Jackie Scanlon
1/23/19
PA1 for Algorithmic Robotics

Part 1: A Few Things to Remember

I added the sourcing to my .bashrc file. Now upon opening a new terminal window, I get the following for these commands:

```
jackie@jackielaptop:~$ printenv | grep ROS
ROS_ROOT=/opt/ros/kinetic/share/ros
ROS_PACKAGE_PATH=/home/jackie/catkin_ws/src:/opt/ros/kinetic/share
ROS_MASTER_URI=http://localhost:11311
ROS_VERSION=1
ROSLISP_PACKAGE_DIRECTORIES=/home/jackie/catkin_ws/devel/share/common-l
isp
ROS_DISTRO=kinetic
ROS_ETC_DIR=/opt/ros/kinetic/etc/ros
```

Part 2: The "Hello World!" Program

I created the package `hello` and edited `package.xml`. The entire package is in the zip folder in a folder called `hello`. `package.xml` is shown here for convenience.

I ran `roscore` in one terminal. Then I ran

```
python hello.py
```

I got

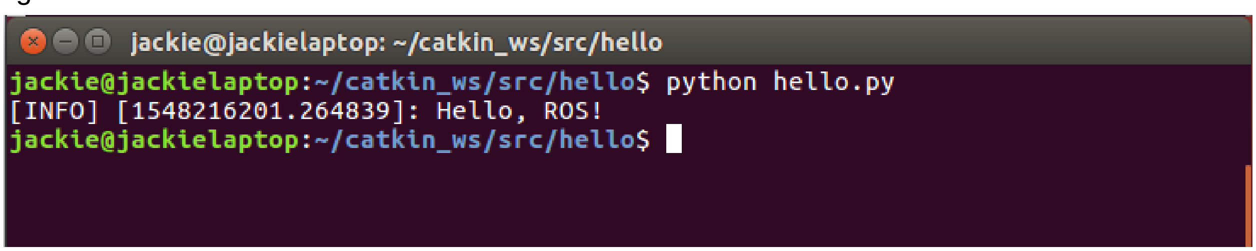


Figure 1: *Output from* `hello.py`

Basically, the node sends the one message and then quits. So by the time I run

```
rosnode list
rostopic list
```

The node and from `hello.py` doesn't show up, because by the time I type those into the terminal window `hello.py` is done.

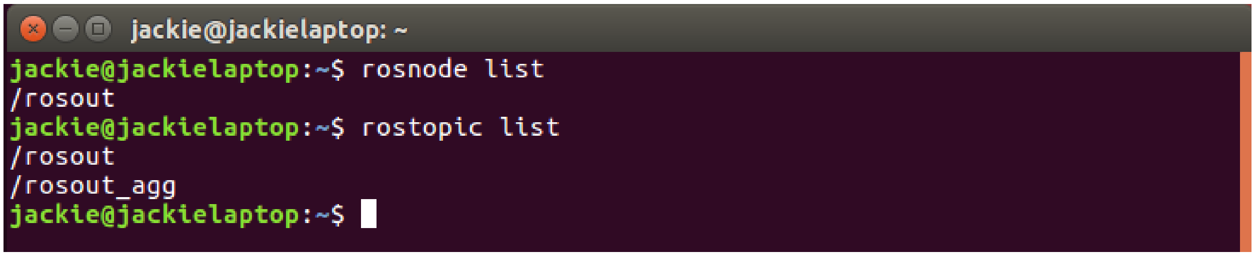


Figure 2: *Output from* `rosnode list` *and* `rostopic list`

If it didn't quit right away, I believe I would be able to see a node called `/hello_ros`.

Part 3: A ROS Publisher Program

I ran `pubvel.py` while turtlesim was running.

1. Describe the behavior of the turtle under the control of the original `pubvel.py`*.*

The turtle moves around randomly. It quickly got stuck in a corner.

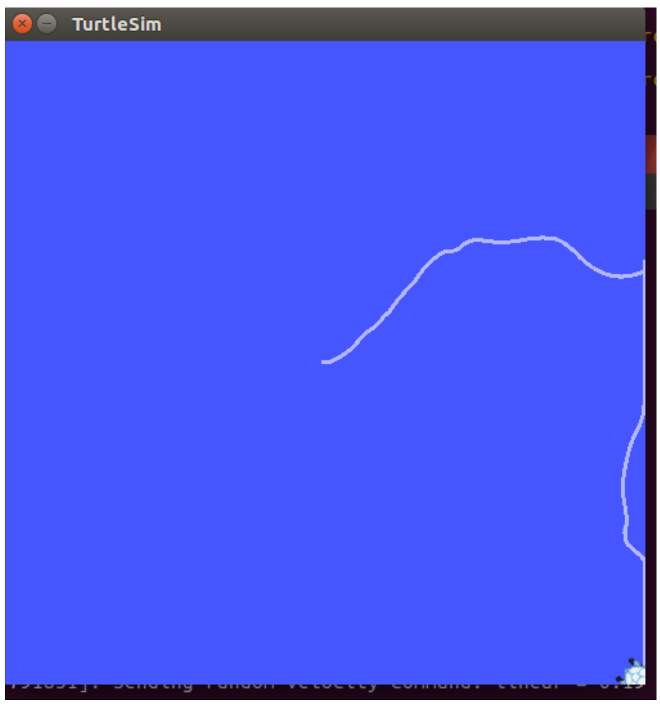


Figure 3: *Output from original pubvel.py*

2. Modify `pubvel` *to make the turtle move on a circle. How could you make the circle larger?*

I modified the code in `pubvel` to set the linear and angular velocity to 1. The code is in the zip folder.

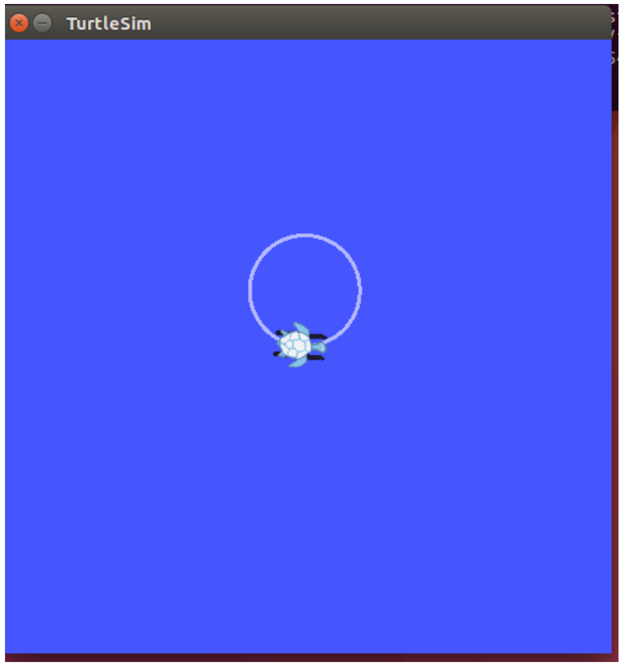


Figure 4: *Output from modified `pubvel.py`*

To make the circle larger, the linear velocity can be increased or the angular velocity can be decreased. For example, setting `msg.linear.x = 2,` I got the following output:

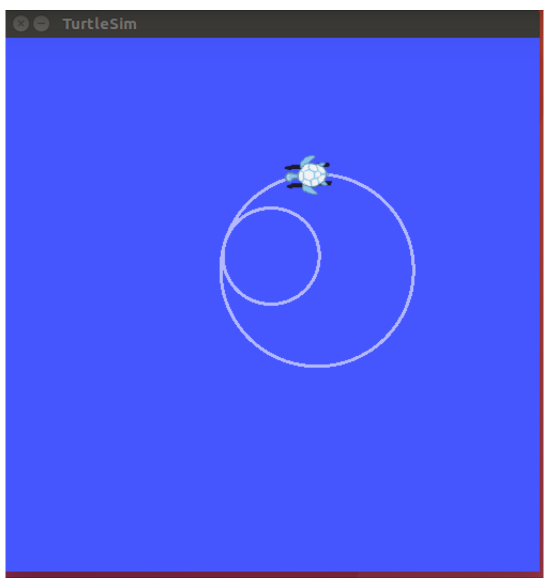


Figure 5: *When linear velocity is increased, the turtle traces a larger circle.*

3. What is the code's intended rate of the publishing loop in `pubvel`?
Line 17 shows us:

```
r = rospy.Rate(2)
```

This means that the node will attempt to publish the velocity command at a rate of 2 Hz, or every half second.

4. What is the actual rate of publishing during execution? (The command `rostopic hz topicname` will be helpful)

It's almost exactly 2Hz/.5 seconds:

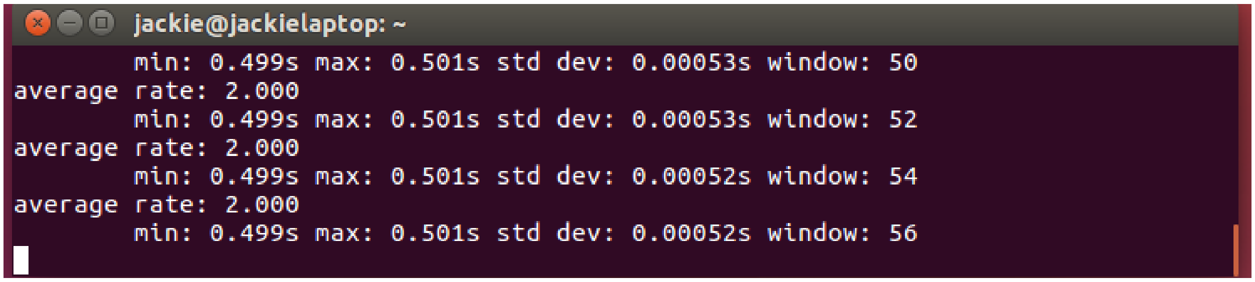


Figure 6: *Actual publishing rate on the `turtle1/cmd_vel` topic.*

Part 4: A ROS Subscriber Program

I ran subpose.py while turtlesim was running.

5. What is the purpose of `rospy.spin()` in this code?

`rospy.spin()` stops the node from exiting, so it will keep performing all its functionality until otherwise shutdown. In this particular code, it means that it will keep listening for messages on `turtle1/pose` until otherwise shutdown.

6. Use a ROS bag to collect the twist messages and plot them with rqt plot. Plot x-position vs time and x-velocity vs time. Based on your plots, comment on whether turtlesim correctly computes position from velocity.

First, I plotted the twist messages:

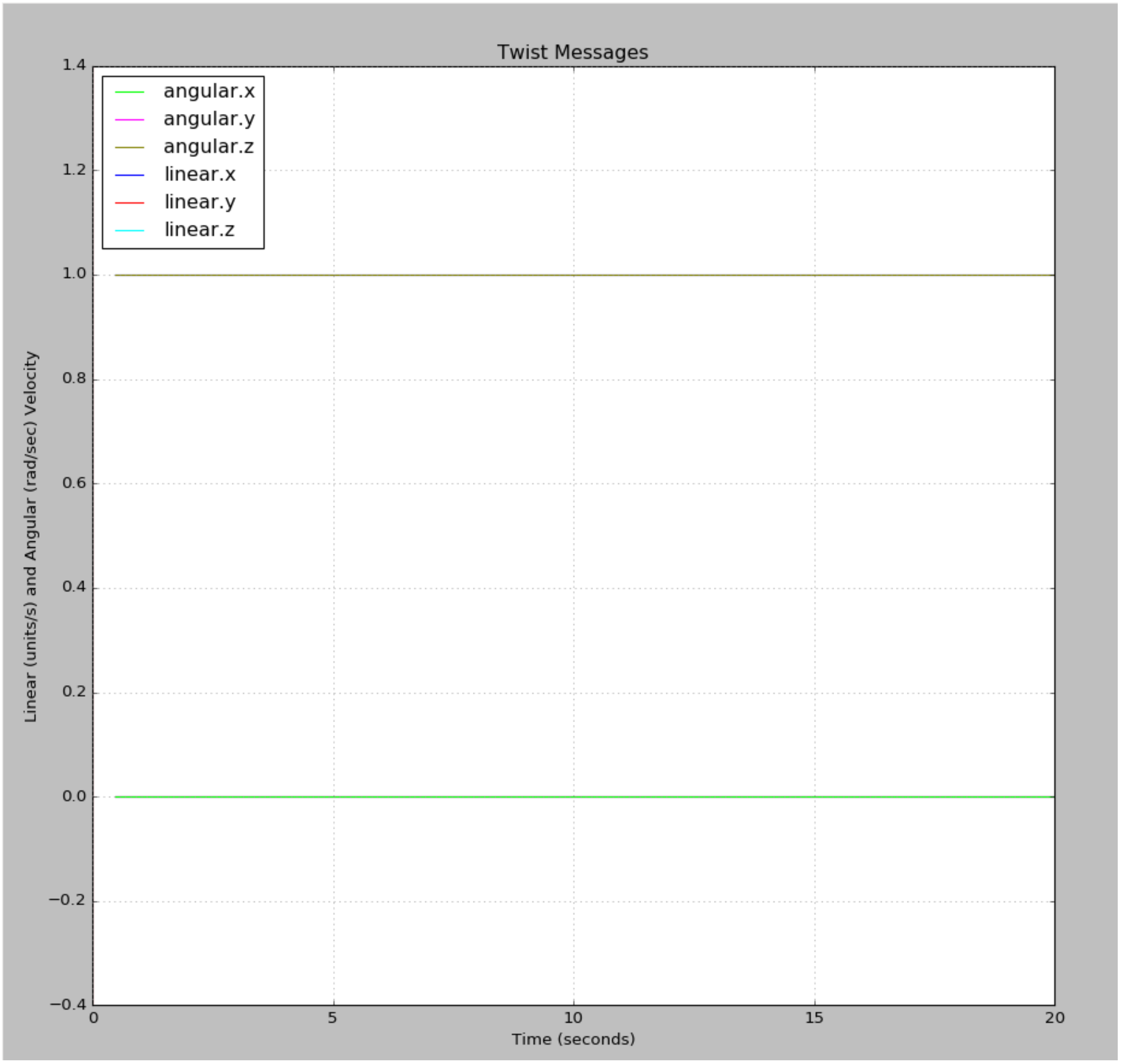


Figure 7: *The Twist messages. Essentially linear.x and angular.z are both at 1, and the rest are at 0. This matches what was programmed in `pubvel.py`.*

Next I plotted x-velocity and x-position vs. time.

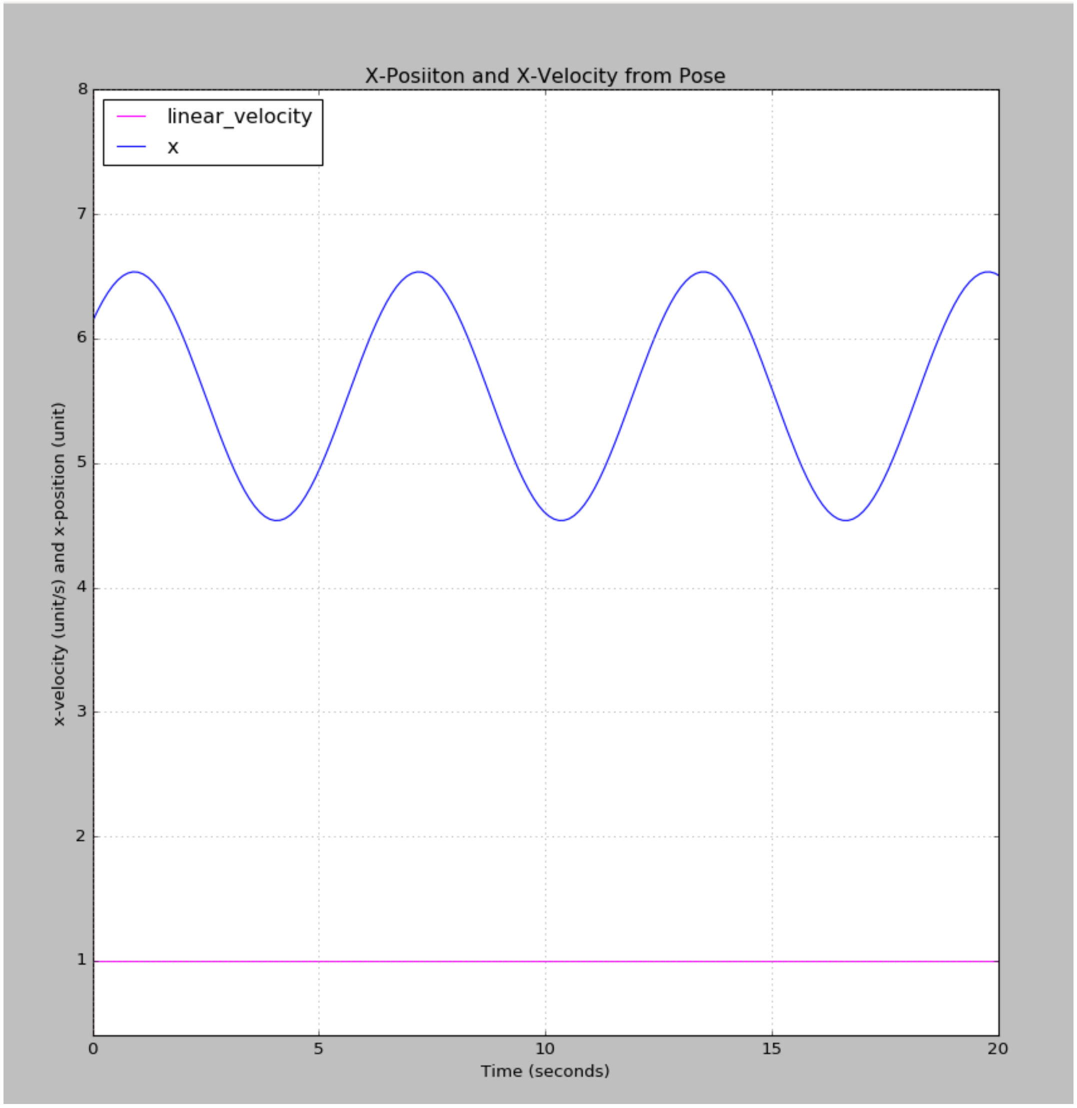
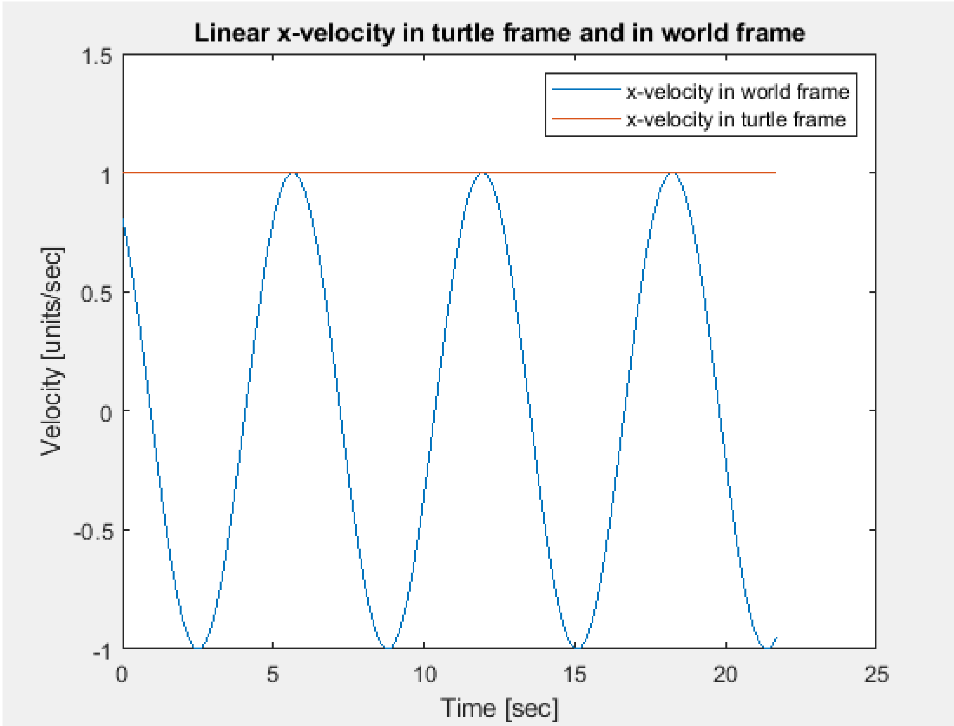


Figure 8: *The x-position and x-velocity as calculated by `Pose`.*

The velocity published by the Twist message (Figure 7) and then velocity calculated by `turtlesim` and published as part of Pose (Figure 8) are the same, 1 unit/s.Thus `turtlesim` calculates these correctly.

We can also view the velocity in the world frame rather than in the turtle frame. The area under the curve is equal to the position as calculated by turtlesim:



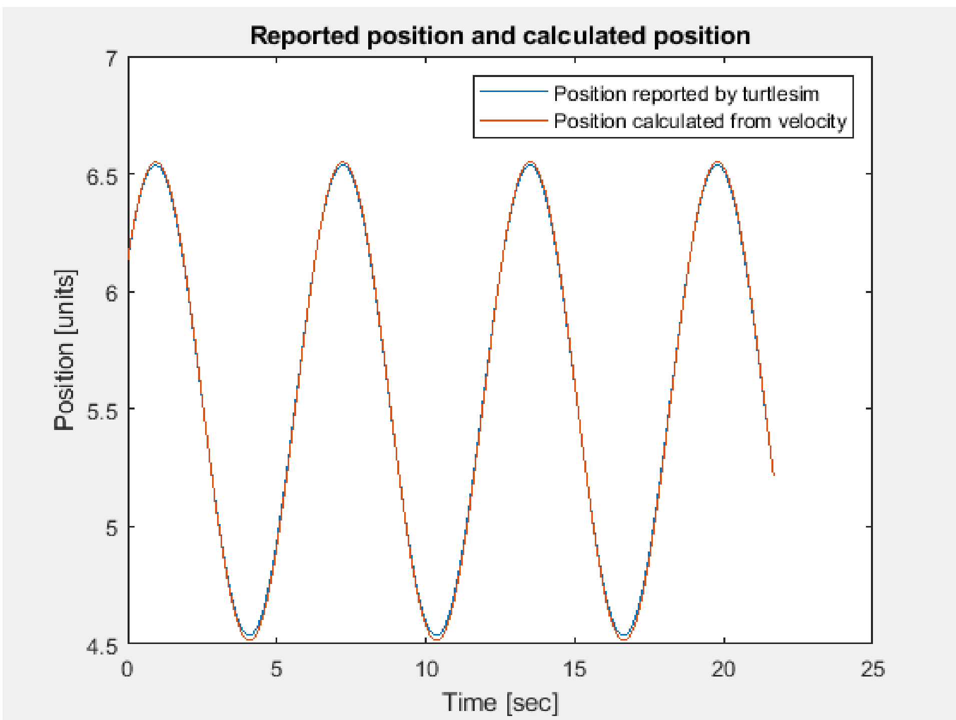


Figure 9: *World frame velocity is used to show that the area under this curve is equal to the position.*

The code to create these graphs is in PA1.m in the assignment folder.

7. After running the simulator, use rqt plot to plot y-position vs x-position. Compare this with a screenshot from the turtle simulator to show that your plot is correct.
I used `rqt_multiplot`.

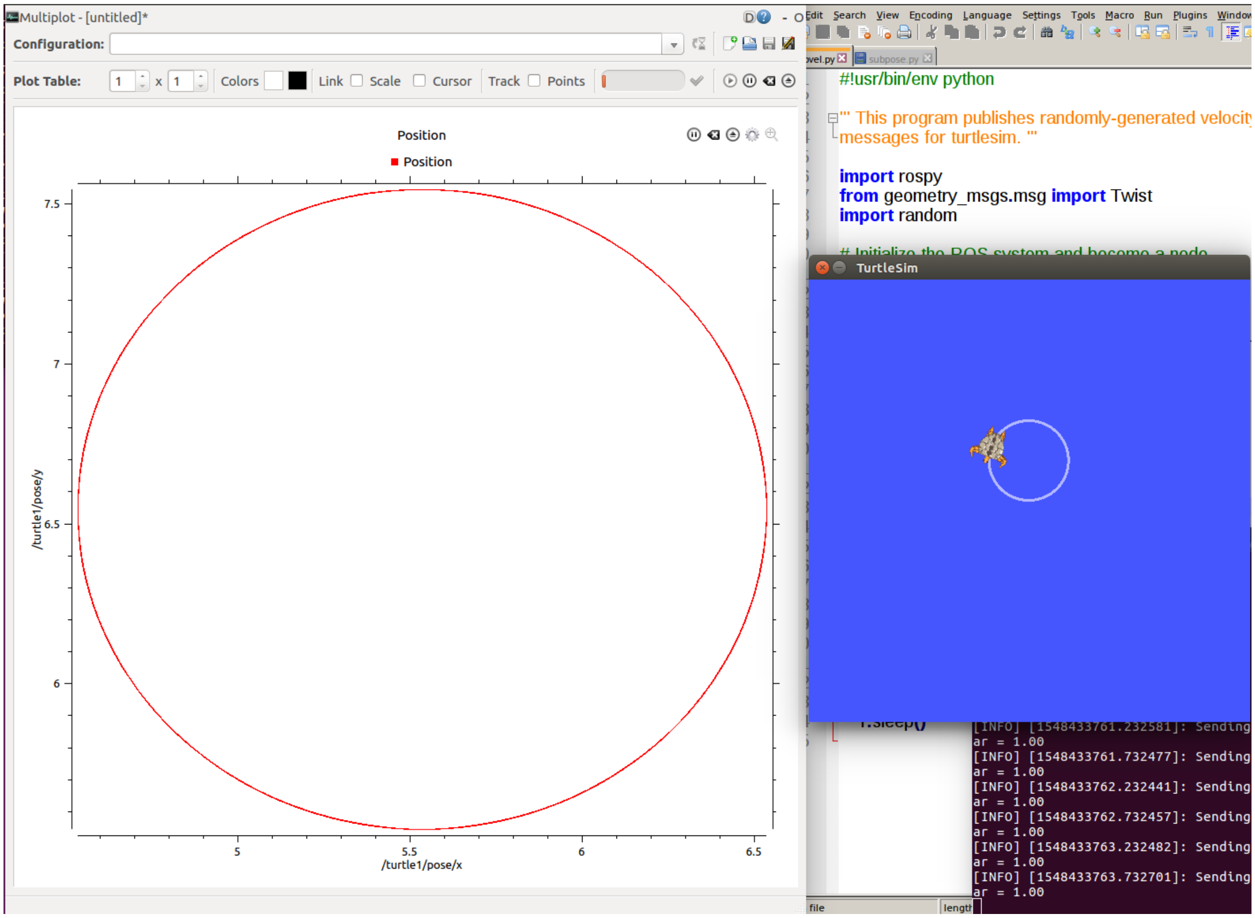


***Figure 10:** Plot of x vs. y using `rqt_multiplot`, compared to `turtlesim`.*

Useful things I learned:

- I learned that the Pose message publishes linear and angular velocity, not just position and orientation
- Learned how to use `rqt_multiplot` to plot anything on the x-axis, not just time as `rqt_plot` permits
- Used the command `roscd` a lot to navigate quickly to my `hello` package
- Learned how to import bag files into Matlab
- Learned how to take screenshots on Ubuntu using the built in screenshot tool