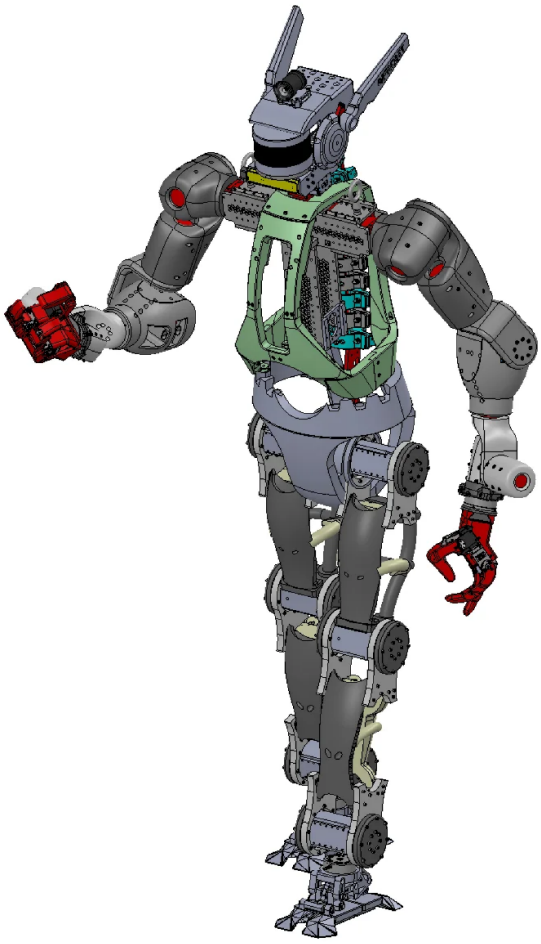
# LIFE-SIZED HUMANOID ROBOT

## ULTIMATE USER MANUAL



# Contents



# Glossary

| Term | Description |
|---|---|
| Mesh | A 3D model of a specific link or physical part of the robot. |
| URDF | Unified robotics description format. An XML file describing the transfer functions between joints and their corresponding meshes to visualize the robot. |
| API | Application Programming Interface. The protocols for connecting different software systems together. |
| GUI | Graphical User Interface. The display which users interact with. |
| IMU | Inertial Measurement Unit. Used to measure position and acceleration |
| Linux | An open source operating system based on the |

| | |
|---|---|
| | Linux kernel. Linux has many different distributions that offer a variety of features. This project uses a popular distribution of Linux named Ubuntu, specifically version 20.04. |
| NUC | Next Unit of Computing |
| ROS | Robot operating system. Open-source set of software frameworks used for robot software development |

# Brief Introduction and Setup

The advanced humanoid robot has been an ongoing project at the University of Calgary's UVS lab since 2014. It has been in continuous development by numerous capstone design teams and graduate students to one day be deployed in humanitarian aid operations and search and rescue.

This documentation was produced in the Winter Semester of 2023. There are previous copies of documentation from prior teams.

The robot's software utilizes ROS(1) Noetic on Ubuntu 20.04 LTS. ROS uses **ROS packages** to launch topics and nodes that allow the different modules to communicate. A **rostopic** is essentially a message board for **rosnodes** to publish (write) and subscribe (read) to. Rostopics run off the **roscore** service which is essential for using ROS.

To set up the ROS environment on your personal computer, it is recommended to try to use Ubuntu 20.04 and not WSL for the sake of simplicity. Many of the packages do not work when using WSL on a windows device including the Intel RealSense camera. That being said, WSL can be used for development and can be installed from the app store.

1) Install Ubuntu 20.04
2) Install ROS(1) noetic, following tutorial at: http://wiki.ros.org/noetic/Installation/Ubuntu (it is recommended you also configure the bashrc file for easy sourcing)
3) Navigate to the catkin_ws directory and run `catkin_make` to attempt to build the ROS packages. If this step fails, you are likely missing packages and dependencies that can be installed with `sudo apt-get install <package>` or `sudo pip install <package>` . Searching the error should uncover which packages to install
4) Test installation by running `source .bashrc` to source /opt/ros/noetic/setup.bash and

devel/setup.bash, then `roscore` to start the roscore service

# ROS Software Packages

Note: If developed by a third-party or external company, most of the following packages have github links in their README's.

## Bota-driver Package

The bota-driver package is a ROS package developed by the BotaSys team. This software package will provide a driver and a ROS interface for the ethercat and serial version of the rokubimini force-torque sensor. This is at the moment just a skeleton to connect to rokubimini devices and support the driver development of its firmware.

## DynamixelSDK Package

The dynamixel SDK package is a ROS package developed by the dynamixel team. The ROBOTIS Dynamixel SDK is a software development kit that provides Dynamixel control functions using packet communication. The API is designed for Dynamixel actuators and Dynamixel-based platforms. For more information on Dynamixel SDK, please refer to the e-manual below.

## Gui_tutorials Package

The gui tutorials package is a ROS package developed by the 2022/23 and prior capstone design teams to visualize the robot's URDF, interface with sensors, and to control motors.

## Imu_tools Package

The IMU_tools package is a ROS package developed by the vectornav IMU team implementing IMU-related filters and visualizers.

## Mpu_6050_release Package

The MPU_6050_release package is a ROS driver for the MPU-6050 IMU sensor on the humanoid.

## Realsense-ros Package

The realsense2_camera package was developed by the Intel RealSense team to launch a display of the Intel Realsense T265 camera through RVIZ. This is used in parallel with the URDF-RVIZ package to launch an RVIZ display of the URDF and the camera stream.

## Rosbag_to_csv Package

ROS logs all sensor data into a rosbag file. This package converts the unorganized and illegible rosbag files to easy-to-read csv (comma separated values) files that can be read from other packages.

## Rviz_camera_stream Package

The rviz_camera_stream ROS package was designed by a third party to publish RVIZ related rostopics to the localhost. This is used to integrate the simulated robot display and Realsense camera stream into the GUI.

## Sensors Package

The sensors package contains four launch files to configure and launch the force-torque sensors, vectornav IMU, VLP-16 LIDAR sensor and all of the sensors simultaneously.

## Sub-sensors Package

This package is used to create a subscriber to the IMU's rostopic to read data values from the IMU and display it in an rqt real time visual.

## URDF-RVIZ Package

The urdf-rviz package is a ROS package developed by the 2022/23 capstone design team to visualize the robot's URDF in an RVIZ window and stream it over a rostopic. This is used to have a dynamic display of the robot's orientation in the GUI.

Launching:
From catkin_ws directory…

```
roslaunch urdf-rviz urdf-rviz.launch
```

OR

```
roslaunch src/urdf-rviz/launch/urdf-rviz.launch
```

This will open an RVIZ window with a display of the URDF and a camera publisher that streams a view of the robot to: `localhost8080***`
Optionally, you can open a joint state interface to manually simulate motor control by opening urdf-rviz.launch (catkin_ws/src/urdf-rviz/launch/urdf-rviz.launch) and changing the "gui" arg to true.

## Vectornav Package

The Vectornav package creates a ROS node for Vectornav INS and GPS sensors (fancy IMU on bottom). This package provides a sensor_msg interface for the VN100, 200, & 300 devices. Simply configure your launch files to point to the serial port of the device and you can use rostopic to quickly get running.

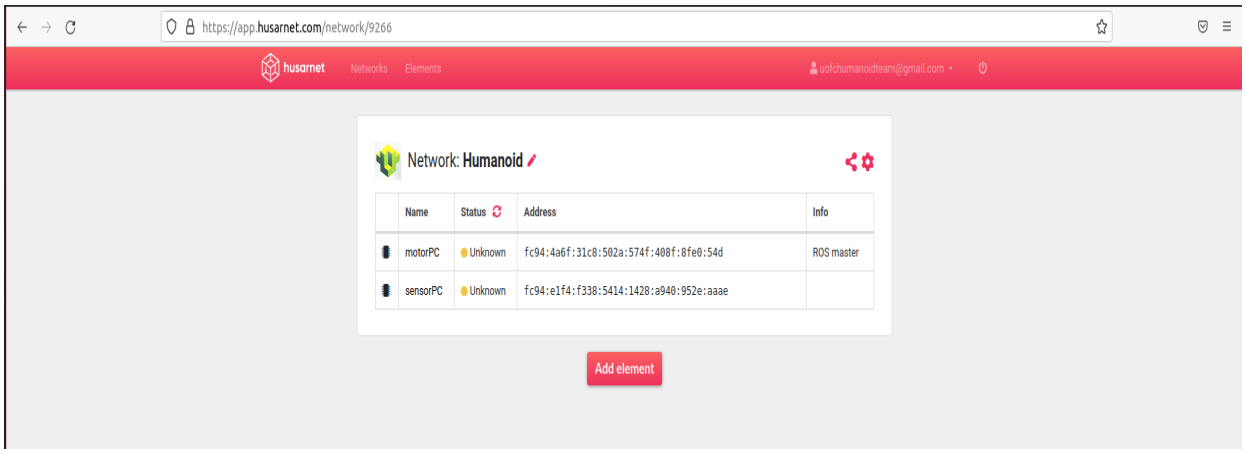
## Velodyne Package

The Velodyne package is a collection of ROS packages that support the Velodyne VLP-16 LiDAR sensor.

## Husarent

Husarnet is not a ROS package but is a peer-to-peer VPN used to enable the two PCs to connect to the same roscore service. The PCs have been configured to the same ROS_MASTER_URI so they can both create rostopics and rosnodes on the same roscore.
Login Info

Username/email: uofchumanoidteam@gmail.com
Password: HumanoidTeam1!




# Sensors

## Velodyne LiDAR VLP-16 Sensor

### Info

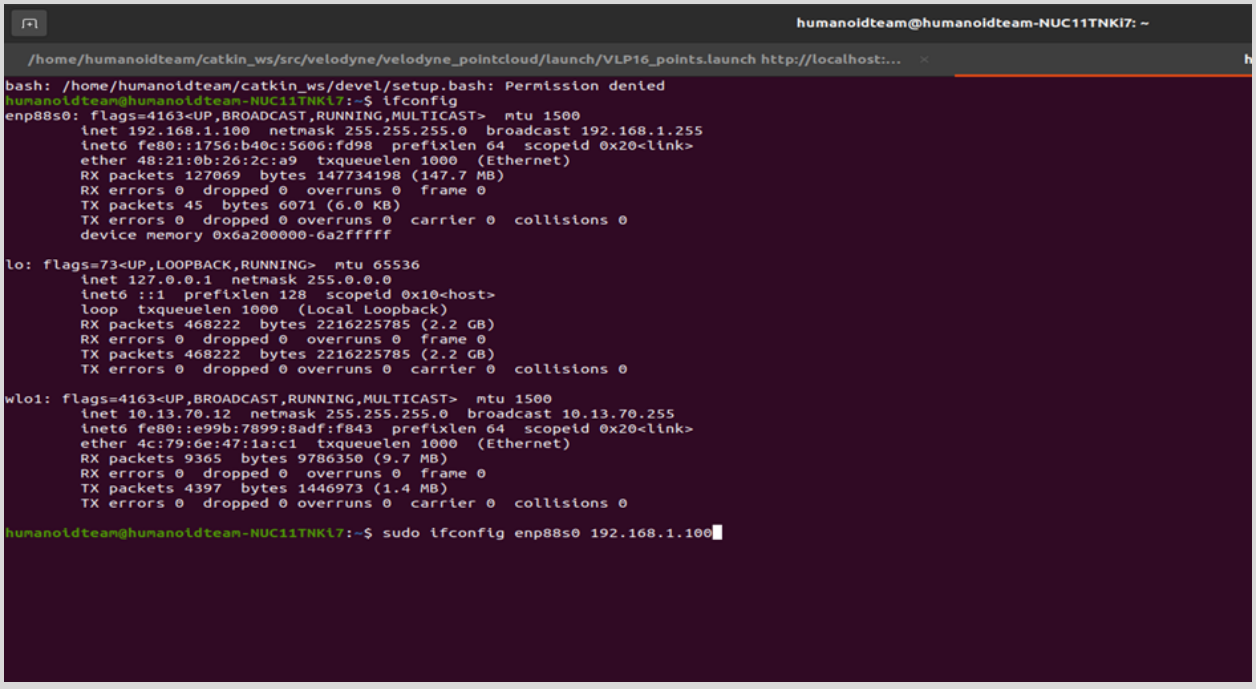
The velodyne LiDAR VLP-16 sensor is used to capture a point-cloud image of the surrounding room. The sensor's SN is AG21917705, its MAC address is 60:76:88:10:45:29, and it was manufactured on 8/7/2017.
Here is a link to the sensors manual:

- https://velodynelidar.com/wp-content/uploads/2019/12/63-9243-Rev-E-VLP-16-User-Manual.pdf

### Configuration

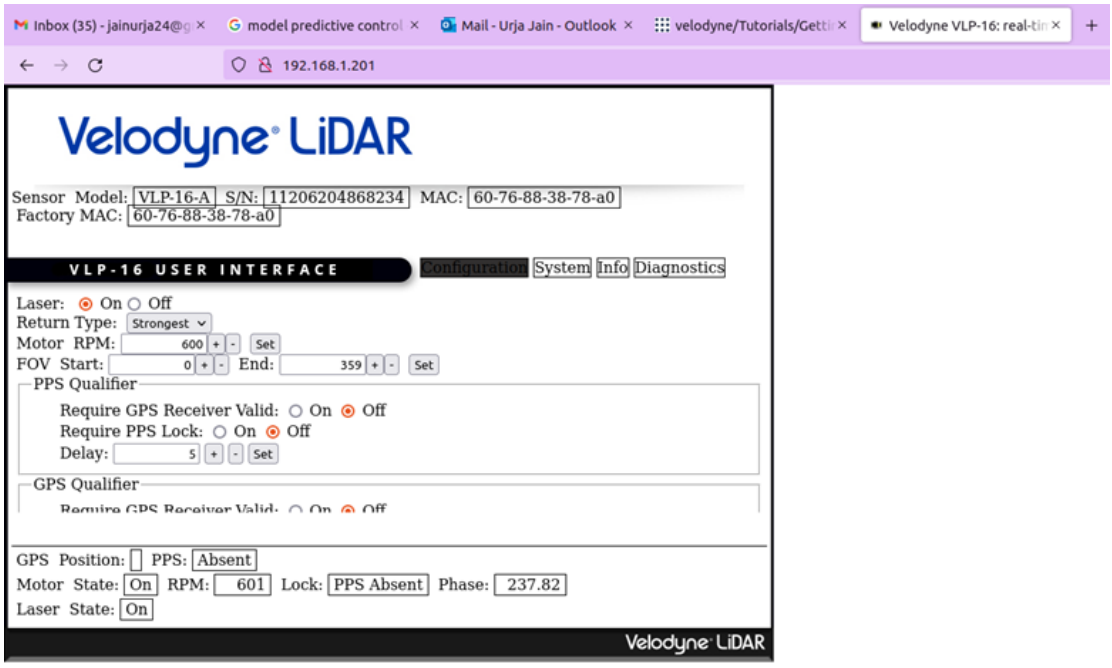
1) Follow the steps in the tutorial: http://wiki.ros.org/velodyne/Tutorials/Getting%20Started%20with%20the%20Velodyne%20VLP16
2) This will require statically assigning the IP (does not require Gnome) of the ethernet port with ifconfig. It must be in the range 192.168.1.X.
    a) Run ifconfig to see ethernet port name (enp88s0 on NUC PCs, but may be different depending on device)
        i) Must have ifconfig installed via net-tools.

b) Enter into the terminal: sudo ifconfig enp88s0 192.168.1.100

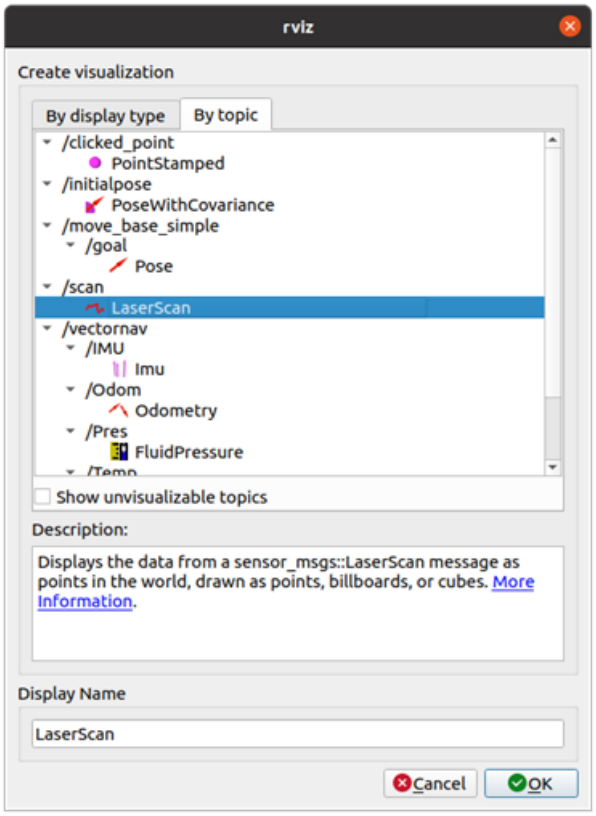



3) Add a static route to the LIDAR's IP address. The factory set address is 192.168.1.201; however, we were unable to connect to it.
    a) Enter into the terminal: sudo route add 192.168.XX.YY enp88s0
4) After running the commands and making the connections, open your browser and access the following network address- 192.168.XX.YY (192.168.1.201, but may be different if using different LiDAR) to check the configurations.

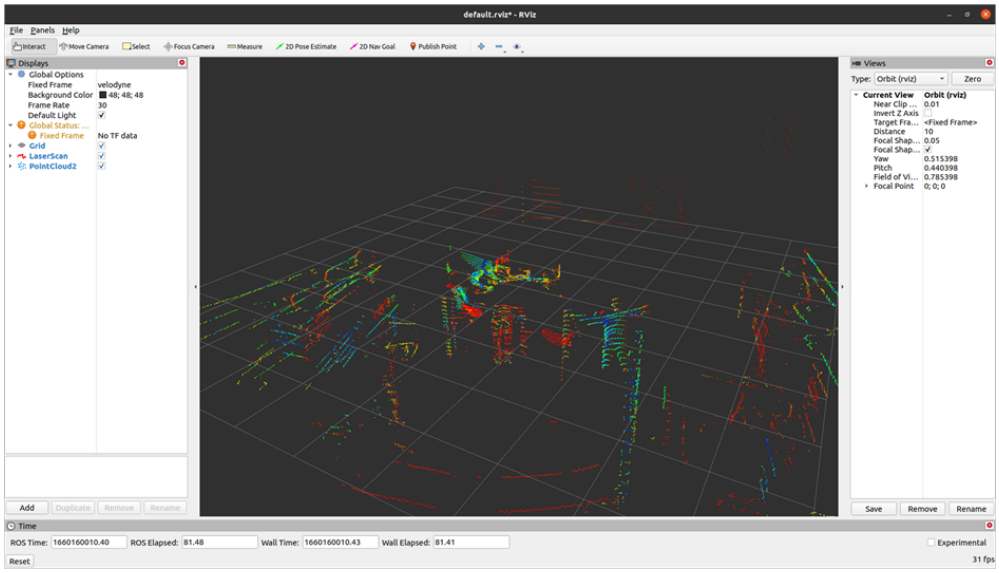
    This is the following window that should open (see notes in next steps below):




5) Connect the RP LiDAR through the ethernet with your CPU or NUC, and also through a power source.
6) Open the GUI and click on LiDAR and Rviz will open. Add the required topic.




7) To visualize a proper Laserscan and/or pointcloud data, open the Velodyne web browser (step- 5) and set the Return Type to Strongest.



### Next Steps

- Determine issues with sensor interfacing and connect.
    - Either the sensor's network interface is broken or the interface box is improperly wired.
        - Network interface is factory set to 192.168.1.201 however we were unable to reach the configuration browser page with LiDAR sensor on the humanoid robot.
            - We were able to connect with another LiDAR sensor in Dr. Alex's lab and reach its configuration page (on the robot dog at this current point in time (April 2023)).
        - This leads us to believe there may be issues with the interface box (hidden under the main power button) as it was re-wired to fit the robot.
- Embed Velodyne LiDAR stream into GUI.

## Intel RealSense T265 Camera

### Info

Developed by the Intel Realsense team. Captures a fisheye image of the camera's display and also has a built-in IMU. The data sheet for this camera is below:

- https://dev.intelrealsense.com/docs/tracking-camera-t265-datasheet

### Configuration

1) Install Intel Realsense SDK from: https://github.com/IntelRealSense/librealsense/blob/master/doc/distribution_linux.md
2) Install T265 wrapper from: https://github.com/IntelRealSense/realsense-ros
3) In the Ubuntu terminal, determine which port is being used by running `dmesg | grep realsense` or in some cases `dmesg | grep movidius`
4) Run `sudo chmod a+rw /dev/<port>` to enable permissions
5) Run `roslaunch realsense2_camera demo_t265.launch` to launch an RVIZ window with the camera display

### Next Steps

- Embed the Realsense IMU 3D orientation rendering into the GUI
- Enable head control from realsense camera page

## Vectornav IMU

### Info

The vectornav IMU (inertial measurement unit) is used to record the movement of the robot, capturing its acceleration, inertia, and relative position.

### Configuration

1) Follow the tutorial at: https://github.com/RevolveNTNU/r18dv_vectornav_old

a) Instead of running `roslaunch vectornav v200.launch`, run `roslaunch vectornav vectornav.launch`
b) If using WSL, you will have to attach your physical port to WSL. Refer to configuration for the Intel Realsense camera.
c) Run `sudo chmod a+rw /dev/<port>` to enable permissions

### Next Steps

- Embed graphical displays of data in GUI rather than opening separate windows
- Coordinate with Realsense IMU to determine movement of body and balance controller

## BOTA Systems Force Torque Sensors

### Info

Developed by Bota Systems to read force-torque data from sensors. These rokubimini force-torque sensors measure the forces in the x, y and z directions as well as the applied torques and optionally the temperature.

### Configuration

1) Go through the tutorial on this website: https://gitlab.com/botasys/bota_driver
2) If you are using WSL, you will have to attach your physical port to WSL…
    a) Open a windows terminal as admin
    b) List available ports to connect to WSL: `usbipd wsl list`
    c) Attach required port: `usbipd wsl attach -b <bus-id>`
    Where bus-id is 3-3, 3-2, etc
3) Permission errors can be solved with `chmod a+rw /dev/<port>`
4) Display the data on the serial port by running `cutecom` and selecting the /dev port connected to the sensor

### Next Steps

- Embed force-torque data into GUI
- Enable measurement readings from multiple force-torque sensors

# Graphical User Interface

### Information

The graphical user interface (GUI) was developed by the 2022/23 capstone design team to visualize the robot's URDF, run demos, control motors, and interface with sensors. It leverages Kivy, an open-source python UI framework, KivyMD, which is compliant with Kivy and adds material design widgets, Python 3.8 and ROS.

Kivy: https://kivy.org/doc/stable/

KivyMD: https://kivymd.readthedocs.io/en/1.1.1/

## Configuration

1. Prerequisites:
    a. ROS-Noetic installed with all GUI dependencies:
        i. Kivy,KivyMD and Python 3.8.
    b. Motor PC and Sensor PC both operational.
    c. RealSense Camera plugged into SensorPC.
2. Open three ubuntu terminals:
    a. One on Motor PC
    b. Two on Sensor PC
3. Source each terminal:
    a. From the root directory on each teminal, first run:
        i. `source .bashrc`
4. On the Motor NUC PC, run:
    a. `roscore t`
        i. This will launch the ros node for the PCs to communicate on through the ROS_MASTER_URI @ https://master:11311
5. On the Sensor NUC PC's first terminal, launch the URDF RVIZ and RealSense Camera doing the following:
    a. From the root of the catkin workspace (catkin_ws) run:
        i. `roslaunch ./src/urdf-rviz/launch/urdf-rviz.launch`
        ii. This will launch the URDF as well as Intel RealSense Camera
    b. On the RVIZ screen, under the 'RobotModel' and 'Image1' Displays, click the checkbox 'Camera' to initialize the RealSense camera so it works for the GUI.
6. On the Sensor PC's second terminal, from the root of the workspace again, launch the GUI using the following command:
    a. `rosrun gui_turtorials mc_gui.py`
        i. This will launch the NUC PC Main Screen as seen below.

**Side notes -** The option to run using 1 NUC-PC is possible; however, ROS_MASTER_URI must be changed from https://master:11311 to https://localhost:11311. This is only for the NUC-PC. On any personal laptop simply following the steps above but with three terminals on the same device.

## Controlling the Humanoid

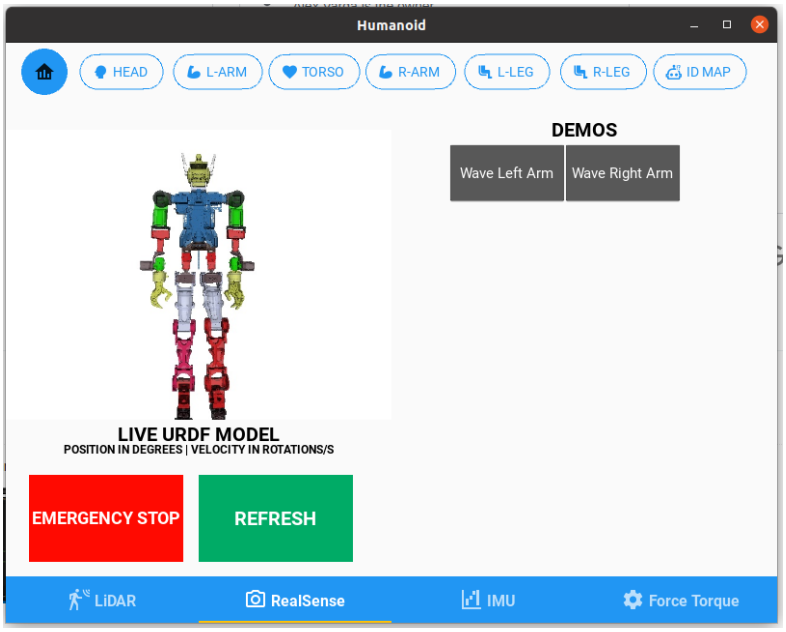
1. Prerequisites:
    a. Prerequisites above.
    b. Motor control cable plugged in.
    c. If the device used operates on Windows, one will need to connect their windows ports to WSL ports. To do so follow the link below:
    https://devblogs.microsoft.com/commandline/connecting-usb-devices-to-wsl/

2. Once connected, you should be able to view the ports by running:
    a. Dmesg | grep tty
        i. You should see ports /dev/ttyUSB0 to /dev/ttyUSB3 in the output
3. Then you must change the permissions on them so that you can control from the GUI. This can be done by running:
    a. Sudo chmod 666 /dev/ttyUSB0 t
        i. This must be done for /dev/ttyUSB0, /dev/ttyUSB1, /dev/ttyUSB2, and /dev/ttyUSB3
4. Once the GUI is launch, there's are two ways to control the robot:
    a. Demo Scripts
        i. Click buttons to run.
    b. Motor Control Panels
        i. Navigate to respective control panel pages.
        ii. Control specific motors via text input or buttons.
        iii. Publish changes and observe movement

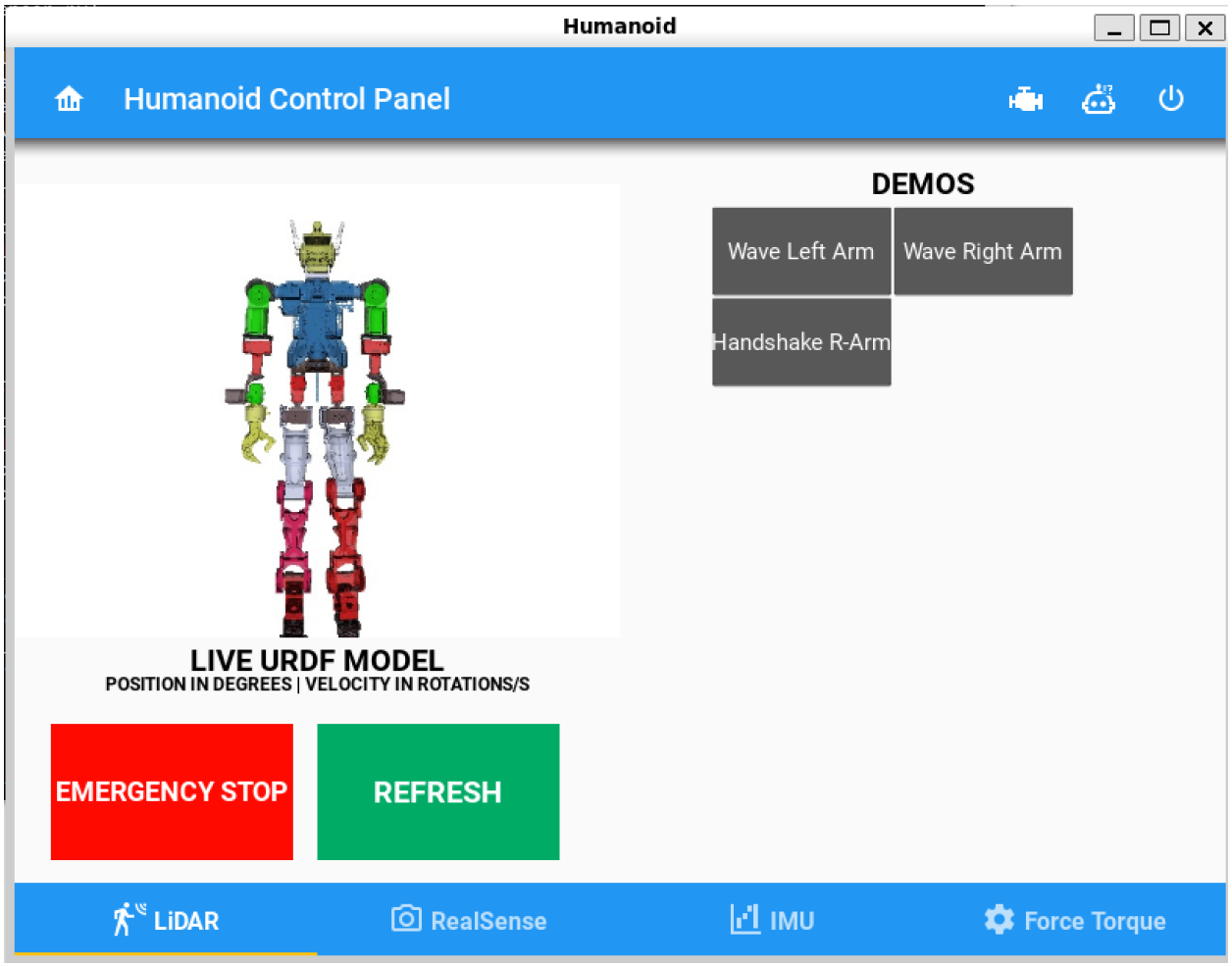
## Components

Notes:

- In all screen shots of the GUI below you will notice a toolbar on the top of the app. This was a component originally implemented as the GUI work was completed on our own devices, both Windows 10 machines using WSL. However, when migrating the GUI to the linux based NUC PC's used for the humanoid the application would no longer run.
- Kivy does not give errors when it does not compile properly (as you will find out if you develop the GUI more) so in order to determine what was causing it to break we commented out everything and slowly uncommented chunks at a time until we discovered it was the MDTopAppBar KivyMD component that would break it.
- We attempted to determine why and find a fix for it, but we could not and ultimately had to change the top app bar to buttons (the GUI on the NUC PCS). This issue also arose with a few other components from the MD suite such as MDFloatingActionButtons.
    - We compared versions, dependencies, and files between our own machines and the NUC and the only difference we noticed was the OpenGL version being used of which we found out that it does cause breaking changes in Kivy but we could not find anything about the MDTopAppBar.



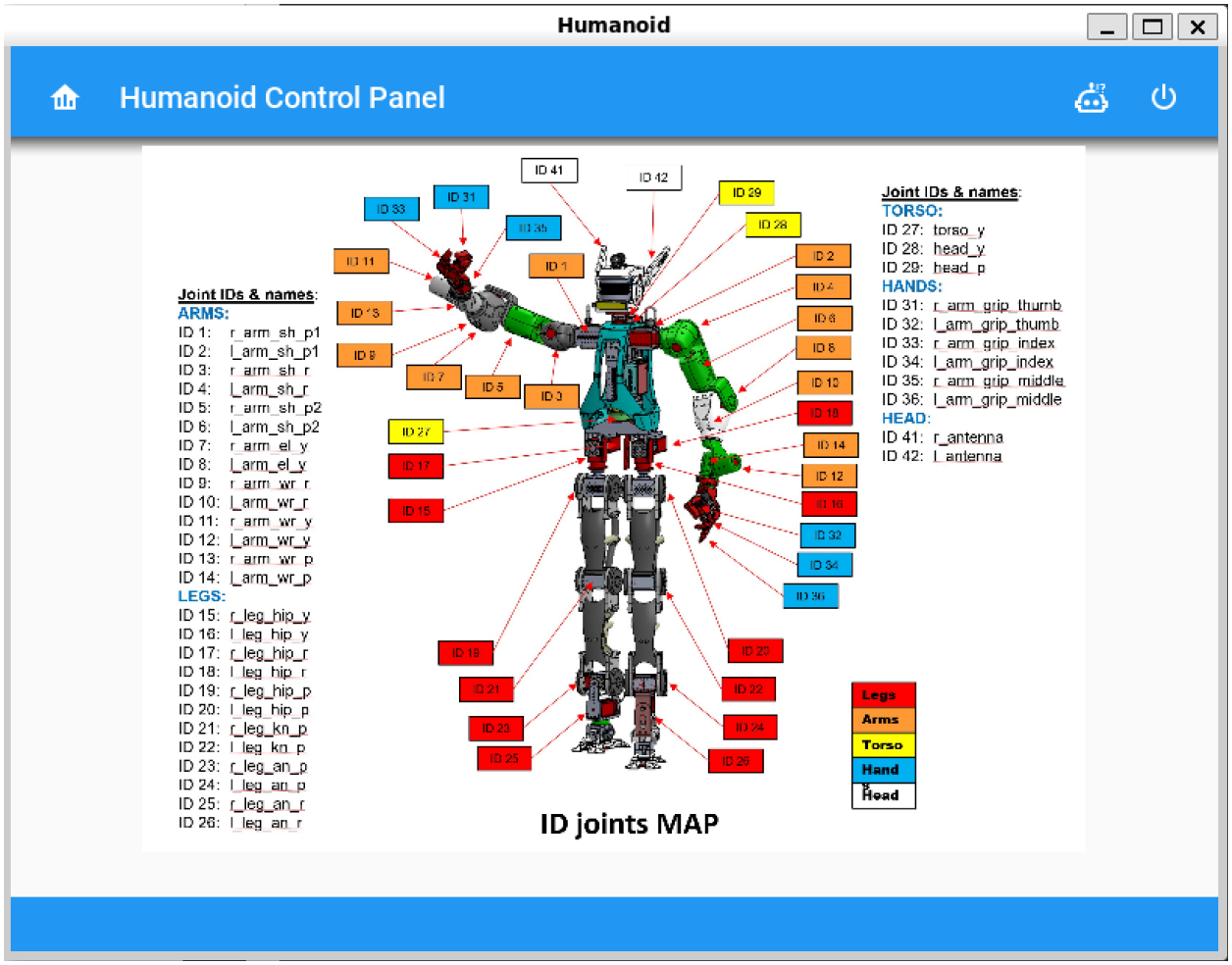

**NUC-PC Main Menu Screen**



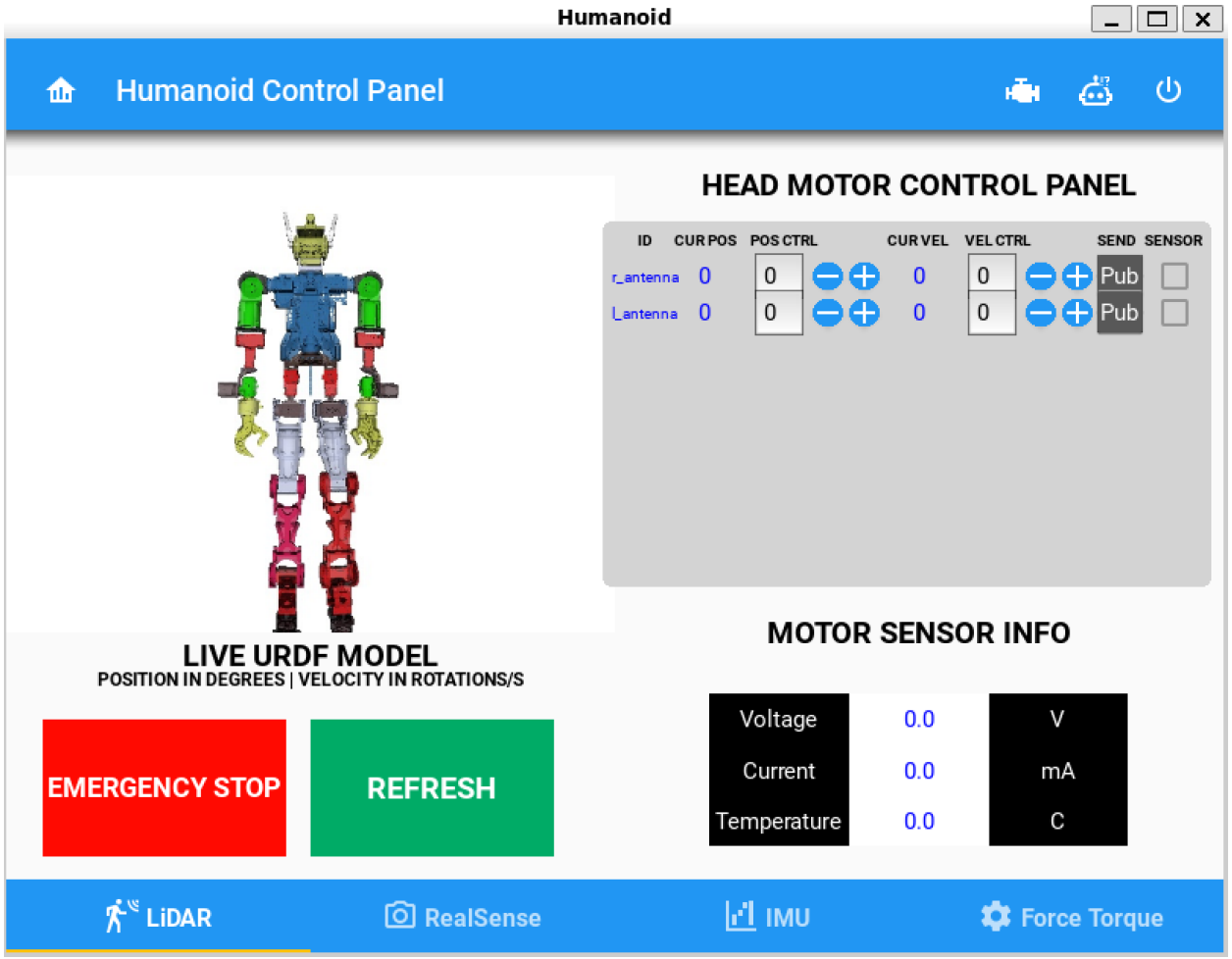
**Laptop Main Menu Screen**

**Functions:**

| Function | Component | Triggers |
|---|---|---|
| Return to Main Menu | TopAppBar Button | Home Icon Top Left |
| Change to Motor Control Screen | TopAppBar Button | Motor Icon 1st Top Right |
| View Motor ID Map | TopAppBar Button | Robot Head Icon 2nd Top Right |
| Close GUI | TopAppBar Button | Power Icon 3rd Top Right |
| Run Demos | Gray Buttons | Respective buttons |
| System Emergency Stop | Red Button | Button |
| Refresh URDF Joint State | Green Button | Button |
| LiDAR Sensor Screen | Tab | LiDAR Tab |
| RealSense Camera Screen | Tab | RealSense Tab |
| IMU Sensor Screen | Tab | IMU Tab |
| Force Torque Screen | Tab | Force Torque Tab |




**Laptop Humanoid Joint ID Map Screen**

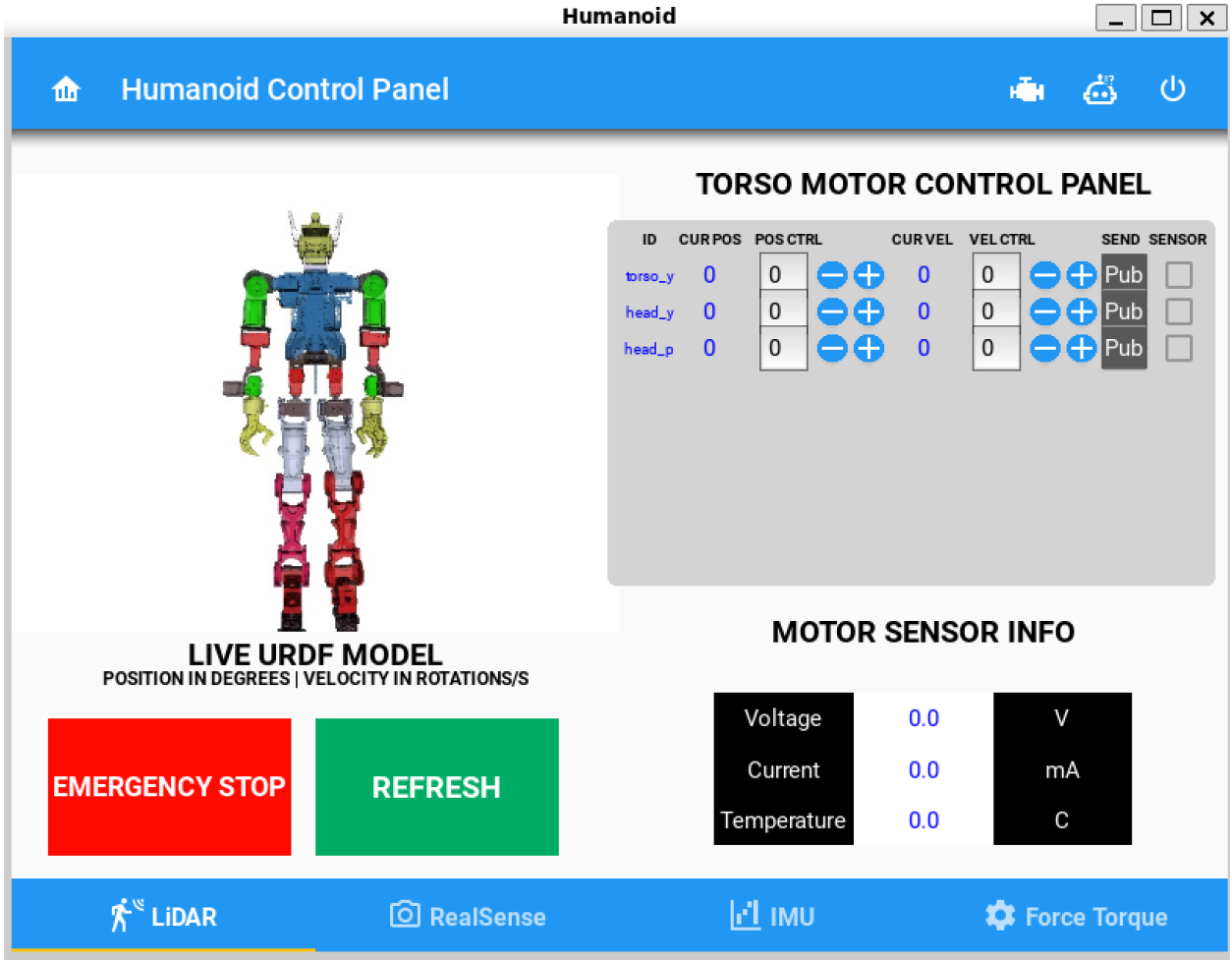



**Laptop Humanoid Head Motor Control Panel Screen**

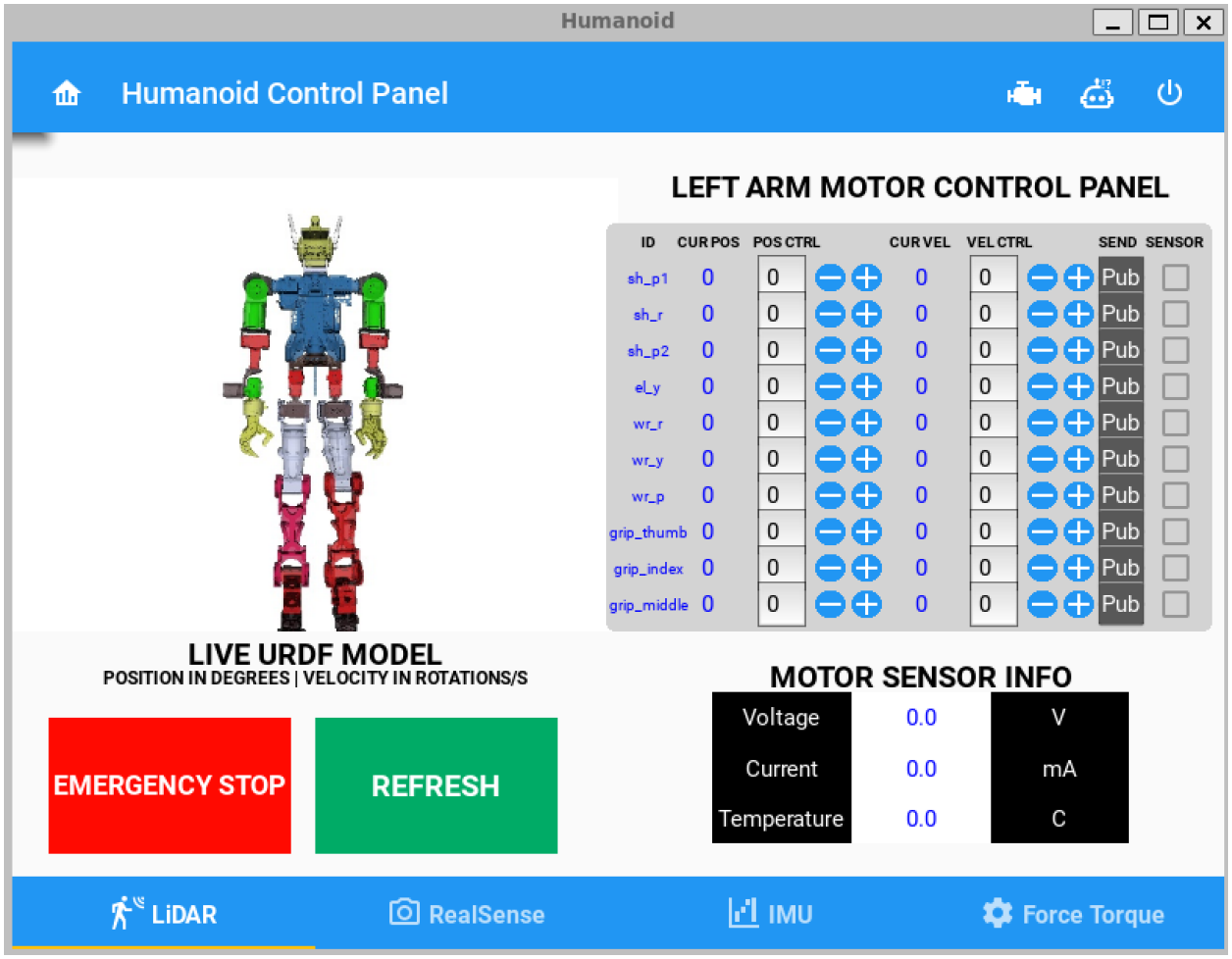
**Functions:**

| Function | Component | Triggers |
|---|---|---|
| Return to Main Menu | TopAppBar Button | Home Icon Top Left |
| Change to Motor Control Screen | TopAppBar Button | Motor Icon 1st Top Right |
| View Motor ID Map | TopAppBar Button | Robot Head Icon 2nd Top Right |
| View Current Position of Specific Motor | CURR POS Label | Reactive to text input and buttons |
| Text Input Position Control | POS CTRL Text Input Box | Manual Input |
| Button Position Control | Blue Minus/Plus Buttons | Button |

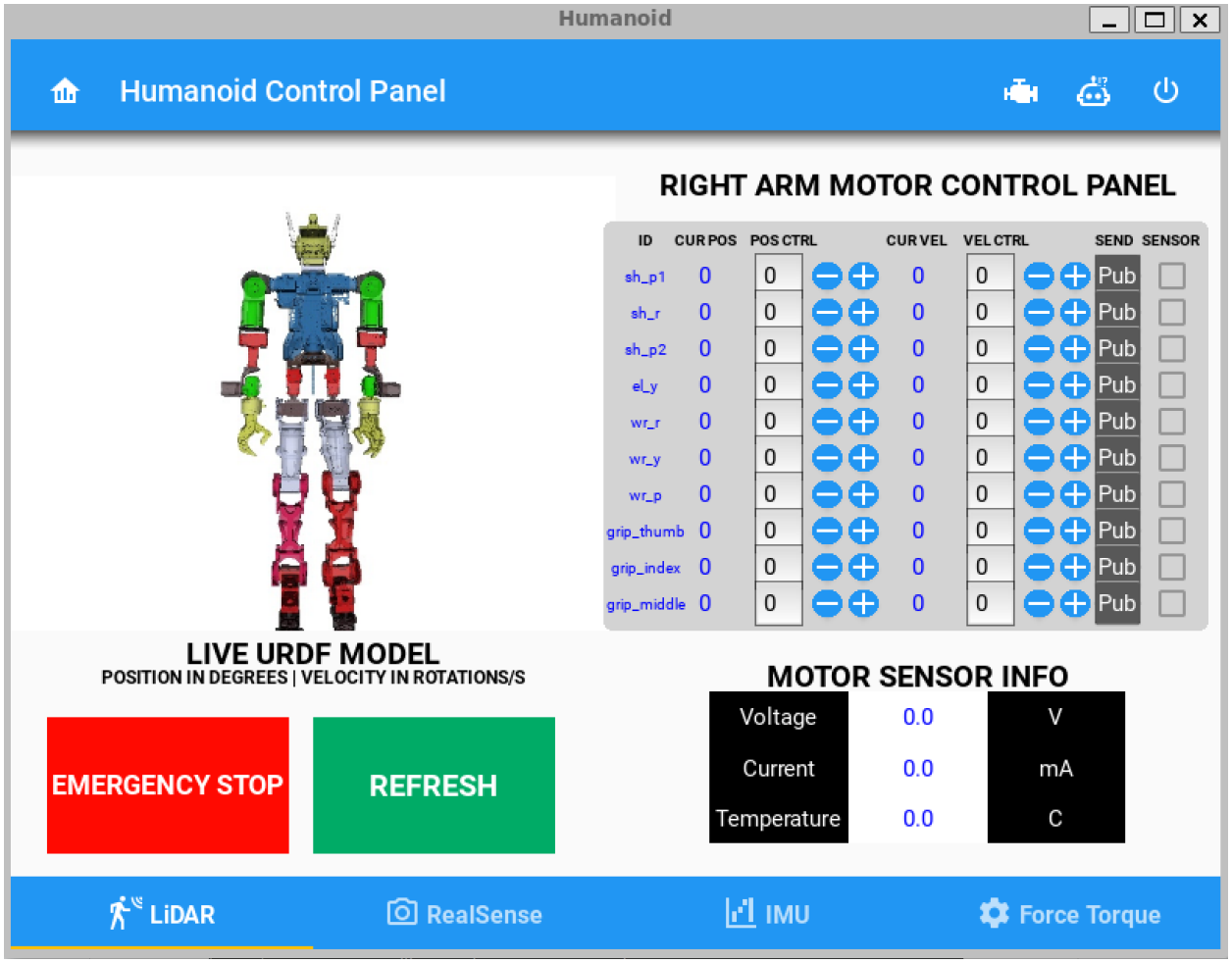
| | | |
|---|---|---|
| View Current Velocity of Specific Motor | CURR VEL Label | Reactive to text input and buttons |
| Text Input Velocity Control | VEL CTRL Text Input Box | Manual Input |
| Button Velocity Control | Blue Minus/Plus Buttons | Button |
| Publish Motor Control Params to Humanoid and URDF | Pub Buttons | Buttons |
| View Motor Sensor Info | Sensor Checkboxes | Checkbox |



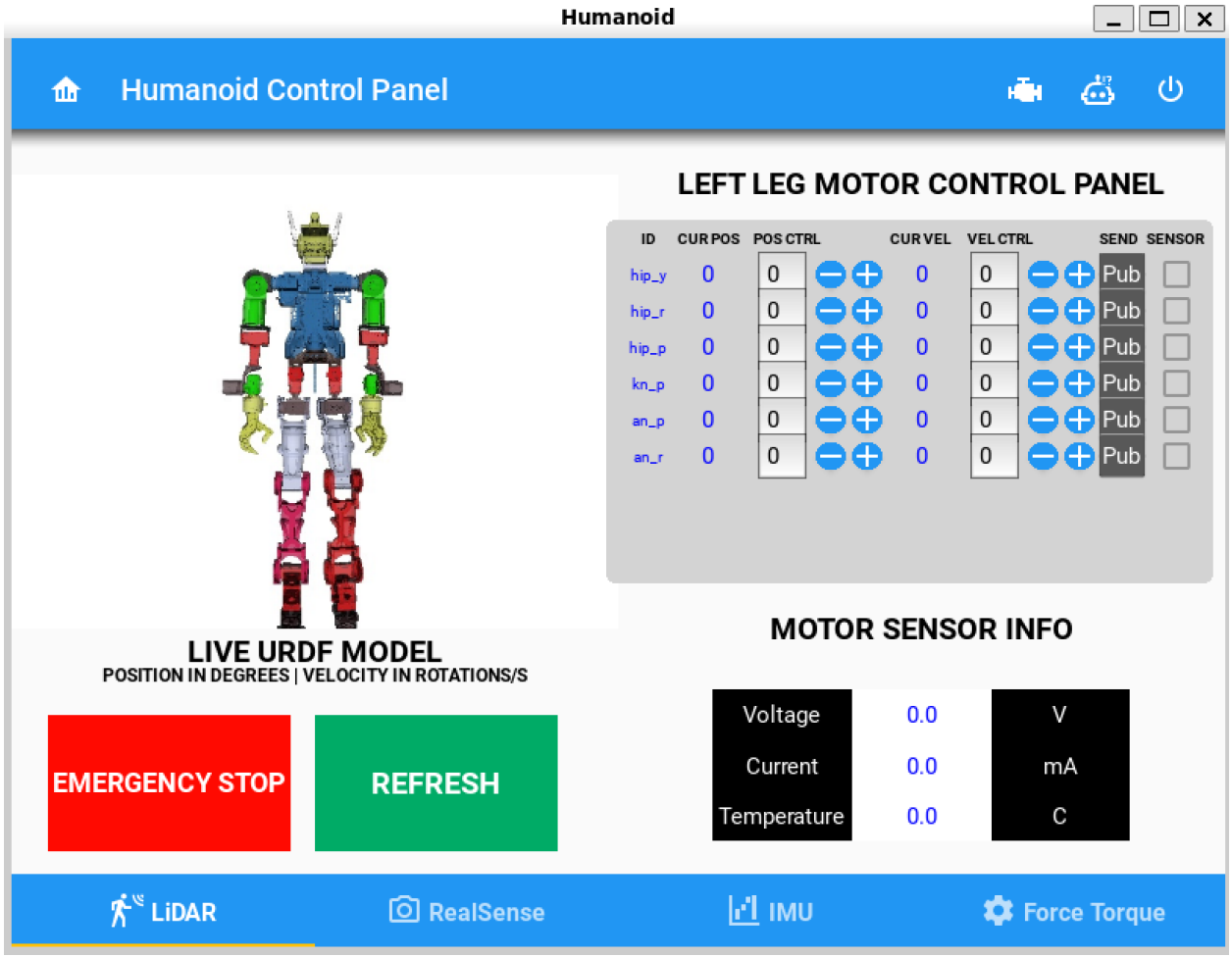

**Laptop Humanoid Torso Motor Control Panel Screen**



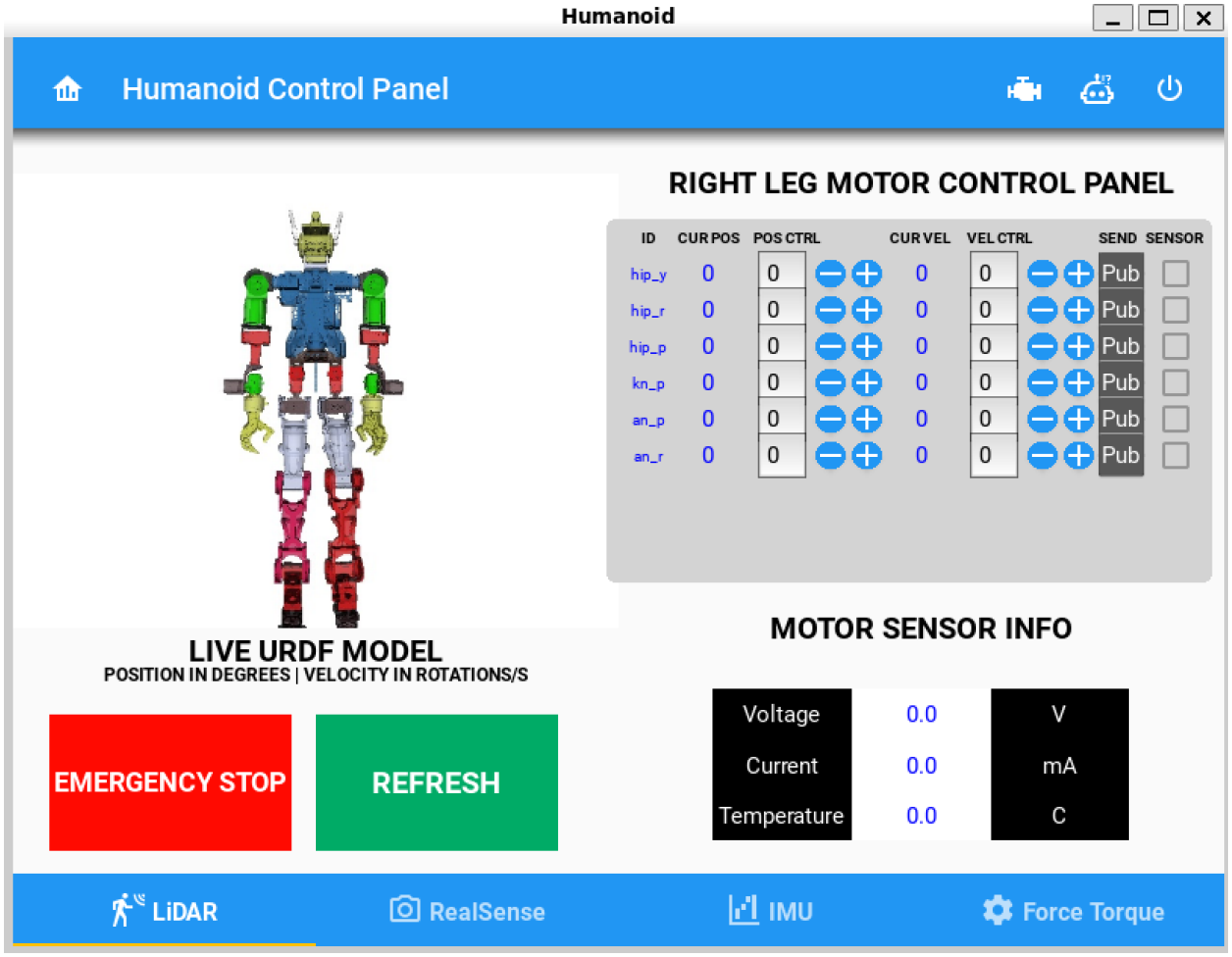

**Laptop Humanoid Left Arm Motor Control Panel Screen**



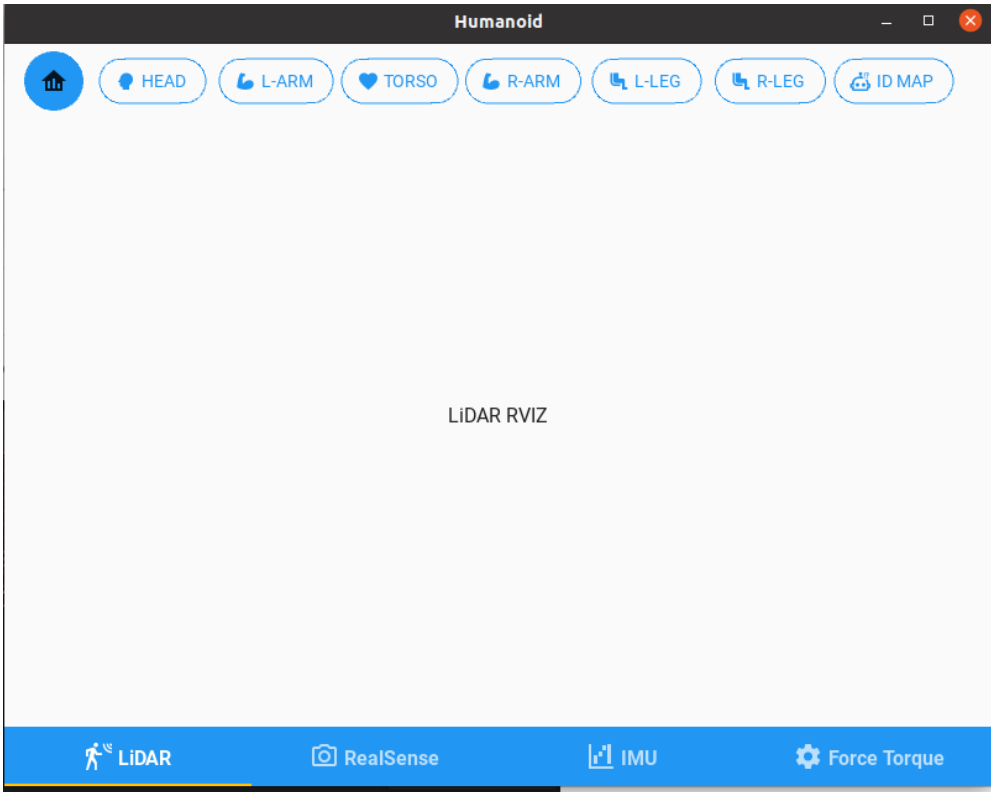

**Laptop Humanoid Right Arm Motor Control Panel Screen**



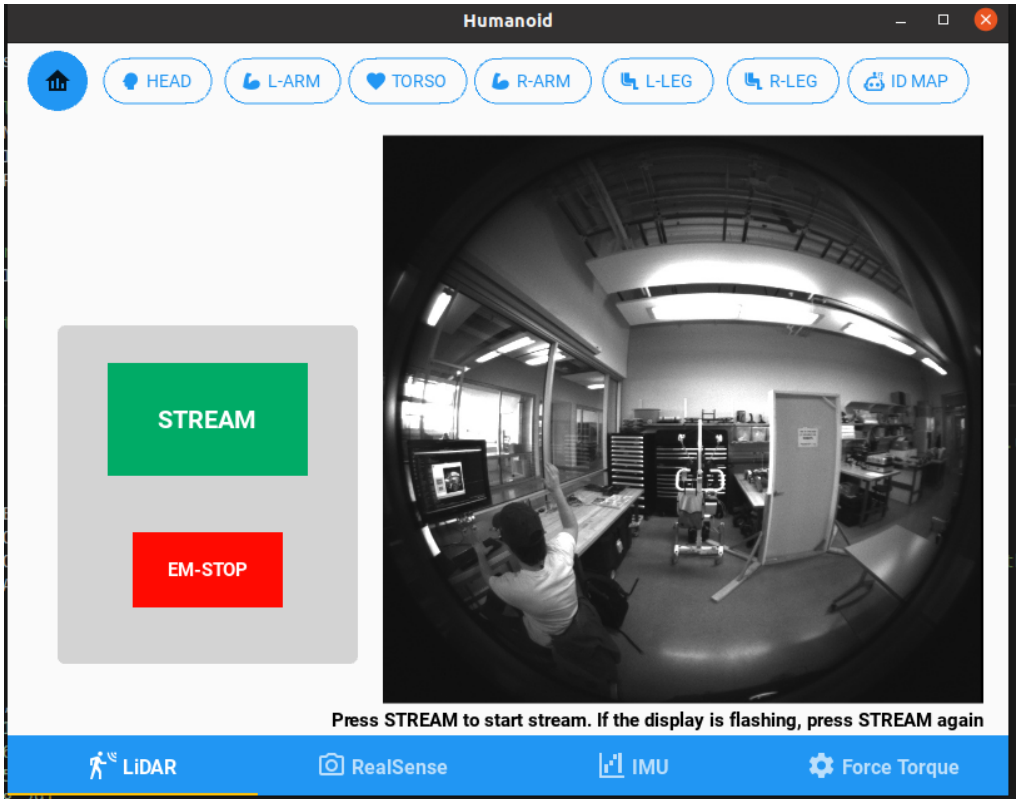

**Laptop Humanoid Left Leg Motor Control Panel Screen**



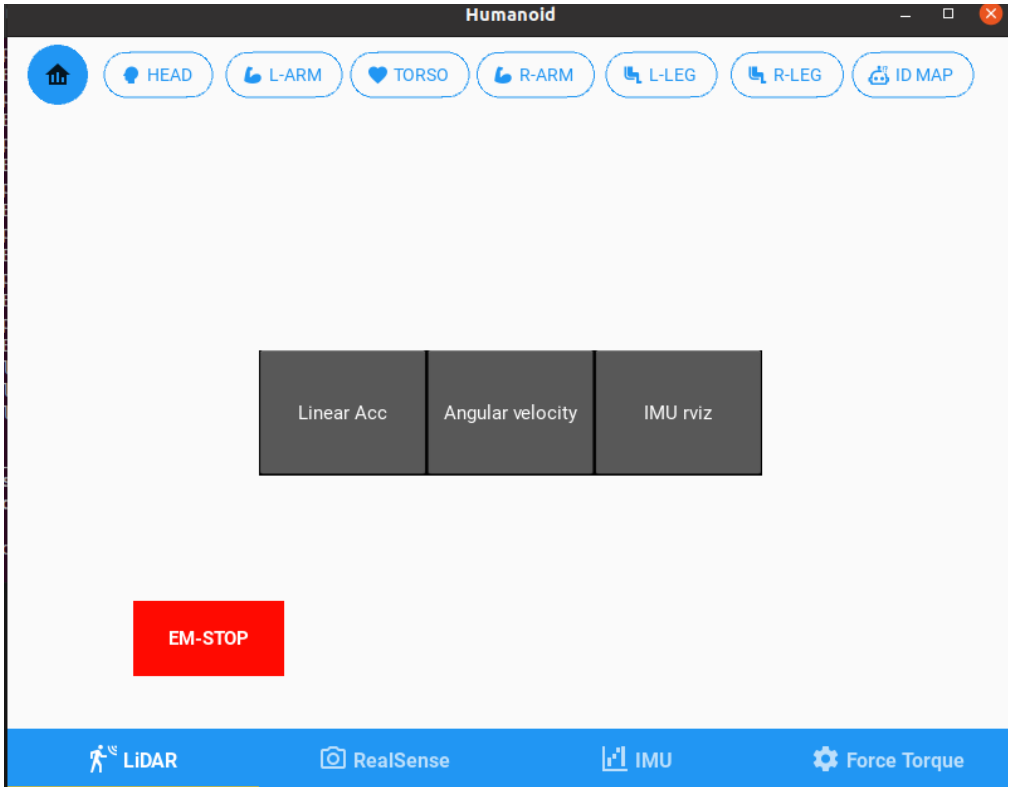

**Laptop Humanoid Right Leg Motor Control Panel Screen**



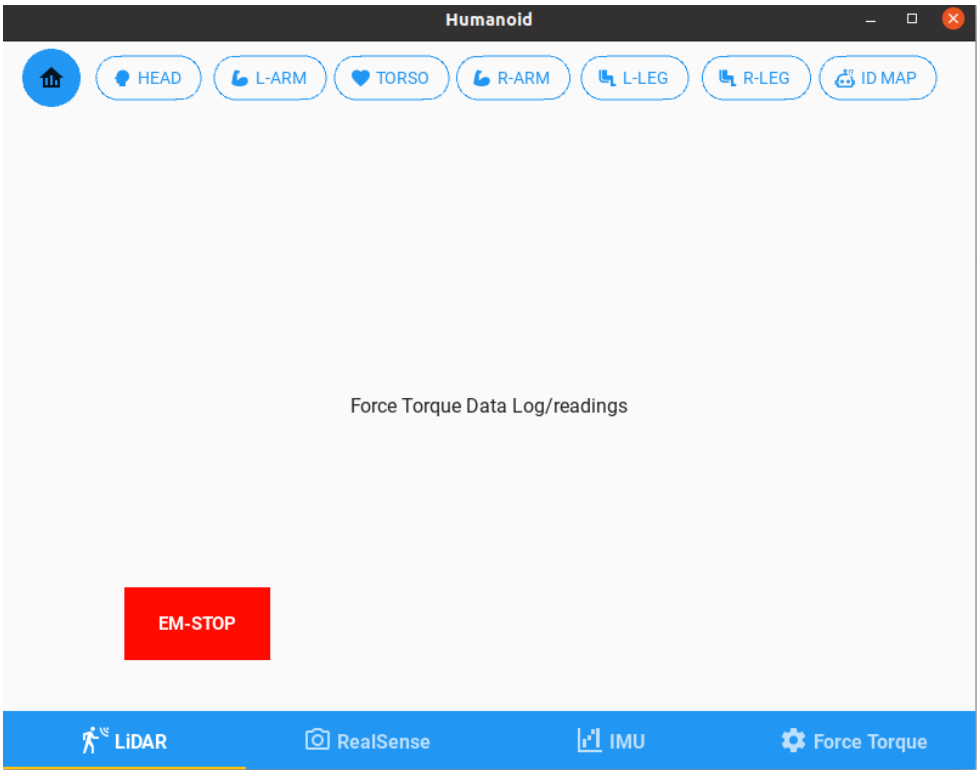

**Humanoid LiDAR Sensor Screen**




**NUC-PC Humanoid RealSense Camera Screen**




**Humanoid IMU Sensor Screen**




**Humanoid Force Torque Sensor Screen**

## Next Steps

- Modify init-urdf() function in mc_guy.py to launch URDF and Realsense with the rosrun of the GUI
- Embed LiDAR sensor stream.
- Embed Force Torque data readings.
- URDF Joint State controller moving with Demo scripts.

## Debug

**Error Type 1:**

```
[rosrun] Couldn't find executable named butt_gui.py below /home/glorycode/catkin_ws/src/gui_tutorials
[rosrun] Found the following, but they're either not files,
[rosrun] or not executable:
[rosrun]   /home/glorycode/catkin_ws/src/gui_tutorials/src/butt_gui.py
```

Fix:

1. From catkin_ws directory run:

```
Sudo chmod +x ./src/gui_tutorials/src/{filename}
```

# Dynamixel Wizard

Whenever the motor firmware malfunctions by accessing the wrong parameters, the motor firmware, parameters, and Ids can be factory reset using the official Dynamixel wizard software.

**How to access Dynamixel Wizard**

1. Download the executable file using the below link by navigating to the "install: Linux" subsection.
    a. https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/#install-linu x
2. Change the permission of the executable file using the commands below listed in the image.

Enter the following command to change the permission.

```
$ sudo chmod 775 DynamixelWizard2Setup_x64
```

Run the install program.

```
$ ./DynamixelWizard2Setup_x64
```

Click on `Next` button to proceed installation.

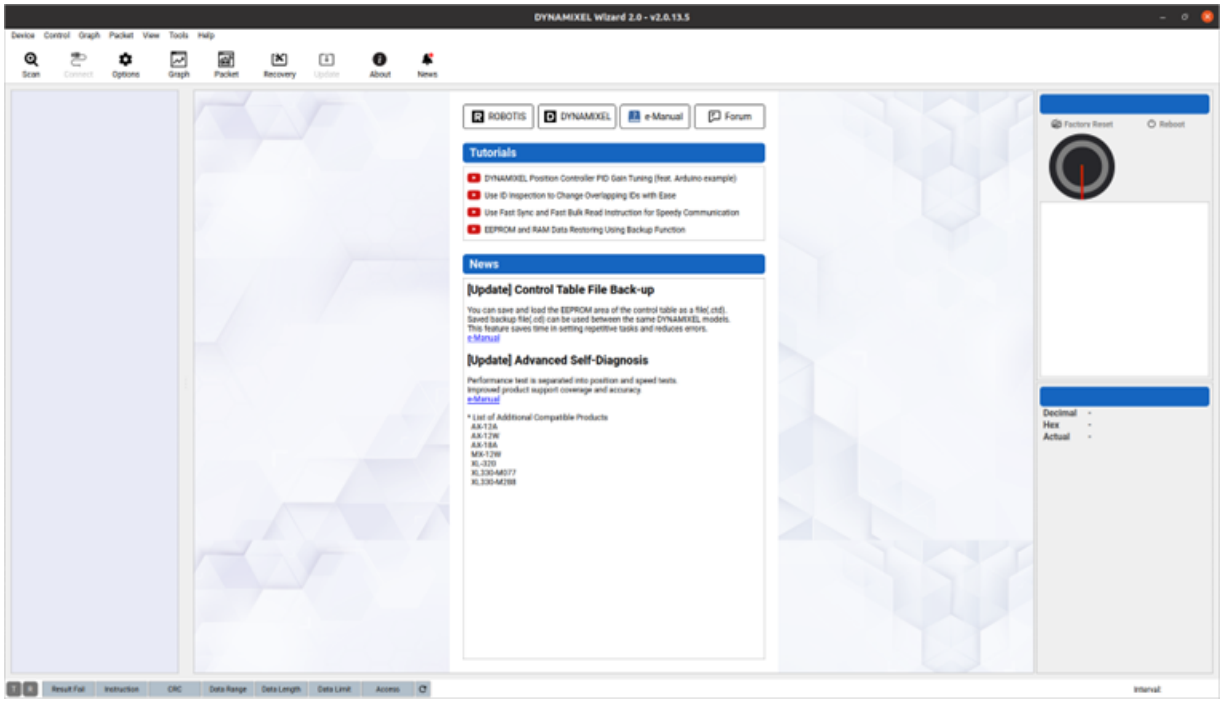
After completing the installation, please add account id to dialout group in order to access the USB port. Replace the in the command below to your actual user id.

```
$ sudo usermod -aG dialout <your_account_id>
```

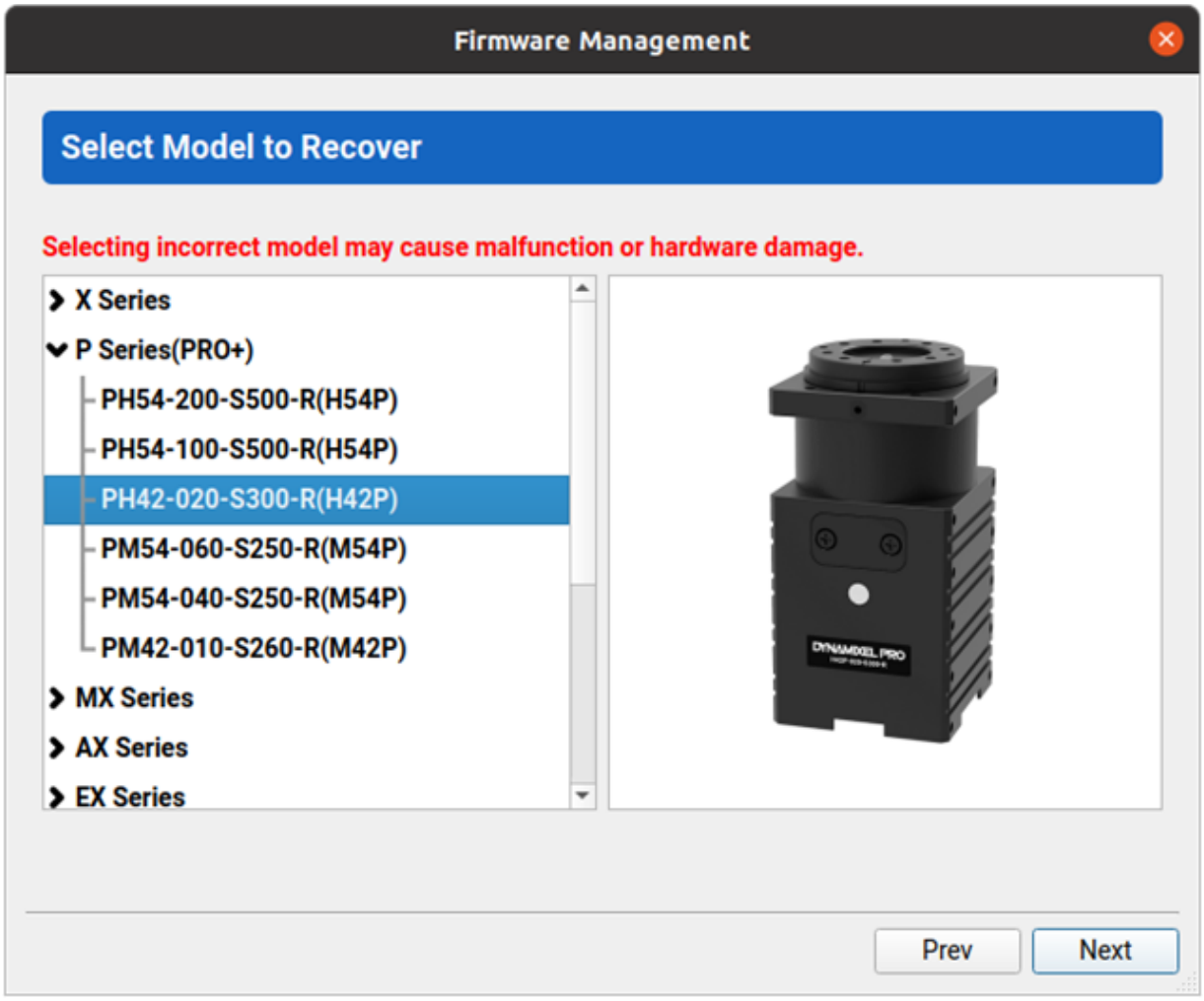
Reboot in order the changes to be effective.

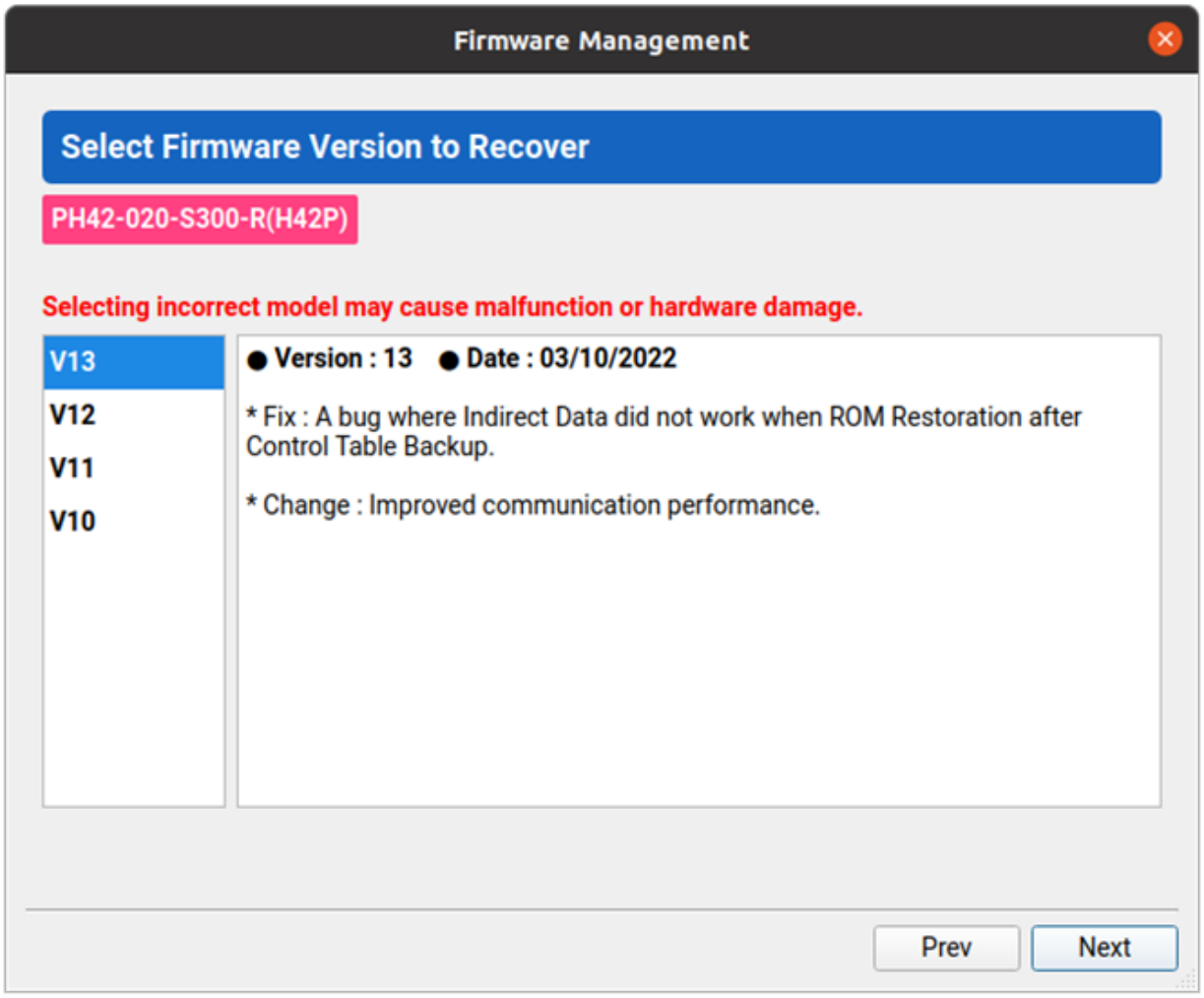
```
$ reboot
```

3. Once the dynamixel software is downloaded, connect all the motors to the CPU and note down the used port. The main software page looks like the figure below
4. The recovery option in the toolbar should be clicked.

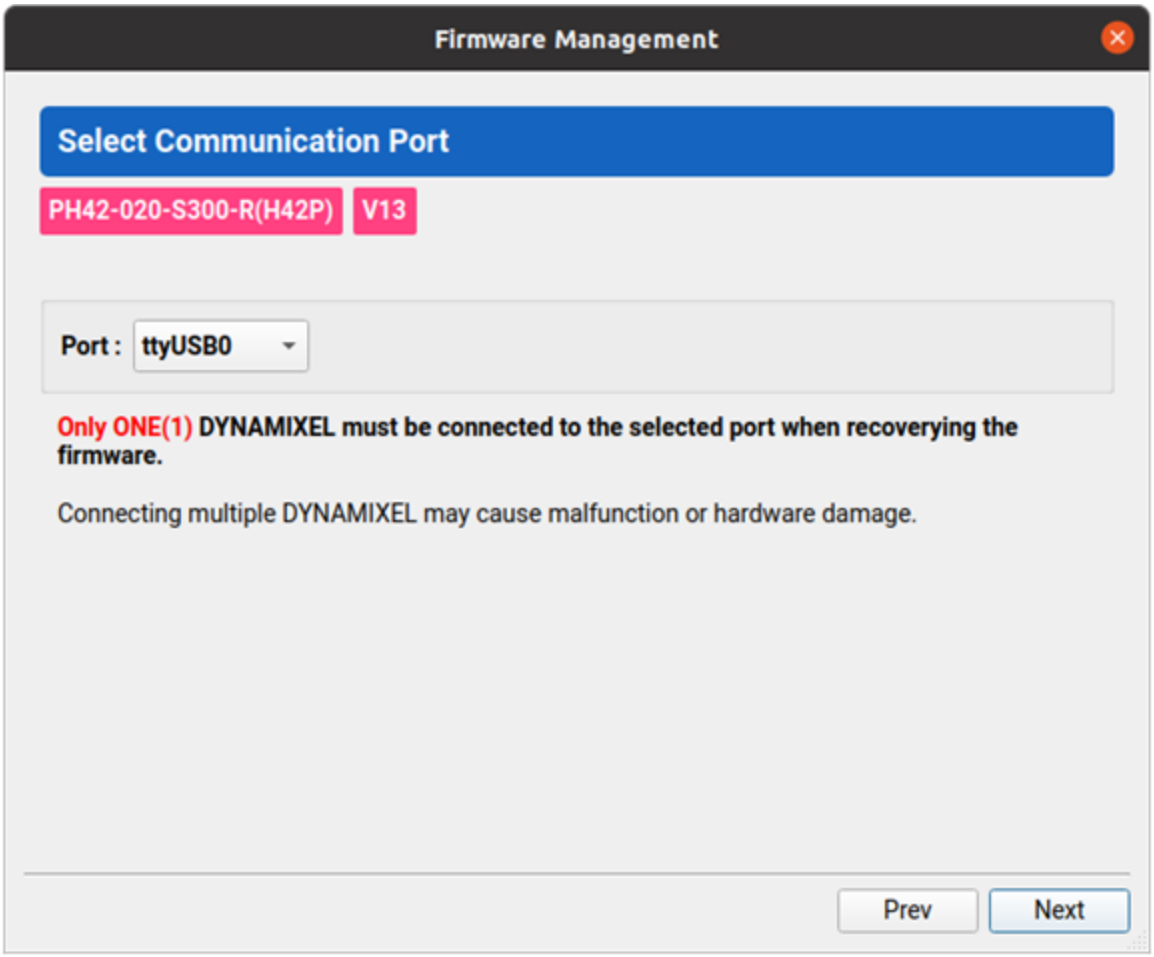



5. Motor series and firmware version should be selected. Note motor series should be handled with utmost care because selecting the wrong series can tamper with the motors permanently.
6. The motor series and version is shown in the figures below

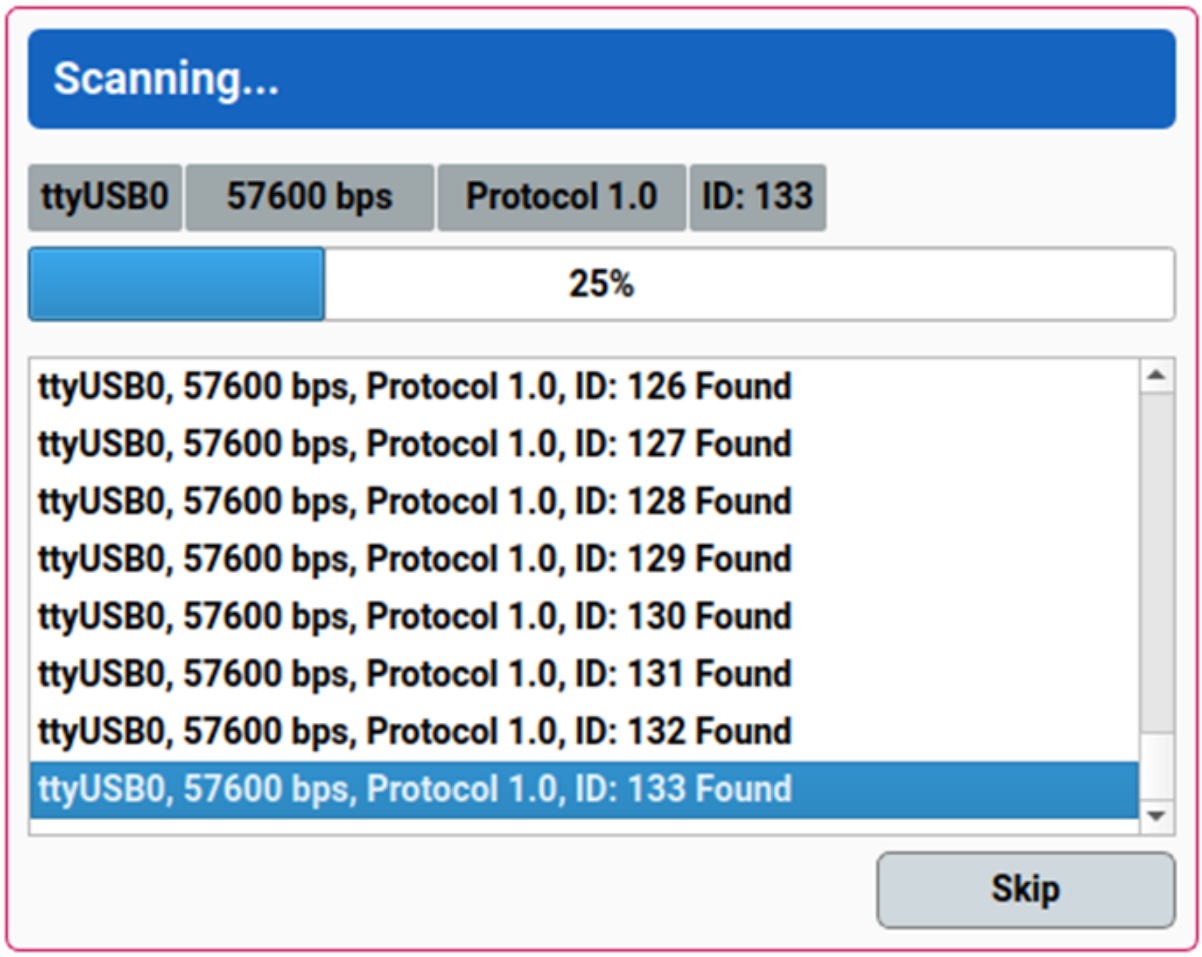

Firmware Management
Select Model to Recover
Selecting incorrect model may cause malfunction or hardware damage.
X Series
P Series(PRO+)
PH54-200-S500-R(H54P)
PH54-100-S500-R(H54P)
PH42-020-S300-R(H42P)
PM54-060-S250-R(M54P)
PM54-040-S250-R(M54P)
PM42-010-S260-R(M42P)
MX Series
AX Series
EX Series
DYNAMIXEL PRO
Prev
Next




7. Select the appropriate ports




8. Disconnect the power supply from the motor for a few seconds. The recovery process will start as soon the power supply is connected back to the motors.




9. All parameters can be accessed for the scanned motors, and the factory reset can be done

# Intel NUC 11 PRO PC's

## Information

The two Intel NUC PC's communicate through a peer-to-peer VPN called Husarnet. The humanoid robotics team has launched the VPN and have attached the motorPC and sensorPC. The network is configured so the motorPC runs roscore and the sensor PC can connect to the same service to share rostopics.

## Sensor PC Login Info

Username: robotis
Password: 111111

### Motor PC Login Info

Username: robotis
Password: 111111

### Husarnet Login Info

Username/email: uofchumanoidteam@gmail.com
Password: HumanoidTeam1!

# Some Debug Links & References

Notes - Often if a file/port does not let you execute it or ROS does not recognize a file you must simply change its permissions:

sudo chmod 666 {file or port}

https://devblogs.microsoft.com/commandline/connecting-usb-devices-to-wsl/
https://stackoverflow.com/questions/61860208/running-graphical-linux-desktop-applications-from-wsl-2-error-e233-cannot-op
http://wiki.ros.org/rviz
https://kivy.org/doc/stable/
https://kivymd.readthedocs.io/en/1.1.1/