

**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, power-efficient, and robust lunar rover. Each robot is graded based on its weight, power consumption, and volume of dirt shoveled. The objective is to create a robot with autonomous locomotion, navigation, and excavation.

These objectives will be met by solving the issues the previous year’s robot had. At competition, the robot had an outage that lasted several minutes. Whether in a controlled environment or roving across the lunar surface, consistency and reliability is crucial. The main reason for these issues lies in the instability and weakness of the robot’s communication system. The thousands of signals emitted by other competitors’ devices would interfere with the electronics on board. To remedy this issue, the implementation of a signal amplifier will drown out noise and sharpen the connection between devices. Secondly, the criteria used to judge each robot places heavy emphasis on autonomy. From transporting around the track to excavating raw materials, the robot needs to exhibit complete independence. Next, to test and improve additional software and hardware features, the robot needs to be shuttled to the testing grounds. This takes time and gradually wears down the robot’s mechanical parts. This problem can be solved with simulation software that can be booted remotely. Having a fully integrated emulation program expedites the development process and keeps the robot in pristine condition. Finally, the robot’s graphical user interface, or GUI, lacks structure, information, and interaction. Vital information like speed, camera feed, and temperature needs to be displayed to the controller. Currently, the GUI is not intuitive and does not display information in a concise order. Overall, these implementations to the source system would develop a competitive robot that balances weight and power, is fully autonomous, is easy to test, and is user-friendly.

## 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 a bug in the communication system causing the driver to lose control over 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 large amounts of electromagnetic interference at the venue, leading to a similar loss of control. 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 offsite locations to perform testing. The robot also lacks any autonomous functionality, which is vital to scoring during the competition. The GUI also needs significant revision, as it is difficult to read, doesn’t display critical information, and doesn’t support multiple camera feeds like we need it to. Lastly, our robot in its current configuration would be dependent on NASA’s arena cameras for arena awareness, which would lead to deducted points.

## 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 on implementing 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 on revising the design of the GUI 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.

## Background

### Key Concepts

**BP-1**

BP-1 (Black Point 1) is a crushed basalt lunar regolith simulant. It is meant to simulate the loose deposits of rock that characterize the lunar surface.

**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 computer 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 CAN Bus interface. This is a standard commonly used in automotive wiring but 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 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’s currently being used.

### Related Work

In previous years, the same networking/communication protocol has been used to transmit data and information to and from the robot [1]. That being said, the current implementation does not check for bit errors. Problems can arise when interference modifies the data in transit. In its current implementation, if bad data is received, it is not thrown out. Instead, it is processed and used. This causes issues with the GUI and, in some cases, causes it to fail. This will be improved upon by rebuilding the networking/communication protocol from the ground up. This includes a new protocol for transmitting/receiving data, new binary data structures for sending information, and allowing more throughput over the network. The new protocol will include parity checking to verify that the information received is the same as what was sent. 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. The problem with this is that the software developers only get a couple of hours to test everything. This leads to potential errors being overlooked, as there is not enough time to properly check all components of the robot. To fix this, ROS2's [2] simulation software [3] will be utilized to construct a 1-to-1 simulation of the robot. This will allow the software developers to run multiple tests within a day instead of just the one test that is currently used.

## Design

### Use Cases

The robot will be fully autonomous most of the time. When necessary, a driver will be able to manually control the robot over Wi-Fi, using live camera feeds to navigate the environment. 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.

### Requirements

### 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**.
* 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.
* There must be one person per 20kg robot mass carrying the robot.
* All parts of the robot must be possible for the team to control at all times.
* The robot cannot use any touch sensors to avoid obstacles.

### Interface Requirements

**Joystick**

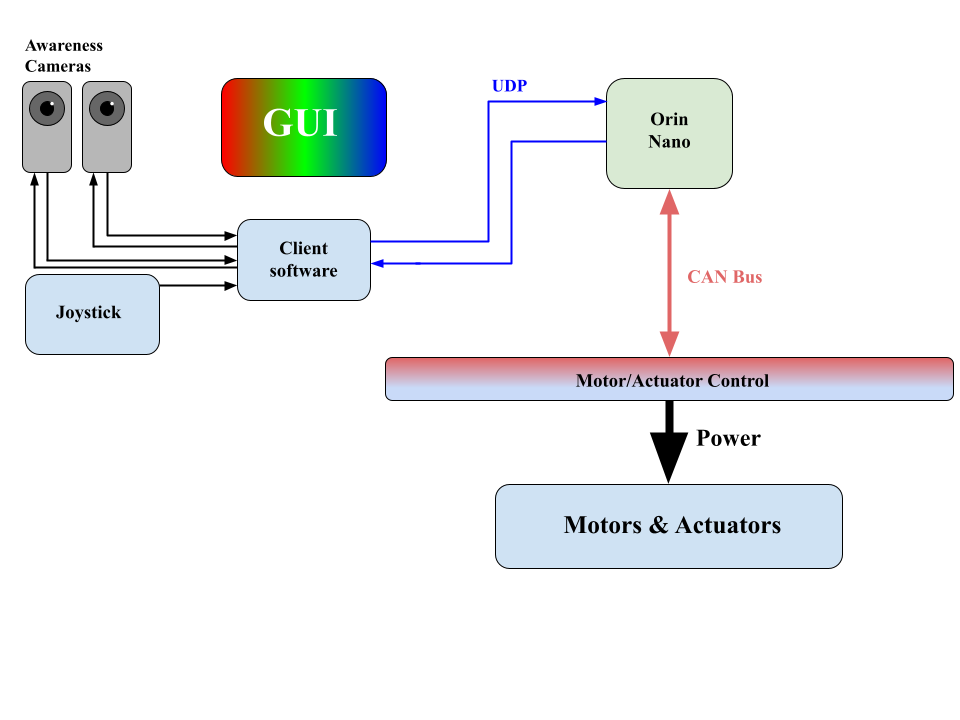
Pushing forward on the joystick must drive the motors forward. Pushing backward on the joystick must drive the motors backward. Pushing left on the joystick must vector the motor speed to the left. This is done by having the left motors turn slowly and having the right motors turn quickly. Pushing right 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 inputs are processed by the controller code. The controller code formats the message for the robot, and in its current state, sends the *variably* sized message to the robot over *TCP*. (Later this will be revised to fixed-size messages over UDP.)

### Performance Requirements

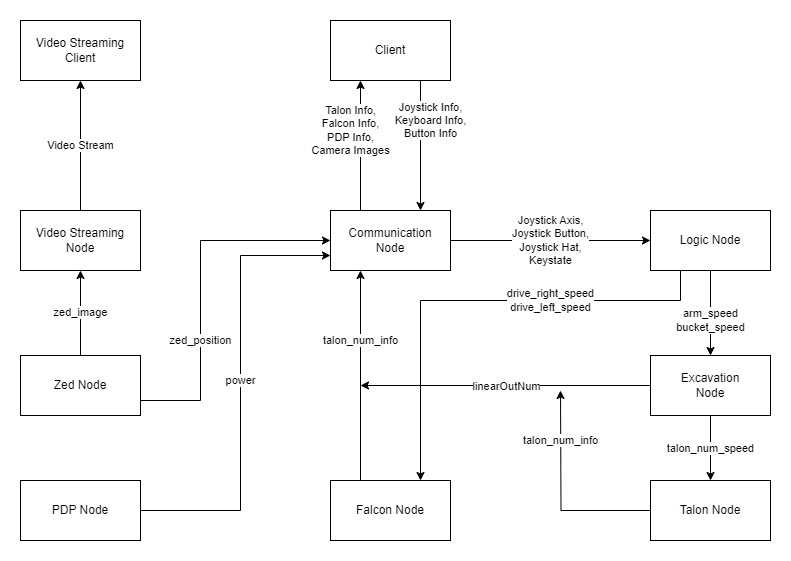
NASA’s criteria outline several performance requirements based on how light, quick, autonomous, and power efficient a robot is. An objective regarding the robot’s run time is to achieve 20 consecutive minutes of operation (navigation, excavation, ect.). Another performance necessity is excavating more than 4351cc/Wh of BP-1 from the arena. Meeting this goal will show improvement in design and functionality when compared to last year’s robot.

### High-Level Architecture

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



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

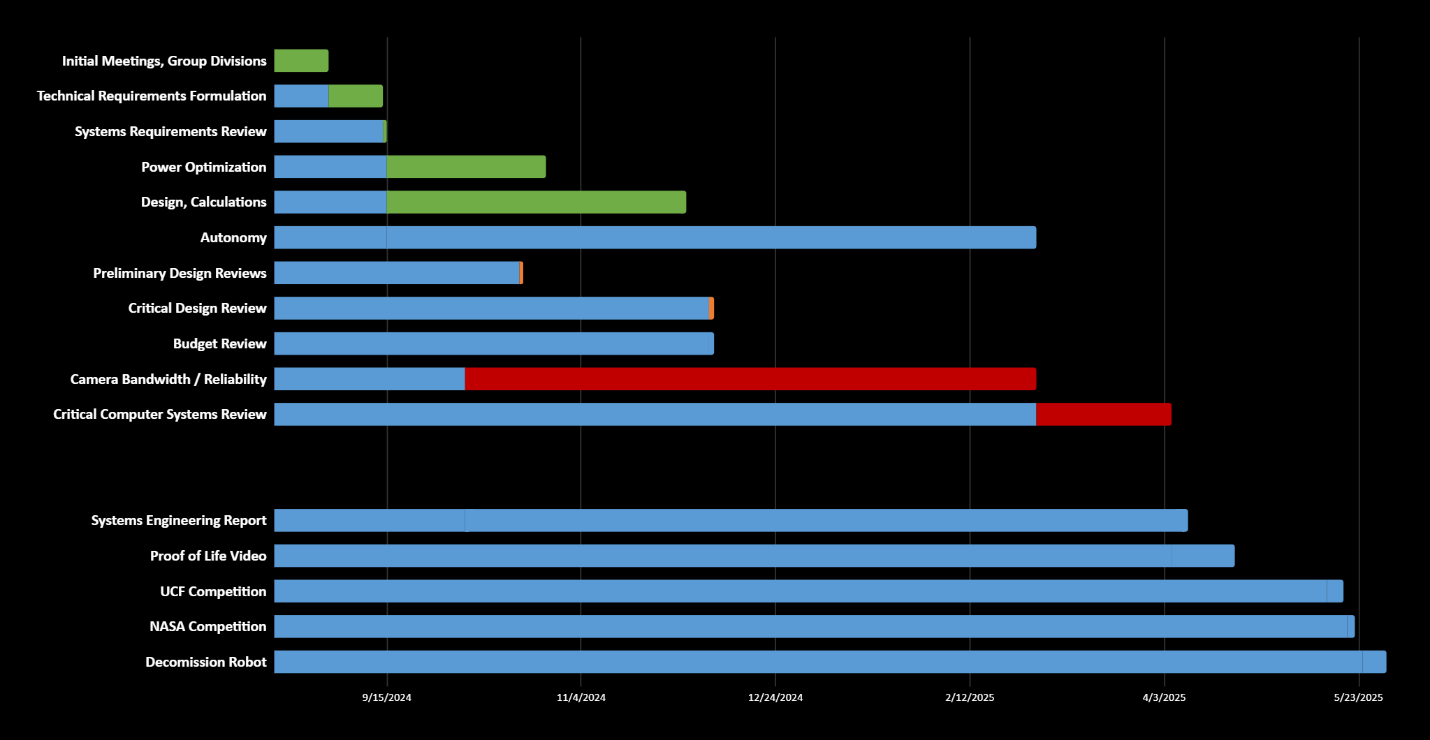


### Development Plan

### Tasks

The current plan involves splitting the team into sub-teams for certain tasks. One task involves trying to improve the driver's experience. The sub-team assigned to this task is Ryan and Landon. This involves improving the graphical user interface, or GUI, to be much easier to understand when controlling the robot. The current GUI is not effective in providing the driver with the necessary information. It is hard to read, doesn’t provide enough information, and needs more 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, implements well into the current C++ GUI and ROS2 library, and is a very solid library overall for implementation. The 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 is Joseph and Blake. During the last competition, there were severe signal interferences that caused the robot to be unable to function for several minutes. The current plan is to increase the reliability of the electronics of the robot. This involves adding a secondary controller in case of the failure of the primary system and controller, increasing the reliability of the signal so video can be streamed from the robot, adding external antennas with higher gain for the Orin processor within the robot, and potentially implementing the ability to switch between a 2.4GHz and 5GHz signal in case of interference. The final task involves creating a simulation of the robot maneuvering throughout the play area. The sub-team assigned to this task is 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 to occur without having to drag out the robot to the remote testing location. The current plan for the simulator design is a program called Gazebo with the ROS2 library.

### 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 | Maxwell Thursby, Kevin Zheng | 1/20 | 2/15 |  |  |
| Camera Bandwidth/Reliability | Joseph Folen, Blake Williams | 2/30 | 3/27 |  |  |
| Graphic User Interface | Ryan Cheng, Landon Reynolds | 2/15 | 3/15 |  |  |
| UCF Competition | All members | 5/7 | 5/14 |  |  |
| NASA Competition | All members | 5/14 | 5/21 |  |  |

### 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.

## 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 as well as the optimization of 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, Razorbotz computer science team lead and president of the Razorbotz club.

Faculty:

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

## Facilities and Equipment

For the most part, the facilities we will use for development are our usual meeting areas, which are the MEEG building and the JB Hunt building at the University of Arkansas. Occasionally, we would also use the remote testing location when we work on testing out the autonomation and simulation tasks. As for equipment, we are utilizing our own computers, which have had ROS-2 installed into an Ubuntu operating system within a virtual machine.

## References

[1] Razorbotz. (2024). *communication\_node.cpp.*

[https://github.com/Razorbotz/ROS2/blob/master/shovel/src/communication/src/communi](https://github.com/Razorbotz/ROS2/blob/master/shovel/src/communication/src/communication_node.cpp) cation\_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>