Exercise:

1. Download ***turtlebot3*** from the following links:
    a. Setup dependencies first: (**CHOOSE NOETIC VERSION IN LINK**)
    https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/
    b. Install simulation package: (**CHOOSE NOETIC VERSION IN LINK**)
    https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/#gazebo-simulation

Lab 1 Review:

2. Make your workspace if you haven't build in the previous step and build it with *catkin_make* or *catkin build*
3. Launch the simulation with roslaunch and inspect the created nodes and their topics using (Lecture 1 Slides 11/12):

```
rosnode list
rostopic list
rostopic echo [TOPIC]
rostopic hz [TOPIC]
rqt_graph
```

    For more information take a look at the slides or:
    http://wiki.ros.org/rostopic
    http://wiki.ros.org/rosnode

4. Command a desired velocity to the robot from the terminal (`rostopic pub [TOPIC]`)
5. Use teleop_twist_keyboard to control your robot using the keyboard. Find it online and compile it from the source! Use git clone to clone the repository to the folder ~/git. (you can use the teleop that comes with turtlebot3 also)

For a short git overview see:
http://rogerdudler.github.io/git-guide/files/git_cheat_sheet.pdf

Lab 2 Review:

6. Launch different Launch files for different worlds. (inspect the turtlebot_gazebo package for launch files)
7. Create the package turtlebot3_highlevel_controller from scratch. You can use the command catkin_create_pkg to create a new package with the dependencies rospy and sensor_msgs. (use roscpp also for C++)
8. Inspect the CMakelists.txt and package.xml files.
9. Create a subscriber to the /scan topic.

Lab 3 Exercise:

10. Add a parameter file with topic name and queue size for the subscriber of the topic /scan.
11. Create a callback method for that subscriber which outputs the smallest distance measurement from the vector ranges in the message of the laser scanner to the terminal. Inspect the message type here
 http://docs.ros.org/en/api/sensor_msgs/html/msg/LaserScan.html

12. Add your launch file from before to this package and modify it to:
    - run the turtlebot3_highlevel_controller node.
    - load the parameter file.
13. Pass the argument laser_enabled from your launch file to the turtlebot3 launch file with value true.
14. Show the laser scan in RViz and add RViz to your launch file. Make sure to set odom as the Fixed Frame (under Global Options) and adapt the size of the laser scan points. You can save your current RViz configuration as the default configuration by pressing ctrl+s.

15. Create a publisher on the topic /cmd_vel to be able to send a twist command to turtlebot3. You need to add geometry_msgs as a dependency to your CMakeLists.txt and package.xml (same structure as with sensor_msgs).
16. Write a simple P controller that drives turtlebot3 towards a point. Remember to use ROS parameters for your controller gains Write the code in the callback method of the laser scan topic.