Lab 1 – Setup and Introduction to ROS
100 points

Overview:
The purpose of this lab is to get the ROS system we will be using installed and working on your computer. You will be working through some ROS tutorials and getting to know the ROS environment. To complete the lab you must submit via Canvas a zip file containing your source code and a video recording of your program execution.

Installation Instructions:
In this class we will use ROS Noetic. ROS Noetic is supported on Ubuntu 20.04. [Other versions of ROS will require other versions of Ubuntu, make sure that you check this if you decide to try a different ROS version, or if you have a more recent Ubuntu version]. The labs may work with other versions of ROS and Ubuntu as well but this is not guaranteed.

Part I: Learning ROS

We will be working with ROS Noetic this semester. To get it installed, go to

http://wiki.ros.org/noetic/Installation/Ubuntu

for detailed instructions. Follow the given steps for your operating system. Install the Desktop-Full Install.

Once you have completed your ROS installation, please work through the Beginner Level Core ROS tutorials. There is a place in the tutorials where you have to pick a build system. We will be using the catkin build option. We will also be primarily using Python this semester, so you can just do those options when given both C++ and Python versions of a tutorial.

First, complete the introductory ROS tutorials (beginner level), which can be found at

http://wiki.ros.org/ROS/Tutorials

Next, use Catkin to setup a workspace for code development

http://wiki.ros.org/catkin/Tutorials/create_a_workspace

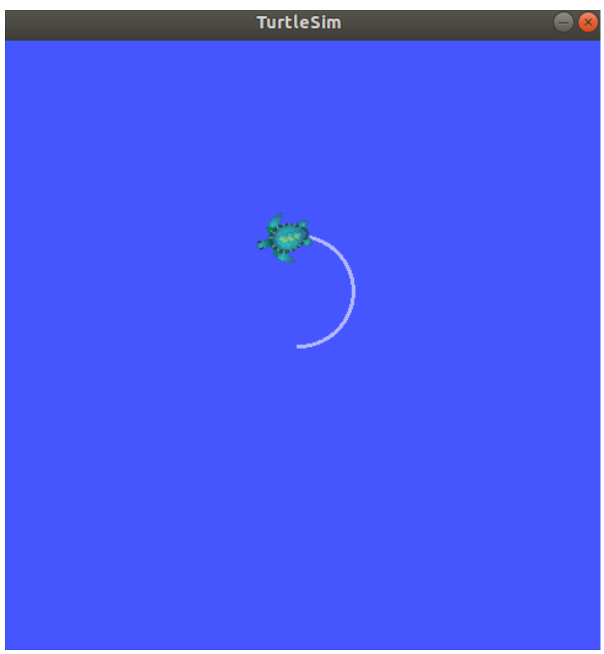
Now you should be set up with ROS and have a basic understanding of how ROS works.

Part II: Using ROS

1. First, we will create a new package to house the code we will work on in lecture.
   - Open a new terminal in Ubuntu and change directory (cd) to catkin_ws/src
   - In catkin_ws/src type the following (all on one line) and then press enter
     - catkin_create_pkg ai_labs std_msgs rospy roscpp sensor_msgs geometry_msgs
   - Unzip programming exercise files from Canvas into new ai_labs folder
   - Then make a directory within ai_labs for our python scripts by running (from that directory)

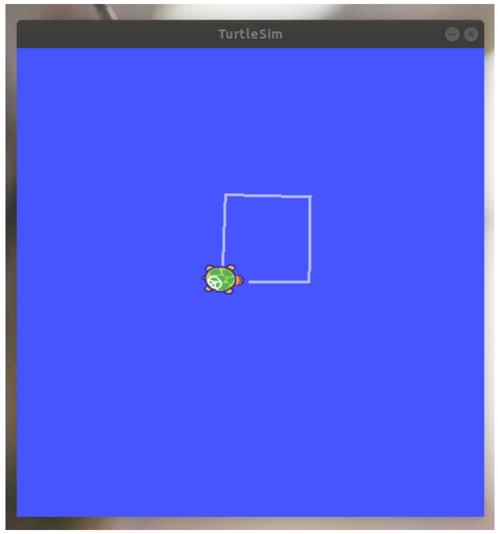
- mkdir scripts
- Then move the python scripts into the scripts folder (e.g. by running:)
  - mv *.py scripts/
- Make sure that the scripts are executable, (e.g.)
  - chmod +x scripts/*
- After you have completed these steps you should have the following directory structure (includes ai_labs directory which you should have set up for lab 1)
  - catkin_ws
    - build
    - devel
    - src
      - ai_labs
        - package.xml
        - CMakeLists.txt
        - include
        - src
        - scripts
          - <python scripts for labs>
        - launch
          - <.launch files for labs>

2. Now we will test that this is all working. Run the following command from the terminal. You should see the simulator window pop up and a robot driving in a circle.
   - roslaunch ai_labs lab1.launch
   - If it doesn't work, you can try to run catkin_make from the catkin_ws folder or to run `source ~/catkin_ws/devel/setup.bash`

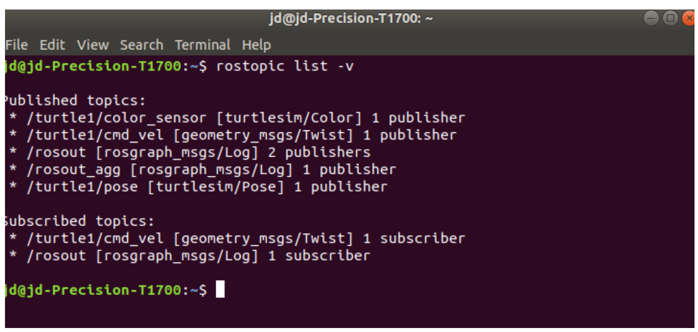


3. Modify the code in square.py so that the robot moves in a square. (70 pts)

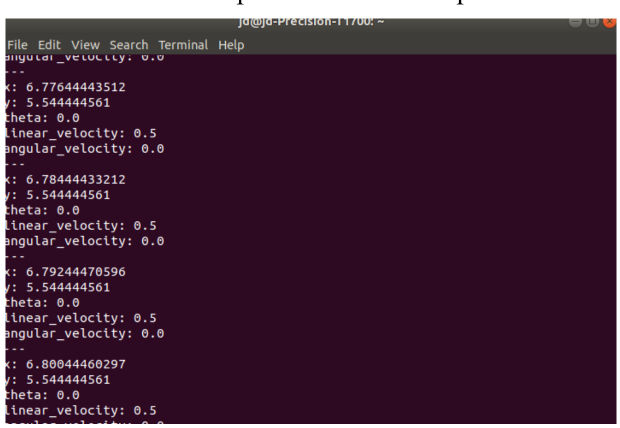
- Note that the code sends a drive command to the robot, then sleeps, awaking 10 times per second (this rate can be changed) to send a new command). To understand commands, please see the appendix below.
- Caution: Try not to hit the wall!



4. Use the *rostopic list -v* command to print out the list of ROS topics. (15 pts)



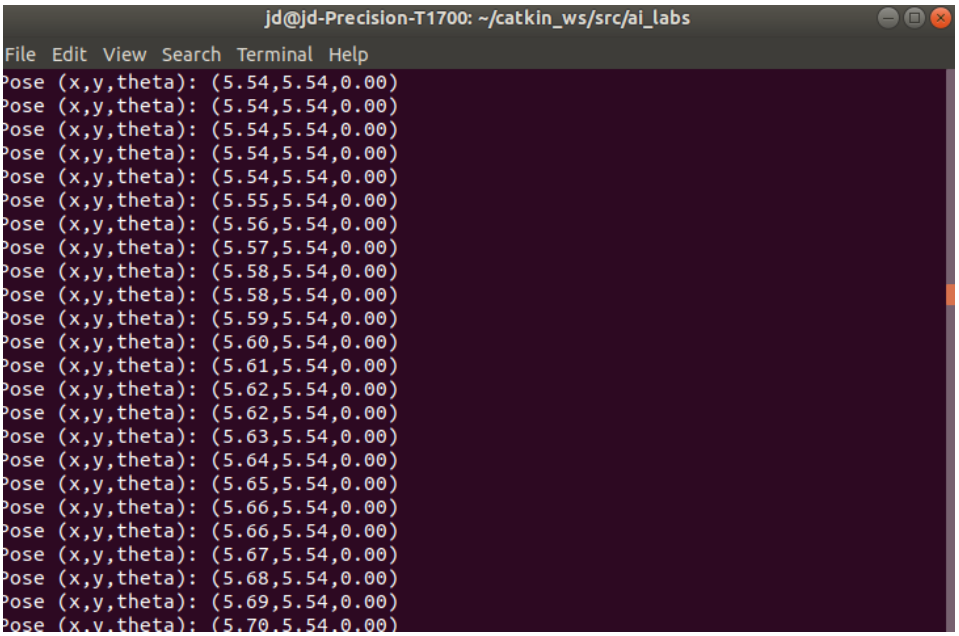
5. Use the *rostopic echo* command to print out the robot pose at each step. (15 pts)



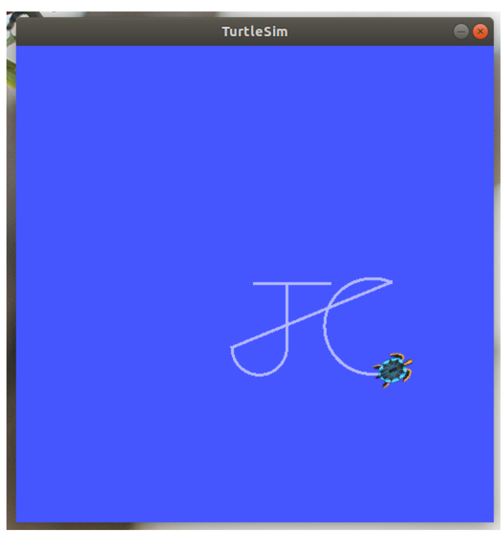
Part III: Extra Credit

1. Create a new Python script named "*subscribe_pose.py*" that implements a simple subscriber to print out the robot pose (10pts). Refer to http://docs.ros.org/en/melodic/api/turtlesim/html/msg/Pose.html to find out the components of a Pose message. The robot pose is published on the topic "*turtle1/pose*". Use the following format:

   Pose (x,y,theta): (8.82,3.59,0.48)



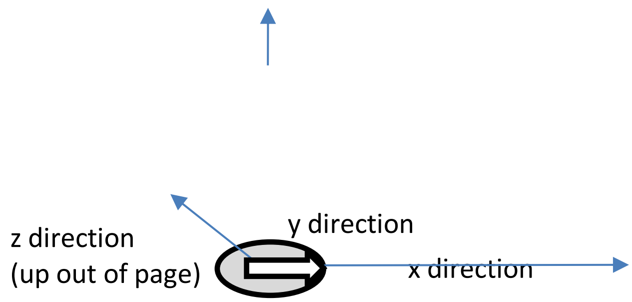
2. Further modify the code in square.py so that the robot moves in a trajectory that spells out your initials. See example below. (20pts)



Tips and Tricks:

Commands are sent via a Twist message:

The following Coordinate system is used, where the grey circle is the robot, facing in the direction of the arrow:



- Twist – A Twist object has two main fields, which allow for linear and angular velocities to be set. This is super general so that it could work for any robot. We will only use two of the fields (**bolded** below), but it helps to remember which two if you understand what they all mean.
    - Twist.linear
        - **Twist.linear.x** – this sets the linear velocity in the x direction, which is forward/backwards (for +/- values). This gives the forward speed of the robot
        - Twist.linear.y – this sets the linear velocity in the y direction, which is sideways. Our robot can't move like this, so we will not use this field.
        - Twist.linear.z – this sets the linear velocity in the z direction, which is up. Our robot can't fly, so we will not use this field either.
    - Twist.angular
        - Twist.angular.x – This sets the angular velocity/ rotation around the x axis, or the roll of the robot. Since we are on the ground, we cannot rotate around this axis, so we won't use this.
        - Twist.angular.y – This sets the angular velocity /rotation around the y axis, or the pitch of the robot (moving front up/down). Since we are on the ground, again, we won't use this.
        - **Twist.angular.z** – This sets the angular velocity/rotation around the z axis, which rotates the robot left/right. We will use this command to turn the robot. Units are in radians/sec - positive is a left turn (increasing the angle/heading of robot), while negative is right turn.

- To drive the robot forward/backward, set twist.linear.x to +/- values. (0.5 seems to work ok)
- To spin the robot, set twist.angular.z to +/- values (again, 0.5 seems a good starting point)
- If you have trouble getting ROS to roscd or errors launching (launch file not known or does not exist) to your directory, try running the following command from the catkin_ws directory. This will make sure that ROS knows about our packages
    - catkin_make
    - source devel/setup.bash
    - Add the line "source <path to your catkin>/catkin_ws/devel/setup.bash" to your .bashrc file to have this happen automatically when you log into a terminal. (make sure that <path to your catkin> is replaced by the path on your system to the catkin_ws folder. You can get this by typing pwd from the terminal when you are in the directory.
    - You can add this to your bashrc file using the following commands (each all on one line) from the terminal -
        - echo "source <path to catkin>/catkin_ws/devel/setup.bash" >> ~/.bashrc

- source ~/.bashrc

Frequently-Asked Questions

Q: Why is my robot not running after executing the `roslaunch` command?
A: The launch file *lab1.launch* requires that the referenced scripts are in the correct locations. Make sure that the file *square.py* is executable, and uses the correct version of Python. Also, make sure the *ai_labs* folder has the correct directory structure.

Q: How can I run ROS code if VirtualBox does not work on my computer?
A: You may use other virtual machine software such as https://mac.getutm.app/ or https://www.vmware.com/products/fusion.html

Q: How to fix the error "`RLException: [lab1.launch] is neither a launch file …`" or "`ERROR: cannot launch node of type [ai_labs/square.py]`" ?
A: Make sure that the lab files are downloaded into the correct directory structure (refer to Part II, Step 1). Run the command `source ~/catkin_ws/devel/setup.bash` to update the ROS search path for packages in your catkin workspace. If ROS cannot detect the ai_labs package, it may be that it is missing the package.xml and CMakeLists.txt files. Make sure that you use catkin_create_pkg command to create the ai_labs package instead of creating it manually.

Q: How to fix the error "`/usr/bin/env: python: No such file or directory`":
A: In line #1 of `square.py`, try changing `#!/usr/bin/env python` to `#!/usr/bin/env python3`

Q: Is it okay if the robot is not moving in a perfect square?
A: Yes. Due to numerical rounding or message timing issues, the robot may not move in a perfect square. Any code that can move the robot in the general shape of a square will be considered valid.

Deliverables

To complete the lab you must submit via canvas a zip file containing your source code and a screen recording of your program execution. The screen recording should involve (i) the robot moving in a square, (ii) output of the *rostopic list* command and (iii) output of the *rostopic echo* command.