**NUS Rover Team Software Recruitment Assignment**

**Introduction**

The Robotics Operating System(ROS) is a robotics middleware that has been successfully deployed in the majority of industrial mobile robots and robotic arms in the world. ROS provides a platform for roboticist to develop control software for robots that often involve communication between different hardware components such as sensors and actuators. At more advanced levels, communication and controls can extend to over multiple robots and multiple computing platforms.

**Objective**

The primary objective of this assignment is for you to get acquainted with ROS and develop a working knowledge for a programmer such as yourself to pick up the more advanced topics in ROS.

**Assignment Breakdown**

The assignment has been broken down to various challenges and the primary objective is to complete as many challenges as possible. If you are able to complete a specified amount of challenges( to be specified to you by the software team lead) we shawls schedule an interview with you for the ROVER Team.

**Challenge 0: Set up Linux**

ROS is available on a variety of OS Distributions. It is preferable to work with ROS On Linux environments due to the tremendous online support and ease of use. Thus, your first challenge is to install the ubuntu 16.04 LTS system as a dual boot on your computer or on an external/internal Drive with sufficient storage capacity alongside your existing OS.

Disclaimer: Please follow the instructions online, we shall hold no responsibility if you brick your PC or accidentally wipe your existing OS and files (very unlikely to happen if you follow the online instructions carefully)

Some references are as follows:

1. <https://hackernoon.com/installing-ubuntu-18-04-along-with-windows-10-dual-boot-installation-for-deep-learning-f4cd91b58557>
2. <https://opensource.com/article/18/5/dual-boot-linux>
3. <https://www.maketecheasier.com/install-dual-boot-ubuntu-mac/>

After you install Ubuntu, you would then have to install the ROS. Please find the required ROS distribution and installation instructions here:

1. <http://wiki.ros.org/ROS/Installation>

Familiarise yourselves with the UBuntu Linux environment Reference material can be found at <https://ubuntu.com/tutorials/command-line-for-beginners#1-overview>

**Challenge 1 : Building and Executing ROS workspaces and packages**

You would now familiarise yourself with the basic ros commands to start nodes, topic and services. You can refer to <http://wiki.ros.org/ROS/CommandLineTools>. SOMe tutorials can be found at <http://wiki.ros.org/ROS/Tutorials>.

Aquaint yourself with building a catkin workspace using <http://wiki.ros.org/catkin/Tutorials/create_a_workspace>

Familiarise yourself with git version control and clone the repo at <https://github.com/hydrargyrum5799/NUSRoverTeamSWTest> into your src directory

Build the catkin workspace and launch the ROS simulation.

You would require a few extra ROS packages to do so. The following are the basic set of ROS packages required:

1. Turtlebot package
2. KartoSLAM

For any build errors that are resulted due to the lack of packages, figure out the necessary packages using the error statements that are thereby resulted and install the necessary package for a successful build. ( Note that if you face an error regarding ergodic\_explore package please comment out the code in the robot.launch file). More info on launch.xml files can be found here <http://wiki.ros.org/roslaunch/XML>

You should then launch the simulation using the roslaunch command. Read the launch files given under the multi\_robot\_sim/launch/include file and figure out which launch file is required to start the simulation. Acquanit yourself with the simulation tools such as RVIZ

(<http://wiki.ros.org/rviz/UserGuide>)

Your next task is to figure out the corresponding and roslaunch the launch file to teleoperate a single turtlebot. Furthermore, you are also required to modify the **RETRACTED** launch file to simulate two turtlebots.

**Challenge 2: ROS Publisher and Subscribers**

You would now be writing your first python script in this challenge. The primary challenge for you is to control a single turtlebot in the simulation using the keys WASD, where W and S is used to move the robot forward and backward, S and D are used to rotate left and right.

You would have to learn how to create a publisher and subscriber in the

<http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29> .

Refer to the robot architecture which can be brought up using rqt\_graph command to figure out which topics and message types are required for you to publish your commands to.

Acquaint yourself with the common msgs package in ROS. common msgs contains messages that are widely used by other ROS packages, including geometry msgs, nav msgs, and sensor msgs. geometry msgs provides messages for common geometric primitives such as points, vectors, and poses. nav msgs defines navigation-related messages such as odometry and map. sensor msgs defines messages for commonly used sensors including range and laser sensors. Learn more about common msgs here: <http://wiki.ros.org/common_msgs>.

You would require it for this challenge and the following challenges.

**Challenge 3: ROS Actions and MOVE BASE package**

Now that you have understood how to use publishers and subscribers, we would now have a detailed look at the ROS Actions and Services. Browse through the tutorial at <http://wiki.ros.org/actionlib_tutorials/Tutorials> that provides a brief overview of ROS Action Library. We would not require a ROS Service.

The primary challenge for you in this task is to develop an application that moves the robot to a desired location in the map. Have a look at the move\_base package and the corresponding tutorials at <http://wiki.ros.org/move_base>.

You would have to primarily write a python script that reads a location in the given format and calls the movebase action server to send the robot to a desired location on the map. Your primary task is to ensure that the robot moves to location as specified by a user (either through a topic or during runtime).

Familiarise yourself with the **tf** package. **tf** is a package for keeping track of different coordinate frames over time. It maintains the relationship between coordinate frames in a tree structure and lets you transform points, vectors, etc. between frames. Info can be found at <http://wiki.ros.org/tf>

**Challenge 4: Frontier Based exploration and OpenCV for computer vision**

We shall now let the robot autonomously perform and exploration process of the entire environment. The objective is to modify the robot.launch file to include frontier based exploration. The m-explore package with the explore-lite package is provided to you. You can find more information here <http://wiki.ros.org/explore_lite>. Call the required launch files in the package within the robot.launch file. Calling roslaunch robot.launch would thenautomatically start the autonomous exploration

During autonomous exploration, the robot must be able to detect obstacles. Thus, each simulated robot has a kinect RGBD camera sensor. The primary objective in this challenge is for you to write a python script that can detect the white/grayish obstacles in the environment such as the big cubes and cylinders. You can use any detection scheme you like and you must display the bounding boxes for each obstacle detected. Output the centroids of the bounding boxes in the image frame on a seperate topic.

You can refer to the OpenCV package for ROS here <http://wiki.ros.org/vision_opencv>.

OpenCV tutorials can be found at <https://docs.opencv.org/master/d9/df8/tutorial_root.html>.

**Challenge 5: Localisation**

We now arrive at the hardest task for you to complete this challenge.

Each robot in the environment needs to know where it is exactly located. This is achievable by using information from the sensors to estimate where the robot is in the environment. The primary task for you is to develop a python script that subscribes to the corresponding robot’s map topic and the laser scan topic. The map topic consists of the known map of the robot’s environment. The laser scan topic provides the laser scan message.

You are required to write a particle filter algorithm that uses the laser scanner’s data to estimate the position of the robot. Currently, the AMCL package does this and more information can be found at <http://wiki.ros.org/amcl>. You are advised to read the code and understand some of the principles behind particle filters.

Some tutorials on particle filters can be found at:

<http://robots.stanford.edu/papers/fox.mcmc-book.ps.gz>

<http://www.cim.mcgill.ca/~yiannis/particletutorial.pdf>

<https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python> provides the code and brief overview of kalman and bayesian filters. Particle filter is a type of a bayesian filter, thus I believe that you would find this repo useful. You are welcome to try out any other methods that you feel are better, but it is recommended tht you write some code for filtering as a good exercise.