AA 274: Principles of Robotic Autonomy
Section 3: Intro to Hardware and Turtlebot Setup

Our goals for this section:

1. Familiarize with the Turtlebot hardware.
2. Gain basic understanding of the Turtlebot software.
3. Build basic tools for interacting with the Turtlebot.

1 The Turtlebot Hardware

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 copy it, go to */catkin_ws/src* in your VM and enter:

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

Then build this library by running:

```
cd ~/catkin_ws && catkin_make
```

2.1 Turtlebot bring up

First, we must take some steps to configure the VM 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?

2. Connect to the correct network. (The TA will tell you which one it is.) 3. Edit rostb3.sh accordingly. 4. Edit your .bashrc. Whenever you open a terminal, it runs all the commands in this file

1. Copy those files in your home directory.
2. Connect to the correct network. (The TA will tell you which one it is.)
3. Edit rostb3.sh accordingly.
4. dit your .bashrc. Whenever you open a terminal, it runs all the commands in this file

   ```
   gedit ~/.bashrc &
   ```

   and add the following lines to the end of the file:

```
alias rostb3='source ~/rostb3.sh'
alias roslocal='source ~/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.

5. 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.

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 the same (ssh-ed) terminal window, begin teleoperating the robot by running:

```
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

3. Now try teleoperating the TurtleBot, but without ssh-ing into the TurtleBot. How would you control it from your machine? Afterwards, teleop the TurtleBot back to (0,0,0)

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 velocity_publisher.py file

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.

Please show your TA the output of your odometry subscriber script.

3.3 Implement skeleton of pose controller node