

Refer Git-Hub link given below for more information:

<https://github.com/abhishekr0001>

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:

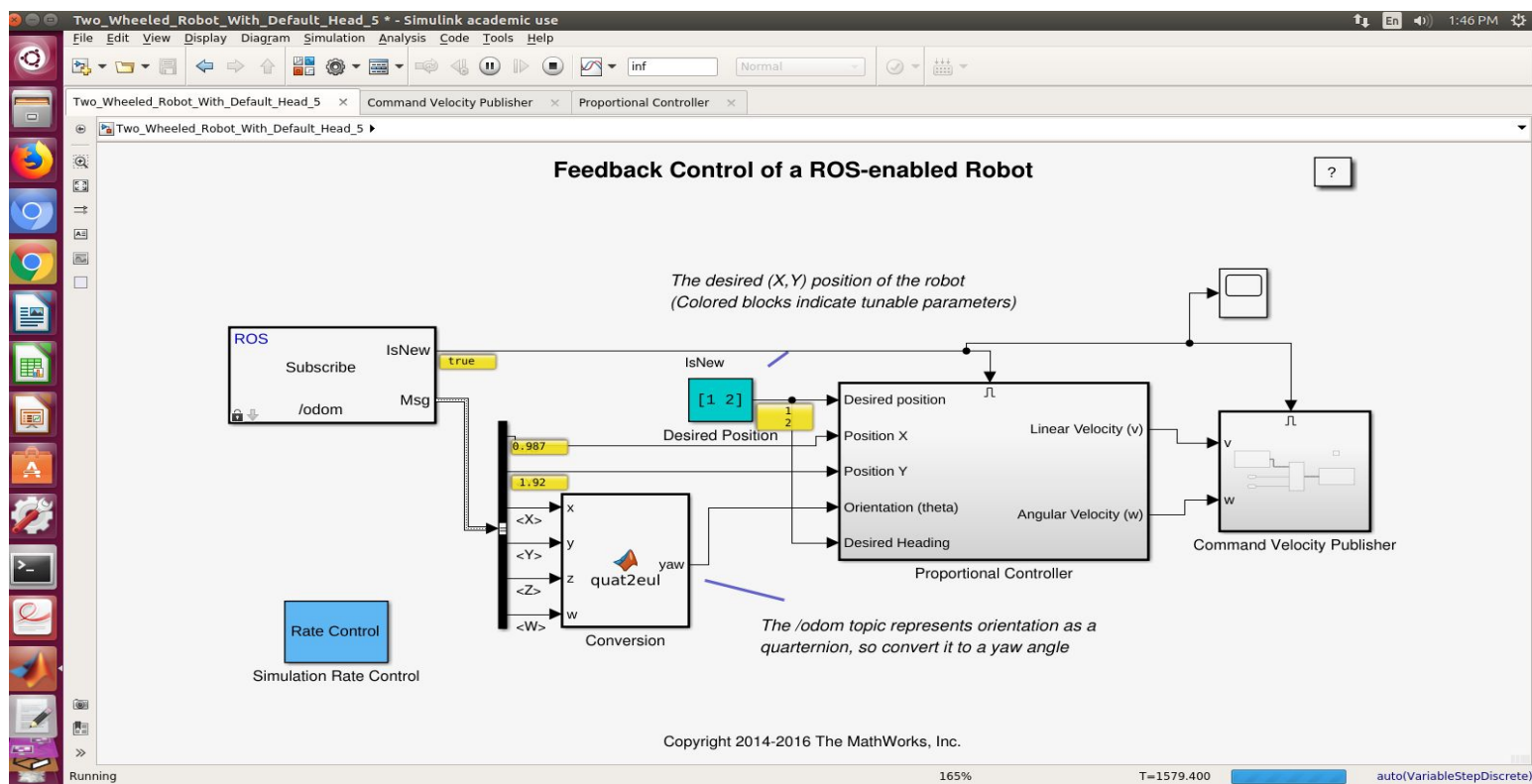
1. Execute following commands:

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

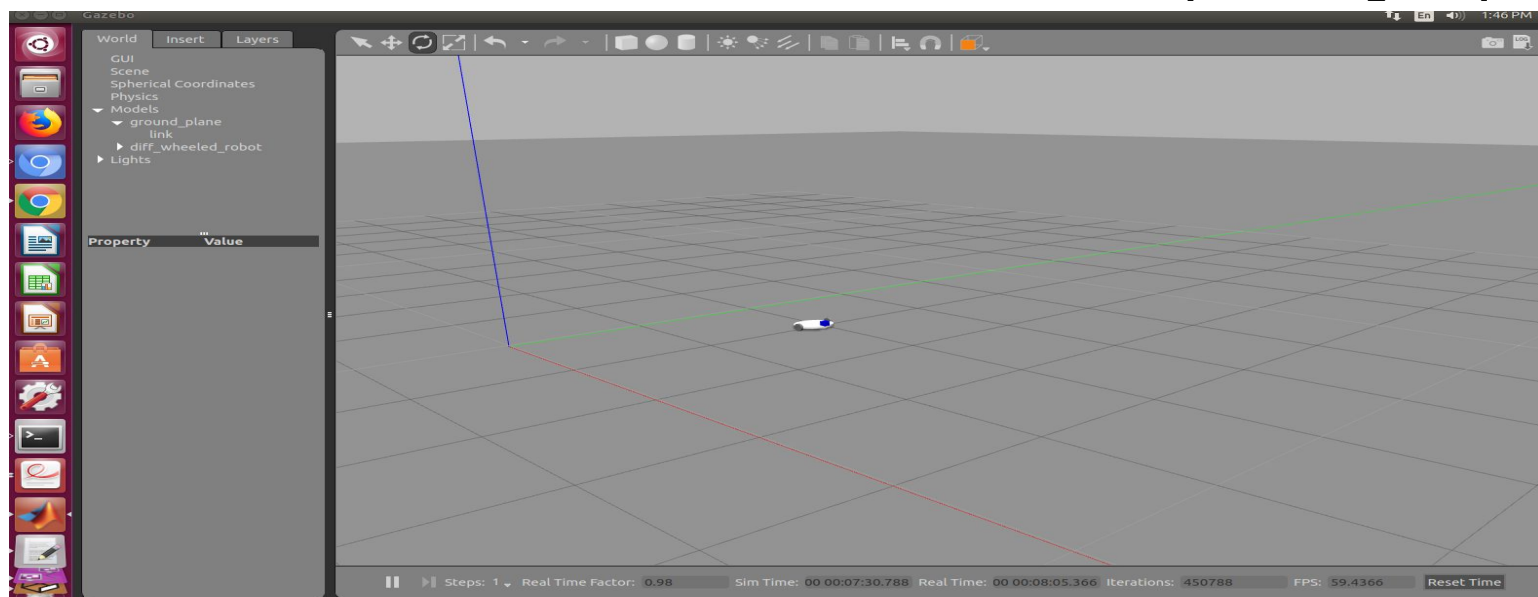
2. Check for topic name by executing following command.

- ***rostopic list***

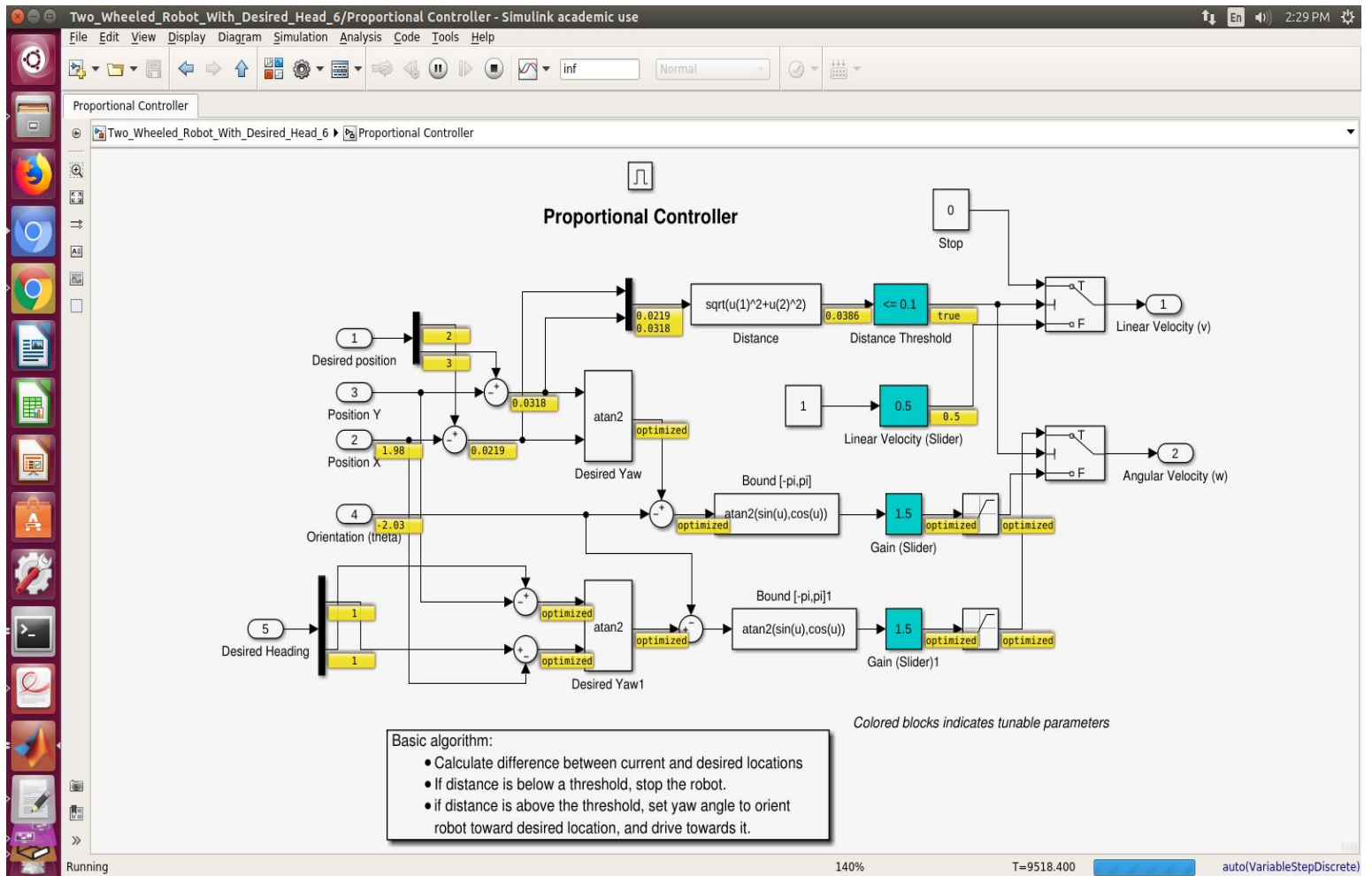
Use the same topic names in a simulink model.



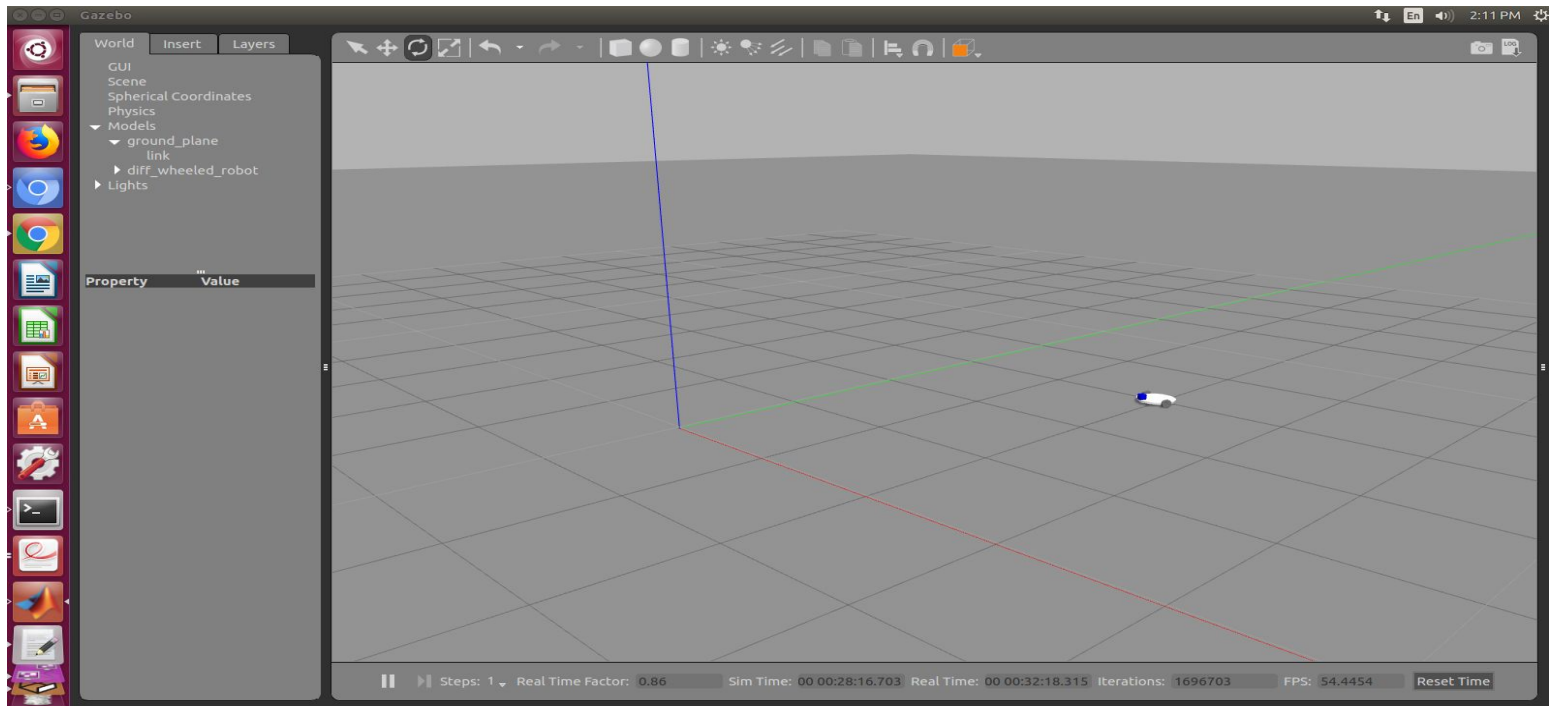
[-Screenshot1_Model1]



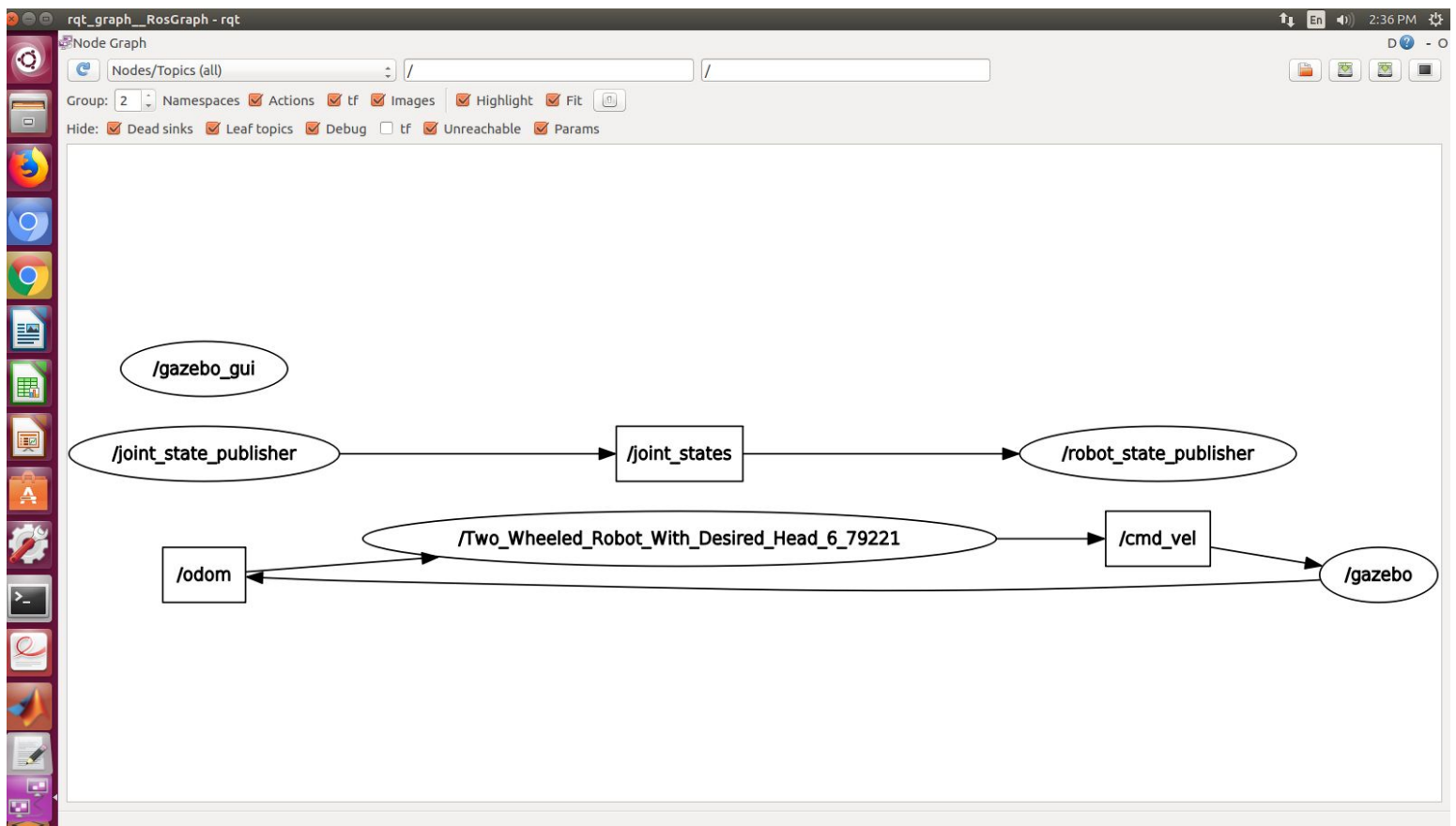
[-Screenshot2_model1]



[-Screenshot3_Proportional controller_ModelI2]



[-Screenshot4_Model2]

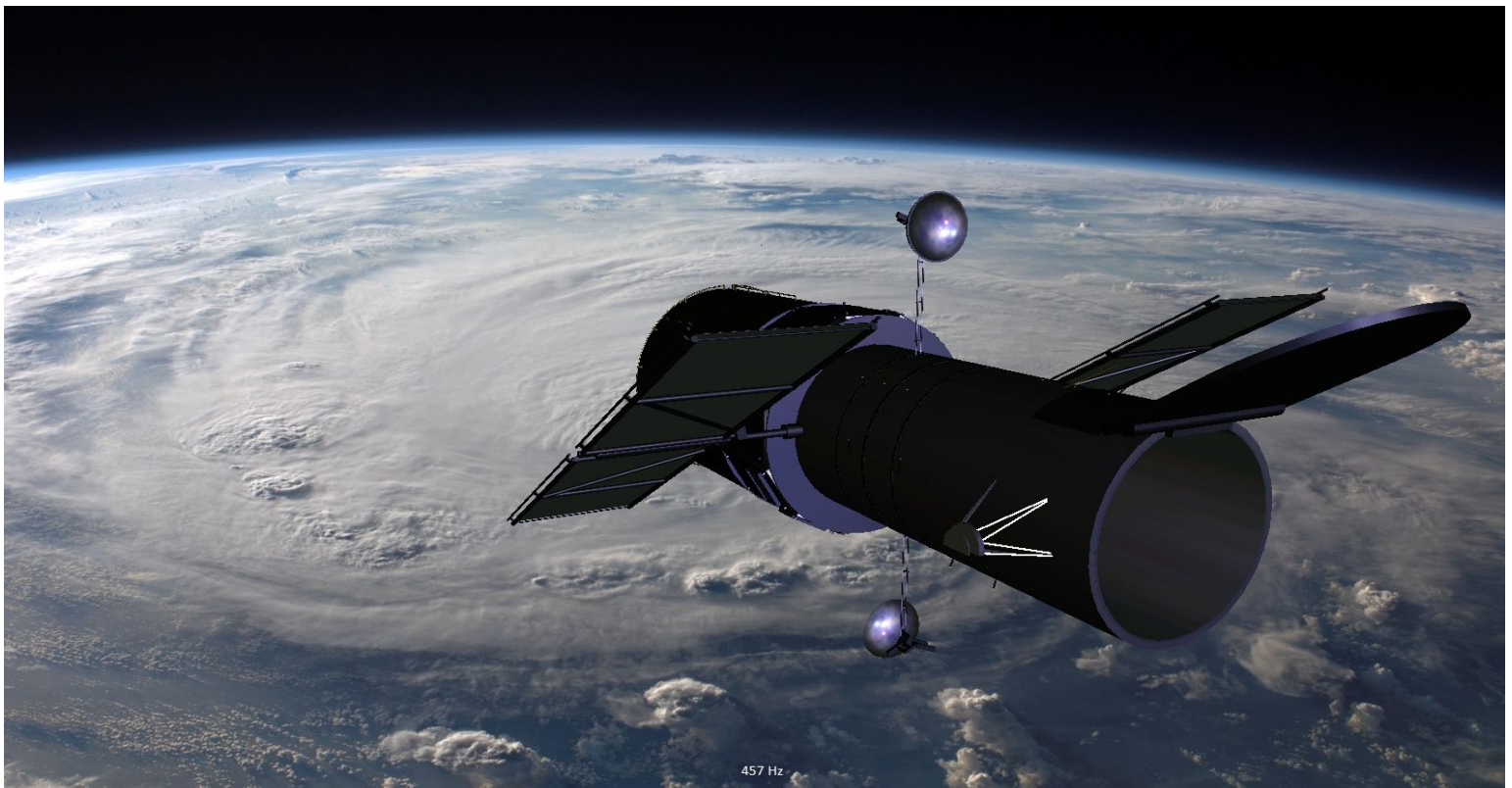


[-Screenshot5_rqtgraph_Model2]

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]


```
sysad@sysad-OptiPlex-5050: ~/catkin_ws_ars/src/2_chai3d-3.0.0/bin
make[3]: Leaving directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/examples/GLUT/27-multiframes'
make -C 28-voxel-basic
make[3]: Entering directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/examples/GLUT/28-voxel-basic'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/examples/GLUT/28-voxel-basic'
make -C 29-voxel-isosurface
make[3]: Entering directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/examples/GLUT/29-voxel-isosurface'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/examples/GLUT/29-voxel-isosurface'
make -C 30-voxel-colormap
make[3]: Entering directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/examples/GLUT/30-voxel-colormap'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/examples/GLUT/30-voxel-colormap'
make -C 31-pointcloud
make[3]: Entering directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/examples/GLUT/31-pointcloud'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/examples/GLUT/31-pointcloud'
make[2]: Leaving directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/examples/GLUT'
make[1]: Leaving directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/examples'
make -C utils
make[1]: Entering directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/utils'
make -C CLI
make[2]: Entering directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/utils/CLI'
make -C cfont
make[3]: Entering directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/utils/CLI/cfont'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/utils/CLI/cfont'
make -C cimage
make[3]: Entering directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/utils/CLI/cimage'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/utils/CLI/cimage'
make[2]: Leaving directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/utils/CLI'
make[1]: Leaving directory '/home/sysad/catkin_ws_ars/src/2_chai3d-3.0.0/utils'
sysad@sysad-OptiPlex-5050:~/catkin_ws_ars/src/2_chai3d-3.0.0$ cd bin/
sysad@sysad-OptiPlex-5050:~/catkin_ws_ars/src/2_chai3d-3.0.0/bin$ ./19-space

-----
CHAI3D
Demo: 19-space
Copyright 2003-2015
-----

Keyboard Options:
[f] - Enable/Disable full screen mode
[m] - Enable/Disable vertical mirroring
[X] - Exit application

█
```

[-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
```

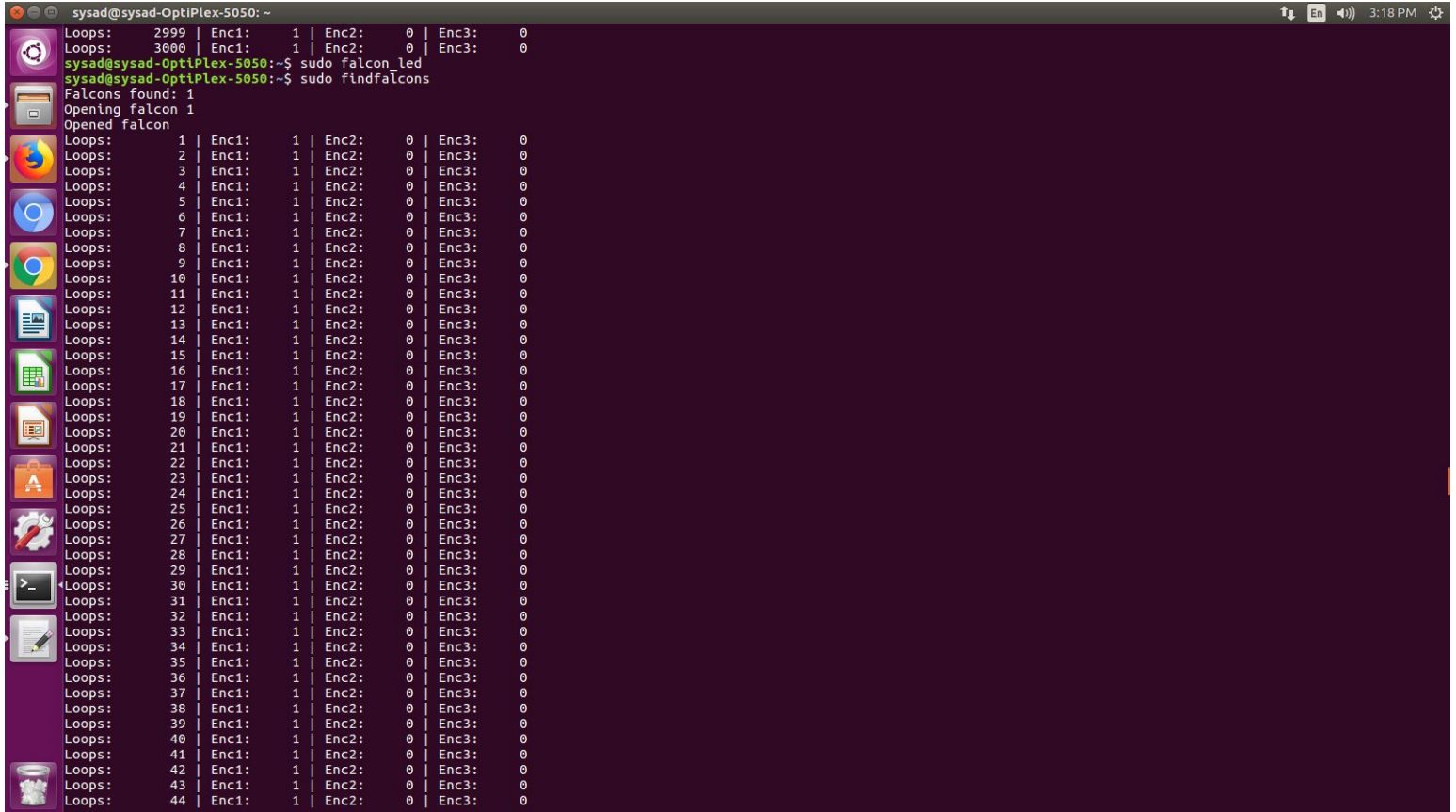
(hint- if you get some error in make install, then use the command)

```
$ sudo make install
```


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

\$ sudo falcon_led

\$ sudo findfalcons



```
sysad@sysad-OptiPlex-5050: ~  
Loops: 2999 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 3000 | Enc1: 1 | Enc2: 0 | Enc3: 0  
sysad@sysad-OptiPlex-5050:~$ sudo falcon_led  
sysad@sysad-OptiPlex-5050:~$ sudo findfalcons  
Falcons found: 1  
Opening falcon 1  
Opened falcon  
Loops: 1 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 2 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 3 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 4 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 5 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 6 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 7 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 8 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 9 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 10 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 11 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 12 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 13 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 14 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 15 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 16 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 17 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 18 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 19 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 20 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 21 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 22 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 23 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 24 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 25 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 26 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 27 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 28 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 29 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 30 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 31 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 32 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 33 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 34 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 35 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 36 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 37 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 38 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 39 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 40 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 41 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 42 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 43 | Enc1: 1 | Enc2: 0 | Enc3: 0  
Loops: 44 | Enc1: 1 | Enc2: 0 | Enc3: 0
```

[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

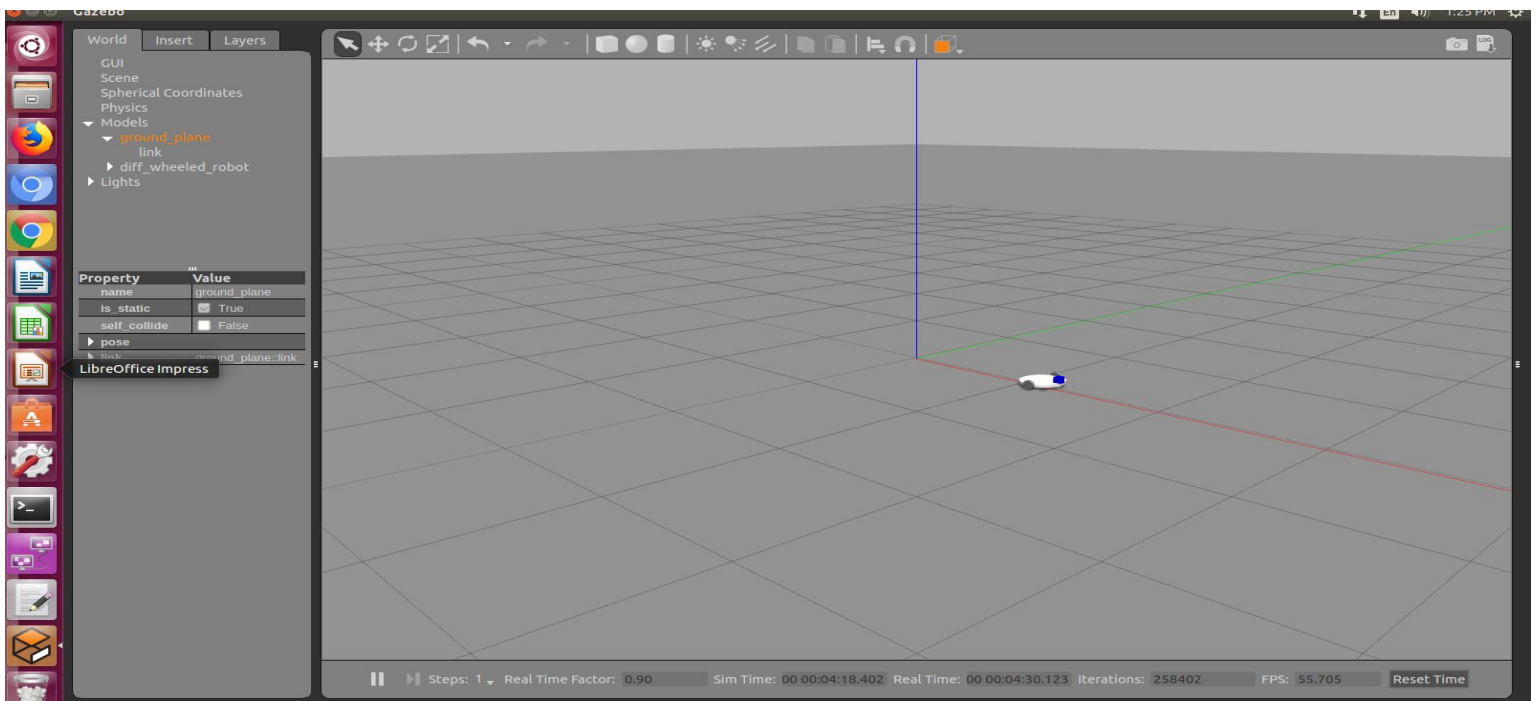
1. Launching the Two Wheeled Robot.

```
$ 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
```

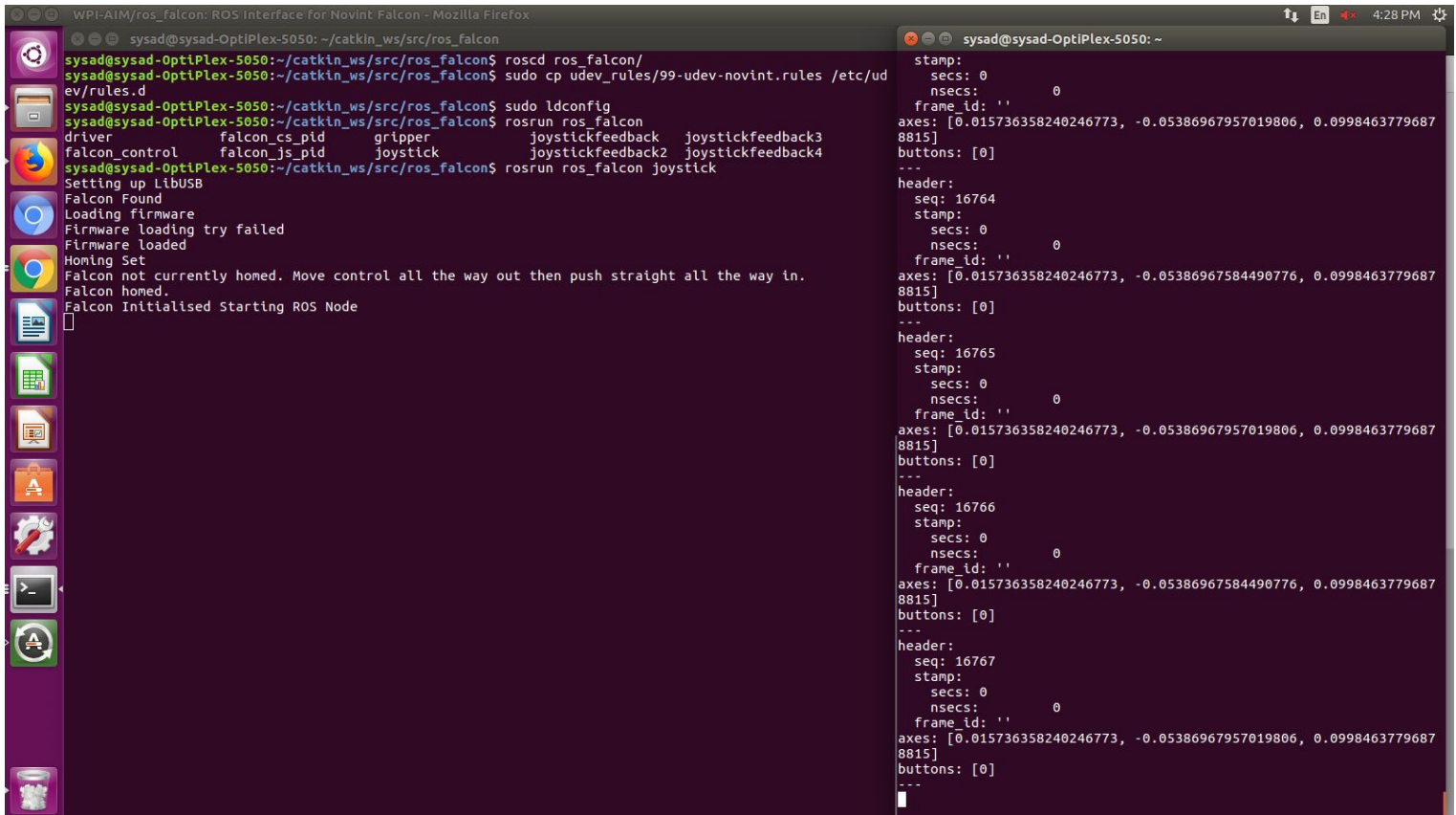
```
$ roscd ros_falcon/
```

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

```
// unplug & replug the falcon //
```

```
$ sudo ldconfig
```

\$ *roslaunch ros_falcon joystick*



```
WPI-AIM/ros_falcon: ROS Interface for Novint Falcon - Mozilla Firefox
sysad@sysad-OptiPlex-5050: ~/catkin_ws/src/ros_falcon
sysad@sysad-OptiPlex-5050:~/catkin_ws/src/ros_falcon$ roscd ros_falcon/
sysad@sysad-OptiPlex-5050:~/catkin_ws/src/ros_falcon$ sudo cp udev_rules/99-udev-novint.rules /etc/udev/rules.d
sysad@sysad-OptiPlex-5050:~/catkin_ws/src/ros_falcon$ sudo ldconfig
sysad@sysad-OptiPlex-5050:~/catkin_ws/src/ros_falcon$ roslaunch ros_falcon
driver          falcon_cs_pid    gripper          joystickfeedback joystickfeedback3
falcon_control  falcon_js_pid    joystick         joystickfeedback2 joystickfeedback4
sysad@sysad-OptiPlex-5050:~/catkin_ws/src/ros_falcon$ roslaunch ros_falcon joystick
Setting up LibUSB
Falcon Found
Loading firmware
Firmware loading try failed
Firmware loaded
Homing Set
Falcon not currently homed. Move control all the way out then push straight all the way in.
Falcon homed.
Falcon Initialised Starting ROS Node

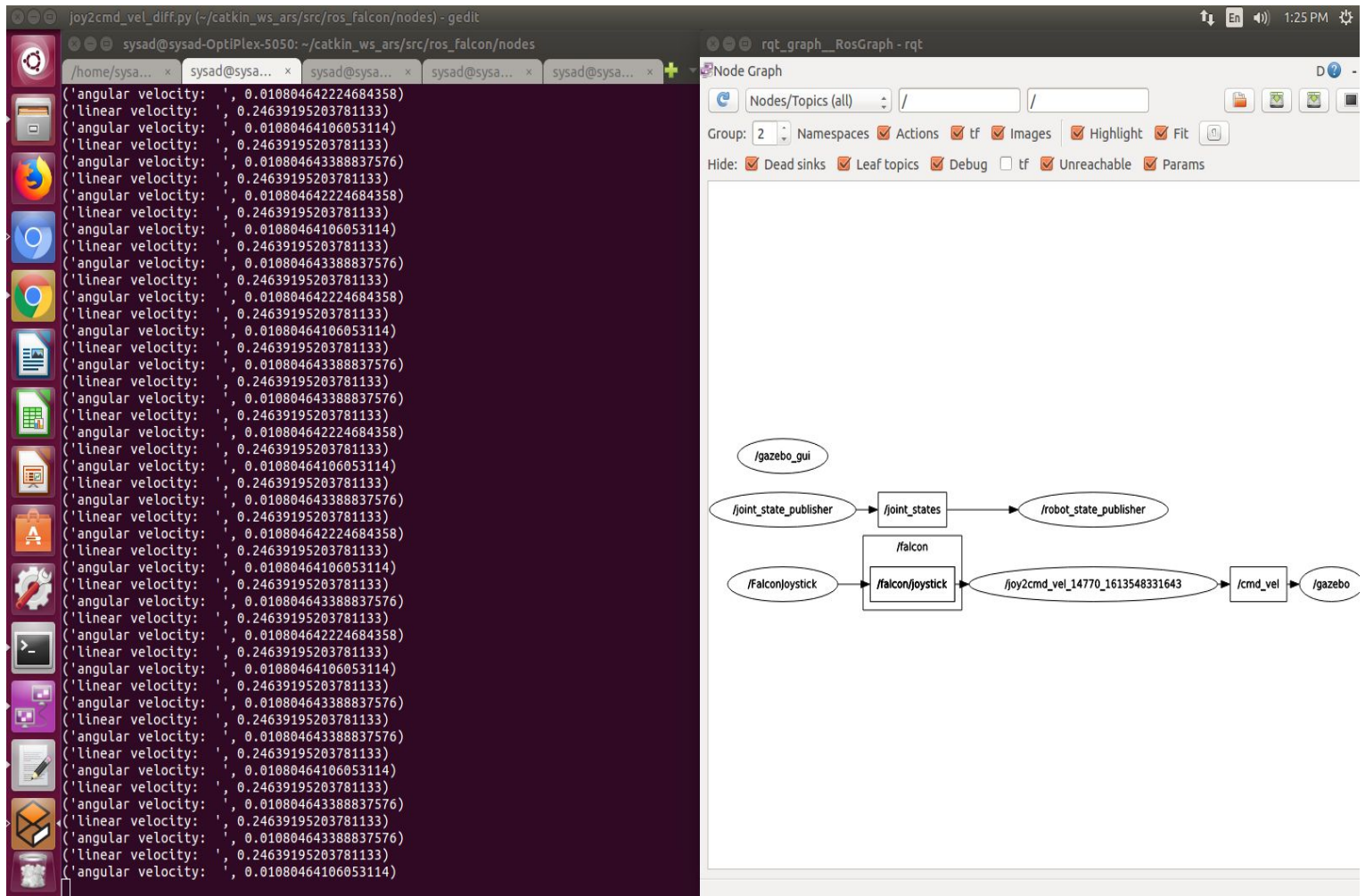
stamp:
  secs: 0
  nsecs: 0
frame_id: ''
axes: [0.015736358240246773, -0.05386967957019806, 0.09984637796878815]
buttons: [0]
---
header:
  seq: 16764
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
axes: [0.015736358240246773, -0.05386967584490776, 0.09984637796878815]
buttons: [0]
---
header:
  seq: 16765
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
axes: [0.015736358240246773, -0.05386967957019806, 0.09984637796878815]
buttons: [0]
---
header:
  seq: 16766
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
axes: [0.015736358240246773, -0.05386967584490776, 0.09984637796878815]
buttons: [0]
---
header:
  seq: 16767
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
axes: [0.015736358240246773, -0.05386967957019806, 0.09984637796878815]
buttons: [0]
---
```

[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)

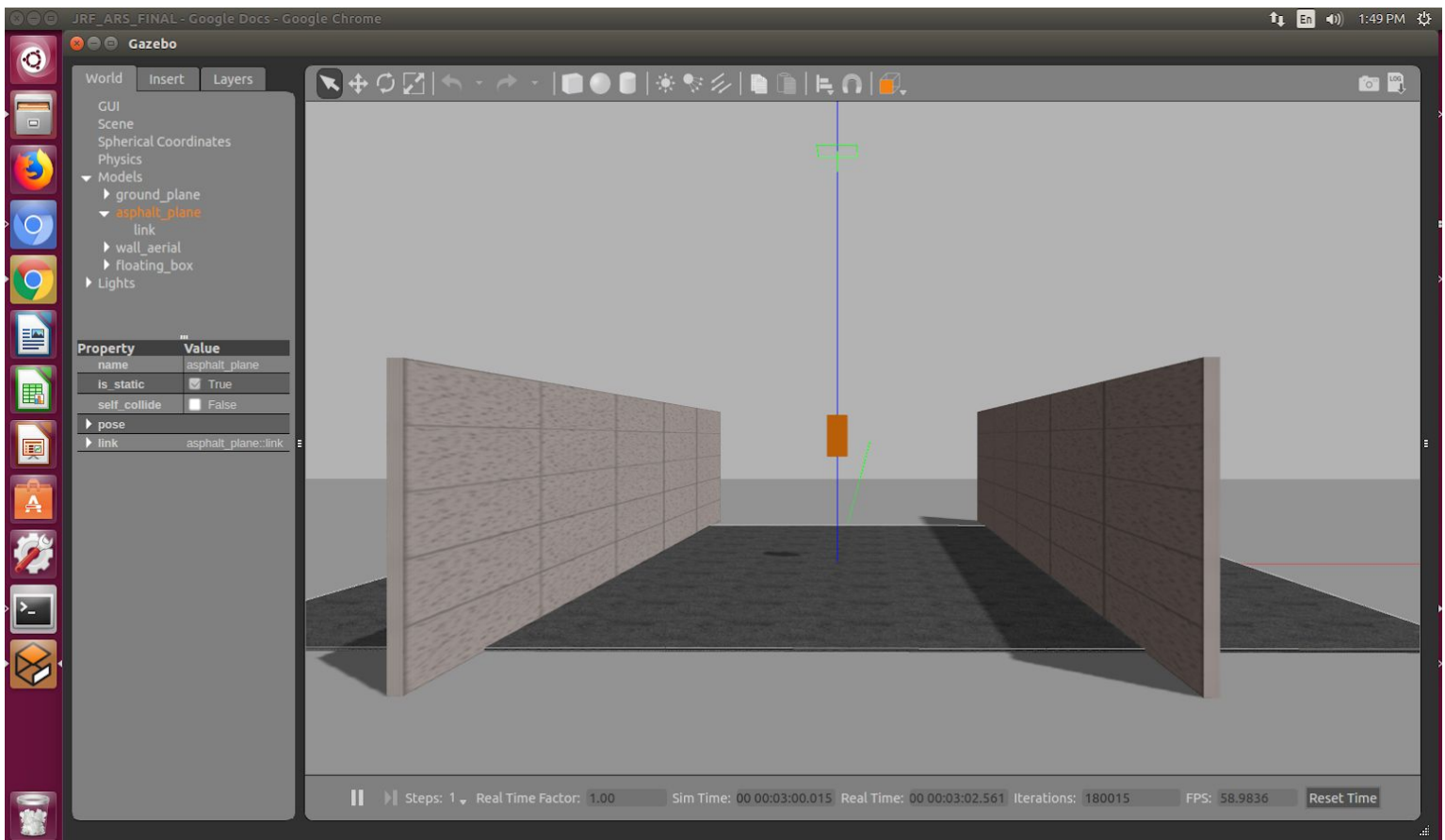
```
$ cd ~/catkin_ws_ars
```

```
$ 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
```

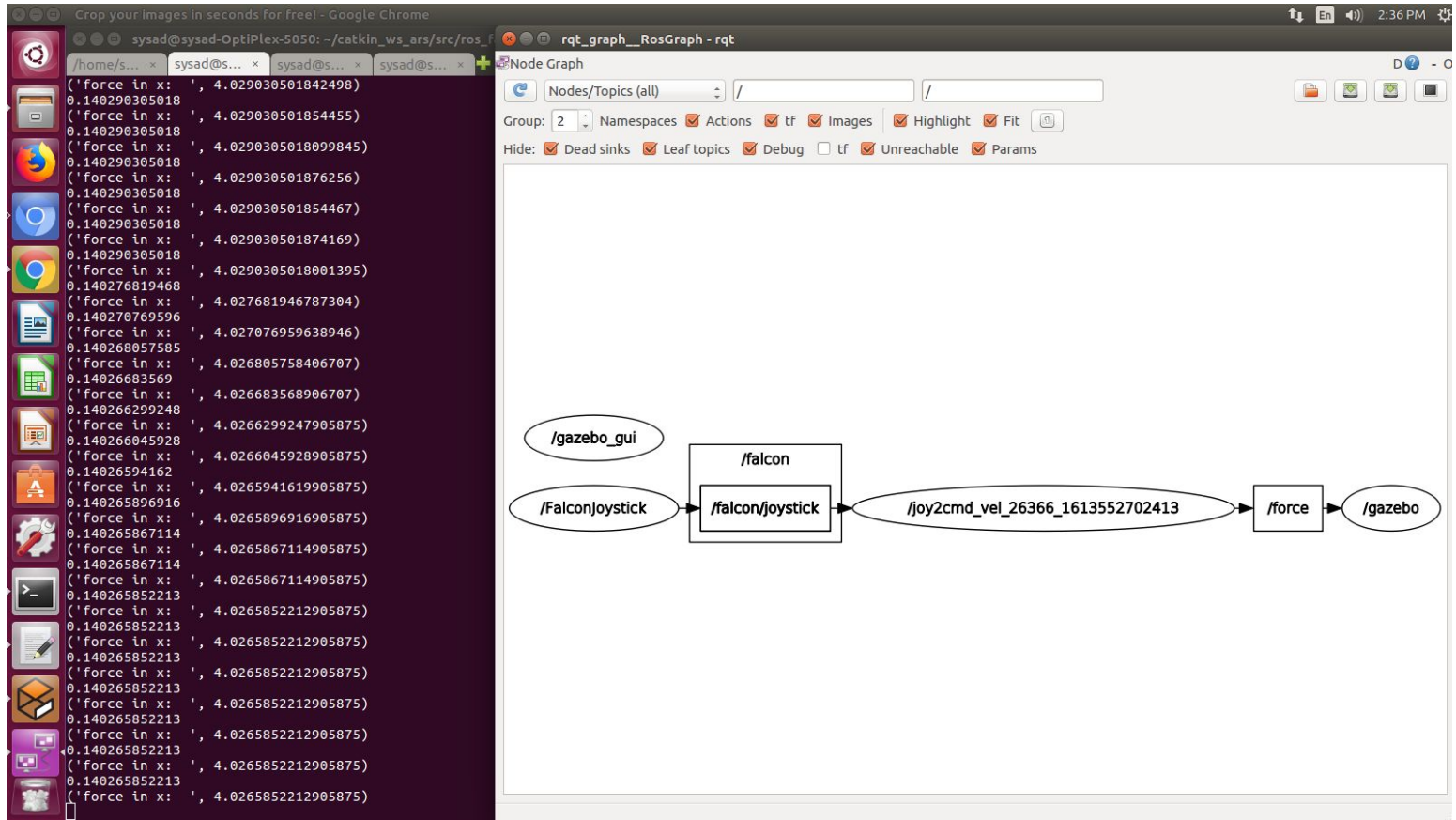
3. Run ros_falcon joystick.

```
$ 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  
  
$ rosrn 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)

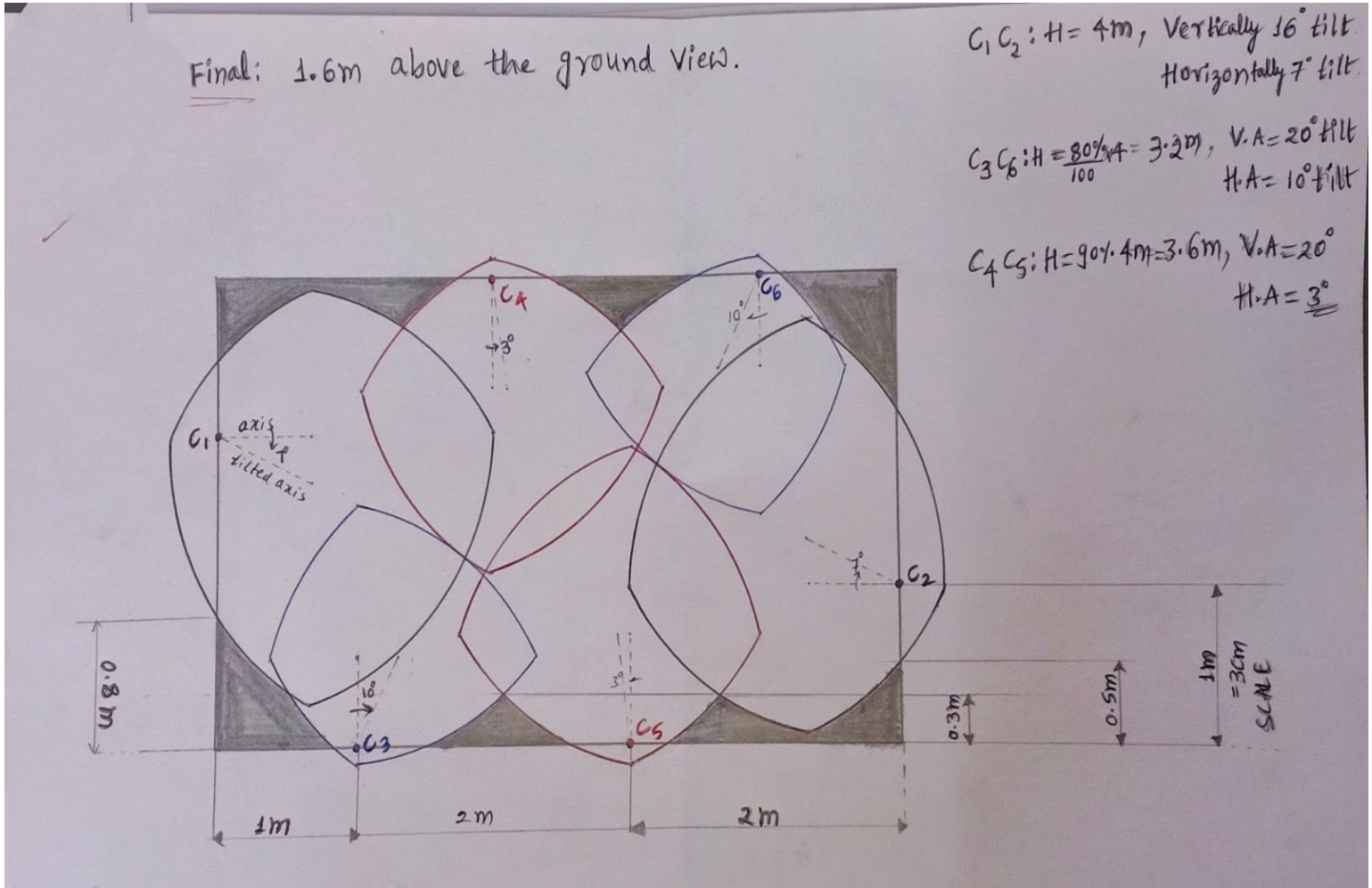
Press 'm' for TDoA, then press 3 for anchor node, then press 'anchor number'

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]

