```
/*********************************************************************
**********************************************/
/* packages that need to be installed (no need to do this) */
/*********************************************************************
**********************************************/
sudo apt-get install ros-indigo-urg-node
sudo apt-get install ros-indigo-hokuyo-node
sudo apt-get install ros-indigo-hector-slam
sudo apt-get install ros-indigo-map-server
sudo apt-get install ros-indigo-amcl
sudo apt-get install python-serial

/*********************************************************************
**********************************************/
/* Create the package (what you did in Lab 1) */
/*********************************************************************
**********************************************/

//If you do not have the catkin folder, go to here and follow tutorial
http://sdk.rethinkrobotics.com/wiki/Workstation_Setup

cd ~/ros_ws/src
catkin_create_pkg wall_follower hokuyo_node urg_node tf map_server rviz
std_msgs nav_msgs rospy roscpp
cd ..
catkin_make
/* If catkin_make does not work, do $ source /opt/ros/indigo/setup.bash,
then do the catkin_make again    */
source ./devel/setup.bash

/* Copy all provided source files to the directory that we just created
in /ros_ws/src/wall_follower/ as following  */
wall_follower/launch/load_map.launch
wall_follower/src/way_point.py
wall_follower/resources/map/hallway.yaml    // open hallway.yaml, edit
the first line so that it matches with your directory
wall_follower/resources/map/hallway.pgm

/* make .py files executable */
cd ~/ros_ws/src/wall_follower/src
chmod +x *.py
cd ~/ros_ws

/*********************************************************************
**********************************************/
/* Use the RViz package to debug your setup. (This requires NO CODE) */
/* YOU MUST DEBUG ANY PROBLEMS YOU FIND! Any error messages on the terminal?
Are the file names correct? What about permissions? What else? */
/*********************************************************************
**********************************************/
/* example to load the map and display way points */
```

```
/*     setup environment    */
source /opt/ros/indigo/setup.bash
source ~/ros_ws/devel/setup.bash
/*     run launch file      */
roslaunch wall_follower load_map.launch

/* Carefully look over the launch and yaml files to understand them!!
This loads the map of the hallway outside MGL 1219. Waypoint.py lets you
click on the map with "2D Nav Goal" in RViz to get way point coordinates.
To see the map the first time in RViz, click “Add” and click “By Topic”
then click “Map” and “OK.” You can also add the “Marker” */

/*********************************************************************
**************************************************/
/* rviz setup */
/*********************************************************************
**************************************************/
Fixed frame: map
To see the topics, they need to be added and displayed: map, Marker (as
noted above)

/* rviz will load last saved configuration by default */



/*********************************************************************
**************************************************/
/* connect to hokuyo laser */
/*********************************************************************
**************************************************/

Hokuyo URG-04LX-URG01:

/* give permission to read Hokuyo URG-04LX-URG01 */
sudo chmod a+rw /dev/ttyACM0

example launch file for URG-04:

<launch>
      <!--  hokuyo laser driver for URG-04LX-UG01, -120 to 120 degrees, 5m
range -->
      <node pkg="hokuyo_node" type="hokuyo_node" name="run_hokuyo_04lx" >
          <param name="min_ang" type="double" value="-2.09439510239" />
      <param name="max_ang" type="double" value="2.09439510239" />
      </node>
</launch>


Hokuyo URG-10LX:

Add an ethernet port driver (eth0 or equivalent) to the computer (not
necessary for the desktop workstation – just unplug the cable and plug in
```

```
the Hokuyo):
IP Address 128.46.112.xxx (other than 128.46.112.200)
Subnet Mask 255.255.255.0
Default Gateway 128.46.0.1

example launch file:

<launch>
     <!-- hokuyo laser driver for URG-10LX, -135 to 135 degrees, 10m range
-->
     <node pkg="urg_node" type="urg_node" name="run_hokuyo_10lx" >
          <param name="ip_address" value="128.46.112.200"/>
         <param name="min_ang" type="double" value="-2.35619449019" />
     <param name="max_ang" type="double" value="2.35619449019" />
     </node>
</launch>

/*********************************************************************
***********************************************/
/* example of sending command to triwheel car through serial in a python
script */
/*********************************************************************
***********************************************/
#!/usr/bin/env python
import serial
import numpy as np

/* initialize serial port to communicate with triwheel car */
console_ser = serial.Serial(port = "/dev/ttyUSB0",baudrate=115200)
     # the port can be changed if there are multiple usb devices
console_ser.close()
console_ser.open()

/* Ackermann steering commands */
// mode of triwheel car, char
mode = 'A'
// throttle of triwheel car, int16, two bytes signed integer ranges from
-2048 to +2048, interpreted as -100% to +100% throttle
gas_pedal = 204
// steering angle of triwheel car, int16, two bytes signed integer ranges
from -2048 to +2048, interpreted as -50 degrees to +50 degrees
steering_angle = 0

The "str()" is one way to convert integers to a string.
You can also use the format commands:
     "%+04s" 45

/* construct and send the serial commands to triwheel car */
console_ser.write("A+1024-0100")
/* follow this format and make sure the string of gas_pedal and
steering_angle has five bytes including the sign  */
```