

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:

```
1 | # run this command on the laptop
2 | git clone https://github.com/StanfordASL/asl_turtlebot.git
```

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

```
1 | # run this command on the laptop
2 | 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.

```
1 | # run this command on the laptop
2 | 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 `rostop3.sh` and `rostopic.sh` in the `asl_turtlebot` folder. These files are important for telling your

computer where roscore lives. Specifically, for the laptop to communicate (send/receive messages) with the robot, it needs to know the network address of the robot. To do so, three environment variables are important: `ROS_MASTER_URI`, `ROS_HOSTNAME`, and `ROS_IP`.

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 `rostdb3.sh` accordingly: define `TURTLEBOT_NAME` at the start of the script. The name of your turtlebot is printed on a sticker on the physical robot.
3. Source `rostdb3.sh`.

```
1 | # run this command on the laptop
2 | source rostdb3.sh
```

4. 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:

```
1 | # run these commands on the laptop
2 | alias rostdb3='source ~/catkin_ws/src/asl_turtlebot/rostdb3.sh'
3 | alias roslocal='source ~/catkin_ws/src/asl_turtlebot/roslocal.sh'
4 | export TURTLEBOT3_MODEL=burger
```

The `TURTLEBOT3_MODEL` should remain `burger`, do not change this. **IMPORTANT:** This will create an alias for `rostdb3` and `roslocal`. If roscore is to run on the TurtleBot, and you want to run nodes from your computer (not ssh), you must type `rostdb3` 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:

```
1 | # run this command on the laptop
2 | source ~/.bashrc
```

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

```
1 | # run this command on the laptop
2 | ssh aa274@<TurtleBot Name>.local
```

with the password `aa274`. You don't need the angle brackets (`<>`). 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:

```
1 | # run this command on the robot
2 | 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.

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

```
1 | # run this command on the laptop
2 | roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

Remember: every time you open a new terminal window on the laptop and want to send ROS commands to the robot, you must type `rostop` first.

- 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 Hardware Shutdown

When you have completed all of the hardware tasks, run the following commands in your ssh-ed terminal window:

```
1 | # run this command on the robot
2 | sudo shutdown -h now
```

with the same password you used to log in: `aa274`. This should log out all windows that were ssh-ed into the robot. Wait 4-5 seconds, then flip the power switch on the powerboard to "OFF".

5 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 only files you need from the lab laptop are the subscriber and publisher that you wrote. Genbu should be accessed from your own personal laptop.

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:

```
1 | 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.

```
1 | catkin_make
```

5.1 Turtlebot bring up

Once logged in to Genbu, start roscore:

```
1 | roscore -p $ROS_PORT
```

In a new terminal window, run the Gazebo environment:

```
1 | roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

5.2 Turtlebot teleoperation

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

```
1 | roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

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

5.3 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`.