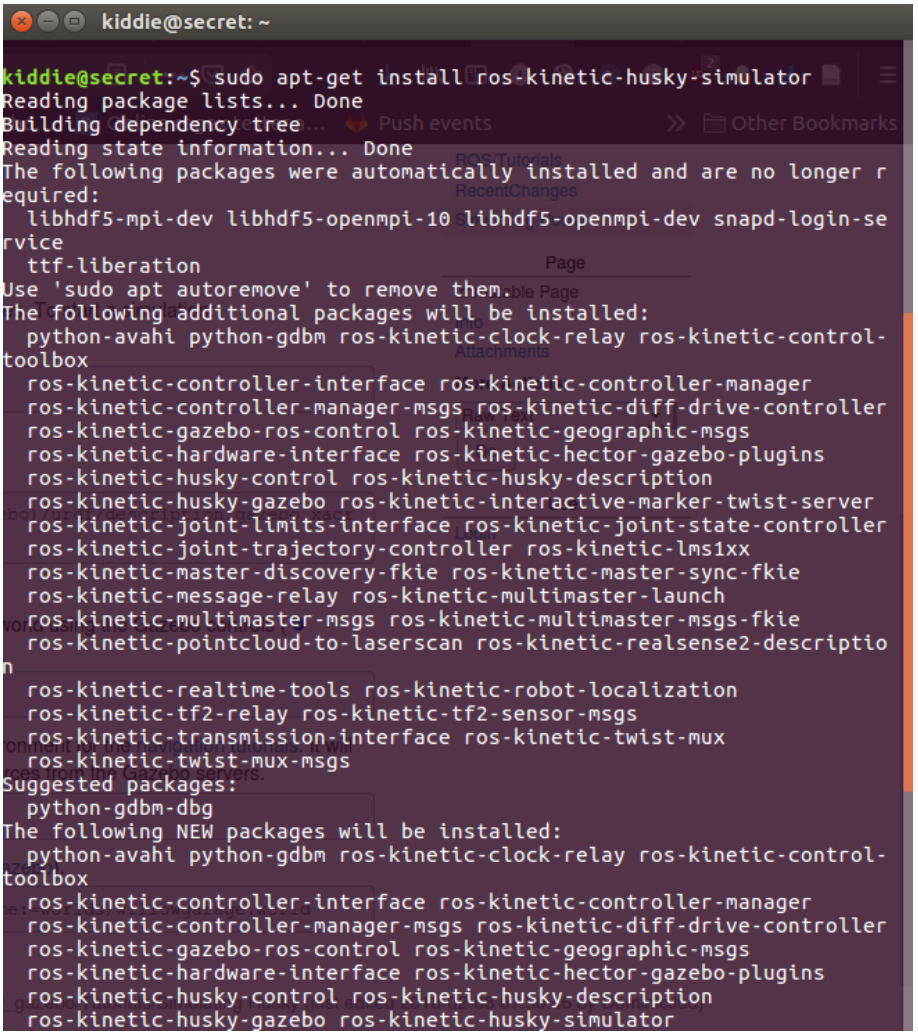
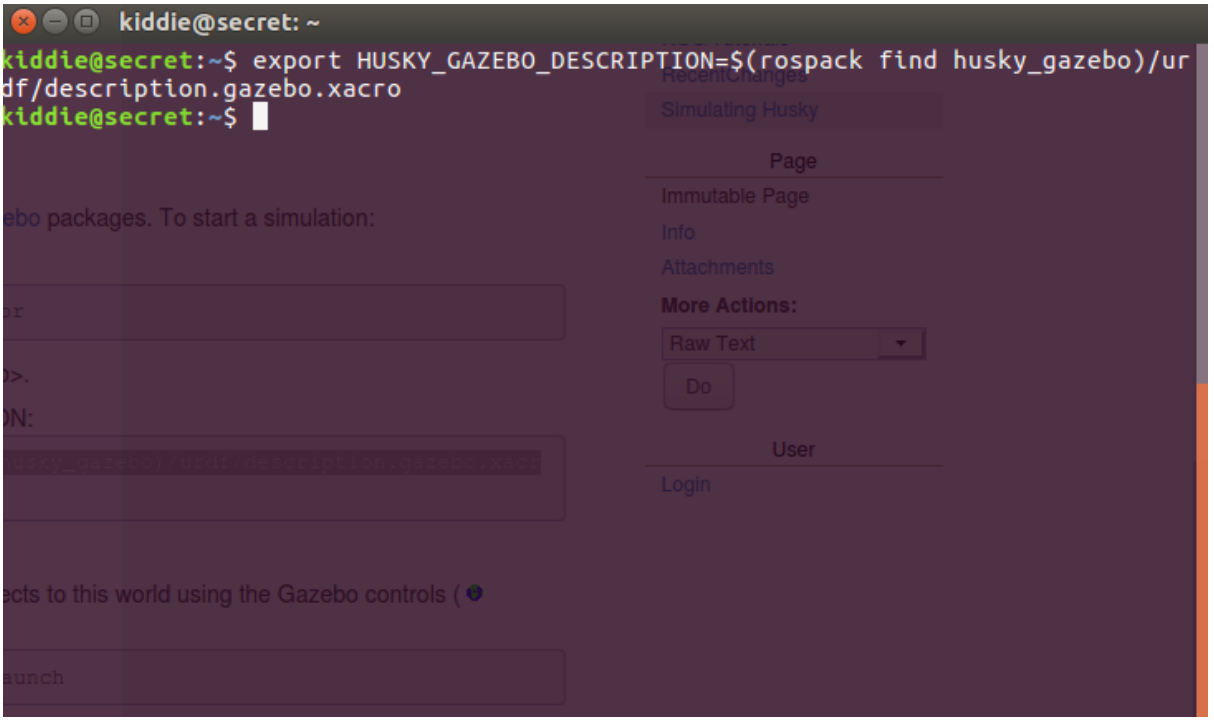
Exercise 1

1. Install husky simulator and set environment
   install husky with command
   sudo apt-get install ros-kinetic-husky-simulator



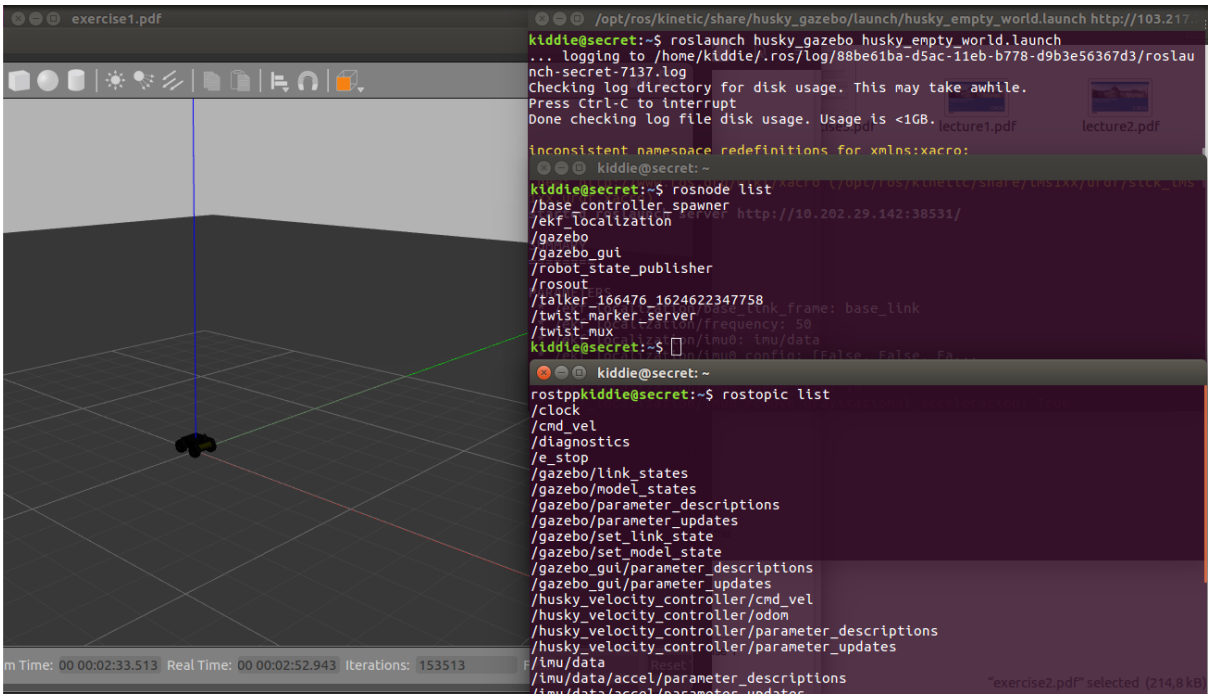
   set environment with command
   export HUSKY_GAZEBO_DESCRIPTION=$(rospack find husky_gazebo)/urdf/description.gazebo.xacro

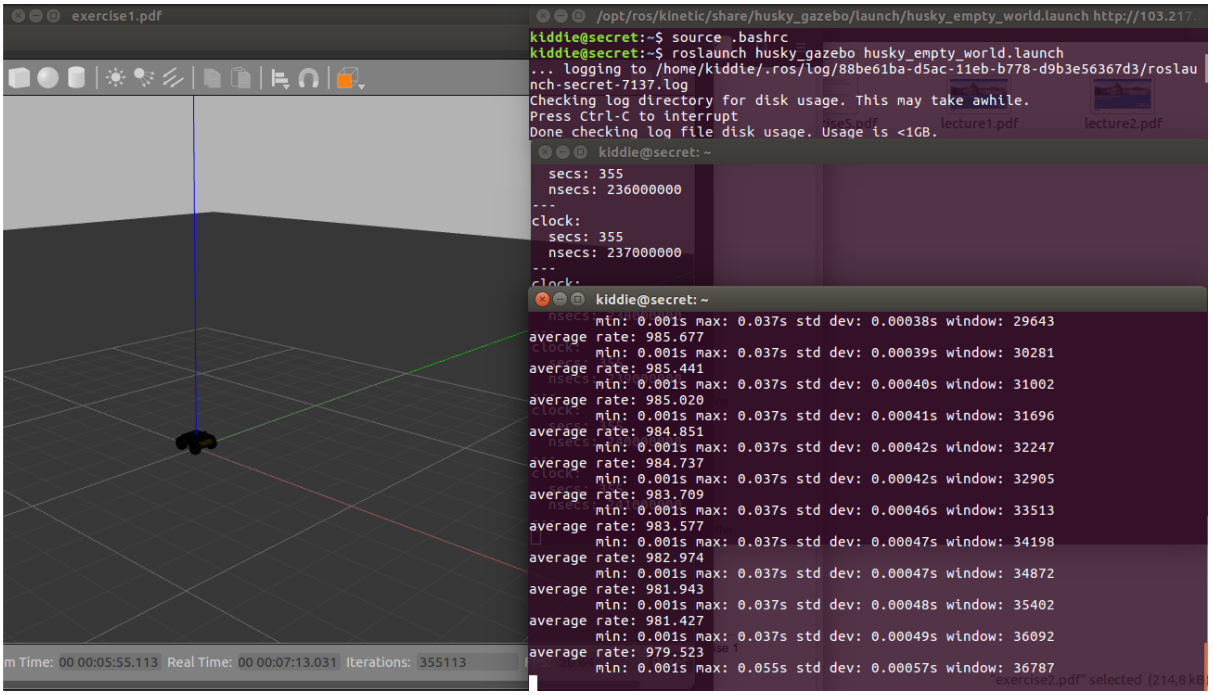


and final step run roslaunch husky_gazebo husky_empty_world.launch to see husky simulator

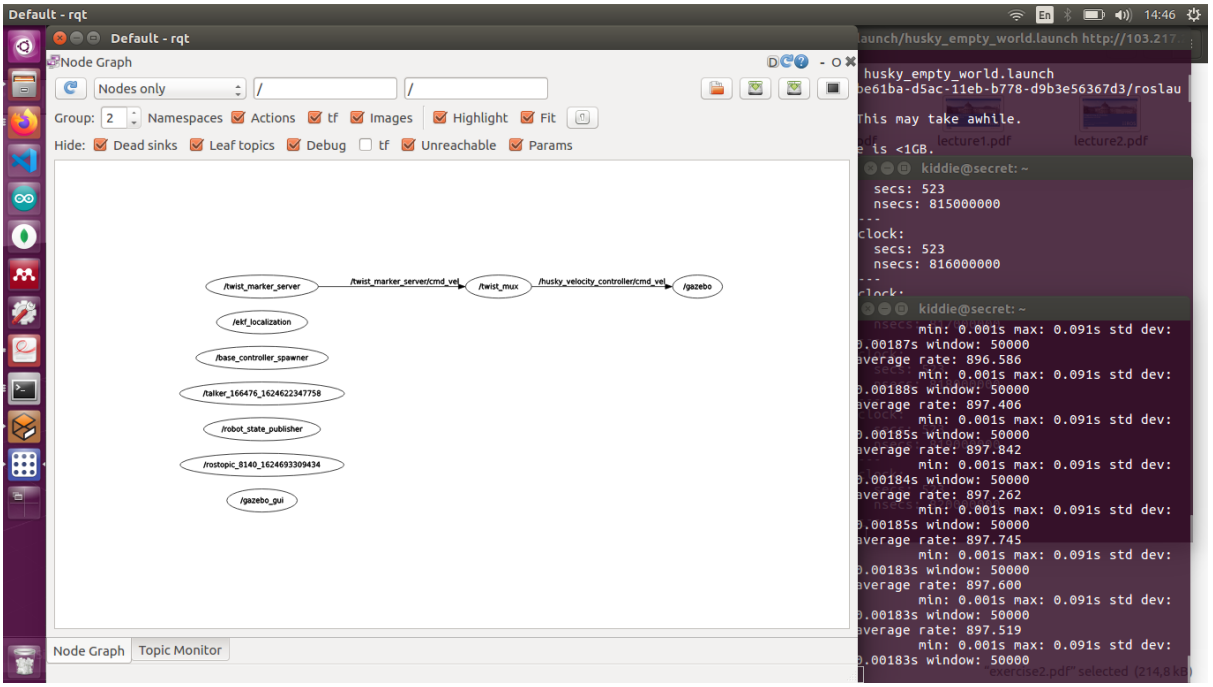
2.

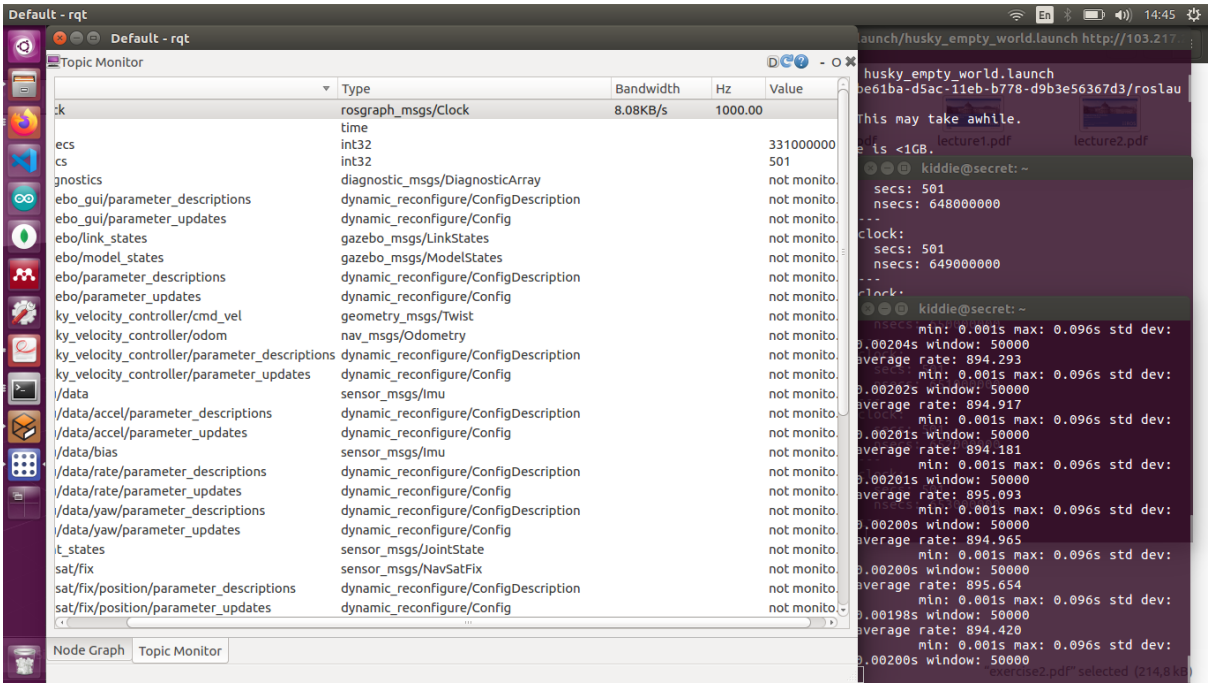
rostopic and rosnode while husky simulator has started



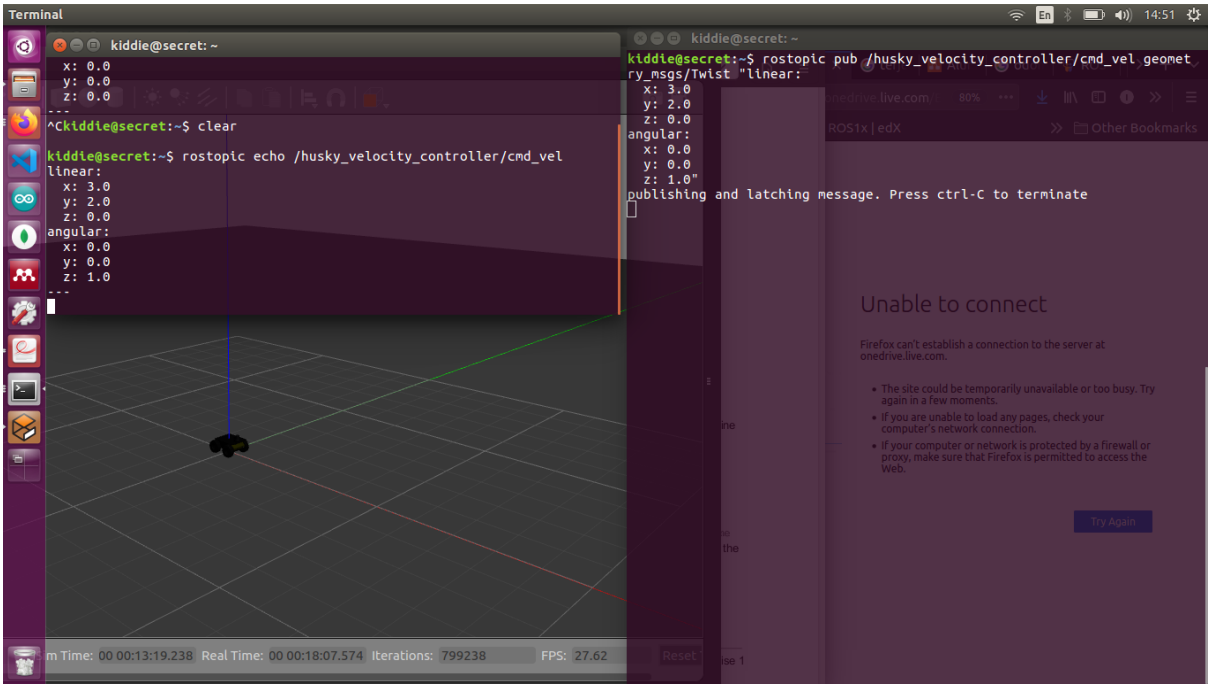


to see node that run on husky simulator





3. publish data that set first

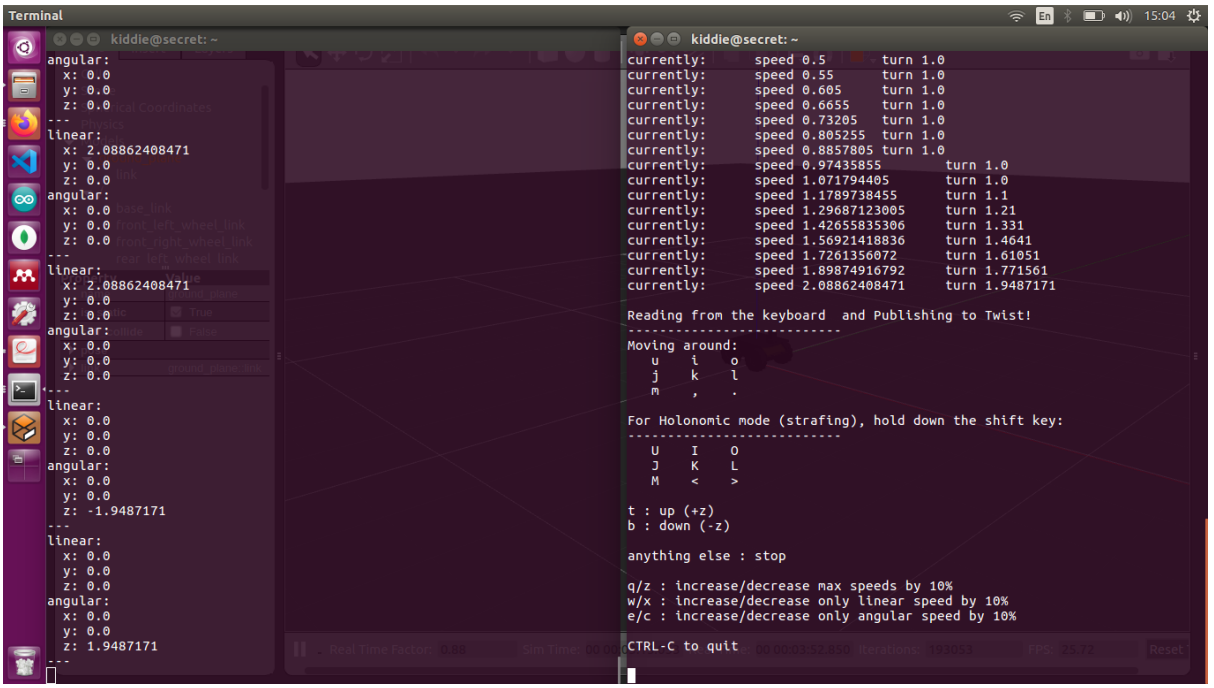


4.

1. git clone https://github.com/ros-teleop/teleop_twist_keyboard.git to ros workspace
   or sudo apt-get install ros-<distro ros>-teleop-twist-keyboard
2. open directory ros workspace
   cd ~/catkin_ws
3. build teleop-twist-keyboard

catkin build teleop_twist_keyboard

4. run source devel/setup.bash
5. run teleop-twist-keyboard with rosrun

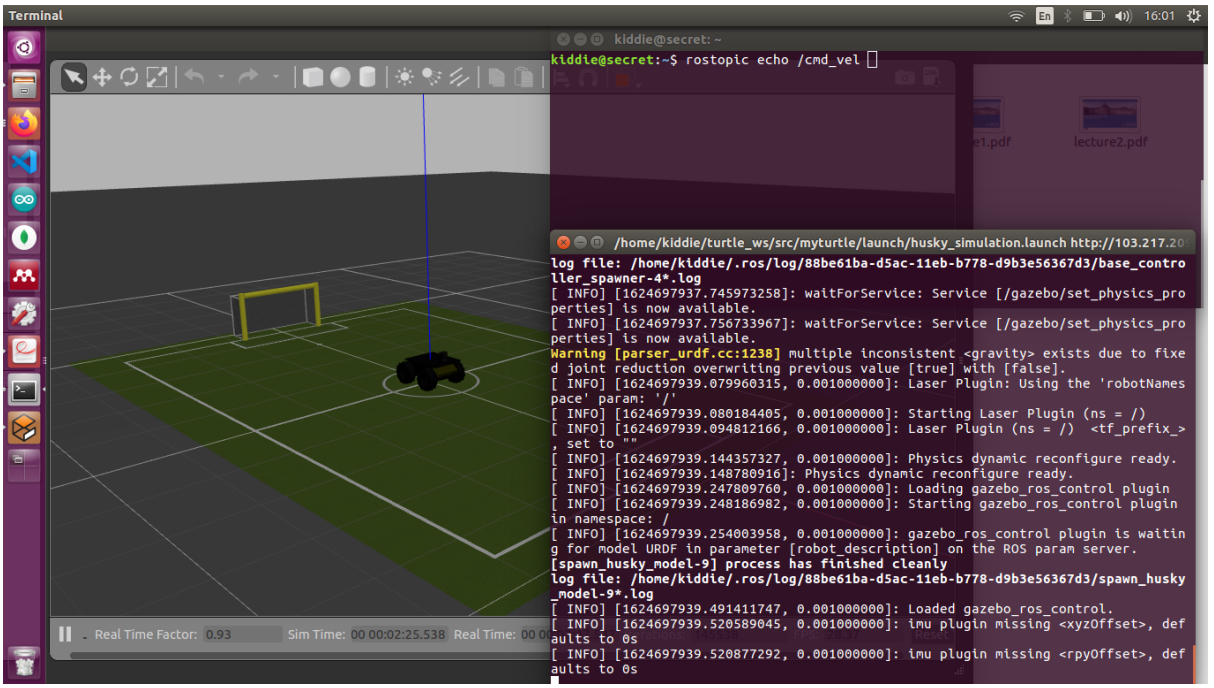


5. make launch file to control husky robot
   1. create file with extension .launch in launch directory
   2. write this code

```
<?xml version="1.0"?>

<!--

-->

<launch>

 <arg name="world_name"
default="/usr/share/gazebo-7/worlds/robocup14_spl_field.wo
rld"/>

 <arg name="laser_enabled" default="true"/>

 <arg name="kinect_enabled" default="false"/>

 <include file="$(find
gazebo_ros)/launch/empty_world.launch">

   <arg name="world_name" value="$(arg world_name)"/> <!--
world_name is wrt GAZEBO_RESOURCE_PATH environment
variable -->

   <arg name="paused" value="false"/>
```

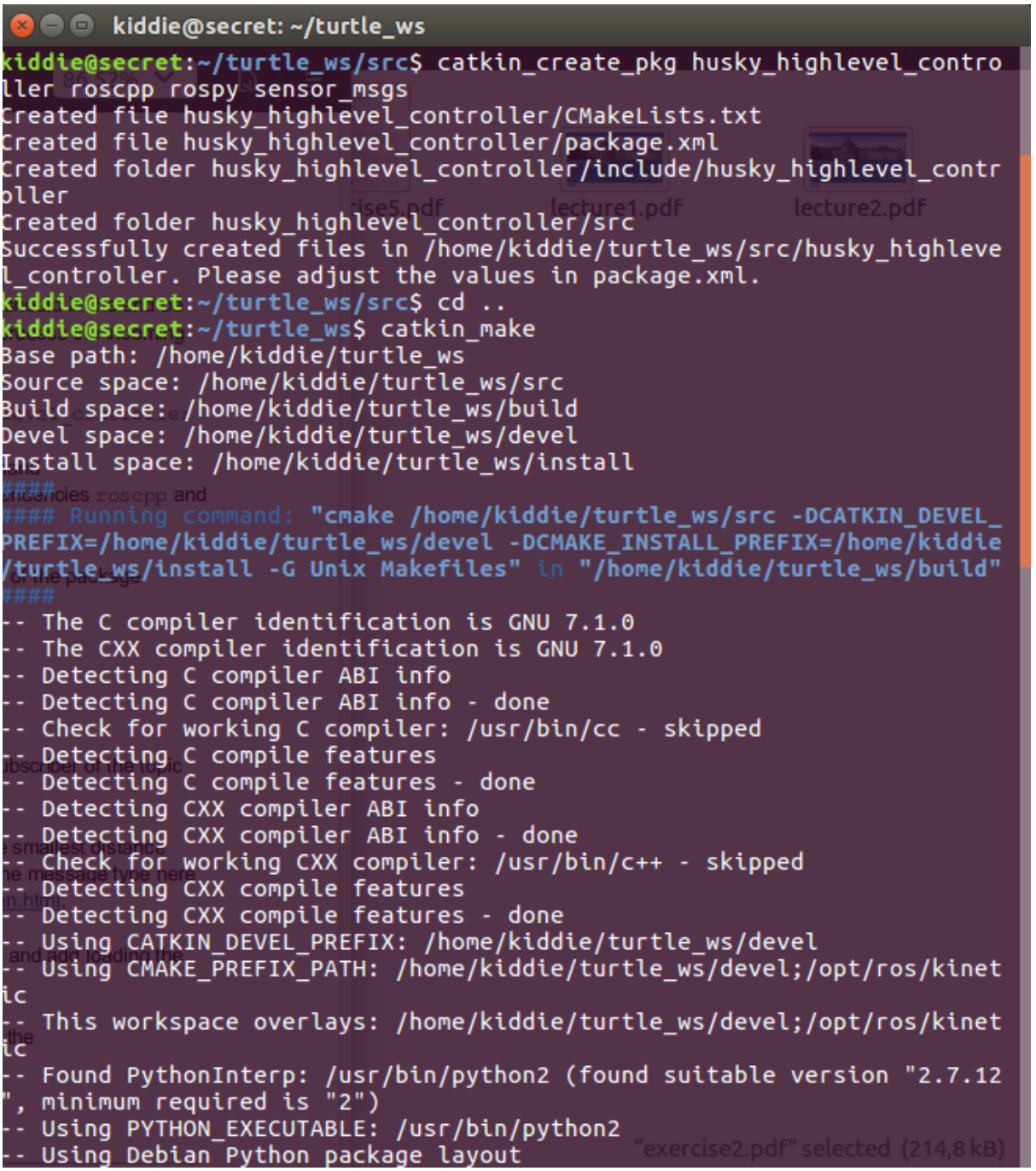
```
    <arg name="use_sim_time" value="true"/>

    <arg name="gui" value="true"/>

    <arg name="headless" value="false"/>

    <arg name="debug" value="false"/>

  </include>

  <include file="$(find
husky_gazebo)/launch/spawn_husky.launch">

    <arg name="laser_enabled" value="$(arg
laser_enabled)"/>

    <arg name="kinect_enabled" value="$(arg
kinect_enabled)"/>

  </include>

  <node pkg="teleop_twist_keyboard" name="teleop_husky"
type="teleop_twist_keyboard.py"/>

</launch>
```

3. run roslaunch <ros package> <launch file>
4.

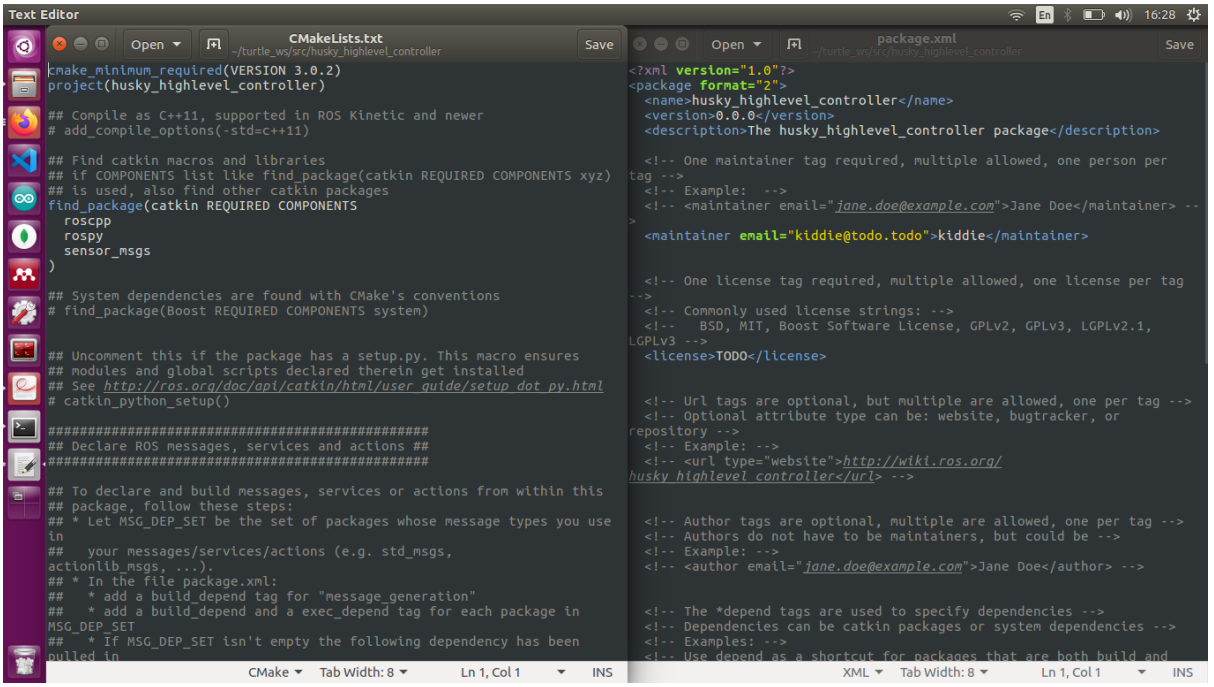


Exercise 2

1. create package with dependency
   run catkin_create_pkg <name package> <depens1><depens2>



2. (simple mode) download from the course website ETH RSL
3. Check the CMakelists.txt and package.xml files. (Lecture 2 slides 5-7 pages)



4. Create a subscriber for the /scan topic. (Lecture 2 Slide 17/19)

```
rospy.Subscriber(topic,queue_size,laser_callback)
```

5. Subscribe to /scan and include a parameter file with topic name and queue size. (Lecture 2 Slide 20/21)

```
laser_scan:
        topic: /scan
        queue_size: 100
```

6. Create a callback method for the user to output the minimum distance measurement result from the laser scanner to the terminal.

```
def laser_callback(msg):

    rospy.info(rospy.get_caller_id() + 'value : %s',
msg.range_min)
```

7. Add your launch file from Exercise 1 and add the current node and add loading the parameter file to the launch file.

```
  <node name="husky_highlevel_controller" pkg="husky_highlevel_controller" type="husky_highlevel_controller" output="screen" launch-
prefix="gnome-terminal --command" >
      <rosparam command="load" file="$(find husky_highlevel_controller)/config/config.yaml"/>
  </node>

  <include file="$(find husky_gazebo)/launch/spawn_husly.launch">
      <arg name="laser_enabled" value="$(find laser_enabled)"/>
      <arg name="kinect_enabled" value="$(find kincet_enabled)"/>

  </include>

  <node name="rviz" pkg="rviz" type="rviz"/>
```

8. Pass the argument laser_enabled from your launch file to the husky_empty_world.launch file with value true

```
    <arg name="laser_enabled" default="true"/>
```

9. Show the laser scan in RViz and add RViz to your launch file

```
    <node name="rviz" pkg="rviz" type="rviz"/>
```

Exercise 3