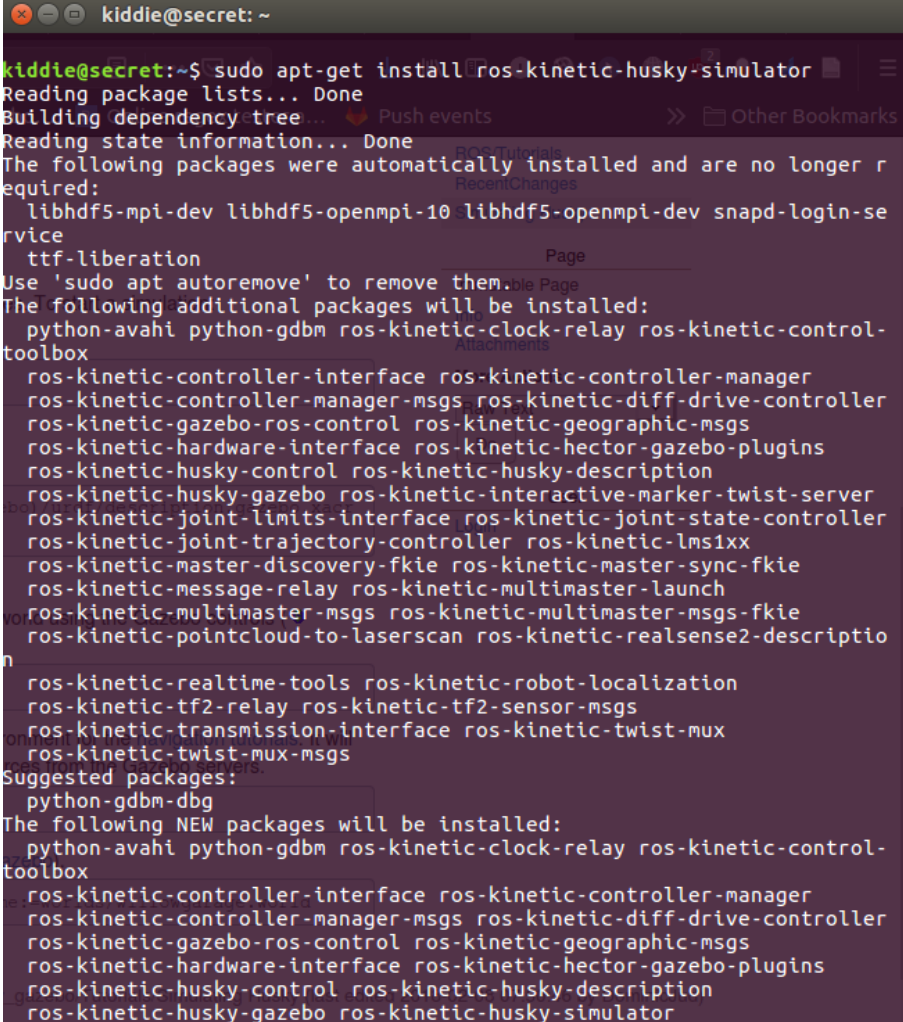


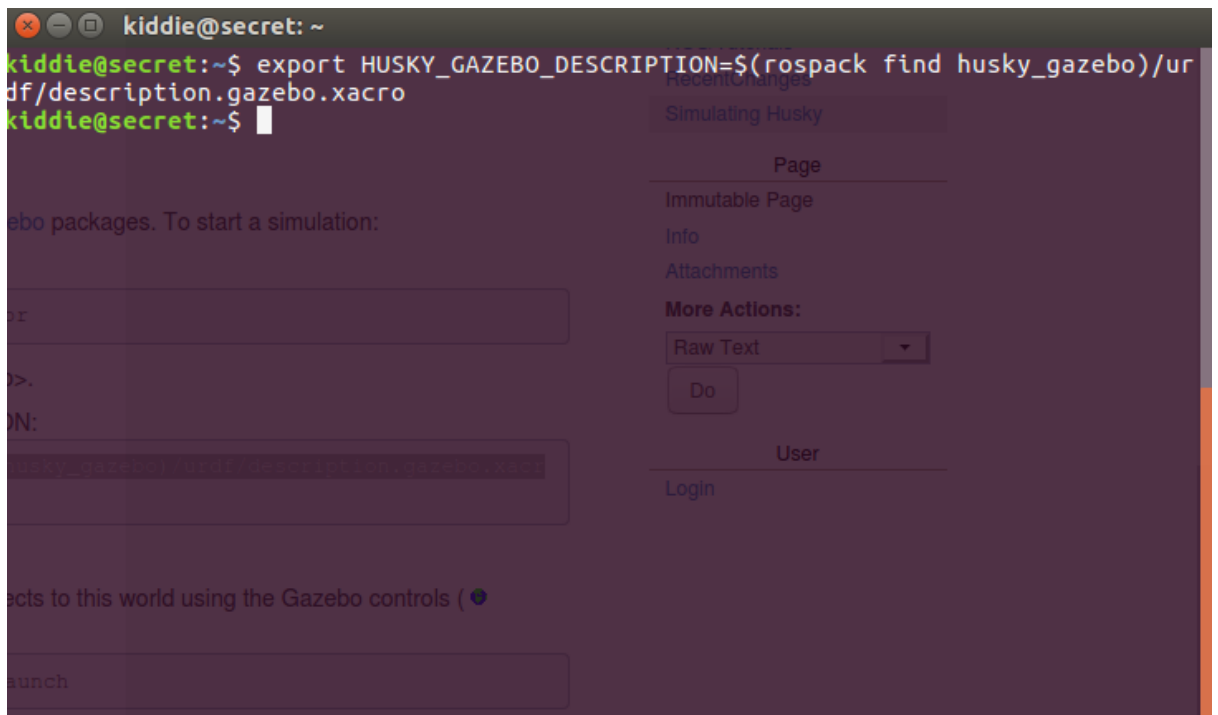
Exercise 1

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



```
kiddie@secret: ~  
kiddie@secret:~$ sudo apt-get install ros-kinetic-husky-simulator  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following packages were automatically installed and are no longer required:  
  libhdf5-mpi-dev libhdf5-openmpi-10 libhdf5-openmpi-dev snapd-login-session  
  ttf-liberation  
Use 'sudo apt autoremove' to remove them.  
The following additional packages will be installed:  
  python-avahi python-gdbm ros-kinetic-clock-relay ros-kinetic-control-toolbox  
  ros-kinetic-controller-interface ros-kinetic-controller-manager  
  ros-kinetic-controller-manager-msgs ros-kinetic-diff-drive-controller  
  ros-kinetic-gazebo-ros-control ros-kinetic-geographic-msgs  
  ros-kinetic-hardware-interface ros-kinetic-hector-gazebo-plugins  
  ros-kinetic-husky-control ros-kinetic-husky-description  
  ros-kinetic-husky-gazebo ros-kinetic-interactive-marker-twist-server  
  ros-kinetic-joint-limits-interface ros-kinetic-joint-state-controller  
  ros-kinetic-joint-trajectory-controller ros-kinetic-lms1xx  
  ros-kinetic-master-discovery-fkie ros-kinetic-master-sync-fkie  
  ros-kinetic-message-relay ros-kinetic-multimaster-launch  
  ros-kinetic-multimaster-msgs ros-kinetic-multimaster-msgs-fkie  
  ros-kinetic-pointcloud-to-laserscan ros-kinetic-realsense2-description  
  ros-kinetic-realtime-tools ros-kinetic-robot-localization  
  ros-kinetic-tf2-relay ros-kinetic-tf2-sensor-msgs  
  ros-kinetic-transmission-interface ros-kinetic-twist-mux  
  ros-kinetic-twist-mux-msgs  
Suggested packages:  
  python-gdbm-dbg  
The following NEW packages will be installed:  
  python-avahi python-gdbm ros-kinetic-clock-relay ros-kinetic-control-toolbox  
  ros-kinetic-controller-interface ros-kinetic-controller-manager  
  ros-kinetic-controller-manager-msgs ros-kinetic-diff-drive-controller  
  ros-kinetic-gazebo-ros-control ros-kinetic-geographic-msgs  
  ros-kinetic-hardware-interface ros-kinetic-hector-gazebo-plugins  
  ros-kinetic-husky-control ros-kinetic-husky-description  
  ros-kinetic-husky-gazebo ros-kinetic-husky-simulator
```

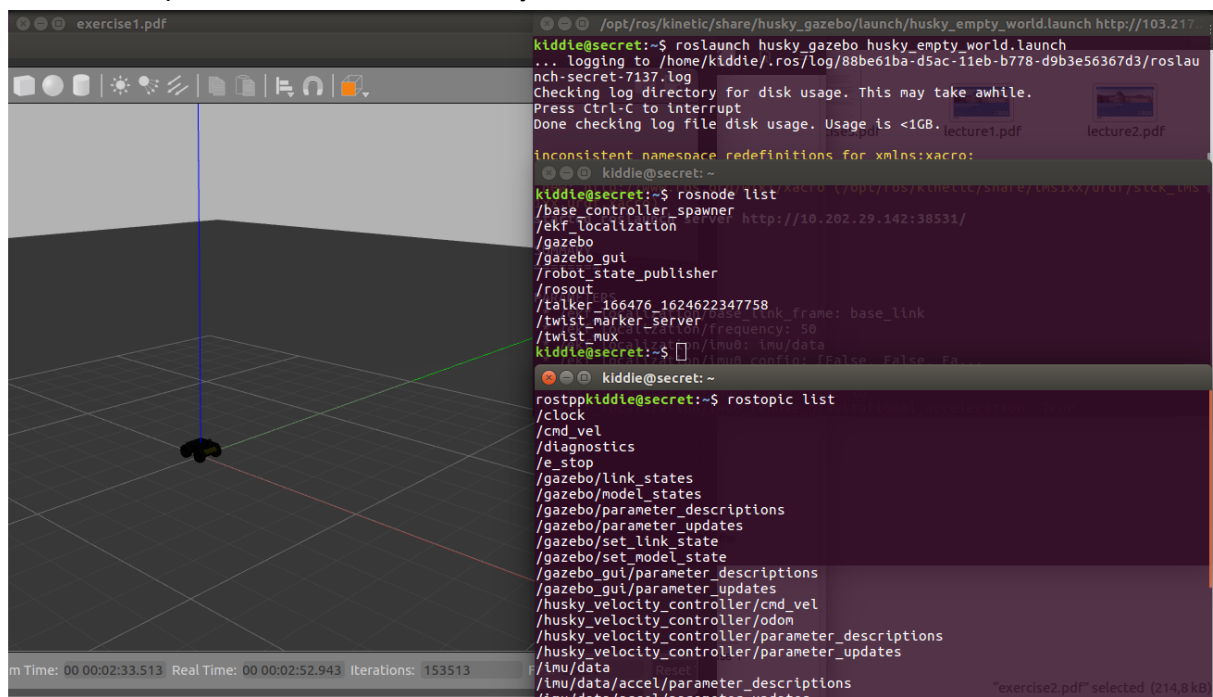
set environment with command
`export HUSKY_GAZEBO_DESCRIPTION=$(rospack find
husky_gazebo)/urdf/description.gazebo.xacro`

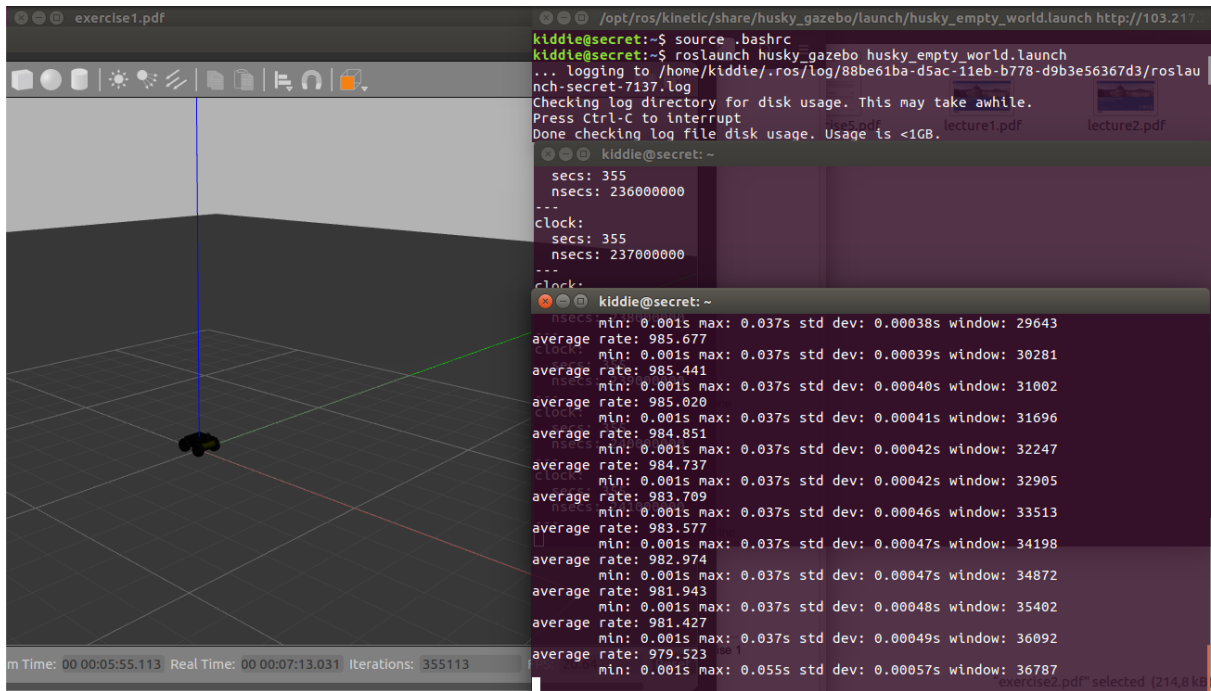


and final step run `roslaunch husky_gazebo husky_empty_world.launch` to see husky simulator

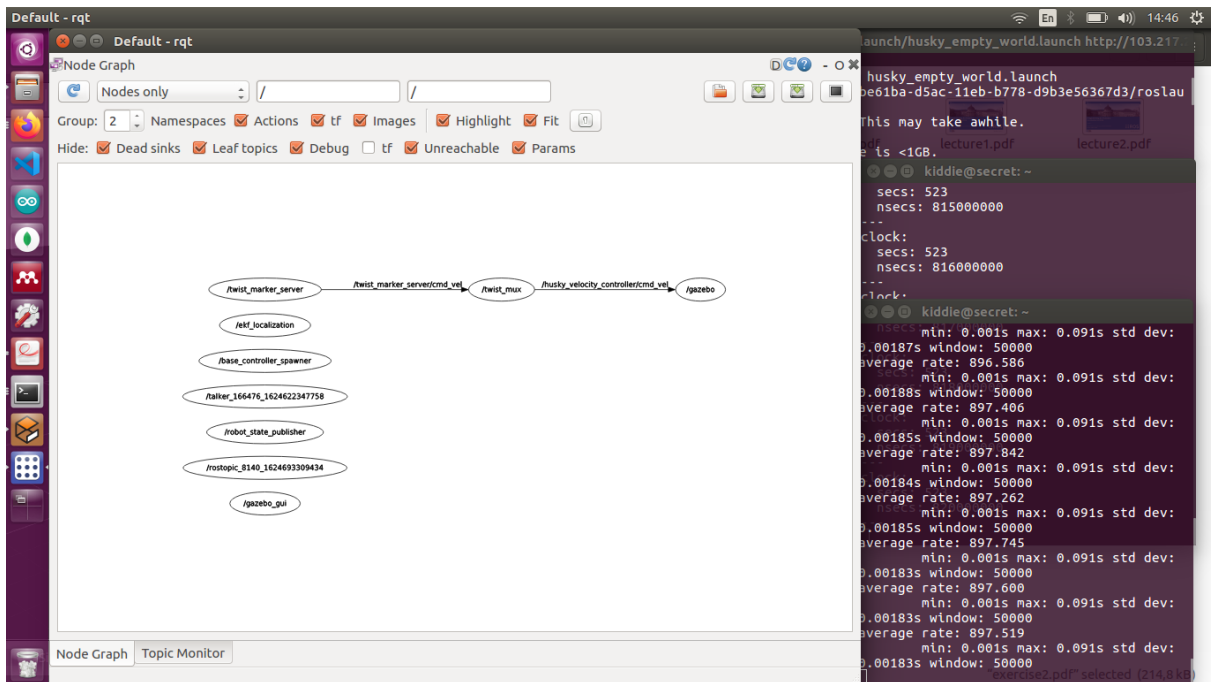
2.

rostopic and rosnod while husky simulator has started

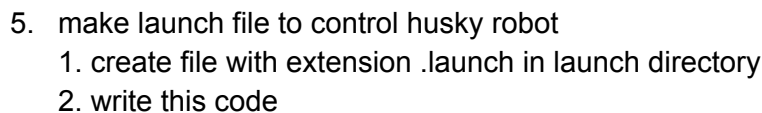




to see node that run on husky simulator



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



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

<!--
-->

<launch>

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

  <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>

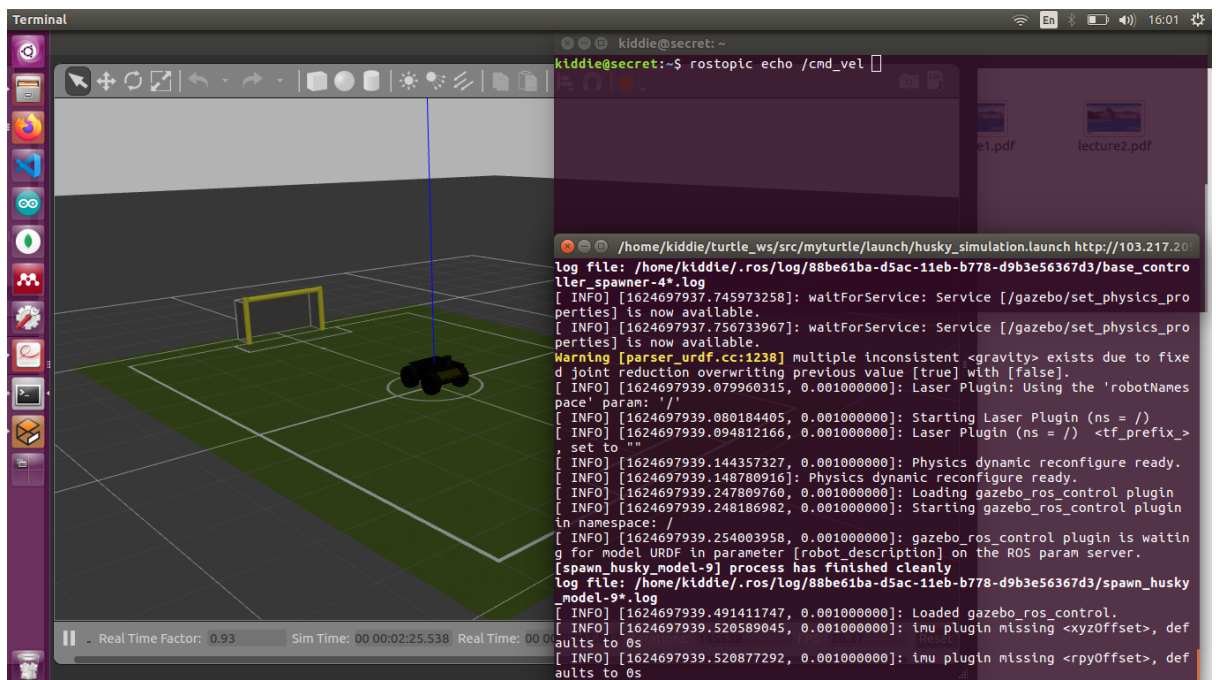
<node pkg="teleop_twist_keyboard" name="teleop_husky"
type="teleop_twist_keyboard.py"/>

</launch>

```

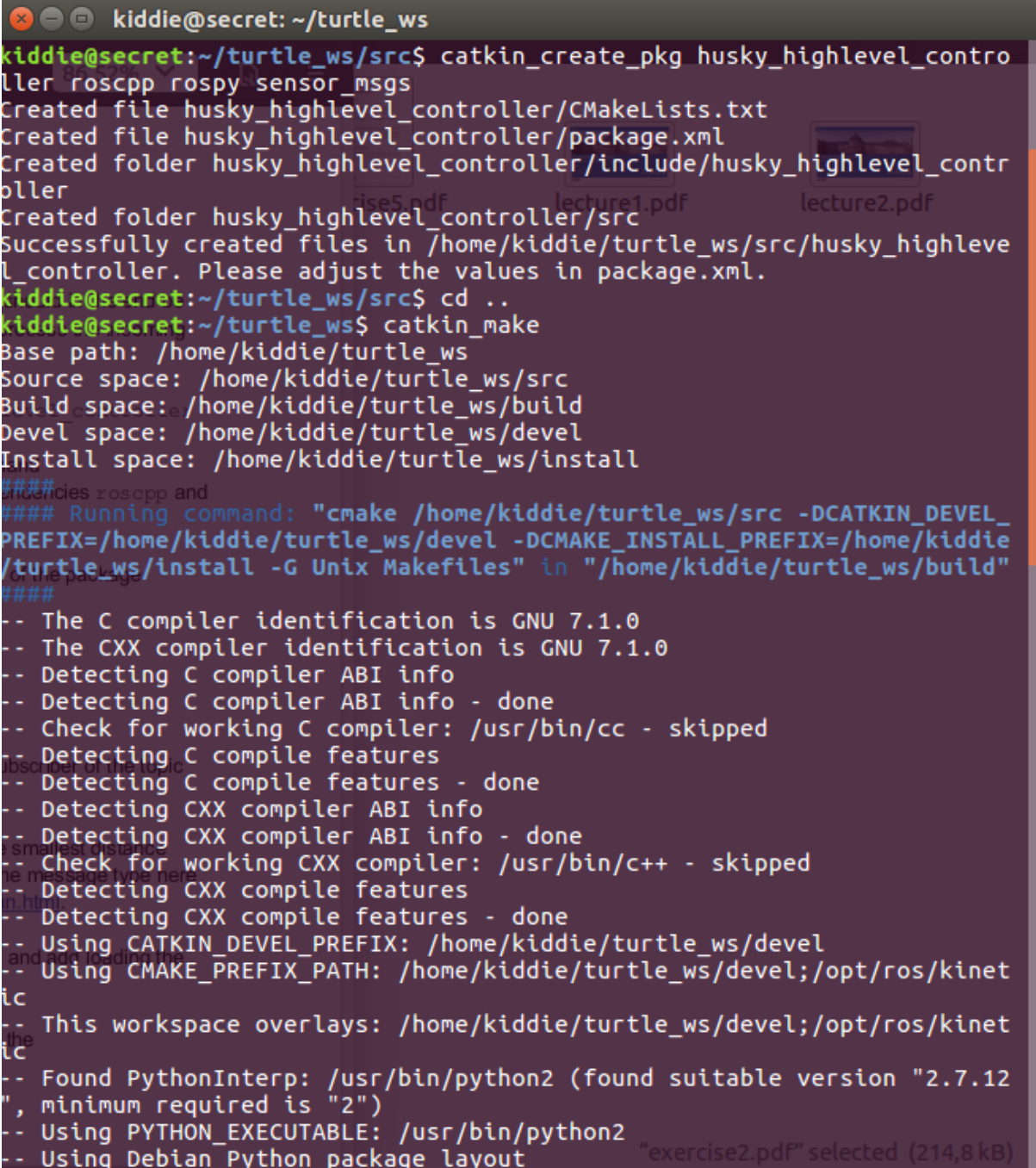
3. run roslaunch <ros package> <launch file>

4.



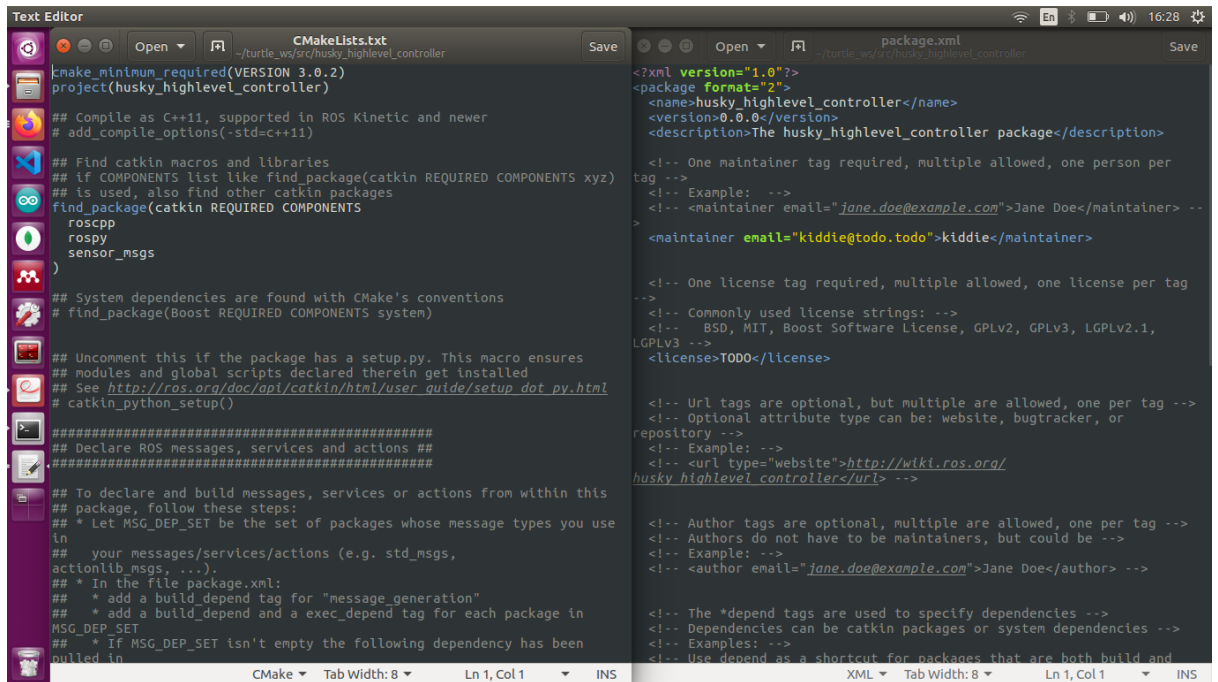
Exercise 2

1. create package with dependency
run `catkin_create_pkg <name package> <depends1><depends2>`



```
kiddie@secret: ~/turtle_ws
kiddie@secret:~/turtle_ws/src$ catkin_create_pkg husky_highlevel_contro
ller roscpp rospy sensor_msgs
Created file husky_highlevel_controller/CMakeLists.txt
Created file husky_highlevel_controller/package.xml
Created folder husky_highlevel_controller/include/husky_highlevel_contr
oller
Created folder husky_highlevel_controller/src
Successfully created files in /home/kiddie/turtle_ws/src/husky_highleve
l_controller. Please adjust the values in package.xml.
kiddie@secret:~/turtle_ws/src$ cd ..
kiddie@secret:~/turtle_ws$ catkin_make
Base path: /home/kiddie/turtle_ws
Source space: /home/kiddie/turtle_ws/src
Build space: /home/kiddie/turtle_ws/build
Devel space: /home/kiddie/turtle_ws/devel
Install space: /home/kiddie/turtle_ws/install
####
#### Running command: "cmake /home/kiddie/turtle_ws/src -DCATKIN_DEVEL_
PREFIX=/home/kiddie/turtle_ws/devel -DCMAKE_INSTALL_PREFIX=/home/kiddie
/turtle_ws/install -G Unix Makefiles" in "/home/kiddie/turtle_ws/build"
####
-- The C compiler identification is GNU 7.1.0
-- The CXX compiler identification is GNU 7.1.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Using CATKIN_DEVEL_PREFIX: /home/kiddie/turtle_ws/devel
-- Using CMAKE_PREFIX_PATH: /home/kiddie/turtle_ws/devel;/opt/ros/kinet
ic
-- This workspace overlays: /home/kiddie/turtle_ws/devel;/opt/ros/kinet
ic
-- Found PythonInterp: /usr/bin/python2 (found suitable version "2.7.12
", minimum required is "2")
-- Using PYTHON_EXECUTABLE: /usr/bin/python2
-- Using Debian Python package layout
```

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="$(find husky_gazebo)/launch/spawn_husky.launch">
  <arg name="laser_enabled" value="$(find laser_enabled)"/>
  <arg name="kinect_enabled" value="$(find kinect_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