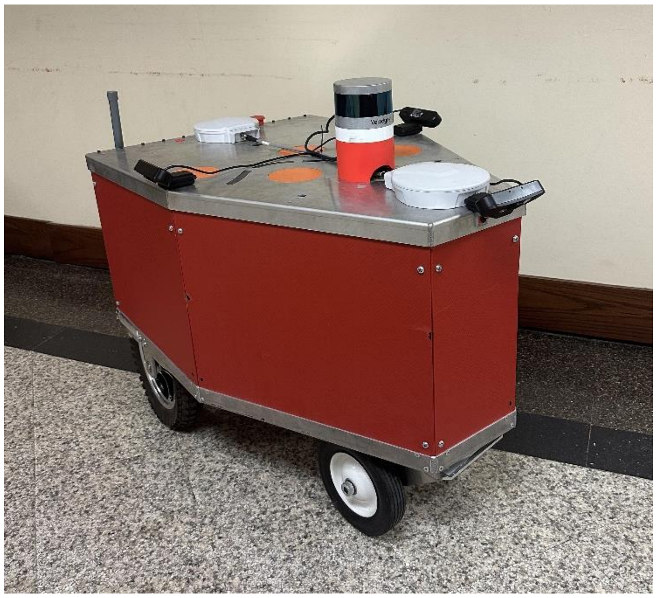
Dokalman MK2.6

University of Cincinnati Robotics Team

31st Annual Intelligent Ground Vehicle Competition

Andrew Gerstenslager ✦ Michael Mcadams ✦ Colin Campbell ✦ Sam Moeller ✦ Phu Le✦ Brian Drobnak ✦ Caden Cruset ✦ Christian Graber ✦ Dom Deliduka

Contact information provided in the "Team Organization" section


Dokalman MK2.6

(companion) Base Station

CERTIFICATION:

I certify that the engineering design in the vehicle Dokalman (original and changes) by the current student team identified in this Design Report has been significant and equivalent to what might be awarded credit in a senior design course.

Janet Dong

Doctor Janet Dong, Advisor

Submitted 5/16/24

Table of Contents

Introduction

After a four-year hiatus, the University of Cincinnati Robotics Team competed in the 2023 Intelligent Ground Vehicle Competition (IGVC). Although we did not qualify for the autonomous component of the auto-nav challenge, we have since made significant strides in improving our robot, Dokalman MK2.6 following close behind from its 2.5 version.

Following the 2023 competition, our priority has been to repair many components damaged during travel and enhance the simulation and navigation systems. Key upgrades include improving the motor controller software, cleaning and documenting electrical wiring, and replacing our camera systems, all aimed at increasing safety and reliability. As of this report, Dokalman's hardware and sensors are fully operational, and all sensor systems are functional. The UC Robotics Team is now focused on finalizing the navigation software in time for the 2024 competition.

Team Organization

Role	Name	Major	Email*
President	Colin Campbell	Computer Science	campb3c7@mail.uc.edu
Vice President	Andrew Gerstenslager	Computer Engineering	gerstead@mail.uc.edu
Treasurer	Michael McAdams	Electrical Engineering	mcadammt@mail.uc.edu
Secretary	Christian Graber	Computer Engineering	grabercn@mail.uc.edu
Member	Phu Le	Computer Engineering	lepq@mail.uc.edu
Member	Sam Moeller	Electrical Engineering	moellesu@mail.uc.edu
Member	Caden Cruset	Computer Engineering	crusetcs@mail.uc.edu
Member	Dom Deliduka	Electrical Engineering	delidudp@mail.uc.edu
Member	Brian Drobnak	Mechanical Engineering	drobnabn@mail.uc.edu

Design Process/Assumptions

The Dokalman MK2.6, maintaining the re-engineered frame and custom gearboxes of its predecessor, optimizes space for integral components while meeting competition size limits. The design assumptions for our robot's success are based on several key factors. Environmentally, the robot must handle sloped and rough terrain, necessitating a rugged design that withstands various stresses, and operate in adverse weather conditions, including rain and varying light levels. Our robot's chassis has limited openings that expose the components to environmental factors and is sealed, protecting the critical computers inside from water damage.

Operationally, the robot must perform tasks safely and reliably, with systems to detect and stop for hardware or software issues. It must maintain an average speed of at least 1 mph over a challenge run, travel above 1 mph over a 44-foot stretch at the beginning to avoid disqualification, and be governed to not exceed 5 mph. To prevent speeds over the limits, the robot contains a software limit in the motor controller to prevent speeds while under autonomous mode. If needed the robot's speed could be tracked by requesting data from either the GPS, IMU, or the motor encoders.

For user interaction, the robot's operation must be easily stoppable through mechanical and wireless E-stop methods. Dokalman displays a red mechanical E-stop button, which is over an inch in diameter, and located at the center rear of the vehicle. The vehicle also has a wireless E-stop which is effective at ranges over 100 feet, and both bring the vehicle to a quick stop. Additionally, the robot must be user-friendly with clear indicators of its mode of operation and status, indicated a solid indicator light that changes color based on mode or waiting state that switches to flashing in autonomous mode. Additional safety features will be outlined in later sections.

In terms of design, the robot's chassis conforms to dimensions of three to seven feet in length, two to four feet in width, and up to six feet in height, excluding the E-stop antenna. The robot's design at its largest has the dimensions of 27” wide, 48” long, and 32” tall. The robot boasts a removable 100AH battery and optional AC-to-DC power ensuring long operation times as well as low downtime for maintenance. Dokalman also has a specially designed holder for a 20-pound payload with dimensions of 18 inches long, 8 inches wide, and 8 inches high which sits right behind the wheels which helps with traction.

Maintenance and documentation are essential for new and existing members, requiring a design for easy disassembly, clear component labeling, and comprehensive documentation. The UC Robotics Team has taken this practice to heart meticulously documenting many aspects of the robot. We have created diagrams outlining connections for the electrical power systems and network. We additionally have in our secure documentation the information of every device on the network, their purpose for the robot, and how to access each device. Additionally, the software is documented in the code and the structure of each package is kept as close to a standard as possible.

For qualification, the robot must meet specified dimensions, verify the functionality of the mechanical and wireless E-stops, safety light, and speed constraints, and demonstrate lane following, obstacle avoidance, and waypoint navigation capabilities. We have ensured all the requirements as of this report writing except for the navigation capabilities which we plan to finalize.

System Architecture

Mechanical:

Our mechanical system is designed to be rugged and durable. The wheels are equipped with a gearbox each with an overall ratio of 30.38:1. These gearboxes ensure better control over the robot's movement and help to create a more fluid driving experience. The gearboxes increase the available torque to the wheels which ensures that the vehicle has the appropriate power to drive over any terrain. There is no suspension, but from past experiences from the IGVC and knowing the physical limits of the gearbox, there is more than sufficient stability for navigation in harsher environments.

Electrical and Network:

The Dokalman MK2.6 can be powered either by a 12VDC, 100AH battery for driving operations or by a 120VAC to 12VDC converter when driving is not required. Several onboard DC-to-DC power converters provide the different voltages needed by the electrical components.

Dokalman MK2.6 houses several computers. The NUC serves as the primary computer, handling most computing and navigation tasks. A Jetson Xavier manages image processing, while a Raspberry Pi oversees serial communications between ROS2 and the Arduino Mega. The Arduino Mega controls the driving code, lights, GPS, and IMU. Additionally, the Raspberry Pi requests encoder information from the Arduino for navigation nodes in ROS2. The Arduino Mega, communicates to a pair of devices created by Dimension Engineering which are the Kangaroo Motion Controller and the Sabertooth 2x60. These Kangaroo receives motor commands and executes them through the Sabertooth or can also receive commands that request the robot's current speed and encoder state.

Many devices are connected on a central network. This network is the main transportation method for data that is shared across all devices to publish and subscribe to different types of data via ROS2. This network is shared with the companion base station which allows remote access and internet into the robot. This allows for a much smoother development process as the devices on the robot are accessed without the need to introduce additional cables or the need to plug into the robot as it is deployed in the field. Only a proximity to the base station is required as the base station can transmit data for 100s of feet with no issue.

Safety Devices:

The robot's operation is governed by multiple safety devices and systems. There are two ways that power can be physically blocked to the motors to stop the robot. A prominent feature is the large red emergency stop (e-stop) button on the roof panel, which cuts all power to the motors when pressed. Additionally, a remote-controlled relay on the same circuit can break the power connection to the motors.

Several software systems also ensure the robot's safety. The Arduino controlling the motors has a few systems in place to protect the device and the operators near the robot. The Arduino monitors the connection to the RC receiver and halts the robot if the controller disconnects. Furthermore, if the RC receiver is connected and set to a specific mode, the robot is prevented from driving by the Arduino sending repeated stop commands. In addition, a ROS2 topic has been implemented to stop the robot whenever any device on the network sets the topic to False which the Pi informs the Arduino to exit autonomous mode and send stop commands. This allows the flexibility for many systems to stop the robot when different situations call for it.

Software Modules:

There are a few primary software modules in the system. These software packages are modular and easily replaceable due to the modular nature of ROS2. For sensing, there are modules for retrieving raw data from the motor encoders, LiDAR, GPS, IMU, and 3 cameras. The camera data is then passed through our custom image pipeline and converted to a point cloud for navigation and localization.

The robot also has a motor module hosted on the Pi and the Arduino to control the motors and LEDs. This module is responsible for interfacing ROS2 to the Arduino while also allowing optional remote control. Additionally, the Arduino manages the robot's indicator lights which display the status of the robot. The main control modes of the robot are managed by the RC controller and these modes determine which input the Arduino should respond to as motor control.

For simulation, we have a software module designed to simulate Dokalman MK2.6. This simulated robot is 3D modeled to have the same geometry as the real-life robot. Additionally, there are simulated sensors that match the design of the sensors on the robot. This is all simulated in Gazebo and the status of the system can be analyzed utilizing the provided ROS2 tools such as RQT and Rviz2.

We are still working on the localization and navigation module for the robot. Currently we have prototype waypoint followers and SLAM map generation working. We are in the process of combining these systems into one robust and cohesive navigation stack utilizing NAV2.

Interactions in vehicle:

All components are powered by the battery or AC to DC converter with the addition of the onboard DC to DC converters that provide required specific voltages. The sensors and computers interact through serial communication and the network with ROS2 to receive and transmit data between various nodes to perform tasks such as image processing and environment mapping. The nodes, after creating a map of the local environment using the sensor data, use ROS2 hardware interfaces and plugins to communicate from the Raspberry Pi to the Arduino Mega which drive commands need to be performed. The Arduino Mega then communicates this information directly to the Kangaroo Motion Controller and 2x60 Sabertooth to drive the motors. The safety mechanisms in place ensure that this is all completed in a safe and stoppable manner.

Key Innovations

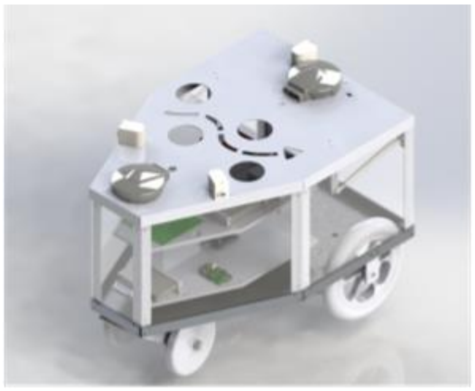
Our robot's vision system is our newest innovation as ROS2 allows us to connect as many cameras as we would like by having reusable ROS2 image pipelines. Currently, we have 3 cameras on the robot. Each camera is calibrated to the ground plane, which allows the raw feed to be skewed to generate an accurate top-down view of the ground, then the images are converted to point clouds and then combined which generates over 270 degrees of ground vision around the robot in over a 10 ft radius. This amount of vision provides the robot with a large ground view that can see any obstacle around the robot. This development is inspired by cars that provide a similar view to drivers as they back up so they can see the ground around them on a screen. Our addition to this technology that we are combining the pointcloud comprised of multiple camera views with a LiDAR's point cloud. This allows barrels and lines to aid the other data type where there are overlaps in the data.

Additionally, Dokalman does not have a sensor pole. Our team believes that a tall sensor pole is not feasible for a production ready robot and could be easily damaged. This means that the robot can stay compact and if taken into production, all the components could be safely secured and installed in the robot without fear of theft.

One of the key innovations for our robot is its companion base station, an auxiliary robot that serves as a network connection hub. This base station is equipped with a Wi-Fi access point, allowing wireless devices such as laptops to connect to the network. This network is then shared between two paired M2 Bullet devices, which act as a bridge between the robot and the wireless access point utilizing radio communication. This setup enables remote access to devices on the robot and provides network access to computers on the Wi-Fi network and all computers on the robot. The network can connect to an existing network via Ethernet from another router/wall or use its onboard 5G router to link all devices to cellular data. The base station is innovative in the fact that the robot can be remotely accessed and controlled through its network for development purposes. Remoting into the robot to develop code, updating packages, or connect to the central code repository becomes trivial.

Mechanical Design

Overview



The hardware configuration of Dokalman MK2.6 remains unchanged from 2023, maintaining the same components and design elements as in the previous version, with the exception of an additional camera.

Dokalman's elements consist of four main sections: the drive platform; frame; hinged top panel; and the electronic panels. The drive platform consists of the bottom tray, two gearboxes, and the front caster wheel assembly. The top panel acts as a weatherproofing shield around the drive platform and electronics panels while also allowing quick access to the robot's interior through an electronic latch. Mounted on the top panel are three wide-angle cameras, a LiDAR sensor, and two GPS antennas. The electronics panels include two MDF panels for power management, networking, computers, and an IMU sensor. The status lights are mounted to the lidar assembly of the robot.

Frame Structure, Housing, and Structure Design

The drive platform consists of two panels which act as a lid/rim on top and a pan on the bottom. Both of the panels were laser cut and bent to shape. The framing of the robot is created from laser cut aluminum struts attached by rivets to the pan and rim. The top panel was then attached to the top rim by a rubber hinge. The all-aluminum construction of the frame allows for a robust and lightweight frame. Acrylic panels line the outside and are secured with screws to the frame. The top panel is hinged and locked down with an internal latch that allows access to the electronics while also being secure from unwanted access. To account for heat in a closed system, two fans located at the back of the robot act as an air intake and exhaust to help circulate fresh air through the system.

Suspension

With the rugged design of the gearboxes and robot assembly, we do not have a suspension system on the robot. Because of this we do remove a failure point on the robot. As the competition is held on pavement this year as well the terrain does not warrant the need for increased suspension. Though adding suspension could help with the accuracy of the camera line detection systems and LiDAR readings, we think any negative effect is marginal.

Weather Proofing

The Dokalman MK2.6 is capable of functioning in almost all types of adverse weather scenarios, barring the most extreme. The minimal openings in the robot’s enclosure protect the robot from rain and dust entering the inside of the robot. Additionally, the electronics are raised on panels inside the robot meaning that in the rare case that water enters the enclosure, the water will not have any opportunity to contact any components.

The top circumference of the robot features a sealed top panel, achieved by compressing a rubber weatherstrip, creating a secure seal when the panel is shut. Instead of a conventional hinge that may allow water seepage through crevices, the panel's hinge is constructed from solid plastic for enhanced protection.

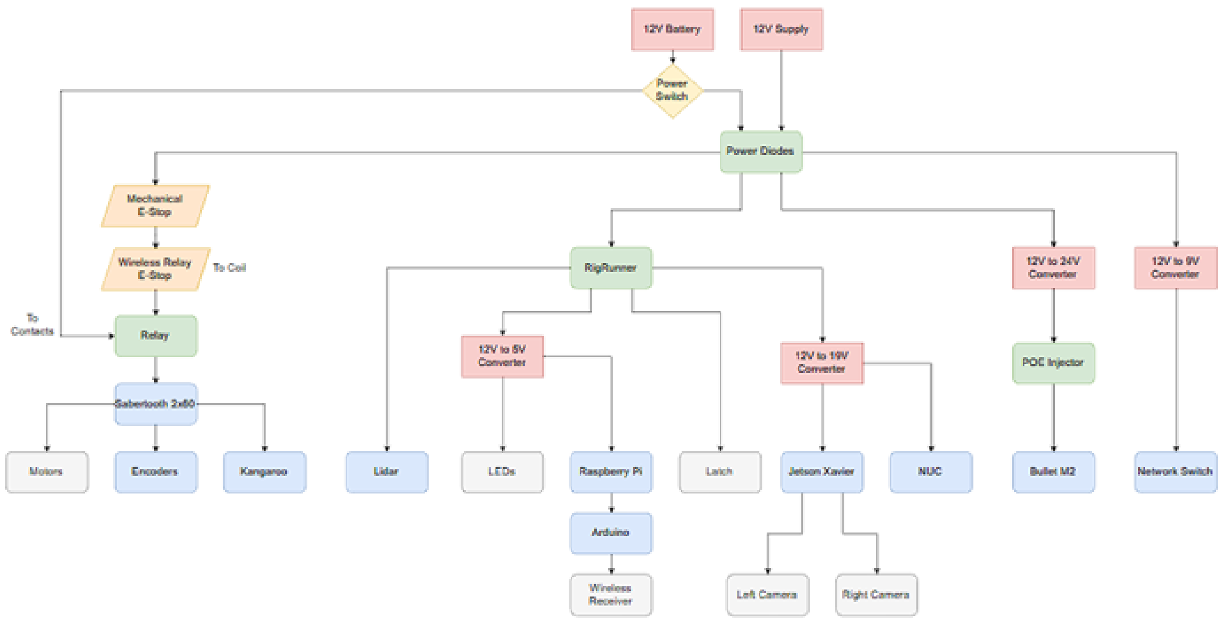
The top and bottom panels are bent and welded seamlessly to prevent the penetration of water or dust. Any apertures present in these panels are equipped with a grommet or sealed in other ways.

For added security, the robot's interior electrical panels are elevated from the base panel. This design ensures that in the event of any water intrusion, the liquid would accumulate at the bottom, thus minimizing the potential damage to the electronic components.

Electronic and Power Design

Overview

Dokalman MK2.6 is powered by a 12VDC car battery when operating the motors or an AC-to-DC converter when under maintenance. It is designed with a fuse, several relays, and several emergency stop components in case there is a fault. DC-to-DC converters are used throughout to provide the other required voltages for some of the components and a Rigrunner is used to regulate the 12VDC to the more vital and sensitive electronics. The central electronics suite vital to the operation of Dokalman MK2.6 consists of onboard sensors, 3 computers, and the motor drive electronics. The sensors and computers can always remain in operation while the motor drive electronics are enabled through several emergency stop devices.



Power Distribution System

Dokalman MK2.6's primary power source is a single 12V 100AH deep-cycle lithium iron phosphate battery that provides 3.3 hours of driving and recharges in 8 hours. Power for the robot is controlled by a 120A resettable fuse which will trip in the case of a short or other high current event. Off this fuse, the 2x60 Sabertooth motor controller power is gated by a 120A relay controlled by e-stop circuitry. The e-stop circuitry consists of a 120A relay controlled by the large e-stop button on the top of the robot as well as an additional remote relay. Auxiliary power for the robot for sensors, computers, and other electronics are first passed through a networked RIGrunner, then either regulated or boosted to the required voltage. Network components, such as the M2 Bullet and network switch, and components used for driving the motors do not receive power through the Rigrunner.

Electronics Suite

The central electronics suite for Dokalman MK2.6 can be separated into several subcategories: motor drive electronics, computer network, and sensors. Other electronics that are not part of the main function of the robot consist of the other network electronics that allow Dokalman MK2.6 to be accessed and updated remotely. Most of the robot consists of either individual wires for short connections to nearby components or shielded cabling for sensors and components further apart.

Motor drive electronics incorporate 4 12VDC motors (2 for each wheel), a 2x60 Sabertooth, a Kangaroo x2 Motion Controller, an Arduino Mega, mode LEDs, and an RC receiver with its controller. The 2x60 Sabertooth and Kangaroo x2 Motion Controller allow for better control and tuning of the robot while also allowing the user or ROS2 network to get the speed and position of the motors through UART serial communication. The Arduino Mega is used with the RC receiver and controller for manual control of the robot, setting the modes, and ensuring the LEDs reflect the current condition.

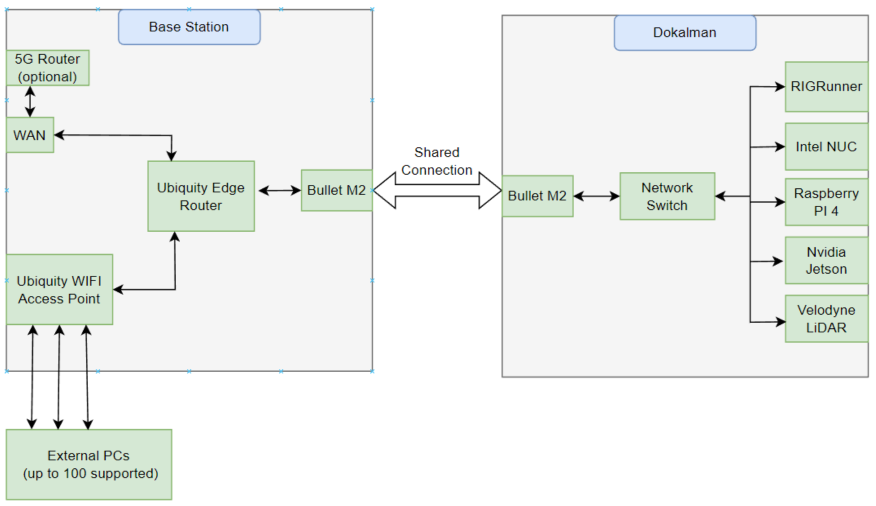
The computer network is one of the most vital aspects of the robot and consists of 3 computers: the Jetson Xavier, NUC, and a Raspberry Pi. Each computer handles a specified series of tasks for the function of the robot and communicates them to each other and other components through either UART serial communication or ROS2 nodes. The Software System section of this report goes over these tasks in more details.

Dokalman MK2.6 features several sensors used for navigation. Two GPS receivers are used to obtain the location and heading of the robot which is compared with readings from an IMU and 2 motor encoders to ensure accuracy over time and distance. For obstacle navigation, the robot uses 3 cameras, located on the front left, front, and front right of the top cover, and a 2D LiDAR located on the center of the top cover. The cameras with the Jetson Xavier are used find the lines and simulated potholes for the navigation to avoid. LiDAR is used primarily to avoid barrels and any other obstacles that are taller than Dokalman MK2.6.

Network

Other electronics not vital to driving and navigation consist of the M2 Bullet antenna and the network switch on the robot and the base station. The base station electronics consist of a WiFi access point, 5G router, a WAN connection port, and an M2 Bullet antenna. These allow connection to the base station that extends the network permitting internet and remote access to work on software and view sensor readings and environment mapping. Connections between the base station and Dokalman MK2.6 are shown in the figure below.

Network Configuration



DISCLAIMER: THE BASE STATION WILL BE USED FOR MAINTENANCE AND VIEWING THE READINGS ONLY. NO COMPUTATIONS OR COMMANDS WILL BE SENT TO DOKALMAN MK2.6 AT ANY POINT DURING DRIVING.

Emergency Stops

There are several emergency stops in place to prevent the robot from running off. The top cover has a red emergency button near the back that can be pushed to cut power to the motors. Similarly, there is a separate E-stop remote that is wired in series to the button to do the same thing. Alternatively, the robot could be placed into stop mode (shown by red LEDs) by the RC controller, or the controller could just simply be turned off. There is also a software stop implemented as a ROS2 node when there is a communication failure from one of the components on the network.

Software System

Overview

All the code on the robot is managed and connected through the Robot Operating System 2 (ROS2) framework. This allows us to modularize different software components as well as utilizing premade packages for some of our hardware by running code inside of nodes. These nodes communicate with each other via ROS communication primitives, consisting of publish-subscribe (ROS topics) and remote procedure calls (ROS services). These nodes can communicate with each other as long as they are running on the same network with the same domain id. The devices running such ROS2 nodes are the Intel NUC, NVIDIA Jetson Xavier, and the Raspberry Pi. There is also firmware running on an Arduino that controls the motors and LEDs.

Currently, we have the vision data processed into a usable format (described in detail below). We plan to integrate the GPS as the ground truth updates for location but will use dead reckoning between GPS updates to localize the robot. This will be calculated with a Kalman filter using the encoders and imu.

The navigation is still in its prototype phase. Dokalman has proven success in simulation with basic waypoint following, and we plan to finalize the navigation systems this coming week.

Motor Control, Operating Modes, and Sensors

The Arduino communicates with both the Raspberry Pi and the Kangaroo motor controller. It also receives data from an RC receiver and the encoders. The Arduino detects when the Kangaroo is powered on by sensing the voltage from the device and then waits for the controller to bind. To calibrate the ranges and dead zones on the controller, the user moves the joysticks to their maximum range in all directions and then the Arduino will proceed into the next mode of operation.

The Arduino's main loop is dependent on the state of a three-state toggle switch on the RC controller, which includes the following modes: STOPPED, SELF DRIVE, and RC CONTROL. In the STOPPED mode, the Arduino continuously sends stop commands to the motor controller to ensure the robot remains stationary. In RC CONTROL mode, the Arduino reads the joystick signals from the RC receiver. The left joystick controls the robot's speed, while the right joystick controls its direction, allowing for both indoor and outdoor operation.

In SELF DRIVE mode, the Arduino is stopped and then waits for an initialization command from the Raspberry Pi. Once the command is received, the Arduino will pass through motor commands over the serial port. The Arduino at the same time will report encoder values and the status of the system at regular intervals over the serial port, which the Raspberry Pi then broadcasts to ROS2. The Pi is also responsible for relaying the commands of whichever mode (I.e. GPS, lane following, or a combination of both) is currently being calculated and run.

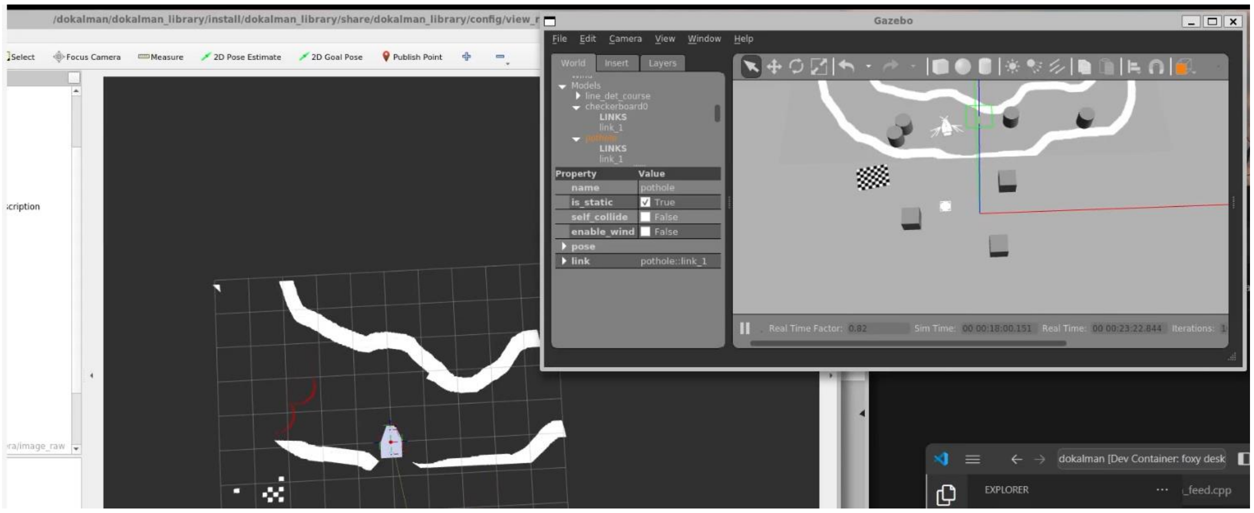
The Raspberry Pi handles the other various serial communication tasks. The Pi communicates to the Arduino, GPS, and IMU over various USB serial connections. The Pi is also running the Velodyne LiDAR node as well to initiate data streaming. The Pi then broadcasts this data to the network over ROS2 so other devices can utilize the available data.

The Nvidia Jetson Xavier handles the input from three separate USB cameras. These cameras are calibrated before operation to determine the optimal threshold for line detection as well as the transformation matrix used to skew the images to a top-down view.

Vision System

The vision subsystem plays a crucial role in handling and processing the vast amount of visual data, refining it into more actionable information for the control subsystem. This subsystem consists of an NVIDIA Jetson Xavier module accompanied by three wide-angle USB3.0 cameras. To effectively manage the data, the NVIDIA Jetson Xavier module efficiently processes each of the cameras through C++ image pipelines running through ROS2.

The pipeline for each camera consists of a few components. First the raw image is extracted. Then the image is skewed such that the view of the ground is calculated to a top-down view to make the representation of the ground more accurate. Then lines are detected from the image using thresholding and blob detection to find large continuous sections of white in the image. The binary image (line and not line) is then converted to a point cloud. The robot can have an accurate top-down representation of these obstacles in approximately a 300-degree radius with a range of over 10 feet in those directions. This point cloud is then projected into space with ROS2. This process is repeated for each camera. An example of this is shown below:


Gazebo
File Edit Camera View Window Help
World Insert Layers
Select Focus Camera Measure 2D Pose Estimate 2D Goal Pose Publish Point

The top right environment is a test world in Gazebo where the lines, barrels, and calibration checkerboard are all visible. On the bottom left is the Rviz2 view where the lines on the ground are visible as white point clouds as well as LiDAR object detection represented by the red point clouds visible.

World Representation

The vision and LiDAR point clouds can be used together to help detect all obstacles required by the competition. Running SLAM on this combined dataset will provide an accurate map of the obstacles currently and previously seen. Doing so, an accurate map of the taken path will be generated. The robot also tracks its heading in the world and the bearing towards the next GPS waypoint. This is all used to calculate the optimal path to take. Outlined in the following section.

Navigation

The intel NUC by design is solely responsible for running the navigation on the robot. As we are using ROS2 for our communication between devices, we are utilizing the NAV2 ecosystem provided by ROS2. Some of the tools we use are the slam-toolbox and waypoint follower. We currently have a waypoint follower working in simulation. We have successfully tested the robot with predefined waypoint paths and randomly generated waypoint scenarios. We are in the process of integrating obstacle avoidance with LiDAR and once that is completed, we will add in the vision point clouds with the same process and believe this will be sufficient to navigate through the IGVC course.

Innovations and Experiments

We are experimenting with deep neural networks as an AI powered navigation system. This system operates through a methodology called imitation learning. Using recorded sensors as inputs and the human operator's actions as the truth, we can teach the network different policies. We have tested this on waypoint following with inputs to the network as the heading and bearing angles and the output as steering angle. This has had varied results, and further tuning is required to determine the success of such systems. We plan to test line and obstacle avoidance immediately after completing this report. If this testing is successful, we may integrate this into our final navigation system.

Cyber Security Analysis

Scenario Overview

In the scenario of a rival team attempting to disrupt our robot's software in the pit area, we will employ the NIST Risk Management Framework (RMF) to identify and mitigate these risks. The RMF consists of seven steps: Prepare, Categorize, Select, Implement, Assess, Authorize, and Monitor.

Prepare

To prepare for potential risks, we develop a comprehensive risk management plan. This plan identifies potential threats to our robot's software and outlines our approach to each situation. We follow a baseline control selection approach, which includes categorizing types of risks and selecting a set of security controls based on these risks.

Categorize

We identify several risks associated with our robot's software:

- High Risk: An adversary could modify software on an open PC, especially if it has a remote connection to a device on the robot. Unauthorized software changes could be pushed to our repository and end up on the robot when devices pull updates.
- Moderate Risk: Unauthorized access to computers by discovering passwords. This risk applies to devices like Raspberry Pi, NUC, Jetson, or any team member's PC.
- High Risk: The Arduino can be flashed from any computer, potentially allowing unauthorized code changes.

Select

We select a baseline control policy using NIST SP 800-53B as our guide:

- Control Baselines: Choose pre-defined sets of controls tailored to the impact levels of our information systems (low, moderate, high).
- Tailoring Controls: Adjust these baselines to suit our specific needs and operational context, based on our risk assessment and stakeholder requirements.

Implement

To implement the selected controls our team uses these risk-minimization techniques:

- Physical Security: Team members practice behaviors such as locking unattended laptops and closing remote connections.
- Access Controls: Members must be invited to password-protected repositories, and only designated team leads can approve changes to the main branch.
- Multi-Layer Security: Access to the robot's computers requires multiple layers of passwords. Primary access is through the network of the base station and then remoting into the computer. An adversary would need to gain access to the network, identify the correct computer, and know the username and password.
- Change Tracking: Git's built-in tools are used to easily identify and track changes in the code.

Assess

During the competition we will assess our techniques by:

- Monitoring code changes throughout the competition
- Monitoring access to our network and computers

In the future we will perform penetration tests and determine the security of our network and ROS communications.

Authorize

Authorization measures include:

- Password protection for team member access only.
- Restricting network access to only authorized devices through controls on our router, such as a whitelist of authorized IPs.
- Limiting repository access and code change approvals to a few authorized members.

Monitor

Continuous monitoring methods include:

- Network Monitoring: Use the router to monitor devices on our network.
- Code Monitoring: Track changes in the repository and on our devices using Git tools.
- Incident Response: Establish protocols to quickly respond to and mitigate any detected security incidents.

Conclusion

Implementing the NIST RMF allows us to systematically identify, select, implement, and monitor security controls, ensuring robust protection for our robot's software against potential disruptions by rival teams. malicious code was injected and saved, we can always track when this incident occurred, and identify and remove those changes.

Vehicle Analysis

Lessons Learned

Some of our top lessons from the hardware were from the durability standards of some of the components previously on the robot. The battery holder was made of a cheap material and not reinforced.

During travel, the robot was transported with the battery inside, destroying the battery holder and crushing a few components. Additionally, we had one of our main power distributors, the Rigrunner, held in place vertically with tension of 4 screws, this force eventually cracked the panel, and we redesigned the layout to be more optimized for the task. Additionally, the LiDAR holder was printed in a different orientation, so any tangential forces do not shear the layers apart.

Some integration lessons learned are that it is hard to standardize all our computers to the same operating system. Nvidia Jetsons require specific Nvidia operating systems, which limits the version of Ubuntu and ROS2 consequently. Additionally, the Raspberry Pi disincentivizes users from installing desktop versions of older operating systems, meaning that we had to install the server version, then install the desktop version which took a non-trivial amount of time to figure out.

Additionally, we face lessons from running real hardware. GPS signals can be noisy and images from cameras are dependent on light conditions. Simulation conditions are much easier to control and many of the sensors operate perfectly. We are still working on resolving these issues with sensor fusion and calibration.

Top Hardware Failures

Some of the top hardware failures we could encounter would be the cameras shifting their position. We are currently developing a way to hold them in place. If a camera shifts, its calibration will be disturbed which can make the sensor data for mapping the environment unreliable.

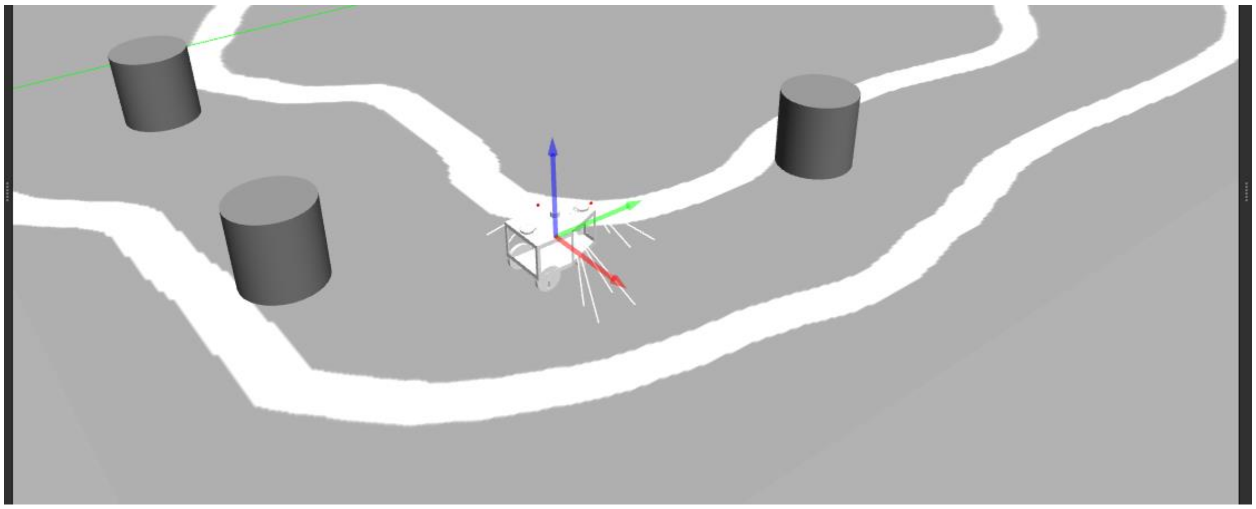
Additionally, the mechanical connections of the wires could come apart. Such an issue can cause devices to power off or in the worse case scenario, wires can short to unwanted connections and cause hardware to become damaged and inoperable. To prevent this, we try to implement terminal connectors designed for screw terminals, we also rigorously test each connector we build by pulling on each wire with a force stronger than anticipated from regular use. We also organize and hold down wires with wire holders and zip ties to prevent wires from crossing or pulling.

Software Testing and Control

We track our todo items and bugs in a tracker on our Microsoft Teams channel. From there, members can pick their task and work towards completing it. These tasks are written to be as clear as possible with the tools required and a checklist of steps to take to achieve the task. We control the software changes through our central repository where members are free to make branches, but only the owners of the repo are allowed to approve of changes to the main branch.

Virtual Environment

We are using Gazebo to test our robot. We have a series of test worlds. We start with some baseline worlds with lines on the ground with patterns such as straight paths, curved paths, and circle paths. We then can add barrels, lines, or spots on the ground to mimic obstacle scenarios for the robot. Our robot’s geometry and sensors are defined with a URDF file (universal robotic description format) where we match to the real robot as closely as possible. In this virtual environment we have the robot detecting lines and barrels and plan to finalize the navigation as mentioned before. Below is an example of the robot on a set of lines. We also have models for traffic cones as well that we use for testing.



Physical Testing to Date

Our physical testing has been limited as our main priority has been developing the simulation. The simulation allows us to safely test and develop NAV2 systems were deploying the real robot without verifying the work could cause us to damage the robot or face unwanted failures. Currently the only issues we can foresee is that sensors may not publish as fast due to latency on the network. Further testing is required to determine if this is an issue as well as the severity of it.

Initial Performance Assessments

To date, we have the sensors and hardware complete. We have verified that all of the sensor data can publish across the network with ROS2 and is visible by all computers on and off the robot. Our performance assessments for navigation have been solely in simulation and plan to work on the real robot’s navigation in the following week. We will implement the waypoint following, speed control demonstration, and the lane/obstacle avoidance as required by the IGVC. We are confident in our ability to make the robot navigate for the IGVC this year.