Computer Science & Computer Engineering

# NASA Lunabotics Competition

**CSCE:** Ahmed Moustafa, Jackson Burger, Jackson Newman, Justin Kilgo, & Rohit Kala
**MEEG:** Shane Allder, Chandler Dye, Kaili Henry, Ryan McGuinness-Correa, Matthew Norris, Venkata Panabakam, Lauren Skartvedt, & Shel-Twon Warren
**Sponsors:** Dr. Uche Wejinya & Andrew Burroughs

UNIVERSITY OF ARKANSAS
College of Engineering

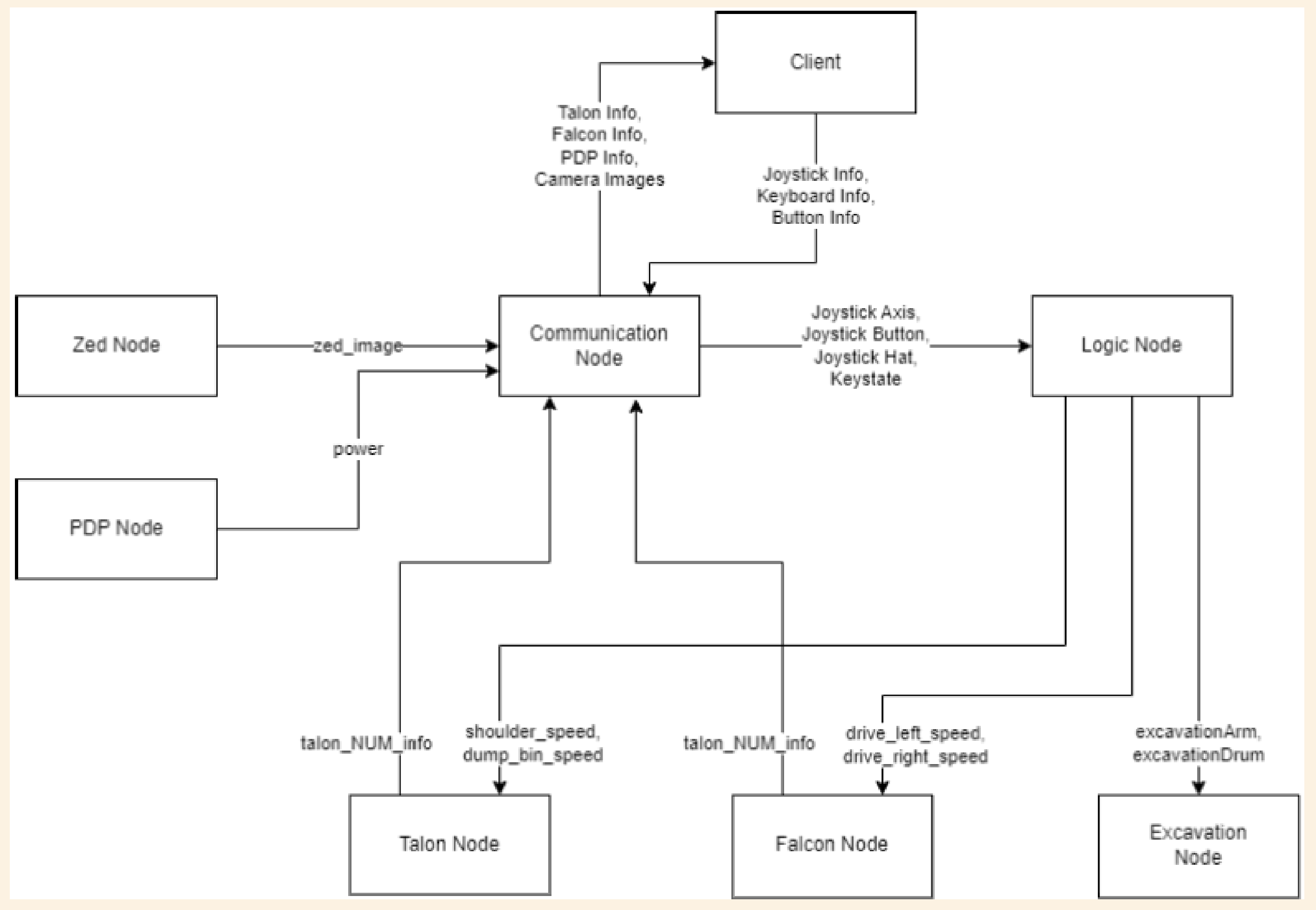
## Introduction

To sustain the growing population, NASA's Artemis program seeks resources off-planet, with the moon being a prime candidate. However, human travel to the moon is both dangerous and expensive, necessitating the development of efficient robots for mining and research. This is where the Lunabotics competition comes in, aiming to design a semi-autonomous lunar excavator that can maneuver through rough terrain and extract regolith. The team will integrate advanced sensors, robust software, and hardware components to achieve this objective safely and reliably.

## Manual Control

The Communication node is a central element of the Lunabotics excavator's software architecture that takes in data from other nodes and publishes data through *rclcpp* APIs to move the robot based on user input. It relies on other nodes such as Client, Logic, Excavation, Falcon, Talon, Zed, and the Power Distribution Panel for data.

The Client transmits joystick information, and the Zed node publishes images to calculate the robot's position and detect obstacles. The Logic node updates the wheel and excavation motor speeds based on real-time information from the Communication node. The Falcon and Talon nodes control motor controllers responsible for vertical motion and height control of the excavation apparatus and bucket system. Lastly, the Excavation node controls the system, allowing for controller input and autonomous excavation macros.


Client
Talon Info, Falcon Info, PDP Info, Camera Images
Joystick Info, Keyboard Info, Button Info
Zed Node
zed_image
Communication Node
Joystick Axis, Joystick Button, Joystick Hat, Keystate
Logic Node
power
PDP Node
talon_NUM_info
shoulder_speed, dump_bin_speed
talon_NUM_info
drive_left_speed, drive_right_speed
excavationArm, excavationDrum
Talon Node
Falcon Node
Excavation Node
*Figure 1: Flowchart of ROS2 Nodes*

## Automation

To ensure that the robot is autonomous, we will primarily use the information that comes from the robot's ZED camera. The camera tracks various information such as the X, Y, and Z coordinates as well as the roll, pitch, and yaw of the robot in addition to the video stream from the camera. We split up the automation of the robot into Driving and Excavation autonomies.

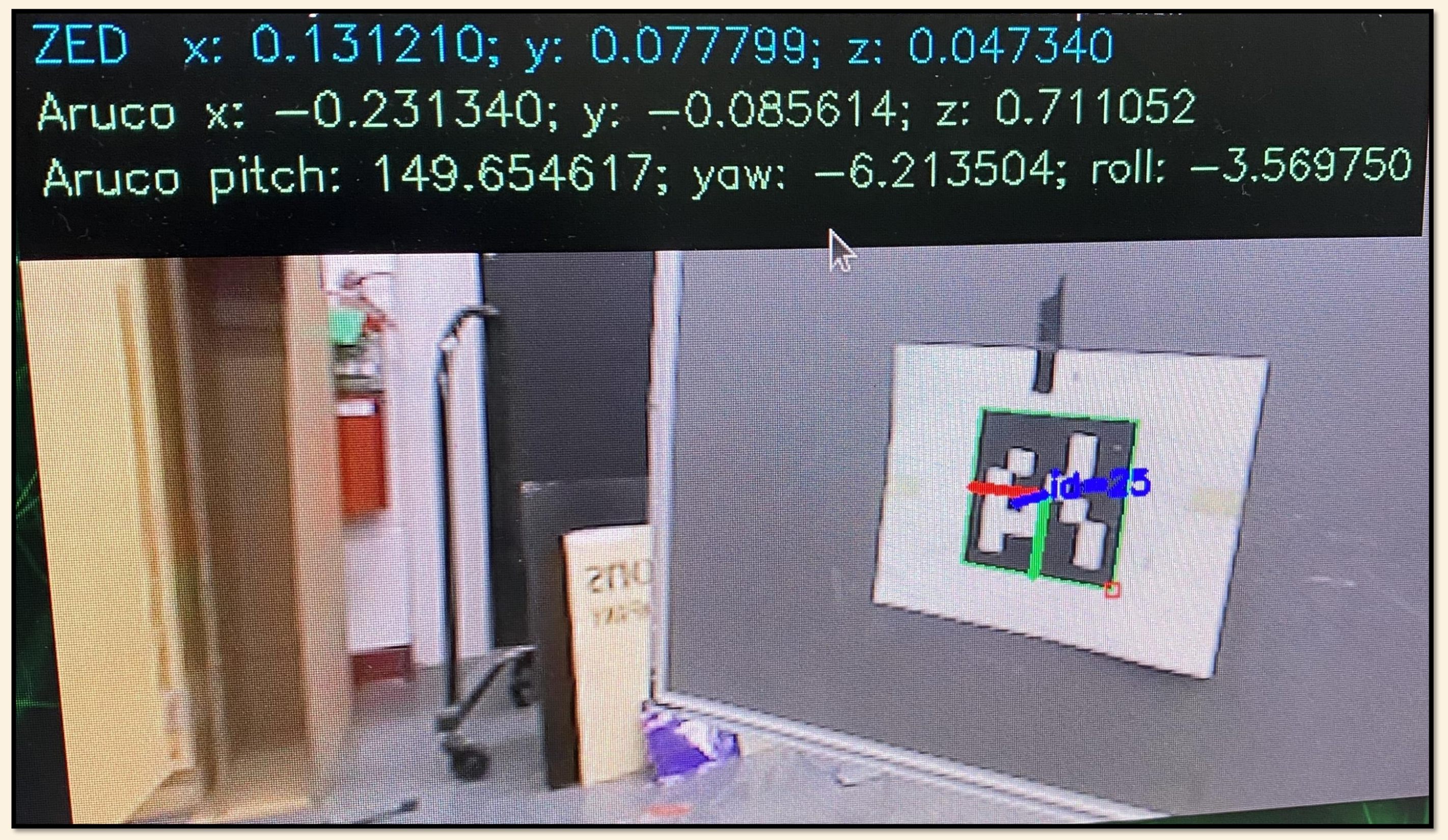
### Driving

The driving autonomy of the robot consists of the following:

- The robot rotates until it locates an ArUco marker using the OpenCV library as well as the ZED camera.
- Once this marker is found, the robot rotates 90° to orient itself with the excavation zone.
- The robot then drives 1 meter forward to get to the excavation zone.
- It then calls upon the excavation automation to extract the material.
- After excavating the material, the robot rotates 180° to orient and drive itself to the dumping zone.
- The robot then dumps the material.

### Excavation

The excavation autonomy is called when the robot arrives at the extraction zone where the regolith is located. The robot uses linear actuators to lower a drum that spins to extract the material. This material is deposited onto a conveyor belt which delivers it to a storage bucket.



```
ZED  x: 0.131210; y: 0.077799; z: 0.047340
Aruco x: -0.231340; y: -0.085614; z: 0.711052
Aruco pitch: 149.654617; yaw: -6.213504; roll: -3.569750
```

*Figure 2: ZED Camera display for ArUco marker detection*

## Challenges/Roadblocks

There were many adversities we faced coming into this project. The largest challenge in the software development process was having to alter the CSCE schedule and tasks to meet the delays in MEEG robot construction. Taking this into account, we had to take a different approach and focus on code we could complete without the robot at first. This pre-requisite code would help with the development of the Autonomy code.

Once we had portions of the physical robot we had instances of mechanical failure including shattering a gearbox and shearing a motor mount which caused further delays.

## Results

After two semesters of work, we successfully created a semi-autonomous rover that can fully drive. Our CSCE sub-team delivered C++ code for ROS2 Nodes that enable Manual Joystick Control and Autonomous Driving.

Unfortunately, due to delays in the Excavation hardware construction, we were unable to demonstrate our excavation autonomy code. However, our code is accessible on the 2022-23 Razorbotz GitHub repository.