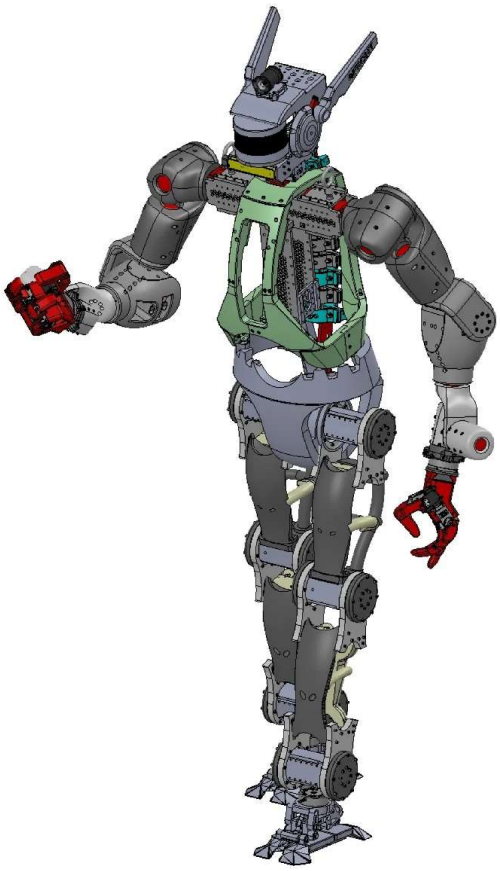
LIFE-SIZED HUMANOID ROBOT

ULTIMATE USER MANUAL



Contents

Glossary

Term	Description
Mesh	Refers to a digital representation of a 3D object or surface. It is made up of a collection of small geometric elements, that are connected to form a continuous and smooth surface.
URDF	Unified Robotics Description Format An XML file describing the relations between joints and their corresponding meshes to visualize the robot.
API	An API (Application Programming Interface) is a set of rules and protocols that allows different software applications to communicate and interact with each other.
GUI	Graphical User Interface The user interface using which the user can easily interact with the system
IMU	Inertial Measurement Unit A sensor that can be used to measure the orientation, angular velocities, and linear accelerations of a system along all 3 axes (X,Y and Z)
Linux	Open-source operating system
Intel-NUC	An Intel NUC (Next Unit of Computing) is a small form factor computer kit developed and manufactured by Intel. It is a compact and powerful mini PC that is designed for a wide range of computing tasks.
ROS	Robot Operating System It is an open-source middleware framework widely used in robotics research and development

Introduction

In the realm of Urban Search and Rescue (USAR) missions, the deployment of robotic systems has emerged as an indispensable need. With their exceptional capabilities to navigate efficiently and quickly through complex and hazardous environments, these advanced machines can play a pivotal role in helping rescue operations.

Recognizing this need, the advanced life-sized 31-DOF humanoid robot has been an ongoing project at the University of Calgary's UVS lab since 2014.

The humanoid robot has been in continuous development by numerous capstone design teams and graduate students over the years and will one day be deployed in humanitarian aid operations, search, and rescue.

Setup

The robot's software uses ROS-Noetic and Ubuntu 20.04 LTS (Linux OS) to communicate with the different sensors and actuators in the physical robot.

How does ROS work?

ROS (Robot Operating System) communicates with different external peripherals using a publish-subscribe messaging system.

In ROS, each functional module is represented as a **node.** For example, a node is responsible for reading data from a camera attached in the robot.

Nodes communicate by publishing and subscribing **"messages"** to **"topics"**. A topic is a named channel through which messages are exchanged. Nodes that generate data publish messages to a specific topic, and nodes interested in that data subscribe to the same topic to receive the messages and can process the message to suit their needs.

Prerequisites for ROS installation:

It is recommended that the user has ubuntu 20.04 LTS installed in their system (or dual booted along with windows). Do not use "Windows Subsystems for Linux" (WSL) as the execution of processes become very slow.

If you haven't installed Ubuntu already, you can follow the steps mentioned at:

https://ubuntu.com/tutorials/install-ubuntu-desktop#1-overview

To install ROS, please follow the tutorial at: http://wiki.ros.org/noetic/Installation/Ubuntu

Once you have successfully installed ros, please run the following commands in your terminal to create a workspace and source it in bash.

Navigate to your root directory in the terminal and type:

```
$ echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
```

To create a new ROS workspace, execute the following commands.

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/
$ catkin_make
```

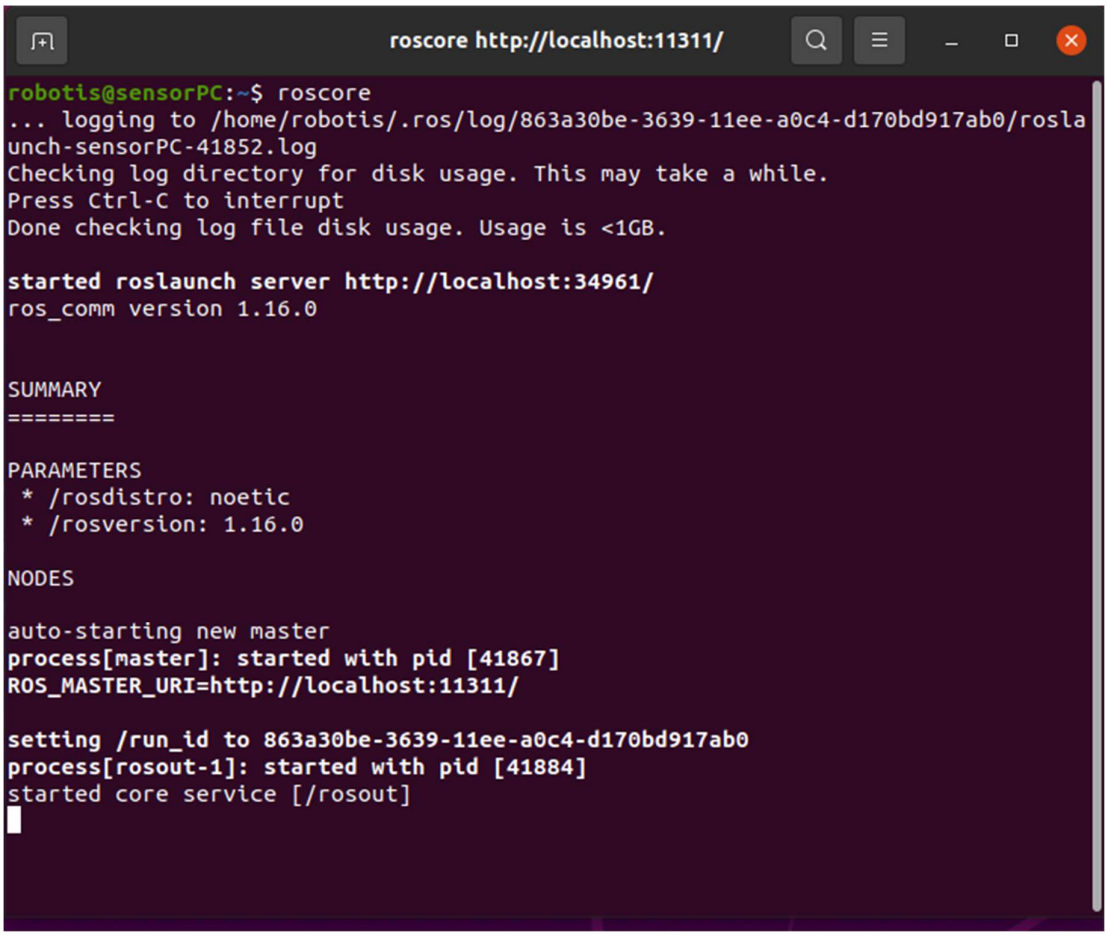
To finally source your ROS workspace, you can execute the following command

```
$ echo "source devel/setup.bash" >> ~/.bashrc
```

When you have completed all the above mentioned steps, run the following command to launch the ROS master

```
$ roscore
```

You should be getting an output as shown in Img. 1



Sensors

The below section provides information about all the sensors present in the robot and guides the user on how to install the required software packages and communicate with the physical sensor.

Velodyne LiDAR VLP-16 Sensor

Info about the sensor :

The VLP-16 sensor uses an array of 16 infra-red (IR) lasers paired with IR detectors to measure distances to objects. The device is mounted securely within a compact, weather-resistant housing. The array of laser/detector pairs spins rapidly within its fixed housing to scan the surrounding environment, firing each laser approximately 18,000 times per second, providing, in real-time, a rich set of 3D point data.

Some important specifications :

Power supply : 12 VDC (1.5 A-rated)

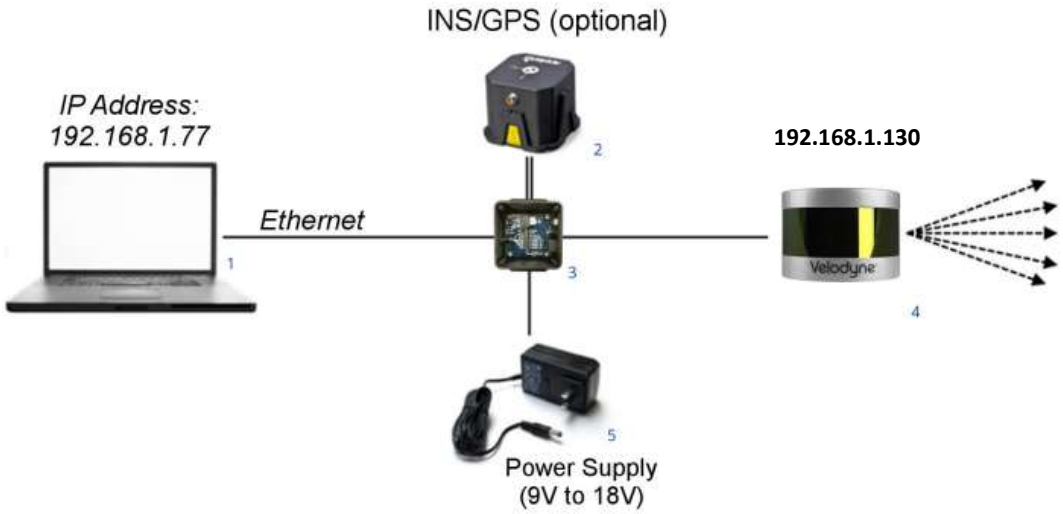
Sensor ID : AG219177050505

IP address : 192.168.1.130

MAC address : 60:76:88:10:45:29

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

Connection Diagrams :

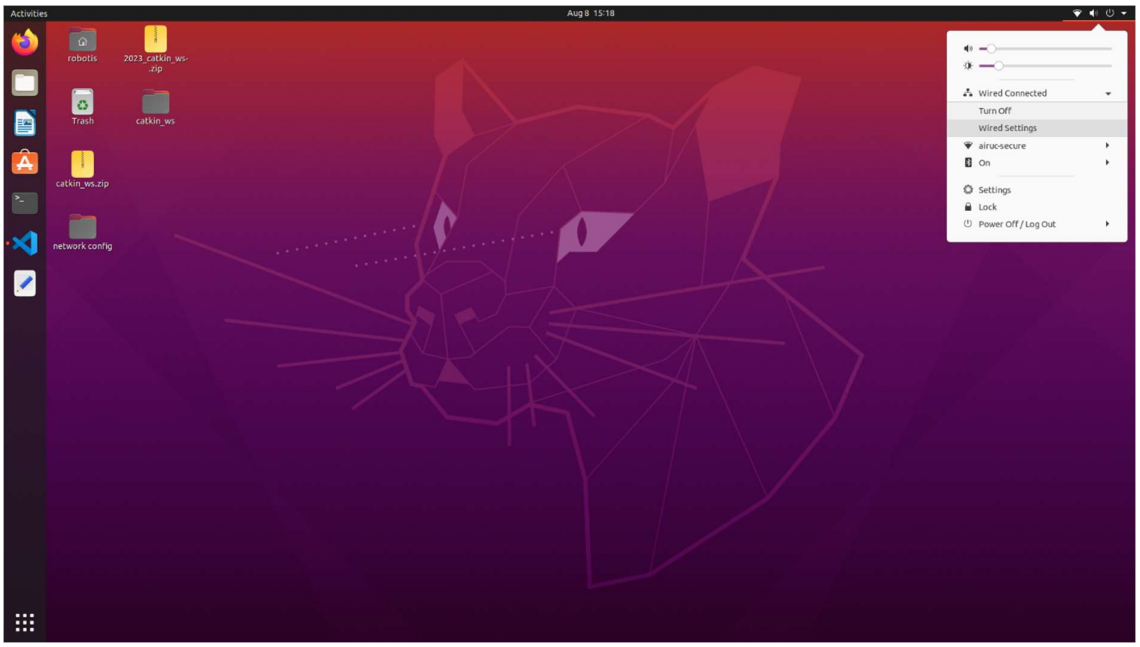


Ethernet Configurations

Connect the ethernet cable from the liDAR interface-board to any of the ehternet ports in the Intel NUC PC or to your laptop.

1. Open the computer's Network Connections page.

2. Open the applicable Ethernet adapter and make sure the interface is enabled.

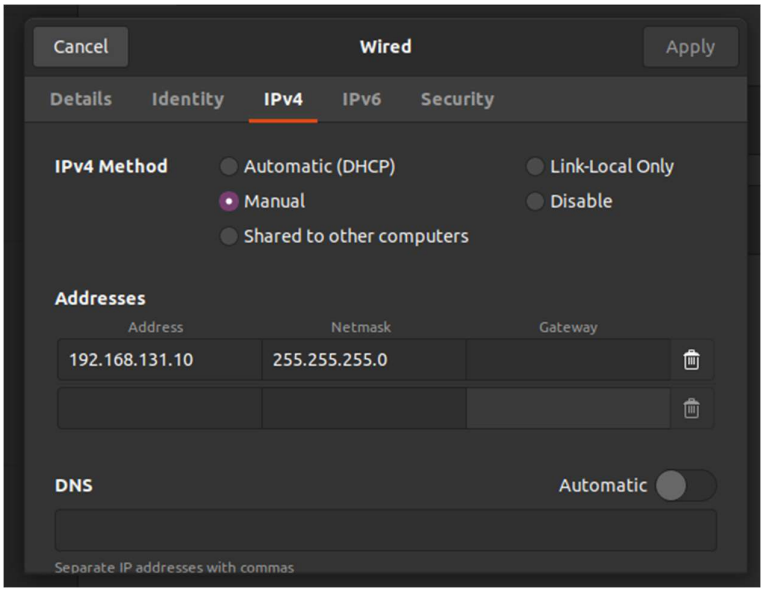


3. Open Internet Protocol Version 4 (IPv4) properties.

4. Select the IPv4 method as Manual

5. Enter the IP address as: 192.168.131.10

6. Enter the subnet mask: 255.255.255.0 and save all the changes.

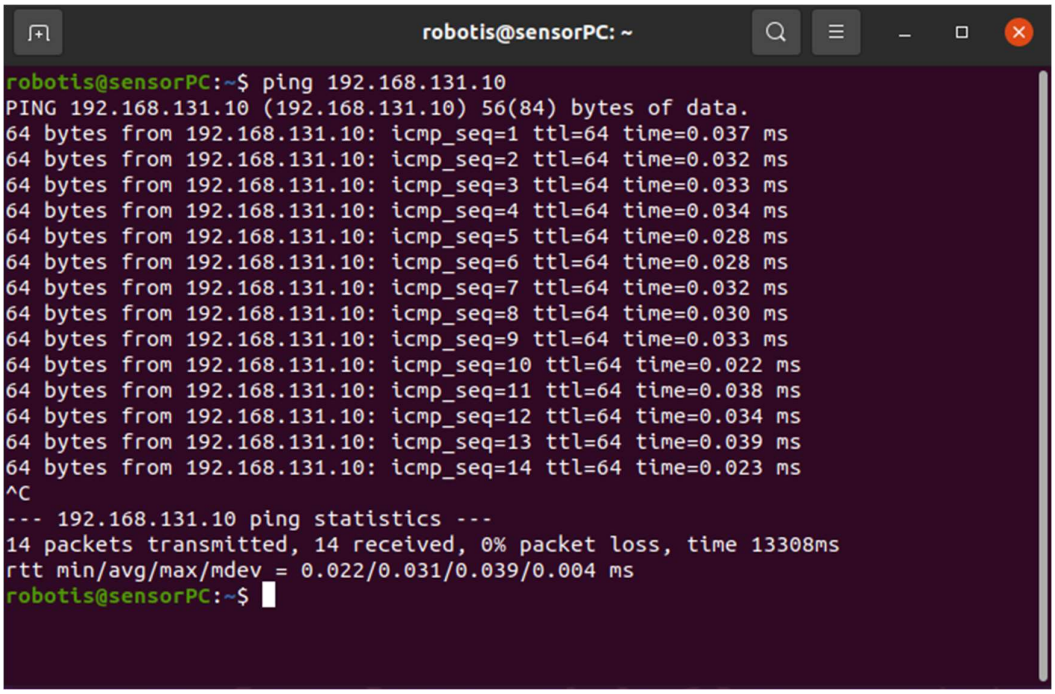


Now you can power the sensor.

To verify if your PC is communicating with the LiDAR, type the following command in the terminal.

```
$ ping 192.168.1.130
```

If you have configured everything correctly, then you should be able to see an output like what is shown in Img 5.



Now, on the computer, point a browser to http://192.168.1.130

The sensor Web Interface should appear

The Web Interface provides access to many of the sensor's control settings.

Visualizing the point cloud data

Using the company's software :

We will be using the "**VeloView**" software to visualize the point cloud data generated by the LiDAR.

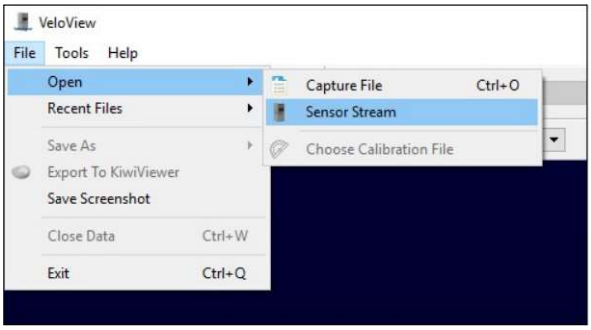
Download the VeloView software from the following link: https://www.paraview.org/veloview/

Once you have downloaded the file, extract the files to your preferred location.

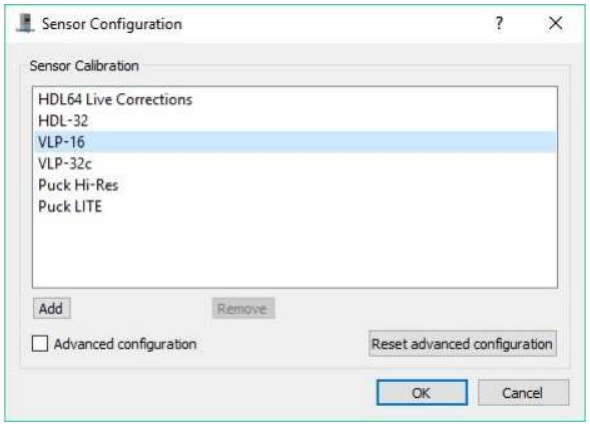
To run the software, navigate to the bin folder inside the VeloView folder and then execute the following command.

```
$ ./VeloView
```

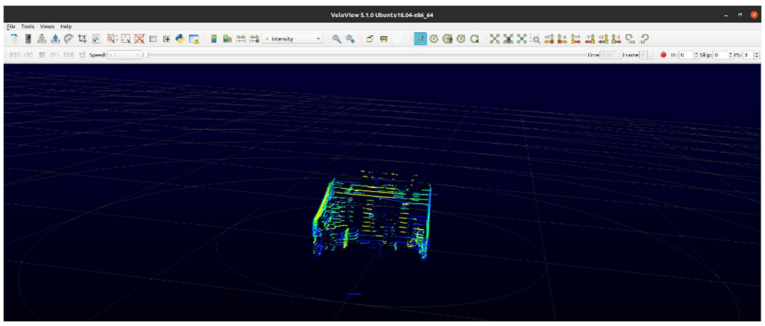
To visualize the data, Click on File->Open and select "Sensor Stream"



The Sensor Configuration dialog will appear. Select the correct sensor type then click OK.



The VeloView software will then begin to display the sensor readings.



Visualization using ROS and Rviz:

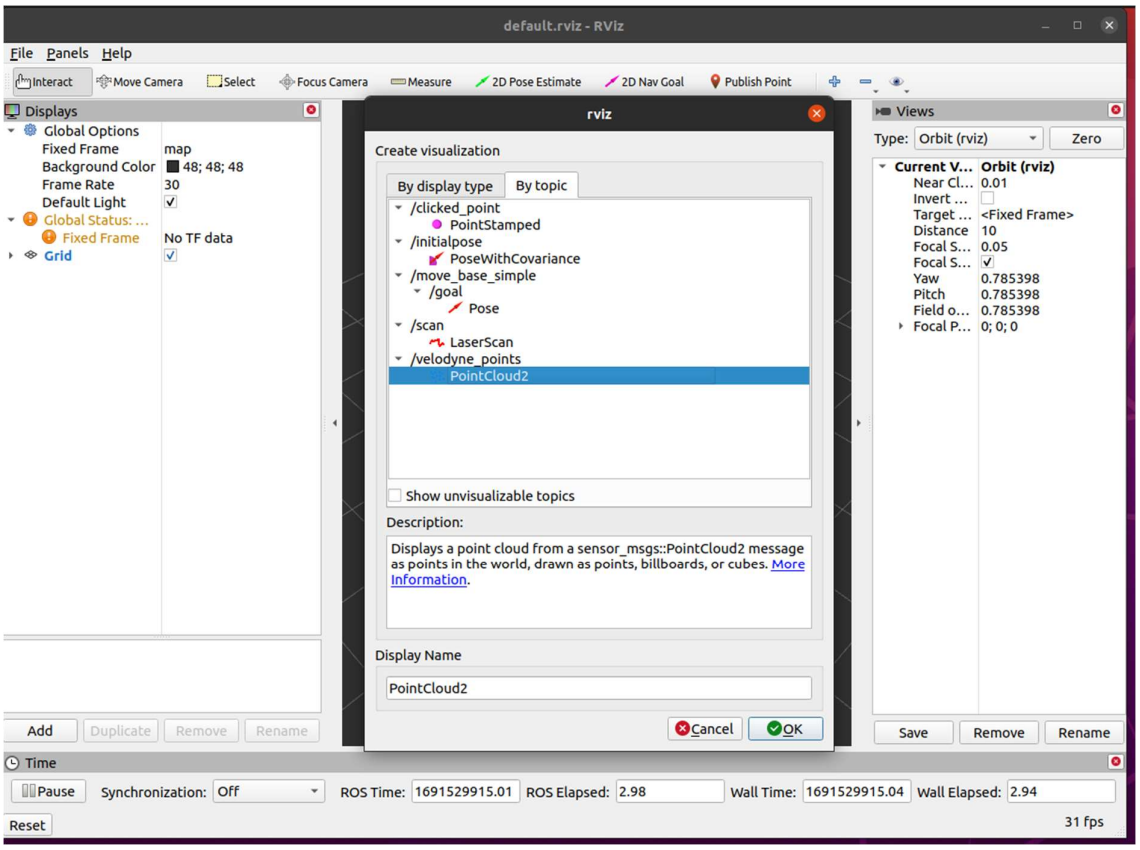
Clone the Velodyne ROS driver repo into the src folder of your catkin workspace and build your workspace by executing the commands below.

```
$ cd catkin_ws/src
$ git clone https://github.com/ros-drivers/velodyne.git
$ cd ..
$ catkin_make
$ cd
$ source ~/.bashrc
```

Power the sensor and run the following command.

```
$ roslaunch velodyne_pointcloud VLP16_points.launch
```

Now open Rviz and add the required rostopic like shown.



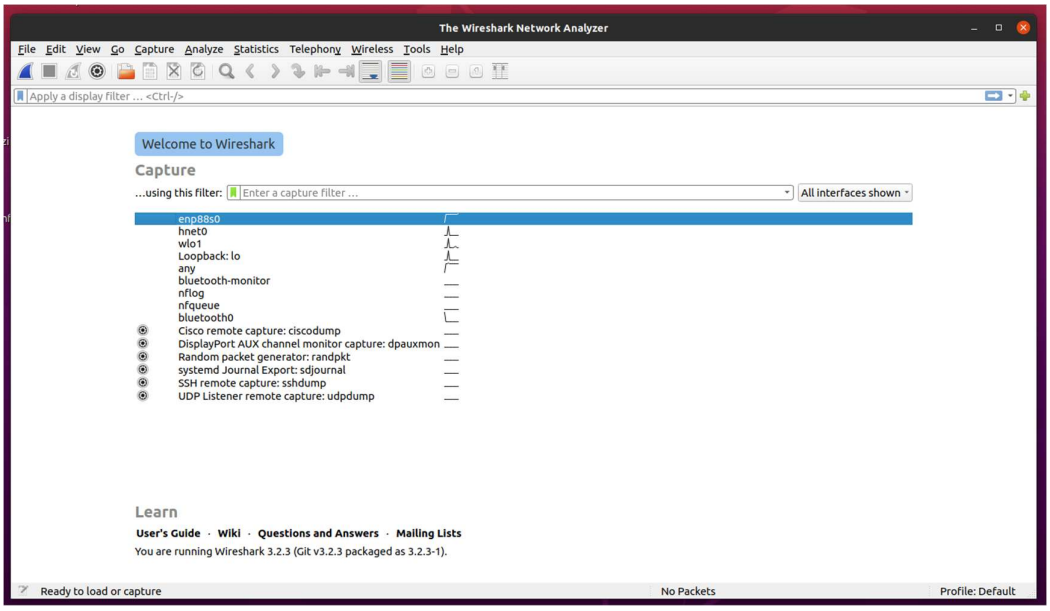
Debugging methods in case of errors

To debug the error in case the user faces any, we must be able to understand the data being communicated by the sensor (if any) through the ethernet cable. For this, we will be using another software called **Wireshark**.

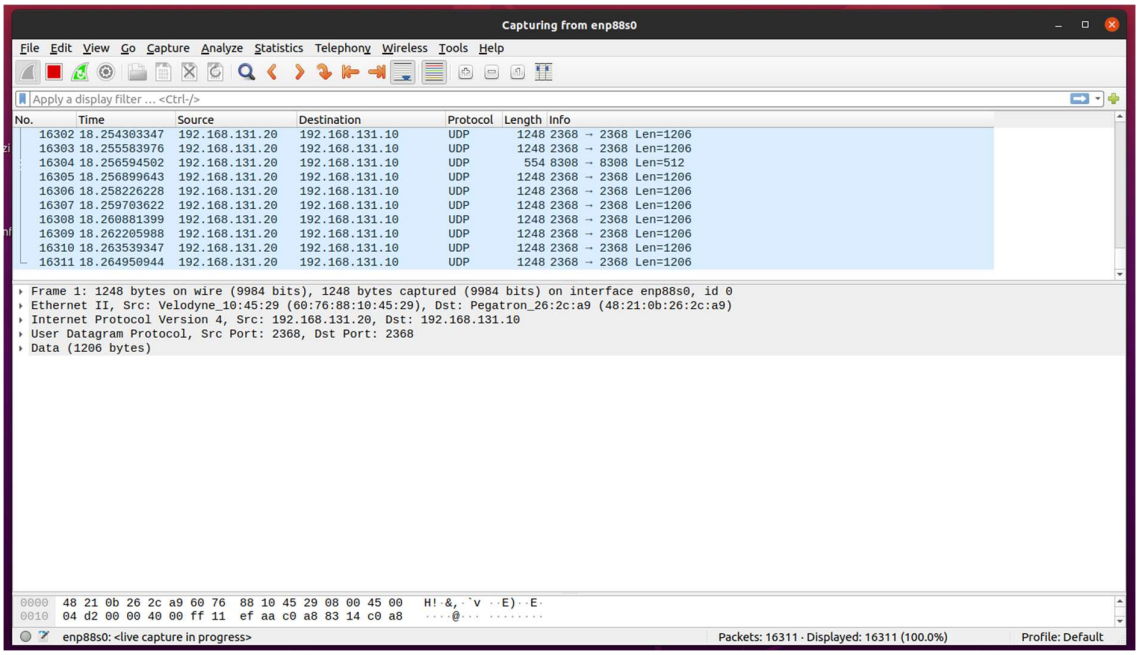
To install the wireshark software, follow the steps mentioned in the below link :

https://linuxhint.com/install_wireshark_ubuntu/

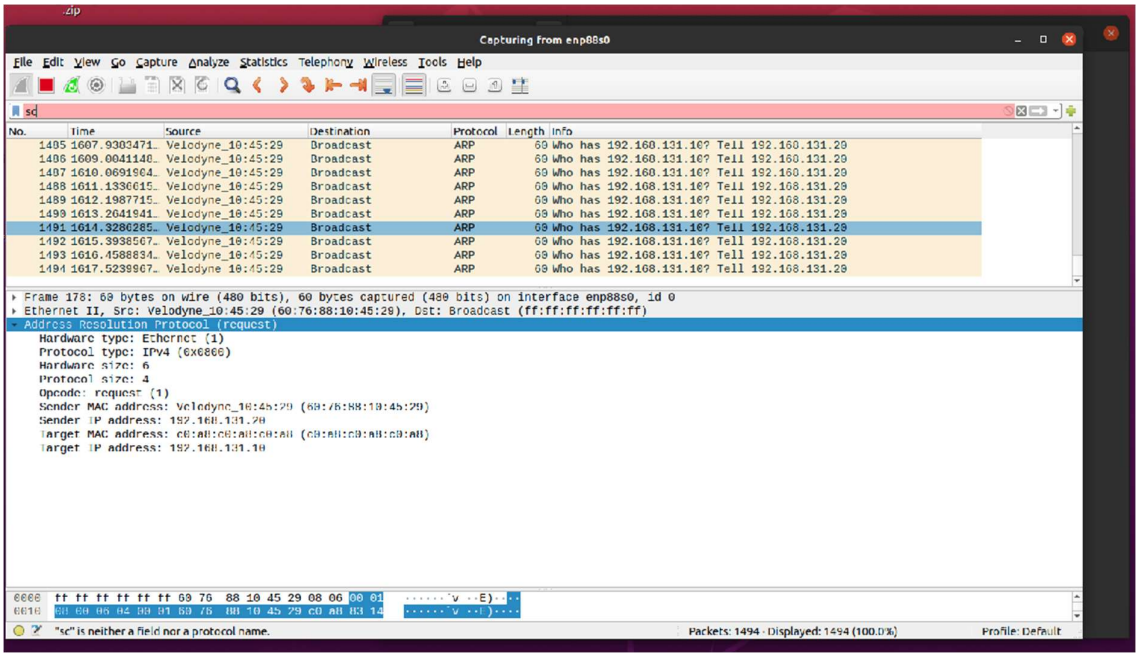
Open the software and choose the ethernet communication channel. (Img 10)



If you have configured everything correctly, you should be able to see a continuous stream of UDP data packets (Img 11).



In case you see something like Img 12 :



This means that the IP address of the LiDAR is not specified to the PC correctly. Please change the IP address in the IPv4 tab in the PC's network setting to the highlighted IP address being displayed in the Wireshark software.

If you are not able to see any data streams in the window, it means that the sensor is not connected properly with the interface board or that the ethernet cable is either faulty or not connected properly from the interface board and the PC.

Bota Systems Force-Torque sensor

Info about the sensor :

The LaxONE force-torque sensor manufactured by Bota Systems, is an industrial lightweight sensor that measures the force and Torque on the X,Y and Z axes.

There are 4 such sensors in the robot, each attached in the left, right wrists and in the left, right ankles of the humanoid robot.

Some important specifications :

Sensor Model **:** LaxONE Serial (all 4 sensors)

Power Supply : 5V, 1W

Range (Fxy, Fz, Mxy, Mz) : 1800 N, 4000 N, 60 Nm, 60 Nm

Communication : USB

IMU : Not present

User Manual **:** https://app-eu1.hubspot.com/documents/24886714/view/533163567?accessId=f36bb6

Serial Number :

Left Wrist	Right Wrist	Left Ankle	Right Ankle
00142	00143	00144	00145

Connections

Connect the sensor using the M8-USB cable from the M8 pin in the sensor to any of the USB ports in the PC. **Please be very careful while connecting the cable to the sensor as the pins in the sensors are very fragile!**

It is highly recommended **not** to disconnect the cable from the sensor once connected unless it is extremely required!

Interfacing the sensor with ROS

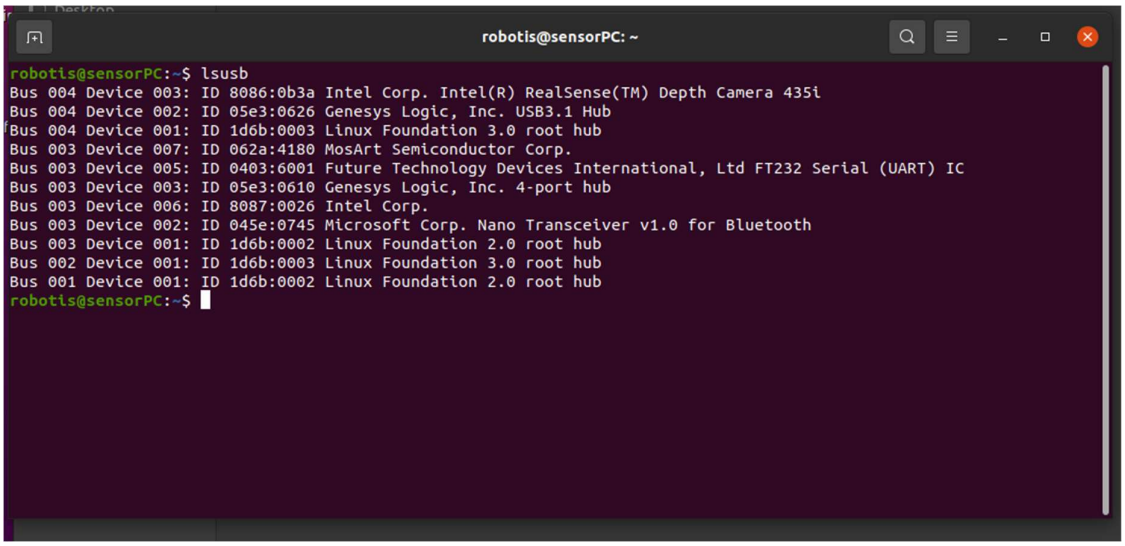
Execute the following commands in **your system** to install and build the ROS driver necessary to communicate with the sensor.

```
$ cd catkin_ws/src
$ git clone https://gitlab.com/botasys/bota_driver.git
$ git clone https://gitlab.com/botasys/bota_demo.git
$ cd ..
$ catkin_make
$ cd
$ source ~/.bashrc
```

Find out which port number the sensor is connected to in your PC by executing the below command.

```
$ lsusb
```

You should be able to see the output like shown (Img 13).



Now, specify the corresponding port number in the BFT_LAX_SER_M8.launch file (Img 14).

Once you have done everything as instructed above, run the following command to communicate with the sensor.

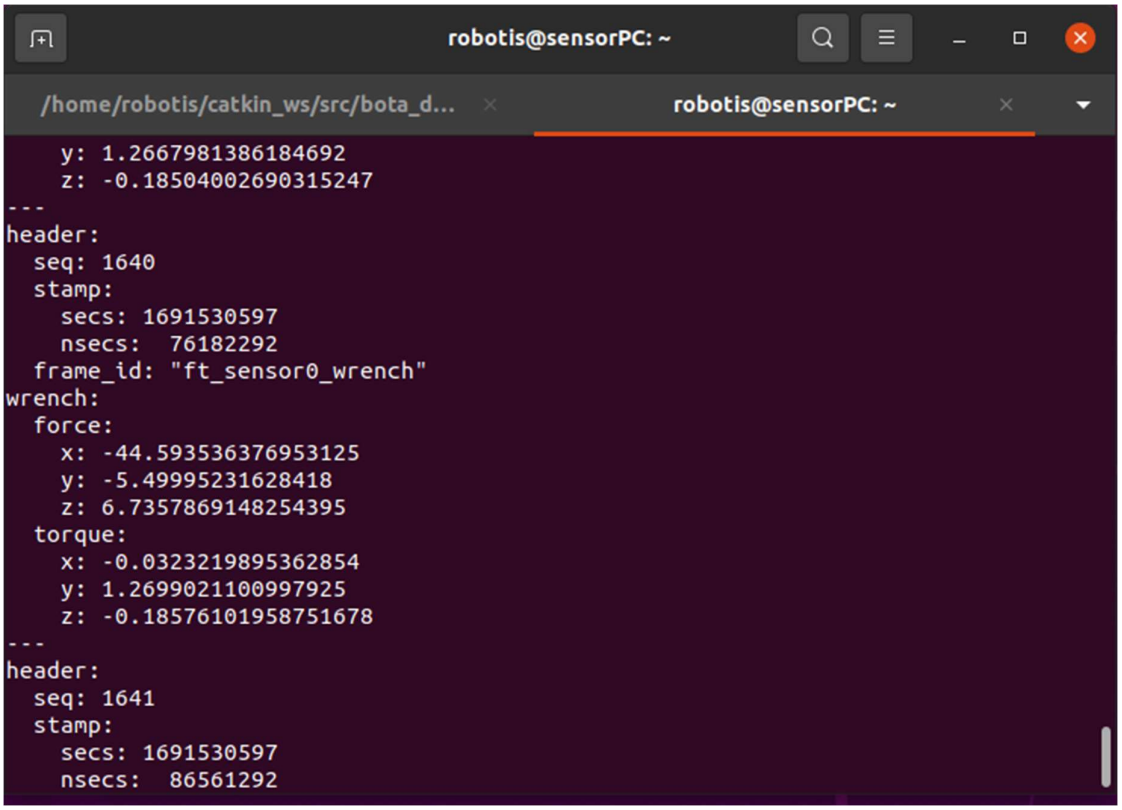
```
$ roslaunch bota_demo BFT_LAXS_SER_M8.launch
```

You can observe that an Rviz window opens along with the spawned sensor’s model and the force-torque vector’s real-time visualization.

If you want to see the numerical values of the components of force and torque values read by the sensor, open a new terminal window and run the following command.

```
$ rostopic echo /bus0/ft_sensor0/readings/wrench
```

The following image is a sample output that you may see after running the above command (Img 16).



Interfacing all 4 sensors using the Intel NUC PC and a USB HUB

In the physical robot, the 4 FT sensors are connected to the Intel NUC sensor PC using a USB hub. All the ROS drivers are already installed and configured.

Sensors and their corresponding virtual port numbers

FT sensor	Port
Left Wrist	/dev/ttyUSB0
Right Wrist	/dev/tty/USB1
Left Ankle	/dev/tty/USB2
Right Ankle	/dev/tty/USB3

To interface the left wrist sensor, execute :

```
$ roslaunch bota_demo BFT_LAXS_SER_M8_l_wr.launch
```

To interface the right wrist sensor, execute :

```
$ roslaunch bota_demo BFT_LAXS_SER_M8_r_wr.launch
```

To interface the left wrist sensor, execute :

```
$ roslaunch bota_demo BFT_LAXS_SER_M8_l_an.launch
```

To interface the left wrist sensor, execute :

```
$ roslaunch bota_demo BFT_LAXS_SER_M8_r_an.launch
```

To interface the all the 4 sensors together, execute :

```
$ roslaunch bota_demo all_4_sensors.launch
```

Sensor and the corresponding rostopic to which the sensor publishes its reading

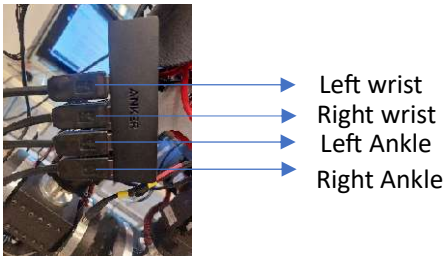
Sensor	Rostopic
Left wrist	/bus0/ft_sensor0/readings/wrench
Right wrist	/bus1/ft_sensor1/readings/wrench
Left ankle	/bus2/ft_sensor2/readings/wrench
Right angle	/bus3/ft_sensor3/readings/wrench

You can always run the following command to access the numerical values of the measurements of the sensors.

```
$ rostopic echo /{topic corresponding to sensor}
```

Note : *The port numbers of the FT sensors might have been changed due to various reasons. Therefore please verify the port numbers of the sensors and update it in sensors' corresponding launch files like shown in Img 15 to establish proper communication.*

Image of the physical connection



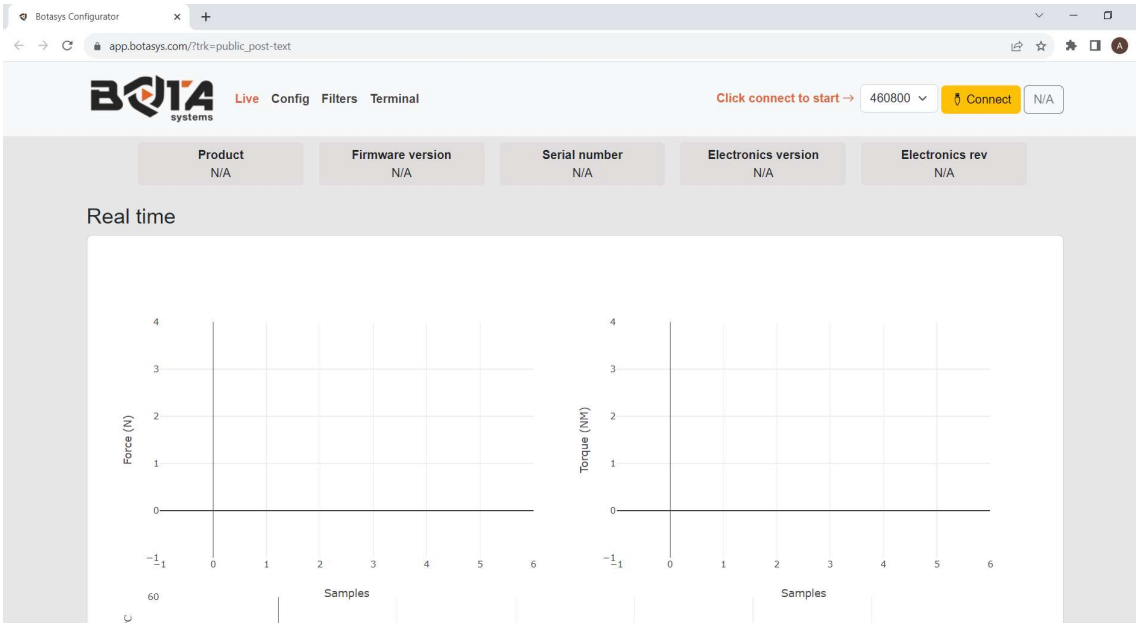
Debugging methods in case of errors

Please check whether the sensors are connected to the PC properly.

Make sure the right port number is specified in the launch files.

To check whether the sensor is able to communicate to the PC, you can use the Chrome web browser created by the company.

Point your browser to https://app.botasys.com



You can connect to the sensor by selecting the appropriate baud rate (default : 460800 bps) and plot the stream of real-time data into the GUI.

Navigate to the config tab to change the configurations and other parameters if necessary.

Intel RealSense T265 Camera

Info about the device :

The Intel RealSense T265 Camera is a sensor module designed for tracking and navigation applications. The T265 Camera is unique in that it's equipped with an inertial measurement unit (IMU) along with two fisheye cameras. This combination enables the camera to perform localization without relying on external sensors or markers.

Some important specifications :

Camera Module : Intel Realsense T265

Camera : 2 Fisheye cameras

Field of view (FOV) : 160-degree horizontal field of view (FOV) per camera

110-degree vertical field of view (FOV) per camera

IMU : 6-axis

Interfaces : USB 3.1 Gen 1 Type-C interface for data and power

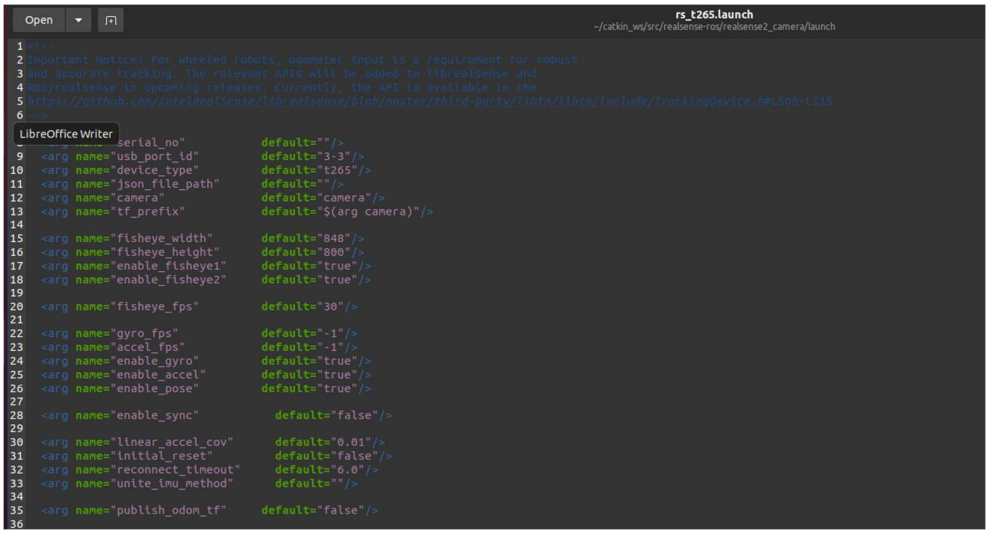
User Manual : https://dev.intelrealsense.com/docs/tracking-camera-t265

Interfacing Camera with ROS

Execute the following commands to build and install the required ROS-drivers for the Intel Real-sense camera

```
$ cd catkin_ws/src
$ git clone https://github.com/IntelRealSense/realsense-ros.git
$ cd ..
$ catkin_make
$ cd
$ source ~/.bashrc
```

Update the correct port number to which the camera is connected in the “rs_t265.launch file as shown (Img 18). Enable the “**enable_fisheye1**” and “**enable_fisheye2**” and set the values for the “**fisheye_width**” and the “**fisheye_hieght**” accordingly as shown.

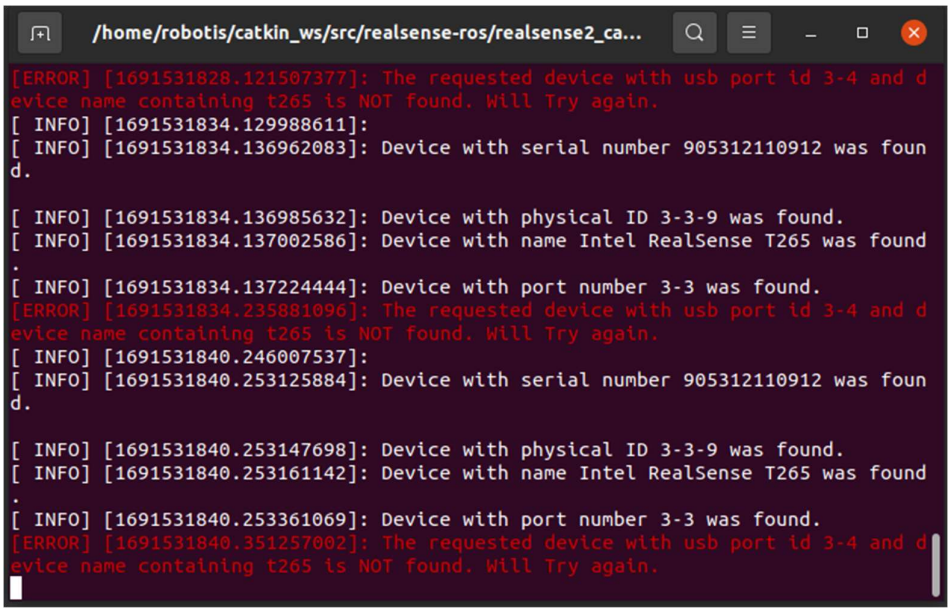


To communicate with the camera, execute the following command.

```
$ roslaunch realsense2_camera demo_t265.launch
```

In case you cannot find the port number to which the camera is connected or have specified it incorrectly, run the above command, nevertheless.

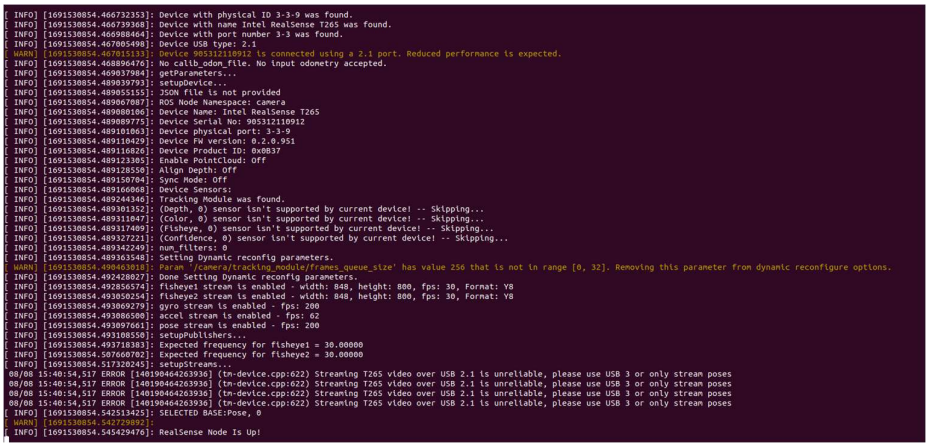
You will be able to view the following errors as shown in the output (Img 19).



The correct port number in which the device is recognized will be displayed in the screen.(See highlighted portion in Img 19).

Please update the port number accordingly and run the same command again.

If everything has been set up right, we will be able to see the output as shown in Img 20



Open Rviz and add the required topic as shown to view the image from the fisheye lenses and the frames published by the camera to view the camera image stream.

One of the main uses of the tracking camera is establishing **visual odometry data**. This data can be viewed by executing

```
$ rostopic echo /camera/odom
```

Info about the sensor :

The VN-200 Vectornav is a miniature, high performance GNSS-Aided Inertial Navigation System (GNSS/INS) that combines 3-axis gyros, accelerometers and magnetometers, a high-sensitivity GNSS receiver, and advanced Kalman filtering algorithms to provide optimal estimates of position, velocity, and attitude.

Some important specifications :

Sensor model : VN-200

Power supply : 3.2 – 5.5 V

Maximum data output rate : 800 Hz

Roll-pitch-yaw accuracy : 0.03 degrees

User Manual :

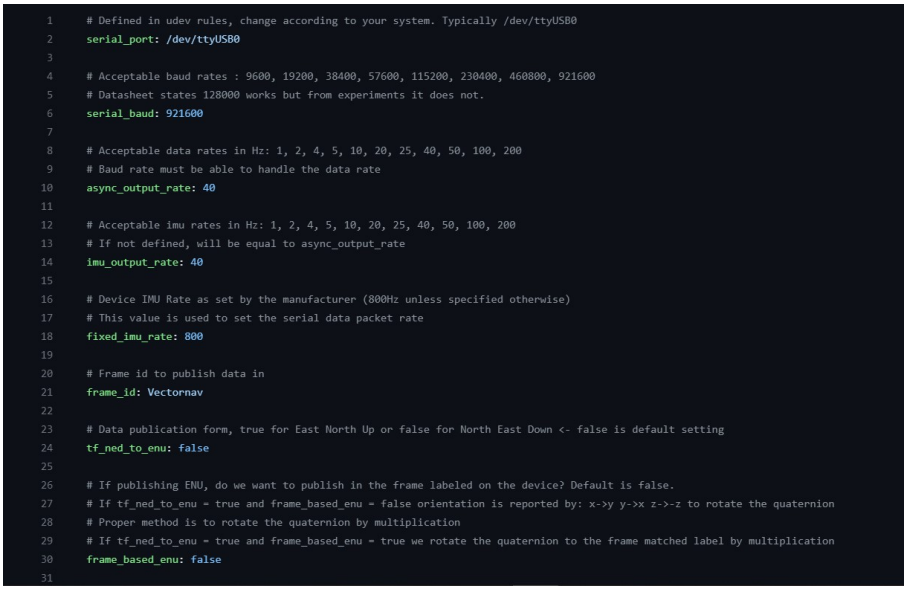
Interfacing sensor with ROS

Execute the following commands to build and install the required ROS-drivers for the Vectornav IMU.

```
$ cd catkin_ws/src
$ git clone https://github.com/dawonn/vectornav.git
$ cd ..
$ catkin_make
$ cd
$ source ~/.bashrc
```

Specify the port number to which the IMU is connected in the params/vn200.yaml file like shown (Img 22) – see highlighted red box

You can also specify the output data rate by setting the value of the “imu_output_rate” in the same file. (Img 22) - see highlighted green box



```
# Defined in udev rules, change according to your system. Typically /dev/ttyUSB0
serial_port: /dev/ttyUSB0

# Acceptable baud rates : 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600
# Datasheet states 128000 works but from experiments it does not.
serial_baud: 921600

# Acceptable data rates in Hz: 1, 2, 4, 5, 10, 20, 25, 40, 50, 100, 200
# Baud rate must be able to handle the data rate
async_output_rate: 40

# Acceptable imu rates in Hz: 1, 2, 4, 5, 10, 20, 25, 40, 50, 100, 200
# If not defined, will be equal to async_output_rate
imu_output_rate: 40

# Device IMU Rate as set by the manufacturer (800Hz unless specified otherwise)
# This value is used to set the serial data packet rate
fixed_imu_rate: 800

# Frame id to publish data in
frame_id: Vectornav

# Data publication form, true for East North Up or false for North East Down <- false is default setting
tf_ned_to_enu: false

# If publishing ENU, do we want to publish in the frame labeled on the device? Default is false.
# If tf_ned_to_enu = true and frame_based_enu = false orientation is reported by: x->y y->x z->-z to rotate the quaternion
# Proper method is to rotate the quaternion by multiplication
# If tf_ned_to_enu = true and frame_based_enu = true we rotate the quaternion to the frame matched label by multiplication
frame_based_enu: false

```

Once you have done this, you can run the below command to facilitate communication between the PC and the sensor.

```
$ roslaunch vectornav vectornav.launch
```

To view the IMU readings, execute :

```
$ rostopic echo /vectornav/IMU
```

Interfacing GPS with IMU

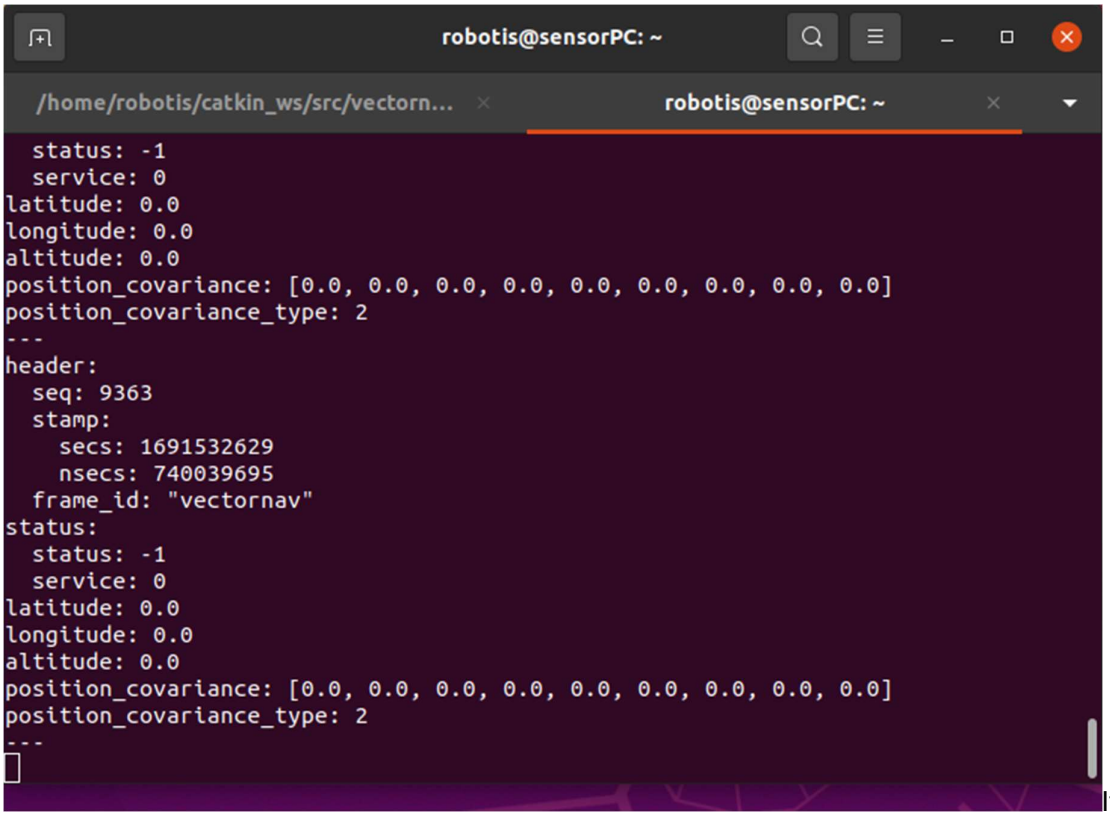
The vectornav IMU also comes with a separate 1 x 72 Channel GNSS Receiver module.

Make sure the GPS module is connected with the IMU.

To view the GPS readings, run:

```
$ rostopic echo /vectornav/GPS
```

Check the **status** parameter in the GPS message published by the sensor (as shown in Img 23.

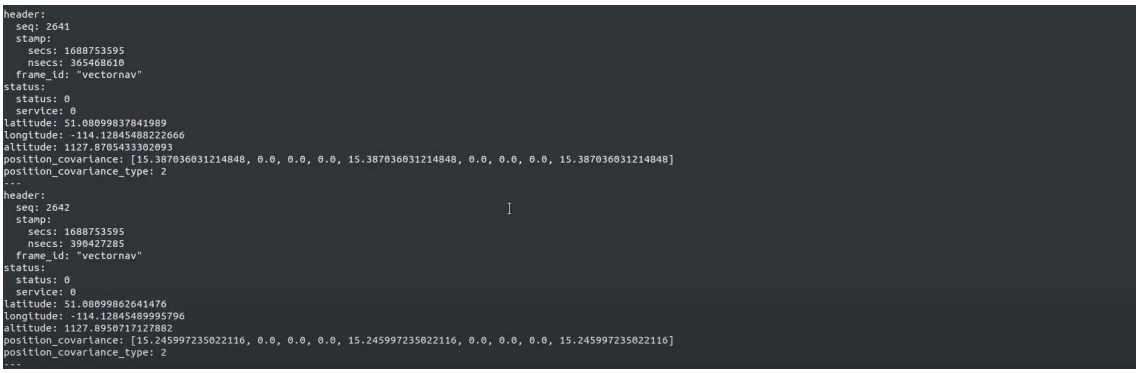


If **status = -1** : Satellite fix has not yet been achieved. Lat, Long data is invalid

Please move to a place where the GPS module has an open sky above it and wait for the ins-status to become 1

If **status = 0** : Satellite fix has been achieved. Lat, Lond data is valid.

The example output message of the GPS system **with satellite fix** has been shown. (Img 24)



You can view the latitude, longitude, and altitude readings from the GPS message.

Note : *Please wait for the status to turn 1.* ***It may take up to 10 minutes*** *as the signal strength depends on various factors such as the presence of satellites above you and cloud cover.*

Graphical User Interface

The graphical user interface was designed by the 2022-23 Capstone design team and enhanced by the 2023-24 team to visualize the robot, run demos, control motors and provide an easy platform to interface the sensors present in the robot. It leverages **Kivy**, an open-source python UI framework, **KivyMD** (a framework complaint with Kivy used to add widgets) , Python 3.8 and ROS.

Installation procedure for Kivy and KivyMD

```
$ pip install Kivy
$ pip install KivyMD
```

Documentation

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

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

Prerequisites

- ROS-Noetic installed with all GUI dependencies
    - Kivy
    - KivyMD
- Motors: connected to the PC and powered
- LiDAR and Intelrealsense cameras: connected and powered
- F/T sensors: connected to the PC via the USB hub
- Vectornav IMU : connected and powered

Launching the GUI

Execute the following commands in your terminal.

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

Open a new terminal window and run :

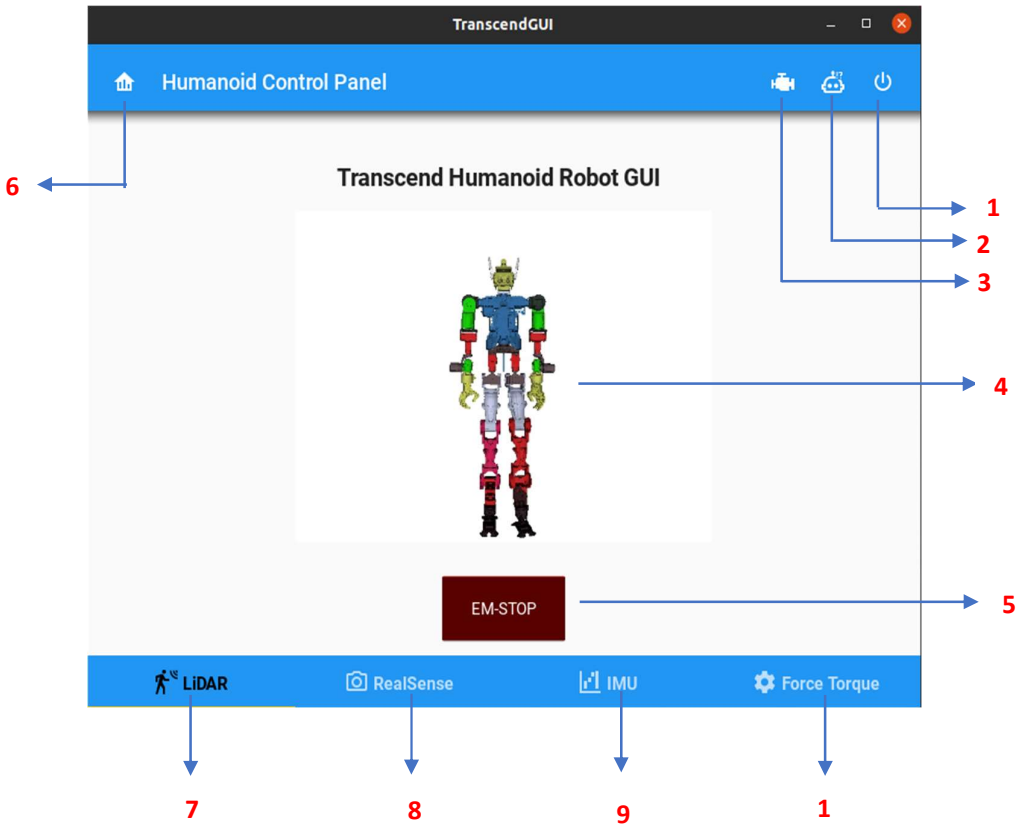
```
$ rosrun urdf-rviz humanoidGui.py
```

Using the GUI

Once you run the GUI python script, a window (like shown below) will be displayed.

The below image describes the function/necessity of each of the components in the GUI screen.

GUI Home Screen Panel



Components and their functions (Home Screen Panel)

1 : Power button – Press the power button to enable all the functionalities of the GUI

2 : Motor ID Panel – Press this button to view the IDs of all the motors in the robot

3 : Motor Control button – Press this button to navigate to the motor control panel

4 : Real time URDF image of robot

5 : Emergency stop button – Activating this button immediately stops all processes and quits the GUI

6 : Home button – Press the home button to reach back to the home screen

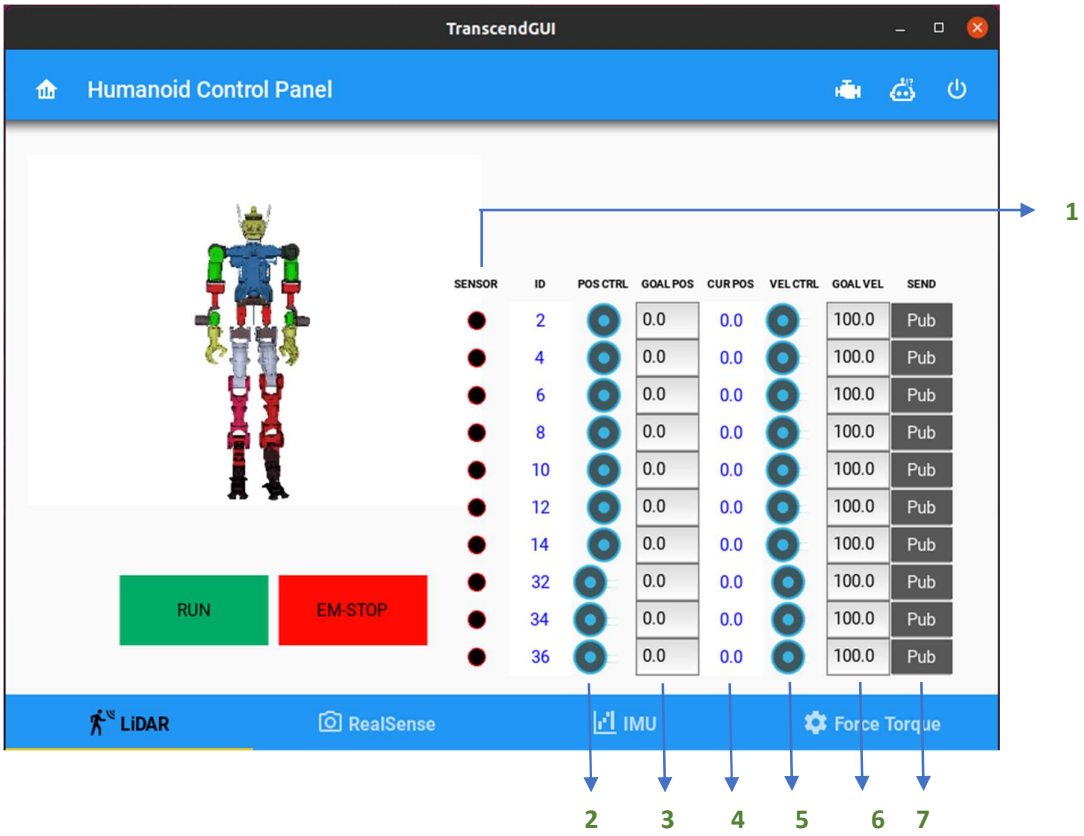
7 : Lidar Panel – Press this button to navigate to the Lidar panel

8 : RealSense Panel - Press this button to navigate to the Real sense camera panel

9 : IMU Panel - Press this button to navigate to the IMU Panel

10 : F/T sensor Panel - Press this button to navigate to the F/T sensor Panel

GUI Motor Control Panel (Left Hand)



Components and their functions (Motor Control Panel – Left hand)

1 : Motor status – The checkbox glows red if the motor is connected and its torque, enabled

2 : Slider for adjusting the goal Position (Position control)

3 : Text box, to input the required goal position directly (Position control)

4 : Displays current position of the motor

5 : Slider to adjust goal velocity (for velocity control)

6 : Text box, to input goal velocity directly (velocity control)

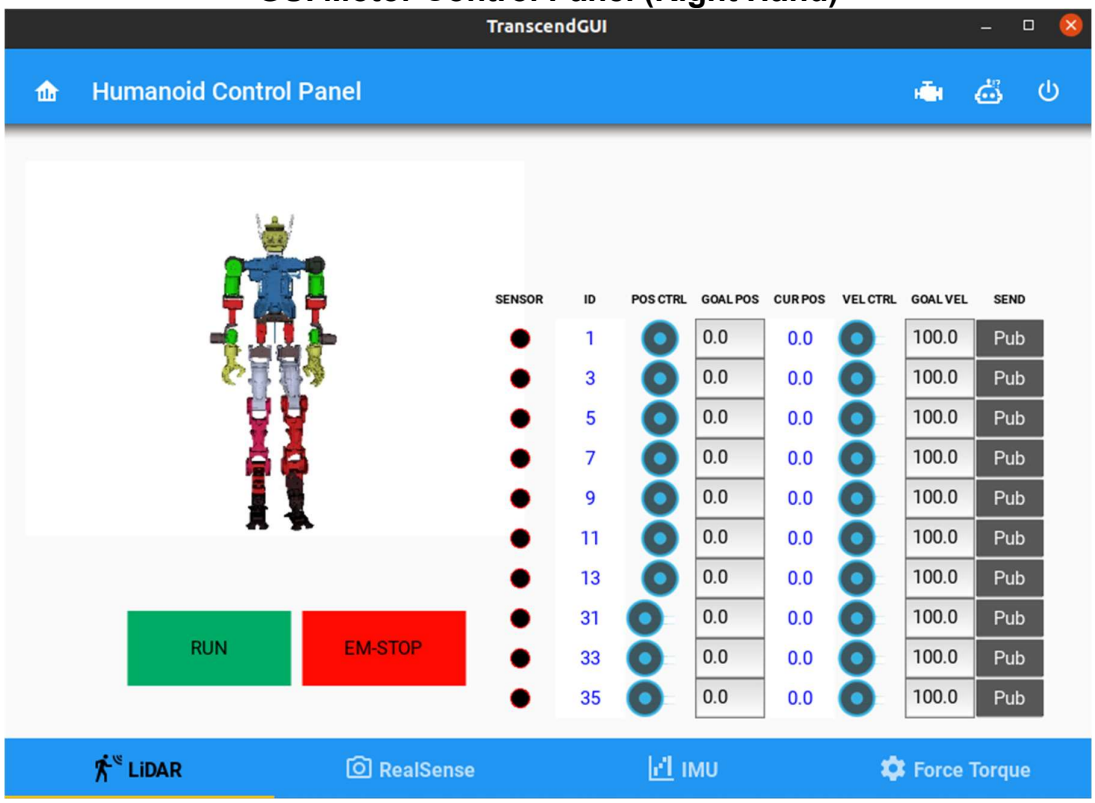
7 : Publish Button – Press this button to publish the changes in the physical system

Steps to control any motor using the GUI

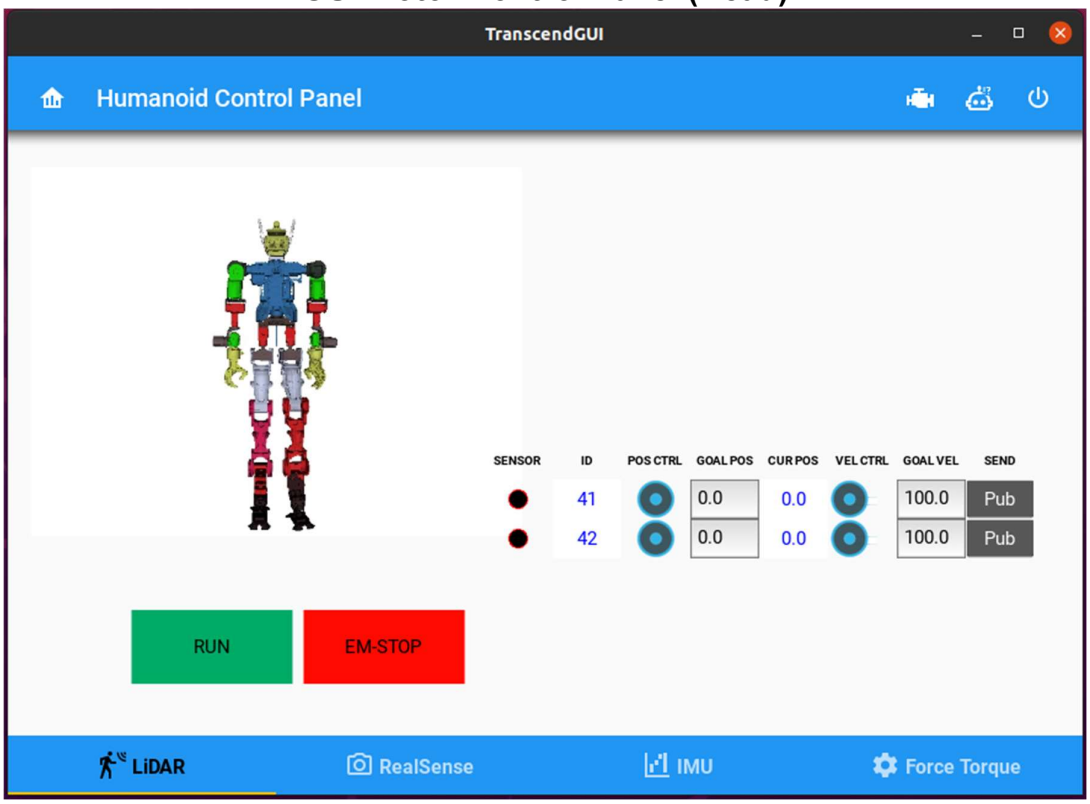
1) For position/velocity control of a motor :
2) Slide the corresponding slider or type the goal value for the motor to be controlled.
3) **Wait for about 2 seconds for the URDF file to reflect the changes**
4) **Observe the URDF file and make sure that your command will not lead to any collisions.**
5) After carefully verifying the new pose of the robot, press the 

button to control the motors in the physical system

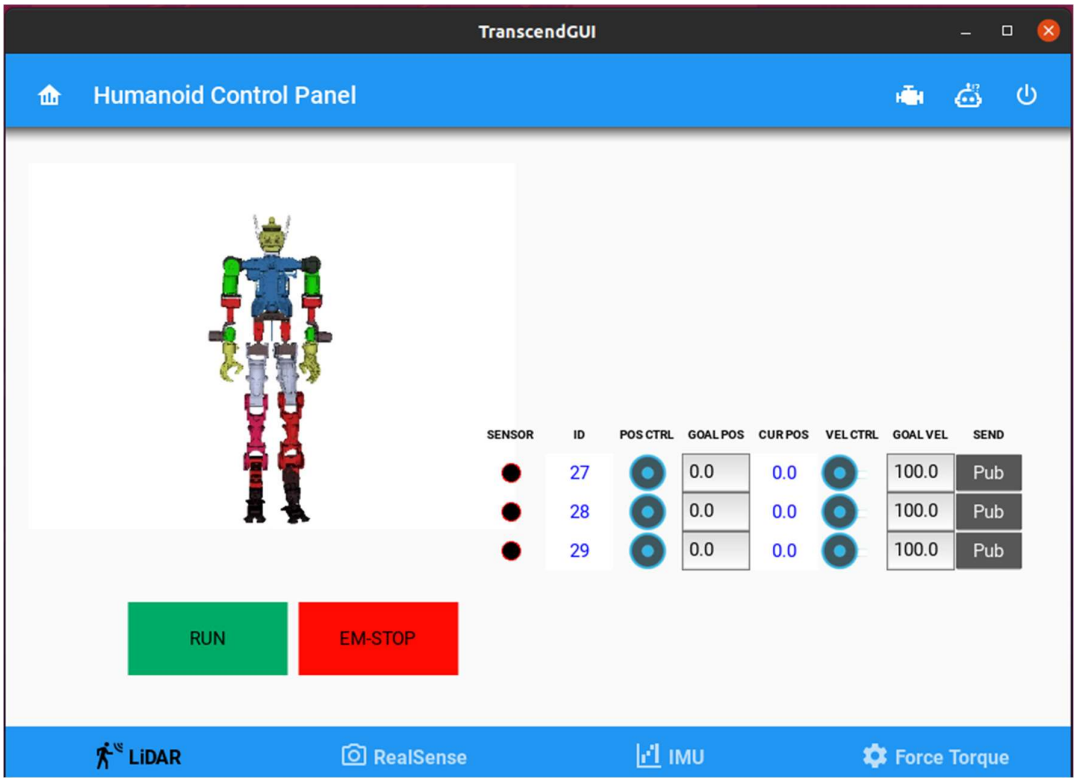
GUI Motor Control Panel (Right Hand)



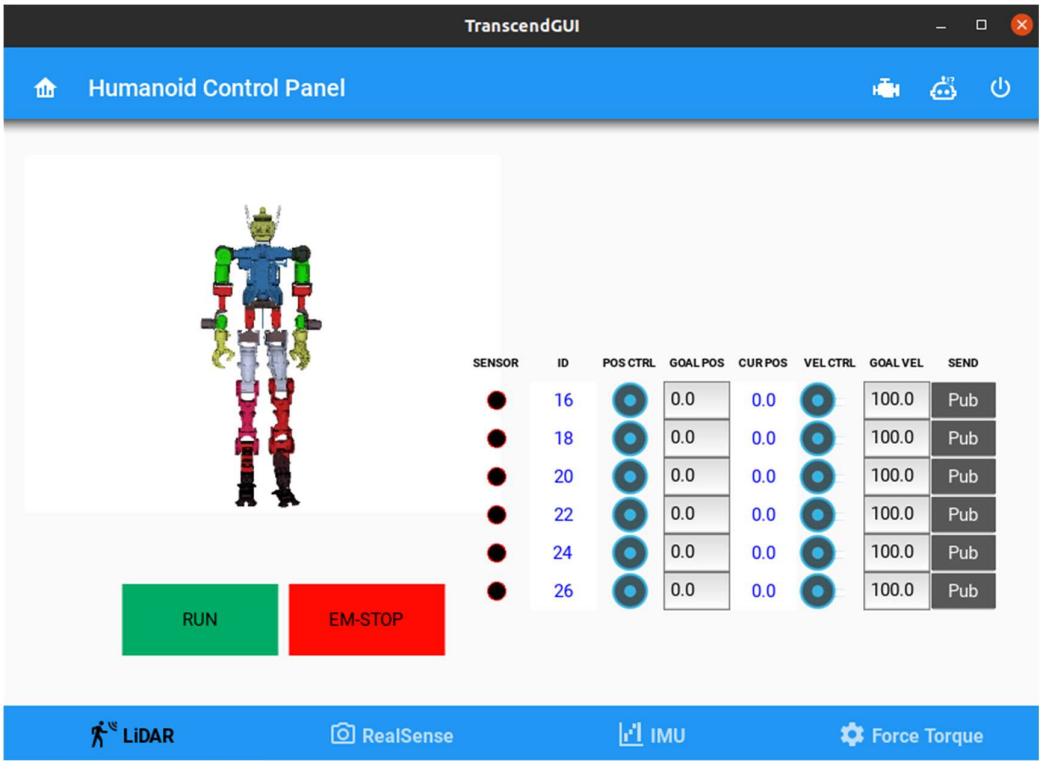
GUI Motor Control Panel (Head)



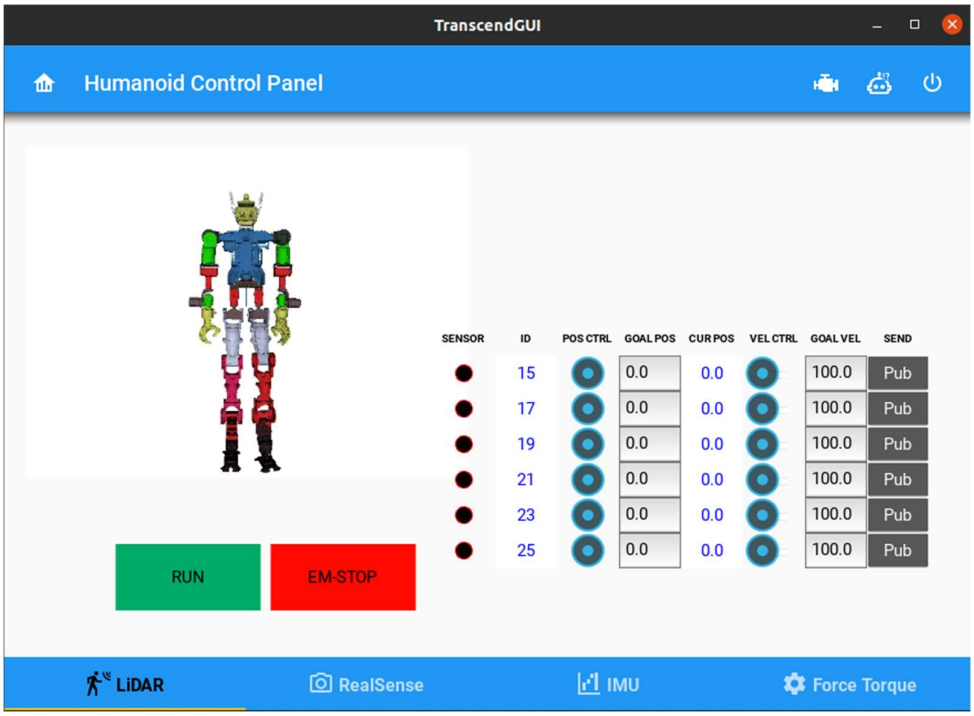
GUI Motor Control Panel (Torso)



GUI Motor Control Panel (Left Leg)

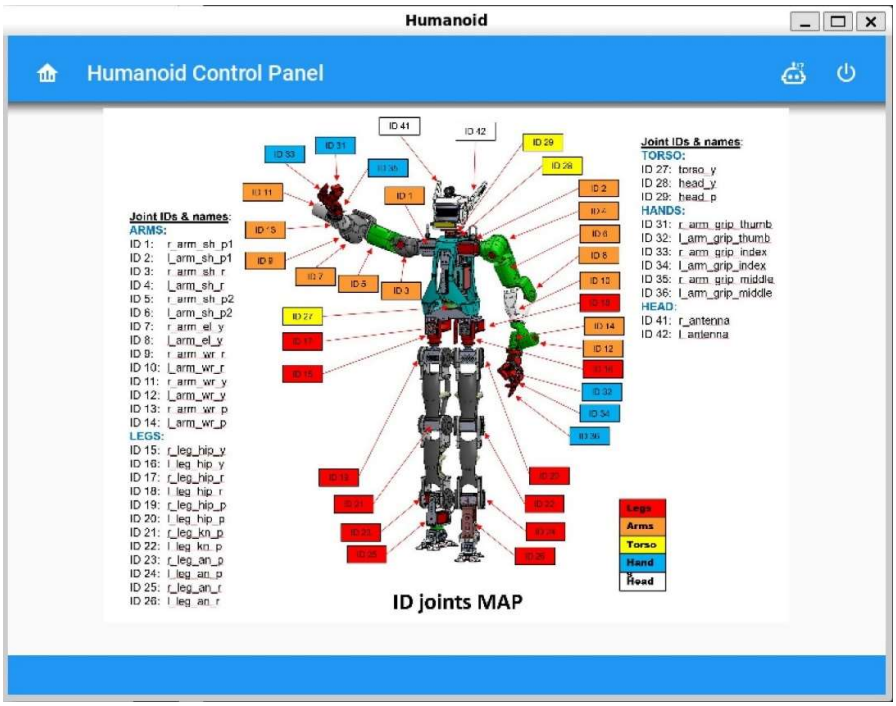


GUI Motor Control Panel (Right Leg)



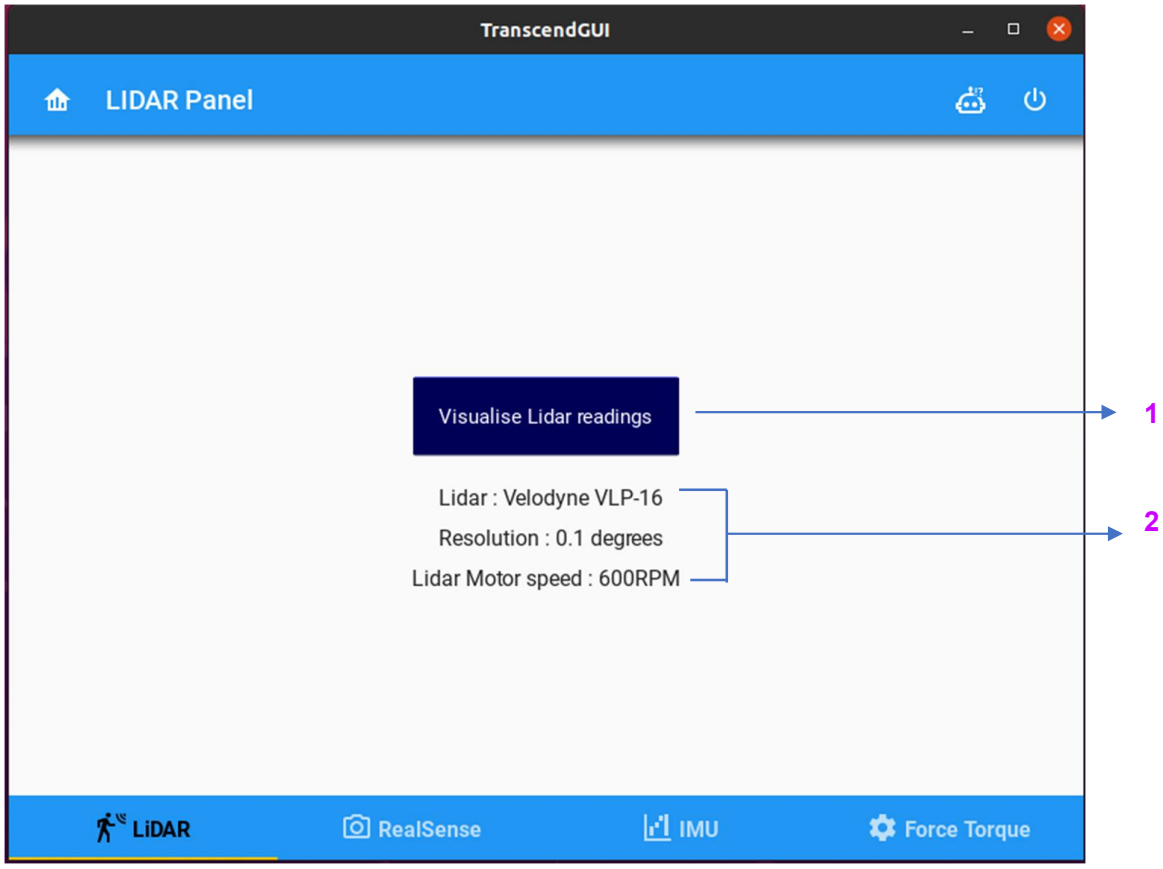
The functionalities are similar to what has been described for the left-hand motor control panel. Please refer to its description and follow the same thing for the other motor control panels as well.

GUI ID Panel



The ID panel specifies the IDs of all the dynamixel motors and their corresponding joint names.

GUI Lidar Panel



Components and their functions (Lidar Panel)

Make sure the Lidar is fully setup and is communicating properly with the PC

1 : Press the “Visualise Lidar readings” button to view the point cloud data.

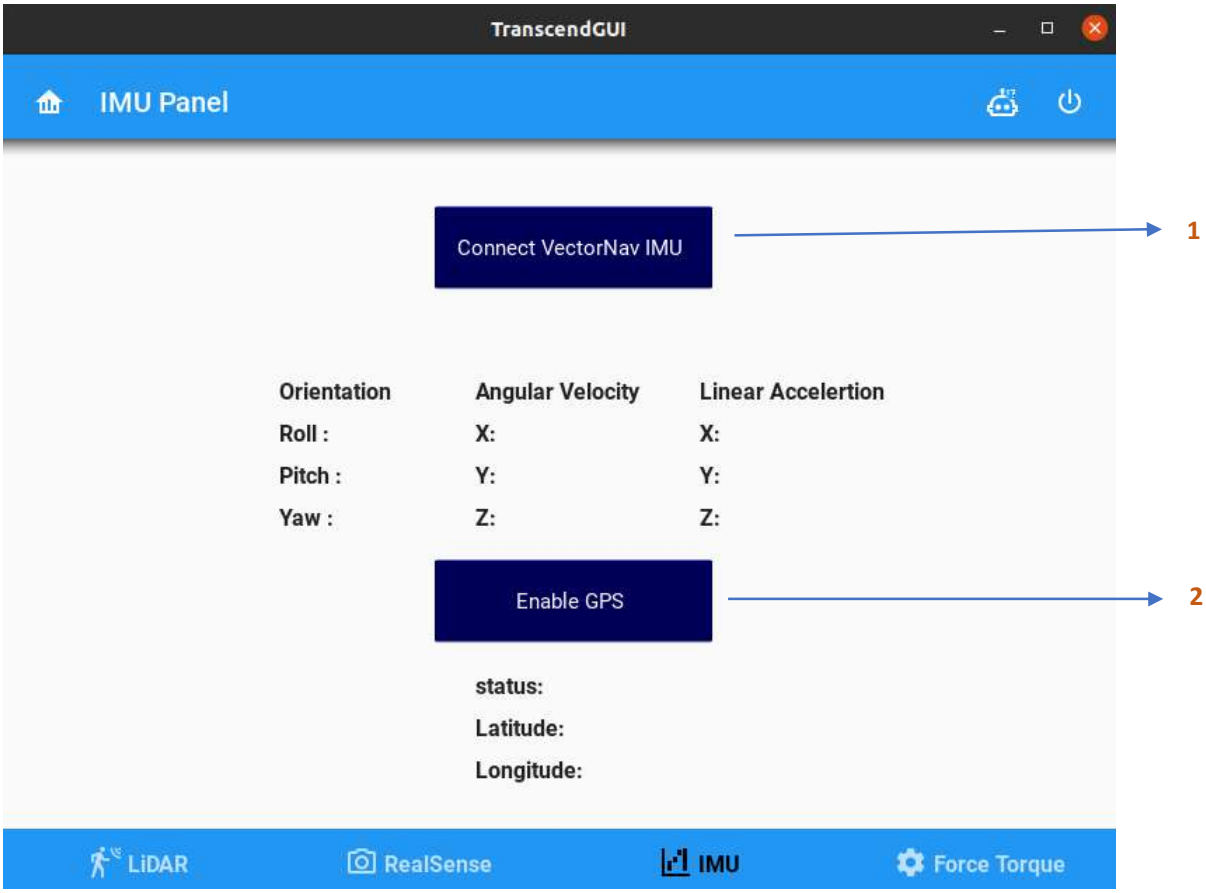
On pressing it, Rviz opens along with the real-time plot of the point cloud data generated by the Lidar.

2 : Some information about the Lidar.

You can refer this YouTube video demonstrating the functionalities of the Lidar panel of the GUI

YouTube link : https://youtu.be/-gZmdyHGZdU

GUI IMU Panel



Components and their functions (IMU Panel)

Make sure the IMU is fully setup and its port number correctly specified in the launch file.

1 : Press the “Connect Vectornav” button to view IMU data

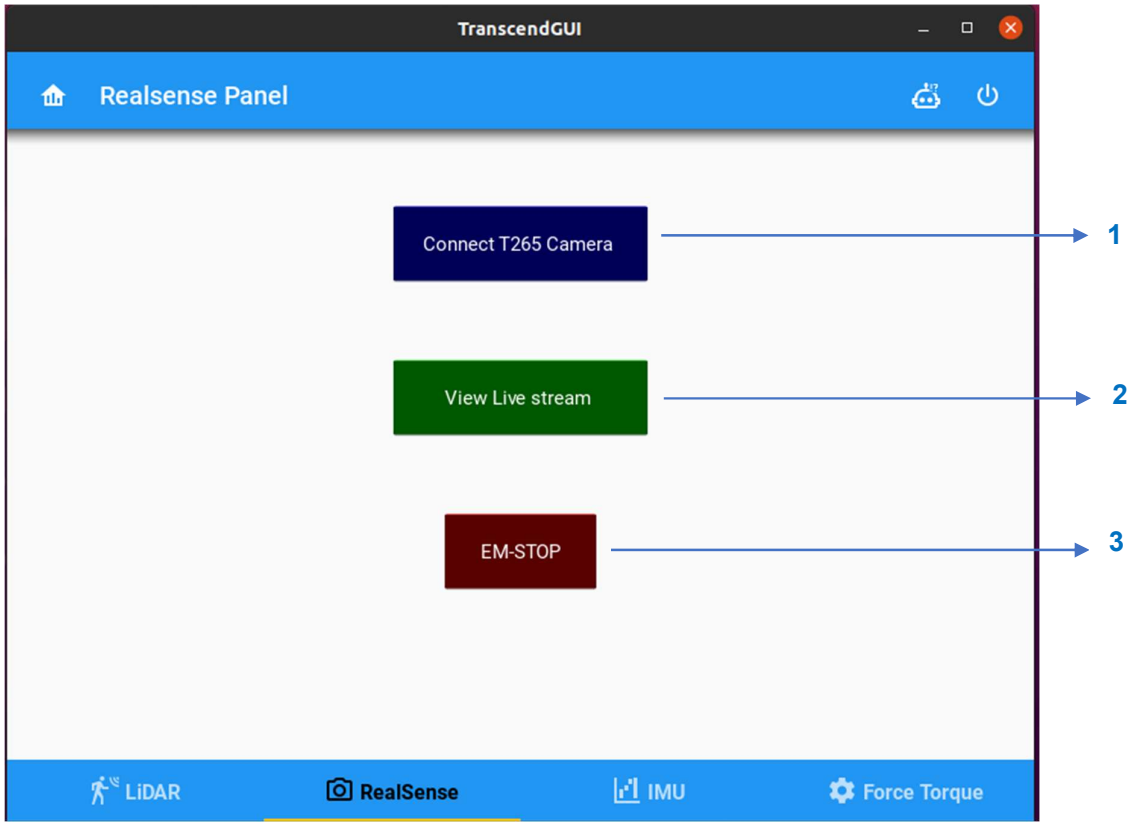
The IMU data stream immediately gets printed next to their corresponding parameter in the GUI.

2 : Press the “Enable” button to view GPS data.

Verify the validity of the Lat, long by checking the status parameter of the GPS module.

Only if the satellite fix has been achieved (status =1) should the data be considered valid.

GUI Realsense Panel



Components and their functions (Realsense Panel)

Make sure the Realsense Camera is fully setup and its port number correctly specified in the launch file.

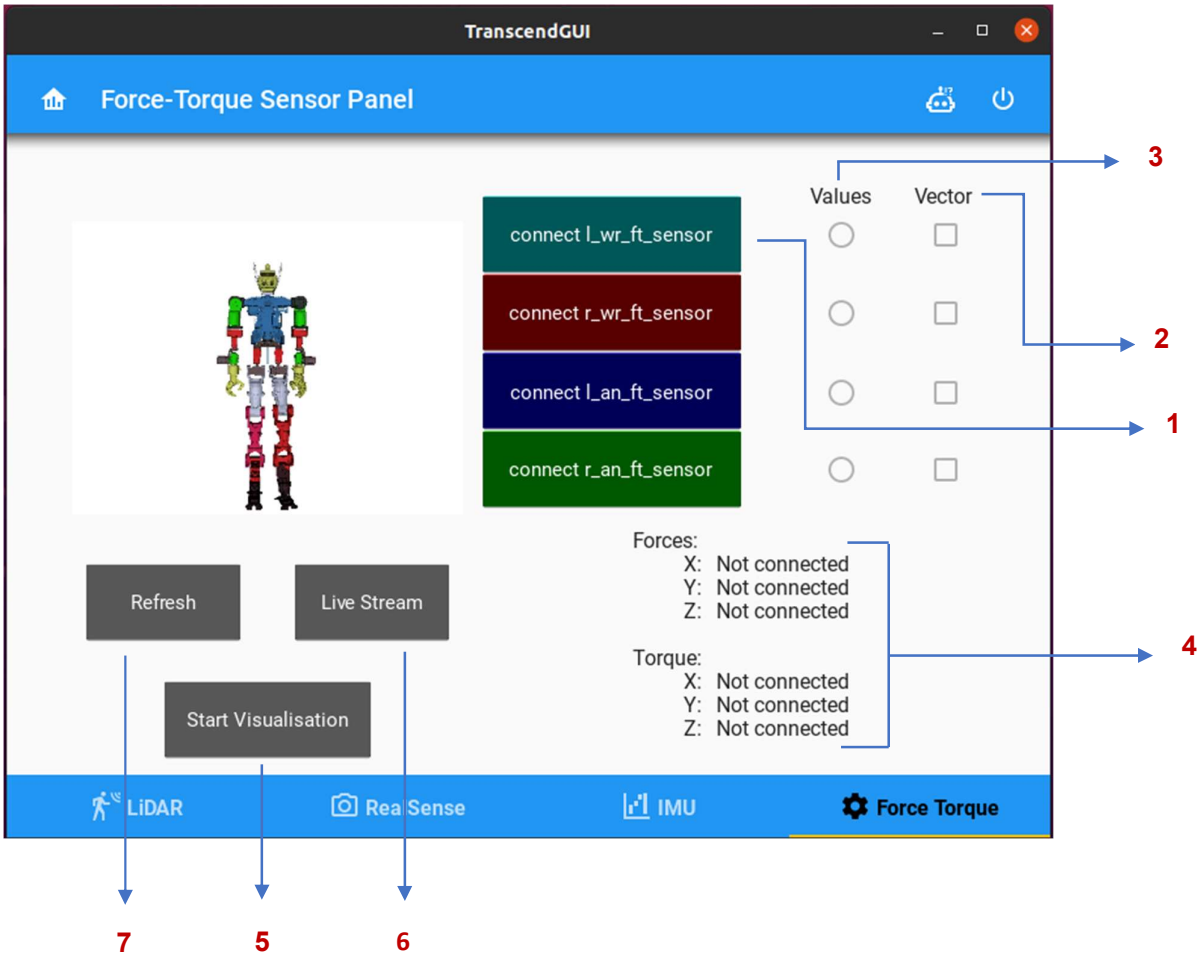
1 : Press the “Connect T265” button to start the communication.

2 : Press “View Live Stream” to view the real-time stream of the camera

The live stream of the image will be visible in a new pop up window,

3 : Emergency stop

GUI Force-Torque Sensor Panel



Components and their functions (Force-Torque Sensor Panel)

Make sure all the FT sensors are fully setup and their port number correctly specified in the launch files.

1 : Press the corresponding FT sensor's button to connect the sensor. The text in the button changes to connected when the sensor starts communicating with the PC.

2 : The user can choose which of the F/T vectors needs to the plotted on the URDF by selecting the appropriate check boxes in the "Vector" column. **(Even all 4 sensors' vector visualisation together are possible)**

3 : The user can choose to view the numerical value of the vectors in the GUI by selecting the appropriate sensor's radio button in the "Values" column. **(Only 1 sensor at a time is possible)**

4 : F/T vector values

5 : Button to start the visualisation of FT vectors.

6 : Press the "Live stream" button to view the real-time plotting of vectors on the URDF file.

Dynamixel Wizard

DYNAMIXEL Wizard 2.0 is an optimized tool for managing DYNAMIXEL from various operating systems. The following features are provided with DYNAMIXEL Wizard 2.0.

- DYNAMIXEL Motor control
- DYNAMIXEL Firmware Update
- DYNAMIXEL Diagnosis
- DYNAMIXEL Configuration and Test
- DYNAMIXEL Data Plotting in Real-Time
- Generate & Monitor DYNAMIXEL Packets

Installation of the Software

To install ROS, please follow the steps mentioned at:
https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/

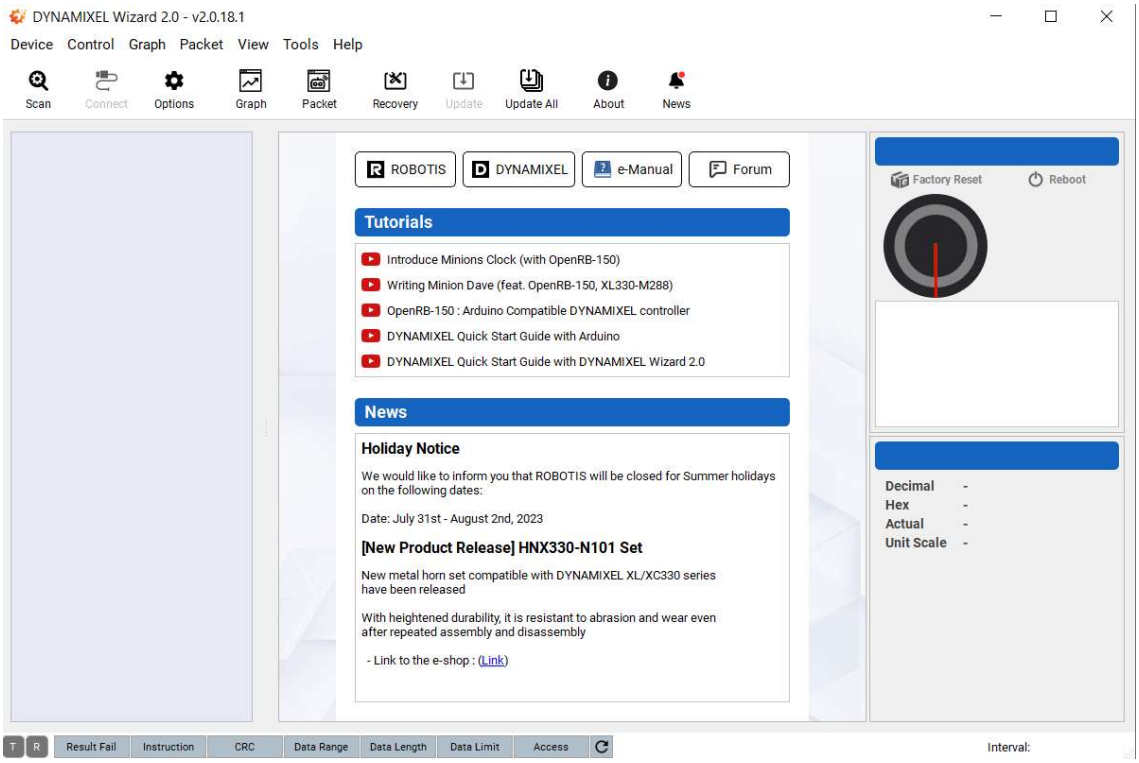
Enter the following command to change the permission.

```
$ sudo chmod 775 DynamixelWizard2Setup_x64
```

Run the program

```
$ ./DynamixelWizard2Setup_x64
```

If everything is installed properly, you should be able to view the home screen.



Control of motors

Go to the **Options** tab and select the USB ports to which the motors are now connected and then select **Scan**

Once the scan is complete, you will be able to view the list of Dynamixel motors detected along with their corresponding ID in the left end of the screen.

Choose the motor you want to control from the list.

To control it, you must always first **enable the torque of the motor** (like shown in highlighted box).

Only then will any of the changes be communicated to the physical system.

You can now choose any of the motor register you want to alter (e.g., goal position).

Input the value in the box provided at the bottom right corner of the screen (see green highlighted box) and then press enter to allow the changes to take place in the physical motors.

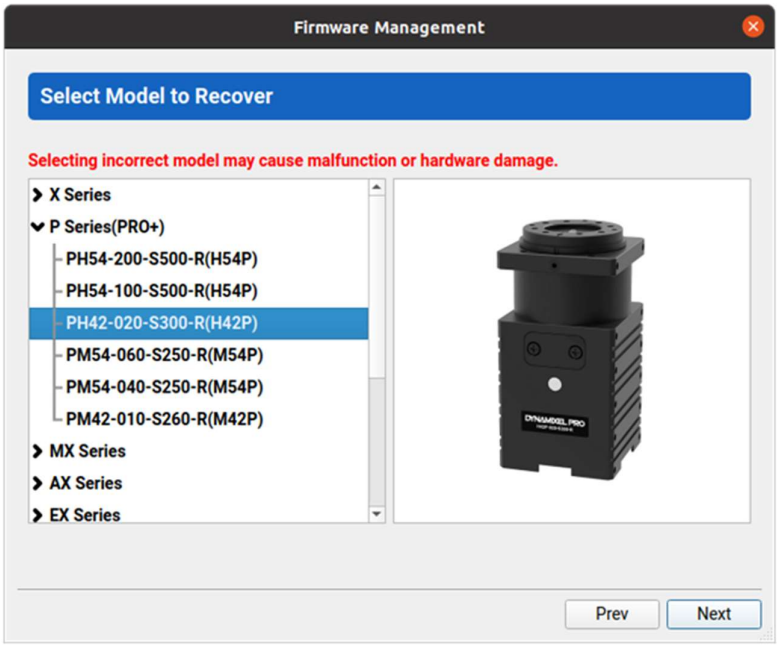
Please be very mindful of the positional limits of the motors in the robot. Trying to rotate the motors to a pose that might lead to a collision might lead to an overload error and damage the components!

Firmware recovery of motors

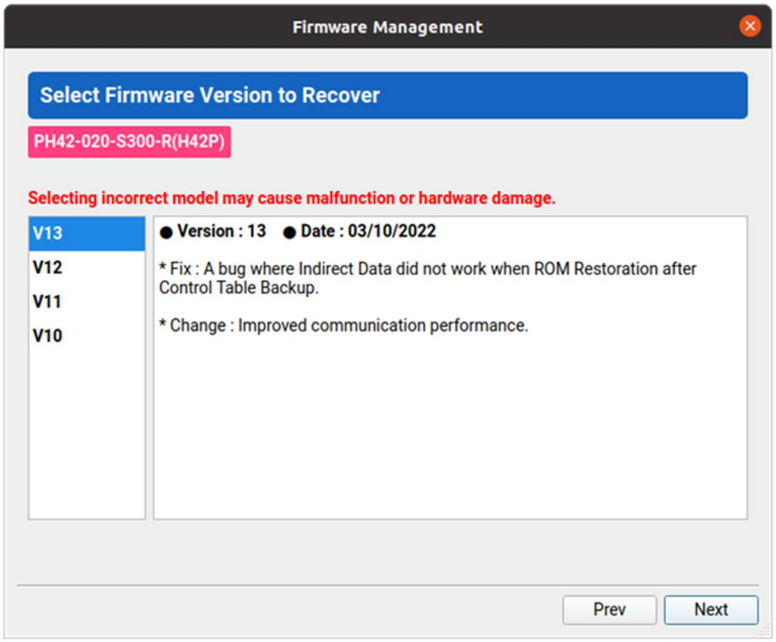
Click the recovery option in the home screen.

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.

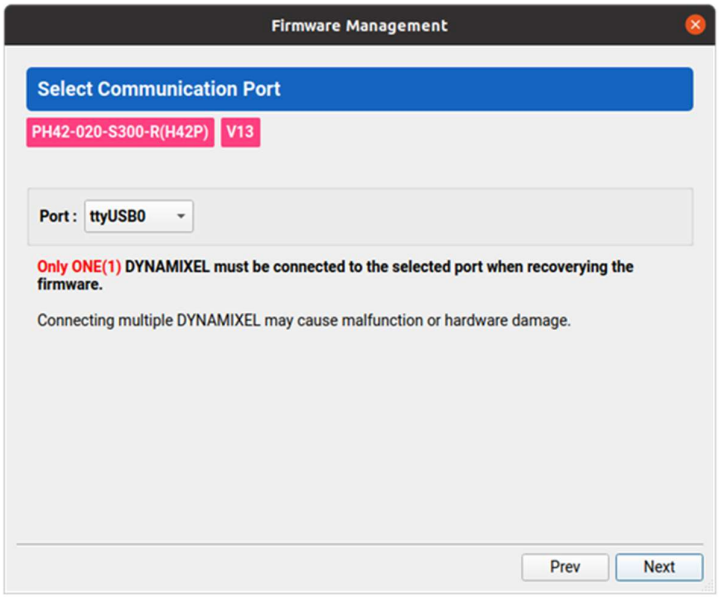
Choose the model of the motor for which you want to recover the firmware for.



Choose the firmware version to recover properly.



Select the appropriate ports



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.

Once the firmware recovery has been done, the motor will start to function normally.

Simultaneous Localization and Mapping

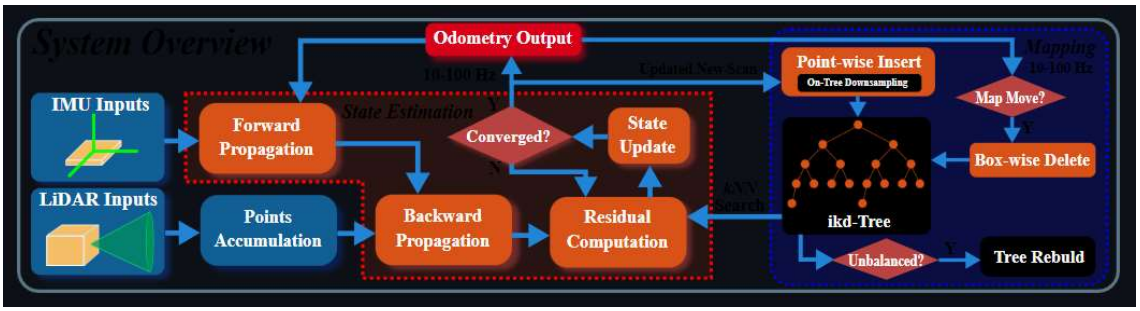
Introduction

SLAM (Simultaneous Localization and Mapping) is a fundamental concept in robotics and autonomous systems. It involves the process of a robot understanding and navigating its environment while simultaneously building a map of that environment. This is crucial for robots to operate autonomously and effectively in unknown or dynamic surroundings.

Implementation of SLAM in the Transcend humanoid robot

The algorithim that is used for SLAM in this robot is known as **FAST-LIO**

FAST-LIO (Fast LiDAR-Inertial Odometry) is a computationally efficient and robust LiDAR-inertial odometry package. It fuses LiDAR feature points with IMU data using a tightly coupled iterated extended Kalman filter to allow robust navigation in fast-motion, noisy or cluttered environments where degeneration occurs.



Installation

Prerequisites

- Ubuntu >= 16.04
- ROS version : Melodic or greater
- PCL Libraries : Version > 1.8
- Eigen Libraries : Version > 3.3.4
- Livox ROS driver

To install PCL libraries run :

```
$ sudo apt install libpcl-dev
```

To install Eigen libraries run :

```
$ sudo apt install libeigen3-dev
```

To compile Livox ros drivers run :

```
$ cd catkin_ws/src
$ git clone https://github.com/Livox-SDK/livox_ros_driver.git
$ cd ..
$ catkin_make
$ cd
$ source ~/.bashrc
```

All the dependencies should be installed as instructed above.

Setting up FAST-LIO

Execute the following commands to compile and build the FAST-LIO package in your system.

```
$ cd catkin_ws/src
$ git clone https://github.com/hku-mars/FAST_LIO.git
$ cd ..
$ catkin_make
$ cd
$ source ~/.bashrc
```

Configuring to suit your robot

Open the **config/velodyne.yaml** file inside the FAST-LIO folder.

```yaml
common:
    lid_topic:  "/velodyne_points"
    imu_topic:  "/imu/data"
    time_sync_en: false         # ONLY turn on when external time synchronization is really not possible
    time_offset_lidar_to_imu: 0.0 # Time offset between lidar and IMU calibrated by other algorithms, e.g. LI-Init (can be found in README).
                                  # This param will take effect no matter what time_sync_en is. So if the time offset is not known exactly, please set as

preprocess:
    lidar_type: 2               # 1 for Livox serials LiDAR, 2 for Velodyne LiDAR, 3 for ouster LiDAR,
    scan_line: 32
    scan_rate: 10               # only need to be set for velodyne, unit: Hz,
    timestamp_unit: 2           # the unit of time/t field in the PointCloud2 rostopic: 0-second, 1-milisecond, 2-microsecond, 3-nanosecond.
    blind: 2

mapping:
    acc_cov: 0.1
    gyr_cov: 0.1
    b_acc_cov: 0.0001
    b_gyr_cov: 0.0001
    fov_degree:    180
    det_range:     100.0
    extrinsic_est_en:  false      # true: enable the online estimation of IMU-LiDAR extrinsic,
    extrinsic_T: [ 0, 0, 0.28]
    extrinsic_R: [ 1, 0, 0,
                   0, 1, 0,
                   0, 0, 1]

publish:
    path_en:  false
    scan_publish_en:  true       # false: close all the point cloud output
    dense_publish_en: true       # false: low down the points number in a global-frame point clouds scan.
    scan_bodyframe_pub_en: true  # true: output the point cloud scans in IMU-body-frame
```

Specify the names of the topic for the point-cloud data and the IMU data according to your setup (See highlighted box).

No need to change anything if you are using the package in the Intel NUC Sensor PC.

Running the SLAM in the NUC Sensor PC

Execute the following commands, **each in the same order specified** and in a new terminal window.

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

```
$ rosrun urdf-rviz pub_req_frames.py
```

```
$ rosrun fast_lio velodyne_mapping.launch
```

If you have configured everything correctly, you can now observe a popped up rviz screen displaying the mapping points along with the real-time frame of the robot.

Force Torque sensor calibration

Introduction

The force and torque sensors measure the force and torque caused due to the external loads attached to the sensor and also due to additional interactions with the environment.

Our goal is to accurately quantify the forces and torques caused only to the external interactions of the robot with the environment.

These interactions may be anything from simple pick-and-place to detecting whether the robot has collided with an obstacle.

Understanding the readings of the F/T sensor in the Transcend Robot

Each F/T sensor has 2 sides :

- mounting side of the robot
- mounting side of the tool

The sensor senses the forces and torques acting only on the **tool mounting side of the robot**.

The forces and torques sensed by the sensor are caused due to **3 different reasons**

- Bias forces and torques due to the pre-tensioning of screws that are used to mount the sensors
- Forces and torques caused due to the already attached hands and legs to the sensor
- Forces and torques caused due to external interactions of the robot with the environment

Therefore, we must **calibrate** the sensor in such a way that we are able to estimate and quantify the forces and torques caused to **each of the above-described reasons.**

The sensor has its own frame of reference. Meaning : The frame of reference 's orientation and translation is not fixed. It rotates and moves along with the sensor.

The sensor measures the X,Y and Z axes components of the force and torque **only with respect to its own frame of reference.**

A simple example demonstrating the behavior of forces

Let **f** be the force caused due to the pre-tensioning of screws

Let $\mathbf{F}^{load}$ be the force caused due to the attached hands/legs

Let $\mathbf{F}^{env}$ be the force caused due to external interactions from the environment.

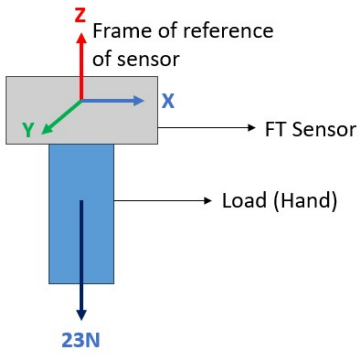
Let $(\mathbf{F})_x$, $(\mathbf{F})_y$, $(\mathbf{F})_z$ be the components of any force in and X,Y and Z axes respectively, where **F** can be **f,** $\mathbf{F}^{load}$, $\mathbf{F}^{env}$

Note : The components are defined with respect to the F/T sensors' own frame of reference.

Let $\mathbf{F}_s$ be the force sensed by the F/T sensor.

$\mathbf{F}_s = \mathbf{f} + \mathbf{F}^{load} + \mathbf{F}^{env}$ (vector addition)

Configuration 1



Example values

f_x = -5 N

f_y = 7.5 N

f_z = -8 N

$(F^{load})_X$ = 0 N

$(F^{load})_Y$ = 0 N

$(F^{load})_Z$ = -23 N

$(F^{env})_X$ = 0 N

$(F^{env})_Y$ = 0 N

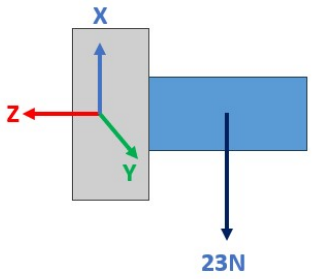
$(F^{env})_Z$ = 0 N

$(F_s)_x$ = -5 N

$(F_s)_y$ = 7.5 N

$(F_s)_z$ = -31 N

Configuration 2



Example values

f_x = -5 N

f_y = 7.5 N

f_z = -8 N

$(F^{load})_X$ = -23 N

$(F^{load})_Y$ = 0 N

$(F^{load})_Z$ = 0 N

$(F^{env})_X$ = 0 N

$(F^{env})_Y$ = 0 N

$(F^{env})_Z$ = 0 N

$(F_s)_x$ = -28 N

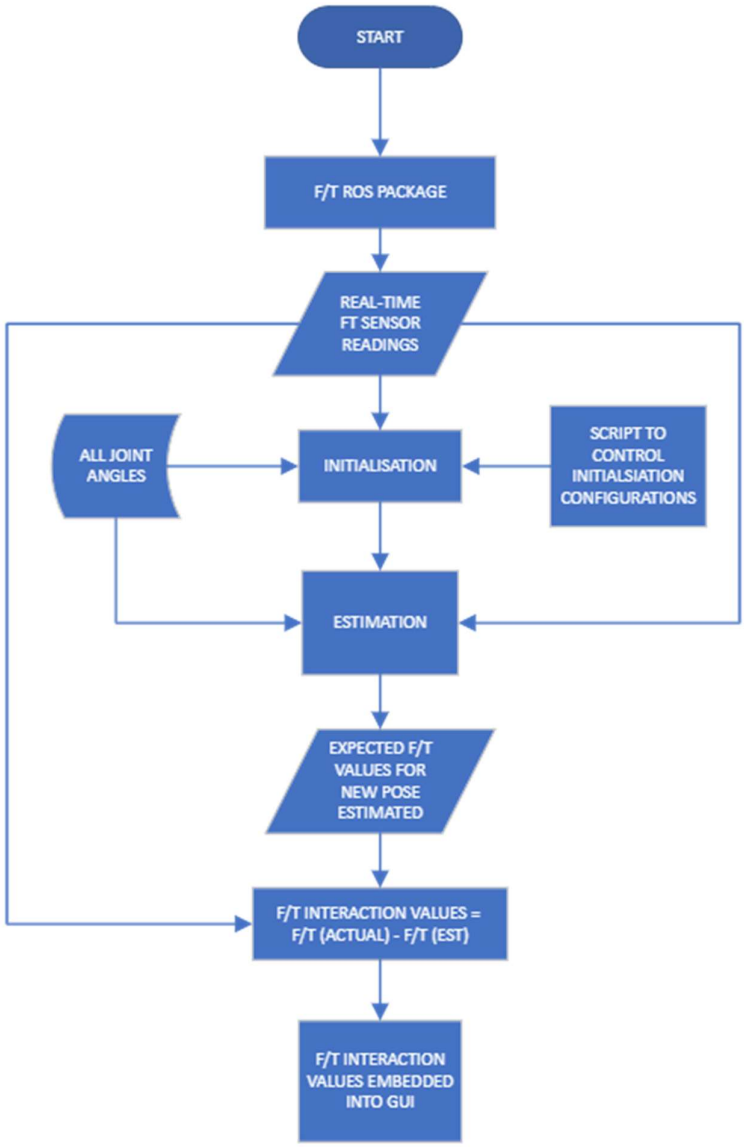
$(F_s)_y$ = 7.5 N

$(F_s)_z$ = -8 N

From the above sample values, we can observe that '**f**' does **NOT** depend upon the orientation of the sensor while the other forces **DO.**

The torques also follow a similar behavior.

Process of calibration



- Real-time F/T sensor readings are obtained from the sensor using the ROS package
- The sensor is calibrated by setting it in 3 different (pre-defined) orientations and calibrated – Forces and torques due to pre-tensioning of screws and hands / legs are calculated.
- The F/T sensor is now moved to a new orientation and its F/T values for that pose is estimated.
- F/T values for additional interactions (if any) are calculated by subtracting the estimated F/T values from the actual F/T values.

Note : If the F/T values for additional interactions are within a particular threshold, the interactions are considered negligible. This is done to account to for unintentional errors in the estimation process.

Getting into the mathematics of the calibration process

Let us consider the F/T sensor attached in the left -arm of the robot.

Force Initialization

All the joints in the left-arm are first initialized to 0 degrees. – **config: 1**

In this configuration,

$$\begin{bmatrix}(\mathrm{F}_s)\mathrm{x}^{c1}\\(\mathrm{F}_s)\mathrm{y}^{c1}\\(\mathrm{F}_s)\mathrm{z}^{c1}\end{bmatrix} = \begin{bmatrix}(\mathrm{f})\mathrm{x}\\(\mathrm{f})\mathrm{y}\\(\mathrm{f})\mathrm{z}\end{bmatrix} + \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}.\begin{bmatrix}(\mathrm{F}_{load})\mathrm{x}\\(\mathrm{F}_{load})\mathrm{y}\\(\mathrm{F}_{load})\mathrm{z}\end{bmatrix}$$ – **eqn: 1** (c1 – config:1)

The sensed F/T values are recorded for this joint angle configuration.

The F/T sensor is rotated by an angle θ_1 along the pitch axis only – **config: 2**

In this configuration.

$$\begin{bmatrix}(\mathrm{F}_s)\mathrm{x}^{c2}\\(\mathrm{F}_s)\mathrm{y}^{c2}\\(\mathrm{F}_s)\mathrm{z}^{c2}\end{bmatrix} = \begin{bmatrix}(\mathrm{f})\mathrm{x}\\(\mathrm{f})\mathrm{y}\\(\mathrm{f})\mathrm{z}\end{bmatrix} + \begin{bmatrix}\cos\theta_1&0&-\sin\theta_1\\0&1&0\\-\sin\theta_1&0&\cos\theta_1\end{bmatrix}.\begin{bmatrix}(\mathrm{F}_{load})\mathrm{x}\\(\mathrm{F}_{load})\mathrm{y}\\(\mathrm{F}_{load})\mathrm{z}\end{bmatrix}$$ – **eqn: 2** (c2 – config:2)

Combining eqn: 1 and 2 :-

$$\begin{bmatrix}(\mathrm{F}_s)\mathrm{x}^{c1}\\(\mathrm{F}_s)\mathrm{y}^{c1}\\(\mathrm{F}_s)\mathrm{z}^{c1}\\(\mathrm{F}_s)\mathrm{x}^{c2}\\(\mathrm{F}_s)\mathrm{y}^{c2}\\(\mathrm{F}_s)\mathrm{z}^{c2}\end{bmatrix} = \begin{bmatrix}1&0&0&1&0&0\\0&1&0&0&1&0\\0&0&1&0&0&1\\1&0&0&\cos\theta_1&0&-\sin\theta_1\\0&1&0&0&1&0\\0&0&1&-\sin\theta_1&0&\cos\theta_1\end{bmatrix}.\begin{bmatrix}(\mathrm{f})\mathrm{x}\\(\mathrm{f})\mathrm{y}\\(\mathrm{f})\mathrm{z}\\(\mathrm{F}_{load})\mathrm{x}\\(\mathrm{F}_{load})\mathrm{y}\\(\mathrm{F}_{load})\mathrm{z}\end{bmatrix}$$

Solving the above equation, we get :

$$\begin{bmatrix}(\mathrm{f})\mathrm{x}\\(\mathrm{f})\mathrm{y}\\(\mathrm{f})\mathrm{z}\\(\mathrm{F}_{load})\mathrm{x}\\(\mathrm{F}_{load})\mathrm{y}\\(\mathrm{F}_{load})\mathrm{z}\end{bmatrix} = \begin{bmatrix}1&0&0&1&0&0\\0&1&0&0&1&0\\0&0&1&0&0&1\\1&0&0&\cos\theta_1&0&-\sin\theta_1\\0&1&0&0&1&0\\0&0&1&-\sin\theta_1&0&\cos\theta_1\end{bmatrix}^{-1}.\begin{bmatrix}(\mathrm{F}_s)\mathrm{x}^{c1}\\(\mathrm{F}_s)\mathrm{y}^{c1}\\(\mathrm{F}_s)\mathrm{z}^{c1}\\(\mathrm{F}_s)\mathrm{x}^{c2}\\(\mathrm{F}_s)\mathrm{y}^{c2}\\(\mathrm{F}_s)\mathrm{z}^{c2}\end{bmatrix}$$

Using this method, we can accurately estimate the force caused due to the pre-tensionsing of screws and due to the hands/legs attached to the sensor.

Estimation for forces for any pose

To estimate the readings of the F/T sensor for a new pose for the arm use the following equation :

$$\begin{bmatrix}(\mathrm{F}_s)\mathrm{x}^{est}\\(\mathrm{F}_s)\mathrm{y}^{est}\\(\mathrm{F}_s)\mathrm{z}^{est}\end{bmatrix} = \mathbf{R}(\emptyset,\theta,\partial) * \begin{bmatrix}(\mathrm{F}_{load})\mathrm{x}\\(\mathrm{F}_{load})\mathrm{y}\\(\mathrm{F}_{load})\mathrm{z}\end{bmatrix} + \begin{bmatrix}(\mathrm{f})\mathrm{x}\\(\mathrm{f})\mathrm{y}\\(\mathrm{f})\mathrm{z}\end{bmatrix}$$

Where $\mathbf{R}(\emptyset,\theta,\partial)$ is the rotational matrix for the frame of orientation $(\emptyset,\theta,\partial)$ are the angles along the X, Y and Z axes. **Please refer to the appendix to know how to calculate the R matrix**.

Calculation of additional interaction forces because of external environment

To calculate the additional forces caused due to external interactions with the environment,

$$\begin{bmatrix} (\mathrm{F})\mathrm{x}^{env} \\ (\mathrm{F})\mathrm{y}^{env} \\ (\mathrm{F})\mathrm{z}^{env} \end{bmatrix} = \begin{bmatrix} (\mathrm{F}_s)\mathrm{x}^{actual} \\ (\mathrm{F}_s)\mathrm{y}^{actual} \\ (\mathrm{F}_s)\mathrm{z}^{actual} \end{bmatrix} - \begin{bmatrix} (\mathrm{F}_s)\mathrm{x}^{est} \\ (\mathrm{F}_s)\mathrm{y}^{est} \\ (\mathrm{F}_s)\mathrm{z}^{est} \end{bmatrix}$$

Torque Initialization

We have discussed previously that the torques are also caused due to 3 reasons : pre-tensioning of screws, load of hands/legs, external interactions in the environment.

Let us assume :

τ^{screws} be the torque caused due to the pre-tensioning of screws.

τ^{load} be the torque caused due to the weight of the hands/legs.

τ^{env} be the torque caused external interactions with the environment.

τ^{s} be the torque sensed by the F/T sensor.

$$\vec{\tau}^{s} = \vec{\tau}^{screws} + \vec{\tau}^{load} + \vec{\tau}^{env}$$

We know that the torque : $\vec{\tau} = \overrightarrow{r_{com}} * \vec{F}$ where:

$\vec{F}$ is the force vector

$\overrightarrow{r_{com}} = \begin{bmatrix} x_{com} \\ y_{com} \\ z_{com} \end{bmatrix}$ is the position of the center of mass of the force

Therefore we can calculate $\tau_{load} = (y_{com}(\mathrm{F}_{load})\mathrm{z} - z_{com}(\mathrm{F}_{load})\mathrm{y})\hat{\imath}$

$$+ (z_{com}(\mathrm{F}_{load})\mathrm{x} - x_{com}(\mathrm{F}_{load})\mathrm{z})\hat{\jmath}$$

$$+ (x_{com}(\mathrm{F}_{load})\mathrm{y} - y_{com}(\mathrm{F}_{load})\mathrm{x})\hat{k}$$

However to calculate τ_{env} we don't know the COM co-ordinates and the value of τ_{screws}

Hence, we will use 1 more configuration of the FT sensor to calculate these values.

At config : 2 –

$$(\tau_s)_x{}^{c2} = (\tau_{screws})_x + (\tau_{load})_x{}^{c2}$$

$$(\tau_s)_y{}^{c2} = (\tau_{scr}\quad)_y + (\tau_{load})_y{}^{c2}$$

$$(\tau_s)_z{}^{c2} = (\tau_{scr}\quad)_z + (\tau_{load})_z{}^{c2}$$

Note : $(\tau_{load})^{c2} = \overrightarrow{r_{com}} * [\mathbf{R}(\emptyset, \theta, \partial).\vec{\mathrm{F}}_{load}]$

Where $\mathbf{R}(\emptyset, \theta, \partial)$ is the rotation matrix corresponding to the orientation of the sensor in c2

$\emptyset = 0$

$\theta = \theta_1$

$\partial = 0$

The sensor is now rotated to θ_2 along the pitch axis only – **config: 3**

We can formulate a similar set of equations for the new configuration c3.

$(\tau_s)_x{}^{c3} = (\tau_{scre\quad})_x + (\tau_{load})_x{}^{c3}$

$(\tau_s)_y{}^{c3} = (\tau_{screws})_y + (\tau_{load})_y{}^{c3}$

$(\tau_s)_z{}^{c3} = (\tau_{screws})_z + (\tau_{load})_z{}^{c3}$

$(\tau_{load})^{c3} = \overrightarrow{r_{com}} * [\mathbf{R}(\emptyset, \theta, \partial).\vec{\mathrm{F}}_{load}]$

Note : τ_{load} is a function of x_{com} , y_{com} , z_{com}

Using the above set of 6 equations, we can now calculate the XYZ components of τ_{scr} and x_{com} , y_{com} , z_{com}

Estimation for forces for any pose

To estimate the readings of the F/T sensor for a new pose for the arm use the following equation :

$\vec{\tau}_{est} = \vec{\tau}_{screws} + \overrightarrow{r_{com}} * [\mathbf{R}(\emptyset, \theta, \partial).\vec{\mathrm{F}}_{load}]$

Where $\mathbf{R}(\emptyset, \theta, \partial)$ is the rotational matrix for the frame of orientation $(\emptyset, \theta, \partial)$ are the angles along the X, Y and Z axes.

Calculation of additional interaction torques because of external environment

To calculate the additional torques caused due to external interactions with the environment

$\vec{\tau}_{env} = \vec{\tau}_{actual} - \vec{\tau}_{est}$

Concluding this section, in this way, we can accurately quantify the forces and torques caused only due to interactions with the external environment.

This will help us in a lot of things from detecting if the end-effector has collided with an object, detecting if the end-effector has grasped an object or not and other things.

Please go through the code in the bota_demo package that implements this logic.

Run it using the command :

```
$ rosrun bota_demo ft_init2.py
```

Reading data from all dynamixel motors simultaneously

The transcend humanoid robot has 37 dynamixel motors in its system. Each of the dynamixel motors provides the following feedback.

- Voltage (V)
- Current (mA)
- Position (encoder value)
- Velocity (rads/s)
- Temperature (Celsius)

It is essential for multiple reasons to get all the above mentioned feedback data from all the dynamixel motors simultaneously.

To achieve this,

Connect and power all the dynamixel motors and run the command below in the terminal.

```
$ roscore
```

Open a new terminal window and run :

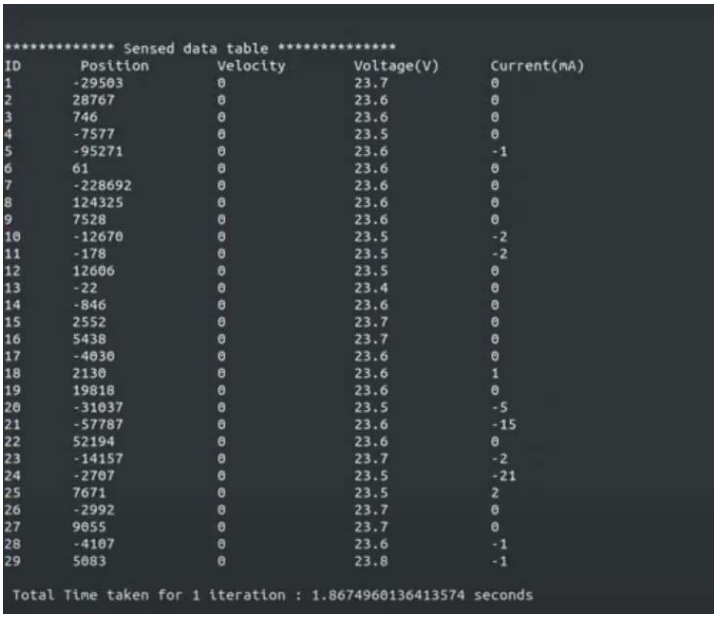
```
$ rosrun dynamixel_sdk_examples read_write_node_multiple.py
```

You can observe that (if the port numbers are set correctly) all the feedback data from all the motors gets printed on the terminal.

Each motor can be identified using its ID number.

Run the same process in a loop to get a continuous stream of data

A sample output image is shown below.



```
************* Sensed data table *************
ID       Position        Velocity        Voltage(V)      Current(mA)
1       -29503          0               23.7            0
2       28767           0               23.6            0
3       746             0               23.6            0
4       -7577           0               23.5            0
5       -95271          0               23.6            -1
6       61              0               23.6            0
7       -228692         0               23.6            0
8       124325          0               23.6            0
9       7528            0               23.6            0
10      -12670          0               23.5            -2
11      -178            0               23.5            -2
12      12606           0               23.5            0
13      -22             0               23.4            0
14      -846            0               23.6            0
15      2552            0               23.7            0
16      5438            0               23.7            0
17      -4030           0               23.6            0
18      2130            0               23.6            1
19      19818           0               23.6            0
20      -31037          0               23.5            -5
21      -57787          0               23.6            -15
22      52194           0               23.6            0
23      -14157          0               23.7            -2
24      -2707           0               23.5            -21
25      7671            0               23.5            2
26      -2992           0               23.7            0
27      9055            0               23.7            0
28      -4107           0               23.6            -1
29      5083            0               23.8            -1

Total Time taken for 1 iteration : 1.8674960136413574 seconds
```

Note : Each iteration takes about 1.8 seconds to complete !

Torque estimation for dynamixel motors

In many cases, it becomes important to quantify the torque supplied by the dynamixel motors.

Theoretically, The torque supplied by the motor and the current drawn are linearly related.

$$\tau = k * I$$

Where τ is the torque (Nm)

I is the current drawn by the motor (A)

K is called the motor constant (Nm/A)

ROBOTIS does not disclose the motor constant of their motors. However, they have given the torque vs current graphs for each of their motors in their website.

https://emanual.robotis.com/docs/en/dxl/p/ph54-200-s500-r/

https://emanual.robotis.com/docs/en/dxl/p/ph54-100-s500-r/

https://emanual.robotis.com/docs/en/dxl/p/ph42-020-s300-r/

Therefore, in order to estimate the torque, we have sampled different points in the graph and have fitted a quadratic expression to interpolate for other current values.

The sampled points and the curve fitting graphs can be found in this link :

https://docs.google.com/spreadsheets/d/1mv5QUwNVsu2fJFc34OygrIlAQ1mn_BXqXhH3BY3_t1o/edit?usp=sharing

The torque – current relations estimated are :

Motor model number	Torque equation
PH54-100-S500-R	$-6.33 + 6.19I - 0.0716I^2$
PH54-200-S500-R	$-10.3 + 7.47I - 0.102I^2$
PH42-020-S300-R	$-2.9 + 5.77I - 0.205I^2$

Please use these equations to estimate the torque produced by the motor.

Appendix

ROS Software Packages in the workspace

Bota-driver
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 force-torque sensor. This is at the moment just a skeleton to connect to devices and support the driver development of its firmware.

Bota-demo
The bota-demo package is a ROS package developed by the BotaSys team. It contains example configuration and launch files to interface the various models of the Bota System's FT sensors.

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.

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.

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.

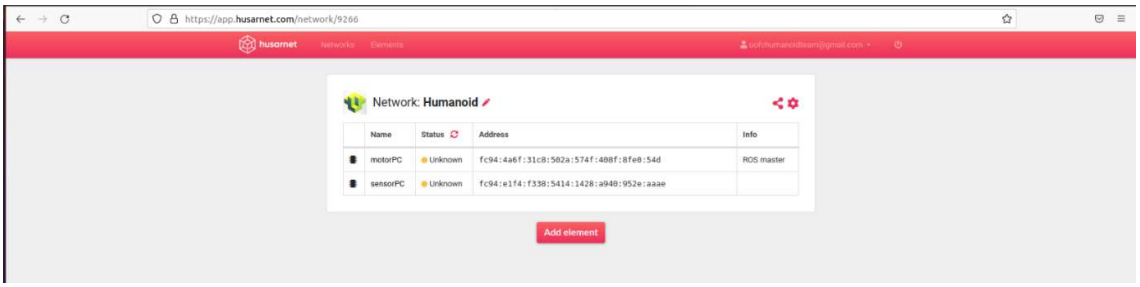
URDF-RVIZ Package
The urdf-rviz package is a ROS package developed by the 2022/23 capstone design team to visualize the robot's URDF in an RVIZ window and stream it over a rostopic. This is used to have a dynamic display of the robot's orientation in the GUI.

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.

FAST-LIO
Fast-LIO is the ROS package that is used to implement the SLAM algorithim.

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!

Rotation Matrix Calculation

Please refer to the link below.
https://www.cuemath.com/algebra/rotation-matrix/