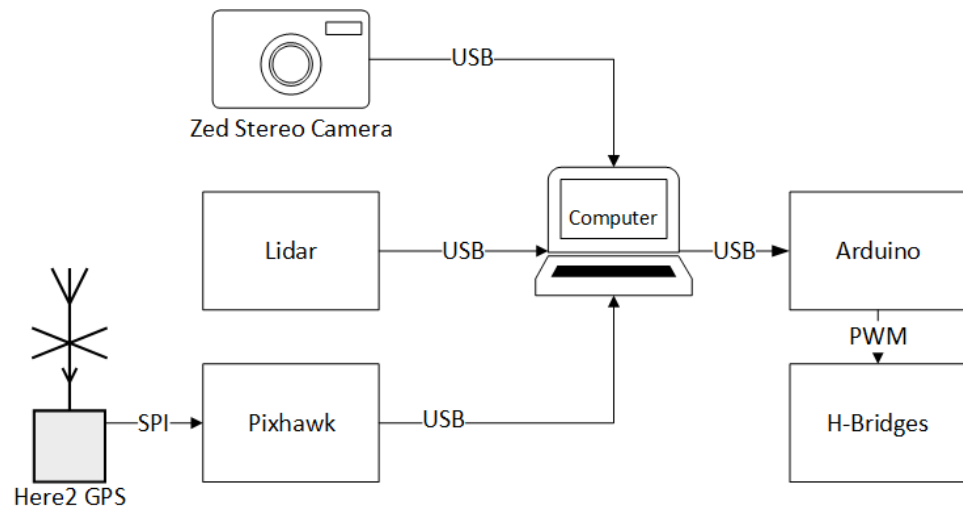
The electronics are housed in an ABS weather resistant case, with gaskets used for external wire routing. Weatherproof housings for the sensors are currently being designed. The battery enclosure has holes in the bottom so that it can drain any water that enters due to light rain or puddles.

5. Electrical Design

5.1 Overview

The electrical system design of the vehicle can be categorized into two parts: the power distribution system, and the electronics suite. The power distribution system includes the batteries, motors, relays, and power circuit design. The electronics suite includes the control computer, ZED stereo camera, LIDAR, GPS, and Arduino microcontrollers. The following sections describe the sensors and power distribution in more detail.

5.2 Electronics suite

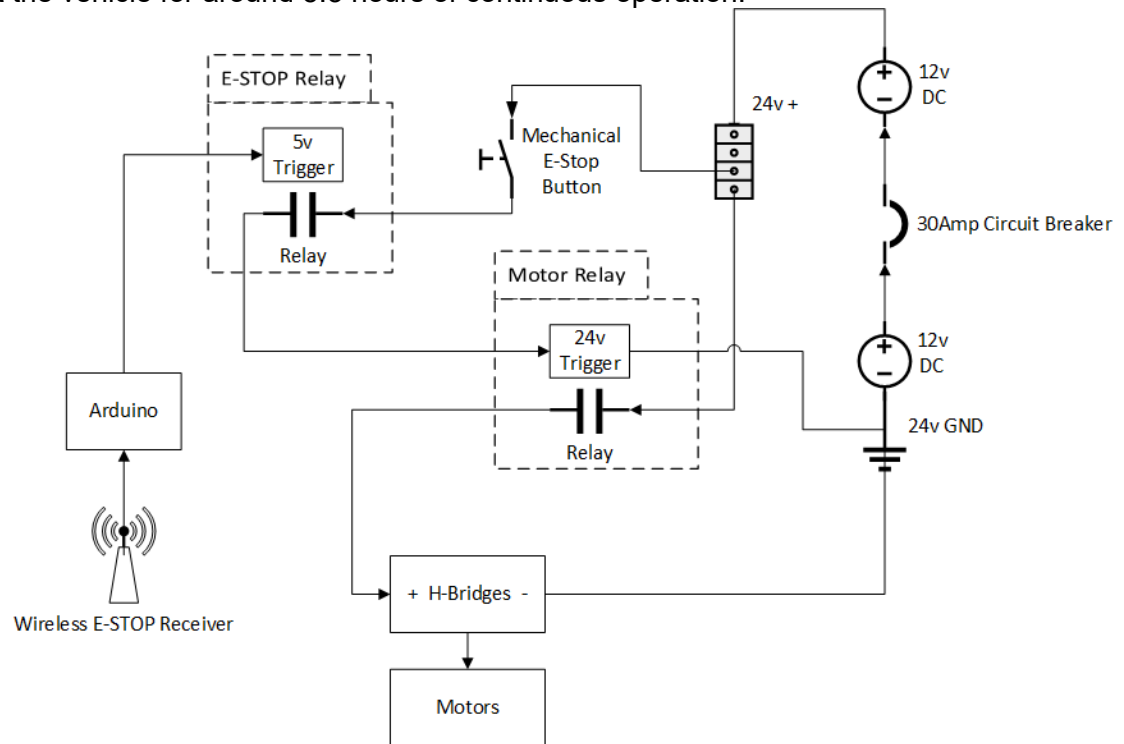


- **Computer:** The computer is a Nvidia Jetson AGX Xavier, which has a small form factor and meets the computational requirements. It has a 512-core Volta GPU, which is compatible with the ZED camera. Its power consumption is around 30 watts and it has a 19v input (which can be stepped down from the 24v vehicle power source). For testing purposes, the Xavier can be disconnected and the sensors can be rerouted to a top mounted laptop.
- **ZED Camera:** A stereo camera is used which provides RGB images, depth images, 3D point clouds and visual odometry. It has compatibility with outdoor environments and an effective depth sensing range of 20 meters.
- **H-Bridges:** The type of motor controller used for our vehicle is the CyTron MD30C, which accommodates the 24v that the motors use. In addition, these H-Bridges can handle well above the typical current draw of the vehicle.
- **LIDAR:** The LIDAR used on the vehicle is a YDLIDAR G4, which has a maximum range of 16 meters.

- **Pixhawk:** A pixhawk v2.4.8 is utilized for its gyroscope and accelerometer. It is also used to interface with the GPS module.
- **GPS Module:** The GPS system consists of a Here2 GPS Module, which uses a ublox NE0-M8N.

5.3 Power Distribution

The two batteries have a 35AH capacity and are wired in series. The motors draw between 3A to 5A (total) depending on speed, acceleration and other factors. The Xavier draws approximately 1.5A (including peripherals). Thus, the upper bound on current draw would be approximately 6.5A. Therefore, the batteries (at full charge) would be able to support the vehicle for around 5.3 hours of continuous operation.



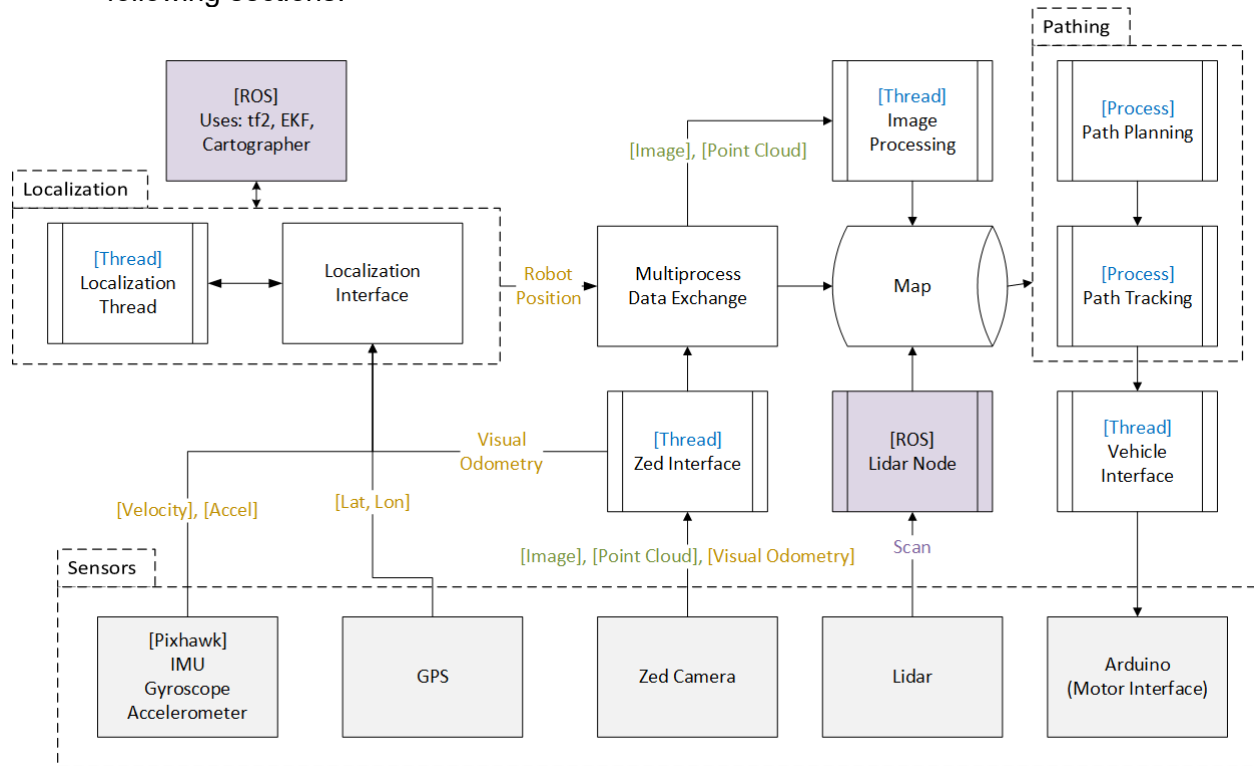
5.4 Safety Devices

- **Mechanical E-Stop:** A latching pushbutton is connected to the relays to insure the safety of the system. Once it is pressed, it will open the motor relay and cut off power to the vehicle's motors.
- **Wireless E-Stop:** A wireless receiver is used in conjunction with an Arduino to provide a remote turn-off mechanism for the motors. As with the Mechanical E-Stop, the Wireless E-Stop opens the motor relay in order to cut the power. Due to the motor relay being 24v, the Arduino uses its own 5v relay in order to trigger the motor relay.
- **Circuit Breaker:** A 30-amp circuit breaker is in place between the two battery connections. This is used in case of a short circuit and to prevent the motors from attempting to draw more power than some of the components can handle.

6. Software Design

6.1 Overview

The software system consists of six custom Python modules which run in separate threads or processes. Certain modules utilize and communicate with ROS nodes using rospy. The modules are Localization, Zed interface, Image processing, Path Planning, Path Tracking, and Vehicle Interface. The different modules are described in detail in the following sections.



6.2 White Line Detection

For image detection we perform multiple iterations of the 2-D Convolution and Gaussian blurring, dilation, edge detection, and followed by more dilation for increase of while line area.

We have attempted five different image smoothing methods (2-D Convolution, Averaging, Gaussian, Median, and Bilateral) for noise removal. For each method we plotted the image histograms to understand the methods ability for noise removal and pixel convergence. The Gaussian method had the best results not only reducing singleton high peaks but performed well blending pixel values across the color range 0 – 255. However, the Gaussian method alone could not result in the degree of noise removal desired. Therefore, in addition to the Gaussian we used the 2-D convolution method and the Median at a smaller scale by reducing the matrix size used for the computation of the average.

Following the image smoothing methods used, we implemented the Morphological Transformation methods which include Erosion, Dilation, Opening, Closing, and Gradient. Then, the Binary Thresholding method is applied, which resulted in the removal (conversion to 0) of all pixel values below a set value (which in our case was 127). The

application of this method resulted in a maximum removal of the grassy area without much loss of the white line.

Finally, we applied the Canny Edge detection method which merely converted the image retrieved from the Binary Thresholding to lines that delineated the edges of the remained pixels. This ensured to remove all color from the image, for us only to detected edges comprised of white pixels. A Dilation method from the Morphological Transformation is applied to increase the thickness of the white line as well as edges of obstacles that were scattered throughout the course for easier detection. Given that, some locations resulted in an undetectable white line or it was detectable to a minimum degree upon which the attempt to remove noise, removed some of the white line. It was important to note that the application of the white line must be consistent, especially on a grassy field where various noise is introduced due to varying lighting conditions as well as varying shades of the grass.

6.3 Physical Obstacle Detection

The physical obstacles present on the course are intended to be detected by the lidar scans. The lidar used on the vehicle is the YDLIDAR G4, which has an effective range of 16 meters. It is positioned in the front portion of the vehicle and is capable of 360-degree scans, although only the forward facing 270 degrees is mapped (certain portions of the vehicle may be pinged in the backward 90 degrees). The scans from the lidar are compiled by a ROS node (designed by the lidar manufacturer) and published on a ROS topic. The topic is subscribed through rospy and used for map generation.

6.4 Map Generation

The map format chosen was that of a numpy array. These can be converted to (and from) ROS messages with relative ease using rospy and have built-in vectorized operations which can increase the speed of bulk write operations to the array. Furthermore, the array is encapsulated in a custom class which handles converting points to a specified distance unit resolution (i.e. half a meter per cell). The currently utilized map dimensions are 2000 by 2000, with a $1/10^{\text{th}}$ of a meter cell resolution. Due to the size, a small memory type is used for each array element, in this case a uint8 was chosen.

Overall, there are three separate maps that are generated by our autonomous ground vehicle program. These maps include a lidar map, white line map, and combined map. The lidar map contains only lidar scan obstacle detections, and the white line map contains only white line detections. These are kept separate in order to accommodate saving the maps separately and to avoid determining which sensor takes precedence over the other (in the case of overwriting/updating observations). The combined map is used for path planning and path tracking.

The lidar map is constructed through transforming the scan messages to the map frame using the ROS tf2 library. Similarly, the white line map is constructed through selecting the points in the point cloud corresponding to the image pixels identified by the white line detection algorithm. The selected points are transformed to the map frame before being placed into the map.

6.5 Path Planning

The path planning algorithm utilized is D*. This reduces computation time compared to A*, as it doesn't need to recalculate map sections that remained the same since the previous iteration. Furthermore, the heuristic for the entire map is calculated using vectorized numpy operations – which (based on testing) calculates the entire

heuristic for a 2000 by 2000 grid in approximately $1/100^{\text{th}}$ of a second. Other path planning algorithms were tested, such as potential field; however, these methods were prone to getting stuck in local minima (depending on the obstacle map layout).

It was chosen to keep the planning process encapsulated as much as possible and run it in its own process. Using a ROS node would achieve this; however, the utilized map is shared across all processes through the use of “sharedctypes”, and this wasn’t supported by ROS-based pathing systems. Thus, the code implementing the D* algorithm was written in Python and is not tethered to ROS.

After the path is generated, it is transferred to the path tracking module – which utilizes the Pure Pursuit algorithm. The path tracking module also runs in its own process and communicates with the path planning module through a Synchronization manager.

6.6 Localization and Goal Selection

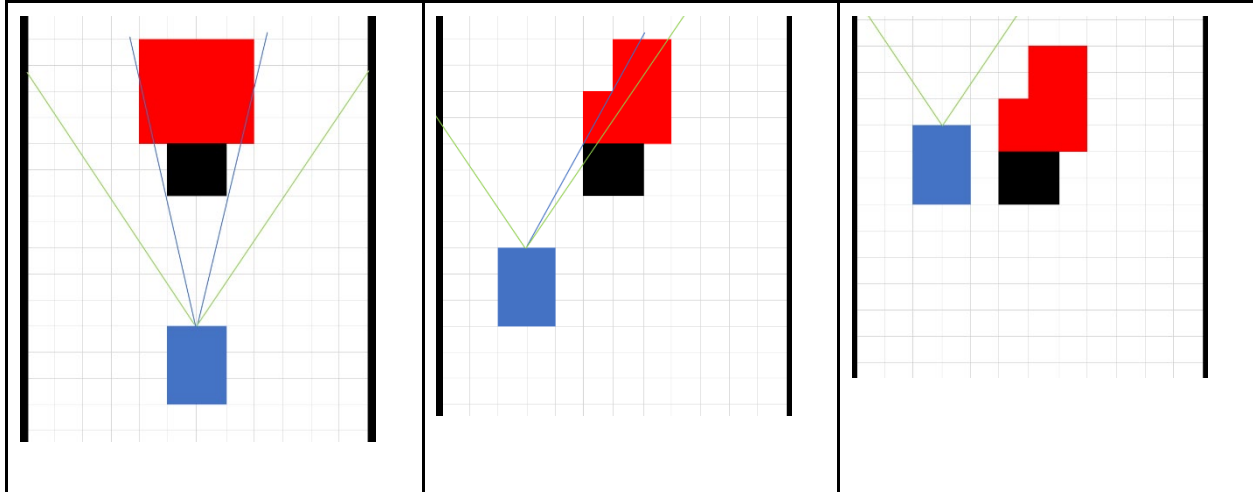
The localization of the vehicle consists of data from a variety of sensors. First, the Zed Stereo Camera provides visual odometry. The use of ROS-based SLAM is also available, which is done using the scan messages from the lidar. A Pixhawk is used to provide IMU, gyroscope, and accelerometer data (although it isn’t used to drive the motors). Last, a GPS module provides latitude and longitude data. Data from all the localization sources is input into an extended Kalman filter (EKF), which fuses the sources and calculates the vehicle’s position. The calculated position is relayed to the other modules via a Synchronization manager.

The goal/waypoint is selected based on previously provided GPS coordinates. The coordinates are queued and targeted in first-in first-out order. Functionality has been added to set immediate goals, for test purposes, but isn’t used/needed for the actual competition events. Upon arrival at the current goal, the path planner is then notified of where the next goal is, based on the queue.

7. Failure modes, failure points and resolutions

7.1 Vehicle failure modes and resolutions

1. **Color detection failure:** The color values returned by the camera for each pixel are primarily used to detect boundary lines and simulated potholes. The zed SDK allows enabling auto white balance to account for dynamic sunlight conditions while detecting white lines and potholes. Each frame is also scanned for white lines and potholes by OpenCV functions as a color detection failure backup to ensure that the same set of pixels is considered sufficiently for both white lines and potholes by the camera’s pixel color values and OpenCV functions.
2. **Mapping failure:** In case of sufficiently high physical obstacles in the center of the track, when the robot selects one of the two paths around it, the map generated contains a set of pixels behind the obstacle that are inaccurately represented as an obstacle due to the camera’s area of viewport. We ignore it because it is not useful for the robot’s path towards its goal of a given GPS waypoint. The figure below demonstrates the inaccuracy:



The blue squares represent the robot, black squares are the obstacle, green lines show the limit of the viewport and blue lines show the boundary of the line of sight obstructed by the obstacle. Since the camera cannot see behind the obstacle, the area in red is marked as an obstacle too in the generated 2d map. As the robot selects the gap on the left and changes its position from figure 1 to 2, the map is updated such that red area reduces. As the robot passes the obstacle in figure 3, some of that red area is maintained as an obstacle in the generated map because of the limit of camera's front facing viewport.

7.2 Vehicle failure points and resolutions

- **Short Circuit:** A circuit breaker is in place to cut power to the system in the event of an electrical short. Resolving this issue would involve finding and correcting the short circuit and reenabling the circuit breaker.
- **Structural fatigue of 3D printed parts:** While the 3D printed parts on the vehicle are made to have a certain level of durability, they may wear over time or break from an impact. Replacement 3D printed components are easy to print, and a cache of spare components are available to swap out any broken components.
- **Arduino Communication:** To not overload the Arduino with too much processing, we send it smaller amounts of data compared to last year. The sent data is used to determine the speeds of each motor, and is therefore extremely important to the functionality of the vehicle. The Arduino code is setup in a manner that resets if critical failure occurs, such that the vehicle experiences only a downtime of a second instead of complete mission failure.

7.3 Testing

- **Simulation:** Running the program locally using a ".svo" file recorded by the zed camera creates a map of obstacles from that file, allowing us to view the path created by the robot as it navigates between the obstacles on the map.
- **Real world:** The robot is tested on NDSU grounds on grassy terrain and a road with construction barrels available to be used as obstacles. Since the main computer is a laptop mounted on it, it is easy to make code changes directly to it in case of unexpected behavior.

7.5 Vehicle safety design concepts

- **Mechanical E-Stop:** A latching pushbutton is connected to the relays to insure the safety of the system. Once it is pressed, it will open the motor relay and cut off power to the vehicle's motors.
- **Wireless E-Stop:** A wireless receiver is used in conjunction with an Arduino to provide a remote turn-off mechanism for the motors. As with the Mechanical E-Stop, the Wireless E-Stop opens the motor relay in order to cut the power. Due to the motor relay being 24v, the Arduino uses its own 5v relay in order to trigger the motor relay.
- **Circuit Breaker:** A 30-amp circuit breaker is in place between the two battery connections. This is used in case of a short circuit and to prevent the motors from attempting to draw more power than some of the components can handle.

8. Simulations Employed

8.1 Simulations in virtual environment

The ZED camera software can record and save in “.svo” format, which contains not only the video but also 3d point clouds for each frame for depth map generation and camera's position relative to the starting point for movement tracking. We recorded “.svo” files in an outdoor environment setup similar to the IGVC course and ran our program against them to virtually simulate our robot's behavior.

8.2 Theoretical concepts in simulations

1. **Pros:** The simulation using .svo files was identical to actually using the camera for obstacle detection, map generation and path planning.
 - a. The spatial location values in the 3D point cloud in every frame of the .svo file allowed us to code for detecting boundary line. The point cloud also allowed for generating a map using the expected distance from the camera to the ground in any given frame vs the actual distance.
 - b. The location information in each frame of the .svo file allowed us to create the path taken by the robot on the generated map.
2. **Cons:**
 - a. The simulation was not useful in determining the correct values for driving the motors.
 - b. The simulation did not account for the shaking of the camera causing problems in map generation while it rode on the robot.

9. Performance Testing to Date

- **Speed:** The vehicle is able to maintain and adjust speeds within the 1-5 mph range. It is capable of greater speeds; however, there are h-bridge PWM caps in place (software side) to prevent exceeding 5 mph.
- **Ramp climbing ability:** The vehicle is able to traverse inclines with slopes exceeding 15-degrees.
- **Reaction time:** The vehicle's path planning reaction time is approximately 0.25 seconds, allowing it to re-plan paths quickly when new obstacles are detected.
- **Battery life:** Approximately 5.3 hours of continuous use.

- **Distance at which obstacles are detected:** 20 meters maximum for white line detection, and 16 meters maximum for lidar-based physical obstacle detection.
- **Accuracy of arrival at navigation waypoints:** Waypoint accuracy is within approximately 1 meter, although this varies based on weather conditions and other factors.
- **E-Stop:** The two E-Stop mechanisms have been tested and are functioning properly.

10. Cost Breakdown

Item	Cost
Nvidia AGX Xavier	\$900
Zed Camera	\$450
CyTron MD30C H-Bridges	\$70
YDLIDAR G4	\$350
Here2 GPS Module	\$90
Relays	\$20
Batteries	\$150
Pixhawk	\$60
Wheels	\$200
Wheelchair Frame & Motors	Donated
Total	\$2,290.00