F1/10 ROS Simulator Lab

Launch the simulator:

```
roslaunch f1tenth-sim simulator.launch run_gazebo:=true keyboard_control:=true
```

Understanding the LIDAR

```
madhur@ubuntu:~$ rostopic list
```

```
/car_1/scan
/car_1/command
```

```
madhur@ubuntu:~$ rostopic info /car_1/scan
Type: sensor_msgs/LaserScan

Publishers:
 * /gazebo (http://ubuntu:36655/)

Subscribers: None
```

```
madhur@ubuntu:~$ rosmsg show sensor_msgs/LaserScan
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities
```

```
madhur@ubuntu:~$ rostopic echo /car_1/scan
```

```
madhur@ubuntu:~$ rostopic echo /car_1/scan -n1
```

```
header:
  seq: 0
  stamp:
    secs: 277
    nsecs: 574000000
  frame_id: "car_1_laser"
angle_min: -2.3561899662
angle_max: 2.3561899662
angle_increment: 0.00436331471428
time_increment: 0.0
scan_time: 0.0
range_min: 0.10000000149
range_max: 30.0
ranges: [2.46343994140625
```

Ranges between -135 to 135 in the coordinate frame of the LIDAR.

Writing a simple LIDAR subscriber:

```
madhur@ubuntu:~$ cd catkin_ws/src/
madhur@ubuntu:~/catkin_ws/src$ catkin_create_pkg lidarlab rospy
Created file lidarlab/CMakeLists.txt
Created file lidarlab/package.xml
Created folder lidarlab/src
Successfully created files in /home/madhur/catkin_ws/src/lidarlab.
Please adjust the values in package.xml.
```

```
madhur@ubuntu:~/catkin_ws/src/lidarlab$ atom .
```

In src, create a file scan_test.py

```python
#! /usr/bin/env python

import rospy
from sensor_msgs.msg import LaserScan

def callback(data):
    print data.ranges

rospy.init_node("scan_test", anonymous=False)
sub = rospy.Subscriber("/car_1/scan",LaserScan, callback)
rospy.spin()
```

Next create a launch sub-directory in the `lidarlab` package.

Create a new launch file: lidartest.launch

```xml
<launch>
  <node name="scanner" pkg="lidarlab" type="scan_test.py" output="screen">
  </node>
</launch>
```

Build the lidarlab package

```
madhur@ubuntu:~/catkin_ws$ catkin_make
```

Launch the node from the launch file:

```
roslaunch lidarlab lidartest.launch
```

Show the different ways to intereact with the LIDAR data

```python
print len(data.ranges)
```

```python
print data.ranges[540]
```

- Launch Rviz

- Add LaserScan Topic and assign to /car1/scan

- Adjust the size attribute: 0.06

- Add Axis for car1*base*link, car1_laser

- In Gazebo, switch to top view

- Add an obstacle in front of the car - cylinder

- Show the center distance change by launching the `lidartest.launch` again.

Moving the car

```
madhur@ubuntu:~/catkin_ws$ rostopic info /car_1/command
Type: ackermann_msgs/AckermannDrive

Publishers:
 * /keyboard_plugin (http://ubuntu:42535/)

Subscribers:
 * /car_1/control_plugin (http://ubuntu:34461/)
```

Demo keyboard teleop [node in simulator>>src>]

Publish directly to command:

```
madhur@ubuntu:~$ rostopic pub -r 2 /car_1/command ackermann_msgs/AckermannDrive
"{steering_angle: 0.0, steering_angle_velocity: 0.0, speed: 0.5, acceleration: 0.0,
  jerk: 0.0}"
```

SLAM

Launch a new sim instance:

```
user@ros-computer: roslaunch f1tenth-sim simulator.launch run_gazebo:=true
keyboard_control:=true
```

Wait for the simulator to start and do not move the racecar, as the initial position of the racecar will be used as the origin of the map. In the second terminal, enter the following command to bringup the mapping and related nodes:

```
user@ros-computer: roslaunch f1tenth-sim mapping.launch
```

You should see an rviz window open with the mapping configuration already loaded and the screen should show the initial occupancy grid color sequence.

Particle filter

```
user@ros-computer: roslaunch f1tenth-sim localization.launch
```

TEB Planner

No need to have sim running beforehand for the following:

```
user@ros-computer: roslaunch f1tenth-sim navigation.launch
```