UNMESH MASHRUWALA
Innovation Cell
IIT BOMBAY

Training: ROS

Introduction

1) So first of all ***What is a Robot?***

A robot is any system that can perceive the environment that is its surroundings, take decisions based on the state of the environment, and is able to execute the instructions generated.

2)Robot Operating System or simply ROS

is a framework that is used by hundreds of Companies and techies of various fields all across the globe in the field of Robotics and Automation. It provides a painless entry point for nonprofessionals in the field of programming Robots.

3) OS and ROS?

An Operating system is a software that provides an interface between the applications and the hardware. It deals with the allocation of resources such as memory, processor time, etc. by using scheduling algorithms and keeps records of the authority of different users, thus providing a security layer. It almost always has a low-level program called the **kernel** that helps in interfacing with the hardware and is essentially the most important part of any operating system.

ROS is not an operating system but a meta operating system meaning that it assumes there is an underlying operating system that will assist it in carrying out its tasks.

4)What are RVIZ and Gazebo?

RViz is a 3D Visualization tool for ROS. It is one of the most popular tools for visualization. It takes in a topic as input and visualizes that based on the message type being published. It lets us see the environment from the perspective of the robot.

Once we have all the code ready and running, we need to test our code so that we can make changes if necessary. Doing this on a real robot will be costly and may lead to a wastage of time in setting up the robot every time. Hence we use robotic simulations for that. The most popular simulator to work with ROS is **Gazebo**. It has good community support, it is open source and it is easier to deploy robots on it.

5) Why should I use ROS?

ROS provides functionality for hardware abstraction, device drivers, communication between processes over multiple machines, tools for testing and visualization, and much more.

The key feature of ROS is the way the software is run and the way it communicates, allowing you to design complex software without knowing how certain hardware works. ROS provides a way to connect a network of processes (nodes) with a central hub. Nodes can be run on multiple devices, and they connect to that hub in various ways.

Prerequisites for using ROS:

You should be confident in at least C++ or Python
Apart from this, there is no pre-req, everything is mostly explained in the tutorials

Installation guide and tutorials:

Download Kinetic for Ubuntu 16.04 and Melodic for Ubuntu 18.04
Link for installation - ROS/Installation - ROS Wiki
Visit this link and choose the ROS version according to the Ubuntu Version and follow the procedure mentioned there
Link for Tutorials - ROS/Tutorials - ROS Wiki
You are supposed to do Core ROS tutorials 1.1 - Beginner Level
Start with Installing and Configuring Your ROS Environment - 1
And end with Examining the Simple Publisher and Subscriber - 13
Some additional tutorial apart from 1-13

1) 17-18 from the same subsection - Regarding data from Bagfiles
2) For defining custom messages - ROS/Tutorials/DefiningCustomMessages - ROS

Also, you should write codes yourself on the terminal, rather than just reading them or copy-pasting them and see the results for yourself
This will be enough for the beginner level

These links are self-sufficient, ie. they have all of the content and definitions that you need, anyways ROS has a massive community if you face any error comes during installation search that on Google, the solution will be available there
ROS is available for both python and C++

At the end of this tutorial,
You should have a basic knowledge of ROS packages, Rviz, Bagfiles, Subscriber/Publisher in C++/Python(At Least one),

TIMELINE:

Day 1:

Install ROS and the dependencies from the website,You are expected to finish all the tutorials mentioned above and resolve any doubts related to the tutorials on Slack.

Day 2:

Focus on applying the concepts that you learned and completing the assignments

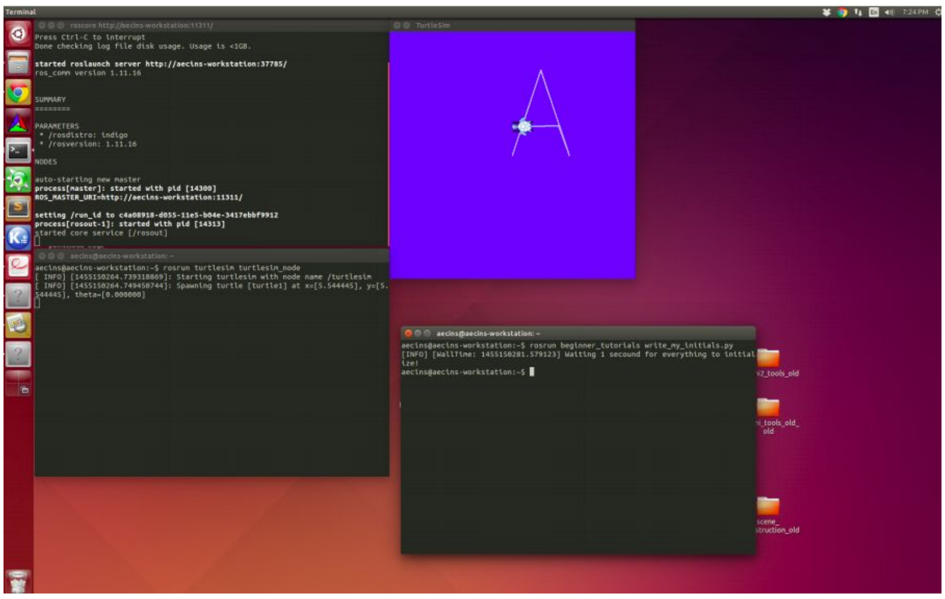
ROS Assignment

Please go through the material provided. Don't attempt the assignments unless you are done with the tutorials. All the best

Basic Assignment:

1. Create a node called my_convertor using python that publishes as well as subscribes data at the same time.
   It should subscribe to *topic1* which has a *custom message type* of quaternions(x,y,z,w).
   The node processes the subscribed data and converts it into Euler angles.
   Refer this link to know more about quaternions and Euler angles.
   It publishes it to *topic2* which has a *custom message type* consisting of roll pitch and yaw values.
   Push both the custom message files and the python file to the repo.

2. Write a node called my_initials using python that will make the turtlesim write the first letter of your name. The output of turtlesim should be similar to the one shown in the figure. If the first letter of your name has curved parts (i.e. an "O") you can approximate them with right angles.
   Push the python file and also the screenshot of your entire screen to the repo.


Terminal
TurtleSim
roscore http://aecins-workstation:11311/
aecins@aecins-workstation: ~

```
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://aecins-workstation:37785/
ros_comm version 1.11.16


SUMMARY
========

PARAMETERS
 * /rosdistro: indigo
 * /rosversion: 1.11.16

NODES

auto-starting new master
process[master]: started with pid [14300]
ROS_MASTER_URI=http://aecins-workstation:11311/

setting /run_id to c4a08918-d055-11e5-b04e-3417ebbf9912
process[rosout-1]: started with pid [14313]
started core service [/rosout]
```

```
aecins@aecins-workstation:~$ rosrun turtlesim turtlesim_node
[ INFO] [1455150264.739318869]: Starting turtlesim with node name /turtlesim
[ INFO] [1455150264.749450744]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

```
aecins@aecins-workstation:~$ rosrun beginner_tutorials write_my_initials.py
[INFO] [WallTime: 1455150281.579123] Waiting 1 secound for everything to initialize!
aecins@aecins-workstation:~$
```

Advanced Assignment: (Try only if you complete the basic one)

This is an extension of the tutorial on recording and playing back ROS data:
http://wiki.ros.org/ROS/Tutorials/Recording%20and%20playing%20back%20data

Here is a ROS bag file that you can download: remapped_turtle.bag

Use the rosbag program to inspect the file. What topics are included in the file?

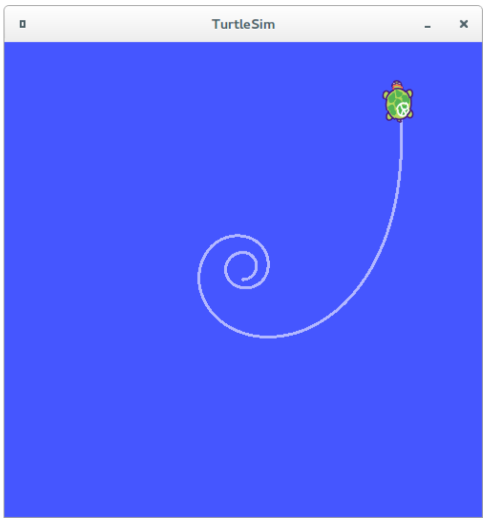
I generated this file based on the tutorial, but I remapped the velocity command from the /turtle1/cmd_vel to a new topic name. You should be able to determine that new topic name.

Now write a new launch file called **turtle_remap.launch** in the **launch** directory of your package. This launch file will start the turtlesim_node and use the remap function to have that node listen to the topic in the remapped_turtle.bag file. That launch file will be as shown below, but you will need to replace the "XXXXXX" with the appropriate topic.

```
<launch>
<node pkg="turtlesim" type="turtlesim_node" name="simulated_turtle">
   <remap from="turtle1/cmd_vel" to="XXXXXX" />
</node>
</launch>
```

- Open a terminal and use roslaunch to execute your launch file.
- Open another terminal and start recording all the ROS data to a new bag file called **myturtle.bag**
- Open a third terminal and plot the rosgraph
- Open yet another terminal and playback the **remapped_turtle.bag file.**

You should see your turtle do something like this...



- Stop your data logging by using Cntrl-C in the terminal where you started the rosbag program.
- Move the **myturtle.bag** file to a new directory in your package called **data**
- Save your rosgraph to a SVG file called **rosgraph.svg** and put it in your project directory**.**

To complete the assignment, make sure that your **myturtle.bag** and **rosgraph.svg** files are added to your git repository, commit and push.