Setting up a Gazebo simulation

In case of any queries related to this tutorial, contact **Ashwin Shenai (9325635325)**

This tutorial will help you familiarize yourself with setting up a Gazebo simulation.
There are no submissions for this part, however you will need this later.

- Install the simulation package using **apt-get**: fire up a terminal and execute -

```
sudo apt-get install ros-<distro>-husky-simulator
```

  - Remember to replace <distro> with **melodic**.
- Set up an environmental variable **HUSKY_GAZEBO_DESCRIPTION**: in the same terminal, execute -

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

  - Execute this command **again** if you open a new terminal during this tutorial.
- Try running the following simulation environment:

```
roslaunch husky_gazebo husky_playpen.launch
```

  - It might take some time, as Gazebo will download files from the net when it runs for the first time.
- Try running the simulation in another world: execute the following command -

```
roslaunch husky_gazebo husky_empty_world.launch world_name:=worlds/willowgarage.world
```

We are always looking for feedback and suggestions.
Contact the creator of this tutorial if you wish to provide feedback on this tutorial.

Source: ROS Wiki

Abhay Varshney | Madhur Deep Jain | Neil Shirude
Robotics Club, IIT Kanpur
roboticsclubiitkanpur@gmail.com

ROS from the Terminal

In case of any queries related to this tutorial, contact **Ashwin Shenai (9325635325)**

This tutorial will help you get familiarized with the ROS terminal commands.
For submissions, follow the instructions and fill **this** form as you do the assignment.

- Launch the simulation we set up previously. Fire up a terminal and execute:

```
roslaunch husky_gazebo husky_empty_world.launch world_name:=worlds/willowgarage.world
```

  - Remember to set the **environment variable** since it's a new terminal.

- Inspect the create nodes and their topics using the following commands:
  rosnode list rostopic list rosservice list rosnode info [NODE]
  rostopic echo [TOPIC] rostopic hz [TOPIC] rossrv show [TYPE]
  rosservice info [SERVICE] rosmsg show rostopic info [TOPIC]

- Execute rqt_graph. Take a screenshot of the resulting graph.
  - Display only active nodes/topics.
  - Remember to press the button next to 'Fit' for a properly viewable graph.

- Analyse the graph generated to find the topic at which you can give **velocity commands**. Use rostopic pub [TOPIC] to publish a linear velocity of 10 in the -x direction to the Husky.
  - If it stops after some time, try specifying the publishing rate (See rostopic --help)

- Use **rosbag record** to record data from the **odometry** topic (Find the exact topic name using the graph.) **Once you have started recording**, execute the **publish** command. **Stop** the recording after **10 seconds**, then **upload the bag** into the form.

We are always looking for feedback and suggestions.
Contact the creator of this tutorial if you wish to provide feedback on this tutorial.

Reference: ETH-RSL : Programming for Robotics

Abhay Varshney | Madhur Deep Jain | Neil Shirude
Robotics Club, IIT Kanpur
roboticsclubiitkanpur@gmail.com

Using open-source packages

In case of any queries related to this tutorial, contact **Ashwin Shenai (9325635325)**

This tutorial will familiarize you with using an open-source ROS package.
For submissions, follow the instructions and fill **this** form as you do the assignment.

- In this section, you will control the robot in simulation using your keyboard. To do this, you will need the **teleop_twist_keyboard** package. Search for it on GitHub. Once you have found it, clone it into the src folder of your workspace.
- Build the package you have just cloned. Remember to source **devel/setup.bash**.
- Launch the Husky simulation in a suitable world. In a new terminal, run the node found in the **teleop_twist_keyboard** package using rosrun.
- Use your keyboard to confirm whether the Husky responds in simulation.
- Using the terminal commands, inspect all the nodes and topics.
- Instead of launching the teleop node in a different terminal, let us launch it with the simulation by creating a common launch file.
- In the same **teleop_twist_keyboard** package, create a new launch file in the appropriate directory. Name it like 'sim.launch' or something.
- In the launch file you have just created, **include** the Husky simulation launch file. Specify the world name as 'worlds/robocup14_spl_field.world'. Use the 'arg' tag as shown:

```
<include file="$(find husky_gazebo)/launch/husky_empty_world.launch">
        <arg name="world_name" value="worlds/robocup14_spl_field.world"/>
</include>
```

- After including the simulation launch file, now add the teleop node.
- Remember to check that all your tags are properly closed.
- Now, open up a new terminal and launch the file you have just written.
- Verify that the launch file works via keyboard.

- Once you have launched the file, open a new terminal and record a rosbag of the topics '/husky_velocity_controller/odom' and '/husky_velocity_controller/cmd_vel'. Increase linear velocity to at least 10, then start recording the bag. Drive the bot randomly for 10 seconds and then stop the recording.

We are always looking for feedback and suggestions.
Contact the creator of this tutorial if you wish to provide feedback on this tutorial.

Reference: ETH-RSL : Programming for Robotics

Abhay Varshney | Madhur Deep Jain | Neil Shirude
Robotics Club, IIT Kanpur
roboticsclubiitkanpur@gmail.com