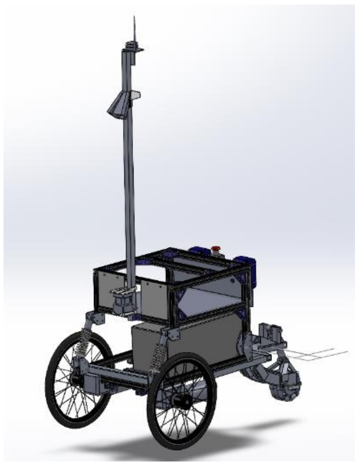
IGVC2023-OBED

BOB JONES UNIVERSITY
OBED



Date Submitted: 5/15/2023

Captain:

Peter Labadorf, plaba417@students.bju.edu

Team Members:

Isaac Graham	igrah563@students.bju.edu
Hannah Hagans	hhaga009@students.bju.edu
Isaiah He	jhe136@students.bju.edu
Aaron Pio	apio290@students.bju.edu
Christopher Zuehlke	czueh332@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 [illegible] Date: May 15, 2022

1. INTRODUCTION

1.1 Overview

Bob Jones University's 2023 Intelligent Ground Vehicle Competition (IGVC) entry, Obed, has several improvements on the previous entry, Noah. Obed uses the same sensors as Noah but has undergone some hardware improvements and major software updates. The following is a list of improvements and changes:

- Codebase ported to ROS2/Python3
- Added router
- Removed debug screen
- NVIDIA Jetson
- Lowered payload
- Improved Back wheel suspension

1.2. Organization

The 2023 team was made up of a software and a mechanical group. Each team member made contributions to program and algorithm or mechanical decisions.

Table 1.1. Student Contributions.

Team Members	Academic Department and Class	Subject Area	Hours
Aaron Pio	Freshman, Engineering	Software	30
Hannah Hagans	Freshman, Engineering	Software	120
Peter Labadorf	Junior, Science	Software	160
Christopher Zuehlke	Senior, Engineering	Software	80
Isaiah He	Junior, Engineering	Mechanical	80
Isaac Graham	Junior, Engineering	Mechanical	30

1.3. Design Assumptions and Design Process

Obed was upgraded to Python 3 and the second Robot Operating System (ROS2) as the basis for its software. Obed is controlled by two Raspberry Pi 4s and an NVIDIA Jetson. The team's objective is to minimize the time taken to navigate while maintaining safety throughout the course.

To accomplish these objectives, we used the design process in Figure 1.1 We made a list of problems from last year's design that needed to be fixed and areas that needed improvement, then delegated team members to solve these problems, and after each problem was solved, the solution was reviewed by the team, and implemented. One example of how this design process was used was to fix the power supply to the computational stack. This power supply needed to be modified to support the new computer architecture. To begin, a team member was assigned to fix the problem. The team member designed a solution to the problem, presented it to the team, then tested the solution, see Figure 1.2a. Once the prototype was tested, the team researched weaknesses and improvements to the design. A custom PCB was designed to fix these weaknesses and incorporate the given improvements, see Figure 1.2b.

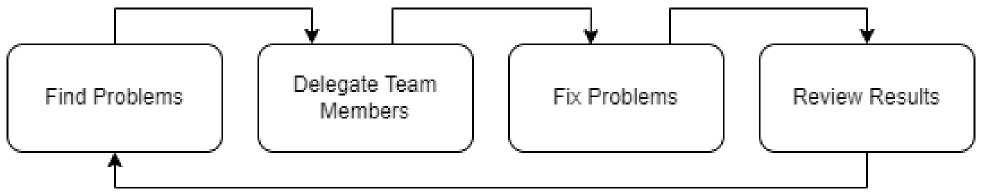


Figure 1.1. Design Process

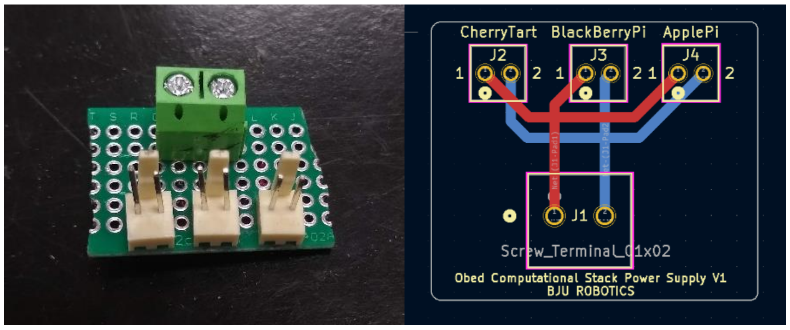


Figure 1.2a-b Power supply Prototypes 1 (left) and final PCB design (right)

2. INNOVATIONS

2.1. GPU Accelerated Lane Detection

Last year, we had significant performance issues with our vision pipeline, which was running at 15 FPS, half the framerate of the camera. This year, we experimented with using GPU acceleration by replacing one of the Raspberry PIs with an Nvidia Jetson Nano. The new vision pipeline runs at the camera's frame rate, and the Jetson also runs the navigation stack.

2.2. Air Shock Based Back Wheel Suspension

A new air shock based supported back wheel was designed to improve the suspension system of Obed, which keeps on last year's omnidirectional wheel, but completely redesigning the wheel mount to include a shock. The new system is 3D-printed in carbon fiber filament.

2.3. Custom Simulation Platform

In previous years, the team struggled with simulating their sensor setup in Gazebo to run end-to-end navigation tests in software. This year, we developed NavSim, which provides drop-in replacements for LIDAR and camera sensors with simple 2D image-based navigation scenarios, instead of the complicated configuration and instability on limited hardware configurations present with Gazebo. Details of the simulation platform are provided in the *Simulations Employed* section.

2.4. Novel Path-Planning Algorithm

Because of runtime issues arising from path-planning algorithm performance, the team developed a new obstacle polygon-based path-planning algorithm which was designed to have better performance in large operating areas with sparse obstacles. Details of the algorithm are provided in Section 8.4.

3. OTHER UNIQUE FEATURES

3.1 Improved Development Workflow

In previous years, the robot's wireless hotspot network was separate from the internal network of the computation stack, and one of the computers acted as a bridge between the networks. This made it difficult to deploy code, launch the software, and debug problems. Updating or installing software from the internet was especially complicated and required changes to network configuration files. This year, we added an internet router to the robot and configured it to use the same network for wired and wireless clients. This allows a direct connection between a developer's computer and a computer on the robot and allows for team members to debug issues on their laptops by subscribing to ROS topics published remotely.

3.2 Hot Swappable Batteries

When working in the field, not having to turn off the robot to change batteries allows the control system to always be on, reducing downtime. Obed's innovative parallel rechargeable Kobalt battery design, shown in Figure 3.1, allows for the control system to run from one battery. This allows for the batteries to be changed without turning off the robot.

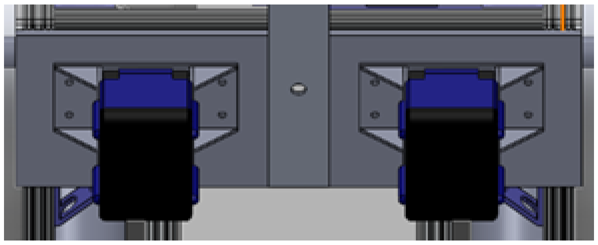


Figure 3.1. Dual Batteries allow for Hot Swapping.

4. MECHANICAL DESIGN

4.1. Overview

Obed's chassis was improved from the Bob Jones University's 2022 IGVC robot, Noah. Obed's design rectifies various flaws in Noah, including re-engineering of the rear suspension, relocation of payload, and location of the front suspension.

4.2. Drivetrain

Obed 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-mounted omni wheel, as seen in Figure 4.1, to provide support with nominal resistance when navigating through the course.

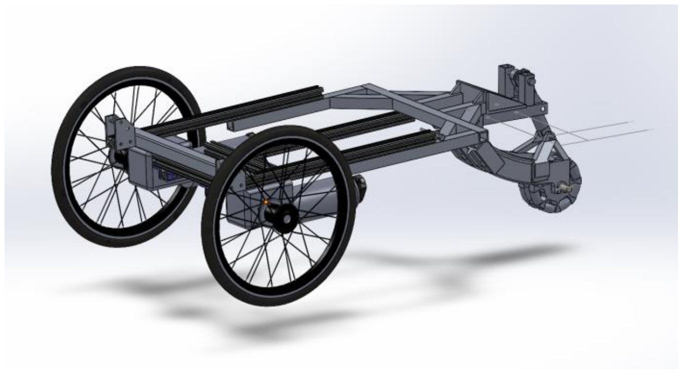


Figure 4.1. Drivetrain and Chassis.

4.3. Structure and Housing

The main framework of Obed is constructed using 1" T-slot aluminum. Most of the upper frame is made from T-slot aluminum, because of its modularity and strength. The lower frame uses welded boxed aluminum combined with T-slot for easier mounting. The sensor mast mounted towards the front of the robot provides mounting locations for Obed's main sensors, including the lane-detection cameras, IMU, GPS receiver, emergency-stop antenna, and light. This mast allows the sensors to be mounted away from the main body of the robot, reducing electromagnetic interference on the sensors. Figure 4.2. shows the sensor mast and sensors mounted on the robot.

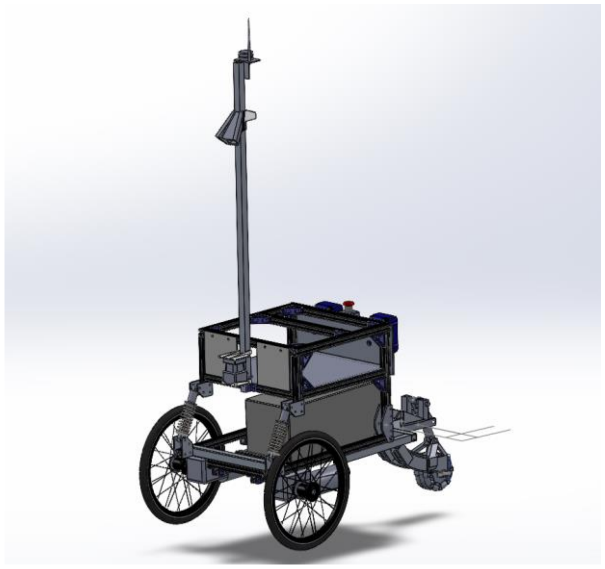


Figure 4.2. Frame and Sensor Mast.

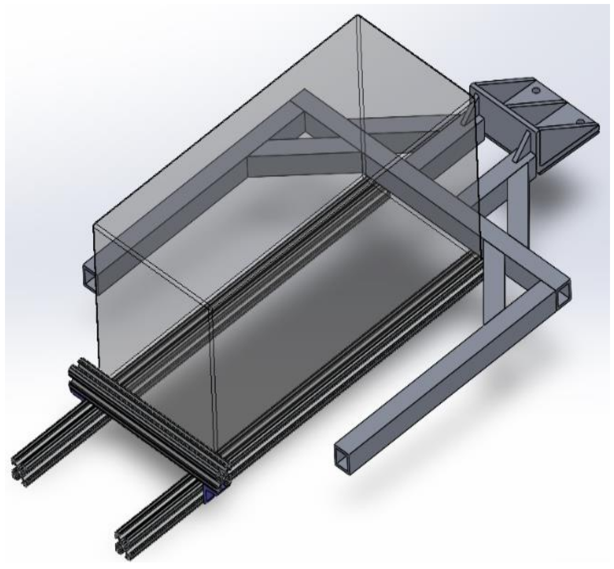


Figure 4.3. Payload Holder.

The robot's lower frame consists of a T-slot payload holder which holds the payload in place as seen in Figure 4.3. This holder's positioning allows for easier and more efficient placement of the payload into the robot and still maintains a low center of gravity.

4.4. Suspension

Obed uses two FastAce Coil-Over Shock Absorbers, as shown in Figure 4.4, to support and reduce the force of impacts on the main body. This year, the back wheel's suspension was changed from a foam cushion to a single Meroca air shock absorber, as seen in Figure 4.5. The front shocks were relocated to ensure that the main body remained horizontal.

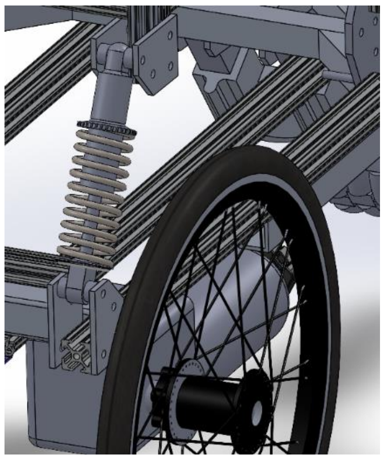


Figure 4.4. FastAce Shock on Robot.

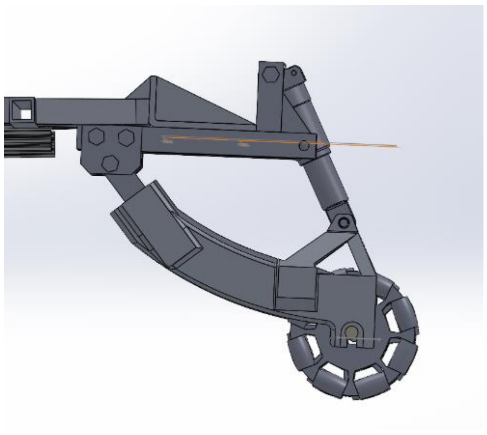


Figure 4.5. Rear Suspension System

4.5. Electronics Mounting

The Computational stack's mounting was upgraded to hold the new Nvidia Jetson, along with two Raspberry Pi's. To resolve a design flaw from last year's design, the Computational stack was orientated so that the inputs/outputs of the computers are facing upward. This orientation creates a simple environment for maintaining and reconfiguring sensors, as seen in Figure 4.6 and 4.7.

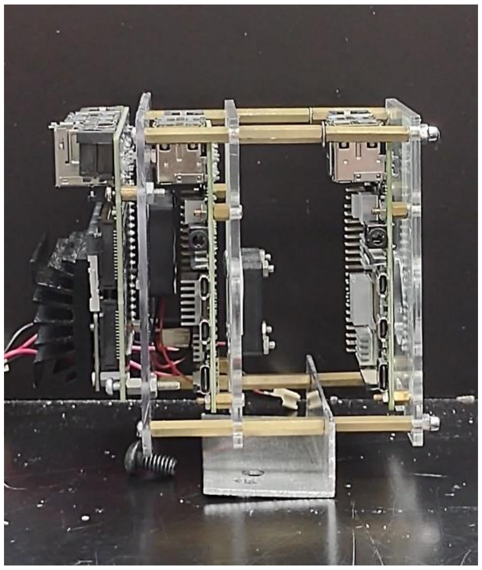


Figure 4.6. Computational Stack Mounting

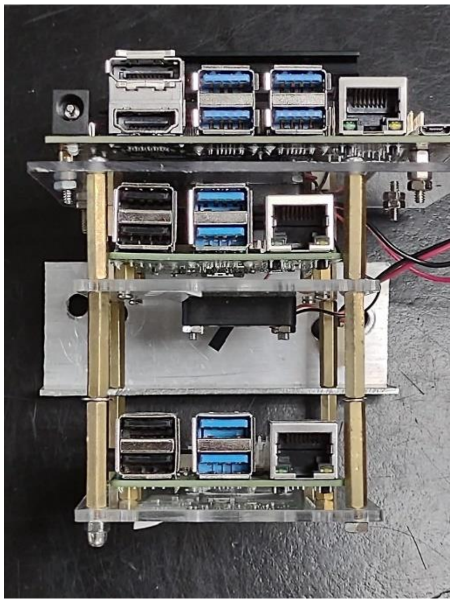


Figure 4.7. Accessible computer ports

4.6. Weather Proofing

All the electrical components, including the computational stack, are encased inside the main frame of the body. The main frame of the body is sealed with clear plastic on each side. This plastic ensures that Obed's sensitive electronics remain untouched by the weather. The sensors that are not inside the main body of Obed have individual weatherproofing systems. The lane detection cameras are enclosed in a 3D printed casing to protect them (Figure 4.8). The LIDAR is protected by a 3D-printed shield above the sensor (Figure 4.9).

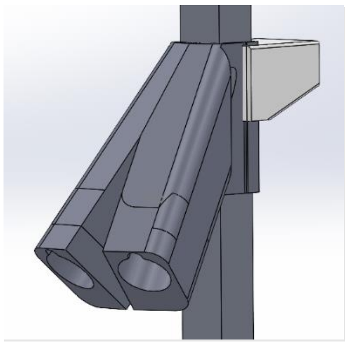


Figure 4.8. Dual Camera Mount.

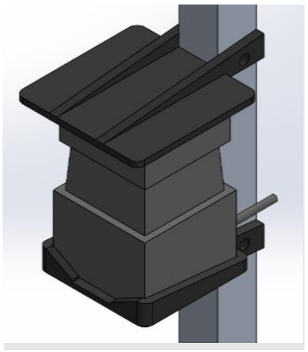


Figure 4.9. Sun Shield on LIDAR.

5. ELECTRICAL DESIGN

5.1. Overview

Obed is powered by two 24V Kobalt drill batteries wired in parallel. These batteries are the main power source for the entire robot. A schematic of the connections of each component can be seen in Figure 5.1. The power supply to the computational stack was also overhauled.

5.2. 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 Obed to still have power when one of the batteries is missing. At a normal speed, Obed uses 4A, making the vehicle's max energy consumption of 96W. This means that a single set of batteries allows Obed to run for approximately 101 minutes (about one hour and forty minutes), and the charging time for each set of batteries is 1 hour and 20 minutes. Since the charging time is less than the running time, this allows us to keep the robot running with a set of 4 batteries. Aided by the fact that Obed's batteries are hot-swappable, Obed can be continuously run without having to be powered down. To improve the efficiency of the power system it was split into two sub-systems. The first sub-system is for the computer stack and router, and the second is for the mechanical interface systems. This is an advantage because both systems are not always needed, and thus power can be conserved. Figure 5.1 shows an overview of the implemented power distribution.

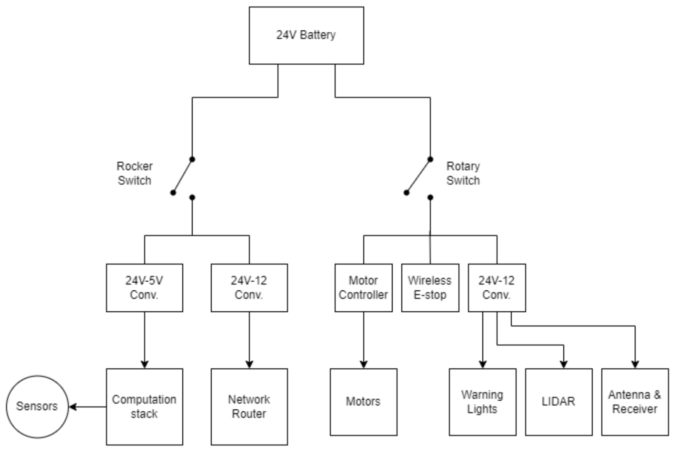


Figure 5.1. Power Distribution Overview.

5.3. Electronics Suite Description

Obed's electronics system is controlled by 2 Raspberry Pi 4's and a Nvidia Jetson. These computers allow feedback control for the robot. The computers are named, from bottom to top, Apple Pi, Blackberry Pi, and Cherry Tart. Figure 5.2 shows all the components as well as their relationship to each other.

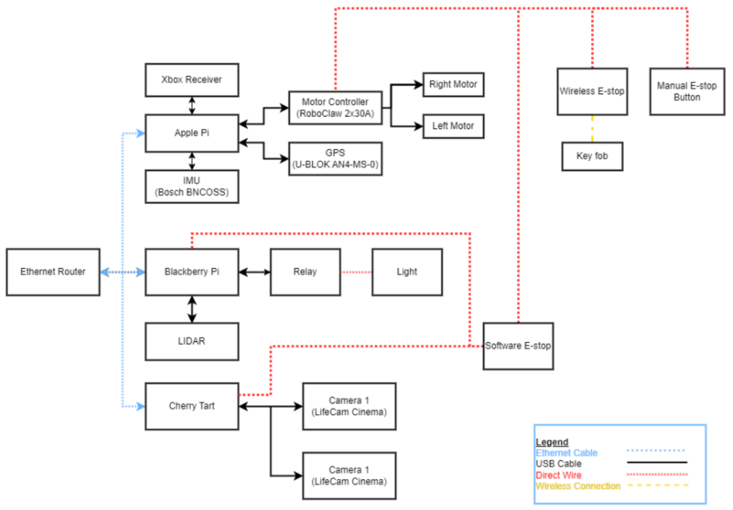


Figure 5.2. Diagram of Control System.

5.4. Safety Devices

There are multiple safety devices integrated into Obed's system. The main two are the e-stop system, and the light indicator. The e-stop system is integrated into Obed's motor control system and will stop the motors in the event of an e-stop being triggered. The e-stop has three ways to be triggered: through the e-stop switch on the back of the robot, through the wireless e-stop fob, or by the software. The second main safety system is the indicator light on the sensor mast. When the Robot is in autonomous mode, the indicator flashes on and off to indicate it is running autonomously.

6. SOFTWARE STRATEGY AND SYSTEMS INTEGRATION

6.1. Overview

Obed's software architecture is designed around ROS 2, which was chosen because of its ability for drop-in addition of third-party software components, its flexibility in configuring sensors and other data streams, and its native ability to be configured as a distributed system. The team leveraged existing ROS packages for sensor drivers, localization, and navigation. Figure 6.1 shows the technology stack used in the design of the software architecture.

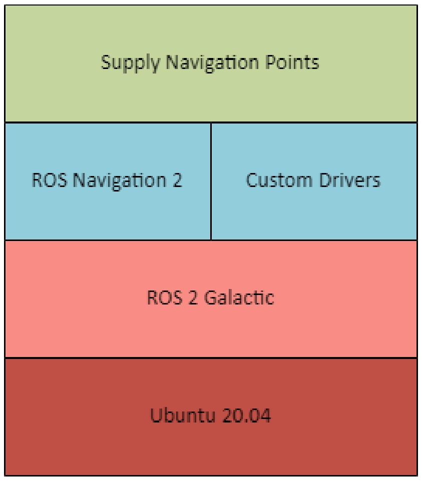


Figure 6.1. Obed's Software Stack.

The team used the Nnvigation2 ROS package to provide map generation, mission control, replanning and recovery, path following capabilities, and developed an open-source plugin implementing a novel path-planning algorithm. Position sensor fusion and localization is accomplished using the robot_localization ROS package. The architectural breakdown can be seen in Figure 6.2 below.

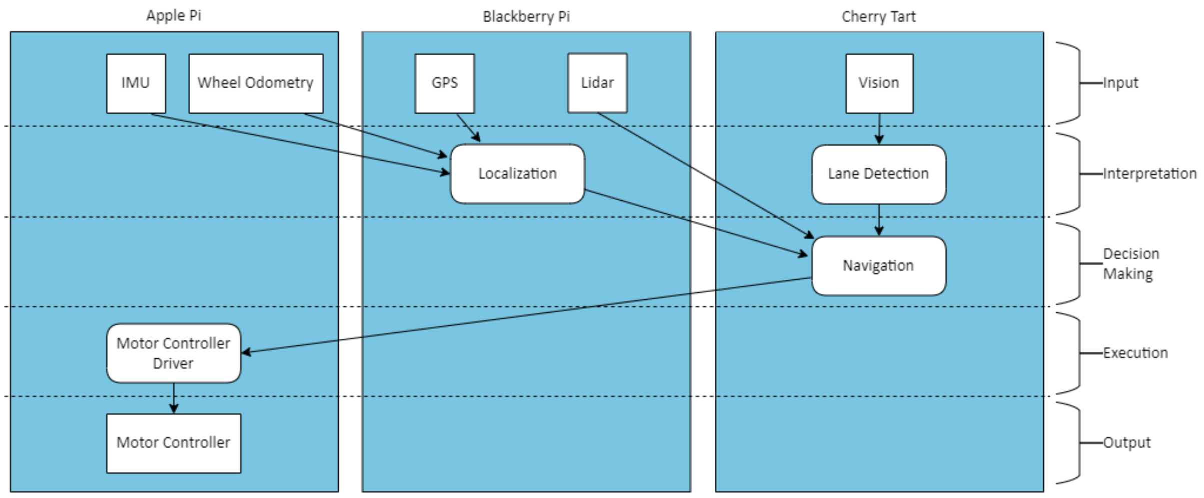


Figure 6.2. Software Architecture Design.

6.3. Obstacle Detection and Avoidance

As seen in Figure 6.1, most of Obed's navigation software was developed on top of the ROS Navigation 2 package, which provides a configurable navigation stack that works from sensor stream to velocity command. Raw obstacle detection capabilities are provided by the Hokoyu LIDAR, and published by the manufacturer-provided ROS driver. Lanes are detected by a custom driver on Cherry Tart and are also treated as obstacles. The sensor streams are combined to create global and local costmaps. The local costmap is a high-resolution grid of costs. The creation of the global costmap is discussed in Section 6.5. Obstacle avoidance is accomplished by the path-following component of Navigation 2, the controller, which takes a path and a local costmap and outputs a motor command that follows the path while avoiding obstacles. Obed uses an open-source controller that implements Regulated Pure Pursuit, which is a modification of Pure Pursuit that avoids obstacles. This is a change from the previous year's robot, which used the DynamicWindow Algorithm, which is usable with more kinematic systems but is more computationally expensive.

6.4. Path Planning

Obed uses a novel algorithm for path-planning, the Right-Left Algorithm. The Right-Left algorithm uses a polygon-based representation of the world instead of grid-based or sampling-based. At a high-level, the algorithm works by drawing a hyper-optimal straight line between the start and goal and then moves toward feasibility by "wrapping" itself clockwise and counterclockwise around the first polygon that intersects the segment currently under review. By recursively applying this algorithm, the robot arrives at an optimal series of right-left choices around obstacles, which translates into an optimal path. There are several implementation details beyond the scope of this report that will be published in a separate paper; however, an abbreviated description is provided here.

At a lower level, the pathfinding algorithm works by building a Directed Acyclic Graph (DAG) with the start point as the unique source and the goal point as the unique sink. At each step of the algorithm, if the segment between the current point and the lowest-costing child has not been checked for intersecting obstacles, it is checked for obstacles. If there are intersecting obstacles, a clockwise and counterclockwise wrapping is performed around the first intersecting obstacle, such that each angle in the new path is convex. Then, the child under inspection is removed from the parent and placed at the end of the new paths that wrap around the obstacle. Otherwise, if the child has no obstacles, or has already been checked, the algorithm is applied to the child. The details absent from this abbreviated description are additional steps that handle concavity in the path and obstacles as well as corner cases in certain starting positions.

Figure 6.3 illustrates the DAG that results from the Right-Left algorithm being run on a space with two obstacles. The green arrows represent the optimal path, the black arrows represent suboptimal paths, and the dotted arrows represent path steps that have not been checked for intersection. Note that, in this example, an alternative is not feasible and is not checked because the shortest path is feasible.

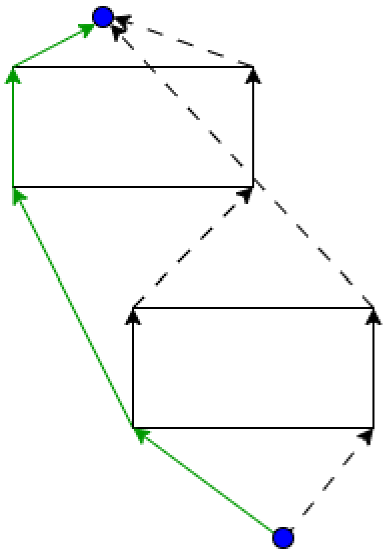


Figure 6.3. Optimal and alternative paths considered by the Right-Left algorithm.

6.5. Global Map Generation

Obed uses the Navigation 2 package to generate a global costmap from Lidar and Camera data. Because the obstacles may have white reflective surfaces, detected lane points beyond an obstacle detected by LIDAR are removed in the point pruning step in Figure 6.4. Since the path-planner in Section 6.3 is polygon-based, the planner converts the costmap into polygons by considering cells above a certain cost to be obstacles, performing an edge tracing algorithm on each cluster of 8-connected cells, and reducing the number of vertices such that the original polygon is contained with the new polygon, and "most" of the original polygon's detail is maintained.

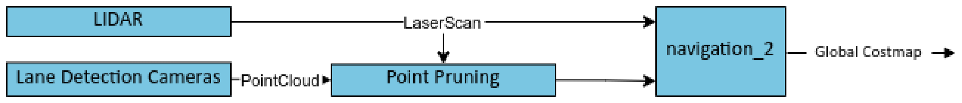


Figure 6.4. Map Generation Flowchart

6.6. Goal Selection and Path Generation

Obed uses a waypoint-based strategy to complete the course. In addition to the points provided by the competition, the team determines the coordinates of corners in the course and sets them as waypoints for the path planning. Path generation is accomplished using the algorithm described in Section 6.4.

7. FAILURE MODES, POINTS, AND RESOLUTIONS

7.1. Vehicle Failure Modes and Resolutions

Table 7.1. 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 the case of tipping despite preventive measures, sense, and E-stop

7.2. Vehicle Failure Points and Resolutions

Table 7.2. 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	Ensure power connectors to each computer in the Computational stack are secure
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 Keep up to date clones of SD cards

7.3. All Failure Prevention Strategy

We considered end states that would cause a robot failure, such as crashing, losing power supply to the computation stack, and having to perform an e-stop. The team then reviewed ways these states could be reached, for instance, a damaged sensor, excess motion, or short-circuiting. This strategy has led to ensuring the safety of Obed.

7.4. Failure Testing

Table 7.3. Testing Plan.

Testing Plan	Expected Result
Unplug the motor controller from Raspberry Pi while running	Motor controller stops the 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

7.5. Vehicle Safety Design Concepts

This section has been consolidated with Section 5.4: Safety Devices.

7.6. Cost Estimate

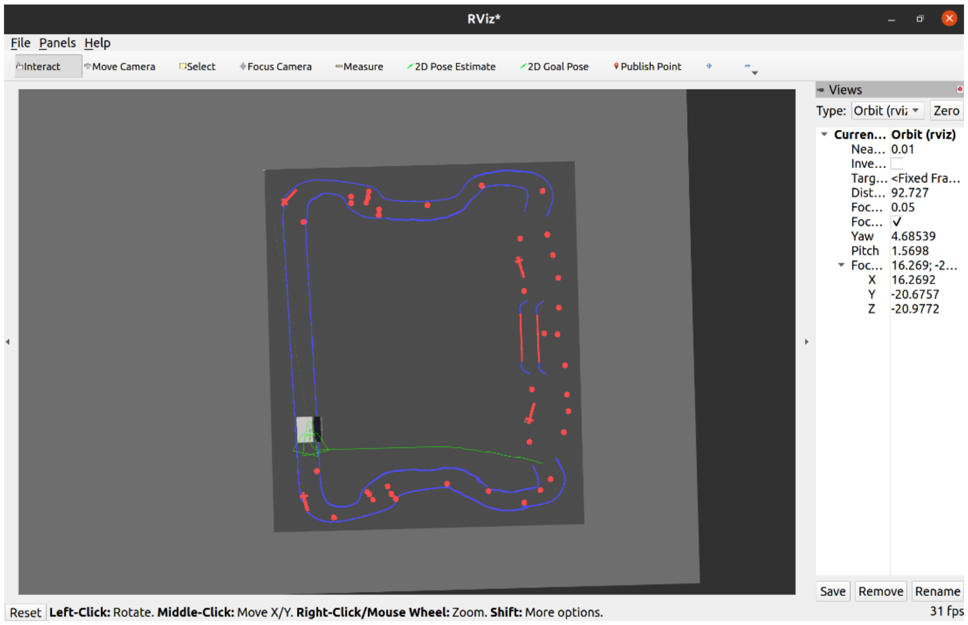
Table 7.4. Itemized Cost Estimate.

ITEM	QUANTITY	COST ($)
Digital Camera	2	115.98
LIDAR	1	4675.00
24-5V DC power converter	1	12.98
IMU	1	34.95
GPS	1	179.00
Blackberry Pi 4s (4GB RAM)	2	110.00
NVIDIA Jetson	1	148.00
Omni Wheel	1	44.86
Front Wheels	2	32.00
Motor Controller	1	184.80
Wiring and Framing	-	375.00
Batteries	4	220.00
165 mm bicycle air shock	1	67.17
Total Cost		$6199.74

8. SIMULATIONS EMPLOYED

8.1. Simulations In Virtual Environment

To simulate our navigation software, we developed navigation simulation software that replaces the motor controller and sensor inputs with simulated inputs with data extracted from an image representing the course, with the rest of the software running as-is. This software allowed us to simulate the path finding and obstacle avoidance systems virtually, as seen in Figure 8.1.



Type: Orbit (rvi: | Zero

Curren…	Orbit (rviz)
Nea…	0.01
Inve…	
Targ…	<Fixed Fra…
Dist…	92.727
Foc…	0.05
Foc…	✓
Yaw	4.68539
Pitch	1.5698
Foc…	16.269; -2…
X	16.2692
Y	-20.6757
Z	-20.9772

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:** Zoom. **Shift:** More options.

Figure 8.1. Rviz view of NavSim simulation

8.2. Theoretical Concepts in Simulations

Obed was modeled in Gazebo as shown in Figure 8.2. 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.

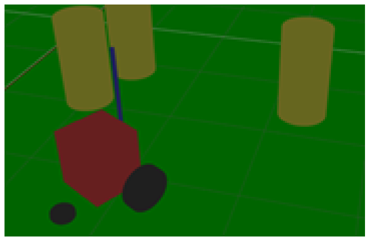


Figure 8.2. Gazebo Simulation of Minimum Turn Radius

9. PERFORMANCE TESTING TO DATE

- The team timed how fast the robot moved 20 feet multiple times. The average max speed the robot achieved was 4.89 mph. Due to the limits of the motor controller, this is the max speed that can be obtained with hardware limiting, while staying under 5 mph.
- The robot has an estimated battery life of 101 minutes and has 2 sets of batteries with a recharge time of 80 minutes, allowing for continuous operation.
- Barrels can be detected from 100 feet. Lines and potholes are detected at a maximum of 7.8 feet in front and 8.4 feet along the sides.

10. INITIAL PERFORMANCE ASSESSMENTS

- Multiple speed tests were carried out, and the average maximum speed for the robot was found to be 4.89 mph, and no speed measurement was above 5 mph
- Obed was able to climb and descend a ramp of 20% grade.