Preparing your Development Environment

1. Operating System

Robotics Development relies heavily on **Linux**. We recommend using Ubuntu 18.04 since it is the most widely used and supported variant of Linux. If you already have Ubuntu 16.04, this is also OK, although for versions older than this we recommend upgrading to 18.04.

For those who currently have **Windows** as the only OS on their machine, the best way to start using Ubuntu would be to **dual boot**. Here is a guide on how to do this. For **MacOS** users, dual booting is an option but we recommend using up a virtual machine.

If you are unable to dual boot for any reason, you can try setting up a **virtual machine**. The first step in this is to install a virtualisation software. For Windows you can use either VirtualBox (free) or Vmware Workstation and for MacOS either VirtualBox (free), Vmware Fusion or Parallels. After getting one of the above, follow the instructions given here (skip ahead to the *Download Image* section). After completing the given procedure you will be equipped with all the basic tools required for Robotics including ROS, catkin and git.

In the unfortunate case that the **above options do not work**, for Windows users there is still a way - WSL. Do be warned however, this path is fraught with frustration and much debugging. Only continue if you have exhausted other options. For a guide on setting up WSL for ROS, look here.

For those whom none of the above are possible, consider using the online browser based ROS Development Studio. Keep in mind that it has a limited access time per week and performance may be questionable.

2. Robotics Operating System (ROS)

Note that this part is unnecessary if you followed the given instructions to set up a VM. For everyone else, this part is **essential**. Different versions of Ubuntu need different variants of ROS. Instructions give below -

- ➔ Ubuntu 18.04: ROS Melodic
- ➔ Ubuntu 16.04: ROS Kinetic

3. Useful tools to make your life easier

- Git: Fundamental tool in open source software development. Used for version control and sharing of code.

```
sudo apt install git
```

- Terminator: Terminal Emulator useful for having multiple terminals in a window.

```
sudo apt install terminator
```

- Code Editors: A good editor can go a long way in boosting productivity. We recommend VSCode which has plugins for python and ROS. A comprehensive guide for how to integrate ROS into your favourite IDE can be found here.

4. ROS Packages

➔ You can install already developed ROS packages using the apt (packet manager for Ubuntu). Replace `<package_name>` name of the ROS package

```
sudo apt install ros-$ROS_DISTRO-<package_name>
```

➔ For example in this course you will need to install Turtlebot3 and its related packages with the following command -

```
sudo apt install ros-$ROS_DISTRO-turtlebot3-*
```

➔ Set the default Turtlebot model as *"burger"* by adding the following line at the end of your `bashrc` file -

```
export TURTLEBOT3_MODEL=burger
```

➔ Test your Turtlebot and Gazebo setup by launching a sample launch file using the following command

```
roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
```

If you see the Turtlebot in the gazebo environment, then you may assume that the installation is successful!

5. Tips for getting things to work + some helpful facts

➔ Make a habit of running `sudo apt update` before installing packages in linux.

➔ For the uninitiated, your `bashrc` file is the configuration file for your bash terminal (the thing you type commands into). It's usually located in your home directory at ~/`.bashrc` For more info, check out this.

➔ Don't forget to source the workspace you want to use. For convenience you can source the workspace on startup by editing your `bashrc` file to include the following line. Replace `<workspace_path>` with the path of your workspace

```
source <workspace_path>/devel/setup.bash
```

➔ You cannot source two workspaces at the same time.

➔ Anaconda and ROS cannot be used in the same environment because they have a conflicting python path. As given here, to deal with this, edit your `bashrc` file by commenting the anaconda python path like this -

```
// export PATH="/home//anaconda3/bin:$PATH"
```

➔ Use python `pip` to install python dependencies. Anaconda should be avoided.