Introduction to ROS with TurtleSim

The purpose of this lab is to guide you in setting up your computer and getting familiar with the Linux operating system and ROS.

First you will install a Ubuntu virtual machine on your computer. Then you will try to run some basic commands using the Linux terminal and Sublime editor. Finally you will begin to get familiar with the Robotic Operating System (ROS), which you will use extensively throughout the remainder of the course. While you will be able to submit some future lab assignments as a group, **every student must submit this lab assignment individually.**

Task 1 Installing Ubuntu virtual machine with Robot Operating System (ROS)

1. In this class we will work mainly with **ROS** (Robotic Operating System). ROS provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is designed to work with Ubuntu operating systems, which runs on the BARC's Odroid. In order to allow you to become familiar with ROS outside of the BARC environment, we provide a Ubuntu virtual disk file with ROS installed, which you can run from your non-Ubuntu laptop.

   (a) Setting up the virtual machine
      i. Download the virtual machine (VM) onto a laptop by clicking **here**.
         The link consists of one file which is about 8 GB. It can be downloaded directly onto an external memory stick, if your own laptop is low on space.
      ii. Download and install the software Oracle VirtualBox onto the same laptop by clicking **here**.
      iii. Open Oracle VirtualBox and click on **New**.
      iv. Choose a name for the VM and set the Type to **Linux** and the Version to **Ubuntu 64-bit**. Then click **Continue**.
      v. Set the memory size to at least 1 GB (4 GB is recommended if the memory size on your computer is 8 GB) and click on **Continue**.
      vi. For the Hard Disk select **Use an existing virtual hard disk file** and use the `ME131.vmdk` file that was downloaded in Step 1. Then select **Create** to finish setting up the VM.
      vii. Launch the VM by clicking **Start** in VirtualBox. You should have a Ubuntu system now that is all setup and ready to go. The VM Password and Username are both **student**
      viii. **Note:** If, after following these steps, your virtual machine's Ubuntu program is 32-bit instead of 64-bit, come and see us at the beginning of your lab section. This is an occasional machine-dependent error that can be fixed by changing a particular setting.

   Some users have noticed that the VM can run a little slowly – note that you will only rely on the VM for the next two weeks, and its performance will not affect your future BARC algorithm runtimes at all.

Task 2 Introduction to Linux and the Sublime editor

1. The easiest way to interface with the Ubuntu operating system is through the command line interface (CLI). The following exercises are adapted from [1] and are designed to help you get acquainted with the CLI and the Sublime text editor. We will use these tools throughout the rest of this course, so feel free to check out other resources as well until you are comfortable with the general workflow.

   (a) Navigating with the CLI

      i. In your VM window, open a new command line terminal by clicking the **Terminal** icon at the bottom of the left hand side bar
      ii. Type `$ pwd`. The `pwd` command outputs the full pathname of your current working directory, and is a helpful tool if you find yourself lost in the Linux system.
      The output you see now is your *home directory*
      iii. To list the items in your current working directory, type `$ ls`. The `$ ls` command lists all contents of your current directory whose names do not begin with a dot (so-called *hidden files*). To list all files in the directory, including hidden files, type `$ ls -a`
      iv. To make a new directory titled ME-131-work in your current working directory, type `$ mkdir ME-131-work`. Check that the directory was created as expected using the `$ ls` command. You should see your new directory listed along with the others
      v. You can change your current working directory by typing `$ cd directory-name-here`. Change your working directory to the ME-131-work directory you created in the previous step, and list its contents (this should be an empty list, as we have not created any new files within the directory)
      vi. You can refer to your current working directory using the dot operator (.). Type `$ cd .` – you should stay in your current `ME-131-work` directory
      vii. To move to the parent of your current working directory, type `$ cd ..`. Use this command to move back to the parent of `ME-131-work`, which is the system home directory you began this tutorial in
      viii. In general, you can always move back to the system home directory by typing `$ cd`. You can also refer to your home directory using the ~ character. For example, no matter your current working directory, `$ cd ~/ME-131-work` will move you back to your home directory and then into the subdirectory you created at the beginning of this exercise
      ix. Once you are comfortable with the individual steps, you can combine the above commands to save time. For example, from the home directory, use the command `$ ls ME-131-work` to list the contents of the `ME-131-work` directory. Note that this command only works in its current form because the `ME-131-work` directory is a child of your current working directory. What command would allow you to directly print the contents from `ME-131-work` if your current working directory was a different child of the home directory?
      x. Change back to your system home directory using your preferred method. Create a new text file in the directory by typing `$ cat > my-first-file.txt` and pressing `Enter`. The `$`

[1] http://www.ee.surrey.ac.uk/Teaching/Unix

`cat > file-name-here` command creates the text file and automatically moves you into the terminal's built-in text editor, so you can begin typing the contents of your new text file right away. Write a short note to yourself, and then press `Enter` and `CTRL-D` to move back to your directory. Check that your text file appears in your directory using the `$ ls` command. *Note: We will cover text editors in more detail later in the exercise*

xi. You can also use the `$ cat` command to print the contents of an existing file to the terminal. Practice this by typing `$ cat my-first-file.txt`. This command does not open the built-in text editor, but simply displays your previous file entry in the terminal

xii. The command `$ cp original-path/original-file-name new-path/copied-file-name` makes a copy of the `original-file-name` file in the `original-path` directory, and pastes it into the `new-path` directory with the name `copied-file-name`. Note that if no `copied-file-name` is provided, the file will retain its `original-file-name`. Copy your `my-first-file.txt` file into your `ME-131-work` directory. List the contents of `ME-131-work` to ensure the file was copied correctly

xiii. The `$ mv original-path/original-file-name new-path/copied-file-name` command moves the `original-file-name` file in the `original-path` directory to the `new-path` directory with the name `copied-file-name`

xiv. *****Deliverable**: Within the `ME-131-work` directory, create a new subdirectory called `bad-vibes`. Navigate into this directory, and create a new text file titled `stress.txt`. Fill the text file with a list of stresses, then exit out of the text editor. Take a screenshot of your terminal that shows that you have a text file titled `stress.txt` within a subdirectory called `bad-vibes`.

xv. We can delete a file using the command `$ rm path-to-file/file-name`, and delete a directory using the command `$ rmdir path-to-directory/directory-name`. Remove first the `stress.txt` file, and then the entire `bad-vibes` directory. Check the contents of `ME-131-work` to make sure only good vibes (and your `my-first-file.txt` file) are left

xvi. To clear the terminal screen of your previous command line inputs, use `$ clear`.

(b) Using the Sublime text editor

While you are free to use whatever text editor available on Ubuntu that you are most comfortable with, we provide a brief introduction to Sublime below.

i. Open a command line terminal, and navigate to the `ME-131-work` directory

ii. Use the command `$ subl my-first-file.txt` to open your previously created text file in the `Sublime` text editor. You should see the content you previously created in the terminal's built-in editor. Within the editor, you can can make changes to your text file as necessary

iii. Be sure to save your file by typing `CTRL-S` before closing out of the Sublime window. Sublime does not give you warnings about unsaved files!

iv. To create a new file in Sublime, use the command `$ subl new-file-name.type`. A different type of document will be created for you depending on the document type you specify at the end of the file name. For example, when you create a new python file by declaring the type `.py`, Sublime will automatically implement syntax highlighting for python code.

v. *****Deliverable**: Create a new `.py` file in Sublime, and write a python function that prints 'Hello World' to your screen. Close the Sublime editor, and run the function from your terminal using the command `$ python python-file-name.py`. Take a screenshot of your terminal showing the output.

If you would like to explore more Unix tutorials, you may find these resources helpful:

- References of Unix commands: https://files.fosswire.com/2007/08/fwunixref.pdf and http://cc.iiti.ac.in/lcommands.pdf
- Additional Unix/Linux tutorial: https://ryanstutorials.net/linuxtutorial/

Task 3 Turtlesim

1. In Task 1 you installed a Virtual Ubuntu Machine containing ROS on your computer. In this exercise, we will walk you through some of the basic functionalities of ROS so you can begin to get familiar with the tools.

   ROS is a set of libraries and tools that facilitates inter-process communication across various languages and platforms. The ROS workflow consists of three core concepts:

   - Nodes, which refer to each process involved in your robotics system (such as a sensor or actuator).
   - Topics, which are channels over which nodes can exchange messages by subscribing and publishing.
   - Messages, which are the data structures containing the information sent and received between different nodes within a particular topic.

   In this lab, you will run various turtlesim tutorials on your virtual machine. Turtlesim is a tool made for teaching ROS and ROS packages. The turtlesim package provides you with a virtual turtle robot, so you can get started learning ROS concepts without any hardware.

   (a) Installing turtlesim
   In order to use turtlesim, you need to install the turtlesim package from the ROS tutorials by typing the following in your virtual machine's terminal:
   ```
   $ sudo apt-get install ros-kinetic-ros-tutorials
   ```
   It will install the turtlesim package into the directory `/opt/ros/kinetic/lib/turtlesim` automatically.

   (b) roscore
   `roscore` is used to initialize the ROS enviroment. It is the first thing you should run whenever you want to use ROS. To initialize, simply run:
   ```
   $ roscore
   ```

   (c) rosrun
   `rosrun` allows you to use the package name to directly activate a node within a package.
   Usage: `$ rosrun [package_name] [node_name]`
   To run the `turtlesim_node` from the `turtlesim` package, open a **new terminal tab** (`CTRL-SHIFT-T`) and run:
   ```
   $ rosrun turtlesim turtlesim_node
   ```
   Say hello to your turtle!

   (d) ROS nodes
   The `rosnode [option]` command displays information about the ROS nodes that are currently running.

i. rosnode list

The `rosnode list` command lists all currently active nodes. In a **new terminal** (`CTRL-SHIFT-T`), run:

```
$ rosnode list
```

You should see that there are two nodes currently running: `rosout` and `turtlesim`, which we initialized in the previous step. Note that the `rosout` node will always run during a ROS operation, logging all nodes' debugging output.

ii. rosnode info

The `rosnode info` command returns information about a specific node. Run:

```
$ rosnode info /turtlesim
```

This command provides more information about the `turtlesim` node, including that it subscribes to the rostopic `turtle1/cmd_vel` and publishes the rostopic `turtle1/color_sensor`. (`turtle1/cmd_vel` and `turtle1/color_sensor` are nothing but rostopic names).

```
Node [/turtlesim]
Publications:
* /turtle1/color_sensor [turtlesim/Color]
* /rosout [rosgraph_msgs/Log]
* /turtle1/pose [turtlesim/Pose]
Subscriptions:
* /turtle1/cmd_vel [unknown type]
Services:
* /turtle1/teleport_absolute
* /turtlesim/get_loggers
* /turtlesim/set_logger_level
* /reset
* /spawn
* /clear
* /turtle1/set_pen
* /turtle1/teleport_relative
* /kill
```

(e) *****Deliverable**: Turtle keyboard teleoperation

Now that we have created a turtle, we need something to drive its movement. In a **new terminal**, run:

```
$ rosrun turtlesim turtle_teleop_key
```

This launches the `turtle_teleop_key` node from the `turtlesim` package. Now you can use the arrow keys of the keyboard to drive the turtle across the screen. Record a video of your computer screen which shows the turtle moving in response to your keyboard commands (instructions for Ubuntu screen recordings can be found with a Google search - use whichever process you prefer). If you can not drive the turtle, select the terminal window of the turtle_teleop_key before typing, to make sure that your keys are recorded.

(f) *****Deliverable**: rqt_graph

We can use an `rqt_graph` to dynamically see which nodes and topics are currently running. The `rqt_graph` command is part of the `rqt` package. To install the `rqt` package, run:

```
$ sudo apt-get install ros-kinetic-rqt
$ sudo apt-get install ros-kinetic-rqt-common-plugins
```

In a **new terminal**, run the `rqt_graph` node from the `rqt_graph` package by typing:

```
$ rosrun rqt_graph rqt_graph
```

If that command doesn't work, try `rqt_graph` alone. Select `Nodes/Topics (all)` from the drop-down menu. Take a screenshot, and make sure you understand what you see.

(g) ROS Topics

The `rostopic` tool allows you to get more information about individual ROS topics. You can use the help option to see all available sub-commands for `rostopic` by typing:

```
$ rostopic -h
```

We will cover the most immediately useful ones here.

i. rostopic echo

`rostopic echo` shows the data published on a topic.

Usage: `rostopic echo [topic]`

Let's look at the command velocity data published by the turtle_teleop_key node.

```
$ rostopic echo /turtle1/cmd_vel
```

You probably won't see anything happen because no data is being published on the topic. Let's make turtle_teleop_key publish data by pressing the arrow keys. Remember if the turtle isn't moving you need to select the turtle_teleop_key terminal again.

ii. rostopic list

`rostopic list` outputs a list of all topics currently subscribed to and published to, along with the number of nodes interacting with each topic by subscribing or publishing. Open a **new terminal**, and run:

```
$ rostopic list -v
```

Selecting the `-v` option causes the command to print both subscribers and publishers of all topics. To see a list of all available options, use:

```
$ rostopic list -h
```

iii. rostopic type

Different ROS nodes communicate by publishing and subscribing to messages on a topic. In order for this type of communication to work, the nodes must be programmed to send and receive messages of matching structure, or type. To check the type of message transferred through a node, we can use the command `rostopic type [topic]`. For example, type:

```
$ rostopic type /turtle1/cmd_vel
```

You should see the topic type described as `geometry_msgs/Twist`. To learn more about this Twist structure, type:

```
$ rosmsg show geometry_msgs/Twist
```

Now you should see that the Twist structure consists of six different floats, and we can ensure that publishers and subscribers are designed to work with this message structure.

iv. ***Deliverable**: rostopic pub

The `rostopic pub` command publishes our messages to a specific topic.

Usage: `rostopic pub [topic] [msg_type] [args]`

We know from the previous exercise that the `/turtle1/cmd_vel` topic expects messages to be in two column vectors with three floating point entries each. To move the turtle with a linear velocity of 2.0 and in-plane angular velocity of 1.8, type:

```
$ rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]'
'[0.0, 0.0, 1.8]'
```

As with previous commands, the `rostopic pub` command has options. The `-1` option indicates we only want to publish this one message and then exit the topic publisher. Using this option means the turtle stopped moving at our indicated velocity after a time. Try again, replacing the `-1` with `-r 1`, to send a *recurring* stream of commands at 1 [Hz]. Now the turtle walks in circles! Confirm this by typing in a **new terminal**:

```
$ rostopic echo /turtle1/pose
```

The `--` option between message type and message content indicates that the following entries are not options, but messages (since options are given in the format `- [option]`. This was not necessary for our chosen velocity, but is critical when publishing negative values. Record a video of your computer screen which shows the turtle continuously moving in a circle.
Update your `rqt_graph` by clicking the refresh button in the upper right hand side of the terminal, and take a screenshot.

v. rostopic hz
To check the rate at which data is published, we use: `rostopic hz [topic]`
Check how fast the `turtlesim_node` is publishing `/turtle1/pose` by typing:

```
$ rostopic hz /turtle1/pose
```

We see that turtlesim is publishing pose information for the turtle at 60 [Hz].

(h) *****Deliverable**: rqt_plot
The `rqt_plot` command displays a scrolling time plot of the data published on particular topics. This can be a very useful tool for debugging! We can initialize the `rqt_plot` package by typing in a **new terminal**:

```
$ rosrun rqt_plot rqt_plot
```

If that command doesn't work, try `rqt_plot` alone. A new window will pop up, and you can use the text box in the upper left corner to begin adding topics whose messages you are interested in. Begin typing `/turtle1/pose/x` into the text box, until the green plus sign lights up. Click this button to begin plotting the turtle's x-position. Also add the turtle's y-position to the plot. Take a video recording of your plot.
To remove a particular topic from the plot, type the topic name into the text box and press the red minus sign once it lights up.

(i) ROS Launch Files
The `roslaunch` tool allows you to run multiple nodes (and other launch files) at once. It also allows you to set certain parameters that you can use in those nodes. You can use the help option to see all available sub-commands for `roslaunch` by typing:

```
$ roslaunch -h
```

(j) Exercise: Let's write a launch file and python script to override the turtlesim movements. We want to make the turtle move in the reverse direction of the arrow keys.

(k) Check out more ROS turtlesim tutorials at http://wiki.ros.org/ROS/Tutorials if you'd like! When you are done exploring, use `CTRL-C` to kill the ROS terminals.