Refer Git-Hub link given below for more information:
https://github.com/abhishekrs0001

---

1. Controlling a ROS-Enabled Robot using Matlab simulink:

1.Reference: Examples by ROS Toolbox User's guide(MATLAB 2017).

2.Creating own Two wheeled Differential Robot model.

3.Launching the Two Wheeled Robot.

$ ***roslaunch ros_robotics diff_wheeled_gazebo_full.launch***

- Files required: (**cd ~/catkin_ws_ars/src/ros_robotics**)
  - ❏ diff_wheeled_gazebo_full.launch
  - ❏ diff_wheeled_robot_with_sensor.xacro
  - ❏ wheel.urdf .xacro

4. Run the Matlab (**cd ~/Computer/usr/local/MATLAB/R2017b/bin**)

$ ***sudo ./matlab***

Model_1:

>> ***open_system('Two_Wheeled_Robot_With_Default_Head_5.slx');***

Model_2:

>>***open_system('Two_Wheeled_Robot_With_Desired_Head_6.slx');***

- Files required:
  - ❏ (**~/Desktop/ARS_JRF/JRF_WORK/1_MATLAB_WORKSPACE/SIMULINK_MY_MODELS**)

- Reference model:Examples by ROS Toolbox User's guide.

>> ***open_system('robotROSFeedbackControlExample.slx');***

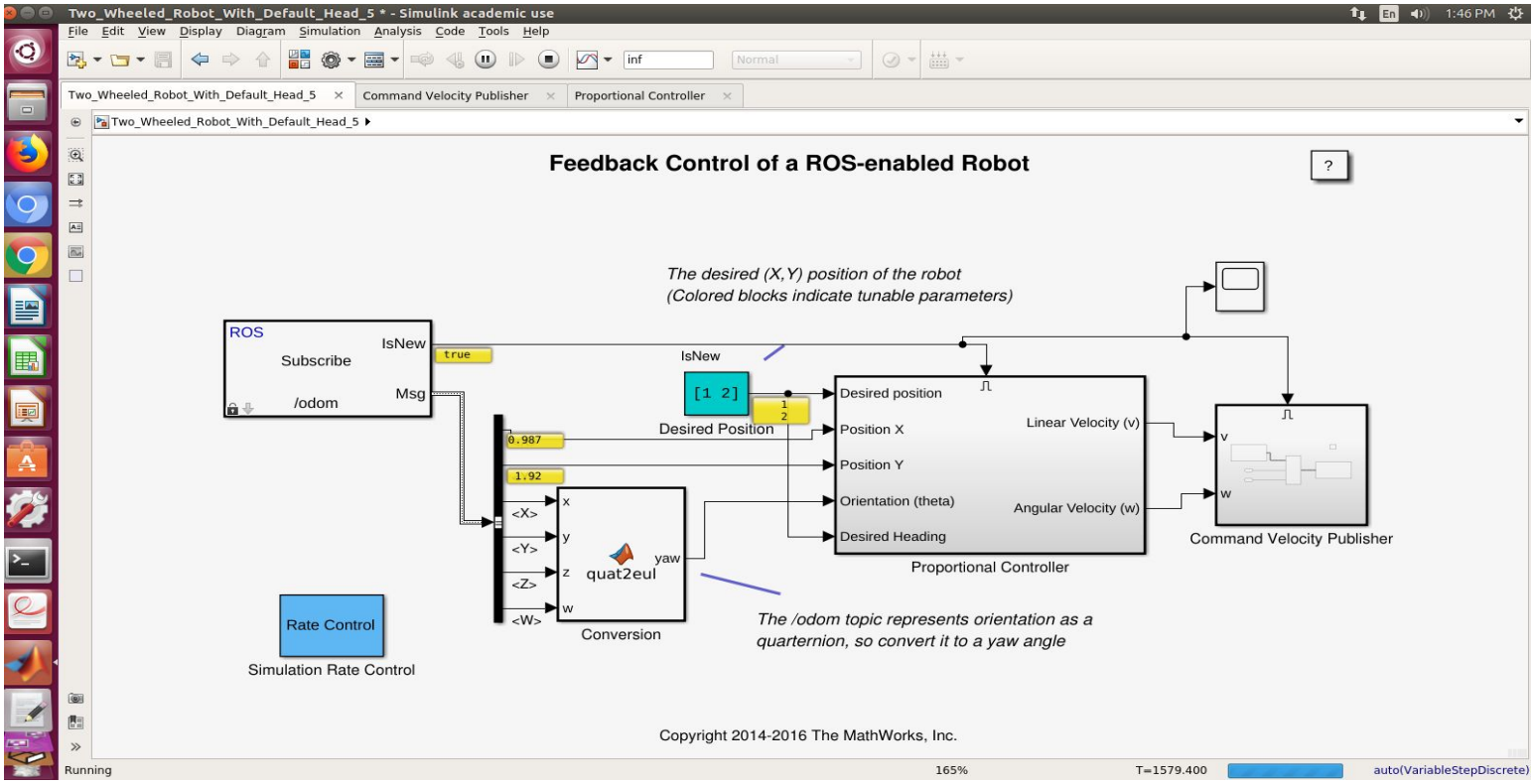
5. Configure Simulink to connect to the ROS network.

➢ **Main menu-Tools-Robotic Operating System-Configure Ros Network address**
ROS Master section:
-Hostname/IP add: **10.250.12.226** (ROS-Master)
-Port: 11311

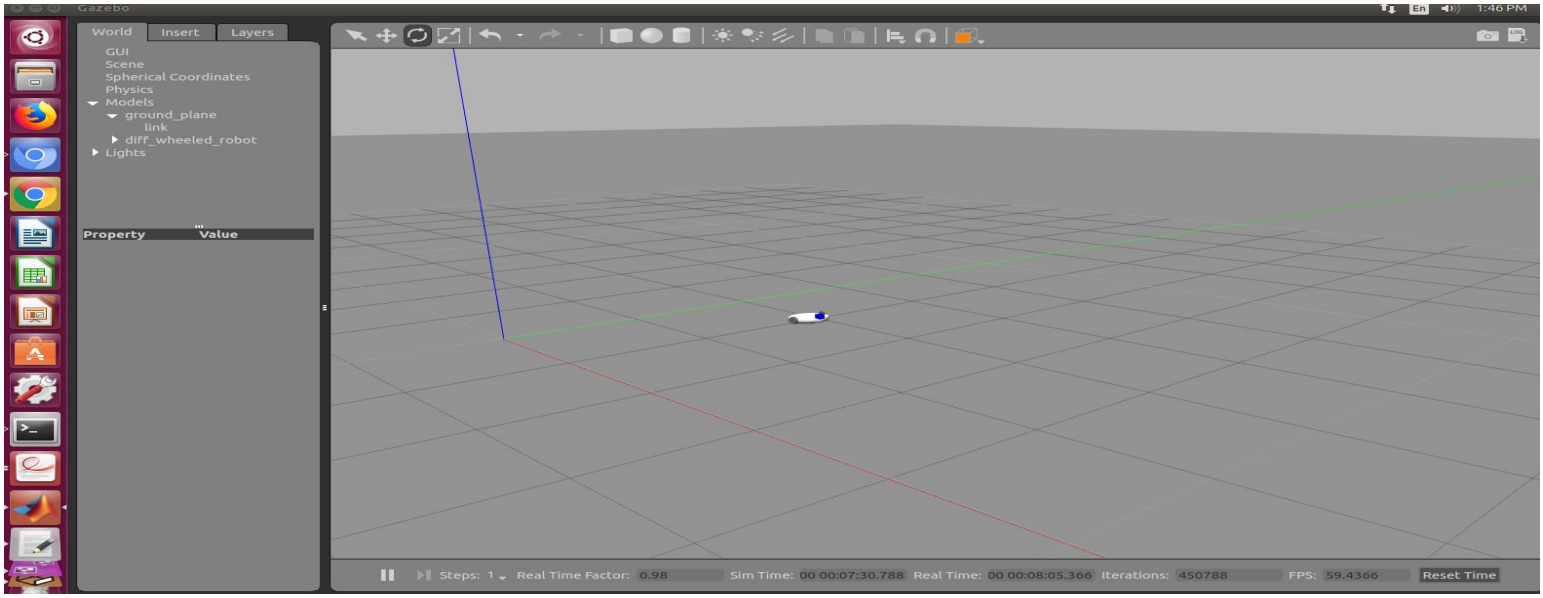
★ TROUBLESHOOTING:

1. Execute following commands:
   - ***cd ~/catkin_ws_ars***
   - ***catkin_make***
   - ***source devel/setup.bash***
2. Check for topic name by executing following command.
   - ***rostopic list***

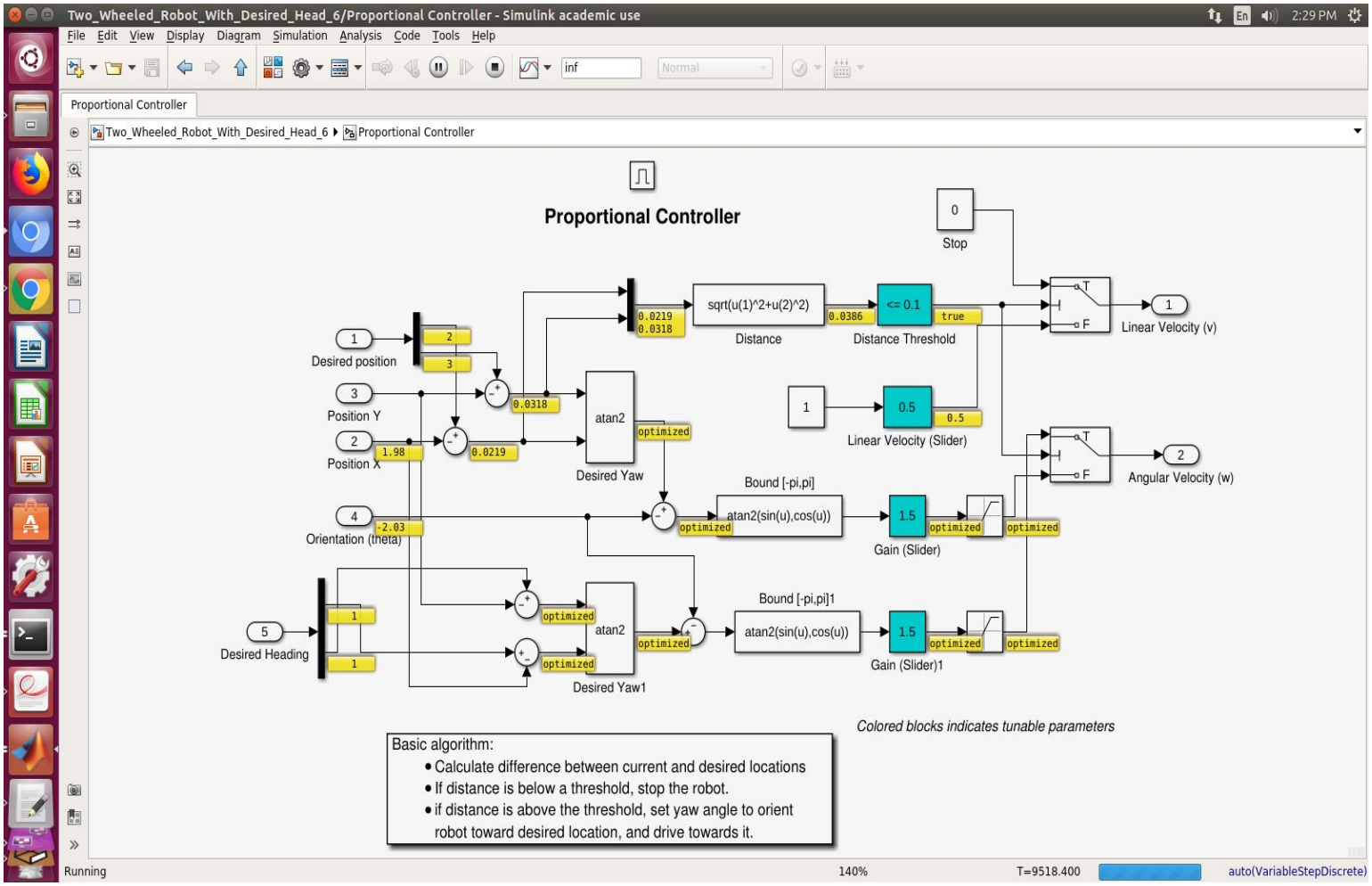
   Use the same topic names in a simulink model.



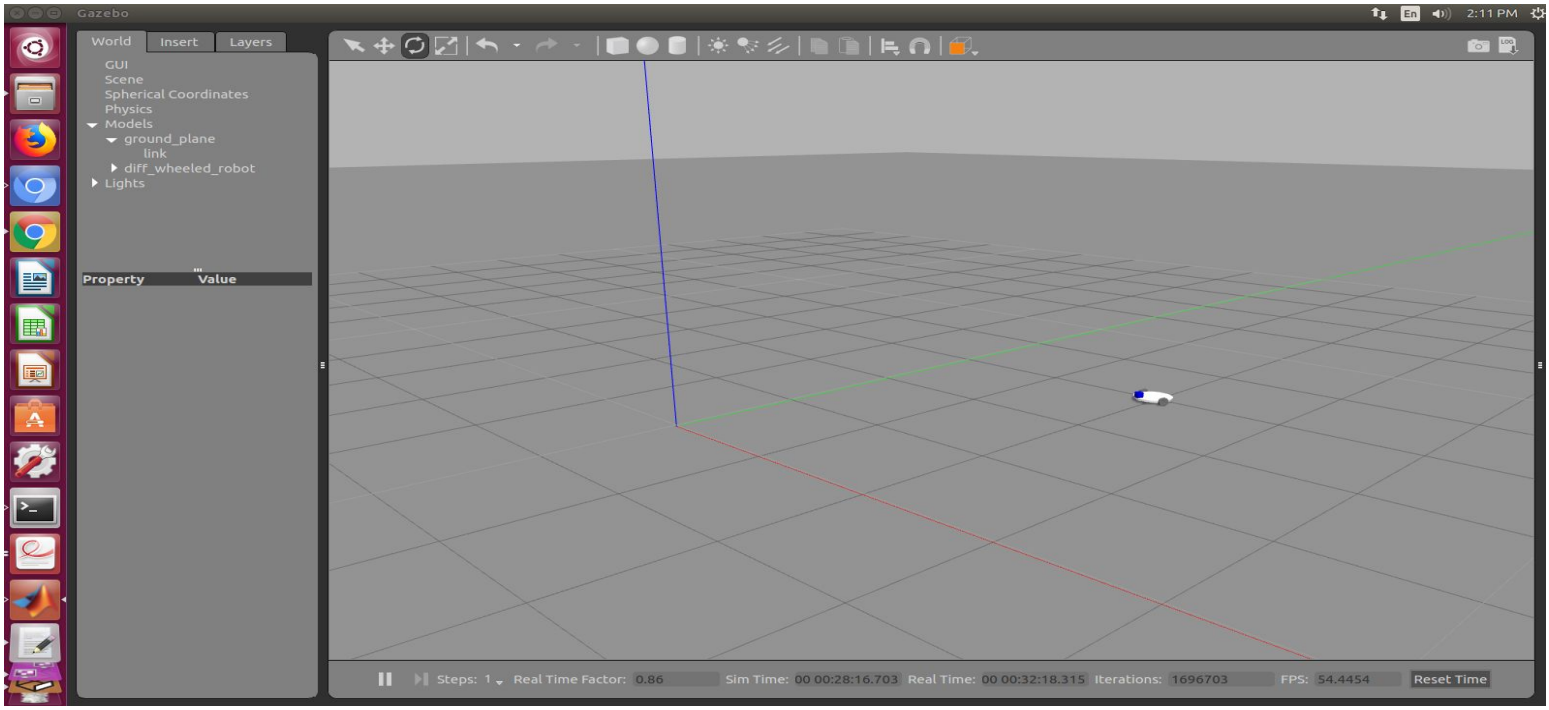
[-Screenshot1_Model1]



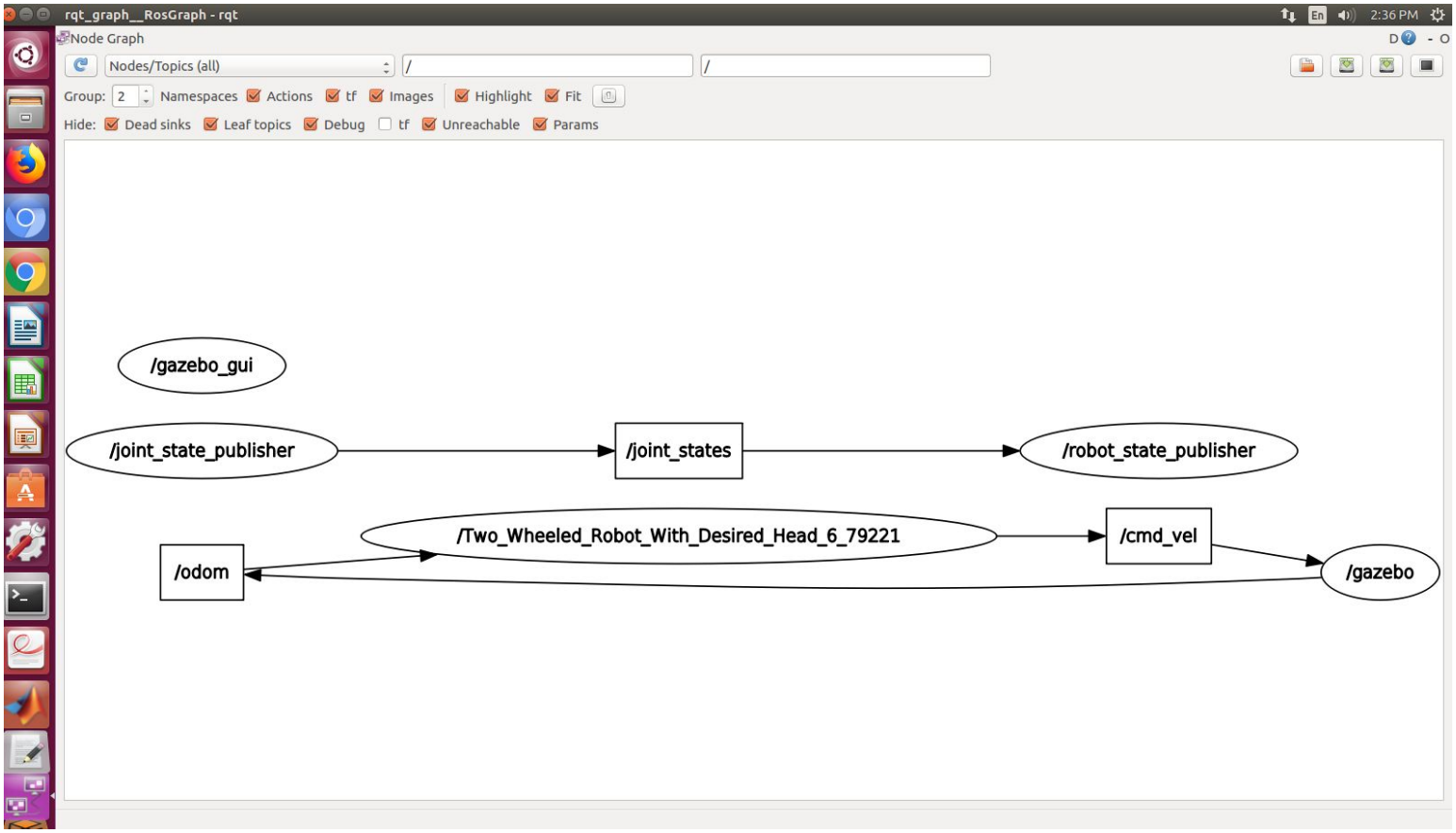
[-Screenshot2_model1]



[-Screenshot3_Proportional controller_Model2]



[-Screenshot4_Model2]



[-Screenshot5_rqtgraph_Model2]

2. Interfacing the Novint Falcon haptic device and testing its functionality with Chai3D:

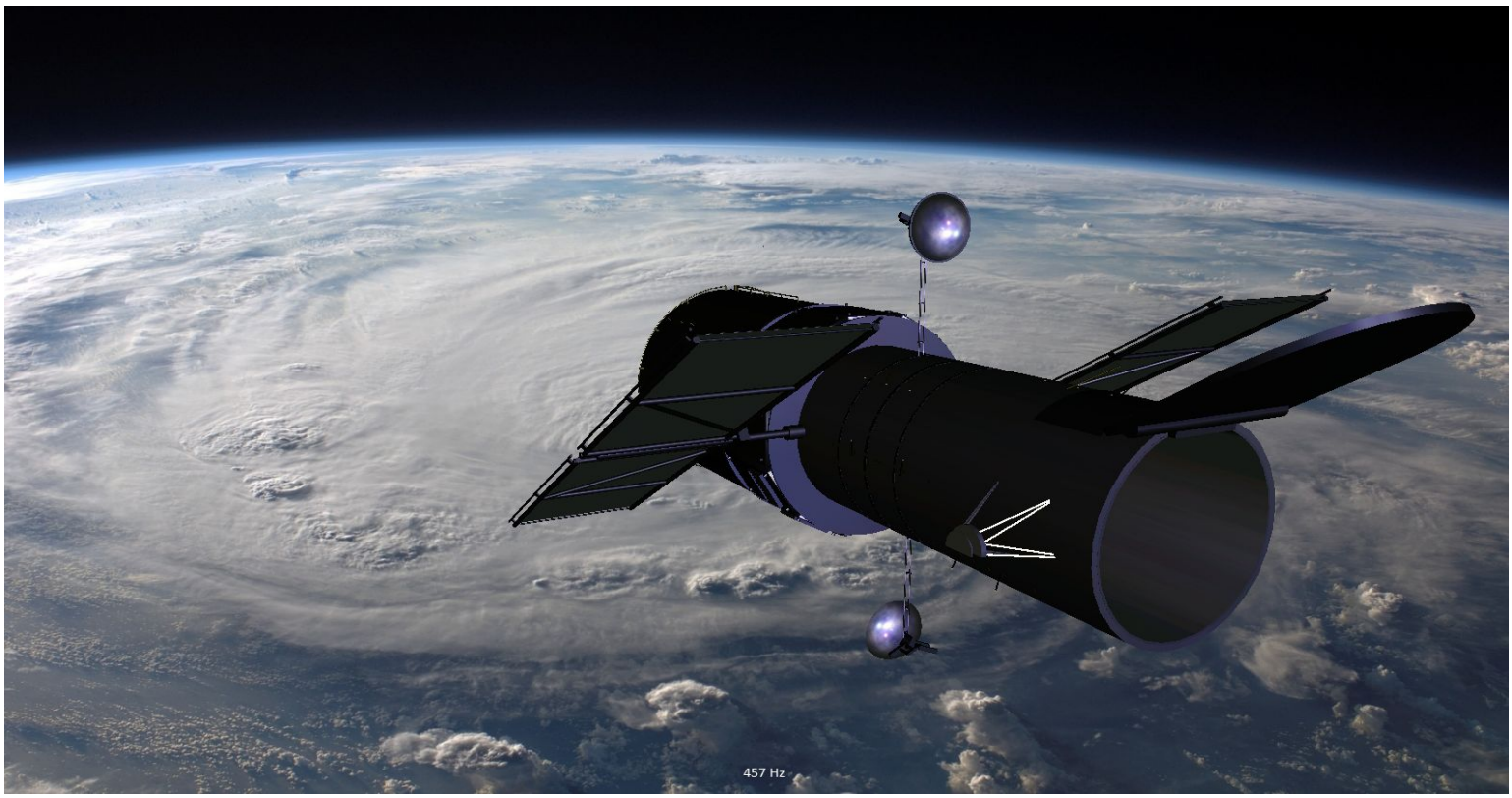
➢ Download the compatible version of chai3D from the website *https://www.chai3d.org/download/releases*

- Required supporting library files:

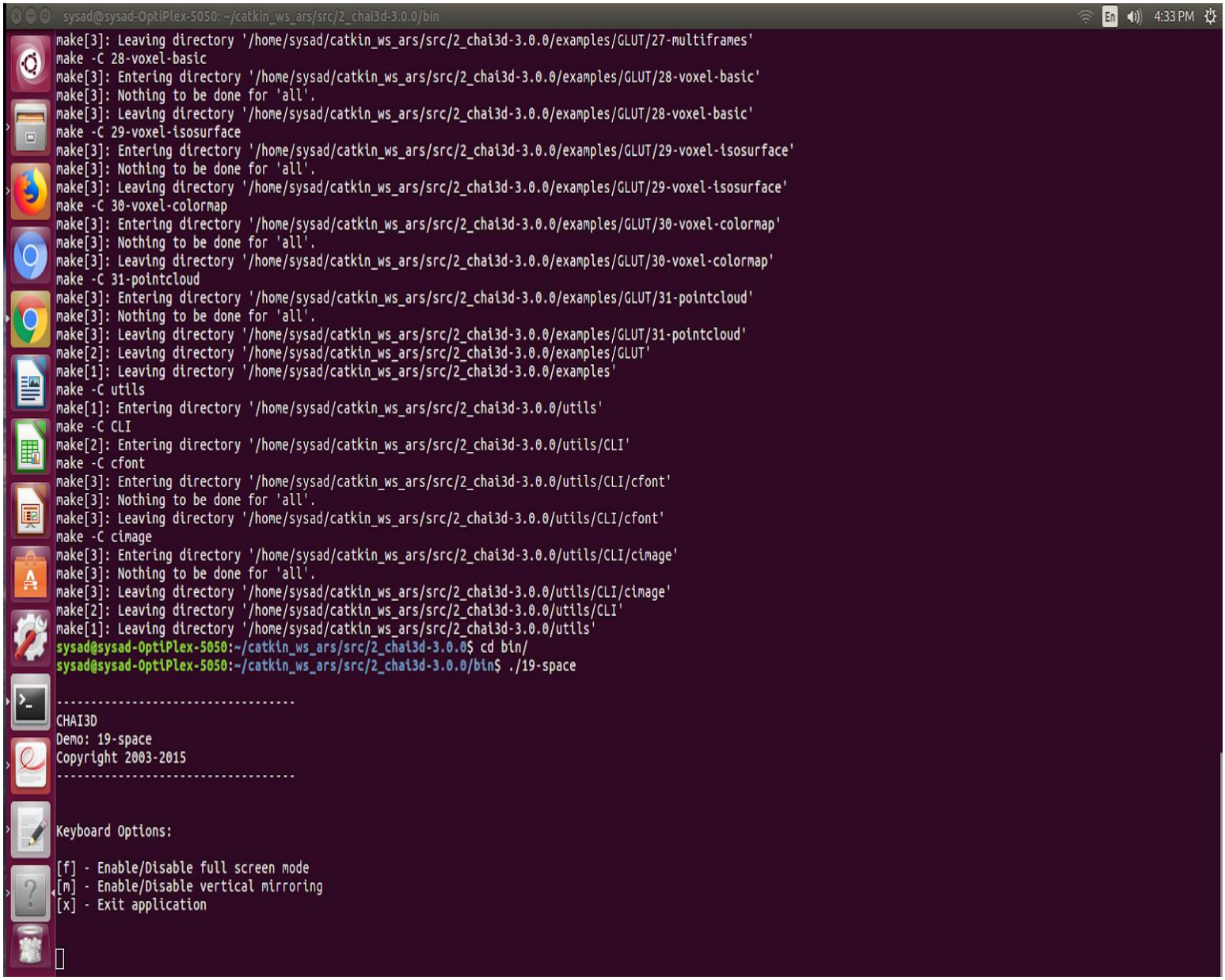
❑ **(cd ~/Desktop/ARS_JRF/JRF_WORK/2_INTERFACE/2_Libraries)**

➢ Steps to be followed to test the Novint Falcon functionality:

$ ***cd ~/catkin_ws_ars/src/2_chai3d-3.0.0***
$ ***make***
$ ***cd bin/***
$ ***./19-space***



[-Screenshot6_SpaceExample]



[-Screenshot7_Chai3D]

---

3. Interfacing the Novint Falcon + ROS using *libnifalcon:*

"libnifalcon is a library that helps to interface ROS
(<kinetic> version & others like melodic etc...) with Linux Ubuntu 16.04"

- Required other supporting libraries:

1. cmake
   http://www.cmake.org/

2. libusb 1.0 (Recommended for Linux or Mac)
   http://www.libusb.info

3. GLUT 4. libasound2 (used these while starting with CHAI3D)

5. ftd2xx (Recommended for Windows)
   http://www.ftdichip.com/Drivers/D2XX.htm

➢ "libnifalcon installation steps"

1. clone the libnifalcon library at home folder.

   $ git clone https://github.com/libnifalcon/libnifalcon.git
   $ cd libnifalcon
   $ mkdir build
   $ cd build/

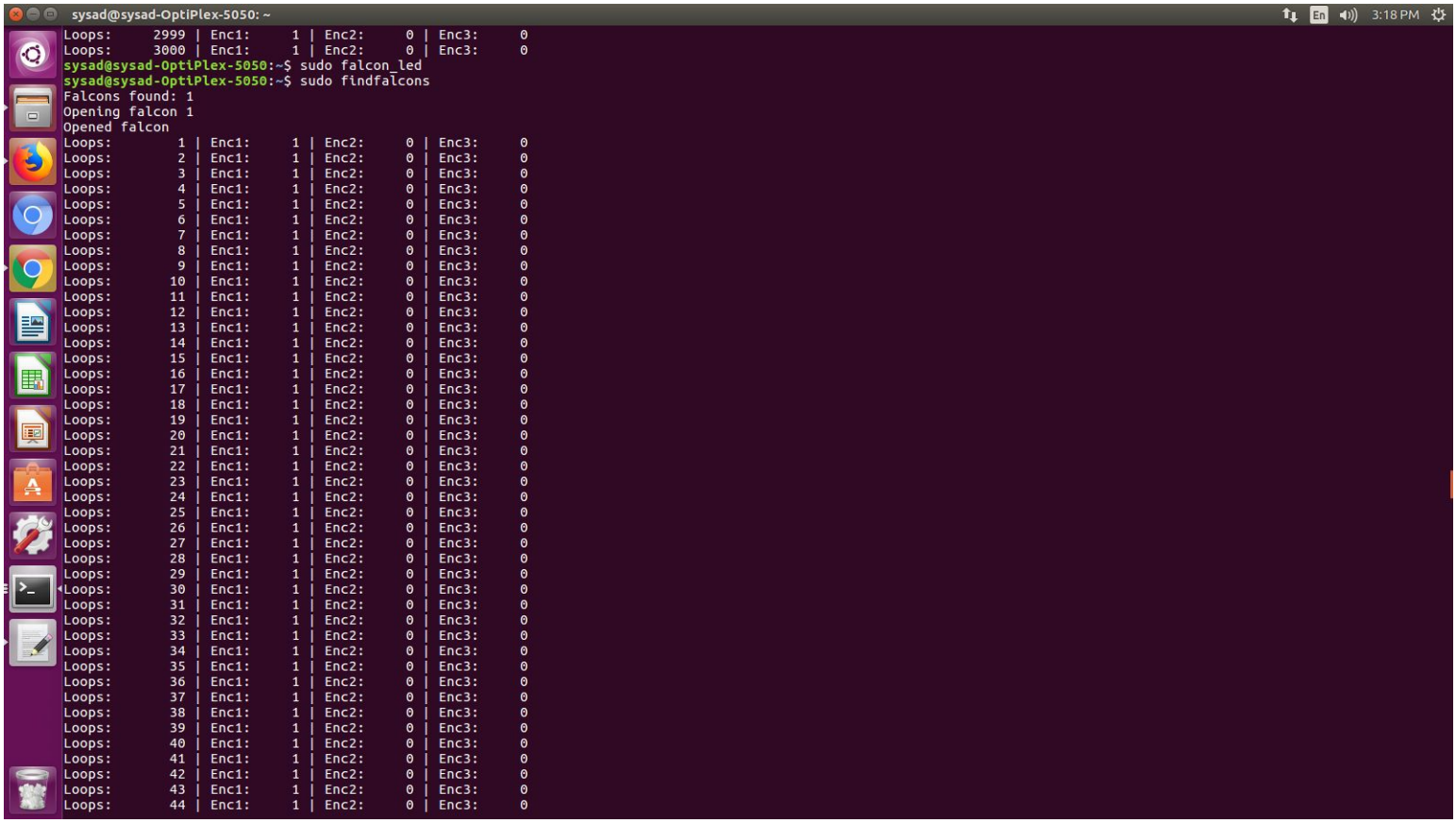
2. cmake it in build using the command below :

   $ cmake -G "Unix Makefiles" ..
   $ make
   $ make install
   (hint- if you get some error in make install,then use the command)
   $ sudo make install

➢ Run the following examples and test the Novint Falcon functionality:

$ sudo falcon_led

$ sudo findfalcons



[-Screenshot8_Findfalcons]

.....THESE STEPS MAKE SURE THAT LIBNIFALCON IS INSTALLED SUCCESSFULLY.....

---

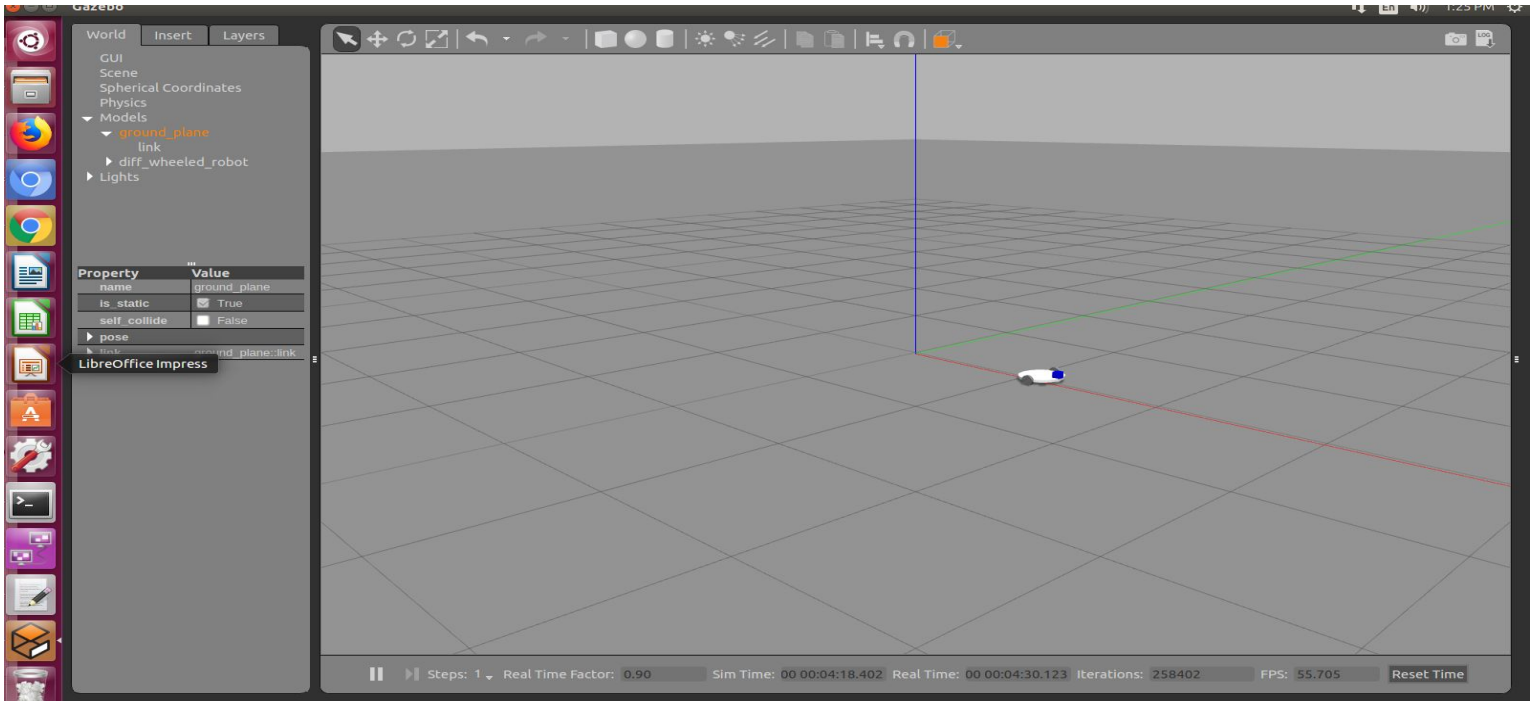
4. Controlling a ROS-Enabled Robot using Novint Falcon:

A. Two wheeled Robot

➢ Reference : ros_falcon package
https://github.com/WPI-AIM/ros_falcon.git
haptic_turtlesim
https://github.com/ben-greenberg/haptic_turtlesim.git

1. Launching the Two Wheeled Robot.

$ ***cd ~/catkin_ws_ars***
$ ***catkin_make***
$ ***source devel/setup.bash***
$ ***roslaunch ros_robotics diff_wheeled_gazebo_full.launch***



[-Screenshot9_TWR_launch]

- Files required: (**cd ~/catkin_ws_ars/src/ros_robotics**)
  - ❏ diff_wheeled_gazebo_full.launch
  - ❏ diff_wheeled_robot_with_sensor.xacro
  - ❏ wheel.urdf .xacro

2. Run the python joystick node.

- Files required: (cd ~/catkin_ws_ars/src/ros_falcon/nodes)
  - ❏ joy2cmd_vel_diff.py

$ ***cd ~/catkin_ws_ars***
$ ***catkin_make***
$ ***source devel/setup.bash***
$ ***roscd ros_falcon/nodes***
$ ***python joy2cmd_vel_diff.py***

3. Run ros_falcon joystick.

- Files required: ros_falcon

(cd ~/catkin_ws_ars/src/)

$ ***cd ~/catkin_ws_ars***
$ ***catkin_make***
$ ***source devel/setup.bash***

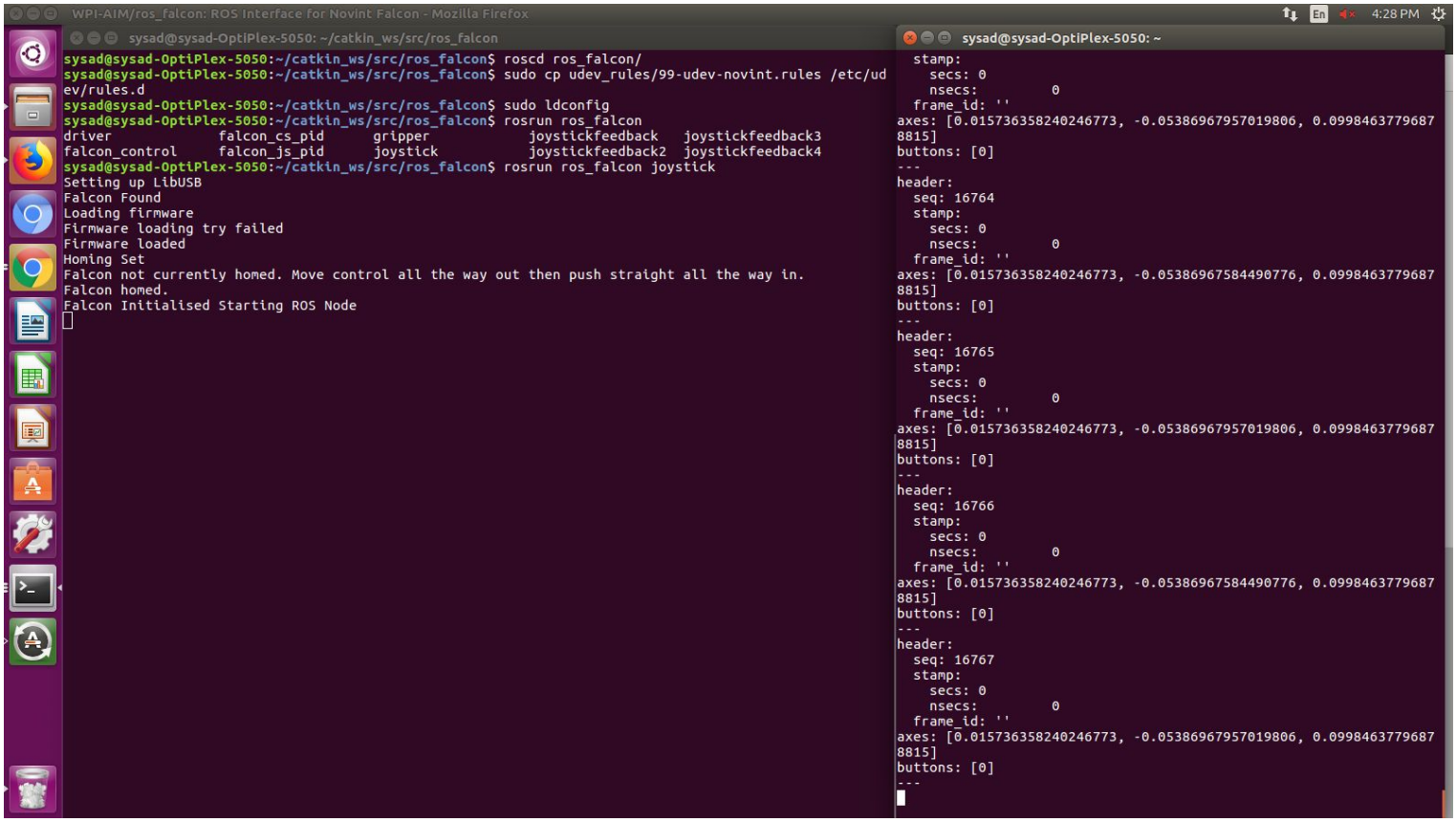
$ ***roscd ros_falcon/***

$ ***sudo cp udev_rules/99-udev-novint.rules /etc/udev/rules.d***

// unplug & replug the falcon //

$ ***sudo ldconfig***

$ ***rosrun ros_falcon joystick***

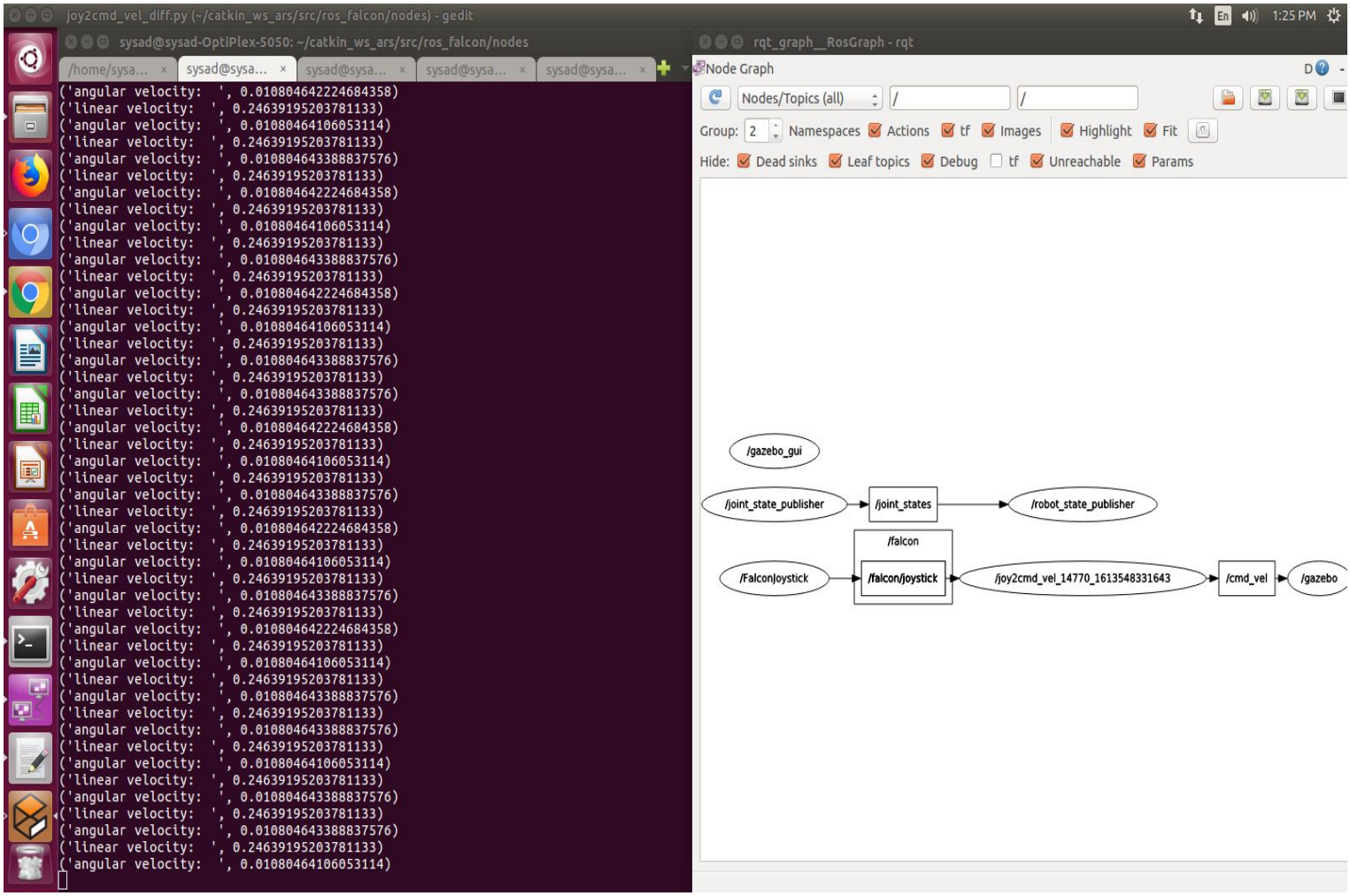


[-Screenshot10_joystick_axis_position]

4.Observe the robot motion and rqt graph

$ ***rqt_graph***

This graph makes sure about the various connections of ros-nodes.



[-Screenshot11_NodesConnection+falcon_joystick_data]

B. Controlling Floating box model using Novint falcon haptic device:

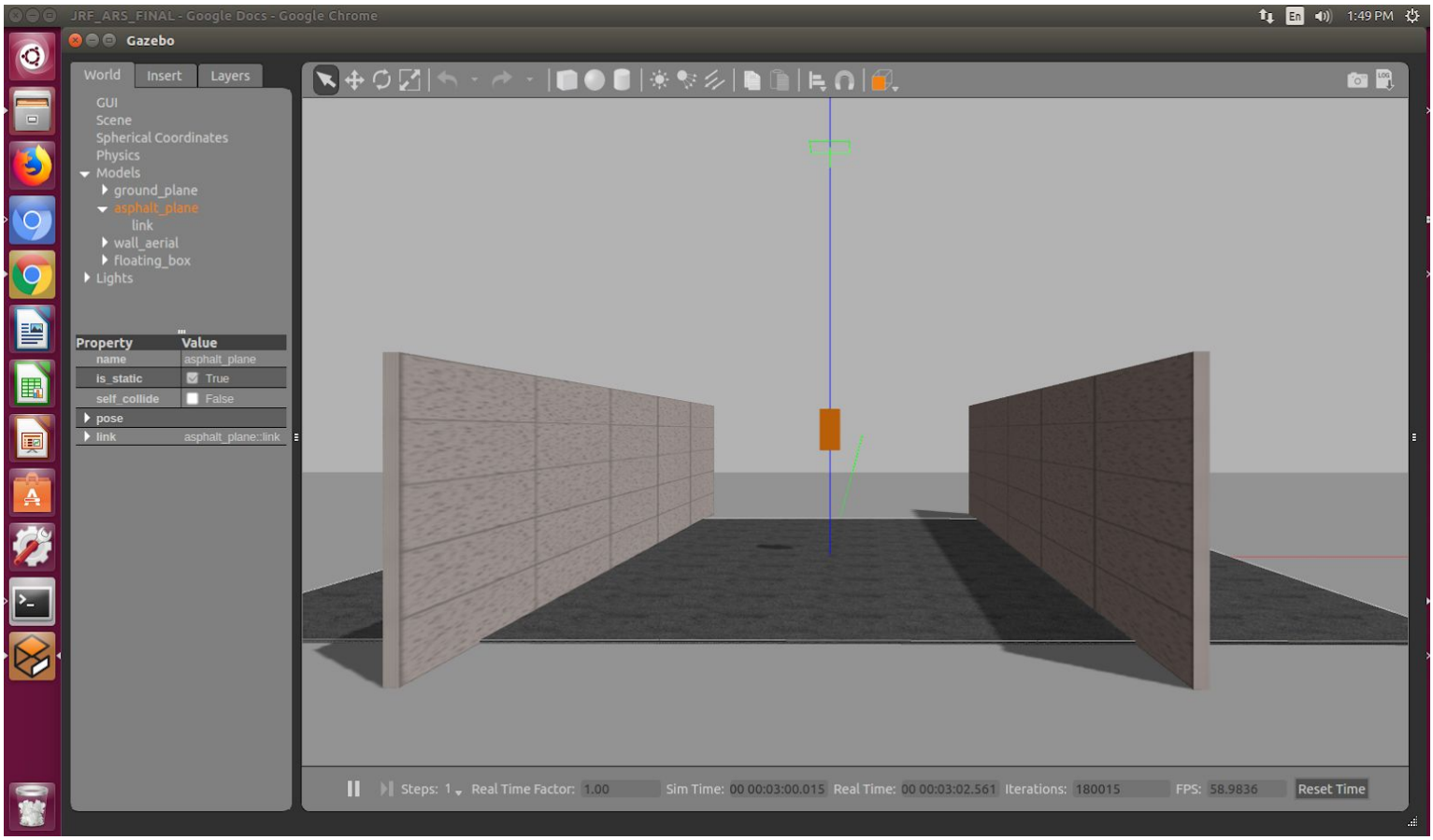
#---------Steps to control the floating box using Novint falcon------#

1. Creating Floating box (box without any force acting on it)

   $ ***cd ~/catkin_ws_ars***
   $ ***catkin_make***
   $ ***source devel/setup.bash***

   $ ***roslaunch ros_robotics box_gazebo.launch***

- Files required: (**cd ~/catkin_ws_ars/src/ros_robotics**)
  - ❏ box_gazebo.launch
  - ❏ floating_box_11.gazebo



[-Screenshot12_floating box]

2. Run the python joystick node.

- Files required: (cd ~/catkin_ws_ars/src/ros_falcon/nodes)

❑ joy2cmd_vel_box.py

$ cd ~/catkin_ws_ars
$ catkin_make
$ source devel/setup.bash
$ roscd ros_falcon/nodes

$ python joy2cmd_vel_box.py

3. Run ros_falcon joystick.

$ ***cd ~/catkin_ws_ars***
$ ***catkin_make***
$ ***source devel/setup.bash***

$ ***roscd ros_falcon/***

$ ***sudo cp udev_rules/99-udev-novint.rules /etc/udev/rules.d***

// unplug & replug the falcon //

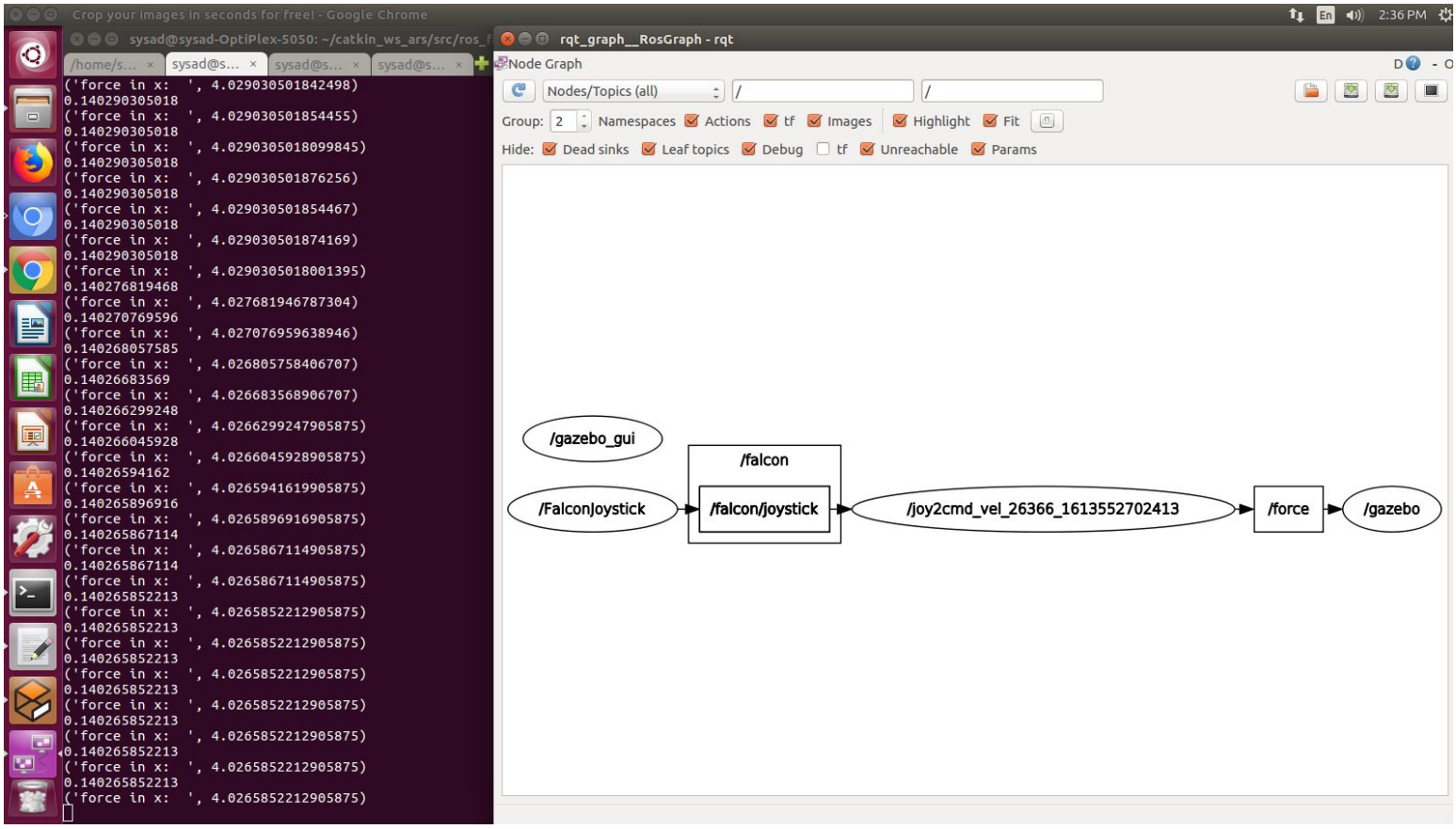
$ ***sudo ldconfig***

$ ***rosrun ros_falcon joystick***

★ TROUBLESHOOTING:
$ ***catkin_make***
$ ***source devel/setup.bash***

// unplug & replug the falcon //

$ ***sudo ldconfig***



[-Screenshot13_force]

5. Loco-positioning-system Configuration

Reference:
https://kth.instructure.com/courses/4962/pages/setting-up-the-loco-positioning-system#:~:text=Setting%20up%20the%20system&text=Restart%20the%20Crazyflie%20for%20the,button%20to%20configure%20the%20anchors

VIDEO: https://www.youtube.com/watch?v=ZgH4bLZdq2A

- Execute the following commands.

  $ git clone https://github.com/bitcraze/lps-node-firmware
  $ cd lps-node-firmware
  $ git submodule init
  $ git submodule update

You can accomplish the same things directly with
$ git clone --recursive https://github.com/bitcraze/lps-node-firmware

$ sudo apt-get install libncurses5:i386

- Upon restarting the shell, you should get the following when checking the version of your compiler using

$ arm-none-eabi-gcc --version

- Once you have the compiler installed, you should now be able to make the firmware by typing make in the ~/lps-node-firmware directory.
- To flash an anchor, plug it into the computer using a high-speed mUSB cable. Note that the cables used to charge the batteries and Crazyflie typically cannot be used, you will need a high-speed mUSB cable available in most labs and at eg Kjell and Company.

- We also need to ensure that the dfu-util package is installed

  $ sudo apt-get install dfu-util

  $ sudo apt-get install picocom

  $ sudo apt-get remove modemmanager

Now compiler is at: (**cd ~/lps-node-firmware**)

- Now execute the following commands

  $ make clean

  $ make

“ Now press and hold DFU toggle switch on the anchor and press reset then release the DFU button ”

  $ sudo make dfu

“ press reset button on anchor ”

  $ dmesg

  $ sudo picocom /dev/ttyACM3

➢ method1(TWR)

- Press 'a' to set the anchor mode to TWR. You can also set other modes by pressing 'm'. Another option would be TDOAv2, but for a small number of UAVs the TWR mode is best, since it typically gives larger available flying space.
- Type CTRL+a and then CTRL+q to exit picocom

➢ method2(TDoA)

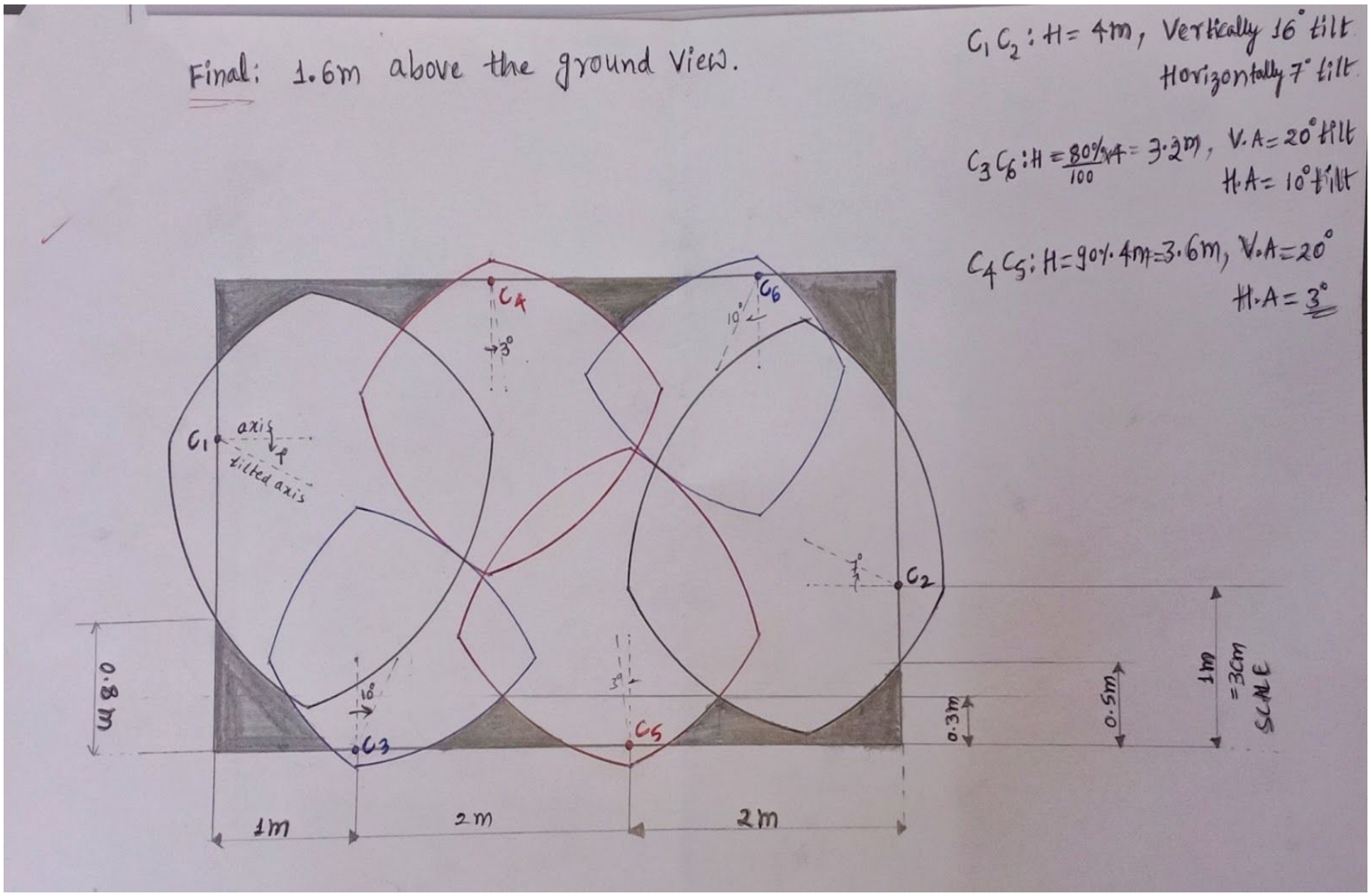
Press ‘m’ for TDoA, then press 3 for anchor node, then press ‘anchor number’

With this tool installed, repeat the following steps for each anchor to complete the configuration step.

- Mark the anchor physically with a unique integer **X** from 0-5 using the label maker
- Run `dmesg` (similar to the previous section) and find out which address the anchor is registered on (let's say ttyACM**Y**)
- Start picocom by ``picocom /dev/ttyACMY`` (if you did not add your user to dialout, you need to run this with sudo)
- Type h to see the available options
- Set the anchor to the address **X**
- Press 'a' to set the anchor mode to TWR. You can also set other modes by pressing 'm'. Another option would be TDOAv2, but for a small number of UAVs the TWR mode is best, since it typically gives larger available flying space.
- Type CTRL+a and then CTRL+q to exit picocom

Repeat this process for each and every anchor, until you have 6 anchors flashed with new firmware with addresses matching the labels on the anchors. You are now ready to set up the LPS-system and fly!

6. Design of Optimal positioning of flex camera



[-Screenshot14_DesignofOptimalPositiinofFlexCamera]