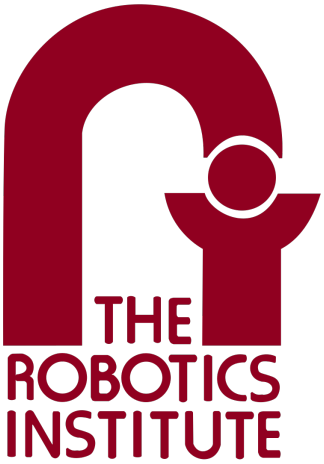
# Individual Lab Report - 07




# Lunar ROADSTER

Team I

Author: **Bhaswanth Ayapilla**
Andrew ID: bayapill
E-mail: *bayapill@andrew.cmu.edu*

Teammate: **Deepam Ameria**
ID: dameria
E-mail: *dameria@andrew.cmu.edu*

Teammate: **Boxiang (William) Fu**
ID: boxiangf
E-mail: *boxiangf@andrew.cmu.edu*

Teammate: **Simson D'Souza**
ID: sjdsouza
E-mail: *sjdsouza@andrew.cmu.edu*

Teammate: **Ankit Aggarwal**
ID: ankitagg
E-mail: *ankitagg@andrew.cmu.edu*

Supervisor: **Dr. William "Red" Whittaker**
Department: Field Robotics Center
E-mail: *red@cmu.edu*

September 25, 2025

Carnegie Mellon University
Robotics Institute

# Contents

# 1 Individual Progress

## 1.1 Computations

### 1.1.1 Software Fix for Arduino Reset Issue

Since last semester, our team has encountered recurring issues with the Arduino failing to reset properly during operation. Data packets are sometimes dropped between the Arduino and Jetson, leading to stalled commands and the rover not being able to move. Previously, we mitigated the problem by manually pressing the Arduino's reset button whenever communication broke down. However, this is not the most efficient fix and did not work always, as evident from our SVD Encore. To address this we designed a software-driven reset mechanism, which is yet to be implemented.

Our solution leverages the Jetson Orin's GPIO pins to trigger the Arduino RESET line. After going through several NVIDIA Forums, I found that the 40-pin GPIO header provides 3.3V/1.8V max. But the Arduino RESET pin requires $5V$. Hence we need to pass it through a step-up circuit.

**Hardware setup:**

1. Jetson GPIO (3.3 V) $\rightarrow$ 10k$\Omega$ resistor $\rightarrow$ NPN transistor base
2. Transistor collector $\rightarrow$ Arduino RESET
3. Transistor emitter $\rightarrow$ GND (shared)
4. Arduino RESET has built-in pull-up $\rightarrow$ remains HIGH during normal operation
5. When GPIO drives transistor $\rightarrow$ RESET pulled LOW $\rightarrow$ Arduino restarts

**Software setup:**

1. Arduino (micro-ROS): Publishes `/arduino_heartbeat`
2. Jetson (ROS2 C++ node):
    - Subscribes to `/arduino_heartbeat`
    - If no heartbeat is detected for $>$ 5,s, the Jetson triggers a reset by toggling the GPIO line via `libgpiod`.
    - A $\sim$ 200ms LOW pulse on the RESET line forces the Arduino to restart, restoring communication automatically.

### 1.1.2 Global Path Planner Methodology

I worked on finalizing the global path planner methodology, and I will be implementing it for PR9. The global path planner for our rover is designed to follow a predefined circular "ring" trajectory around the crater field while safely navigating obstacles. The ring itself is constructed using crater detection results from Simson's algorithm. Centroids of the detected craters are first extracted, and all large craters (greater than 0.5,m in diameter) are excluded as obstacles. The remaining crater centers are sorted in counter-clockwise order, and a spline is fit through them to generate a smooth reference path, which serves as the nominal ring corridor.

**Planner and Motion Model:** The planner is based on a Hybrid-A* lattice search that explicitly models the rover's Ackermann steering geometry. Feasible motion primitives are

generated as forward-only Dubins arcs, which enforce the rover's minimum turning radius. Although reverse maneuvers are permitted, they are heavily penalized to discourage unnecessary backward motion. Directionality is constrained to counter-clockwise motion, because we will be demonstrating the grading operation only in one direction.

**Cost Function (g-cost):** The accumulated path cost reflects multiple factors relevant to rover navigation:

- Arc length: shorter paths are favored.
- Curvature penalty: sharp steering is discouraged by penalizing tight turns.
- Obstacle cost: proximity to large craters adds significant cost, pushing the rover to take safer detours.
- Ring deviation: paths that drift away from the reference ring corridor are penalized, encouraging solutions that hug the ring trajectory.

**Heuristic (h-cost):** To guide the search efficiently, the heuristic combines two terms:

- The Dubins shortest-path distance to the goal, which respects the rover's nonholonomic turning constraints.
- A small additional penalty for misalignment with the ring, biasing the rover toward paths that stay consistent with the circular reference.

**Outputs:** The resulting global plan is a collision-free, curvature-feasible path that follows the ring while smoothly bypassing large crater obstacles. The path naturally passes through crater centers along the ring trajectory and rejoins the ring corridor after detours.

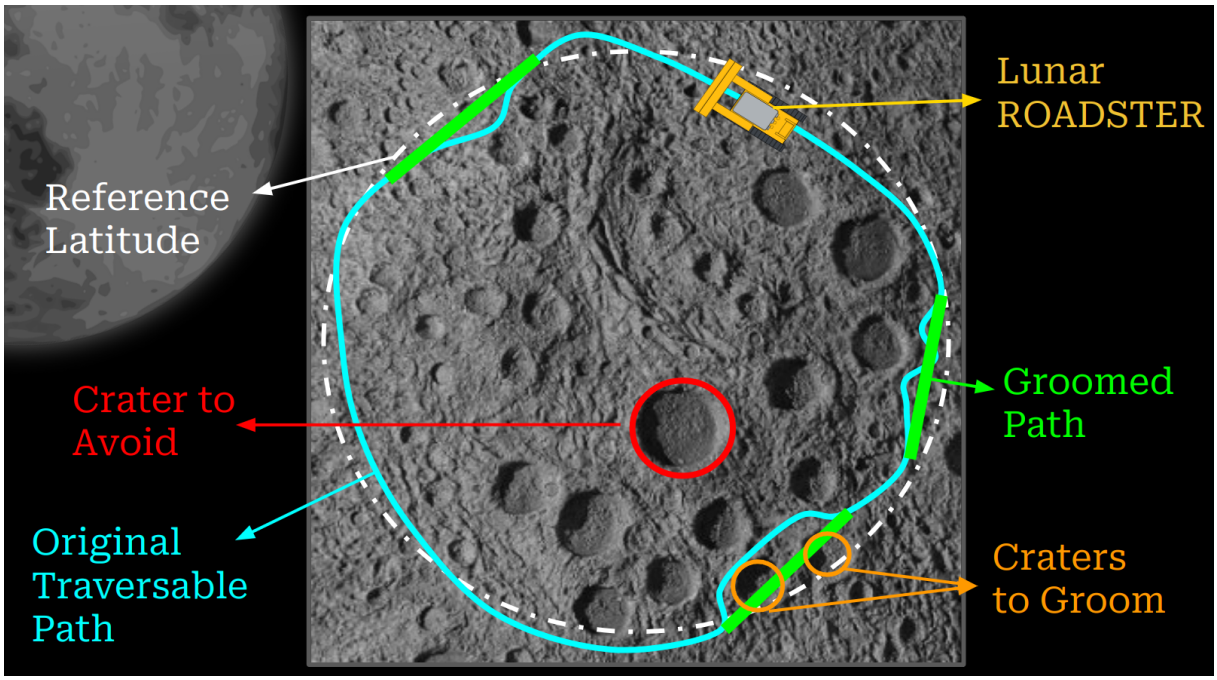



**Figure 1:** Global path planner hugging the ring in blue

### 1.1.3 Global Controller Methodology

I worked with Simson in finalizing the methodology for the global navigation controller, which will be based on Pure Pursuit. It is responsible for tracking the planned path and

converting it into actionable steering and velocity commands for the rover.

**Workflow:**

1. Receive the global path from the global path planner as a sequence of waypoints.
2. Obtain the current robot pose from the localization stack.
3. Identify a lookahead point located at a fixed distance $L$ along the path ahead of the robot.
4. Convert the lookahead point from the global frame into the robot's local coordinate frame.
5. Compute the curvature required to reach the lookahead point and from it derive the steering angle. The forward velocity command is coupled with the steering output to ensure stable tracking.
6. Send the computed steering and velocity commands to the rover's actuators.
7. Repeat steps 2–6 until the final goal is reached.

## 1.2 Electronics

### 1.2.1 Sourcing New IMU

In the previous PR, I mentioned that I was exploring IMU options for our rover. After discussions with several vendors, we decided that the same VectorNav VN-100 was the best fit for our requirements. The order has been placed and the unit arrived a few days ago. We will mount and integrate the new IMU during our next test, this Saturday.


**Figure 2:** VectorNav VN-100

### 1.2.2 Hardware Maintenance

Our team performed the hardware maintenance tests in order to do a full run of our Spring Validation Demo. During this process, we encountered issues caused by incorrect wiring connections. I worked with William to perform detailed unit testing, systematically checking every connection between the motors, Roboclaws, Arduino, and power lines. Through this process we identified the root causes: loose encoder connections that prevented proper reading and control of values, reversed motor directions due to incorrect power supply wiring, flipped TX RX connections to the Arduino. Both issues were resolved during testing. To further improve reliability and prevent similar problems in the future, the team placed an order for new motors, connectors, and headers, ensuring more robust and durable wiring.

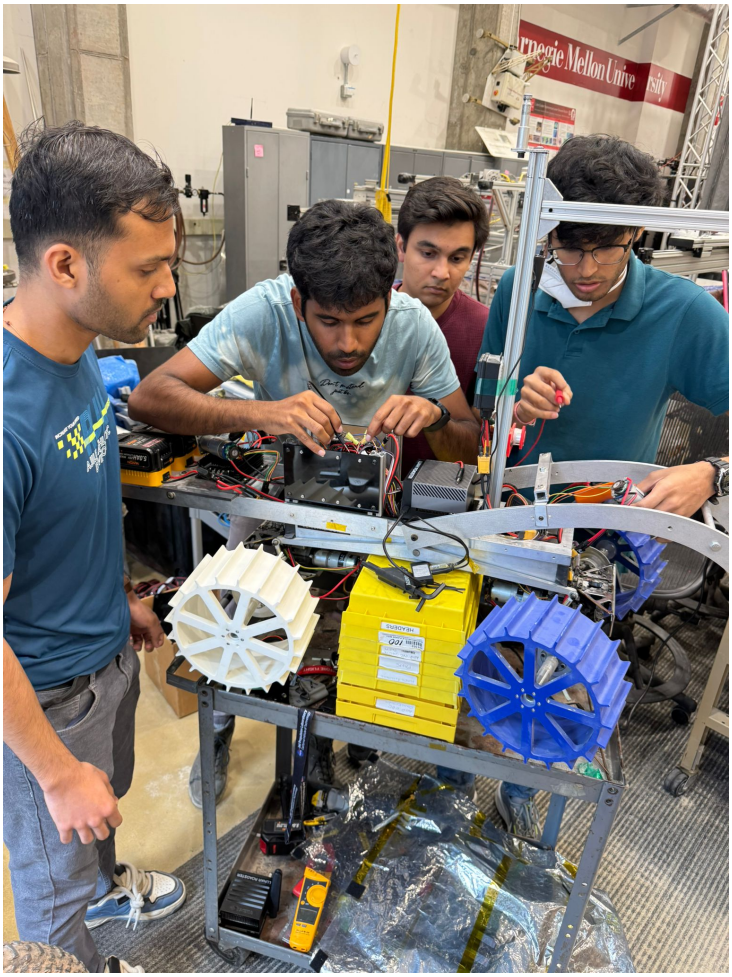


**Figure 3:** Hardware maintenance

## 1.3 External Infrastructure

### 1.3.1 Total Station Resection Method

Last semester, we configured the total station using the "Orientate to Line" method, which uses two prisms to fix a coordinate axis and establishes the total station position relative to them. This approach worked reliably during normal localization runs. However, we noticed that whenever the total station batteries were swapped and the setup had to be repeated, the system introduced slight offsets in the measurements.

To fix this, I implemented the Resection method, which requires three known prism positions in order to precisely determine the total station's location. By observing these points, the total station can triangulate its exact position and orientation within the defined coordinate frame. This eliminates the risk of drift or offset errors associated with single-axis orientation methods, as we did previously. The setup method was originally taught to us by Chuck Whittaker. I documented the entire procedure on our Notion Engineering Wiki page for future reference.

**Validation**:

To confirm the accuracy of the resection setup, I performed a checkpoint test:

1. A prism was placed at a fixed, known location.
2. Its coordinates were recorded with the total station.

3. The total station was then moved to a different position, and resection was recalculated using the same three reference points.

4. Measuring the fixed prism again produced identical coordinates.

This demonstrated that the resection method is highly precise and consistent, regardless of the total station's placement. Figures 4, 5 show this.

**Figure 4:** Total Station Position 1

**Figure 5:** Total Station Position 2

# 2 Challenges

# 3 Team Work

- **Bhaswanth Ayapilla:** My initial work involved working on the resection method for the total station. I worked with the rest of the team in performing hardware maintenance of the rover, and closely collaborated with William in performing unit testing to fix the connection issues. I also worked with Simson in finalizing the

global controller methodology. Additionally, I am working with Ankit and Deepam to debug the Arduino reset issue and implement the software fix.

- **Ankit Aggarwal:** Ankit's work involved finalizing the methodology for the perception stack. He worked with Deepam and finalized a Deep Learning method to detect craters and extract geometry information. They collected a preliminary dataset for the YOLO Model and a video to run inferences. He also worked with Deepam to refine the dozer performance by debugging the jittery motion. Additionally, he manufactured mounts for the ZED and Orin.

- **Deepam Ameria:** Deepam's work involved finalizing the methodology for the perception stack of our system. He worked with Ankit and decided to go ahead with a Deep Learning based method to detect craters and extract geometry information. For this task, they worked together to carry out initial research and also collect preliminary data (images of craters) for training a YOLO model and run inference on a test video. He also worked with Ankit on refining the Dozer. They debugged the problem of jittery motion of the actuator at intermediate positions. He is also working with me and Ankit for solving the Arduino Reset issue.

- **Simson D'Souza:** Simson's work focused on map generation and refining the crater detection logic. He explored different plane-fitting techniques and simpler approaches for identifying gradable craters; however, the original method proved to be more reliable and accurate, so I reverted to it. In addition, he collaborated with me on the navigation planner and contributed to formulating the global navigation controller methodology, which he will soon begin implementing.

- **Boxiang (William) Fu:** William's work since the last progress review focused on debugging software connections between the new Orin compute with the Arduino (in collarboration with me) and joystick. The joystick issue was resolved by installing updated Logitech joystick dependencies and the Arduino problem was resolved by reinstalling MicroROS. He also collaborated with me in resolving the sudden jolts caused by steering commands. Finally, he worked on researching on how to implement the rover's validation unit using point cloud sensor data and elevation mapping.

# 4 Plans

The following are my goals for progress review 9:

1. Implement arduino reset hardware and software fix
2. Finish hardware maintenance
3. Finish new IMU setup
4. Implement the global path planner
5. Implement the global navigation controller along with Simson
6. Implement the local navigation controller