AA 274A: Principles of Robot Autonomy I
Section 3 (In-person): Introduction to Turtlebot Hardware and Gazebo

Our goals for this section:

1. Become familiar with the Turtlebot hardware.
2. Gain a basic understanding of the Turtlebot software.
3. Use basic tools for interacting with the Turtlebot.

1 The Turtlebot Hardware

Welcome to the robot portion of the course! In front of you is a surprisingly expensive robot. *SO PLEASE BE CAREFUL WHEN HANDLING IT OR MOVING ANY OF THE WIRES!*

2 The Turtlebot Software

Most of the forward-facing Turtlebot software you will work with is located in the `asl_turtlebot` repository on Github. To get it, go to `~/catkin_ws/src` on the provided laptop and if the `asl_turtlebot` repository doesn't already exist, run:

```
git clone https://github.com/StanfordASL/asl_turtlebot.git
```

or if the repo does exist, then `cd` into the directory and run:

```
git pull
```

to update to the latest code.

Since we downloaded a new catkin package, we need to rebuild the workspace by running the following from the `~/catkin_ws` directory.

```
catkin_make
```

2.1 Turtlebot bring up

First, we must take some steps to configure the laptop in order to be able to connect to a TurtleBot. You will see *rostb3.sh* and *roslocal.sh* in the `asl_turtlebot` folder. These files are important for telling your computer where roscore lives. Take a look at the files. Can you figure out roughly what is it doing?

We will now set up these scripts so it's easy to switch between them.

1. Connect to the correct network. (The TA will tell you which one it is.)
2. Edit `rostb3.sh` accordingly.
3. Open your `.bashrc` with a text editor. All the shell commands in this file will get run whenever you open a terminal. Add the following lines to the end of the file:

   ```
   alias rostb3='source ~/catkin_ws/src/asl_turtlebot/rostb3.sh'
   alias roslocal='source ~/catkin_ws/src/asl_turtlebot/roslocal.sh'
   export TURTLEBOT3_MODEL=burger
   ```

   IMPORTANT: This will create an alias for rostb3 and roslocal. If roscore is to run on the TurtleBot, and you want to run nodes from your computer (not ssh), you must type rostb3 EVERY TIME you open a terminal window. Otherwise, if you want to run things locally on your machine, you should run roslocal.
4. For these modifications to take effect in the current terminal, run:

   ```
   source ~/.bashrc
   ```

Next, in a terminal window, ssh into the TurtleBot using:

```
ssh aa274@<TurtleBot Name>.local
```

with the password `aa274`. This remotely logs into the onboard robot computer. The necessary ROS packages and drivers for TurtleBot operation have been pre-installed so we can go ahead and run:

```
roslaunch turtlebot3_bringup turtlebot3_core.launch
```

to launch core packages to start up the TurtleBot.

Problem 1: Once this is all running, which rostopics are available? Paste this list in your submission.

3 TurtleBot Teleoperation

Now, let's explore teleoperation with the TurtleBot.

1. ssh into the TurtleBot from another terminal window. We can start exploring the existing ROS topics. What are all the messages that are being published right now? In particular, look at the odom topic. What is the message type being published to this topic and what information is contained within these messages? HINT: rostopic info odom might help.
2. In a new (ssh-ed) terminal window, begin teleoperating the robot by running:

   ```
   roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
   ```

3. Try to teleop the TurtleBot back to $(0, 0, 0)$.

Problem 2: What is the message type being published to `odom` and what information is contained within these messages?

3.1 Pub to cmd_vel

Using our code from last week's section, create a publisher that publishes to the `cmd_vel` topic and sends a zero velocity signal at every timestep. The skeleton code for this included in this week's code in the `vel_publisher.py` file. In particular, you should send out a message of type `geometry_msgs/Twist`, with information for how to populate it available online. Some resources that help are the ROS documentation on it as well as our own TurtleBot code (look at line 155).

Problem 3: Paste your code in your submission, as well as any of its running output.

3.2 Sub to odom

Similarly, create a subscriber that subscribes to the odom topic and prints out what it receives. The skeleton code for this is located in the `odometry_subscriber.py` file.

Problem 4: Paste your code in your submission, as well as any of its running output.

4 Transfer to Genbu

Because the final project this year will be remote (in simulation), we need to know how to run everything on the hardware *and* simulated on Genbu.

The following commands run a simulated version of what you just did on the robot. See `s3_virtual.pdf` for detailed instructions.

To get the `asl_turtlebot` repository on your Genbu account, go to `~/catkin_ws/src` and run:

```
git clone https://github.com/StanfordASL/asl_turtlebot.git
```

Since we downloaded a new catkin package, we need to rebuild the workspace by running the following from the `~/catkin_ws` directory.

```
catkin_make
```

4.1 Turtlebot bring up

Once logged in to Genbu, start roscore:

```
roscore -p $ROS_PORT
```

In a new terminal window, run the Gazebo environment:

```
roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

4.2 Turtlebot teleoperation

1. In a new terminal window, begin teleoperating the robot by running:

   ```
   roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
   ```

2. Try to teleop the TurtleBot back to $(0, 0, 0)$.

4.3 Genbu Cleanup

When you're about to log out, please shut down all of your running processes (like roscore or any publishers/subscribers) and clean up your catkin workspaces for the next groups. In particular, commit and remove the code you wrote for the section as well as any catkin packages you created for the section within `catkin_ws/src`.