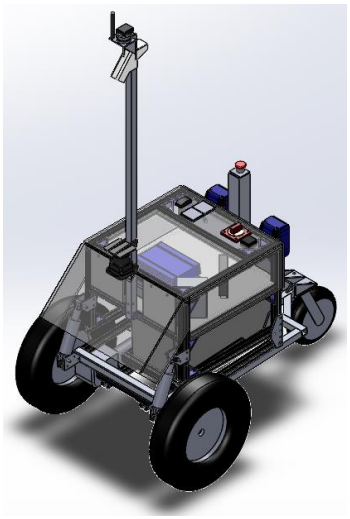
BOB JONES UNIVERSITY
MELCHIZEDEK



Date Submitted: 5/14/2021

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

Team Members:

Peter Labadorf	plaba417@students.bju.edu
Isaac Pinter	ipint279@students.bju.edu
Daniel Zhuang	dzhua133@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: Bill Lovegrove, PhD
Professor
Department of Engineering
Bob Jones University

Faculty Advisor Signature: Bill Lovegrove Date: 5/14/2021

INTRODUCTION

Melchizedek is an upgraded version of BJU's previous IGVC contestant, Lazarus, which competed in 2017. Like Lazarus, Melchizedek relies on a digital camera, Light Detection and Ranging sensor (LIDAR), Inertial measurement unit (IMU), and Global Positioning System (GPS) to traverse the course. Unlike Lazarus, Melchizedek is operated by three Raspberry Pi computers, removing the need for a laptop mounted on the vehicle, without sacrificing the necessary computing power. Melchizedek has all new software. The frame and payload mechanism were also redesigned for Melchizedek in order to give the payload enough clearance when driving across ramps, and to remove the need for tipping the vehicle on its nose when securing the payload. Several other changes were also made to improve overall performance and safety.

ORGANIZATION

The 2021 team was split into three separate subject areas: mechanical, electrical, and software. Each team member worked independently on developing a specific part of the design or software.

Table 1. Student Contributions.

Team Members	Academic Department and Class	Subject Area	Hours
Peter Labadorf	Computer Science, Math, Junior	Software	120
Christopher Zuehlke	Engineering, Junior	Software	80
Daniel Zhuang	Electrical Engineering, Senior	Electrical	120
Isaac Pinter	Engineering, Junior	Mechanical	100
Joshua Heinrich	Engineering, Junior	Mechanical	120

DESIGN ASSUMPTIONS AND PROCESS

This year's team modified the robot Lazarus from a previous Bob Jones University IGVC team. The robot was built for the IGVC Auto-Nav competition. The software would use the Robot Operating System (ROS). The objective of the team was to complete the course in the shortest time possible. Following this objective, problems were analyzed and dealt with using the design process shown in Figure 1 below.

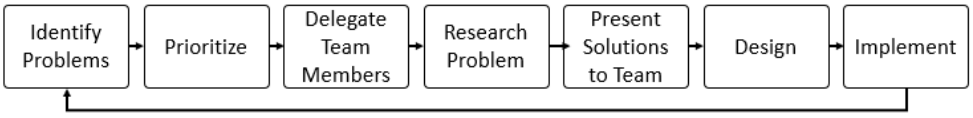


Figure 1. Design Process.

INNOVATIONS

Payload Access

One major problem with Lazarus was the payload mechanism, shown in Figure 2a. To attach the payload, the robot had to be tipped forward to access the payload holder. This previous design

allowed for a very low center of gravity; however, attaching and securing the payload was difficult. Melchizedek's frame is specifically designed to allow for the robot to be loaded by only one person, without the tipping the robot over, and to maintain a low center of gravity, as shown in Figure 2b. See also Figure 8b. The payload structure allows a large tolerance in payload dimensions, while still holding the payload securely. The payload holder is attached to the lower frame of the robot, which lessens the force on the shock absorbers.

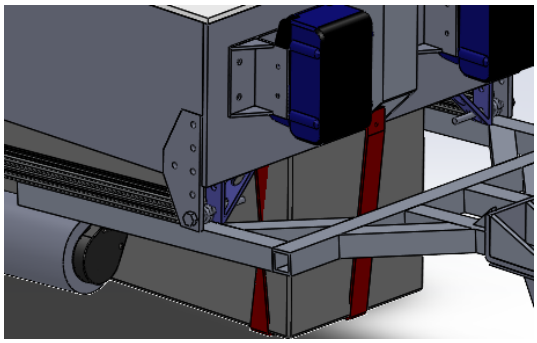


Figure 2a. Lazarus Payload Placement.

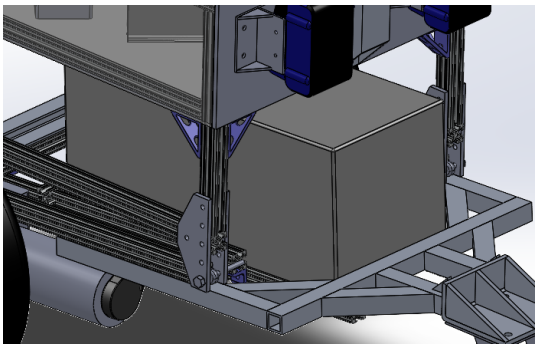


Figure 2b. Melchizedek Payload Placement.

3D Printed Carbon Fiber LIDAR Mount

Several design requirements were considered when designing the LIDAR mount for Melchizedek. The mount was to be strong enough to easily withstand accidental impacts, as the LIDAR is expensive and positioned on the front of the vehicle. The mount was also to be lightweight, to reduce vibration on the mast, and visually appealing. Finally, the mount was to contain an aluminum heat sink to help prevent the unit from overheating. To accomplish these goals, Melchizedek features a reinforced carbon fiber 3D printed LIDAR mount, shown in Figure 3.

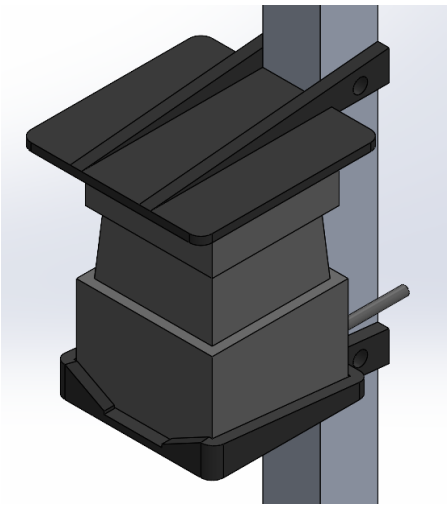


Figure 3. LIDAR Mount Design.

Computational Load Balancing with Raspberry Pi's

Melchizedek uses three networked Raspberry Pi's (the Tri-Pi) for its control system, instead of a single computer. Raspberry Pi's are cheaper, and lighter, and need less power while using a combination of them provides equivalent processing power than a traditional computer.

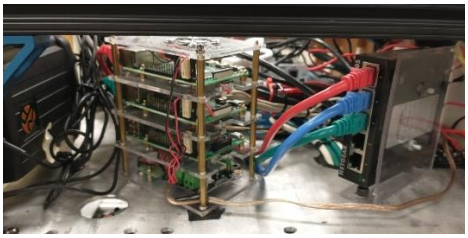


Figure 4. Side view of Tri-Pi.

Spin Detection Software

Using the IMU in combination with the drive encoders allows Melchizedek to calculate the difference between the acceleration that the encoders are reporting and the acceleration that the robot experiences. This allows Melchizedek to automatically detect if it is spun out and take appropriate measures.

Initial Start Position Estimation

Melchizedek employs a novel method to calculate its initial global position more accurately than with a single GPS measurement, without staying in one place to average out several GPS measurements. This is done by a moving weighted average based on odometry information as well as GPS.

Software E-stop and Watchdog

Failure analysis of Melchizedek revealed that safety could be increased by allowing its software to trigger an E-stop. This would allow the robot to E-stop immediately if it tips over or critical sensors fail. Melchizedek also runs software watchdogs so it will be E-stopped if critical nodes, such as the motor controller driver, fail or hang.

Power System

A second power switch was added to the system. To avoid wasting time and battery life by turning the entire vehicle on whenever code was to be deployed, the components were divided into two separate power systems, each with an independent power switch. The software circuit powers the switch, Tri-Pi, and attached USB-powered sensors, while the mechanical circuit powers the motor controller and larger sensors.

Figure 5. Dual Power Switches.

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. Melchizedek's innovative parallel rechargeable Kobalt battery design, shown in Figure 6, allows for the control system to be run from one battery. This allows for the batteries to be changed without turning off the robot.

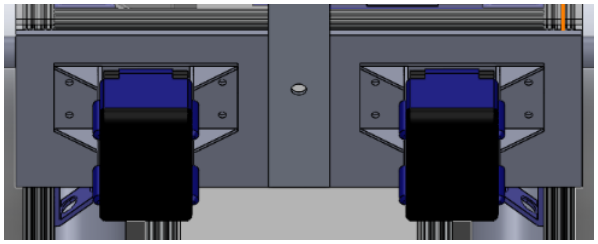


Figure 6. Dual Batteries allow Hot Swapping.

MECHANICAL DESIGN

Overview

Melchizedek's chassis was modified from the Bob Jones University's 2017 IGVC robot, Lazarus. Several flaws were identified in Lazarus' design that are corrected with Melchizedek. Melchizedek features the improved payload mechanism, new LIDAR mount, more space for electronics, and better weather proofing.

Drivetrain

Melchizedek features a differential drive, employing 15-inch wheels, driven by two National Power Chair R81-series motors through a worm gearbox. Each motor features a US Digital E6S encoder to allow the motor to provide feedback to the RobotEQ AX2850 motor controller. The robot uses a single caster wheel, as seen in Figure 7, to provide support with minimal resistance when maneuvering through the course.

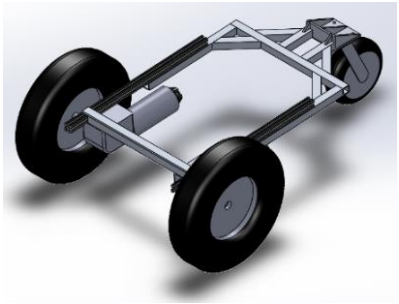


Figure 7. Drivetrain.

Structure and Housing

The frame of Melchizedek is built using 1" T-slot aluminum and 1" box 1/8" wall tubing. T-slot makes up most of the robot, as it allows for modifications in the future and is very strong. Welded box tubing is used to construct the lower frame with some T-slot to allow the lower frame to be

connected to the upper frame. Melchizedek's sensor mast, positioned in the middle of the two wheels as seen in Figure 8a, allows for the lane-detection camera, 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 8a. Frame and Sensor Mast.

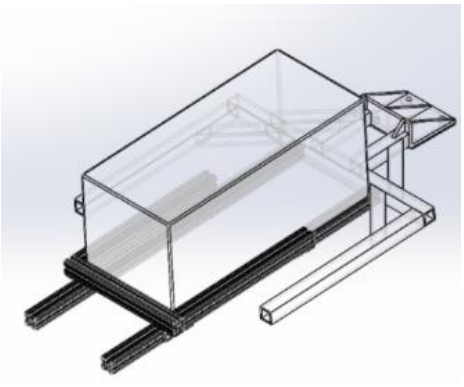


Figure 8b. New Payload Holder.

The payload is held in place using a T-slot attached to the lower frame of the robot. This allows for the payload to be easily and quickly placed into the robot while still maintaining a low center of gravity.

Suspension

Melchizedek uses two FastAce Coil-Over Shock Absorbers, as shown in Figure 9, to support and reduce the force of impacts on the main body.

Figure 9. FastAce Shock on Robot.

The mounting of the TRI-PI and ethernet switch was done with extra care, as the vibration and stress limits of those electronics were unknown. The TRI-PI and ethernet switch were both mounted to the frame using polycarbonate to ensure they were properly retained as seen in Figure 10a and Figure 10b.

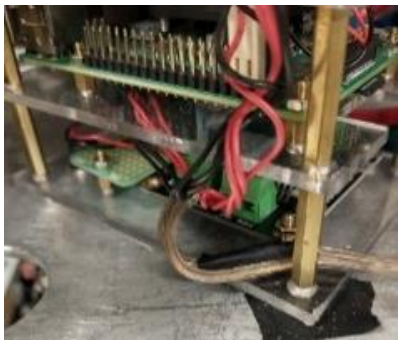


Figure 10a. Raspberry Pi Mount.

Figure 10b. Ethernet Mount.

Weather Proofing

Weather proofing on the previous robot design was an inadequate afterthought. Weather proofing early in the design process would be advantageous and allow for easier construction. Using the Tri-Pi, instead of a bulky laptop, allowed for complete weather proofing. The Raspberry Pi's and other electronics were moved inside the main body of the robot, which is surrounded by polycarbonate, as shown in Figure 11. This design choice ensures that no matter the weather, the robot can still operate properly.

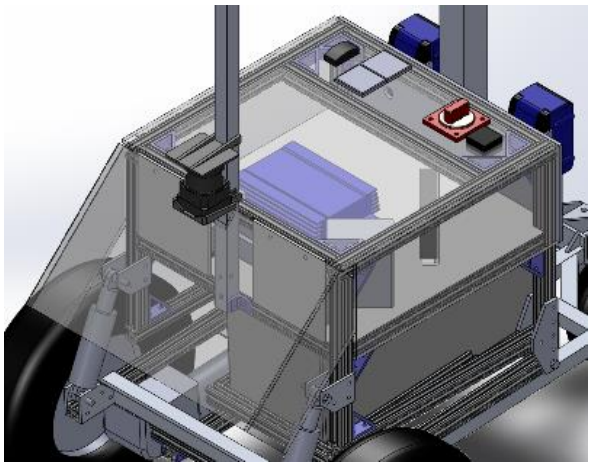


Figure 11. Polycarbonate Covering Electronics.

For sensors that were not able to be placed inside the main body, such as the camera or LIDAR, much care was taken to guarantee limited exposure to unwanted weather. For example, the camera is completely encased inside a 3D printed enclosure (Figure 12a) and the LIDAR is covered by a sun shield which also protects it from rain (Figure 12b).

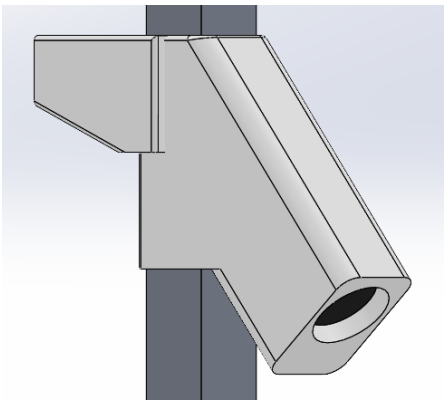


Figure 12a. Camera Mount.

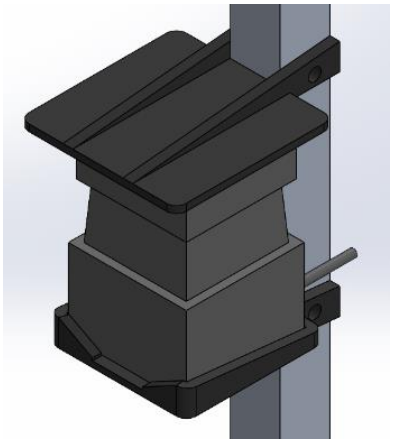


Figure 12b. LIDAR with Sun Shield.

ELECTRICAL DESIGN

Overview

Melchizedek is powered using two 24V Kobalt batteries. These batteries are wired in parallel, so they can be swapped without turning off the robot. Multiple voltage converters are used to provide the correct voltage to the robot's electronic components. As shown in Figure 13, power converters are used to distribute the power at the correct voltage for each component.

Power Distribution System

The robot power system consists of two 24V 5Ah batteries in parallel with an energy capacity of 240Wh. While the vehicle is idle, the robot has an estimated power consumption of 75 W. This calculation was determined using datasheets from the parts and determining the power each part needed. Estimations were made that maximum power consumption should occur when the robot was climbing the ramp at maximum speed. While climbing the ramp, the robot's motor system needs 240W and the robot has a maximum power consumption of 314W. Using equation 1, the power needed to power the motors was calculated. Equation 2 calculates the force needed to move the robot up the ramp at max speed, 5 mph, where Fr = 13lb W = 133 lb $\theta = 8.53°$ r = 0.65ft. Equation 3 simplifies the equation on how to calculate the maximum power. The 2 batteries that plug into the robot enable it to run for 81 minutes. The batteries have a charge time of 1.5 hours, so the recharge rate would be 3.3C, or 3.3A. 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.

$$P = \tau \times \omega \tag{1}$$

$$F = F_r + W \times \sin(\theta) \tag{2}$$

$$P = \tau \times \omega = \tau \times \frac{v}{r} = F \times r \times \frac{v}{r} = Fv \quad (3)$$

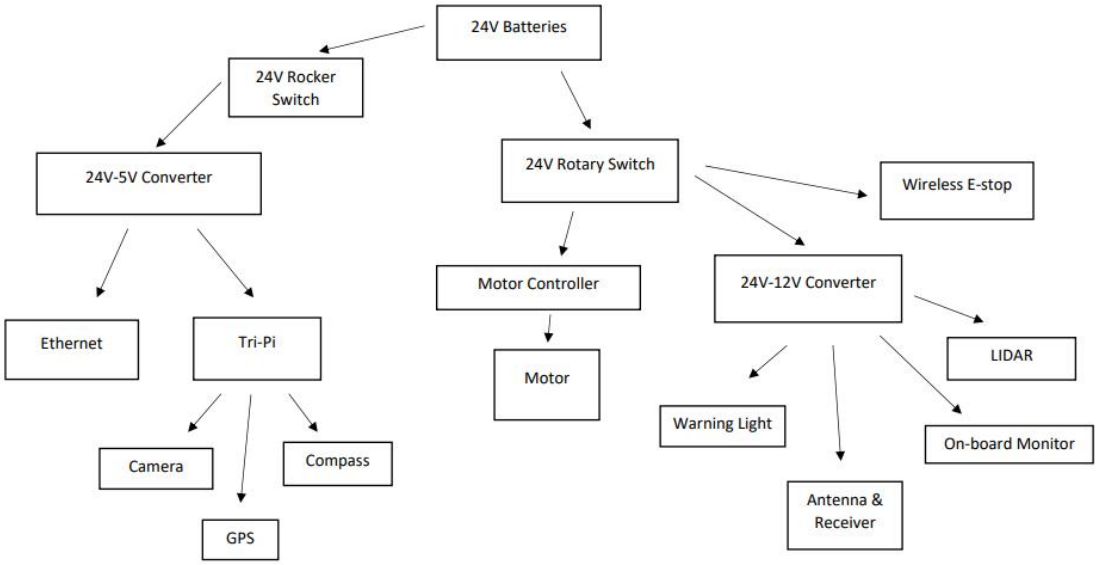


Figure 13. Overview of the Power Distribution.

Electronics Suite Description

Melchizedek's electronics system is controlled by 3 Raspberry Pi 4s. These computers allow feedback control for the robot. To make things simple the 3 Raspberry Pi's were named, from top to bottom, Apple Pi, Blackberry Pi, and Cherry Pi. Figure 14 shows how all the components are connected.

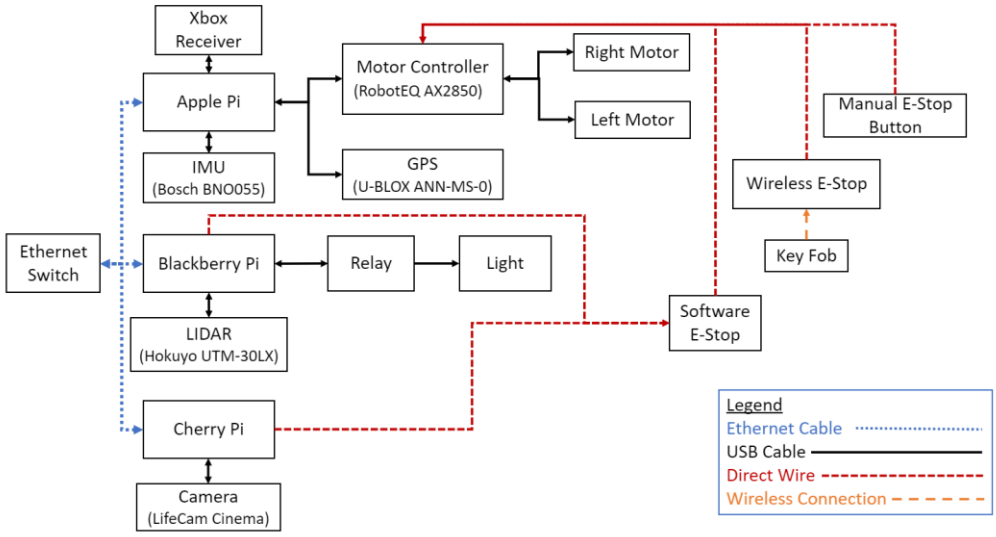


Figure 14. Diagram of Control System.

Safety Devices

Melchizedek has a manual button E-stop and a wireless E-stop. This device allows anyone to be able to shut down the robot immediately in case of an emergency. When either the manual or wireless E-stop is pressed, the circuit sends a signal to the motor controller so it will transition to E-stop mode and immediately turn the motors off.

Another safety device was the software E-stop. This allows safety services running on separate nodes on the Tri-Pi to trigger an E-stop just as if the E-stop button had been pressed.

Fuses were also added in Melchizedek's circuit to make sure that there is not any overcurrent and that its circuit would not be damaged from excess heat.

SOFTWARE STRATEGY AND MAPPING TECHNIQUES

Overview

The software written by the team is distributed across the Tri-Pi. Cherry Pi handles vision and lane map generation, Blackberry Pi handles lidar obstacle detection and mapping, and Apple Pi handles odometry, localization, and motor control.

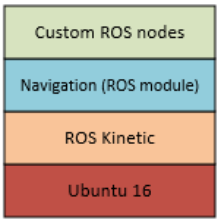


Figure 15. Software Stack.

Software Strategy

The team's custom code runs on top of the prebuilt ROS navigation module, which takes a user supplied goal, LIDAR data, and a map, and generates commands to the motor. The custom ROS nodes interpret and fuse LIDAR and vision data, provide odometry using an Extended Kalman Filter, check for tipping and spinning out, and communicate to the motor controller. Refer to Figure 16 to see how the nodes interact.

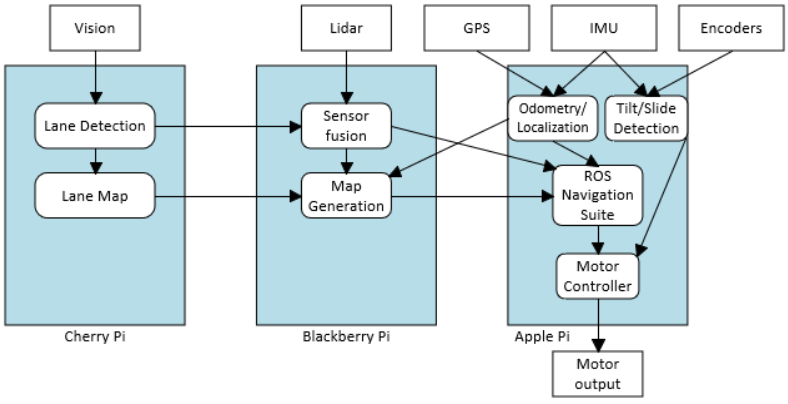


Figure 16. Software Architecture Design.

Obstacle Detection and Avoidance

Obstacle detection and avoidance is done with a combination of camera and LIDAR data. The camera detects lanes and potholes, which are then fused with the point cloud representing the physical obstacles produced by the LIDAR, which is fed into the navigation algorithm.

Map Generation

Our software produces two maps, one of which just contains the lane lines and is assumed to be the same between matches. This map allows us to start the match with a global map capable of producing a global path at the start of the run, which is modified during runtime by the second map, generated from LIDAR data.

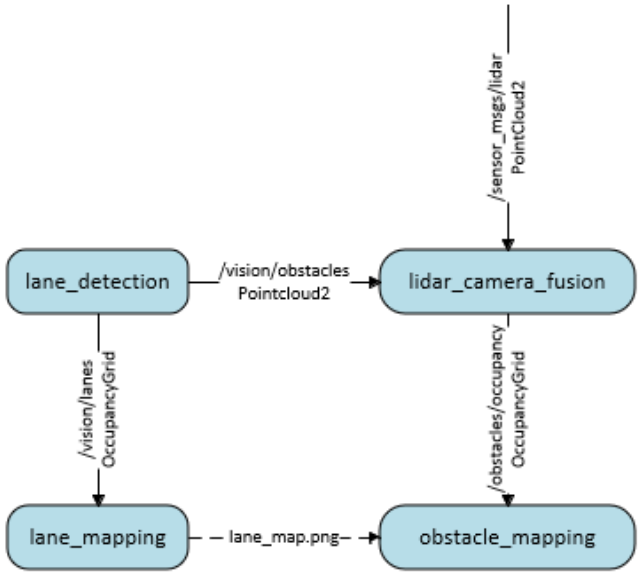


Figure 17. Sensor Processing Flow.

Goal Selection and Path Generation

The main problem encountered with path selection was reconciling the locally accurate but globally inaccurate odometry produced by an Extended Kalman Filter on the encoder and LIDAR odometries with the local inaccuracy of the GPS unit. Specifically, GPS coordinates of a previous position needed to be estimated to a higher degree of accuracy then the GPS can detect. This needed to be done so that more accurate knowledge could be obtained of whether the robot had arrived at its goal. The team developed a novel technique for averaging GPS measurements without parking the robot.

Given x_i is the unknown true position at time i, odometry measurements $O_0 = \vec{0}, O_i \approx x_i - x_0$ with accuracy α^i, and GPS measurements $G_i \approx x_i$ with accuracy β, it is stated, without proof, if $\alpha > \beta$ an estimate of x_0 with accuracy greater than β is

$$\hat{x}_0 = \frac{(1-\alpha)\sum_i^{\log_\alpha \beta} \alpha^i (G_i - O_i)}{1-\alpha^{\lfloor \log_\alpha \beta \rfloor}} \tag{5}$$

This equation is the weighted average of $G_i - O_i$ where each term has the weight α^i, summing only $\log_\alpha \beta$ terms, because if $i > \log_\alpha \beta$, $\alpha^i > \beta$ and the GPS measurement is more accurate than

the odometry. The denominator of the average was simplified because it is a geometric series. The accuracy of a measurement is equal to

$$e^{-\frac{d}{c}} \tag{6}$$

where d is the average distance of the measurement from the true value, and c is an arbitrary constant distance.

α and β are numbers that can be calculated from data sheets and tested in the field, so the team believes this algorithm will be useful in robot-to-world localization needed to accurately meet way-points.

Additional Creative Concepts

In previous years, the robot's wheels would spin, incurring a penalty for tearing up the grass. This year, before the decision to have the course on asphalt, anti-spin software was included, which compares the encoder with acceleration data from the IMU, and thus if the encoder outputs indicate acceleration while the IMU has no acceleration, then the robot knows that it is in a spin state and can stop and try to get out of the state.

Closely related to this, the team decided to include the ability for 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, or if a critical node does not reset the watchdog. 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.

Another concept the team developed was a custom deploy script to facilitate rapid development. This was implemented by hosting a git server on Apple Pi with a post-receive hook to trigger the other two RPi's to pull their code. This allows developers to, with one command, deploy code to all the RPi's, drastically reducing the time needed test an update.

FAILURE MODES, POINTS, AND RESOLUTIONS

Vehicle Failure Modes and Resolutions

Table 3. Failure Modes.

Failure Modes	Resolutions
Motor controller could disconnect while running	Enable watchdog in motor controller and implement in software
Sensor failure	Pause the run and wait for a reconnect.
Tipping over due to high acceleration	Preventive tipping by limiting yaw rate at high speeds and max forward acceleration
	In case of tipping over 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 poly-carbonate mounts to reduce constant vibrations on smaller electronics
3D printed parts have the possibility of breaking	Important parts are carbon fiber to maximize strength
Sensor mast could break or bend	Reduce the chance for robot could tip in software, polycarbonate front plate absorbs most of the impact in case of crash
Raspberry Pi microSD's could be-come 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 stop. The team then reviewed possible ways these states could be reached, for instance, a damaged sensor, a short circuiting, or loss of connection to the motor controller. With these possible failures in mind, the team proposed ways to prevent the failures, for example, having redundances for each sensor, adding correctly sized fuses, and enabling the motor controller watch-dog. This strategy has led to several design changes to increase the safety of Melchizedek.

Failure Testing

Table 5. Testing Plan.

Testing Plan	Expected Result
Unplug motor controller from Raspberry Pi	Watchdog shuts off motors
Simulate robot in Gazebo to test tipping over	Find the max velocity and yaw the ro-bot can withstand without tipping
Simulate loss of sensor data in Gazebo	Depending on the sensor loss, robot ei-ther self-stops or continues driving

SIMULATIONS EMPLOYED

Melchizedek was modeled in Gazebo, as seen in Figure 18. Gazebo is used to simulate the robot and test new code to ensure there are no major bugs before testing it on the physical robot.

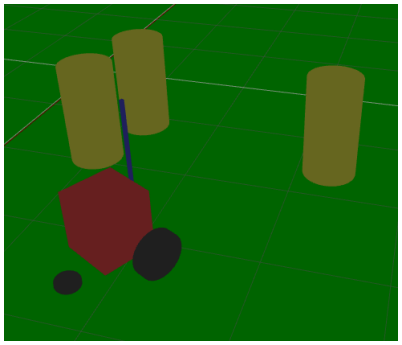


Figure 18. Gazebo Simulation of Melchizedek

PERFORMANCE TESTING TO DATE

- The team timed how fast the robot moved 30 feet multiple times. The fastest time the robot achieved was 4.73 seconds which results in a speed of 6.34 ft/s or 4.32 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 81 minutes. In practice, the robot only ran for 48 minutes. This reduction in battery life resulted from the lack of current supplied to the Tri-Pi from the batteries.
- Barrels can be detected from 50 feet. Lines and potholes are detected at a maximum of 7.5 feet.

INITIAL PERFORMANCE ASSESSMENTS

The team was only able to complete simple checks on the Melchizedek's driving ability. We discovered that it drives with a noticeable drift to the right, but other than that, the drivetrain is functional. Initial testing of the camera shows that the white lines can be accurately detected using a simple HSV filter.

CONCULUSION

Melchizedek improves on many of the flaws of Lazarus. Using a digital camera, LIDAR, IMU, and GPS Melchizedek traverses the course with ease. At the time of writing, the software is still being implemented. With the upgraded frame and payload mechanism, Melchizedek's payload rack securely holds the payload, without the need to tip the robot over. Along with the numerous safety additions, Melchizedek is on track to compete successfully at IGVC 2021.