Western Illinois University
Rock-E

Submitted: May 15, 2024

Team Captain

Justin Scott	je-scott3@wiu.edu	Mechanical Engineering

Team Members

Jacob Kruse	j-kruse@wiu.edu	Electrical Engineering
Caiyun Wurslin	c-wurslin@wiu.edu	Electrical Engineering
Aaron Flemming	am-flemming2@wiu.edu	Mechanical Engineering
Kanika Palecanda	kg-palecanda@wiu.edu	Mechanical Engineering
Harsh Shah	hp-shah@wiu.edu	Mechanical Engineering

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

Dr. Il-Seop Shin
Professor, WIU School of Engineering and Technology

CONDUCT OF DESIGN PROCESS

Introduction

Rock-E is an intelligent, autonomous robot specifically designed to compete in the 2024 Intelligent Ground Vehicle Competition (IGVC) located at Oakland University in Rochester, Michigan. The vehicle is a collaborative effort of senior electrical and mechanical engineering students of the Western Illinois University School of Engineering and Technology. All work presented in this report was completed by the aforementioned senior design team during the Fall 2023 and Spring 2024 semesters.

Organization

In order to streamline the development of Rock-E, the senior design team divided into two groups of emphasis: mechanical/electrical and navigation. The mechanical/electrical team consisted of group members that focused on the mechanical aspects of Rock-E, along with the development of an electrical power distribution system. **Figure 1** includes breakdown of each member's primary responsibility.The navigation team for Rock-E consisted of group members whose primary responsibility was to develop the vehicle's navigation system. Development included research and procurement of all navigation related equipment along with the software development for the vehicle.

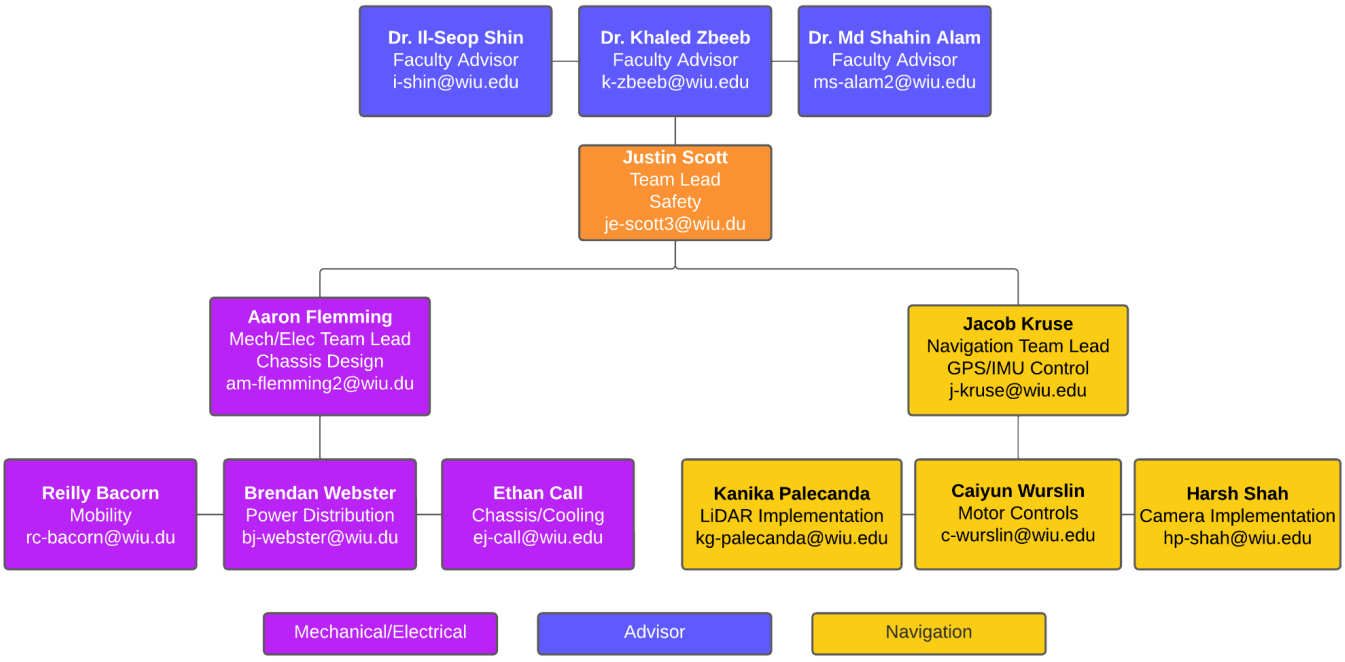

Figure 1: Team Organization

Design Assumptions/Process

The design process for Rock-E started by gathering data on previous top competitors at the IGVC along with data on designs created by previous WIU competitors. Team Rock-E used this data and the design rules of the competition to create a draft of the overall system and any peripherals that would be needed. A rough layout, on paper, was created by the team which was followed with design iterations based on further research. This design approach allowed the team to create individual categories of the design that each member could focus on.

After a rough plan was developed, the team was split into two factions (mech/elec and navigation) to allow for more focus on the overall team's objectives. Each faction nominated a leader to keep track of delivery dates and to report their progress. Meetings were held 3 times a week with faction members and once a week with the full team. Gantt charts were used to report progress weekly and to keep the team on track. As design elements were completed, team members would migrate to other areas of the design that were still in development.

SYSTEM ARCHITECTURE

Software Modules

Rock-E utilizes the Robotic Operating System 2 (ROS2) as its main software. The system is divided into five sections to accomplish the functions of lane following, obstacle avoidance, and waypoint navigation. The software structure and devices can be seen in **Figure 2** below. The Central Processor is a Dell Precision 3480 running the Humble Hawksbill distribution of ROS 2. The LiDAR, GPS, IMU, and Camera components are connected directly by USB to the computer. They provide respective data to the Central Processor to process the information. The Camera and LiDAR data is used to generate the Local and Global Costmaps, which are used to perform the Obstacle Avoidance of the vehicle. Since data stability is critical for robot control, the Extended Kalman Filter is used to eliminate inaccuracies. The Wheels Encoders, IMU, and GPS data are combined in the filter to provide the Local and Global Localization of the robot. Using the Costmaps and Localization data, the Global and Local Planners generate paths to guide the robot to its destination. Furthermore, when the robot moves backward, the Collision system, constructed with three Ultrasonic Sensors, will detect obstacles behind the vehicle, and the Center Processor will change the velocity commands to prevent an accident.

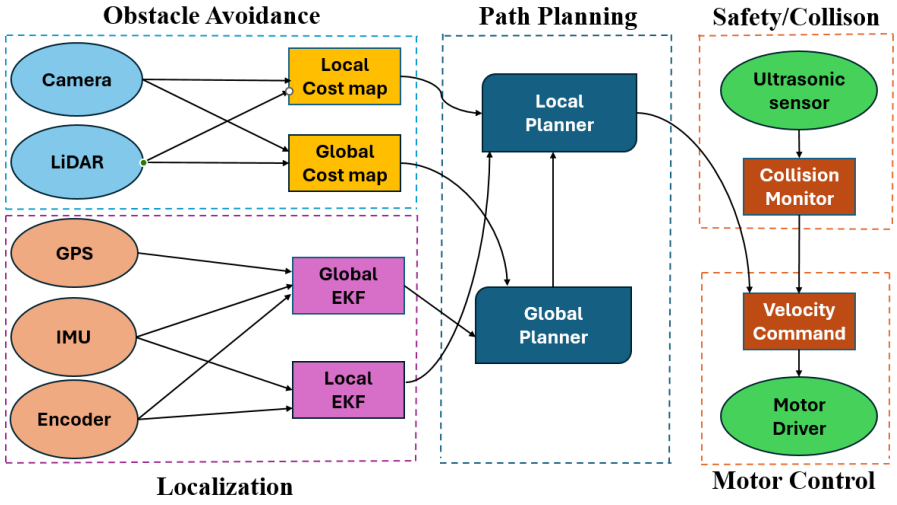

Figure 2: Software Structure and Devices

EFFECTIVE INNOVATIONS

Suspension

Once testing was under way for Rock-E, it was determined that a suspension system was needed on the rear casters of the vehicle. A suspension system would help alleviate vibrations that were being generated when moving over pavement. After many iterations, the mechanical team developed a 3D printed suspension with an adjustable spring rate (**Figure 3**). The system consists of a 3D printed control arm, a 3D printed sliding support (wheel attachment point), and a 3D printed spring. The control arm and slider are printed out of Nylon with interwoven layers of carbon fiber while the spring is printed from Polyethylene terephthalate glycol (PETG). Nylon/carbon fiber was chosen for the control arm/slider because of its strength to weight ratio, while the spring was printed from PETG because of its resistance to deformation and ease of use. Spring rate is determined by the thickness of the spring cross-section. The cross-section can be adjusted by increasing or decreasing the thickness of the layers on the 3D printed spring.

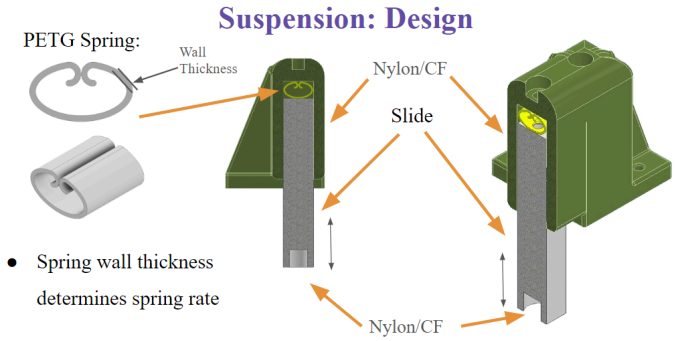

Figure 3: 3D Printed Suspension Design

Ultrasonic Sensors Collision Monitoring

In today's highly technological world, more advanced technology is usually preferred. However, it was decided to include, as some might say, outdated technology with the Ultrasonic Sensors. With the decision to have the LiDAR sensor at the front of the vehicle, the robot was left with a large blindspot. To combat this, three Ultrasonic Sensors were placed in the back of the vehicle. The field of view of these sensors can be seen in **Figure 5** below. Although these sensors have no input into the path planning algorithm, they provide extra safety for the robot when it is backing up and navigating as normal. If an object is detected within the "keep-out" zone (**Figure 6**), the current velocity commands are overridden and stop the robot from moving any further backwards. It still has the ability to move forward and turn to escape the detected obstacle, making this simple design effective.

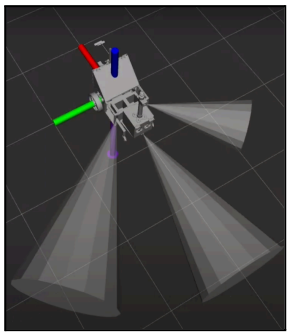

Figure 5: Ultrasonic Sensors' Field of View

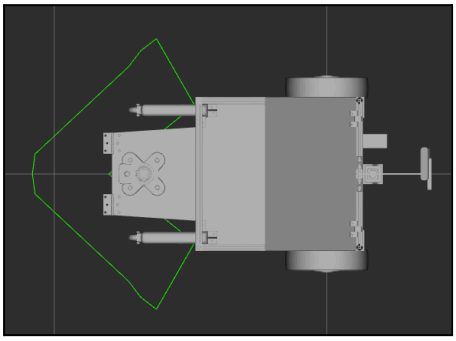

Figure 6: Ultrasonic "Keep-Out" Zone

Motor and Encoder Mount

Initially the plan was to purchase a motor with a built in encoder. However, the supplier that made them was out of stock and had no estimate on when or if another would be available. Because of this the team needed to design its own encoder mount and initially planned on trying to recreate what was observed in the out of stock option. This approach was determined to be too much of a risk of damage to the components so an alternative solution was needed. The team decided to go with a belt driven encoder that was mounted alongside the motors seen in **Figure 4**. Each encoder has the ability to relay 360 pulses per revolution. When connected to the timing belt system (reduction ratio 4:1) each turn of a drive wheel will account for 1,440 pulses per revolution at each encoder. To accomplish this, the motor mounts needed to mount the encoder close enough to the motor's drive shaft to allow a belt to be connected across a gear mounted to the shaft of each.The encoder mount consists of a face-mounted fitting that will be attached to the encoder itself and the fitting then will be attached to the motor mount by an adjustable slot system. The adjustable slot system allows for fine tuning of the belt tension.

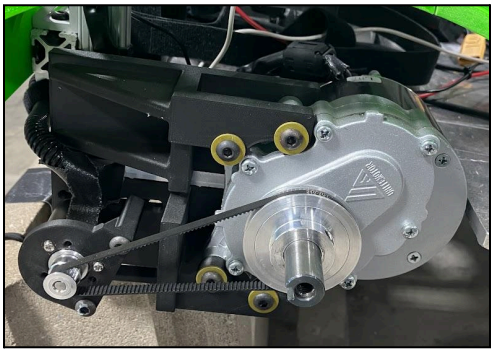

Figure 4: Motor and Encoder Mount with Belt

LiDAR and Camera Integration

In order to simplify the decision making of the navigation system, the LiDAR and Camera data was combined. Although the Camera is used to detect lines and potholes while the LiDAR is used to detect barrels, all of these things are treated as obstacles. In order to avoid different coding for lines, barrels, and potholes which would make the system more complex, it was opted to treat all of these things as obstacles. This way cost is generated around all of these objects and the robot will navigate to avoid everything. The generated maps are better explained and can be seen in the Software Design Section.

MECHANICAL AND ELECTRICAL DESIGN

Overview

After studying previous IGVC competitors, it was noticed that teams would have varying degrees of issues with the discrete electronics. The issues would vary from minor electronic glitches to inoperable electronic devices, under certain conditions. Team Rock-E's theory is that the electronic devices, mainly the motors and motor drivers, were causing interference with the more sensitive electronics. To combat this, Rock-E's power system consists of a 12-volt and a 24-volt system that run separate from each other.

Motors

One decision that needed to be kept in mind was motors that would provide enough torque to propel itself across the course and over a 15% gradient and also hardware limit the speed to 5 mph. The pair of motors used in Rock-E are used in powered wheelchairs, the Unitemotor MY1016Z. The pair provide a torque of 80 Nm and 120 rpm, which will limit Rock-E to a theoretical max speed of 4.28 mph due to the 12 inch wheels. Rock-E has a mass of 57 kg which will require a torque of 13.483 Nm to successfully climb the ramp. Rock-E's motors will provide more than enough torque to do so.

Electronics Box

The electronics box was designed with 2 main goals in mind. Keep the sensitive electronics in a cool, weather proof environment and remain easily accessible for troubleshooting electronics issues. As seen in **Figure 7** and **Figure 8**, the electronics box utilizes sliding vertical shelves enclosed in an acrylic box. Combined with the fan the vertical panels makes it easier for hot air to easily be pulled out of the box as opposed to horizontally aligned panels which would cause heat to be trapped in between layers. The panels also slide for accessibility for ease of troubleshooting rather than having to unsecure circuit boards to diagnose issues.

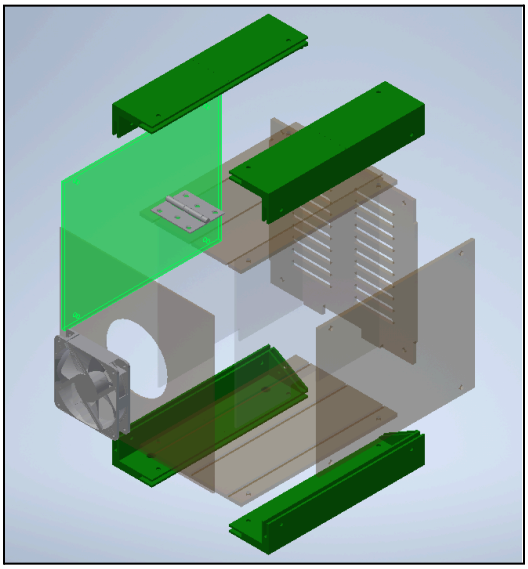

Figure 7: Disassembled Electronics Box

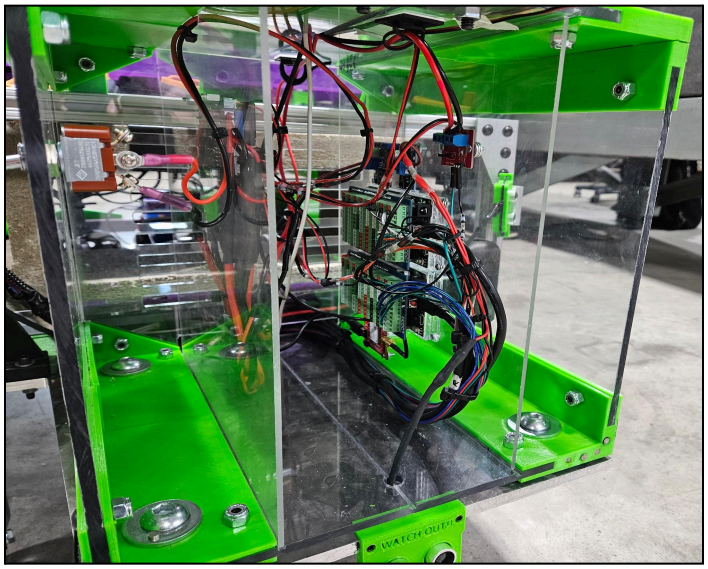

Figure 8: View Inside Electronics Box

Frame

The inspiration for the framing of the vehicle came from zero-turn lawn mowers. This choice was made for maneuverability. The frame was designed for modularity and room was left for some adjustments to be made throughout the design process. The main chassis of Rock-E is made out of an aluminum base plate and T-slotted aluminum extrusion. This decision was made factoring cost, weight, strength, and machinability. The frame of Rock-E (**Figure 9**) can be viewed as having four sections. The rear section contains the electronics box and safety features such as the E-stop and the light. The middle section contains the payload shelf, rear wheel suspension, and a channel for wiring. The lower front section houses the motors and batteries. While the upper front section is used for the CPU and detection equipment.

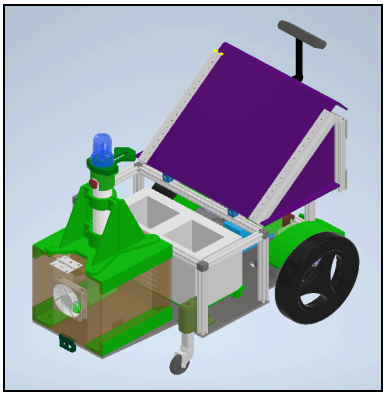

Figure 9: Rock-E Rear View

Weatherproofing

Knowing that the IGVC will take place outdoors and that Rock-E may have to operate in light rain, some design choices were made to mitigate this risk. The electronics box was designed to provide a weather proof environment for the small electronics. While most of the smaller electronics are housed within the electronics box, the Ultrasonic Sensors and the IMU are located outside the vehicle in the front and in the rear. These pieces of equipment were protected using 3D printed covers and potting to weather proof as much as possible. Finally a removable cover was made for the laptop shelf which can be used to protect the laptop and USB hubs in the event of rain.

Power Distribution System

Rock-E's 24-volt power system consists of a 100 amp hour battery, 2 motor drivers, and two wheelchair motors that each produce 250 Watts of power (**Figure 10**). Maximum amperage draw of the drive motors during testing was 11 amps. This gives Rock-E a max-speed run time of 9 hours. Maximum recharging time is 12 hours.

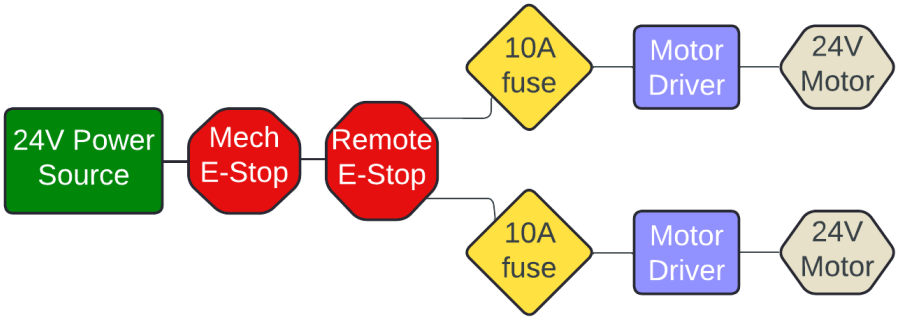

Figure 10: 24 Volt Power Distribution

Rock-E's 12-volt power system consists of a 2 amp hour battery, a power distribution module, two Arduino Due microcontrollers, ultrasonic sensors, an indicator light, and a laptop with a suite of electronic devices attached (**Figure 11**). Max amperage draw during testing has average around 1 amp, giving Rock-E a run time of 2 hours. Rock-E can be charged between runs if needed. Maximum recharging time is 45 minutes.

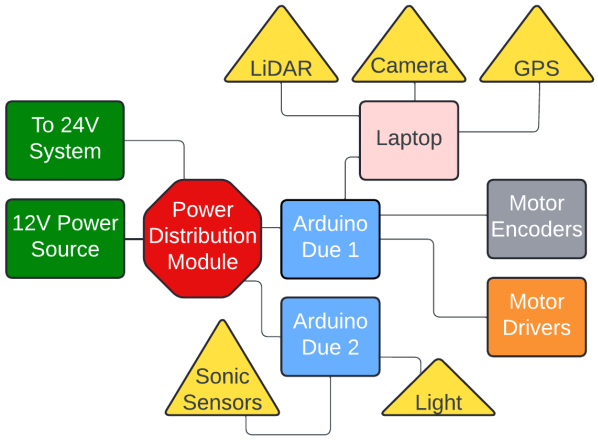

Figure 11: 12 Volt Power Distribution

Electronics Suite

Rock-E is equipped with three processors: a Dell Precision Workstation 3480 and two Arduino Dues (**Figure 12**). The Arduino Due's each have a specific task. Arduino Due 1 is a translator for the motors and motor encoders. It receives information from the motor encoders and converts it to data the laptop can decipher. The laptop then outputs converted velocity commands to the Arduino which will send the corresponding PWM signal for the motor drivers. Arduino Due 2 receives a signal from the laptop to illuminate the safety light. Depending on the operating condition, the light will flash or remain solid. The Arduino Due 2 also reads data from the ultrasonic sensors and relays the distance information to the laptop.

The Dell Workstation is the main processor for Rock-E. It takes in all data from the sensor suite (Camera, LiDAR, and GPS), along with the data from the Arduino Due's and uses a custom-built software program to determine the output movements Rock-E will make.

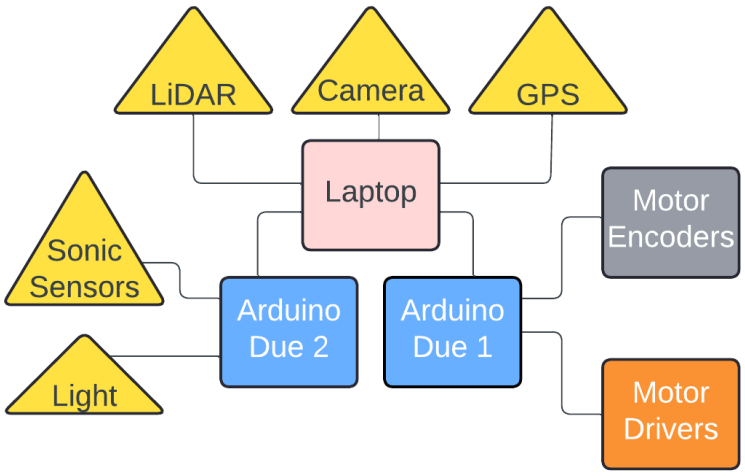

Figure 12: Electronics suite for Rock-E

Mechanical and Wireless E-Stop

Rock-E is equipped with both a mechanical and remote E-stop solution per IGVC 2024 rules (**Figure 13**). The mechanical E-stop is a standard mechanical, actuating E-stop with a 40mm diameter red push button in a location pursuant to the IGVC 2024 rules. The remote E-stop for Rock-E is a Kar-Tech wireless E-Etop. This E-Stop is specifically designed to work with remote vehicles at a distance of up to 600 feet. The unit operates at 2.4 Ghz (certified) and uses a dual-channel setup. This means the remote must be in contact with the base unit located on the vehicle at all times. If the signal is interrupted between the remote and the base unit, whether the vehicle is out of range or the E-Stop is activated, the base unit will shut down the motor circuit to Rock-E. This will bring the vehicle to a complete stop while still allowing the discrete electronics to stay active.

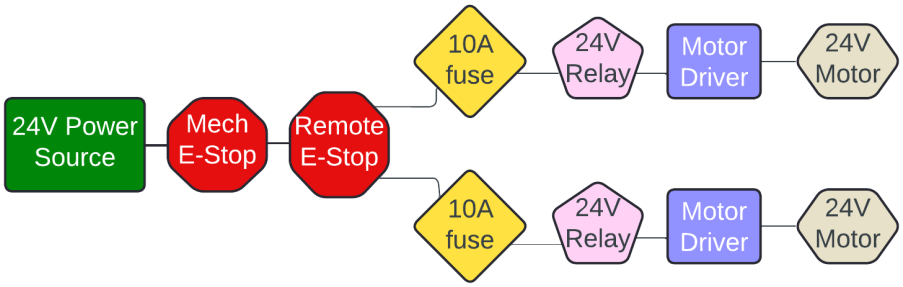

Figure 13: E-Stop Circuit

SOFTWARE DESIGN

Overview

The software system of the robot is built around the Robot Operating System 2 (ROS2) as it provides a modular structure and the ability for each subsystem to run simultaneously while communicating with each other. Along with the main computer, two Arduino Due microcontrollers are used for peripheral connections and motor control. A Rplidar S3 (LiDAR) sensor is used for barrel detection while a ZED 2i Camera is used for lane and pothole perception for the obstacle detection of the vehicle. A SparkFun ZED-F9P GPS receiver is used to receive global coordinates, an Adafruit BNO055 IMU is used for inertial values of the vehicle, and two Taiss Incremental wheel encoders are used to measure the wheel velocity for the localization of the vehicle. The open source Nav2 package is utilized for the path planning of the robot. In addition to this, three Ultrasonic Sensors provide a failsafe for the vehicle when backing up.

Motor Control

It has been decided that this project will utilize Nav2 as the main navigation software. With information from all systems, it generates a path which is carried out by sending “Twist” velocity commands to the motors. **Figure 14** below demonstrates the motor control data flow. After a path is generated, the Controller Server will begin publishing velocity commands to achieve the desired waypoint. This information is sent into the Velocity Smoother where the given velocity commands are made less erratic, allowing the robot to accelerate and decelerate gradually. These smoothed commands are then sent to the custom “twist_to_motors” code where an equation converts the velocities into a Pulse Width Modulation (PWM) signal, which is required for the Cytron MD30C motor drivers (**Figure 15**).

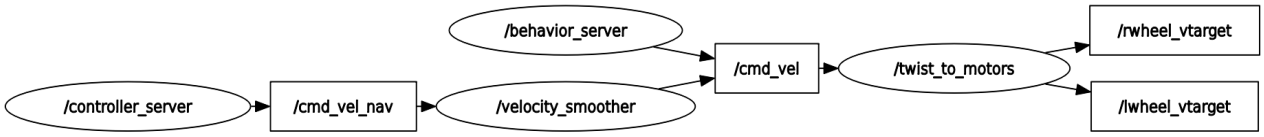

Figure 14: Motor Control Data Flow

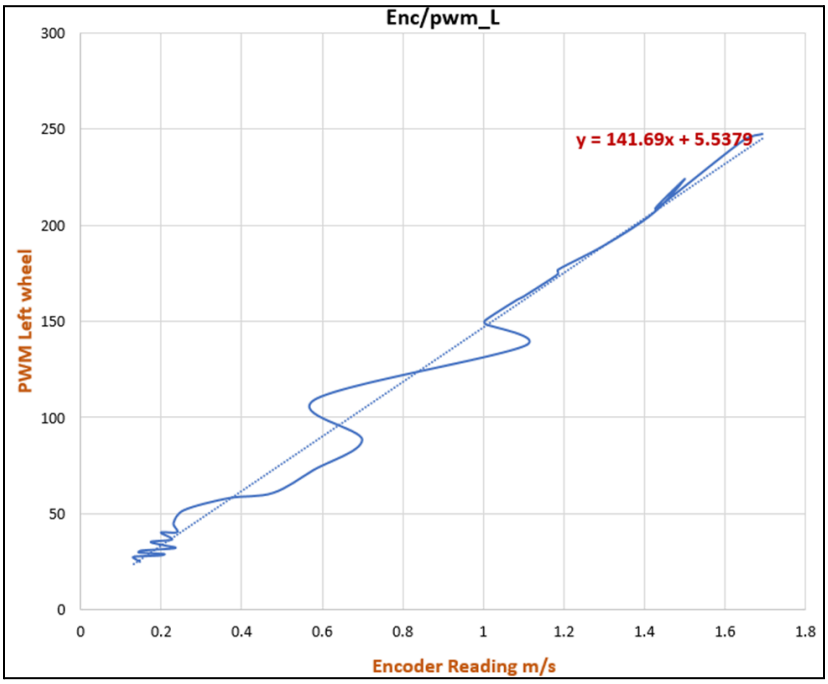

Figure 15: Left Wheel Mapped Velocity to PWM Curve

Obstacle Avoidance

For the robot to navigate autonomously, it must be able to detect and avoid obstacles. This is accomplished using two sensors, a depth sensing camera, and a LiDAR. The devices used are the ZED 2i camera and the RPlidar S3. The LiDAR detects and transmits data regarding distance and angles of obstacles with respect to the vehicle, while the Camera detects white lines and potholes. The Camera data is transformed to the same data form as the LiDAR for easier integration of their data so that all the incoming data from these two sensors can be considered as obstacles (See Innovations).

The obstacles are detected and stored in a permanent map, and a temporary map is also produced that only considers the obstacles in a specified area around the vehicle. The permanent map is developed using the SLAM software, which stores all the scanned data from the LiDAR and Camera which has any permanence. Regions of cost are added around the map created by the SLAM software using a Nav2 package, specifically costmaps_2d. These regions of cost indicate to the robot how close it can get, and how wide of a radius around the objects is necessary for the robot to safely navigate around the perceived obstacles. The permanent map with cost applied is used as the global costmap while the temporary map with cost is used as the local costmap for the path planning algorithm. **Figures 16, 17, 18,** and **19** below show this process.

Figure 16: Test Course

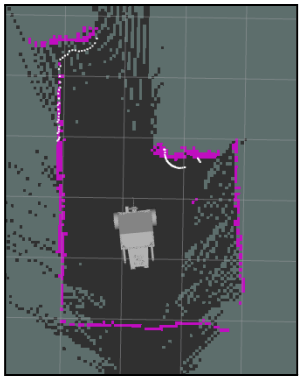

Figure 17: SLAM Generated Permanent Map

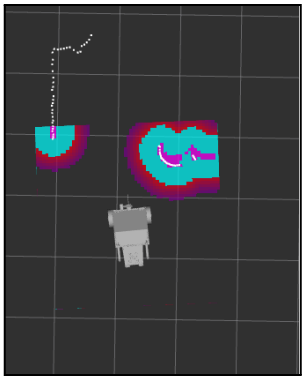

Figure 18: Local Costmap (Temporary Map)

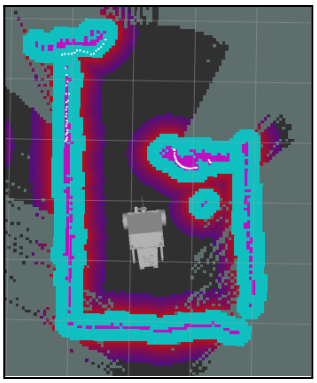

Figure 19: Global Costmap (Permanent Map)

Localization

A vital function of the robot is the ability to tell where it is in space. The vehicle needs to know where it is in reference to obstacles and lines in order to avoid them, while also being able to tell its global position and heading to navigate to GPS waypoints. In order to localize itself as accurately as possible, the robot fuses data from a number of sensors. This technique is made possible with an Extended Kalman Filter, which is great for the system as this filter can handle unpredictable, nonlinear systems. In most cases, encoder and IMU data is sufficient to effectively localize a robot in its environment, but GPS waypoint navigation is necessary. GPS data can be very inaccurate, and is subject to discrete jumps, so two instances of the Extended Kalman Filter are run simultaneously. The first filter takes data from the BNO055 IMU and wheel encoders. This filter is accurate in the short term, and produces continuous data that is used for the Local Planner and for short term navigation. The second filter fuses all of the data, which includes the GPS. In order to be used with the other data, the GPS data is first converted into Universal Transverse Mercator coordinates. The traditional Geographic coordinates are based on a spherical, three-dimensional system, and are subject to distortion. However, UTM coordinates are two-dimensional and fix most distortion, making them more suitable for the system. The converted GPS data is then combined with an absolute orientation provided by the Adafruit IMU. This orientation has to be based in the ENU frame, which means the IMU has to read 0 degrees when facing East, and 90 degrees when facing North. The data from the wheel encoders is then fused to produce a combined output that is used in the Global Planner for waypoint navigation.

Path Planning

Path Planning is arguably the most important part of the navigation system. Even if every other system is working to perfection, the robot would not be able to navigate autonomously without this ability. The Nav2 package provides a simple way for the vehicle to make effective obstacle avoidance and waypoint navigation decisions. In short, the path planning of the robot is broken down into a global and local plan to allow for more precise GPS wayfinding in the long term and much safer obstacle navigation in the short term.

The global plan represents the overall path the robot needs to follow in order to reach its ultimate goal, which in this case, is each GPS waypoint. This path is generated with information from a bunch of different areas. The global costmap, global localization, and each waypoint are all required to produce a global plan. Nav2 allows for custom algorithms to be developed and implemented into the code, but it was decided to use the Smac 2D Planner for this project. This provided planner uses the A* algorithm to produce a global path depending on the goal, detected obstacles, and its current location. Additionally, the Smac Planner was chosen because it can be used for robots that do not have a circular footprint. Due to the robot being more rectangular in shape, the popular NavFn Planner was ruled out since it is used for circular robots. The Global Planner saves all of the cost data so that the system knows the location of all detected obstacles and lines in the past and present. This way it can make more educated decisions on how to proceed to the next waypoint. Images showing the global costmap and the global plan produced in reference to our test course can be seen in **Figure 20** and **Figure 21** below.

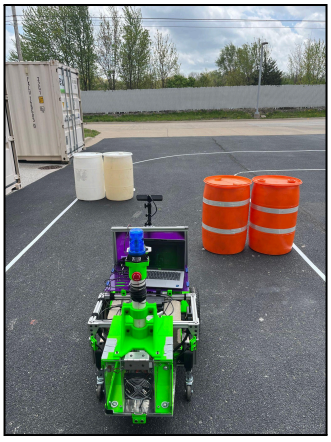

Figure 20: Test Course

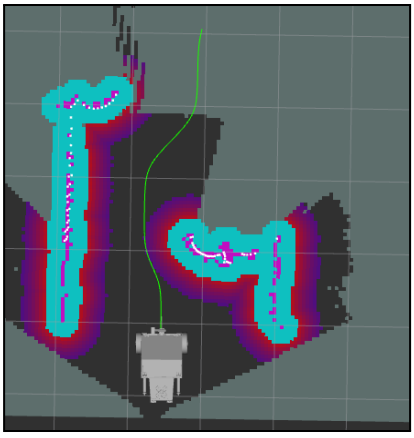

Figure 21: Generated Global Plan

While the Global Planner holds the ultimate goal of the robot, the Local Planner is used for short term navigation. Much like the Global Planner, the local path needs the local costmap, output of the local filter, and short term goal to produce a path. The main goal of the local path is to avoid nearby obstacles and stay within the lines while staying as close as possible to the global plan. The Nav2 Controller Server is used to produce this path, and it was decided to use the default DWB Controller from the package. This controller takes in the current local plan, outputs the required velocity commands to proceed, and continuously checks to see if the goal has been reached. Unlike the Global Planner, the Local Planner only sees obstacles in its immediate area from the local costmap. This reduces the computation power and allows for more efficient paths to be generated. In addition to this, the Local Planner receives data for the robot footprint to produce a much more accurate path and to avoid collisions. The robot footprint used and the local plan produced from the local costmap is pictured in **Figures 22** and **23** below.

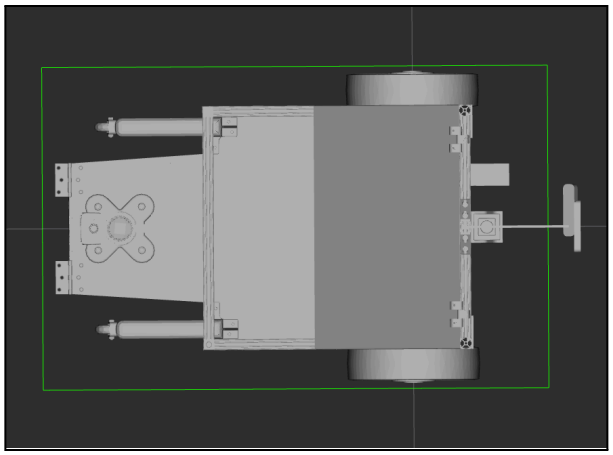

Figure 22: Robot Footprint

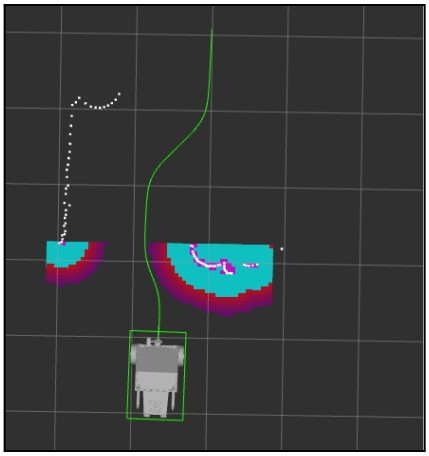

Figure 23: Generated Local Plan

Failure Modes

As confident as we are with the state of the system, failure is always a possibility. Because of this, a number of fallback actions and things have been put into place to ensure the effective operation of Rock-E.

In the case of autonomous navigation, it is always possible that the robot could get stuck without knowing how to proceed. To combat this, a number of behaviors have been added to the system for this situation. The first action of the robot, if it is unsure how to proceed, is to wait. This gives the computer and software time to make more computations and figure out an alternative path forward. If this does not help, the robot will then spin, if the space permits. This can give the robot the ability to learn more about its surroundings and may give it the right information to proceed. If Rock-E passes through both of these actions without finding a solution, it will back up, space permitting. This action is done for the same reason as the previous, to get a better picture of its environment, but backing up is the last resort. The robot will cycle through these actions for a certain period of time before coming to a stop if it is completely stumped. Although this is unlikely, it was desired to have a timeout to safely retrieve the robot from the course. This also allows the team to better understand why it failed.

In addition to fallback actions, a number of safety actions have been taken for the motor control. All of the systems on Rock-E have no threat to individuals or the robot itself, except for the motors. If the robot loses communication or control of the motors, this could lead to a collision. In the early stages of testing, this occurred a couple of times, and after assessing the error, a number of things were discovered. Due to the publisher-subscriber structure of ROS2, the subscription callbacks of the velocity commands would only update if a new velocity command was received. Because of this, custom timeouts were added to a number of motor control codes that will set the motor velocity to zero if no command is received. In addition to this, two Arduino Due's are being utilized in the system. In order to communicate with the computer, a micro-ROS agent is run. Sometimes, however, this agent can fail, causing the microcontroller to lose communication with the computer. Due to this, code was written into the Arduino's to set the motors to zero if communication is terminated. Both of these steps have been tested and assessed to be successful.

CYBER SECURITY

Although it is unlikely, the threat of rival teams disrupting the software of the robot had to be considered. A common and baseline way of assessing risk is by using the NIST RMF process. This process has 7 steps: Prepare, Categorize, Select, Implement, Assess, Authorize, and Monitor. By following these steps, one can effectively identify and rate threats, develop and carry out solutions to these troubles, and evaluate the system for success in the present and into the future. This process was executed to the best ability of the team when considering the risks pertinent to this competition

Threats and Level of Risk

With the vehicle operating solely on the commands of the main computer, the software is vulnerable to sabotage from other teams at the competition. A number of threats were determined and assessed based on the danger they could pose to the system. Because the robot navigates based on the information processed by each sensor, each peripheral's data stream could be sabotaged. Also, direct sabotage of the code through modification could occur if a rival team member had access to the computer. Different things could be plugged into the computer ports to upload some sort of disruptive virus. With this being said, the most severe threat is the direct code sabotage, followed by the virus injection, and finally the subsystem software.

Solutions and Implementation

Due to the possibility of cyber attack, a number of things are in place while other solutions can be implemented in the future. At this time, the computer has a main password in order to access it on startup. This should stop most threats initially. If somehow a rival team member had access to the computer when it is opened, there is another failsafe. Before any changes are made to the computer, another password must be entered to achieve root access. Additionally, power settings on the computer cause it to lock and return to home after a period of inactivity for more security. On most Linux devices, USB read-write access must be given by the root user. To combat sabotage through these ports, "/udev" rules are in place to give read-write access only to devices that are recognized by serial number as components of the navigation system. On top of all this, virus protection is installed on the laptop in cases the previous things fail.

To make the system even more safe, some other measures can be taken in the future. Encryption of certain data could be done in order to maintain its integrity. Data logs could be put into place each time code modification occurs so that old code can be restored before it has been sabotaged. Furthermore, different settings and rules could be developed on the laptop to counter cyber attacks.

ANALYSIS OF COMPLETE VEHICLE

Construction and System Integration

The construction of Rock-E led to valuable lessons in craftsmanship and design. The chassis went through a number of iterations to make the vehicle easier to work on which were not apparent in the beginning. The fitment of parts was an issue at first until tolerances were established. Material choice also went through iterations. Initially all 3D printed parts for Rock-E were made out of PETG. After testing, it was determined that the structural elements of the 3D printed parts could not withstand the rigor needed to perform in a rough environment, so we switched to nylon and carbon fiber. The new material is holding up much better outdoors.

Hardware Failures to Anticipate

Team Rock-E believes the rear suspension of our vehicle is a critical component that may fail during the competition. To mitigate this the team has created an extra set of suspension arms that can easily be replaced in case of failure. Other failures could include the Arduino microprocessors or other small electronic components. The team has mitigated this problem by stocking extra components that will be brought to the competition. If there is a failure the part will be replaced on location.

Software Testing

With the necessity for each subsystem working cohesively, testing on each was conducted. The Camera filter for line and pothole detection was optimized throughout the working period. It was tested in different light conditions and with different shades of white. The LiDAR sensor is very accurate, but it was tested for accuracy after the combination of the Camera data. Localization of the robot was very heavily tested as the GPS data is very inaccurate. With the combination of the IMU and wheel encoders in the Extended Kalman Filter, the covariance values of these sensors were tweaked often to affect the output of the filter. Different data was also fed into the filter like the visual odometry of the Camera and the pose published by the SLAM system, but ultimately it was opted to remove this information as the filter's performance worsened. Motor control was also optimized through Pulse Width Modulation (PWM) and Twist command mapping. In order for the robot to match the Twist velocity commands which are given in meters per second, the wheel speed was measured at different PWM values and used to develop an equation used for velocity conversions. This way, the Arduino Due can be used to control the motor drivers effectively.

Virtual Environment

Although the team chose to avoid testing in Gazebo with favor to physical testing, RViz2 was used frequently. This visualization tool helped the team see the sensor data being published and used by the system. For example, the filtered PointCloud used for the Camera line detection could be seen in real time using the software and tuned for increased perception. It was also used for visualizing the Camera and LiDAR combined data for obstacles, the Odometry data published by the filters, the Ultrasonic sensors range, and much more. An example case of this software being used is shown in **Figure 24**.

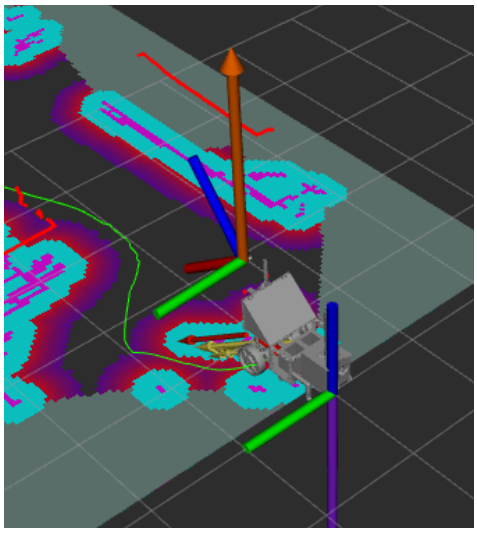

Figure 24: RViz2 Sensor Visualization

Physical Testing

Rock-E has been physically tested for robustness and operational integrity. Drive tests have been performed in indoor and outdoor settings along with testing on individual components. One of the first tests performed was a torque output test for the drive motors. Manufacturers specifications for the drive motors showed a maximum power output of 250 Watts per motor and in testing we were able to achieve this threshold albeit at a lower amperage than stated by the manufacturer (10 amps instead of 13 amps). The suspension system went through multiple rounds of testing before a final design was established. Most of our testing focused on the rigidity of the suspension. Our calculations showed that PETG was going to have a rigidity issue with our design. When the test was performed, failure occurred at the exact spot shown in our simulations. After the test we switched to nylon/carbon fiber.

Performance Assessment

At this time, Rock-E can navigate to multiple GPS waypoints while avoiding detected obstacles or potholes and staying between the perceived white lines. The physical design and wiring has stood the test of time and received lots of improvements to things like the suspension and wiring. Although this is more than satisfactory to compete in the Intelligent Ground Vehicle Competition, optimization of the system is still being done. The speed of navigation is currently being tested and improved in order to decrease the time the vehicle will traverse the course. The accuracy of the GPS localization with mapping is being fine tuned as well. With that being said, the system is very effective and ready to compete.

APPENDIX A: Cost of Build

Figure 25 includes a breakdown of the cost to build Rock-E. Team Rock-E was given a $5,000 budget to build the vehicle and spent $3.562.48 of this budget. The no Purchase total includes parts that were reused from previous projects which did not count towards our budgetary spending. A comprehensive bill of materials can be found here.

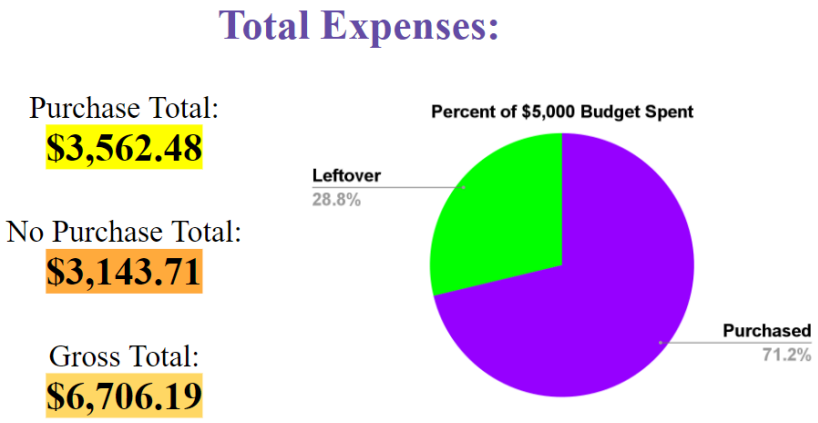

Figure 25: Rock-E Expenses