CSCE 574 – Robotics – Fall 2018

Courtesy of Alberto Quattrini Li.

Notes on the Turtlebot 2

This document contains some details on how to use the Turtlebot 2 robots. For any question, please email the instructors.

1. Important Documents

These notes are intended to help you get started, but are not even close to being a replacement for the real documentation of the robot. The most important reference to consult can be found here:

http://wiki.ros.org/Robots/TurtleBot

A guide that shows how to work with the robot on Ubuntu 14.04 can be found here:

http://learn.turtlebot.com/

An important node to be used for the RGBd sensor (Microsoft Kinect) is the following:

http://wiki.ros.org/depthimage_to_laserscan

2. Lab facilities

The robots are picked up by each group by contacting the instructor and will be stored in Swearingen 1D49. Access to this room is controlled using a combination lock, for which the combination will be given in class.

As said during the class, the best way to work with the robots is to use your own laptops to write the program and then upload it to the Raspberry Pi mounted on the robots. The desktop computers in the lab, which can be accessed with your usual CSE department credentials, can also be used to write the program (contact Ryan Austin (Systems Administrator) if you have trouble logging in). However, to load the program to the Raspberry Pi an external laptop should be used.

Please ensure that the robots batteries are charged, by returning each robot to the charging station when you're done working with it.

3. About the robot

As shown in the class, the Turtlebot 2 robot has the following components:

- Kobuki mobile base with internal battery,
- A Kinect sensor that collects RGBd images,
- A Raspberry Pi 3 with external battery where the USB cable that goes to the mobile base and the USB cable for the Kinect connect to.
- has no user-accessible computer. Instead, it has a Raspberry Pi B+ mounted on top of it, which is connected to the robot by a serial cable. As such, your programs need to be uploaded and run on the Raspberry Pi through your computer, which will be connected with the provided LAN cable.

4. What you get

When you pick up the robot, you will get:

- The Turtlebot 2, with the components as described in Section 3,
- A charger for the Kobuki mobile base

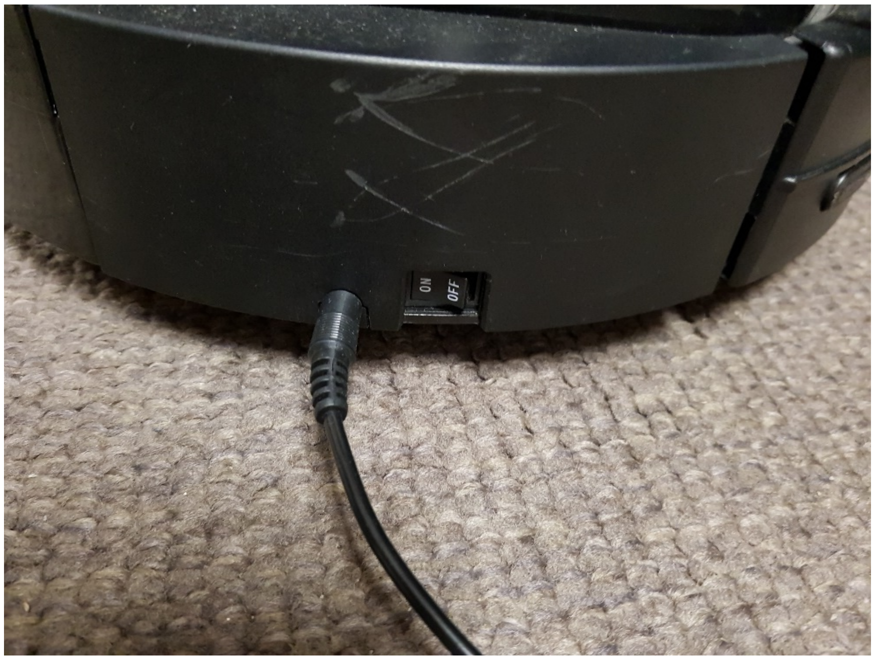
5. How to charge it

When the robot is not used, the batteries should be put in charge so that you are not having problems next time to work with the robot.

There are two batteries on the robot that need to be charged:

- The Kobuki mobile base with the provided charger,
- The external battery for the Raspberry Pi.

The first one, can be charged by connecting the cable on the corresponding port, as shown in the



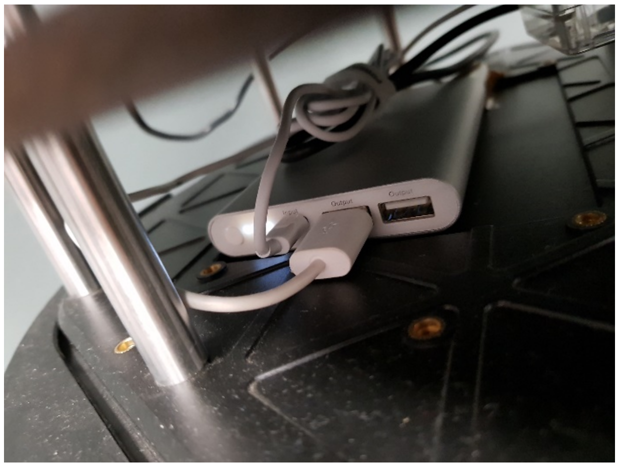
following picture

The second one, requires that two USB cables are connected together, as shown in the following picture:

6. How to turn on the robot

This is the procedure to turn on and use the robot:

- Disconnect the two USB cables that recharge the external bettery
- Disconnect the charger from the Kobuki mobile base
- If the Raspberry Pi did not turn on---namely no LED is visible from the Raspberry Pi---, press the button on the battery, shown in the following picture.



After 1 minute, the Raspberry Pi should be operational and ready to be used.

7. Writing programs to control the Robot

Here are the basic steps to write and execute a program that is going to be running on the Raspberry Pi:

- Use your favorite text editor to write or modify your ROS program.
- Once the robot is on as specified in Section 6, connect the provided LAN cable from the Raspberry Pi to your computer.
- To simplify the operation, first `ssh` to the Raspberry Pi to get the IP address of the WiFi interface (IP address of the Raspberry Pi: `192.168.1.1`; username: `pi`; password: `raspberry`).
  ```
  ssh pi@192.168.1.1
  ```
- Get the IP address by typing

```
ifconfig
```
The output includes the IP address of the WiFi interface, that is connected to uscguest. As such, your computer should be connected to uscguest.

- Write down the IP address and exit the ssh session, by pressing ctrl+d.
- Unplug the LAN cable from your computer.
- Copy the program to the Raspberry Pi by using scp (IP address of the Raspberry Pi: `<IP address identified>`; username: `pi`; password: `raspberry`).
  e.g., using a Windows-based machine, you can use https://winscp.net/eng/download.php application to transfer the data from your laptop to your computer
  OR
  using a Unix-based machine, to copy a folder called `project_03` in the `catkin_ws/src/` directory of the Raspberry Pi
  ```
  scp -r project_03 pi@<IP address identified>:~/catkin_ws/src
  ```
- Connect to the Raspberry Pi by using ssh.
  e.g., using a Windows-based machine, you can use PuTTY to access the Raspberry Pi
  http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
  OR
  using a Unix-based machine,
  ```
  ssh pi@<IP address identified>
  ```
  and input the password.
- Use `screen` to start a shell window within the ssh session so that the shell is active even through network disruptions.
- Run the main nodes for the Kobuki base
  ```
  roslaunch turtlebot_bringup minimal.launch
  ```
- Exit from the `screen` holding ctrl and pressing a and then d
- Run another `screen`
- Run the nodes for the Kinect
  ```
  roslaunch turtlebot_bringup 3dsensor.launch
  ```
- Exit from the `screen` holding ctrl and pressing a and then d
- Run another `screen -S project`
- Run your program using `rosrun`.
  e.g.,
  ```
  rosrun project_03 project_03_node
  ```
- You can leave the session open, or you can close the screen, again with ctrl+a and d; and exit the ssh session.

To stop the robot:

- Connect again to the robot with ssh through WiFi.
- Resume the screen about the project
  ```
  screen -r project
  ```
- Terminate your program by pressing Ctrl+c.
- Turn off the Raspberry Pi: `sudo halt`.
- Wait 1 minute and then follow the procedure to charge the robot.

8. Coding style

The program should be written by you and the code should be following an adequate and consistent coding style. In particular, the following coding style from Google is adopted https://google.github.io/styleguide/pyguide.html. This general guidelines should drive your code writing:

- **Don't repeat yourself.** Source code in which the same information appears in multiple places can be difficult to understand and, therefore, to debug.
  a) If you find yourself repeatedly copying and pasting a block of code, you should probably write a function to abstract that block.
  b) If your code contains explicit "magic numbers" that are repeated or copied from the documentation, you should probably replace those explicit numbers with named constants.
- **Write for your audience.** The "audience" for your source code includes, as a minimum (a) you and your group members, (b) your instructor, and (c) your interpreter. Therefore, your goal should be to write code that is easy for humans to read and understand. Important steps toward achieving this goal include:
  a) Using names that accurately describe the data or code that they name.
  b) Writing comments for each function and section of code explaining its purpose and clarifying anything that's not obvious about how it works.
  c) Using a consistent style for indentation and spacing, including indentation of nested blocks.

Lastly, please don't fall into the trap of writing bad code with the intention of "fixing" it to meet these general guidelines after it works. By doing that, you'll experience the pain that goes with these constraints but miss out on the benefits.

The use of external software is allowed only if:

- The use of it does not trivialize the assignment.
- The source should be clearly reported.

N.B.: Please remember to always **backup** your code, the best way being using a version control system (e.g., git, svn, or hg)!

N.B.2: As a suggestion, ensure that you modify the code in one place, e.g., your laptop, and synchronize it with the Raspberry Pi on the robot. If you modify the code on the Raspberry Pi, you might have inconsistent copies of your code across the computers and possibly it could lead to overwriting the files and thus losing important parts of code.

9. Communicating with the robot

As ROS is used, there are the topics that should be used to send velocity commands or read sensor data. In the following a list of the main topics that are useful to know. As a reminder, you can use

```
rostopic list
```

And

```
rostopic info
```

To get more information about the available topics and the type of messages required.

Control

The node for controlling the base has a multiplexer node that has 3 different topics to send velocity commands. In particular the topics are the following:

- `/cmd vel mux/input/navi`
- `/cmd vel mux/input/teleop`
- `/cmd vel mux/input/safety controller`

The rationale is that for example you would like to take teleoperative control of the robot to override automatically generated commands sent by your software.

Odometry

Raw data from the odometers and the IMU are available in the following topics:

- `/odom`
- `/imu/data`

Kinect

Data that are used as if the Kinect is a laser range sensor is available at the following topic:

- `/scan`