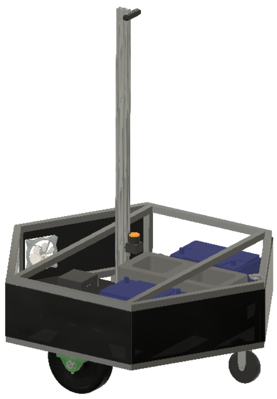
RUTGERS

Pathfinder



Members[†]

Mechanical		Electronics		Software	
Agam Modasiya	ajm432	**Ronan John**	rtj37	**Jack Lowry**	jrl253
Joshua Chung	jyc70	Nidhish Sharma	ns1300	**Karun Kanda**	kk951
Saurabh Bansal	sb1770			Gunjan Adya	ga402
Jeremiah Hou	jth155			Shreya Pandey	sp2072
Andy Yang	awy140			Jinam Modasiya	jjm601
Zacharry Soriano	zss28			Daniel Jin	hj301

I hereby certify that the design and engineering of the vehicle (original or changes) by the current student team has been significant and equivalent to what might be awarded credit in a senior design course.

Sincerely,

[illegible]

Chung-Tse Michael Wu, Ph.D

May 16, 2022

[†]Team Leads are in bold, and emails are id@scarletmail.rutgers.edu

0 Table of Contents

1 Introduction

1.1 Team Introduction

Rutgers IEEE is a large club at Rutgers University that contains many individual divisions that focus on many different subjects. We represent the robotics division in which IGVC one of the two competitions we decided to participate in for the 2021-22 season. We decided to participate in this competition to bring an opportunity for many of our robotics members to dive into more advance robotics and learn robotics techniques that are closely affiliated with industry level practices (i.e: CADing, simulation testing with Gazebo, ROS, and much more). Rutgers University competed in IGVC previously in 2011 and 2012, and we hope that bringing back the program will fill a niche in robotics opportunities at the university that was previously unfilled.

1.2 Organization

The Rutgers IEEE team consists of three teams corresponding to the disciplines of IGVC: mechanical, electronics and software. Each team has to one or two team leads who are in charge of organizing tasks for the meetings and making sure everything is ready before the deadline. Every member of each team always had a task to complete and the team overall put in many hours working on Pathfinder. The table below shows the role of each member in each of the teams.

Team Organization			
Member Name	Position	Major	Year
Agam Modasiya	Mechanical Lead	Mechanical Engineering	2022
Joshua Chung		Computer Science	2023
Saurabh Bansal	Mechanical	Computer Engineering	2022
Jeremiah Hou		Mechanical Engineering	2024
Andy Yang		Mechanical Engineering	2022
Zacharry Soriano		Mechanical Engineering	2024
Ronan John	Electronics Lead	Electrical Engineering	2023
Nidhish Sharma	Electronics	Physics	2025
Jack Lowry	Software Lead	Computer Engineering/Computer Science	2023
Karun Kanda		Computer Science	2022
Gunjan Adya	Software	Computer Engineering	2024
Shreya Pandey		Computer Engineering	2025
Jinam Modasiya		Electrical Engineering	2024
Daniel Jin		Computer Science/Math/Linguistics	2023

1.3 Design Methodology

The design process for this year started with a lot of research of prior teams. Being a new team, we had to build our knowledge base from the ground up. Rutgers University had competed in IGVC in 2011 and 2012, so we had their old robot as a base to go on, but besides that we spent a lot of time compiling the products and methods other teams have used to build their robot in the past. From here, each subteam broke out to begin designing models or schematics for their respective system, and began to buy parts to start prototyping them.

The mechanical sub-team wanted Pathfinder to be as rigid, robust, simple and durable for the stresses it would endure throughout its course and the load it would need to carry throughout the duration. With this in mind the sub-team needed to follow a simple process of brainstorming, researching, designing, testing, iterating, and repeating until we were happy with its performance.

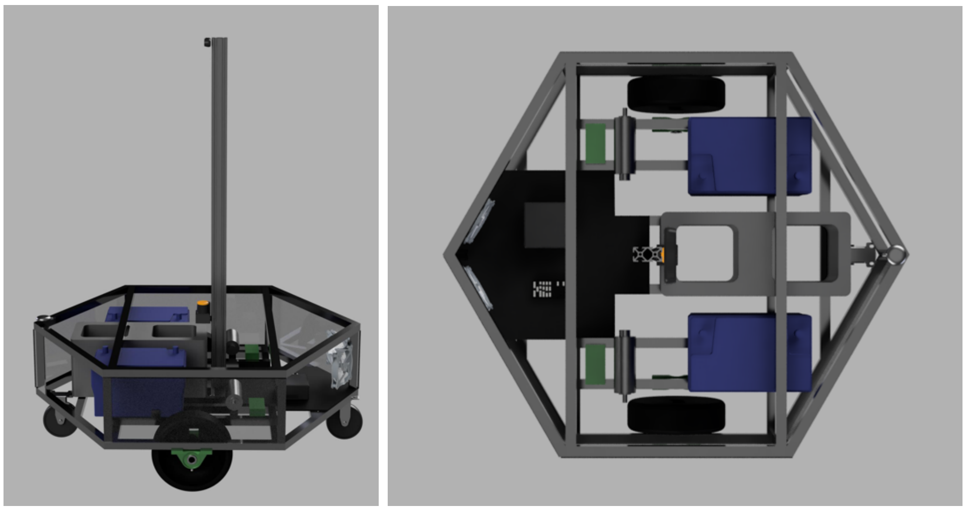
The electronics sub-team wanted to focus on building a simple and robust electrical system that could be constructed modularly. While the design was still being iterated upon, we wanted to make sure that any components we wanted to use would be compatible with the rest of the system. Additionally, the system was designed with a clear path of iterations and improvements in the future, using modules now that could be integrated onto custom circuit boards in the future. Overall, the design process revolved around identifying components and functionality that we need on the vehicle, and making sure the system could support it.

The software sub-team spent most of it's design time prototyping algorithms inside Gazebo and simulating what we could. Because most of the software team were sophomores and freshman, we spent much of the year learning about and implementing different industry standard mapping and planning algorithms we could use for this challenge. This experience helped us decide which of those would work well for the bot, as well as speed up the process of applying them to the physical robot.

2 Mechanical

When generating a design for Pathfinder, it was essential to follow the design process highlighted in the intro with our goals guiding our considerations. Our goals prioritized designs that were as rigid and resilient as possible yet still be simple, with the ability to handle the stresses of the course and the loads we applied while still being a robust solution for the other sub-teams. With this in mind, we used CAD programs like Fusion 360 and Solidworks to model our design and perform static simulations to reiterate and improve as many weak points as possible.

2.1 Housing Design



Pathfinder's housing needed to contain the electronics bay, the payload bank, and our battery compartment. A hexagon frame was adapted instead of any typical rectangular design to take advantage of an easy Minkowski's Sum while calculating for future collisions. The robot can rotate in place without changing its coordinates,so the vehicle can be considered holonomic and a circular with a radius r. Thus, any obstacle can be drawn with an additional radius for collision checking, simplifying the robot to a single point instead of calculating collisions after each action. Once we had decided on the shape of Pathfinder, it was simply dividing the space so that each of the compartments was easily accessible. Our next problem was figuring out the materials we would build with. Through our iterations, we landed on using Silver Anodized Aluminum T-slot extrusions. These extrusions allowed us to place more bracing where we needed to reinforce the structure and hold the load Pathfinder needed to handle.

Pathfinder's drive train uses two motors with planetary gears for additional torque. Motors with chain links and sprockets in a five revolution input to 1 revolution output ratio are directly connected to the tires to ensure we can move through the course and handle the terrain it could encounter. With this, we can utilize Minkowski's Sum, as mentioned previously. To stabilize Pathfinder from tipping forward or backward, we utilized caster wheels that can rotate in any way without adding friction to the system.

2.2 Suspension

Suspension in Pathfinder included spring-loaded caster wheels incorporated in the front and back for passive suspension. While the vehicle traverses through different terrain, the vehicle will pivot on the main 10.5-inch wheels. If the vehicle experiences any uphill, the rear spring-loaded caster wheels will compress so that all wheels maintain traction. If the vehicle experiences any downhill, the front spring-loaded caster wheels compress so that all wheels maintain traction. The main reason why suspension was not implemented on the main two wheels is that the vehicle is moving at a low velocity, and thus any minor bumps or potholes will be negligible. Suspension was used was for the ramps, as the vehicle needs to compress around 1.5 inches in the back to get on the ramp.

2.3 Weather Proofing

Weatherproofing is exceptionally critical to ensure none of our electronics get damaged. The simplest way to weatherproof the vehicle was to divide the different compartments and weatherproof accordingly in the order of importance. Therefore, the electronics bay was the main compartment that was weatherproofed as it contained everything electronics-related. After installing the acrylic plates, all the corners were sealed with silicon and tested for leaks by using compressed air on the outside and feeling for any breeze in the inside bay. Secondly, the sub-team incorporated countermeasures to prevent mud splashing inside the compartment in the battery compartment. Finally, for the payload area, basic rust proof was measures was established. In the case of rain, a rain tarp cover was designed to cover the entire vehicle.

3 Electronics

The Pathfinder is equipped with everything it needs to gather information about its environment, process that information, and navigate through the environment. This en-

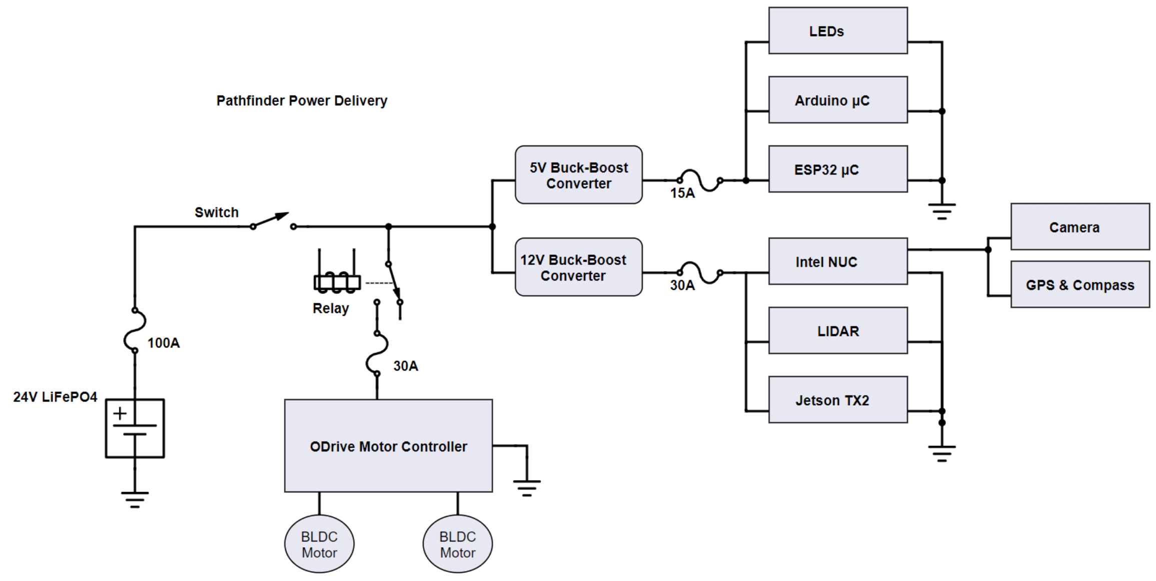


Figure 3.1: Power delivery on Pathfinder

tails several sensors, computers capable of vision processing and path finding, motors, and a way to power it all.

3.1 Power Delivery System

Pathfinder is powered by a 50 amp hour 24 volt LiFePO4 battery. 12v and 5v voltage rails are regulated by buck boost converters to power all needed electronics on the vehicle. Component voltages and the total current draw of the system are constantly measured by analog inputs of an Arduino microcontroller on board. This data is used to get a measurement of the power being drawn from the system. After more data is collected, we will be able to know the average current draws based on the state of the vehicle. This will allow us to make accurate estimates of battery life. Initial estimates place average current draw at 30 amps during normal operation. With the battery we have, expect around 1.5 hours of operation before the battery voltage drops below operable levels. The battery can be fully charged from mains power over the course of 2.5 hours.

3.2 Communication

Most on-board communication is over USB. The NUC and the Jetson TX2 communicate over Ethernet. Wireless communication with the vehicle is achieved using either Bluetooth or low power radio. The lower ranged Bluetooth is used to control the vehicle directly with a controller, using any generic game-pad protocol implementing BLE. Radio is used at ranges up to 400ft to activate the E-Stop and periodically send diagnostic information to a base station if desired. Radio communication happens over a separately powered ESP32 microcontroller. Another identical microcontroller can be used to transmit E-Stop signals from a distance.

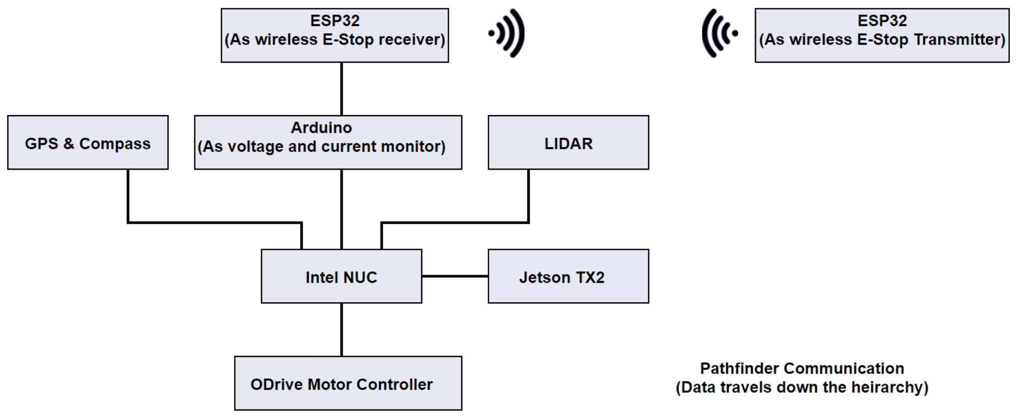

ESP32
(As wireless E-Stop receiver)
ESP32
(As wireless E-Stop Transmitter)
GPS & Compass
Arduino
(As voltage and current monitor)
LIDAR
Intel NUC
Jetson TX2
ODrive Motor Controller
Pathfinder Communication
(Data travels down the heirarchy)

Figure 3.2: Communication on Pathfinder

3.3 Electronics Suite

3.3.1 Motor and Motor Controller

The motor and motor controller used in Pathfinder were chosen to fit the needs of the vehicle. The motors used are two 24V brushless DC motors, geared down with a ratio of 40:1. The gear reduction lowers speed, but allows us to use lower powered motors that prolong battery life. Given that the vehicle should travel at no more than 5 mph, this is a worthwhile trade off. The ODrive motor controller was chosen because a single unit can drive two motors, and because the ODrive can tolerate a wide range of voltages. The ODrive also can monitor its input voltage and gather data from the encoders on the motors.

Regular electric speed controllers were considered for the motor drivers, but were not chosen due the inherent limitation that an ESC will have less control over the exact position of the motor.

3.3.2 Computation

An Intel NUC is responsible for all computation on board the Pathfinder. The NUC was an easy choice for its strong specs, compact form factor, and wide range of tolerable input voltages. The NUC communicates with a Jetson TX2 over Ethernet for computationally intensive path finding tasks.

3.3.3 Sensors

Perception of the immediate environment is done with a Hokuyo UTM 30LX LIDAR, and an Intel RealSense Depth Camera D435. The Intel Depth Camera is essential for identifying the track that the vehicle must stay in, as well as the potholes on the track. The camera can measure the distance of an object up to 10m away, but also is also capable of capturing image in color, making it easier to use computer vision to identify different objects. The depth camera is complemented by the Hokyuo LIDAR, which has a 270 degree range of vision and can detect objects anywhere from 0.1m to 30m away. These two sensors together will work to create a 3D representation of the vehicles environment in software, which we can then work to navigate through.

For navigating the course as a whole, we used a Pixhawk 4 module along with a GPS for a reliable way to know our heading as well as absolute position on the track. This data will also be communicated to the computer system to contribute to the path finding.

3.4 Safety Devices

The ESP32 microcontroller responsible for the wireless E-Stop is powered independently of other components. The microcontroller can directly control a relay that will disconnect the motor controller from any power. This design decision was made so that that the wireless E-Stop could be actuated regardless of the state of the rest of the system. The physical E-Stop button is wired in line with the signal from the microcontroller. Both the physical button and the wireless E-Stop must be active for any power to be delivered to the motors. Additionally, the relay is normally open, so if the microcontroller loses power for any reason, the motor will not run.

The same microcontroller controls warning LEDs that ensure any person near the vehicle will know when it is powered on, or operating autonomously.

4 Software

The software stack that we decided to use to develop Pathfinder is ROS (the Robot Operating System). ROS is an operating system that provides tools and libraries that can help develop robot applications. ROS provided an easy form of communication between the robot and the software with the help of subscriber and publisher nodes. ROS helped ease the use of a programming language to develop in because it supports any language with the help of control interface and nodes. For the project we primarily used Python because everyone in the team had knowledge in Python or could learn it significantly faster.

For the hardware that we had to design the software, we were working with Intel NUC mini PC with an Intel i7-8665U and 32GB of RAM. We've also included a Jetson TX2 for vision processing.

4.1 Obstacle Detection and Avoidance

Our object detection strategy combined two different approaches. The first used the Hokuyo lidar to collect a laser scan of all raised objects on the field. Simulateneously, we are using the Intel Realsense camera to collect RGB-D images of the field. We threshold the image for the white lanes and potholes, and then generated a 3-D pointcloud in the reference frame of the robot of the detected lanes and potholes. These pointclouds are combined to create a 2-D occupancy grid of the environment. This represents the places in the environment that are occupied and need to be avoided. This occupancy map is used for the mapping stage

4.2 Mapping

The mapping subsystem for our robot utilizes the SLAM (Simultaneous Localization and Mapping) algorithm[1] [2]. SLAM takes in two inputs, the robot's sensor observations and it's odometry observations, and attempts to generate a map of the environment and the robots location in the map based on these inputs. The sensor observation we are using for SLAM is the laser scan generated by our lidar. The odometry estimate is generated using measurements from the motor encoders and the onboard IMU, which are then put through an Extended Kalman Filter to reduce error. With this, we are able to create a map for the environment that we are able to path plan through.

4.3 Path Planning

Our robot incorporates two different path planners. The first planner, a global planner, implements an A* planner over the entire occupancy map generated. This planner is complete and optimal given that the generated map is accurate. The goal for this generated is given by the GPS waypoints provided by the IGVC team. The second planner is a local planner, which plans over a smaller region that is close to the robot. This planner utilized the Dynamic Window Approach algorithm [3], which samples a number of controls and generates a path from this series of controls. It then scores each sampled path on criteria like distance to the goal, time, and the distance from obstacles. The local planner runs more often than the global planner, but uses the global path as a goal to plan towards. This approach provides a good balance of minimizing time complexity while generating the most optimal paths possible.

5 Conclusion

5.1 Failure Points and Resolutions

Mechanical Failures

In terms of mechanical Failure of the vehicle, it can be separated into 3 different elements.

- Normal Wear and Tear: In the case of normal wear, failure can occur after multiple experiments and trials in which parts are not replaced quickly or forgotten about such as the dirty wheels on laboratory floors, oil on bearings, and dirt and grass in sprockets after an outdoor experiment. This will not cause a catastrophic failure as it can be quickly fixed within the hour and often times the vehicle remains operational. Therefore, the resolution of this issue is frequent check up on the vehicle on normal wear items.

- Structural failure: In the case of structural failure, extrusions may bend and screws may become lose during testing and operation. This is a large failure point as if this happens outside of a lab setting, it will be difficult to fix before the problem occurs. To prevent such an issue, all screws that are on the motors, wheels, and have constant vibrations have all been secured with loctite. And extrusions that are holding most of the weight of the vehicle have been secured using a threaded rod with nods through the extrusions rather than normal nuts to prevent any chance of loosening up. If there is a failure point in which a screw does come lose, the nut and screw is replaced along with a higher tolerant loctite.

- Misuse Failure: Finally, the biggest failure point of the mechanical design is misuse such as carrying from none secured points on the frame. If the robot is not carried using the main bars going across the entire frame, the connection between the extrusions will experience tension and slowly bend outwards. To resolute this, there are markers on the vehicle to signal where the robot needs to be lifted up during transport and other general use.

Electrical Failures

- Under-voltage Failure: The battery used to power the vehicle has a nonlinear voltage curve that means the system must tolerate higher voltages at full charge, and constantly decreasing voltages at lower charge. Higher voltages can be regulated, but when the components begin to be supplied with too low of a voltage, they may exhibit unexpected behavior. To prevent this, the battery voltage must be monitored over time.

- Electromagnetic Interference Failure: During normal use, the wires from the battery and motor may have current spikes that produce electromagnetic interference. For the more sensitive lower voltage components on the vehicle, this introduces the potential for data loss from interference. This is especially true for the GPS, which is very sensitive to nearby current. To mitigate this issue, the lines that carry the highest current are spaced out from sensitive components. EMF can be difficult to entirely account for in the entire system, so components like the GPS must be tested on the vehicle to make sure they function properly during normal use.

- Short Circuit Failure: If a component is connected incorrectly, or a structural failure breaks a line, there is potential for a short circuit. This problem is especially dangerous with a battery like we are using, that can supply hundreds or thousands of amps for a short time. To mitigate the risk from this problem, many fuses are in line with different components. The entire battery source is in line with a 100 amp fuse that will prevent the worse case scenario, and several lower current fuses are in line with smaller components to protect them.

Software Failure

- Mapping Failure: This can be caused by two primary issues, location drift and sensor issues. Location drift occurs when our odometry begins to drift away from our actual position in the world, which can cause path planning to fail automatically. This is remediated by incorporating an extended kalman filter into the odometry output, which helps reduce drift and error in the sensors. The SLAM algorithm also ensures that our sensor observations match our expected position in the world, while accounting for sensor observations.

- Path Planning Error: The path planner either produces an optimal path, or is unable to find a path through the obstacles given. Assuming there are no mapping errors, which could cause this, this is alleviated by tuning the parameters for our local path planner to ensure that path's it considers to be feasible match the capabilities of the physical robot.

- Vision Errors: This could be caused by errors in our lane detection. Missing a potholes or lanes on the course could result in point deductions or disqualifications for driving over them. Fixes for this included tuning our detection algorithm to be more generous in its classification of lane pixels, and extensive testing.

5.2 Testing

5.2.1 Simulations Employed

Simulations in Virtual Environment

Gazebo, a 3d robotics simulator, was vital in developing the software and testing the robot in a similar environment. First, we setup Pathfinder details in urdf format allowing for an accurate representation of the robot as if we were at the competition. Once that was done, we setup the environment using a similar terrain as in previous IGVC competitions and used randomization to place obstacles around the terrain to a get a sense of the accuracy of our object detection. Then to test other features of Pathfinder, we setup a test environment in which we can test if the different sensors (the lidar sensor, camera, etc) work and how they function with the robot. This enabled us to get a better idea on where to place these parts on our final robot. Gazebo enabled us to make sure the code was properly functional before deploying onto Pathfinder for the competition.

Simulations Concepts Tested

We tested many different scenarios in Gazebo to make sure everything is ready for competition time. Some scenarios we tested were random simulation of environments, sensor simulations, robot behaviors, and testing robotic kinematics. The first test, randomization of the environment, tested our robot's accuracy in any type of environment. When we get to the field we will not know how it will be. So the primary focus of this test is to create as many possibilities to the game day environment so the robot won't run into any hiccups. The second test, sensor simulations, we made a static environment as simple as possible to just test the sensors. So using each sensor we tested the different outputs and see if they produces the outputs that were desired. The third test, robot behavior testing, we tested the robot in one of the randomized worlds and see if the movement was as desired. We wanted to see if mechanically, the parts we put into Pathfinder is desirable or do we need to change anything. For the final test, testing the robot kinematics, we wanted to see the performance based on velocity and acceleration from Pathfinder. We wanted to see the maximum and minimum values of acceleration and velocity we can get from the robot. It will be crucial to see how fast we can get through the course.

5.3 Performance Estimates

5.3.1 Performance Testing to Date

Pathfinder was tested on an outside environment that is similar to the IGVC competition. The white lines, the obstacles, and ramp were placed to provide an ideal similar environment to the IGVC competition. We only went outdoors to test once we were completed with housing all the components of Pathfinder and all the parts were intact.

Mechanically we validated Pathfinder upon final assembly, mounting all the electronics, sensors, batteries, etc. In addition to the essential elements, we added a mock payload. We visually inspected any points of failure like the bearings, sensor, and camera tower and made sure that our electronics were not affected by the environment.

Electronically we validated Pathfinder by continuously testing components as they were added to the system, and making sure everything was delivered sufficient power. The motors were tested to ensure they are powerful enough to move the load we must carry.

In the software we validated Pathfinder by checking if the robot performed similar to the simulations. If it wasn't performing similar to the simulation then we needed to adjust back in the simulation and retest outside.

5.3.2 Initial Performance Estimates

Max Speed	4.6 mph
Acceleration	3 mph/s
Ramp Climbing	19 degrees
Reaction Time	25 to 250 ms
Battery Life	10 hours stand by and 4 hours during operation

5 References

[1] Randall Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5, 02 1987.

[2] Randall Smith, Matthew Self, and Peter C. Cheeseman. Estimating uncertain spatial relationships in robotics. *CoRR*, abs/1304.3111, 2013.

[3] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, 1997.