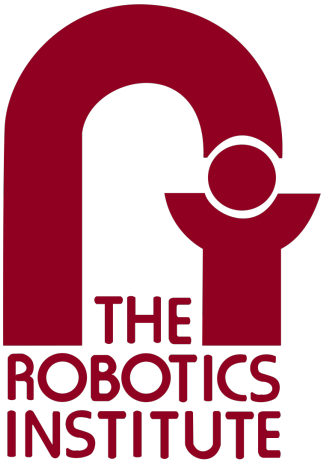
Individual Lab Report 4



Lunar ROADSTER

Team I

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

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

Teammate: **Bhaswanth Ayapilla**
ID: bayapill
E-mail: *bayapill@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*

March 21, 2025

Carnegie Mellon University
Robotics Institute

1 Individual Progress

Since the last progress review, I worked on finalizing the localization and sensor stack of the rover. The code is finished, however there were a number of issues for both the localization and sensor stack, which I will discuss below. Additionally, during testing we noted that the odometry topic was not publishing the encoder feedback. Debugging this took a considerable amount of time.

1.1 Odometry Topic Debugging

During testing of the localization stack, we noticed that some of the odometry feedback readings from the wheel encoders were not being updated. Referring to Figure 1, the `drive_` values were staying constant at zero, while weirdly enough the `steer_` values were being updated correctly. The control commands were also working perfectly and the rover reacts to joystick inputs from the user. The only issue was the feedback of the `drive_` values.

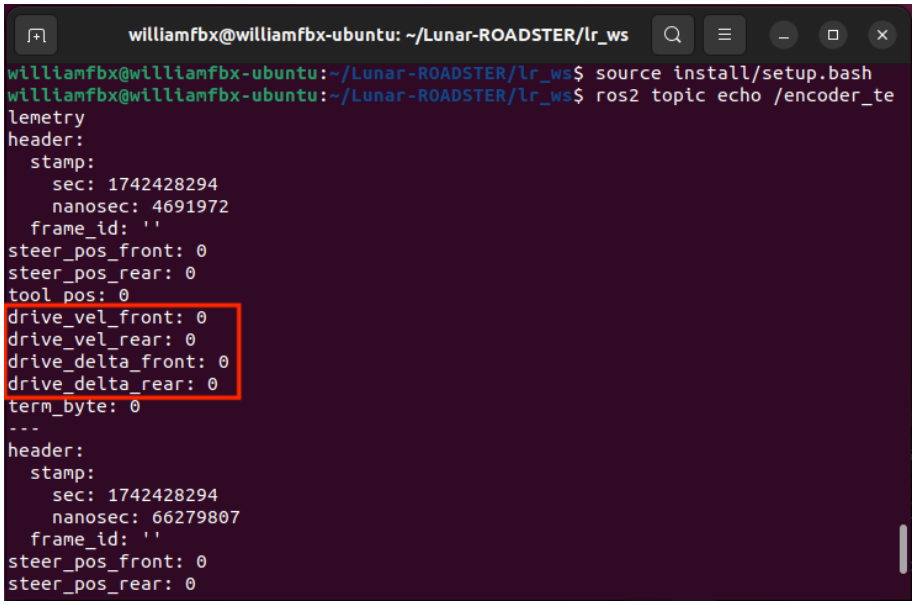


Figure 1: Odometry feedback topic

This issue took a considerable amount of time to debug. We tried the following:

1.1.1 Checking code for Jetson to Arduino interface

We checked the `serial_interface_node` that interfaces the Jetson with the Arduino was functioning correctly. We checked this by examining the converted feedback on the `/encoder_telemetry` topic versus the raw bit-wise feedback on the `/arduino_feedback` topic to narrow down the scope of the issue. We saw that the raw bit-wise feedback on the `/arduino_feedback` topic was not being updated. This meant that the issue was not the Jetson to Arduino interface (which was via the `/encoder_telemetry` topic).

1.1.2 Checking code for Arduino to RoboClaw interface

Since the feedback on the `/arduino_feedback` topic was not being updated, we checked the INO code that interfaces the Arduino to the RoboClaw. An initial hypothesis that we thought was that since we have not implemented the `tool_pos` value yet, the callbacks from the RoboClaw may be invalid or pointing to a Nullptr. This would

cause issues when typecasting into `int8` format, which is the ROS topic format for `/arduino_feedback`. We tried manually setting the `tool_pos` value to 0, but that didn't solve the problem.

1.1.3 Checking the encoder

After going through the code, we next checked the encoders as we thought it might have broke during testing. We wired up the encoders to the RoboClaw pins that `steer_` uses (since it was working correctly) and noticed that we were receiving feedback (albeit not the correct values since the QPPS on the steer and drive motors were different). This meant that the encoders were working correctly.

1.1.4 Checking jumper wires

Finally, we went through the jumper wires connecting the encoders to the RoboClaw. Since we knew the steer jumper wires were correct, we mapped out which pin should connect to which pin (see Figure 2). It turns out that the problem was with the jumper wiring. For the front drive encoder, it turned out that when the hardware team replaced new motors for the rover, the wiring was connected in reverse. This meant that the feedback was unable to be sent back to the correct RoboClaw pins. The front encoder feedback was simply fixed by connecting the jumper wires in reversed order. For the back drive encoder, the wiring was correct, but one of the connections was loose. This was not noticeable initially as the connections were masked over using tape. We only realized the loose connection when taking off the tape.

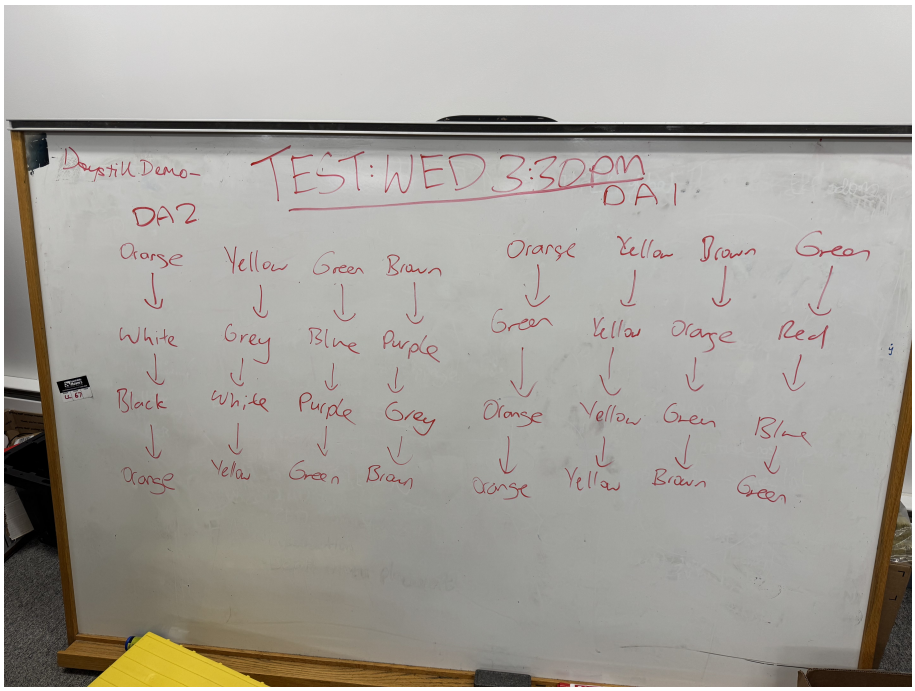


Figure 2: Jumper wire connections

Albeit the simple fix, it took a lot of time and checking to re-establish the feedback from the drive encoders. It took up a considerable amount of time to debug and was a major blocker for my other tasks.

1.2 Localization Debugging

We finished and implemented the code for the localization stack before the previous progress review. However, since the odometry topic was not publishing and it was needed for local localization, we were not able to test and tune the localization stack until the odometry topic debugging was complete. Luckily, we were able to solve the odometry issue and we were able to test our localization stack in the Moon Yard on March 19th. The local localization (`odom` to `base_link` transform) was working correctly, but the global localization (`map` to `odom` transform) was very sporadic and requires debugging. The transform would localize the rover for the first 2-3 seconds, but would then immediately "fly off" in a random direction for some reason (see Figure 3, the left figure corresponds to the initial spawn location, the right figure corresponds to the `odom` frame flying off).

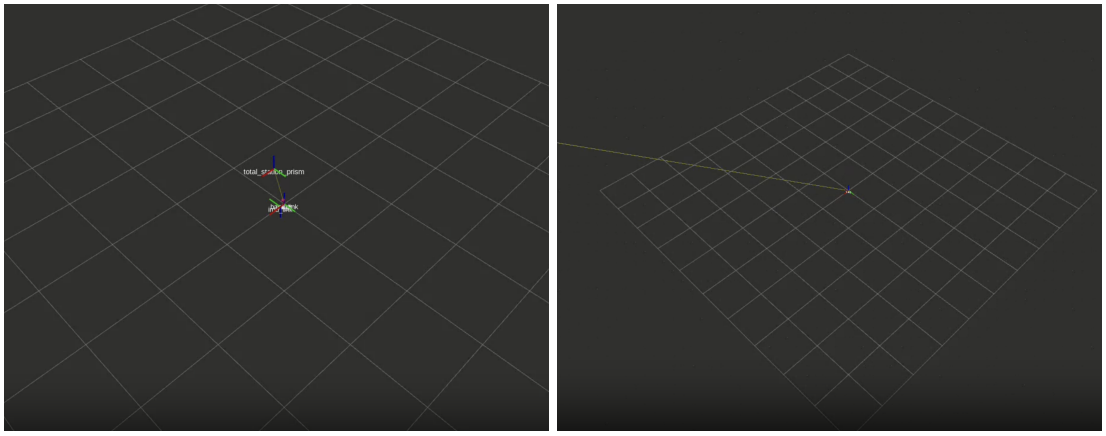


Figure 3: `odom` frame "flies off" due to bug in global localization

We were not able to debug this during our previous test. However, we plan on testing some hypotheses during our next test day on March 21st. One hypothesis is that the TF tree was incorrectly set up with `total_station_prism` as a child frame of `base_link`. This may result in the localization algorithm thinking that whenever a global position is received from the total station (say $\{X, Y, Z\} = \{2, 2, 0\}$), the `total_station_prism` frame is this much away from the `base_link` frame. Since the two frames are rigidly linked, this will also cause `base_link` to move by $\{X, Y, Z\} = \{2, 2, 0\}$ whenever a new total station measurement is received. We have already adjusted our localization code and will test this on March 21st.

1.3 Sensor Stack Integration

The final task that I completed during the period from the last progress review is setting up the sensor stack of the rover. This involved setting up the stereo camera drivers and interfacing the SDK with the Jetson. I initially set up the ZED 2i stereo camera and interfaced it with the `elevation_mapping` ROS package on my development laptop (see Figure 4). However, we could not get it to interface with the docker container on the Jetson. The reason is that the Ubuntu version inside the docker (v22.04) is different than the Ubuntu version on the native Jetson (v20.04). This causes issues with CUDA compatibility as the ZED SDK uses CUDA drivers to post-process the stereo camera feed.

A solution would be to manually install the CUDA dependencies in the correct version requirements. However, this would take a considerable amount of time and would

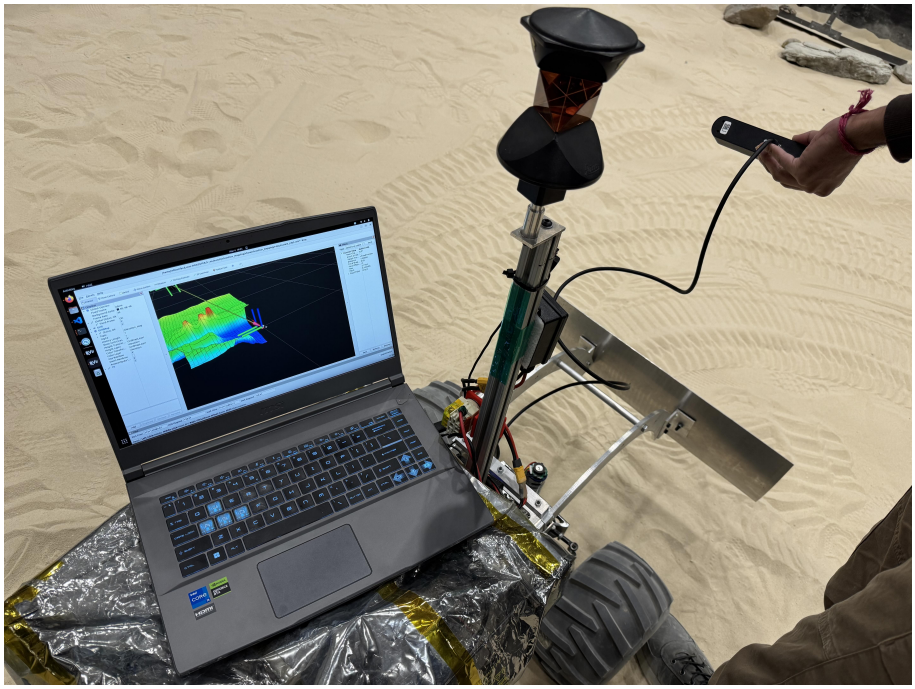


Figure 4: ZED 2i stereo camera setup

be a blocker for our other tasks as active mapping is required for the tool planner, navigation, and validation. Instead, we reverted back to using a RealSense camera as it does not require CUDA drivers and there was documentation for interfacing with a docker container on the Robotics Knowledgebase wiki and prior config files from team LunarX. The setup in the docker container is complete and the point cloud obtained from the stereo camera is correctly being published to a ROS topic.

Finally, I also tested different locations of the depth camera to find an optimal placement location. The finalized location is $\{X, Y, Z, R, P, Y\} = \{0.5, 0, 0.6, 0, -30, 0\}$ from `base_link`. At this location, the height and declination is ideal so that the it is clear of the dozer when fully raised and generates a clear point cloud of obstacles in front. The next step is to convert the point cloud into an elevation map for the tool planner.

2 Challenges

The first major challenge was that debugging the odometry topic of the rover took a considerable amount of time. While I knew the hardware team changed the drive motors, I always expected them to rewire up the jumper wires back correctly. However, this was not the case. The team had to spend a considerable amount of attention checking every unit that makes up the odometry subsystem and unit test each component. This made me fall behind on schedule for my localization subsystem tasks.

Another challenge faced is that bugs still exists in the localization implementation of the rover. We are currently still unit testing each component that makes up the localization subsystem. We have ruled out the local localization as dead reckoning using only the IMU and wheel encoders works as intended. The main culprit for the bug is on the integration of the total station data. Since the localization stack is crucial for other major subsystems, we are working on an expedited schedule to identify the problem and find a solution.

A final challenge was integrating the ZED 2i SDK and CUDA drivers with the docker container. As mentioned previously, the different Ubuntu versions on the host and inside the docker causes compatibility issues with the graphics card hardware. We tried looking for a solution but all involve manually installing the CUDA drivers. In the interest of time, we decided to push back this integration and instead use the RealSense camera which does not require CUDA.

3 Teamwork

A breakdown of the contributions of each team member are tabulated below:

- **Ankit Aggarwal**: Ankit's work mainly focused on the tool planner methodology. Ankit took inputs from the team for insights on the best way to set up the planner to minimize integration issues. He then worked with Simson and Deepam for to set up a manufacturing plan for the E-Box. Ankit also worked on debugging wheel odometry with William. Additionally, Ankit worked with Deepam to mitigate the issue of rover breakdown due to a worn-out rear drive axle.
- **Deepam Ameria**: Deepam's primary work was to try different actuators of varying gear ratios and finalize the best one for our use case. He worked with Bhaswanth on making the tool capable of teleoperation. The oscillations at intermediate positions still need to be debugged. Deepam also collaborated with Simson to develop an ideal terrain by flattening the MoonYard and creating craters of various shapes and sizes, in order to develop a global map using FARO Laser Scanner. Deepam worked together with Ankit to mitigate the issue of the rover breaking down due to a worm out rear axle. We scavenged the spares off a twin rover and successfully replaced it on ROADSTER. He also used the E-Box design made by Ankit to laser-cut the walls of the E-Box at TechSpark. Moving on, Deepam be working with Ankit to develop the tool planner methodology and its software stack. We will be using inputs from the elevation maps created by Simson and William.
- **Bhaswanth Ayapilla**: Bhaswanth's work with William involved testing the localization stack in the Moon Yard. During testing, we realized that the issue is now with the global localization and we are debugging it together. He also worked with Deepam in helping him implement dozer teleoperation. Bhaswanth also worked with Simson on the initial navigation stack setup on our Jetson board, and will be collaborating together more on completing the navigation stack.
- **Simson D'Souza**: Simson worked on refining the global costmap and tuned parameters to obtain an accurate ground plane. To achieve this, Simson collaborated with Deepam to flatten the Moon Yard and create craters of various diameters and depths, allowing for a more precise terrain model. Additionally, he developed an algorithm to identify gradable craters and extract their coordinates, which will be used in navigation. Simson also worked on the navigation stack setup, collaborating with Bhaswanth to configure and integrate it on the NVIDIA Jetson. Furthermore, in collaboration with Ankit, the required parts for E-box manufacturing were finalized.

4 Plans

From now until ILR5, I plan to finish debugging the localization stack. This would primarily involve unit testing the global localization and how the total station data is in-

tegrated into the EKF.

Once this task is finished, I plan to finish the sensor stack for the rover. The end result for this would be an elevation map that is obtained from the RealSense point cloud data. My current implementation methodology is to transform this into the `map` frame that aligns with the coordinates of the Moon Yard. However, depending on the requirements of the tool planner subsystem, this could also be transformed to the local `base_link` frame and have the elevation map be relative to the rover.

Finally, if time permits, I will be working on integrating the various subsystems that I contributed (localization, sensing, external infrastructure, FSM planner) into a coherent system. This allows us to do integration testing in preparation for the Spring Validation Demonstration.