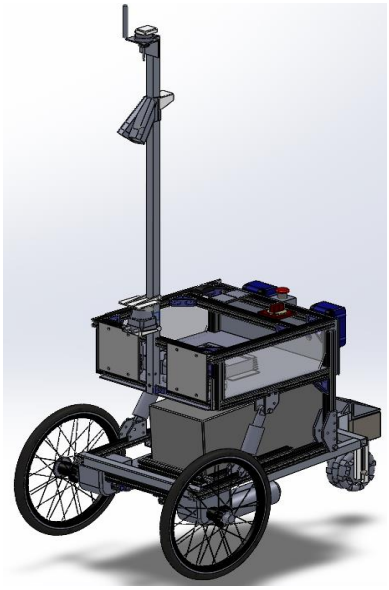
IGVC2022-NOAH

BOB JONES UNIVERSITY
NOAH



Date Submitted: 05/14/2022

Captain:
Joshua Heinrich, jhein429@students.bju.edu

Team Members:

Shouyu Du	sdu568@students.bju.edu
Debanhi Flores	dflor912@students.bju.edu
Peter Labadorf	plaba417@students.bju.edu
Jemima Lin	jlin972@students.bju.edu
Shiphrah Matapathi	smata162@students.bju.edu

Statement of Integrity:

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

Faculty Advisor: James Collins, MSc
Associate Professor
Department of Engineering
Bob Jones University

Signature_____ Date: May 14, 2022

INTRODUCTION

Noah is an enhanced version of Bob Jones University's 2021 Intelligent Ground Vehicle Competition (IGVC) entry, Melchizedek[1]. While Noah uses the same sensor suite and has a similar structure to Melchizedek, numerous designs and feature improvements have been added to Noah. Here is the list of new features:

- Road bike front wheels
- Rear omni wheel
- Redesigned frame
- Weight reduction
- Dual-lane cameras
- Sensor mast shortening
- Motor controller
- Suspension relocation

ORGANIZATION

The 2022 team was organized into three teams: electrical, mechanical, and software. Each team member was assigned to independently develop a specific part of the design.

Table 1. Student Contributions.

Team Members	Academic Department and Class	Subject Area	Hours
Shouyu Du	Engineering, Junior	Software	40
Debanhi Flores	Engineering, Senior	Electrical, Software	136
Joshua Heinrich	Engineering, Senior	Mechanical	136
Peter Labadorf	Computer Science, Math, Junior	Software	48
Jemima Lin	Engineering, Junior	Electrical	64
Shiphrah Matapathi	Engineering, Senior	Mechanical	136

DESIGN ASSUMPTIONS AND DESIGN PROCESS

The 2022 team designed Noah based on Melchizedek, the robot from last year's Bob Jones University IGVC team. Noah was designed and assembled for the IGVC Auto-Nav competition. Noah uses the Robot Operating System (ROS) as the basis for its software and is controlled by three stacked Raspberry Pi 4s (Tri-Pi). The team's objective is to minimize the time taken to navigate the course.

[1] Some parts of this report are derived from the 2021 BJU Melchizedek Design Report.

To accomplish this objective, we used the design process in Figure 1. We used ticketing software to track and identify issues that arose with the design over the course of development. Using CAD prototypes, our team was able to quickly develop several components of the robot by printing them to add more features or to repair broken parts. Several iterations of the CAD prototypes were created using design tools like SolidWorks, as seen in figure 2, and through a process of testing and modification, the final parts were implemented on the robot.

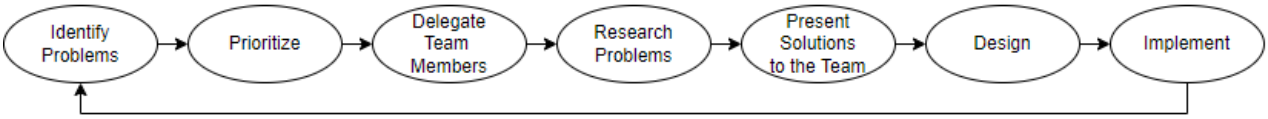


Figure 1. Design Process

Figure 2. Prototypes of the Camera Mount

INNOVATIONS

Road Bike Wheels

This year's IGVC Auto-Nav Challenge takes place on asphalt pavement. To adapt Noah to this change, our team decided to update Noah with 14-inch bicycle road wheels instead of last year's tractor wheels. We created a custom adaptor to convert from the round output shaft of Noah's worm gearbox to the wheel. This innovation not only reduces the weight of the overall vehicle but also reduces its width.

Figure 3. Picture of tires on Noah

Omni Wheel Back Wheel

The change from grass to asphalt pavement allows our team to use a lightweight Omni wheel in place of a heavy caster wheel as the rear wheel. Prior to 2022, a wheel of this size would not be feasible to use on the grass, but the change to pavement allows us to use a Rotacaster 125mm Triple Polyurethane Roller Sealed Bearing Omni wheels.

Figure 4. Picture of Omni wheel on Noah

3D Printed Onyx Back Wheel Mount

We use a 3D carbon fiber printed mount for our back wheel, not only because we had to create our own mount for the new Rotacaster Omni wheel but because we wanted to reduce the overall weight of Noah. This innovation works with high density foam (the block in Figure 5) to provide fewer vibrations and weight distribution; and it reduces the weight of the overall vehicle by approximately 4 pounds.

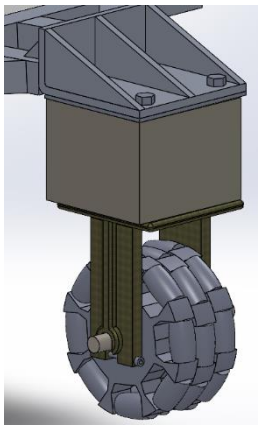


Figure 5. CAD Picture of back wheel mount

Dual Lane Detection Cameras

One major improvement with Noah is our improved camera setup, shown in Figure 6. Previously, Melchizedek employed a single camera to detect the lanes and potholes. The field of view of the single camera limited the path planning algorithm to efficiently plan short trajectories. We created a dual camera mount to detect the lines and potholes farther to the side of Noah, allowing for more complete mapping. To provide maximum stiffness and a low cost, the dual camera mount was 3D printed using polylactic acid (PLA). The innovative design improved the maximum width of the view of the camera from 3.2 m to 4.2 m.

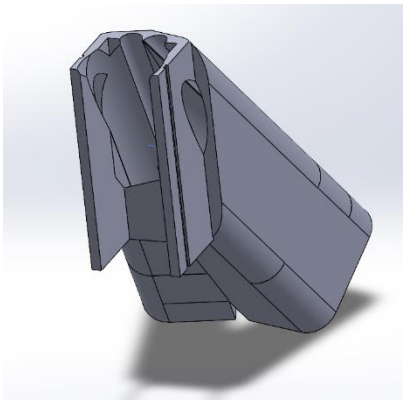


Figure 6. Design of Dual Camera Mount on Noah

Weight Loss

The changes above plus several other modifications such as the battery brackets, the e-stop tower, lighter motor controller, frame simplification, etc. resulted in a total weight reduction of 34 pounds, 28% lighter than the previous vehicle.

OTHER UNIQUE FEATURES

Hot Swappable Batteries

The dual batteries wired in parallel allow one battery to recharge while the other one is being used, as seen in Figure 7. The running time of the robot with full batteries is 150 minutes, and the recharge time of each battery is 80 minutes. The hot-swappable batteries combined with the fact that the recharge time is less than the operating time enables a continuous runtime for the robot.

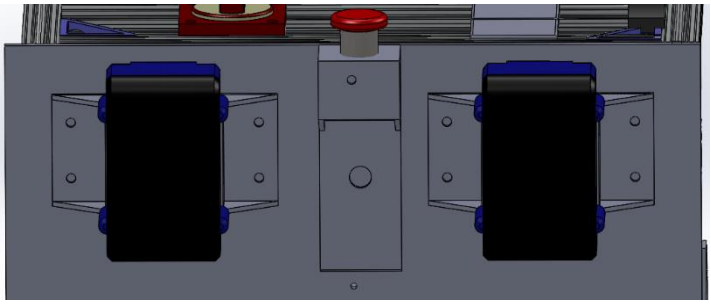


Figure 7. Hot Swappable batteries

Computational Load Balancing with Raspberry Pi's

Instead of a single computer, three networked Raspberry Pi's (the Tri-Pi) are used in the control system of Noah. The use of the Tri-Pi is cost efficient, makes the robot drastically lighter, and consumes less power. The combination of Raspberry Pi's provides a higher processing power than traditional computers.

Software E-stop and Tip Watchdog

Because of a few safety incidents with Melchizedek and Noah where the robot tipped over because of high deceleration, we decided to use the roll and pitch values from the IMU to determine if the robot has tipped over, and then electrically trigger an emergency stop using IO pins on the Raspberry Pi. This innovation prevents any further damage by spinning wheels in case of another tipping incident.

MECHANICAL DESIGN

Overview

Noah's chassis was improved from the Bob Jones University's 2021 IGVC robot, Melchizedek. Noah's design rectifies various flaws in Melchizedek, such as the wheels, bulky frame, mast height, and location of the suspension. Noah features a drive base designed for use on asphalt, a new camera system, and an excellent mechanical design overall.

Drivetrain

Noah uses a differential drive, employing 14-inch bike tires, driven by two National Power Chair R81-series motors through a right-angle worm gearbox. Each motor features a US Digital E6S encoder to allow the motor to provide feedback to the RoboClaw 2x30A Motor Controller. The robot uses a single rear omni wheel, as seen in Figure 7, to provide support with nominal resistance when navigating through the course.

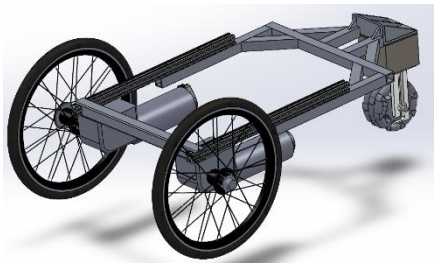


Figure 8. Drivetrain and Chassis.

Structure and Housing

Noah is constructed using 1" T-slot aluminum, along with 1" box 1/8" box wall tubing. The upper frame is made from almost exclusively T-slot aluminum, because of its modularity and strength. The lower frame uses welded boxed aluminum along with some T-slot to allow for easier mounting. Noah's sensor mast, positioned in the middle of the two wheels as seen in Figure 9a, allows for the lane-detection cameras, IMU, GPS receiver, e-stop antenna, and light to be mounted away from the main body of the robot. This feature reduces visual and electromagnetic interference on the camera, IMU and GPS.

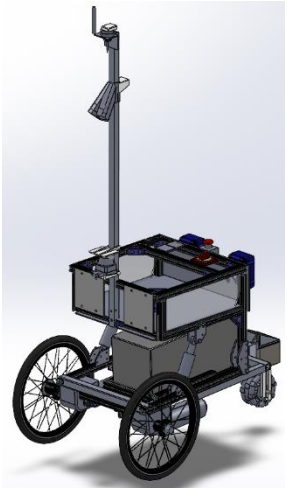


Figure 9a. Frame and Sensor Mast.

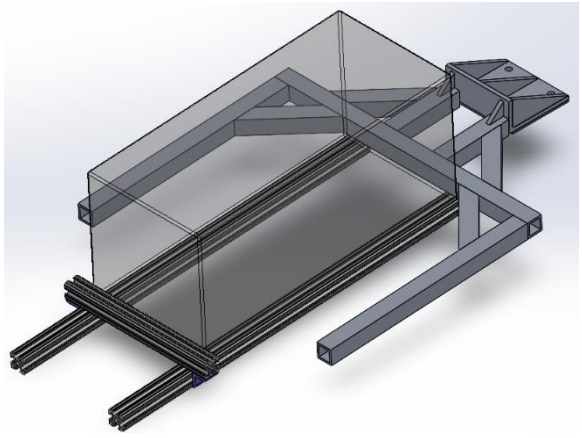


Figure 9b. Payload Holder.

The lower frame of the robot consists of a T-slot payload holder which holds the payload in place as seen in Figure 9b. The positioning of this holder allows for an easier and efficient placement of the payload into the robot and still maintains a low center of gravity.

Suspension

Noah uses two FastAce Coil-Over Shock Absorbers, as shown in Figure 10, to support and reduce the force of impacts on the main body. The suspensions are relocated to reduce the body size and reduce weight. On the back wheel, the foam (Figure 11) supports and acts as a shock and vibration reducer.

Figure 10. FastAce Shock on Robot.

Figure 11. Foam on the wheel bracket

Electronics Mounting

The TRI-PI and ethernet are encased in a polycarbonate frame to ensure the safety and protection of these electronics as seen in Figure 12a and 12b. Using polycarbonate ensured that the frame could withstand heavy vibrations caused by the robot better than other common plastics.

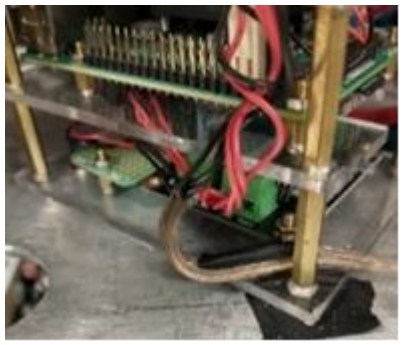


Figure 12a. Raspberry Pi Mount.

Figure 12b. Ethernet Mount.

Weather Proofing

All the electrical components and Tri-Pi's are encased in sheets of corrugated plastic around the main body which protects the electrical components and ensures that the robot functions unerringly in any given weather as seen in Figure 9a. The sensors, such as the camera and the LIDAR, that could not be

placed in the main body are guaranteed minimal to no exposure to harsh weather. The cameras are enclosed in 3D printed casing (Figure 13a) and a 3D printed sun shield protects the LIDAR (Figure 13b).

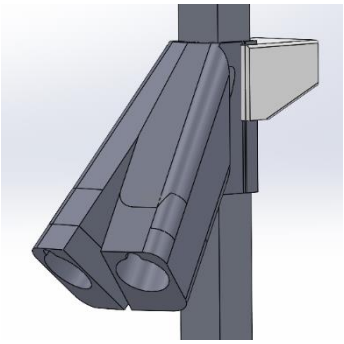


Figure 13a. Dual Camera Mount.

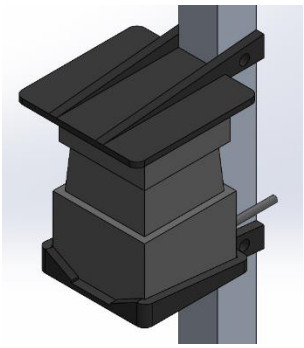


Figure 13b. Sun Shield on LIDAR.

ELECTRICAL DESIGN

Overview

Noah uses two 24V Kobalt batteries, which are wired in parallel, as a power supply. These same batteries power all the electrical components by using multiple voltage converters to supply the exact voltage as shown in Figure 14. A new motor controller was implemented onto Noah thus replacing the outdated one previously used on Melchizedek. A schematic of the connection of each component can be seen in Figure 15.

Power Distribution System

The robot power system consists of two 24V 5A batteries in parallel with a combined energy capacity of 240Wh. Connecting these batteries in parallel allows Noah to still have power even when one of the batteries is missing. At a normal speed, Noah utilizes a total of 4A making the vehicle max energy consumption of 96W. This means that a single set of batteries allows Noah to run for about 150 minutes, and the charging time for each set of batteries is 1 hour and 20 minutes. This is a great advantage since it enables Noah to be hot-swappable. For the power system, the power distribution was split so that one power switch would control the mechanical systems, and the other power switch would control the computer systems. Figure 14 shows an overview of the implemented power distribution.

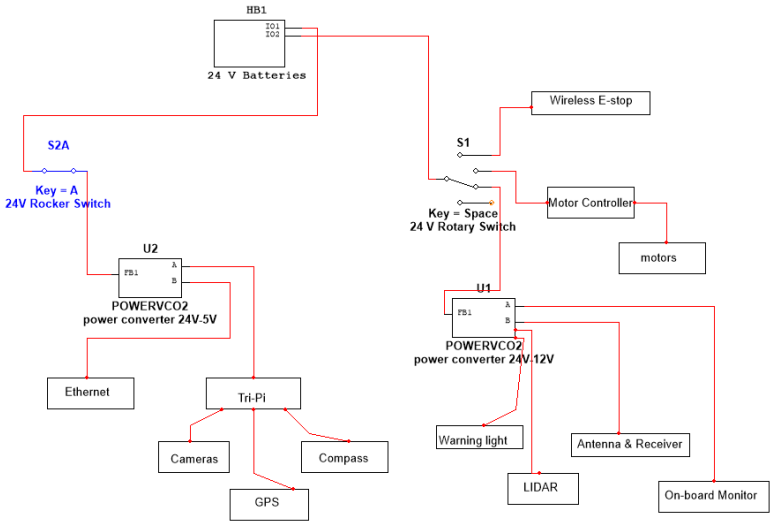


Figure 14. Power Distribution Overview.

Electronics Suite Description

Noah's electronics system is controlled by 3 Raspberry Pi 4s. These computers allow feedback control for the robot. The 3 Raspberry Pi's are named, from top to bottom, Apple Pi, Blackberry Pi, and Cherry Pi. Figure 15 shows all the components as well as how they are each connected.

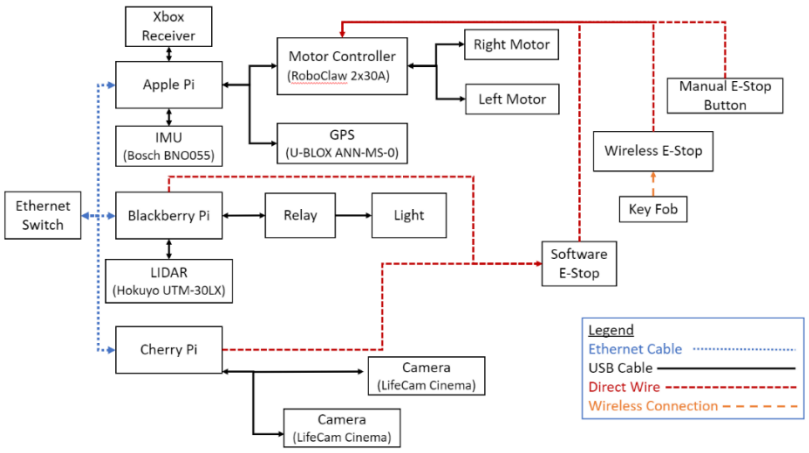


Figure 15. Diagram of Control System.

Motor Control

The previous motor controller that was used in Melchizedek is no longer supported. Because of ongoing issues with feedback loops, Noah now uses a RoboClaw 2x30A controller. This motor controller has several advantages over the previous controller used. The smaller form factor and decreased weight allow for cleaner wiring and better handling, and the pre-written ROS motor controller allows the developers to focus on other parts of Noah. Figure 14 shows the RoboClaw is powered by a 24V rotary switch and powers both motors.

SOFTWARE STRATEGY AND MAPPING TECHNIQUES

Overview

Our team uses the inherently parallel nature of ROS to distribute the computational load across the Tri-Pi. As seen in Figure 16, we extensively leveraged open-source code to simplify the development experience. Using default ROS localization and navigation packages allowed the developers to focus on integrating sensors into an uncommon localization configuration and on the fine-tuning of navigation performance.

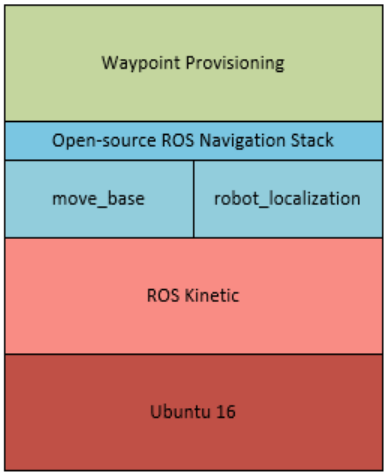


Figure 16. Navigation Software Stack.

Our team designed the software architecture around the move_base and robot_localization ROS packages. As seen in Figure 17, we fed GPS, IMU, and wheel odometry measurements into an extended Kalman filter to produce an exact position estimate, and we combined the lidar and vision data into a cost map using functionality provided by the move_base package, which uses the cost map and the position estimate to produce a velocity command intended to direct the robot towards its intended goal.

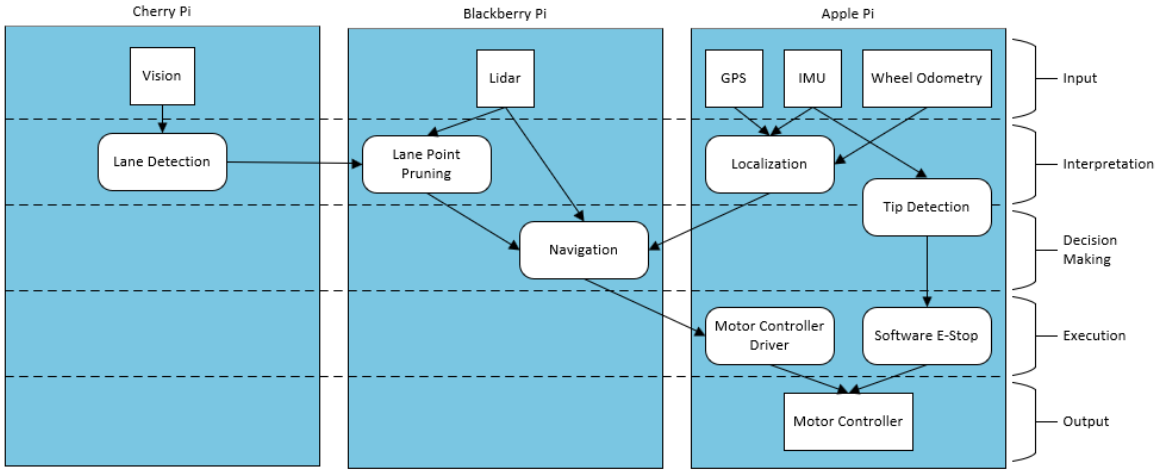


Figure 17. Software Architecture Design.

Obstacle Detection and Avoidance

As seen in Figure 18, obstacle detection is implemented by using the ROS move_base support for lidar and camera data to fuse them into a cost map, creating a global cost map using a local planner to follow the global path while avoiding obstacles. We used the dwa_local_planner ROS plugin as our local planner. That planner implements the Dynamic Window Algorithm, which simulates a range of trajectories that a robot could take in the next few seconds, filters out the trajectories that cause the robot to collide with an obstacle, chooses the trajectory according to several factors, and commands the robot to take that trajectory. The team chose the Dynamic Window Algorithm because it performs well with static obstacles, and because it is easy to understand and debug.

Software Strategy and Path Planning

The software implements a waypoint-based path generation strategy to produce global plans for the robot to follow. The GPS coordinates are projected to an SI coordinate system with the origin at the starting position of the robot. The global path is then calculated by the NavFN global planner, which uses Dijkstra's shortest-path algorithm on the costmap to produce the path.

Map Generation

Noah's configuration of move_base dynamically generates the local and global maps during robot motion. As seen in Figure 18, the ROS move_base library automatically produces costmaps based. Because the white stripes on obstacles can get recognized by our vision system as lanes, the software deletes points in the lane detection.

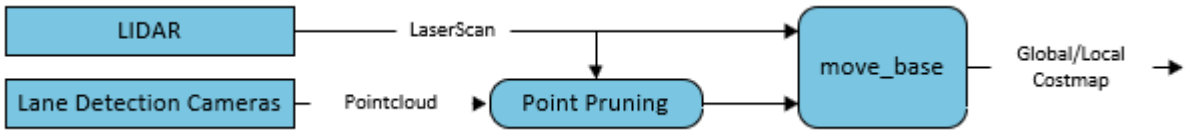


Figure 18. Map Generation Flowchart

Additional Creative Concepts

Because of safety incidents in the past, the team decided to include the ability of software to trigger a hardware e-stop. This is useful when the IMU detects that the robot is tipped over, which has happened in testing. Custom nodes were written that subscribe to error topics, and if the error is fatal, a hardware e-stop is triggered. The hardware implementation of this is described in the Vehicle Failure Points and Resolutions section.

FAILURE MODES, POINTS, AND RESOLUTIONS

Vehicle Failure Modes and Resolutions

Table 3. Failure Modes

Failure Modes	Resolutions
Sensor failure	Pause the run and wait for a reconnection
Tipping over due to high deceleration	Preventive tipping by limiting yaw rate at high speeds and maximum deceleration
	In case of tipping despite preventive measures, sense and E-stop

Vehicle Failure Points and Resolutions

Table 4. Failure Points.

Failure Points	Resolutions
Possible short circuit and overcurrent in a wire	Ensure that all wires are properly fused, and exposed connections are covered by 3D printed covers
Unsteady power connection to CPUs	Use USB power connectors instead of GPIO pins
Electronic components fail due to high vibration	Shocks mounted to the main body to absorb large impacts and polycarbonate mounts to reduce constant vibrations on smaller electronics
3D printed parts have the possibility of breaking	Important parts incorporate carbon fiber to maximize strength
Sensor mast could break or bend	Reduce the chance for robot could tip in software
Raspberry Pi microSD's could become corrupted	Pack extra microSD's pre-imaged with the latest version of the code

All Failure Prevention Strategy

We considered end states that would cause a robot failure, such as crashing, fire, and having to perform an e-stop. The team then reviewed possible ways these states could be reached, for instance, a damaged sensor or short-circuiting. With these possible failures in mind, the team reviewed ways to prevent the failures, for example, having redundances for each sensor and adding correctly sized fuses. This strategy has led to ensure the safety of Noah.

Failure Testing

Table 5. Testing Plan.

Testing Plan	Expected Result
Unplug the motor controller from Raspberry Pi while running	Motor controller stops motors
Simulate robot in Gazebo to test tipping over	Find the max velocity and yaw the robot can withstand without tipping
Simulate loss of sensor data in Gazebo	Depending on the sensor loss, the robot either stops itself or continues driving

Cost Estimate

Table 6. Itemized Cost Estimate.

ITEM	QUANTITY	COST ($)
Digital Camera	2	115.98
LIDAR	1	4675.00
IMU	1	34.95
GPS	1	179.00
Blackberry Pi 4s (4GB RAM)	2	110.00
Blackberry Pi 4s (8GB RAM)	1	75.00
Omni Wheel	1	44.86
Front Wheels	2	32.00
Motor Controller	1	184.80
Wiring and Framing	-	375.00
Batteries	4	220.00
Total Cost		**$6046.59**

SIMULATIONS EMPLOYED

Noah was modeled in Gazebo as shown in Figure 19. The physics engine in Gazebo was used to estimate the minimum turning radius to avoid tip over when driving at maximum speed. The physics engine in Gazebo was used to estimate the minimum turning radius to avoid tipping over when driving at maximum speed. Noah was modeled in Gazebo as shown in Figure 19 and 20.

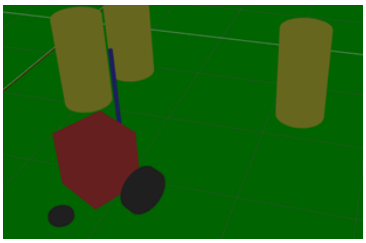


Figure 19. Gazebo Simulation of Minimum Turn Radius

We also use Gazebo to simulate and test the parameters for our navigation code before deploying it to the robot. Figure 19 shows a simulation of Noah planning a trajectory.

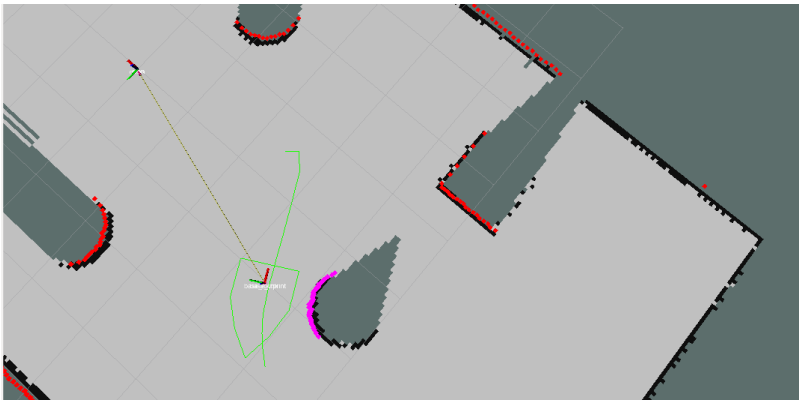


Figure 20. Gazebo Simulation of Noah's Navigation Configuration

PERFORMANCE TESTING TO DATE

- The team timed how fast the robot moved 10 meters multiple times. The fastest time the robot achieved was 4.8 seconds which results in a speed of 2.08 m/s or 4.65 mph. Due to the limits of the motor controller, this is max speed that can be obtained with hardware limiting, while staying under 5 mph.
- The robot has an estimated battery life of 150 minutes and has 2 sets of batteries with a recharge time of 80 minutes, allowing for continuous operation
- Barrels can be detected from 50 feet. Lines and potholes are detected at a maximum of 7.5 feet in front and 8 feet along the sides.
- The robot navigates to waypoints 30 feet away while avoiding obstacles.

INITIAL PERFORMANCE ASSESSMENTS

- Last year's issue with unintentionally curving to the right is fixed
- Navigation correctly identifies and avoids obstacles.
- The robot was initially able to drive above the 5 MPH speed limit, but after reconfiguring the motor controller, the speed is correctly limited.

CONCLUSION

Noah improves on the existing framework and systems of Melchizedek to adapt the robot for this year's IGVC competition. By adding an assortment of features – including two digital cameras, LIDAR, IMU, GPS, and new front and back wheels, Noah is built to easily traverse the competition course and complete competition tasks. At the time of writing, the software is still being implemented. Noah is on track to successfully compete at IGVC 2022.