

EE 481 – SENIOR DESIGN II

NASA LUNABOTICS COMPETITION TEAM

USER MANUAL

Team Members

Kevin Jordan, Electrical Engineering - Lead
Samuel Sondreal, Electrical Engineering

Project Advisors: Dr. John Nordlie, Computer Science & Dr.
Jeremiah Neubert, Mechanical Engineering

28 April 2023

TABLE OF CONTENTS

WARNINGS, SAFETY, & CAUTIONS.....	1
1. INTRODUCTION.....	1
1.1. INTENDED AUDIENCE	
1.2. PURPOSE OF THIS MANUAL	
1.3. SYSTEM OVERVIEW & COMPONENTS	
1.4. ROVER FUNCTIONS & FEATURES	
2. QUICK START GUIDE.....	3
2.1. STARTUP	
2.2. OPERATING THE ROVER	
3. DETAILED SYSTEM INFORMATION.....	6
3.1. HARDWARE	
3.1.1. Power & Batteries	
3.1.2. Switches	
3.1.3. VEX Motors & TalonSRX Motor Controllers	
3.1.4. Sensors & Circuits	
3.1.5. WiFi / Communications	
3.1.6. Microcontrollers	
3.2. SOFTWARE	
3.2.1. Terminal Basics (Linux/macOS)	
3.2.2. Logging into the Jetson via SSH	
3.2.3. ROS Launch	
3.2.4. ZED Live Video Feed	
3.2.5. Keyboard Controller	
3.3. SYSTEM OPERATION	
3.3.1. System Deployment	
3.3.2. Driving	
3.3.3. Mining	
3.3.4. Hopper	
4. TROUBLESHOOTING & HELP.....	19
4.1. COMMON HARDWARE ISSUES	
4.1.1. Power & Continuity	
4.1.2. Signal Integrity	
4.2. SOFTWARE DOCUMENTATION	
4.2.1. ROS Quick Help	
4.2.2. List of ROS Packages	
4.2.3. List of ROS Nodes	
4.2.4. List of ROS Topics	
4.2.5. SSH Issues	
5. TECHNICAL SPECIFICATIONS.....	22
5.1. MECHANICAL GENERAL STRUCTURES LIST	

- 5.2. HARDWARE LIST
- 5.3. SOFTWARE LIST

WARNINGS, SAFETY, & CAUTIONS

WARNING! This robotic system has many moving and heavy parts and should be operated with caution. In addition to the free motion hazards present, the system includes many possible pinch-points that can cause **SERIOUS INJURY** to the complacent user. Due to these hazards, the system should never be operated alone, should always be operated by a competent person who has read this manual or used the system before after proper instruction.

In addition to motion hazards, the system used high-capacity batteries to drive the very heavy/massive rover. These batteries provide minor shock risk, as well as the possible heat risk in the event of a short circuit somewhere in the system.

Mining is one of the systems main functions, however, the process makes airborne fine particulate material that is unsafe to breathe. If using the mining system to mine or collect material, a respirator mask must be used.

The system has been designed with failsafe's to protect the user as much as possible from accidental or unexpected system malfunction. However, no system is perfect, and these Warnings should be read and observed.

For all software warnings on usage, refer to the LICENSE file in the Team's GitHub at URL:

<https://github.com/KevinJordanENG/lunabotics23/blob/master/LICENSE>

ALL HARDWARE AND SOFTWARE OF THIS ROBOT IS PRESENTED "AS IS" AND IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY ARISING FROM THE USE OF THE ROVER HARDWARE OR SOFTWARE.

1. INTRODUCTION

1.1. INTENDED AUDIENCE

This User Manual is intended to describe how to use the UND Robotics Club Lunar Regolith Mining Rover from the 2023 NASA Lunabotics Competition. The intended audience for this document is the future members of the UND Robotics Club as they attempt to build a bigger and better rover for next year's competition. The UND Robotics Club welcomes members from all departments across campus as well as any year of university. This manual is designed with this in mind, however, assumes an interest and base level familiarity with mechanical systems, electronic hardware, and computer familiarity. While no previous experience in robotics is necessary, it is assumed that the user has enough science, technology, engineering, or mathematics background to be able to understand technical terms, instructions, and figures.

1.2. PURPOSE OF THE DOCUMENT

This document provides instructions on the use of the 2023 Lunabotics Rover. In addition, troubleshooting, technical specifications, and an overview of the system are provided. The Quick Start Guide included in Section 2 provides a brief series of setup steps for those familiar with robotics, and only interested in the specifics use instructions for this rover.

1.3. SYSTEM OVERVIEW & COMPONENTS

The rover system and its constituent subsystems and components can be seen below in Fig. 1-4.

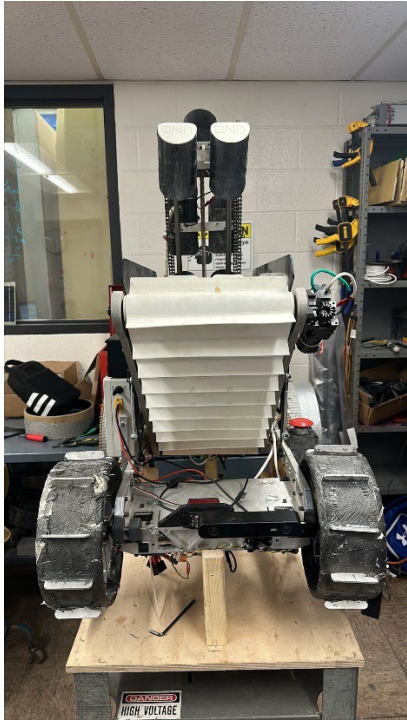


Figure 1. Rover Front View

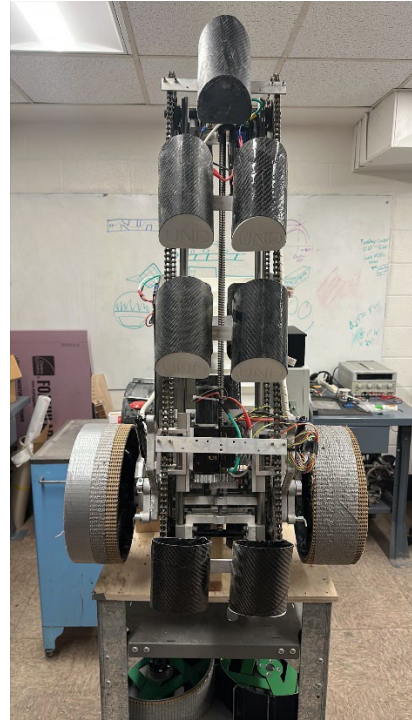


Figure 2. Rover Rear View



Figure 3. Rover Left Side View

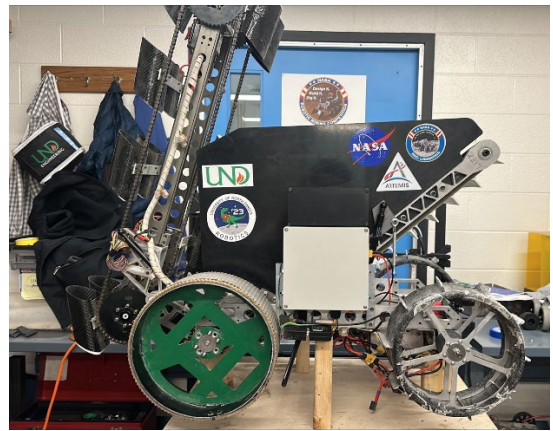


Figure 4. Rover Right Side View

To further clarify the main important electrical components, a full system block diagram is presented below in Fig. 5.

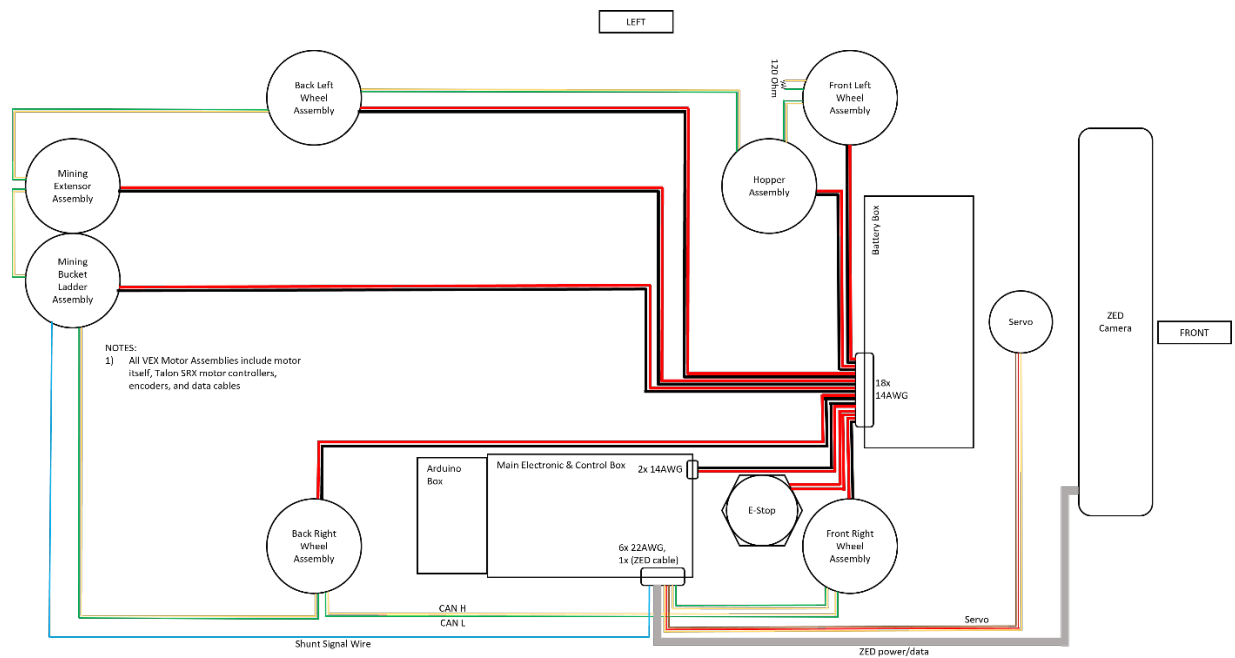


Figure 5. Rover System Block Diagram

1.4. ROVER FUNCTIONS & FEATURES

The rover includes many features and functions needed in order to be within the 2023 competition minimum performance specifications. High level lists of these are detailed as follows:

Features:

- Battery operated requiring no wall power
- Teleoperated via Keyboard Control & Wi-Fi
- Live Video Stream
- Emergency Stop Button for Full Motion System Shutdown

Functions:

- System Deployment via Servo
- Terrain Traversal / Driving
- Mining – Collection of Material
- Mining – Extension / Retraction of Mining Mechanism
- Hopper Conveyor Belt Deposition of Mined Material

2. QUICK START GUIDE

The Quick Start Guide is intended to be used when the rover system is already in a ready-to-operate state including all electrical connections made, batteries charged, etc. The brief instructions here do not replace the full system operating instructions below in Section 3, but are intended to allow a faster startup for those already familiar with similar technologies such as remote-control robots, Linux, ROS, etc. Before using the Quick Start Guide ensure that the Warnings, Safety, & Cautions at the beginning of this document have been read, understood, and no mentioned operating hazards are present.

2.1. STARTUP

Assuming all connections made, and no motion obstructions, the switch on the electrical box (E-Box) to the Jetson can be switched to the on position as shown in Fig. 6.



Figure 6. Jetson / System Switch

If all electrical systems have properly powered on, the TalonSRX Motor Controllers should have their LEDs alternating between red and off as shown in Fig 7. The blink codes for these controllers are shown later in this document.

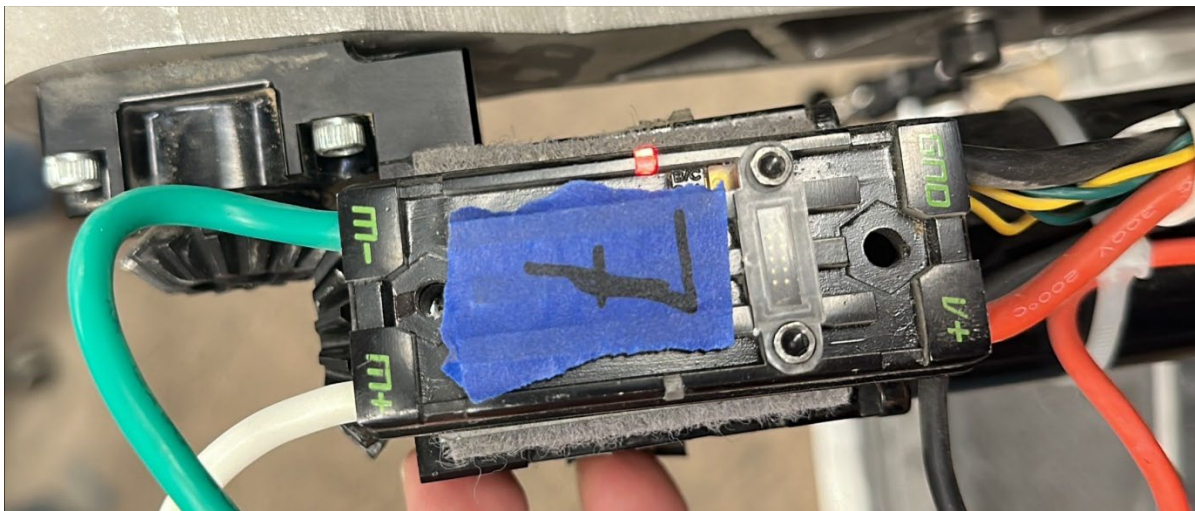


Figure 7. TalonSRX Powered / Idle State

Once this has been verified, the control software can be launched. Open a terminal, connect to the Jetson via SSH, and run the command:

```
`roslaunch launch teleop.launch`
```

This will launch the entire rover software suite and allow control of the rover detailed next.

2.2. OPERATING THE ROVER

The launching of the software suite in the previous step turns the terminal into a persistent menu that allows the control of all rover functions as shown below in Fig. 8.

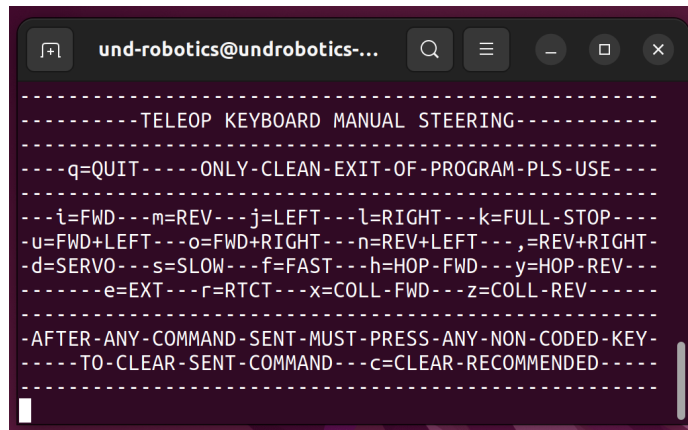


Figure 8. Keyboard Control Persistent Menu

The launch process will also open a window that is the live video feed from the ZED camera. This allows the user to see where the rover is traversing while in motion, an example view into the UND Robotics Lab included in Fig. 9 below. Note that as the ZED is a stereo camera there will be two views presented with a slight offset from the other.

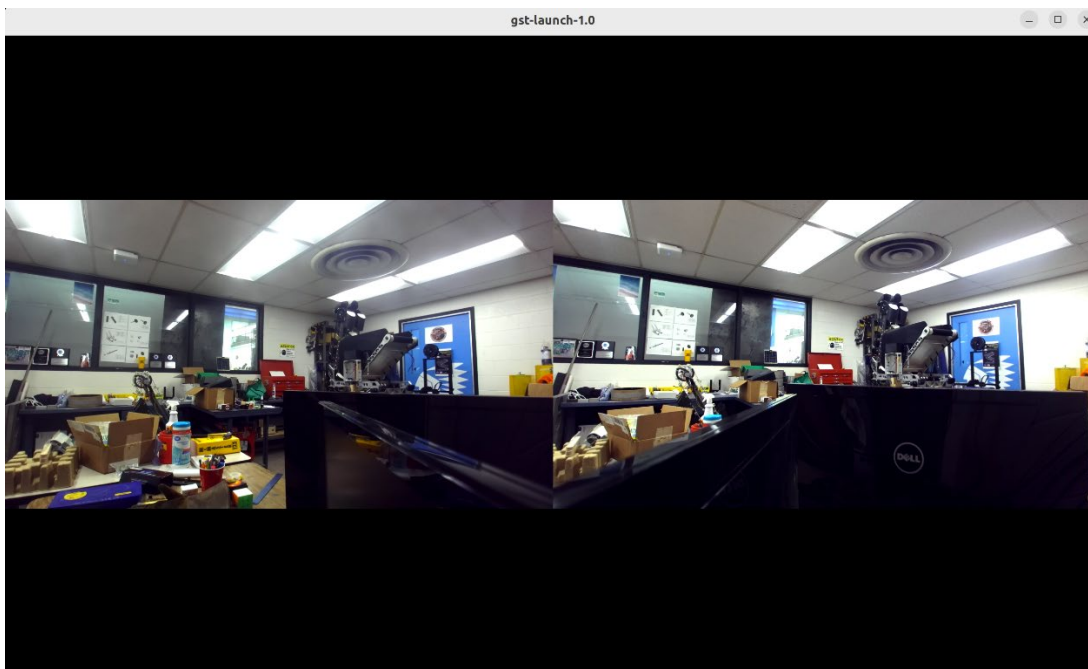


Figure 9. ZED Camera Video Feed Window

From there, operation is as simple as reading the menu and pressing the associated keys! An important note in how the keyboard controller works is that all commands are toggle style aside from `CLEAR`, `SLOW`, `FAST`, and `QUIT`. This means that after a command has been sent it must be cleared with the master `CLEAR` command or alternatively by pressing any other non-coded key. These non-coded keys act as a failsafe shutoff to accidental keypresses. `SLOW` and `FAST` commands do not need clearing as they only store the scalar value used in the calculation of the velocity command signal to the wheels. Whatever value that was last given will remain until changed. For more detailed keyboard controller instructions see Section 2.2.5 below.

3. DETAILED SYSTEM INFORMATION

3.1. HARDWARE

This section provides use instructions for the main hardware systems present in the rover. While main subsystems are detailed, discrete component level specifics are not included as they are not able to be used independently of the respective rover subsystems.

3.1.1. Power & Batteries

The batteries are placed inside the custom battery box mounted underneath the rear panel and is then screwed into at four points on the top of the panel, as seen in Fig. 10. The box was designed to hold a smaller 3300 mAh lithium polymer (LiPo) battery and two larger 8000 mAh LiPo batteries, which will all fit with the respective wires connected and laid flush onto the batteries themselves. Fig. 11 shows the XT60 connectors used within the box and along the frame to connect the motor assemblies to power. The larger batteries both connect to a 2-to-1 adapter which leads to the power and ground buses inside the box, while the small battery connects to the XT60 passthrough connector on the long side of the box.



Figure 10. Battery Box Mounting / Disconnecting from Chassis



Figure 11. XT60 Connectors for Power Components

3.1.2. Switches

The switch on the side of the E-Box, as shown in Fig. 12, provides power to the components inside, including the NVIDIA Jetson and the Arduino MEGA2560. Fig. 13 shows the Commercial Off-The-Shelf (COTS) emergency stop switch which allows power to be cut off to all motor assemblies. The Arduino itself also has a red reset switch as seen

in Fig. 14, which will mainly be used for troubleshooting and testing, allowing the user to reboot the script loaded onto it.



Figure 12. E-Box Jetson / Main Power Switch



Figure 13. E-Stop Switch



Figure 14. Arduino Reset Button

3.1.3. VEX Motors & TalonSRX Motor Controllers

The rover is driven by VEX DC BAG motors controlled by VEX TalonSRX motor controllers. These motor controllers use blink codes with their built-in LEDs to provide the user with the current state of the controllers. For example, the controllers should blink alternate blinking red when powered and idle, or in other words a CAN bus is not detected. Fig. 15 shows the TalonSRX's in their connected state when motor assemblies and E-Box system are powered on, meaning the CAN network is detected but the rover is not receiving any commands. Fig. 16 provides a key for understanding the possible blink codes on the TalonSRX's.

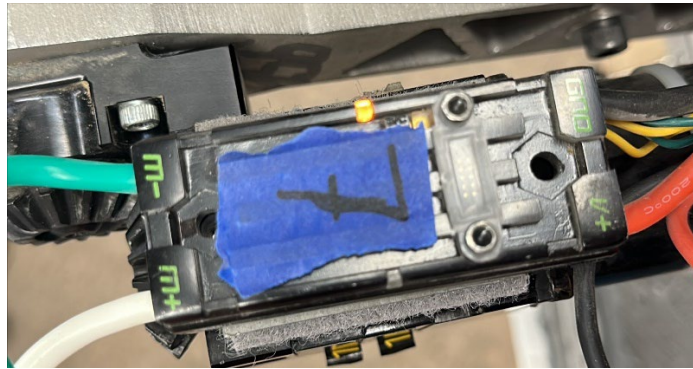


Figure 15. TalonSRX in ready state / CAN Detected

Blink Codes During Normal Operation		
LEDs	Colors	Talon SRX State
Both	Blinking Green	Forward throttle is applied. Blink rate is proportional to Duty Cycle
Both	Blinking Red	Reverse throttle is applied. Blink rate is proportional to Duty Cycle
None	None	No Power is being applied to Talon SRX
LEDs Alternate ¹	Off/ Orange	CAN bus detected, robot disabled
LEDs Alternate ¹	Off/ Red	CAN bus/PWM is not detected
LEDs Alternate ¹	Switching between Red/Orange and Orange/Red	Damaged Hardware
LEDs Strobe "towards" (M+) ²	Off/ Red	Forward Limit Switch or Forward Soft Limit
LEDs Strobe "towards" (M-) ²	Off/ Red	Reverse Limit Switch or Reverse Soft Limit
LED1 Only "closest" to M+/V+	Green/Orange	In Boot-loader
Both	Solid Orange	Neutral throttle is applied. Throttle is zero or is within dead band.

Figure 16. Key for VEX Talon SRX Blink Codes

Fig. 17 shows the Canable USB-to-CAN adapter used to connect the CAN network to the NVIDIA Jetson. Yellow and green CAN wires represent the high and low signals, respectively, that create the daisy-chain between each motor assembly.

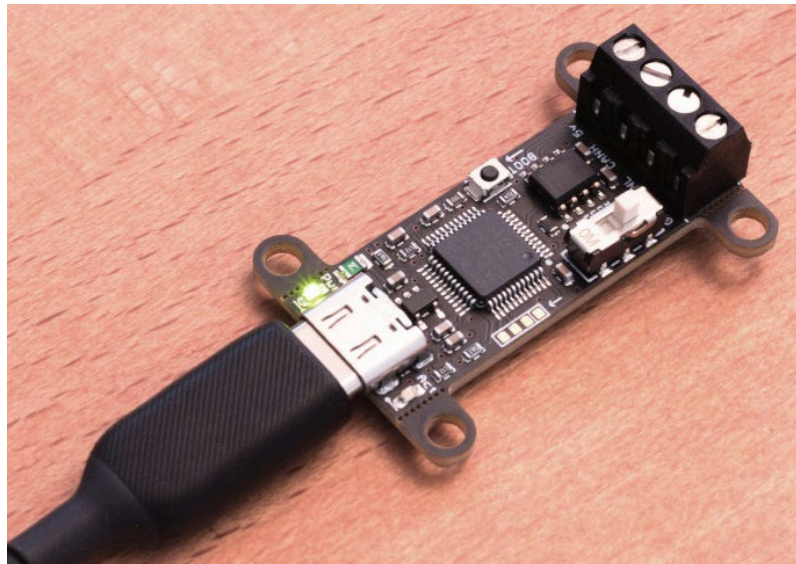


Figure 17. CANable USB-to-CAN Adapter

3.1.4. Sensors & Circuits

A shunt resistor provides a voltage reading to an amplifier circuit within the E-Box. A small voltage drop is measured and amplified so the Arduino can read the voltage, and derive the instantaneous current being drawn by the mining system. Fig. 18 shows a shunt resistor used to measure a small voltage drop across the motor for the mining mechanism.



Figure 18. Shunt Resistor

Step-down voltage regulator modules were used to decrease the voltage from the 14.8 V LiPos into the proper operating voltages for both the Jetson and Arduino. Fig. 19 shows a voltage regulator module. Each has a potentiometer that is used to adjust the output voltage of the module to the user's needs. The modules should be set at ~ 5 V and ~ 7.4 V to provide power for the Jetson and Arduino, respectively.

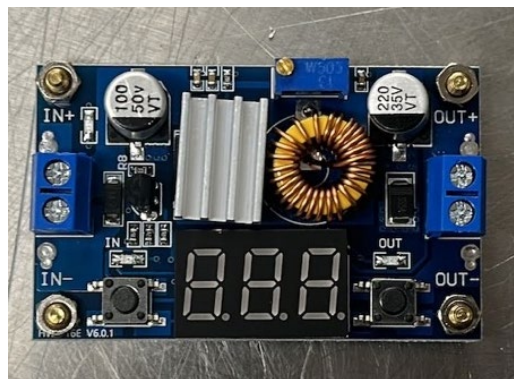


Figure 19. Step-Down Voltage Regulator Modules

3.1.5. Wi-Fi / Communications

To communicate from the base station laptop, a USB Wi-Fi antenna is plugged into one of the USB passthroughs on the bottom of the E-Box. This allows the rover to connect to the private network setup for the robotics team, and allows a private SSH connection between the rover and the laptop.

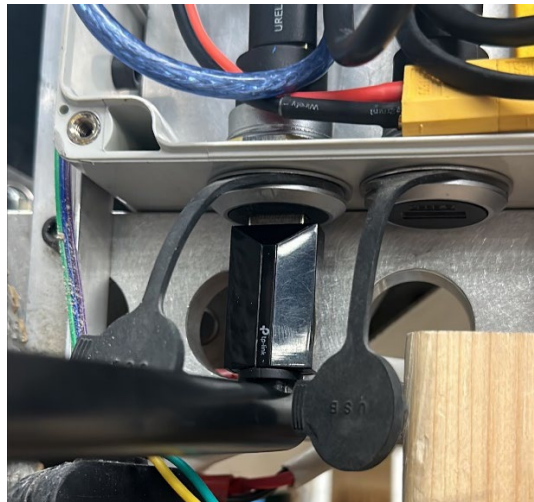


Figure 20. USB Wi-Fi Antenna

3.1.6. Microcontrollers

The central control systems for the rover consisted of the NVIDIA Jetson Nano and the Arduino Mega2560. Sensor circuitry and signal wires are wired into the E-Box via a DB-9 block connector. Fig. 21 shows the inside of the E-box, including the Jetson, Arduino, voltage regulators, and signal processing circuitry.

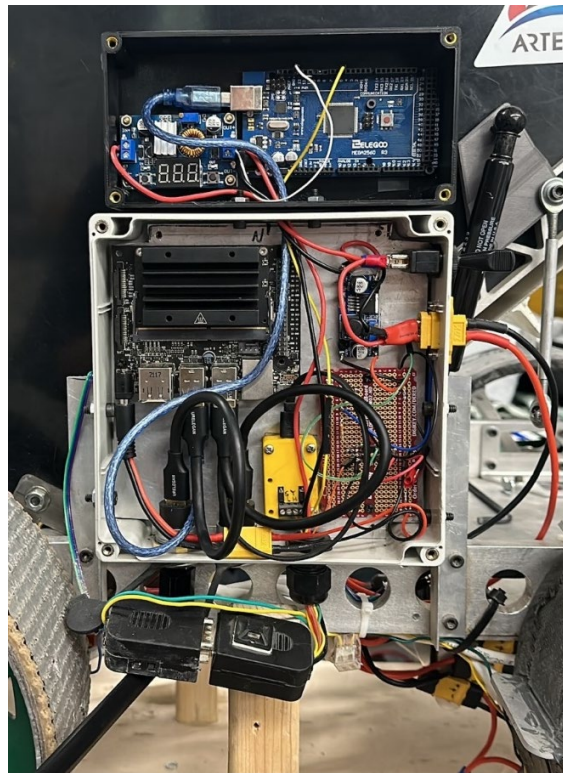


Figure 21. Open View of Inside of Main E-Box and Arduino Box with all Connections Made

The Jetson has 4 USB 3.0 ports which allow the user to connect peripheral devices. Fig. 22 shows the CANable adapter, the USB A-to-B cable for Arduino, and two USB extender cables that allow the ZED camera and Wi-Fi antenna to be plugged into USB passthrough ports on the side of the box.

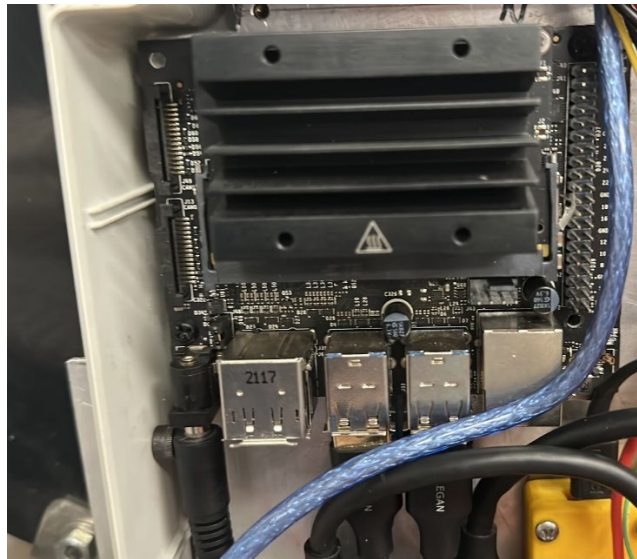


Figure 22. Jetson Closeup and Peripheral Connections

In Fig. 23, a closeup of the Arduino shows the signal wires for the servo connected, and the voltage regulator board powering the servo via 3300 mAh Li-Po battery.

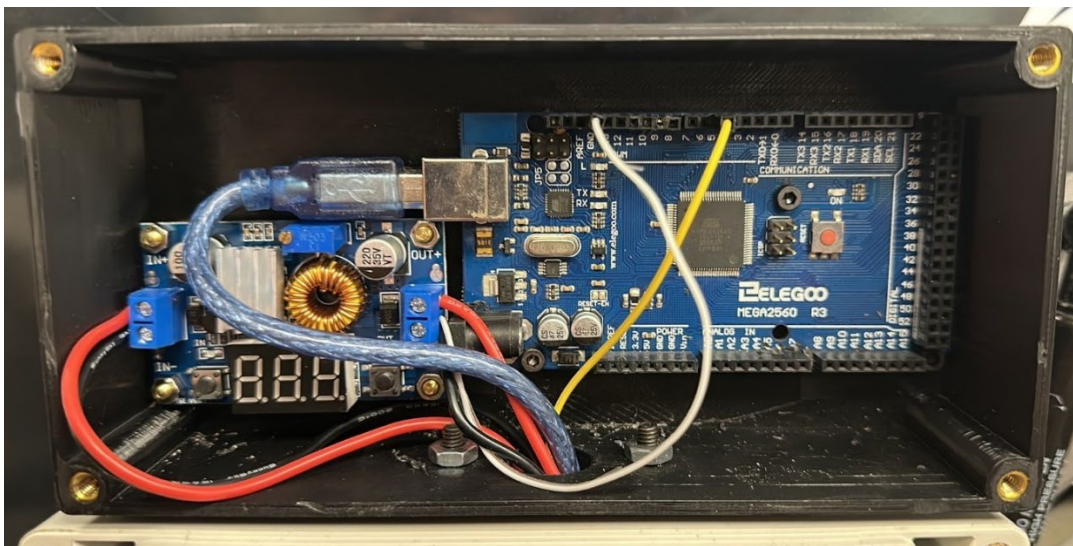


Figure 23. Arduino Box Closeup

3.2. SOFTWARE

The software section describes the main systems, frameworks, and tools that are used to make this rover go. This includes a quick rundown of the tools and their general use as well as their application to this system. The innerworkings of the source code and software architecture are beyond the scope of this guide, however a comprehensive list of system variables and helpful tips is included in Section 4, Troubleshooting & Help.

3.2.1. Terminal Basics (Linux/macOS)

The team's operating system of choice is Linux for various reasons including that it was the most sensible operating system on the rover's main microcontroller the Jetson. This meant seamless integration of the base station command

laptop and the rover. Because macOS is also built on a UNIX framework, the same instructions for use of the base laptop and how to control the rover should be identical if using a Mac as the control laptop.

The rover's software is accessed via command line use of the Terminal. While slightly different between Mac and Linux, both should look very similar or identical to Fig. 24 below.

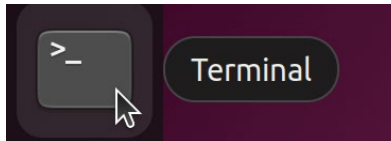


Figure 24. Terminal Icon

The first step is to open a new Terminal which will bring up a simple window as shown below in Fig. 25.

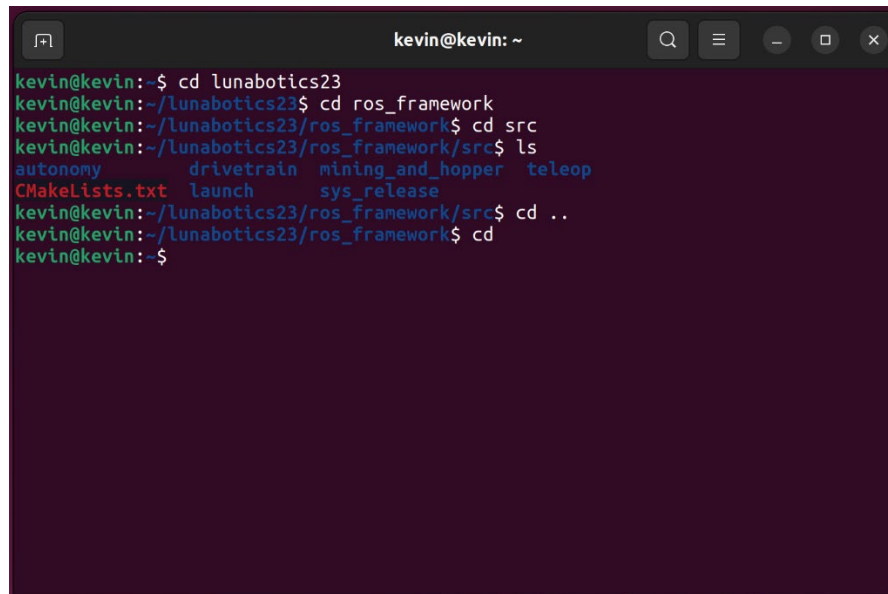


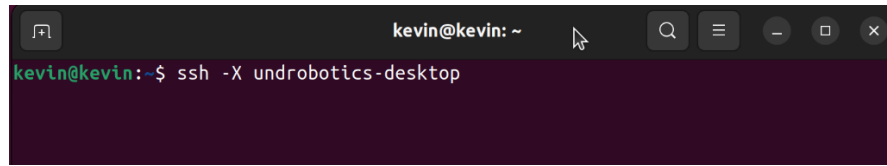
Figure 25. Opened Terminal and Filesystem

Once the Terminal has been opened it is possible to access the filesystem of the control laptop as well as send system commands. Some basic Linux/macOS filesystem commands are also shown above in Fig. 25. `cd` is the command for change directory and can include the folder name only if in the parent directory or the full path to the directory. The first 3 commands shown in Fig. 25 show the successive navigation to the `src` directory. Once inside the target directory, the `ls` command can be sent to display all of the folder contents. In the example above, the `src` directory contains 6 other child folders and a .txt file. If the `cd` command is sent with two periods as `cd ..` you will be returned to the parent folder as seen in the 5th command shown in Fig. 25. The `cd` command alone will return you to the home directory.

3.2.2. Logging into the Jetson via SSH

To connect wirelessly to the Jetson, the secure shell method is used. This allows a user to log into a remote machine using IP and user name of the target computer. To log in via ssh, the command consists of `ssh -X user@123.12.123.123` where the user is the username of the desired remote system and the number string following the '@' are the target system's IP address. For experienced users, an SSH configuration file can be created to simplify this process and name the target device to avoid remembering/typing the IP address each time, an example shown below in Fig. 26. This is discussed further in the SSH Issues section, 4.2.5. For this rover system since we will need to open a

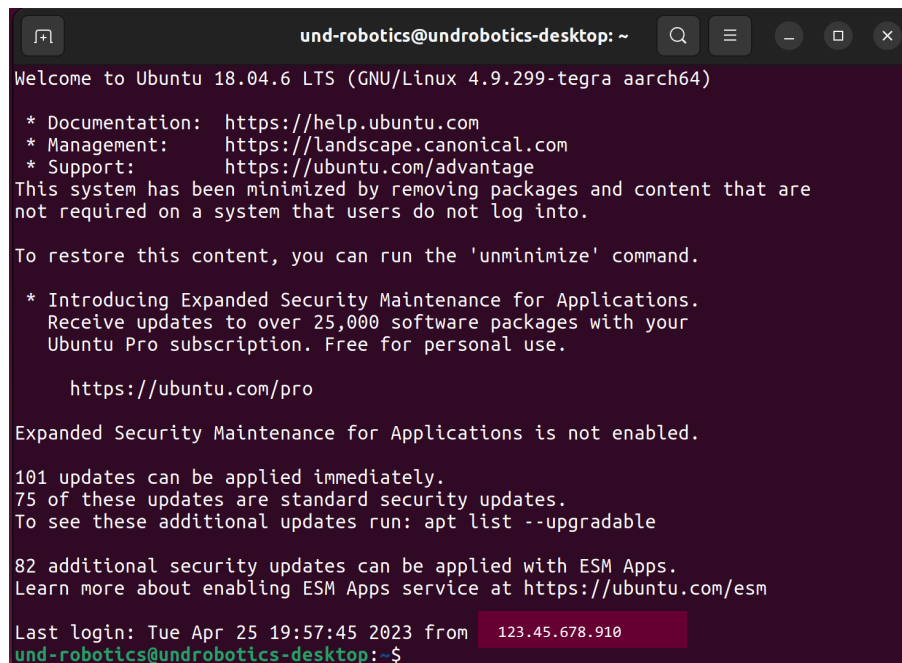
GUI from the remote system to view camera output on the base laptop, it is important to include the `-X` flag after the ssh command. This enables X11 display forwarding allowing the GUI window that would have opened on the Jetson to open instead on the base station control laptop.

A terminal window titled 'kevin@kevin: ~' with search, menu, and window control icons. The command 'ssh -X undrobotics-desktop' is entered at the prompt.

```
kevin@kevin:~$ ssh -X undrobotics-desktop
```

Figure 26. SSH Command with X Display Forwarding

Once the command to connect to the Jetson has been sent, it will prompt the user for a password. For system security the password for the rovers true microcontroller is not included. Upon reasonable request to the authors with valid use case and plan (such as for the 2024 NASA Lunabotics UND Robotics Club Team), the sharing of this password can be arranged. Upon successful login, an output similar to what is shown in Fig. 27 will be displayed in the Terminal. This displays information useful in debugging or further software development such as operating system architecture, version, and others. The last thing to note upon successful login will be the change of username displayed on the command line. Prior to logging in, the local machine was shown in Fig. 25 & 26 as ``kevin@kevin:~$``, but can be seen below in Fig. 27 as changed to ``und-robotics@undrobotics-desktop:~$`` after login. Note for system security the true IP address of the login has been covered/replaced in Fig. 27.

A terminal window titled 'und-robotics@undrobotics-desktop: ~' with search, menu, and window control icons. It displays the Ubuntu 18.04.6 LTS login screen, including system information, update notifications, and the last login details. The IP address is redacted with a pink box.

```
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.9.299-tegra aarch64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

* Introducing Expanded Security Maintenance for Applications.
  Receive updates to over 25,000 software packages with your
  Ubuntu Pro subscription. Free for personal use.

  https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

101 updates can be applied immediately.
75 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

82 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Tue Apr 25 19:57:45 2023 from 123.45.678.910
und-robotics@undrobotics-desktop:~$
```

Figure 27. Successful Login Display with New User at Command Line

3.2.3. ROS Launch

Once the Jetson has been logged into successfully, and assuming all hardware setup has been done such as connecting batteries etc., the ROS framework and rover software suite can be launched. All packages and needed libraries have been properly installed and configured in the system allowing a simple 1-line command startup of the full software suite. The command to launch all rover software is shown below in Fig. 28. The first argument ``roslaunch``, is a standard ROS command and is used to call up a `.launch` file that defines all nodes and processes to be started. ``launch``, the second argument, is the ROS package that contains the launch file, ``teleop.launch``, the third argument.

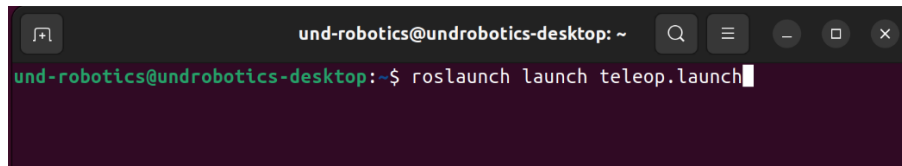
A terminal window titled 'und-robotics@undrobotics-desktop: ~' with search, menu, and window control icons. The command 'roslaunch launch teleop.launch' is entered at the prompt.

Figure 28. Launch Command to Startup ROS Software Suite

Once this launch command has been sent the ROS software framework startup output should be displayed in the Terminal as shown below in Fig. 29. While mostly boilerplate, this output will show all launched nodes as well as any error messages if issues were encountered launching specific nodes. Each ROS node is a separate system process with a unique `pid` or process ID.

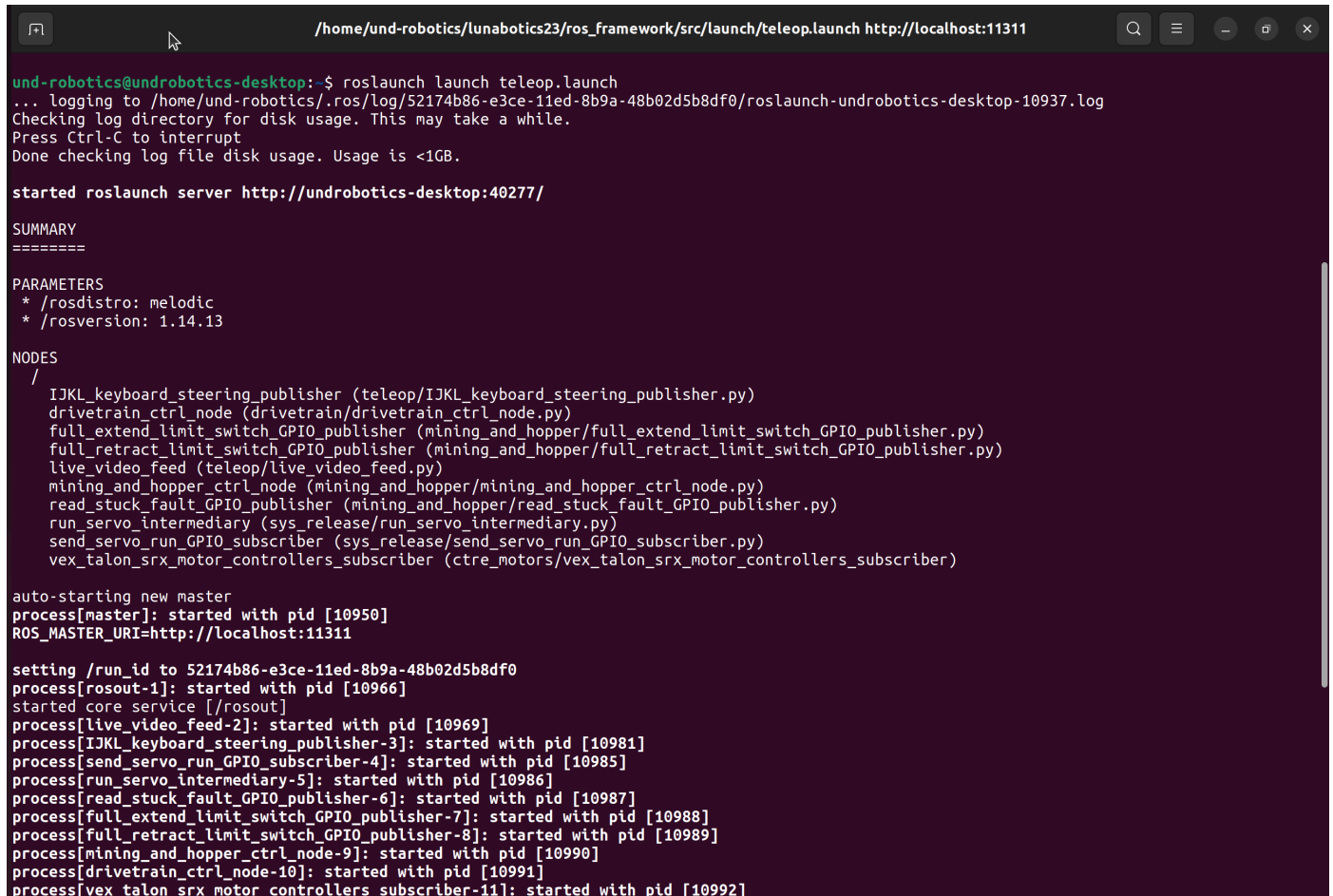
A terminal window titled '/home/und-robotics/lunabotics23/ros_framework/src/launch/teleop.launch http://localhost:11311' with search, menu, and window control icons. The output of the 'roslaunch' command is displayed, including log file location, disk usage check, master server URL, summary, parameters, and a detailed list of 11 nodes with their respective PIDs. The nodes listed are: IJKL_keyboard_steering_publisher, drivetrain_ctrl_node, full_extend_limit_switch_GPIO_publisher, full_retract_limit_switch_GPIO_publisher, live_video_feed, mining_and_hopper_ctrl_node, read_stuck_fault_GPIO_publisher, run_servo_intermediary, send_servo_run_GPIO_subscriber, and vex_talon_srx_motor_controllers_subscriber.

Figure 29. Successful ROS Launch Startup Messages & Node List

3.2.4. ZED Live Video Feed

The live video feed from the ZED camera will be started when the rover software application is launched. Given the ZED 2 is a stereo camera, the window that opens to display the video will have two video frames of the same area with slight offset between the two as seen below in Fig. 30.

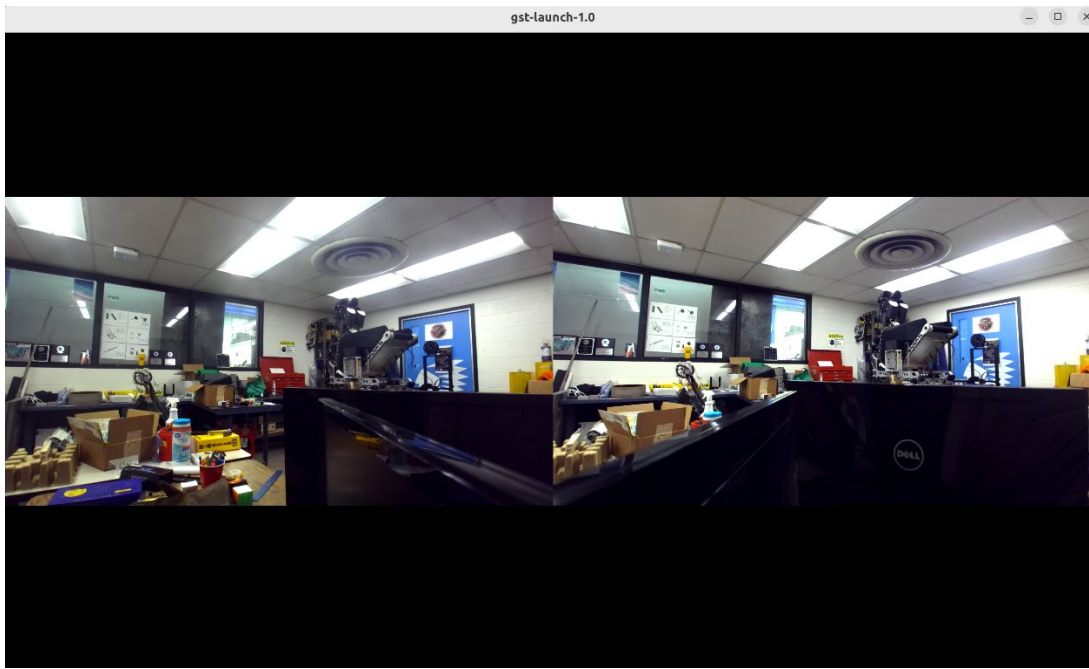


Figure 30. ZED Camera Video Feed Window

3.2.5. Keyboard Controller

To drive and control the rover, after the launch script is run which opens all necessary ROS nodes, the menu prompt shown in Fig. 31 will allow the user to provide key input for tele-operation. This menu is persistent in the Terminal, only exiting when the full system is closed either via the ‘q’=QUIT command or by closing the Terminal via the window ‘X’ and mouse. Upon close of the Terminal or quitting of the menu, all rover software processes will be stopped and exited. A comprehensive description of all available commands is provided in the list below.

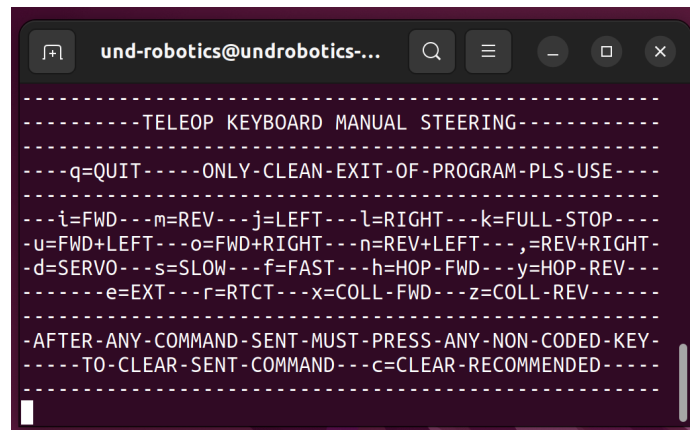


Figure 31. Keyboard Control Persistent Menu

Command List & Details:

‘q’=QUIT: Exactly as it would be expected to work, stops all processes and exits program. Due to threading this is the only thread safe exit. In case of forgetting, CTRL+C for a SIGINT and manual process stop will also work, but will require the pressing of any 1 additional key to return to the terminal command prompt.

‘i’=FWD: Commands the drivetrain wheels to move the rover forward. This command also takes the scalar multiplier stored in the velocity variable controlled by the SLOW & FAST commands and scales the forward speed based on the last commanded/stored velocity.

‘m’=REV: Commands the drivetrain wheels to move the rover in reverse. This command also takes the scalar multiplier stored in the velocity variable controlled by the SLOW & FAST commands and scales the forward speed based on the last commanded/stored velocity.

‘j’=LEFT: Turns the rover in place rotating to the left. Single speed available.

‘l’=RIGHT: Turns the rover in place rotating to the right. Single speed available.

‘k’=FULL-STOP: Stops all drivetrain wheels and commands the rover to immediately stop. Does not stop/affect commands to the extensor, collector, or hopper.

‘u’=FWD+LEFT: Drives rover forward while turning left. Calculates arc / turning profile from fixed turn-in-place speed and stored forward/reverse velocity variable.

‘o’=FWD+RIGHT: Drives rover forward while turning right. Calculates arc / turning profile from fixed turn-in-place speed and stored forward/reverse velocity variable.

‘n’=REV+LEFT: Drives rover reverse while turning left. Calculates arc / turning profile from fixed turn-in-place speed and stored forward/reverse velocity variable.

‘,’=REV+RIGHT: Drives rover reverse while turning right. Calculates arc / turning profile from fixed turn-in-place speed and stored forward/reverse velocity variable.

‘d’=SERVO: Deploys system by activating the Servo to release pins holding system in stowed position. Only will function once unless Arduino reset button is pressed or full rover system is reset/powered off and back on.

‘s’=SLOW: Decrement stored forward/reverse velocity variable within limits of min=1 to max=3. If already at 1 (SLOW), command will maintain value 1. Value stored in velocity variable is persistent and will remain regardless of CLEAR commands sent. Only way to change value is by using SLOW or FAST commands.

‘f’=FAST: Increment stored forward/reverse velocity variable within limits of min=1 to max=3. If already at 3 (FAST), command will maintain value 3. Value stored in velocity variable is persistent and will remain regardless of CLEAR commands sent. Only way to change value is by using SLOW or FAST commands.

‘h’=HOP-FWD: Turns the hopper conveyor belt forwards/material up to be deposited in sieve. Single speed available.

‘y’=HOP-REV: Turns the hopper conveyor belt reverse, helpful in aligning fins on conveyor for material collection. Single speed available.

‘e’=EXT: Extend mining system downwards for mining and collection of material. Traversal/extension rate downward is slower than retraction due to increased resistance of the collection/mining of material.

‘r’=RTCT: Retract mining system upwards after the mining of material has been completed. Traversal/retraction rate upwards if faster than extension to minimize time waiting for full retraction before being able to drive back to deposition site.

‘x’=COLL-FWD: Drives material collector bucket ladder forward actively mining material the buckets contact. Single speed available.

‘z’=COLL-REV: Drives material collector bucket ladder reverse to help relieve from stuck state if high resistance encountered while mining.. Single speed available.

‘c’=CLEAR: Clears all previous commands. All drivetrain motors set to OFF, all mining and hopper motors set to OFF. Performs the same function as all other non-coded keys, but easier to remember as ‘c’ key is commonly CLEAR command.

all_other_non-coded_keys: Clears all previous commands. All drivetrain motors set to OFF, all mining and hopper motors set to OFF. Performs the same function as ‘c’=CLEAR, and acts as a failsafe from accidental key presses.

3.3. SYSTEM OPERATION

This section provides reference to the main functions of the rover. While the Keyboard Controller section above well explains the commands, the systems themselves and considerations in their use are described here.

3.3.1. System Deployment



Figure 32. Servo and Pins in Starting Position

The system begins in a stowed configuration to start a competition run due to NASA provided constraints. Before any other system operation can begin, the system must be deployed into operational position. This is accomplished by a servo that pulls pins that release gas shocks that push the system into a deployed position. To properly use the servo deploy system deployment function, the system MUST be in stowed configuration, servo reset to starting position, and pins in place holding the system in this configuration.

If the only rover functions needed are to be driven and to operate the mining system, this sections instructions can be ignored.

3.3.2. Driving

The rover can be driven though the controls described above in section 3.2.5. However, some considerations about the terrain and how the drivetrain has been designed are discussed here. The most notable is the direct mounting of motor assemblies to the chassis/frame. While low weight and simpler to troubleshoot, the lack of suspension system adds the potential for some system performance faulting. If traversing uneven terrain, such as that in the competition pit, it is important to plan and drive a path that will not leave 1 wheel without sufficient contact. If this situation arises, it is

possible the other wheels will not have sufficient force to return the floating wheel to ground contact. The non-touching wheel will spin freely until contact to the terrain is returned.

3.3.3. Mining

The mining system control is based on two system motors. The collector motor drives the bucket ladder, which digs into the lunar surface and scoops regolith simulant. The second motor controls a power screw which drives the entire bucket ladder chain module up or down, allowing the buckets to dig deeper into the mining zone. It's important to maintain an adequate balance between the extension speed of the entire bucket ladder chain and the rotation speed of the buckets to ensure the buckets gather enough lunar simulant without getting stuck or becoming too heavy. The current motor control script discussed in the Software section is manual control of both mining system motors, which were calibrated to optimize competition performance.



Figure 34. Mining System in Action

3.3.4. Hopper

The hopper is simple to control as it is a one motor system. However, the alignment of the fins that help transport material up the conveyor to be deposited is an important practical use concern. The motor at the top of the hopper drives the belt which carries lunar simulant up towards the sieve in the deposition zone. During runs, lunar simulant and rocks tend to build up at the bottom of the hopper and on the sides of the fins, which can potentially fall into the inside of the belt. Weather stripping can be placed on the sides of the fins to help seal the empty space between them and the belt. Intermittent running of the belt to even the amount of simulant at the bottom of the hopper also allows the belt to move smoothly.



Figure 35. Overhead View of Hopper

4. TROUBLESHOOTING & HELP

4.1. COMMON HARDWARE ISSUES

4.1.1. Power & Continuity

The system has many components and therefore connections. The connections between components can often cause issues due to loose connections, poor soldering, or interference from other objects. In this system, XT60 connectors were used for all motor assemblies to fulfill the modular design and ensure power and ground could not be incorrectly connected. If power issues occur, some of these connectors could be disconnected or need to be resoldered to complete the connections. Alternatively, the batteries should be consistently monitored to keep them within a proper operating voltage. All three batteries used are 4 cell 14.8 V Li-Po batteries which should be used within the range of 12.8-16.8 V, or 3.2-4.2 V per cell, and charged at the smart Li-Po charger station. Each motor assembly has a 30 A circuit breaker to be connected between the battery box and a motor assembly, so if some Talons' are not powered, ensure the power wires coming from the box are each connected to a circuit breaker.

4.1.2. Signal Integrity

The electromagnetic interference that can be caused by high voltage or DC power connections was a consideration in the design for this system. To ensure no interference occurs, the high-current power wires for the motor assemblies should remain along the frame, with signal wires leading to the E-Box positioned perpendicular to those wires. Also note that many of the signal wires are smaller gauge, such as 22, and may have jumper wires that come loose or disconnect when troubleshooting.

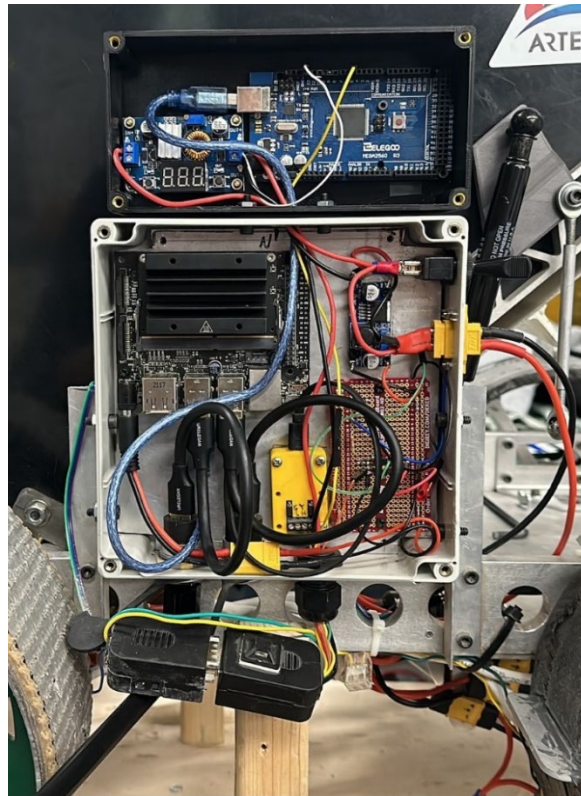


Figure 36. Photo of E-Box Showing Power & Signal Wires being Far Apart

4.2. SOFTWARE DOCUMENTATION

4.2.1. ROS Quick Help

The ROS development environment can be vast and overwhelming. The best place to look for documentation is the source itself, available at the URL included below.

<http://wiki.ros.org/>

Thankfully, most of it can be navigated with relatively few commands and strategies. There are a few vital ROS commands that handle 90% of debugging needs.

First, to run individual nodes and trace processes, the ROS core server must have also been started by running the command ``roscore``. This will print the same ROS startup messages as seen in Fig. 29 minus the list of active nodes as there are none yet running.

It is useful to be able to run only one node, or a select few nodes at a time. This is possible by using the command:

```
`roslaunch [pkg_name] [node_name]`
```

Each node requires its own separate Terminal when running nodes individually in this way. It is worth noting that for nodes developed in C++, as it is a compiled language, the node name will be the executable with the same name as the node without any file extension, for example ``vex_talon_srx_motor_controllers_subscriber`` from the list of nodes below. On the other hand, for nodes developed in Python the executable is the script itself and therefore must include the file extension such as ``drivetrain_ctrl1_node.py`` from the list of nodes below.

Frequently, issues come with proper ROS “message” passing. ROS “messages” (called “topics”) are complex/container datatypes that simplify communication between various hardware nodes and across multiple programming languages.

This way a C++ node needed for one hardware control and communicate directly with a Python node handling another process. The message from any running node can be printed to the terminal window with the help of the command:

```
`rostopic echo [topic_name]`
```

While these are the most useful ways to use the `roslaunch` and the `rostopic` commands, there are many others detailed fully in the ROS tutorials linked at the beginning of this section.

4.2.2. List of ROS Packages

- ctre_motors
- drivetrain
- launch
- mining_and_hopper
- sys_release
- teleop

4.2.3. List of ROS Nodes

- vex_talon_srx_motor_controllers_subscriber
- drivetrain_ctrl_node
- mining_and_hopper_ctrl_node
- read_stuck_fault_GPIO_publisher
- run_servo_intermediary
- send_servo_run_GPIO_subscriber
- IJKL_keyboard_steering_publisher
- live_video_feed

4.2.4. List of ROS Topics

- x_cmd_velocity
- theta_cmd_velocity
- extensor_cmd_signal
- collector_cmd_signal
- hopper_cmd_signal
- x_cmd_code
- theta_cmd_code
- run_extensor_state
- run_collector_state
- run_hopper_state
- stuck_fault_state
- run_servo_key
- run_servo_command

4.2.5. SSH Issues

The SSH tunnel can give rise to some difficulties. The most effective way to solve these issues is to setup a `config` file that saves known remote computers IP and username information. Additionally, the `-X` forwarding parameter can be permanently enabled such as seen in the first line in Fig. 37 below. To properly setup an SSH config file, a directory named `.ssh` needs to be created in the `/home/` directory of the base station control laptop if none exists already. Sometimes this directory does already exist but is hidden which can be found by selecting “show hidden folders” in the file explorer. Once this `.ssh` directory has been setup, a file with the name `config` with no extension must be created. In this file, the same syntax and structure as shown below in Fig. 37 can be used where `Host` will be the given machine name and shorthand SSH handle, `HostName` will be the IP address (replaced below for system

security), and `User` will be the username of the remote system. Seen below in Fig. 37 is an extra option called `ProxyJump`. This allows SSH tunneling through multiple machines if needed, used here due to firewall issues using UND campus WiFi and logging in remotely from Montreal, Canada. With this file setup, the direct shorthand syntax as seen in Fig. 26 can be used, removing the need to always type and remember the IP address.

```

1 ForwardX11 yes
2
3 Host csccluster
4     HostName 123.45.678.910
5     User kevin.jordan
6
7 Host undrobotics-desktop
8     HostName 123.45.678.910
9     User und-robotics
10    ProxyJump csccluster
11
12 Host undrobotics-desktop
13     HostName 123.45.678.910
14     User und-robotics
15    ProxyJump csccluster

```

Figure 37. SSH `config` File with Fixes to Common Issues

5. TECHNICAL SPECIFICATIONS

5.1. MECHANICAL GENERAL STRUCTURES LIST

- Modular Chassis (Frame)
- Carbon Fiber Bucket Scoops
- Chain Drive for Bucket Ladder
- Power Screw
- Drivetrain
- Hopper
- Conveyor Belt Collector System
- Servo Mount
- ZED Camera Mount

5.2. HARDWARE LIST

- NVIDIA Jetson Nano Developer Kit (4Gb)
- Arduino MEGA 2560
- 14.8 V 4C 3300 mAh Li-Po Battery
- 14.8 V 4C 8000 mAh Li-Po Battery
- 14.8 V 4C 9000 mAh Li-Po Battery
- 2-to-1 XT60 Battery Adapter
- 30 A Self-Resetting Circuit Breakers
- Vex Talon SRX Motor Controllers
- Vex 12 V DC BAG Motors
- Vex VersaPlanetary Gearboxes
- Step-down Voltage Regulator boards
- Perfboard
- CANable USB-to-CAN adapter
- Shunt Resistor (50 A, 75 mV)

- FT5335M-FB Ultra-High-Torque Servo
- ZED 2.0 Camera
- USB Wi-Fi Antenna
- Li-Po Battery Low Voltage Alarm
- COTS Emergency Stop Switch
- 2-pole Switch
- Custom Battery Box, E-Box, Arduino Box
- Copper Power Bus Bars with Custom 3-D Printed Mount
- Various Connectors: XT60, XT90, Ring (M4), USB Passthrough, DB9 Terminal Block

5.3. SOFTWARE LIST

- Linux4Tegra
- Linux kernel 4.9
- Ubuntu 18.04 aarch64 (Bionic Beaver)
- ROS Melodic Morenia
- catkin
- Python2
- Python3
- pip
- C++
- Cmake
- XML
- URDF
- Xacro
- Gazebo
- rviz
- CTRE Phoenix API
- socketCAN
- ZED SDK
- SSH
- Bash
- Shell
- Visual Studio Code (VSCode)
- ArduinoIDE
- ArduinoOS