University of Arkansas – CSCE Department
Capstone I – Preliminary Proposal – Fall/Spring 2025

Lunabotics

Ryan Cheng, Joseph Folen, Landon Reynolds, Maxwell Thursby, Blake Williams, Kevin Zheng

Abstract

NASA annually holds a national competition for students to create a lightweight, efficient, and robust lunar rover. Each robot is graded based on its weight, power consumption, and excavation performance.

Currently, we are presented with a robot that can be controlled and does the necessary tasks for the NASA Lunabotics competition. The most basic objective, which has already been met, is to create a lunar rover robot with navigation and excavation capabilities. The robot has a functional communications system, a bucket to excavate dirt, as well as batteries and power management. The robot's code has been given to us to improve. These were the resources we were given at the start of the project. The current robot's design is based on previous years' designs, which have successfully completed several competitions. We were given a solid foundation, but improvements are needed. There are still issues with the robot, such as communications inconsistencies, an incomplete and poorly designed graphical interface, and a lack of autonomous functions.

At a previous competition, the robot had a communications outage that lasted several minutes. To affect success in the competition, ensuring consistency and reliability of communications is crucial. The outage occurred due to electromagnetic interference. At the Lunabotics competition site, thousands of signals emitted by other competitors' equipment can interfere with the electronics on board. Sufficiently capable networking hardware in concert with sufficiently robust communications protocols can help to mitigate the risk posed by electromagnetic interference. Connection reliability can be improved by using more capable hardware. The robot's primary antenna was originally only the embedded antenna of an Nvidia Jetson Orin Nano. This antenna was not very capable. The first measure taken was using an external antenna, but we may also implement a signal amplifier to further boost the signal. On the software side, the main communications protocol currently makes use of variably sized messages. This protocol is more susceptible to errors caused by electromagnetic interference due to inherent length checks. We are designing a new protocol using fixed message size and parity checks to improve reliability.

The current visual interface for the robot is unsatisfactory. It displays lots of information about

the status of the robot's components, but it lacks any form of live camera feed to aid in navigation. The interface does not prioritize mission-critical information; instead, it displays all available metrics in a grid. This means a cluttered interface, despite lacking the types of visual information that are especially useful for this type of application. A front and rear camera feed, staged arena camera feeds, and an arena map are among some of the useful potential additions to the graphical interface.

One major way to earn points in the Lunabotics competition is to add autonomous functions to the robot. We plan to implement a 3D simulation of the robot to allow us to test the robot and develop autonomous functions in a virtual environment. We plan to develop autonomous navigation and excavation. Since the robot can currently only be tested by being driven to an off-site testing area, the development of the simulation environment is incumbent. This environment will expedite the development of autonomous functions and revisions to communications protocols.

1.0 Problem

The NASA Lunabotics competition will test our team's design against realistic constraints similar to those faced by a real lunar robotics system. The largest problem to be addressed is the reliability of the robot's communication systems and protocols. In previous years, the system has been susceptible to an error in the communication system causing the driver to lose connection with the robot for upwards of five minutes, which is a critical vulnerability that could cost our team the competition. Similarly, previous systems have been vulnerable to substantial amounts of electromagnetic interference at the competition, leading to a similar loss of control error. Another pressing challenge is the lack of a comprehensive simulation environment. All testing is currently done in real life using the physical robot. This leads to other problems, such as the need for schedule coordination and trips to off-site locations to perform testing. This prevents rapid development from occurring, as the team is only able to test a couple times in a month. The robot also lacks any autonomous functions, which are vital to scoring during the competition. Finally, the graphical interface needs significant revision, as it is difficult to read, does not prioritize mission-critical information, and does not support multiple camera feeds.

During the Lunabotics competition, teams cannot see into the arena where the robot is. If the robot's camera feed isn't enough for a team to navigate effectively, a team can request additional camera feeds provided by pre-installed cameras in the arena. However, teams that request these arena camera feeds are docked points. A potential solution to this problem is to add a deployable awareness camera system. A deployable awareness camera system would consist of multiple battery-operated wireless cameras, which would be deployed from the main robot system and provide live camera feeds from inside the arena. The live camera feeds could be integrated into the graphical interface to enhance the driver's awareness.

2.0 Objective

The objective of this project is fivefold. First, we will aim to revise the communication system and protocols to improve the overall reliability of system control. Second, we plan to implement a comprehensive software simulation of our robot to expedite testing. Third, we will aim to implement autonomous operation for at least some of the robot's tasks using computer vision techniques. Fourth, we plan to revise the design of the graphical interface to display more

information and additional camera feeds. Fifth, we will aim to design a deployable awareness camera that the robot can drop in the arena so that we are not reliant on NASA's arena cameras.

3.0 Background

3.1 Key Concepts

BP-1

BP-1 (Black Point 1) is a crushed basalt lunar regolith simulant. It is used in the NASA Lunabotics competition, as it is meant to simulate the loose deposits of rock that characterize the lunar surface. Due to the prohibitive cost of BP-1, our testing site uses crushed limestone instead.

GUI

GUI stands for graphical user interface, the visual interface through which the driver will gather information about the robot and view camera feeds.

ROS2

ROS2 (Robot Operating System 2) is a library of middleware tools for robot software development. While it is not an operating system per se, it is an extensive collection of frameworks which make robot software easier to develop and integrate.

Jetson Orin Nano

The Jetson Orin Nano is our primary processor. We are currently using the development kit with 8GB of RAM. It is well-suited for the application because of its impressive onboard compute and its small form factor. The unit features a 1024-core Ampere architecture chip with powerful AI capabilities.

CAN Bus

The motors are controlled through a Controller Area Network (CAN) Bus interface. This is a standard commonly used in automotive wiring, but it is also well suited to this competition because of its fault tolerance in noisy environments.

ROS Gazebo

ROS Gazebo is an open-source 3D robotics simulator. It features a high-fidelity physics engine which will be useful for conducting simulations, thus greatly simplifying testing.

GTKMM

GTKMM is the GUI library we plan to use to build the new GUI. It is very suitable, as it is based on C++, featuring many widgets extensible through inheritance, as well as extensive documentation.

ZED API

The ZED API provides low-level control of our StereoLabs camera hardware. This system is suitable, as it has been used previously, and it interacts well with C++.

UDP

UDP, or User Datagram Protocol, is a connectionless protocol for transmitting packets to a receiver. It lacks the formal TLS handshake of TCP; it also lacks the built-in mechanisms for

fault tolerance endemic to TCP. All these extra functions of TCP have the potential to increase overhead and interrupt data transmission. Since some packet loss is acceptable, UDP appears to be the most suited to our application, as it will speed up communication with the robot. This is due to it lacking the formal handshake of TCP, which is what is currently being used.

3.2 Related Work

In previous years, the same networking/communication protocol has been used to transmit data and information to and from the robot [1]. However, the current implementation does not check for bit flips. Problems can arise when interference modifies the data in transit. In its current implementation, if bad data is received, it is not discarded. Instead, it is processed and used, which causes issues with the GUI and can cause it to fail. This will be improved upon through a complete rebuild of the networking/communication protocol from the ground up. This includes a new protocol for transmitting and receiving data, new binary data structures for sending information, and a new protocol for processing and checking the binary data. The new protocol will include parity checking to reduce the uncertainty of each message's integrity. Though a single-bit parity check will not verify the integrity of messages with certainty, it is a logical first step to add some quality control. To verify the functionality of this new protocol, it must be tested on the robot. In its current implementation, the only way to test the robot is by physically taking it to the test grounds. This leads to potential errors being overlooked, as there is not enough time to properly isolate variables and identify the true cause of an issue. To address this, ROS2's [2] simulation software [3] will be utilized to construct a one-to-one simulation of the robot. This will allow the software developers to run multiple tests within a day instead of just a single test. Previous years' designs have seen limited success, but these new revisions should be effective ways to improve the quality and performance of the system this year.

4.0 Design

4.1 Use Cases

The robot will be fully autonomous during one section of the competition and controlled by a human driver during another section. The Stereolabs dual-camera system will be integral to the robot's fully autonomous functions. The stereo images are processed so that the robot can infer the distance of obstacles and build a real-time 3D map of its surroundings. When the cameras detect obstacles in the arena, they will alert the robot to navigate around them. When necessary, a driver will be able to manually control the robot over Wi-Fi, using live camera feeds to navigate the environment. The driver will use a joystick to drive the robot, with buttons on the top controlling the bucket on the robot. The GUI will display errors and camera feeds for manual navigation. For the simulation software, there will be three different arenas. Each of the arena models will be at a one-to-one scale to the real arena. The robot model will be included in the simulation for testing, and it will have all the functions that are present in the real robot. Motor and actuator speeds will be simulated, and their real-world characteristics will be replicated as accurately as possible.

Each of the simulation arenas will include the obstacles that are present in the real arenas. These simulations will be used to perform automation testing and general testing, eliminating the need to transport the robot to the testing pit, which requires significant effort and time.

The robot will need to consistently and reliably transmit mission-critical information and high-fidelity camera feeds to the driver through the GUI. The driver will transmit commands to the robot over Wi-Fi using the controller.

4.2 Requirements

4.2.1 Competition Constraints

The robot requirements for the Lunabotics competition are as follows:

- Maximum robot dimensions: 1.5m length, 0.75m width, 0.75m height.
- Maximum robot mass: 80kg.
- Navigational aid system (external components to be deployed with the robot but not attached, such as awareness cameras) mass **will be included in robot mass**.
- The navigational aid system must be self-powered.
- External robot antennas are required.
- Four lifting points are required on the robot at minimum.
- Teams are responsible for placement & removal of the robot on the BP-1 surface.
- All parts of the robot must be controllable by the team.
- The robot cannot use any touch sensors to avoid obstacles.

4.2.2 Interface Requirements

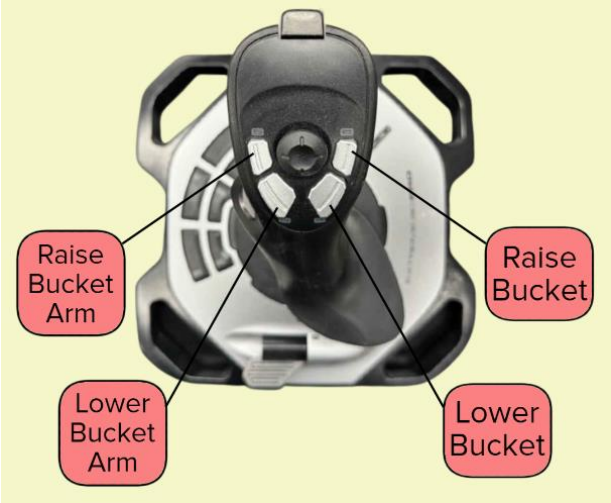
Joystick

Pushing forward on the joystick must drive the motors forward, while pushing backward on the joystick must drive the motors backward. Pushing left on the joystick must vector the motor speed to the left by having the left motors turn slowly and having the right motors turn quickly. Pushing right on the joystick must vector the motor speed to the right, using the same method in the opposite manner. The arms and the bucket of the robot are controlled by four buttons on the top of the joystick.

- The top left button raises the arm of the bucket.
- The bottom left button lowers the arm of the bucket.
- The top right button raises the bucket component.
- The bottom right button lowers the bucket component.

All inputs are processed by the controller code. This code formats the message for the robot and, in its current state, transmits the variably sized message to the robot over TCP. In the future, this will be revised to use fixed-size messages over UDP.

A visual overview of the joystick controls can be seen below:

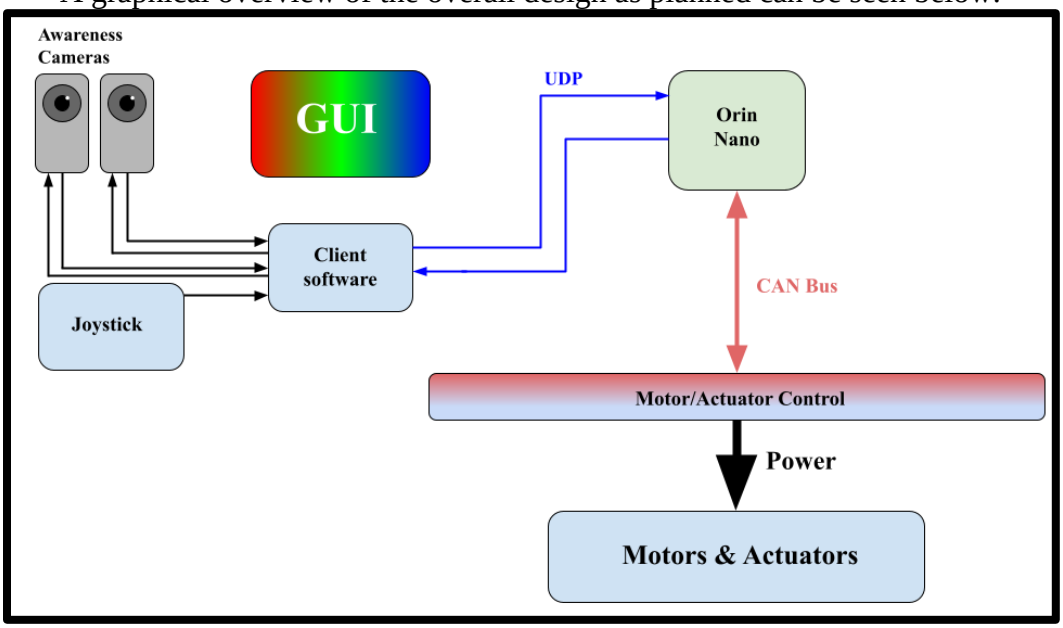


4.2.3 Performance Requirements

NASA’s criteria outline several performance requirements emphasizing the robot's lightness, speed, autonomy, and power efficiency. One key objective is for the robot to achieve twenty consecutive minutes of operation while performing tasks such as navigation and excavation. Additionally, the robot must demonstrate the ability to excavate more than 4351 cc/Wh of BP-1 from the arena. Achieving this goal will reflect significant improvements in the robot's design and functionality compared to last year’s performance.

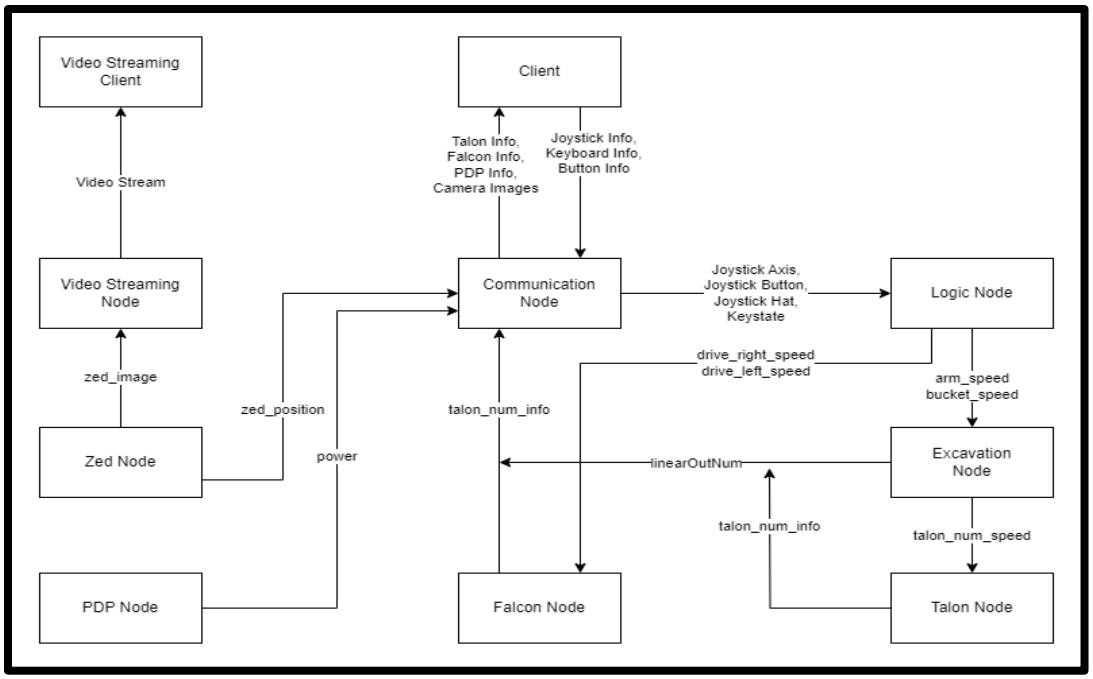
4.3 High-Level Architecture

A graphical overview of the overall design as planned can be seen below:

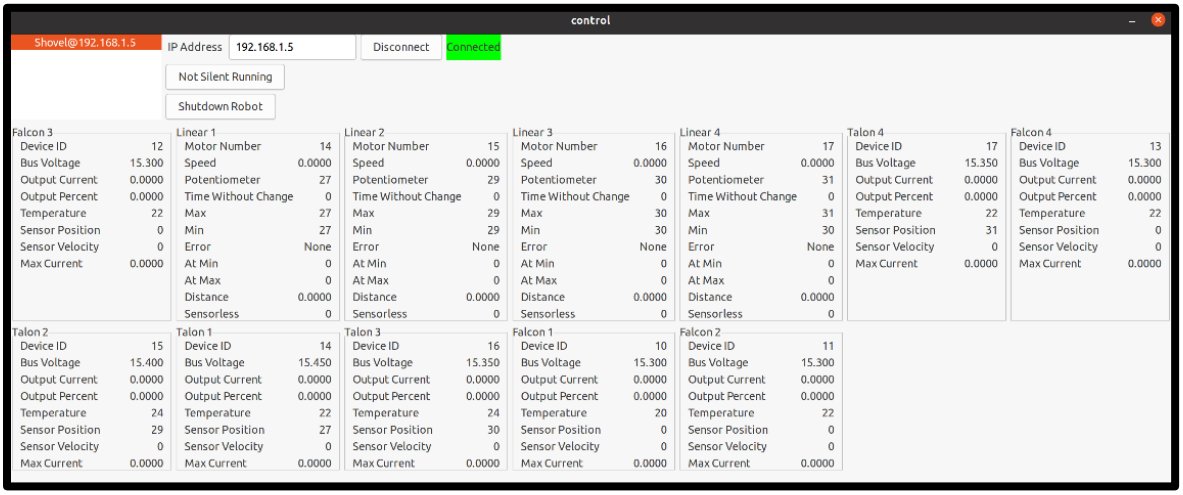


4.4 Detailed Architecture

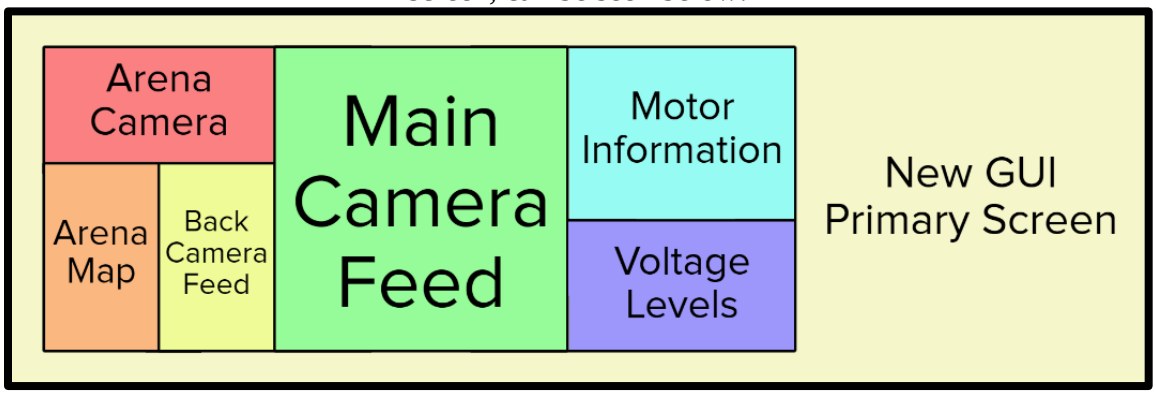
A graphical overview of the ROS2 node design can be seen below:

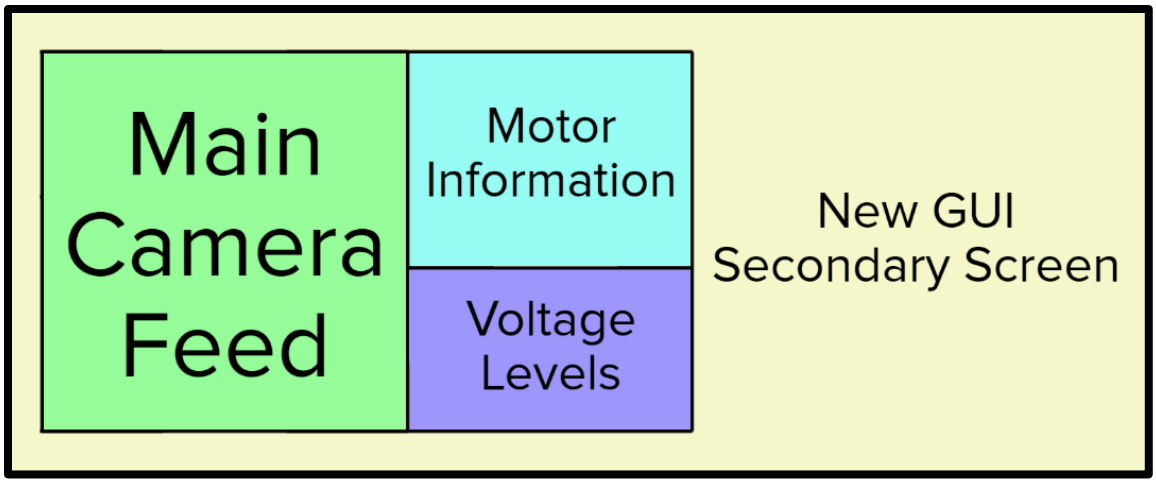


The robot's control GUI in its current state can be seen below:

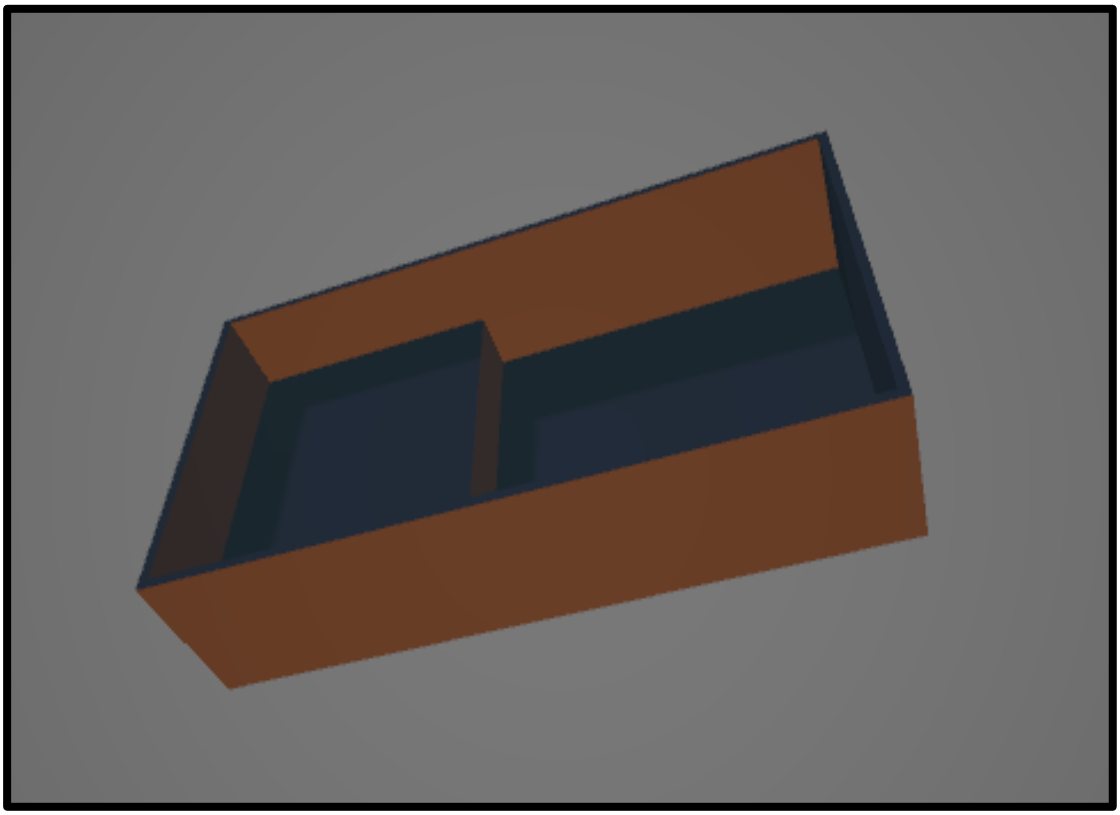


The robot's revamped control GUI mock-up, consisting of a primary screen and secondary screen, can be seen below:





The planned simulation model mockup can be seen below:



4.5 Data Management and Storage

The project does not require any databases or database schemas. The robot processes and reacts to real-time data received from various sensors and datalines, with all processed data being temporary and never stored. The only persistent components are the code files (C++, Python, Make, etc.) necessary to operate the robot. These files are stored in the official Razorbotz GitHub repository, ensuring all team members have access to a shared version of the project. This setup allows team members to make personal changes without impacting others' work.

The GitHub repository is organized into nine distinct sections: communication, excavation, falcon, logic, messages, power_distribution_panel, talon, video_streaming, and zed_tracking. For

instance, the networks team works exclusively in the communication section when updating the protocol. This structure ensures that changes are isolated to the relevant sections, preventing unintended modifications in other areas. The GitHub repository also hosts the simulation files created by the simulation team. This allows all team members to access the simulations and conduct their own tests independently, even without access to the physical robot.

5.0 Development Plan

5.1 Tasks

The current plan involves splitting the team into sub-teams for different tasks. One task involves improving the driver's experience. The sub-team assigned to this task consists of Ryan and Landon. This involves improving the GUI to make it easier to understand when controlling the robot. The current GUI is not effective in providing the driver with the necessary information. It is cluttered, doesn't prioritize the right information, and lacks camera feeds.

The current plan for the GUI is to develop it using a C++ graphical library called GTKMM and a development library called SDL. GTKMM was chosen as the main GUI library as it is relatively low-cost and integrates well with the current C++ GUI and ROS2 library. Overall, it is a very solid library for implementation. SDL should also allow for video playback, which would be beneficial for the multiple camera feeds that are planned.

Another task involves improving the reliability of the robot. The sub-team assigned to this task consists of Joseph and Blake. During the last competition, there was severe electromagnetic interference that caused the robot to disconnect from Wi-Fi for several minutes. The current plan is to increase the reliability of the robot's electronics through redundancy and to improve the robustness of the communications system. A secondary controller will be added and configured for use as a failsafe in case of the failure of the primary controller. External antennas with higher gain will be added to increase signal strength. The networking software will be extensively revised with new protocols and message formatting.

The final task involves creating a simulation of the robot maneuvering throughout the play area. The sub-team assigned to this task consists of Max and Kevin. At the moment, testing the robot requires taking it to a remote testing location, which requires all teams to meet up together, uses up time and money, and causes potential wear to the robot. A simulation program would allow for new autonomy modifications and driver practice without needing to transport the robot to the remote testing location. The current plan for the simulator design is a program called Gazebo with the ROS2 library.

5.2 Schedule

Task Description	Assigned Members	Est Start	Est End	Actual Start	Actual End
Technical Requirement Review	All members	9/11	9/28	9/11	9/28
Systems Requirement Review	All members	9/15	9/20	9/18	9/30
Simulation Application Overview	Maxwell Thursby, Kevin Zheng	1/15	1/22		

UARK Test Bed measurements	Maxwell Thursby, Kevin Zheng	1/25	1/25		
UARK Test Bed Model Development	Maxwell Thursby, Kevin Zheng	1/27	2/10		
UCF Test Bed Model Development	Maxwell Thursby, Kevin Zheng	2/11	3/4		
NASA Test Bed Model Development	Maxwell Thursby, Kevin Zheng	3/5	3/26		
Read ROS2 Documentation and Practice with ROS2 Tutorials	Joseph Folen, Blake Williams	10/15	12/1	10/15	
Design New Protocols and Revise Networking Code	Joseph Folen, Blake Williams	1/15	2/28		
Stress Test and Refactor Networking Code	Joseph Folen, Blake Williams	2/29	4/30		
GTKMM Documentation overview	Ryan Cheng, Landon Reynolds	1/15	1/21		
Current GUI modifications	Ryan Cheng, Landon Reynolds	1/22	2/5		
Motor and Primary Camera GUI Implementation	Ryan Cheng, Landon Reynolds	2/6	2/19		
Back Up Cameras and Arena Camera GUI Implementation	Ryan Cheng, Landon Reynolds	2/20	3/13		
Mini Map GUI Implementation	Ryan Cheng, Landon Reynolds	3/14	4/4		
Competition TBD	All members	TBD	TBD	TBD	TBD

Simulation Group Timeline:

For the simulation group, there are three separate test bed models that need to be developed. Before starting development on any of the test beds, the group must first get familiar with the simulation application. This application will be used to develop the test beds. During the first week, the group will focus on learning and getting familiar with the application. The following week, the simulation group will travel to the UARK test bed location to measure the length, width, and height of the test bed. The week after obtaining the measurements of the UARK test bed, the group will begin developing the simulation model. The plan is to complete the UARK test bed simulation in two weeks. Once the UARK test bed model is finished, the group will proceed with the UCF test bed model. Measurements for this model have already been provided, and the group plans to complete it in three weeks. The final test bed model the group will work

on is the NASA test bed. Since the measurements are also already provided, the plan is to complete the NASA test bed model in three weeks.

Camera Bandwidth/Reliability Group Timeline:

Improving reliability and removing bandwidth restrictions will occur in two phases. The first phase focuses on rewriting the entire networking codebase and removing the previously needed bandwidth requirements. Initially, the networks team will review and practice with the ROS2 software suite. Once the team has a solid understanding of the suite, they will analyze the existing networking codebase and document all necessary details, including how the current protocol works, existing issues with the codebase, and areas for improvement. With this understanding, the team will rewrite the codebase from the ground up using a new communication protocol.

After completing the new codebase, the team will move to the second phase, which involves improving reliability. Stress tests will be developed to push the protocol to its limits, enabling the team to isolate potential vulnerabilities and weaknesses. This phase is expected to be more time-consuming due to the uncertain number of vulnerabilities and limitations in the newly crafted software. The goal is to complete the new protocol by mid-February, allowing two and a half months to focus on reliability. The timeline is expected to conclude by the end of April.

Graphic User interface Group Timeline:

To enhance the aesthetics and functionality of the GUI, the plan includes consolidating data to increase readability, adding multiple camera views, and incorporating a mini-map. To accommodate the crucial data that needs to be presented to the user, three specialized screens will be implemented to display motor, controller, and power information.

To start, the team will study the GTKMM documentation, as neither member is currently familiar with this library. The third week of January will be spent learning the basics of graphical interface construction in GTKMM, such as displaying data on the screen and updating real-time data consistently. Over the next two weeks, the team will begin modifying the current GUI and integrating the three-screen display. In the subsequent two weeks, the motor and primary camera feeds will be added on top of these previous additions. The following three weeks will be dedicated to incorporating a backup camera and an arena camera. Finally, in the latter half of March, a mini-map will be added as the final major feature.

5.3 Deliverables

- Final Project Report – A final report to contain thorough documentation of work done, technical implementation, and project outcomes.
- GitHub Repository – The central repository containing all the C++ code for robot operation and communication, Python node files, and CMake scripts.
- Project Box Folder – A folder that will contain our project overview documents. The document that will be included in the folder will be sponsor information, project summary, meeting location, meeting schedule, project task list, preliminary proposal, and final proposal.

6.0 Key Personnel

Team Members:

Ryan Cheng – Cheng is a senior Computer Engineering major in the Electrical Engineering and Computer Science Department at the University of Arkansas. Cheng is responsible for refinement of the GUI.

Joseph Folen – Folen is a senior Computer Engineering major in the Electrical Engineering and Computer Science Department at the University of Arkansas. Folen is responsible for revising the communication system and protocols for improved reliability.

Landon Reynolds – Reynolds is a senior Computer Engineering major in the Electrical Engineering and Computer Science Department at the University of Arkansas. Process Control Intern; worked with PLC's data scraping. Reynolds is responsible for refinement of the GUI.

Maxwell Thursby – Thursby is a senior Computer Science major in the Electrical Engineering and Computer Science Department at the University of Arkansas. Thursby is responsible for the implementation of the simulation software as well as the optimization of the electrical components of the robot.

Blake Williams – Williams is a senior Computer Science major in the Electrical Engineering and Computer Science Department at the University of Arkansas. Williams is responsible for revising the communication system and protocols for improved reliability.

Kevin Zheng – Zheng is a senior Computer Science major in the Electrical Engineering and Computer Science Department at the University of Arkansas. Zheng is responsible for implementing the simulation software and optimizing the electrical components of the robot.

Sponsor:

Grace Harding – Harding is a Computer Engineering major in the Electrical Engineering and Computer Science Department at the University of Arkansas and Razorbotz computer science team lead.

Faculty:

Uche Wejinya, Ph.D. – Dr. Wejinya is a Mechanical Engineering assistant professor at the University of Arkansas and our primary faculty sponsor.

Assistant Team Member:

William "Andrew" Burroughs - Burroughs is a Computer Science Ph.D. student at the University of Arkansas and Razorbotz CS Technical Advisor. Andrew was responsible for the initial creation of the codebase.

7.0 Facilities and Equipment

Most team activities will take place in the MEEG building's robotics lab and the JB Hunt building at the University of Arkansas. We will use the computers in the lab to help develop our simulation models. Occasionally, we would also use the remote testing location when we work on testing out the automation and networking tasks. The only major pieces of equipment being used are our computers, which have ROS-2 installed into an Ubuntu operating system within a virtual machine. For running networking code, a USB Wi-Fi router is kept in the lab.

8.0 References

[1] Razorbotz. (2024). *communication_node.cpp.*

https://github.com/Razorbotz/ROS2/blob/master/shovel/src/communication/src/communication_node.cpp

[2] Open Robotics. (2024). *ROS 2 Documentation.*

https://docs.ros.org/en/galactic/index.html

[3] Open Robotics. (2024). *Simulation.*

https://docs.ros.org/en/galactic/Tutorials/Advanced/Simulators/Simulation-Main.html