SIT310 – Task 1

Introduction to Robot Operating System (ROS)

Overview

This task will introduce you to Robot Operating System (ROS). ROS isn't really an operating system, but more a framework that provides a lightweight structure for your applications. The real power is in the large number frameworks and implementations of robotics functions (such as planning, vision, data processing). In this unit we will leverage as much of this as possible to build robot applications. This task will first point you to lots of information about ROS, then you will install ROS on a Raspberry Pi. You will then execute some demonstrations of ROS functionality. Upon completion of this task, you should begin to be familiar with ROS and what it can provide for robotics development.

Task requirements

a. You will need to have a Raspberry Pi, micro SD card, SD card reader, and access to a keyboard, mouse and display.
b. If you didn't attend the lecture this week, then either watch the lecture online and/or review the PowerPoint slides on the unit site.
c. Read the introduced concepts / theories / recommended readings below.
d. Read the task instructions
e. Complete the activities in the task guide
f. Review your progress with your lab tutor or cloud tutor.

Task guide

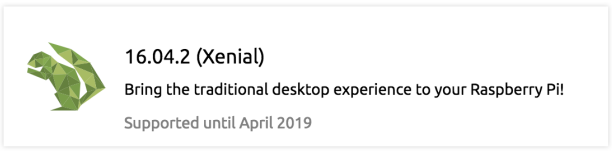
For this lab we will be heavily using tutorials from the technologies we are using. The instructions presented here are cut down versions of these tutorials to allow us to rapidly develop robotic applications. If you want further information, have a look at the "Additional resources" section below.

1. Install Ubuntu on the Raspberry Pi

    - We will be using Ubuntu Mate 16.04.2 (Xenial) as the main Operating System.
    - For this, you will need a Raspberry Pi 3 (or newer), a Power supply or other reliable USB power source, and a Micro SD card (16GB Sandisk Ultra/Extreme recommended). You will also need a screen, USB keyboard and Mouse.
    - If you have a Raspberry Pi 3 B, Download the zip (xz) of Ubuntu Mate, available here: https://ubuntu-mate.org/download/ as follows:

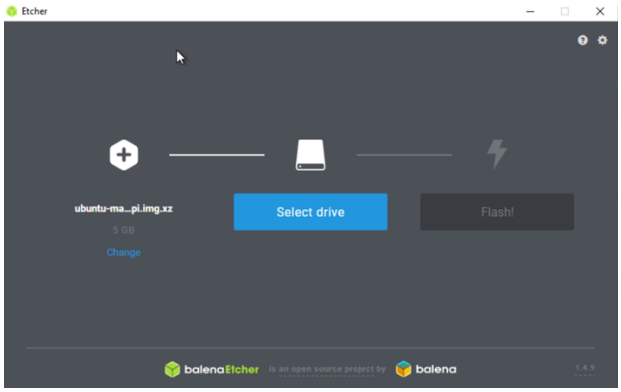
Click on this ->

Click on this to download ->

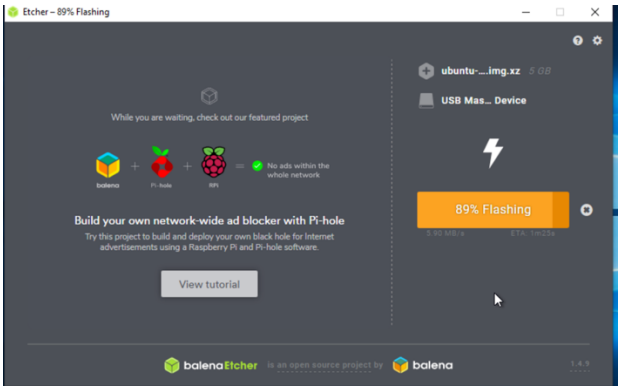


    - If you have a Raspberry Pi 3 B+, then download the zip specially prepared for the unit from the Unit site, next to this document.
    - You don't need to extract the zip file.
    - Insert your microSD card into the USB reader and insert it into your computer
    - You need to write the Ubuntu image to the SD card. The recommended way is using https://www.balena.io/etcher/ so download and install this software.
        - (If you are on MacOS or Linux, you can use dd – consult the unit chair for advice).
    - Load etcher, select your Ubuntu image, select the Micro SD card, and click flash, as follows:

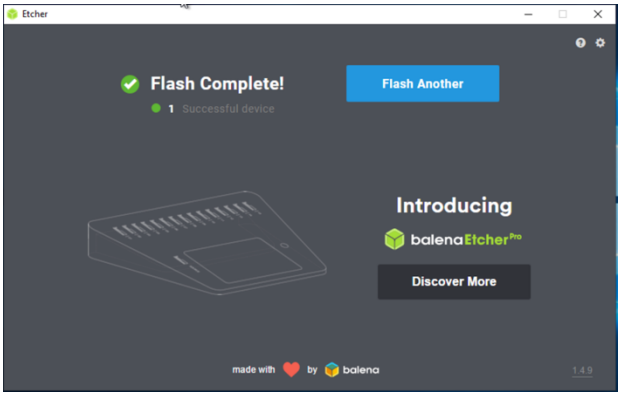
Select your image and drive ->



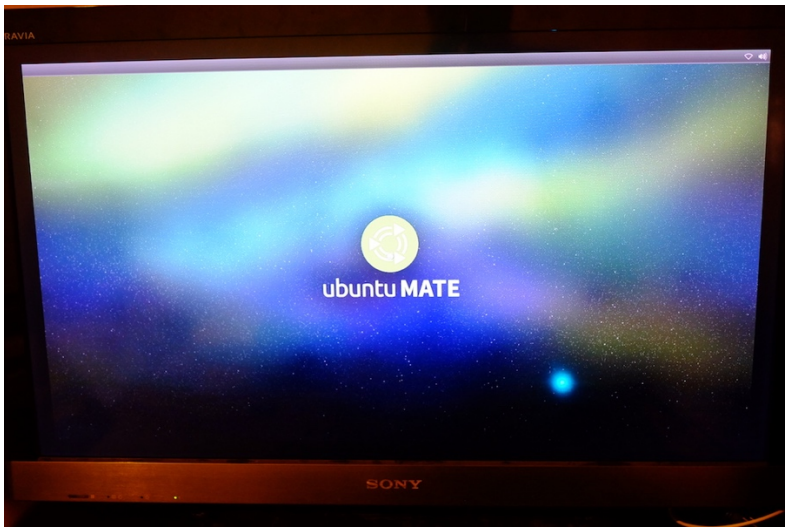
It will then flash the card ->.



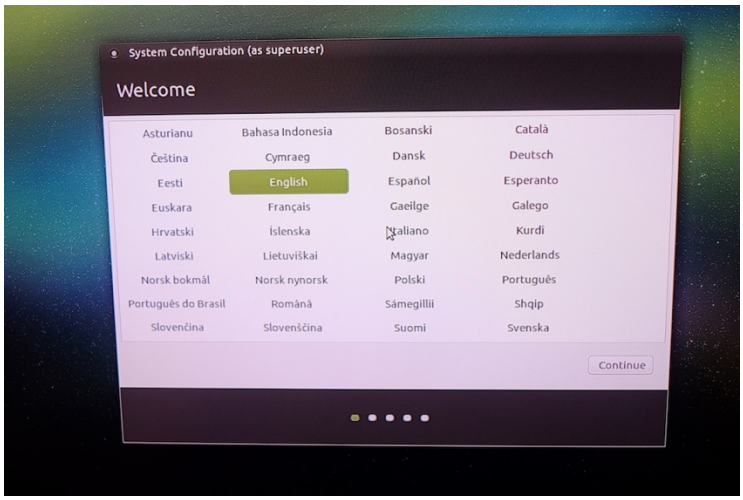
When it is complete ->



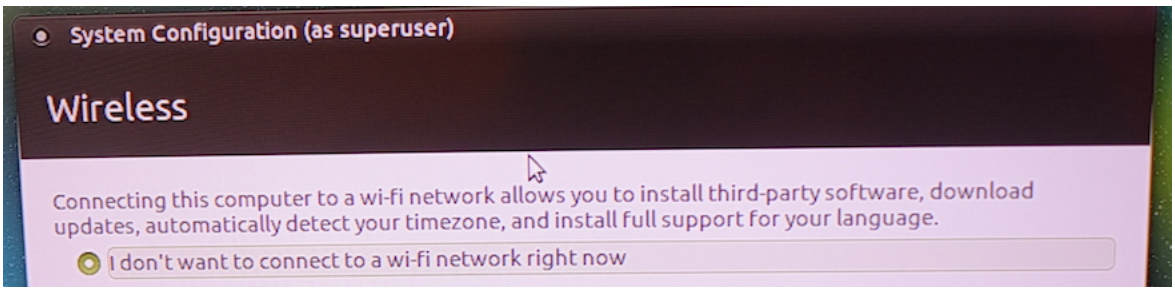
- Eject your card from your PC. put the micro SD card into the Raspberry Pi, connect it to HDMI, and USB keyboard and mouse. Plug in the power.
- Ubuntu will (hopefully) boot and present you with the following screen.



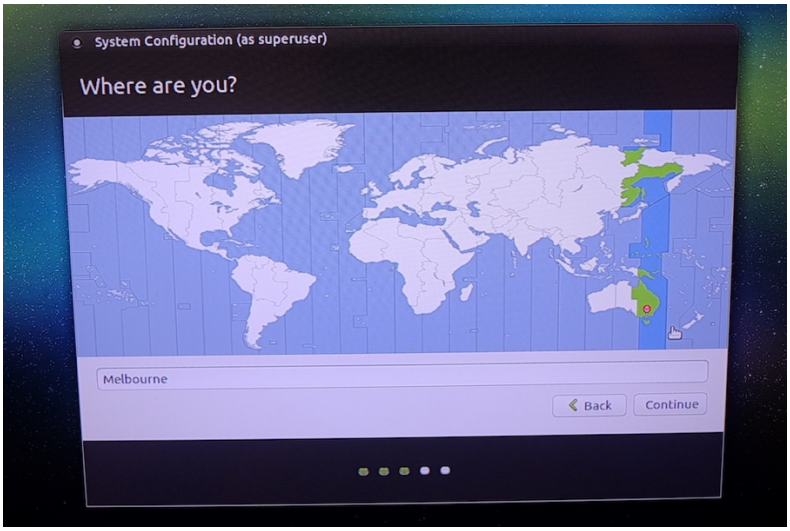
- It will then present you with a series of configuration screens.
- Select English, click continue.



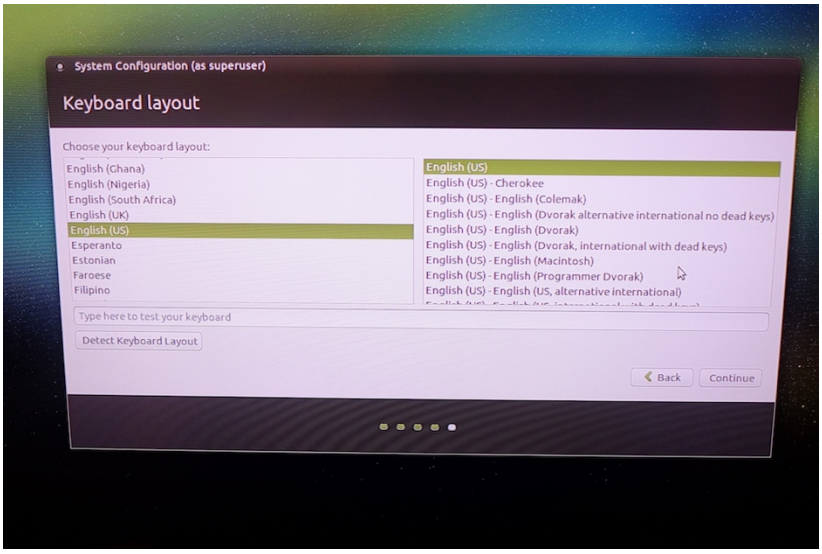
- If you are at home, enter your Wifi details and click continue. If you are at university, I suggest you leave wifi for now, and continue with the setup without network access. It will be easier to add it later.



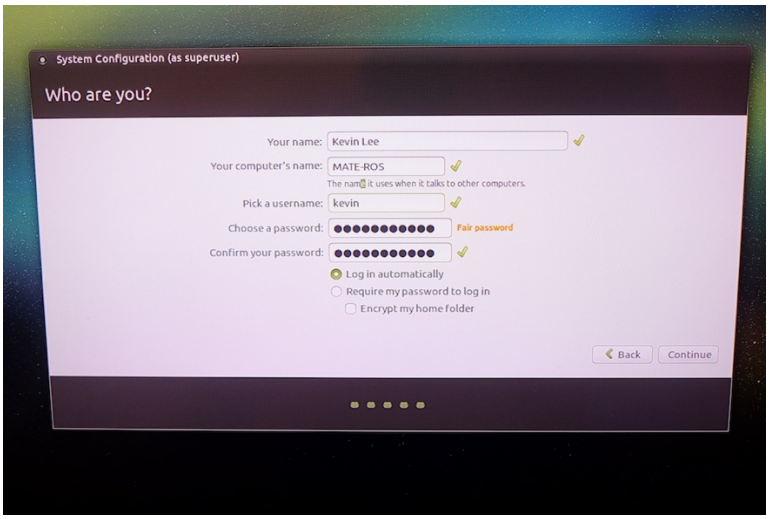
- Select your county and click continue



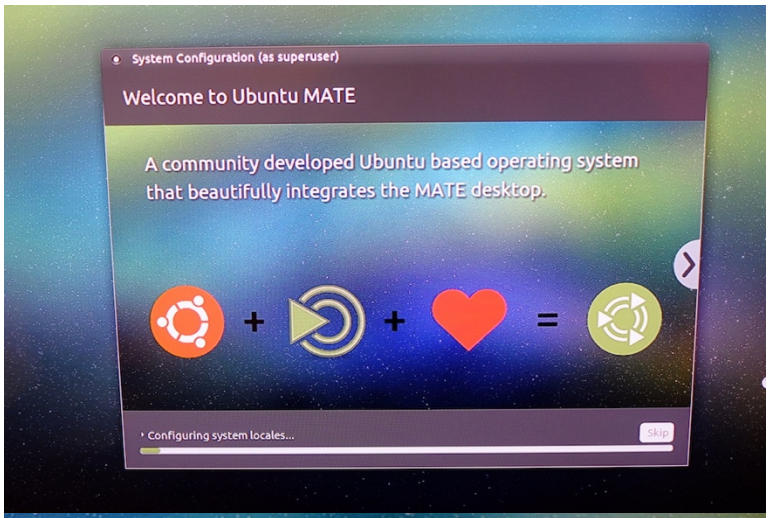
- Configure your keyboard and click continue

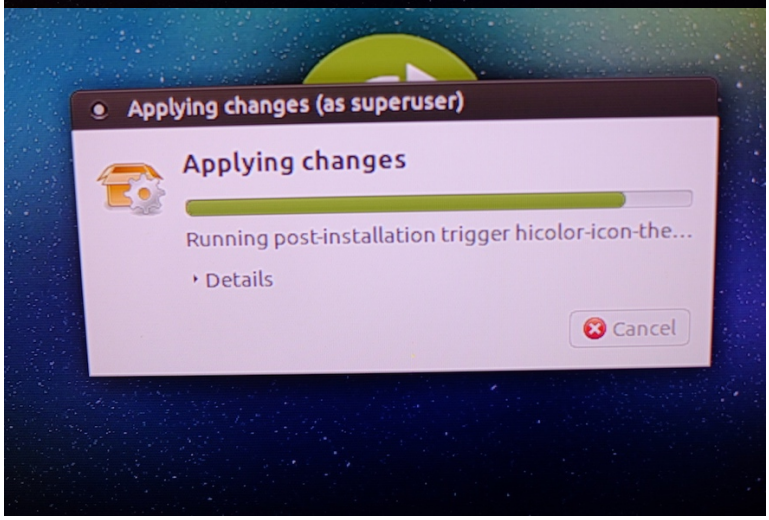


- You should enter sensible user information and click continue.

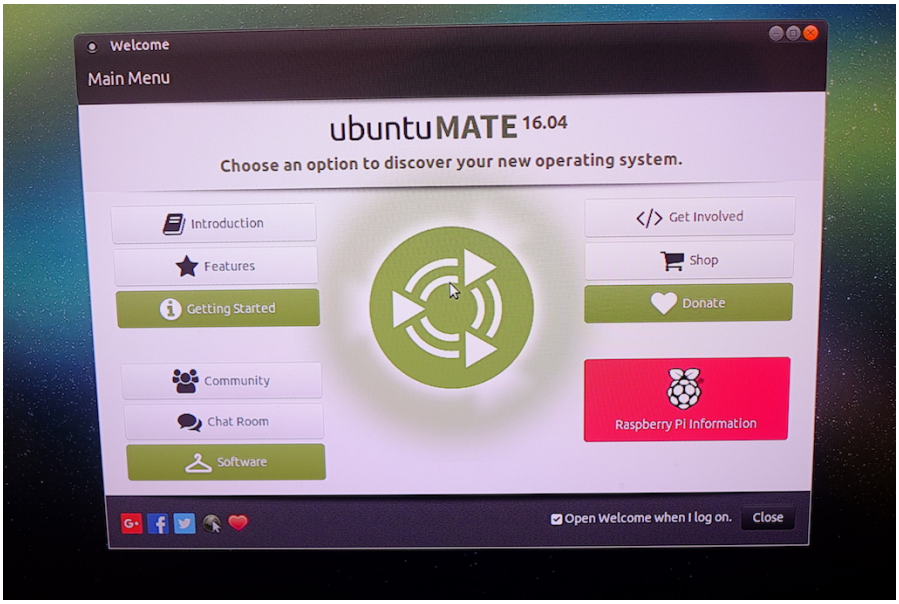


- It will now proceed to configure the system.

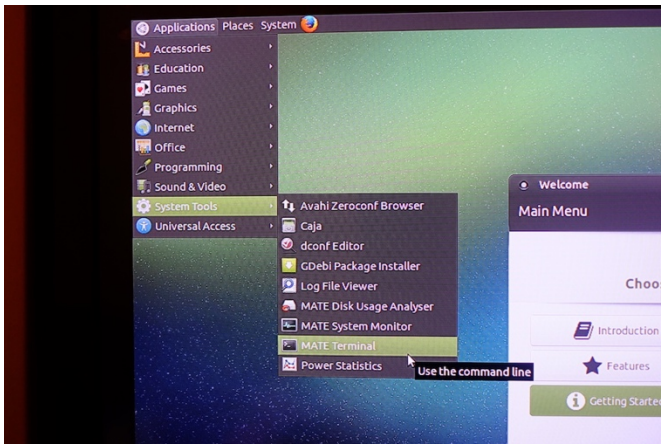




- When it is completed it will reboot the desktop and present you with a welcome screen. You can use this to learn about Ubuntu before continuing with the next step.



- At this stage, it is a good idea to reboot to check everything is working ok.
- We will be using the Mate Terminal a lot. Find it in the Applications menu in the top right, under System Tools.



- If you are at university, now is the time to get wifi working. First connect to "Deakin Setup", open a browser and try to go to any page, then follow the instructions to connect to Eduroam. Your tutor will help you.
- You should now update your system fully. As follows:

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
```

2. Install ROS on your Linux operating System

We will be using the stable version of ROS – Kinetic, which is the stable version at the time of writing.

2.1. Paste the following into the terminal. Which will add the ROS repositories to Ubuntu:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

- Setup the ROS keys:

```
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
```

- Update Ubuntu repositories to take into account the new ROS ones.

```
sudo apt update
```

- Install the full ROS desktop, which includes ROS, rqt, rviz, robot-generic libraries, 2D/3D simulators and 2D/3D perception. See http://wiki.ros.org/kinetic/Installation/Ubuntu for other installation types.

```
sudo apt install ros-kinetic-desktop-full
```

- This will install around 500MB-1GB of packages which will eventually take up around 2GB of space. ROS is a very full featured platform. Depending on your configuration, this will take around 10 minutes. Use your time by browsing the documentation in the "Additional Tasks" section below.

2.2. When its complete. Initialize rosdep which enables dependency management for your programs, as follows

```
sudo rosdep init
rosdep update
```

- To be able to use ROS easily, we need to add ROS environmental variables to our shell (the thing we type into). Execute the following. This will ensure ROS is setup for every terminal we open.

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

- To create your own ROS workspaces, you'll need a few more programs installed. So, execute the following:

```
sudo apt install python-rosinstall python-rosinstall-generator python-wstool
build-essential
```

- This shouldn't take long to install.
- Once this is complete, you have a full install of ROS, ready to go.

3. ROS Workspaces

3.1. When you develop projects using ROS, you use a workspace to organize and build your code. You have two choices, rosbuild (http://wiki.ros.org/rosbuild) and catkin (http://wiki.ros.org/catkin). Both are mature and sophisticated tools, but catkin is the newer and currently maintained one, so we will be using that.

3.2. To create a catkin workspace:

- open a terminal as before.
- cd is 'change directory' and tilde (~) represents you home folder. So, the following will take you back to your home folder.

```
cd ~
```

- The following will create a catkin workspace.

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/
$ catkin_make
```

- The catkin_make command is a tool that create various files and folders that you will need to create ROS projects.
- Like we activated the ROS platform with 'source' before. We need to activate our locale ROS workspace here.

```
$ source devel/setup.bash
```

- You now have ROS installed on your machine and have setup a catkin workspace to develop ROS projects.

4. ROS packages

4.1. To be able to take advantage of the ROS infrastructure, plugins and visualization tools, etc. – software projects need to be created as a catkin ROS package.

4.2. A catkin package contains the following:

- A catkin compliant package.xml file which describes the package. See here for detailed information: http://wiki.ros.org/catkin/package.xml
- A CMakeLists.txt file which is the input file to the build system. See here for detailed information: http://wiki.ros.org/catkin/CMakeLists.txt
- Therefore, a simple ROS catkin package is:

```
my_package/
        CMakeLists.txt
        package.xml
```

4.3. You can just create a catkin package anywhere, but the ROS recommendation is to create them within the workspace that you created before. A normal workspace will look like the following, with 2 packages (projects) visible.

```
catkin_ws/        -- WORKSPACE
        src/                    -- SOURCE SPACE
        CMakeLists.txt       -- 'Toplevel' CMake file, provided by catkin
        package_1/
                CMakeLists.txt     -- CMakeLists.txt file for package_1
                package.xml        -- Package manifest for package_1
        ...
        package_n/
                CMakeLists.txt     -- CMakeLists.txt file for package_n
                package.xml        -- Package manifest for package_n
```

4.4. To create a catkin package, perform the following.

- first navigate to your workspace in a terminal:

```
$ cd ~/catkin_ws/src
```

- Use the automation script catkin_create_pkg to create a new package called sit310_lab1 which depends on the ROS packages std_msgs, roscpp and rospy.

```
$ catkin_create_pkg sit310_lab1 std_msgs rospy roscpp
```

- This will have created a new folder sit310_lab1, with a new package.xml and CMakeLists.txt. Have a look at these files.

4.5. To build your whole workspace, including all packages perform the following.

```
$ cd ~/catkin_ws
$ catkin_make
```

4.6. You have built all your packages, but you also need to make sure your ROS environment is aware of them. Executing the following will ensure that.

```
$ . ~/catkin_ws/devel/setup.bash
```

4.7. More details about packages can be found here: http://wiki.ros.org/ROS/Tutorials/CreatingPackage

5. You will now have ROS setup up both your Raspberry Pi and another configuration on the Cloud, VM or local machine. It's a good time to test the installation works by trying a few commands.

5.1. Installing ROS packages

5.1.1. If you are using a Debian/Ubuntu based Linux distribution then you use the following to find out what ROS packages are available.

```
$ sudo apt-cache search ros-<distro>*
(replace distro with your ROS distribution, e.g. kinetic)
```

5.1.2. You can see that there are a lot of packages available. Install the main ROS tutorial package as follows. You can follow this to install any ROS package.

```
$ sudo apt-get install ros-<distro>-ros-tutorials
```

5.2. Filesystem Tools

5.2.1. rospack

- Once you have installed a package in Linux, you can query it using ROS tools.
- You can find a package using the following.

```
$ rospack find [package_name]
```

- On your system, try the following.

```
$ rospack find roscpp
/opt/ros/kinetic/share/roscpp
```

5.2.2. roscd

- roscd allows you to change to a ROS package directory, as follows.

```
$ roscd [locationname[/subdir]]
```

- On your system, try the following.

```
$ roscd roscpp    (changes to the rosccp package)
$ pwd  (this prints the current directory in Linux)
/opt/ros/kinetic/share/roscpp
```

5.2.3. rosls

- a useful timesaving tool is rosls which allows you to list the contents of a package as follows

```
$ rosls roscpp_tutorials
Cmake msg package.xml rosbuild srv
```

5.2.4. Tab completion

- Like Linux. ROS has lots of timesaving tricks. One of the most useful, on Linux and ROS is tab completion.
- On Linux, when you are typing a command, you can press tab and it will complete the text for you. E.g. apt-g. <<tab>> will result in 'apt-get'
- Try the following.

```
$ roscd roscpp_tut<<< now push the TAB key >>>
roscd roscpp_tutorials/ (this will happen)
```

6. Additional resources

6.1. Learn about linux

- https://www.computerworld.com/article/2598082/linux/linux-linux-command-line-cheat-sheet.html
-

6.2. Learn about ROS.

- http://wiki.ros.org
- A Gentle Introduction to ROS: https://www.cse.sc.edu/~jokane/agitr/
- http://wiki.ros.org/kinetic/Installation/Ubuntu
- http://wiki.ros.org/ROS/Tutorials

Well done for completing this tutorial. Spend some time reading the online resources about Linux and ROS above. Then move on to Lab Task 2.