Introduction to ROS (Melodic) and Control of Pioneer 3-DX Mobile Robot

(This document is a compilation of material from O'Kane, J.M., 2014. A gentle introduction to ROS and various web resources)

Robotics Laboratory

Indian Institute of Technology Jodhpur

Chapter 1: Installing Ubuntu, ROS and Gazebo

This document uses Ubuntu 18.04 LTS (Bionic), ROS Melodic and Gazebo9 to simulate and control mobile robots. Installation procedures for the above are given in this chapter. Following ways can be used to quick start with the user manual, however, *Step 1* is recommended for the first-time user.

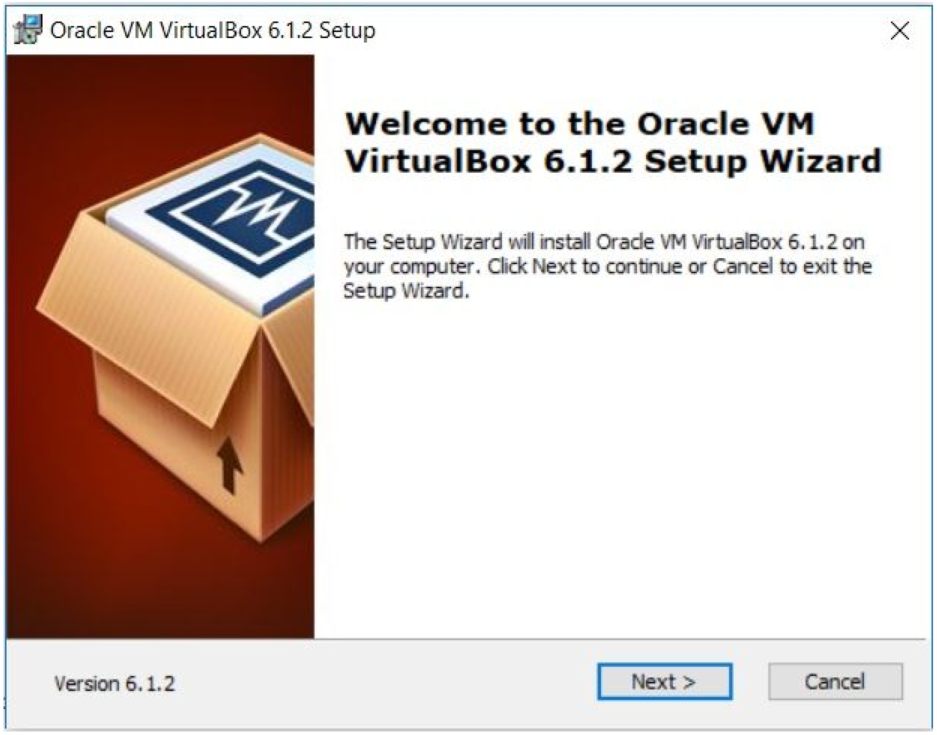
1. If you would like to practice this using Windows 10 machine without installing Ubuntu 18.04 LTS (Bionic) OS, You can install Oracle VM virtual box and link the image of Ubuntu 18.04 LTS (Bionic) OS with pre installed ROS Melodic and Gazebo9 created for this course to quick start with this manual. Refer 1.1 for installation of VM virtual box and linking of Ubuntu image.
2. For smooth performance, it is advisable that you dual boot your window machine with fresh installation of Ubuntu 18.04 LTS (Bionic) OS. Refer Annexure 3 for dual booting of Ubuntu 18.04 LTS (Bionic) OS.
3. ROS and Gazebo can be installed using Annexure 2. If you have Ubuntu 18.04 LTS (Bionic) OS pre-installed in your machine then refer Annexure 2 for ROS and Gazebo Installation.

1.1 Installation of Oracle VM Virtual box and Ubuntu Image on Windows 10

Here you will be provided a procedure to install this software on Windows 10 and running Ubuntu Image using the same.

1.1.1 Installation procedure for Oracle VM Virtual Box

1. Download the Oracle VM Virtual Box from **click here** or from the link. (https://www.virtualbox.org/wiki/Downloads)
2. The file downloaded will have the file name format like VirtualBox-VersionNumber-BuildNumber-Win.exe. Something like this: *VirtualBox-6.1.4-136177-Win.exe*. Double click on the installer to launch the setup Wizard. Click on Next to continue.

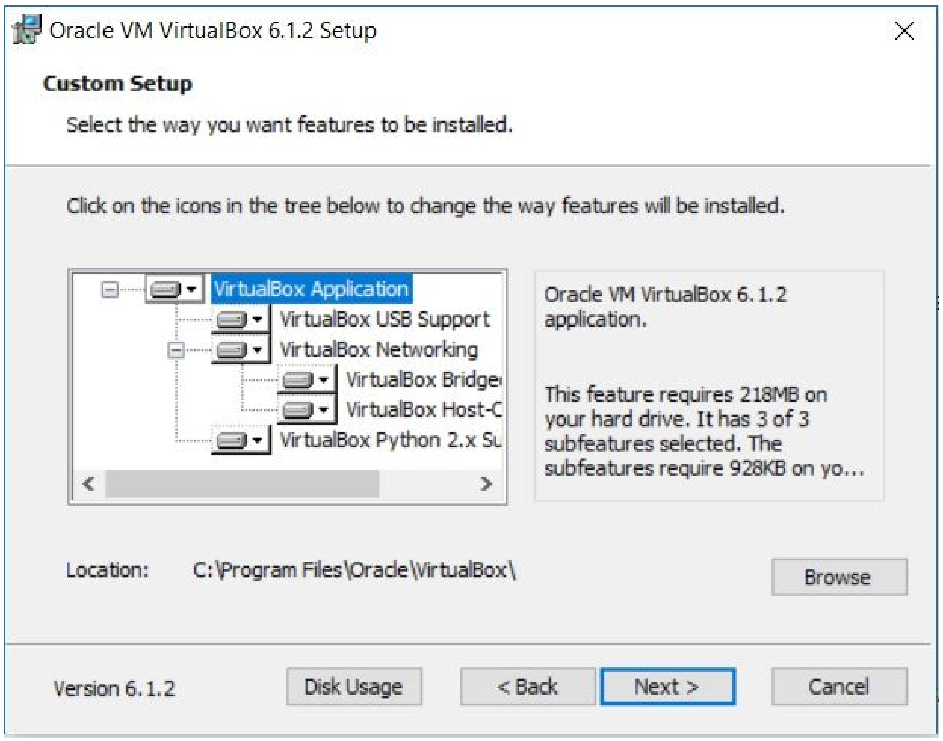

Oracle VM VirtualBox 6.1.2 Setup

Welcome to the Oracle VM VirtualBox 6.1.2 Setup Wizard

The Setup Wizard will install Oracle VM VirtualBox 6.1.2 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Version 6.1.2

3. You will see a custom setup dialog box. There is not much to choose from. You can accept the default and click next. If you wish to change the installation directory, you can change it by clicking on the browse button and selecting the new directory and clicking OK. Normally you may leave it as the default as the whole installation process does not take much space on your hard drive.


Oracle VM VirtualBox 6.1.2 Setup

Custom Setup

Select the way you want features to be installed.

Click on the icons in the tree below to change the way features will be installed.

- VirtualBox Application
  - VirtualBox USB Support
  - VirtualBox Networking
    - VirtualBox Bridge
    - VirtualBox Host-C
  - VirtualBox Python 2.x Su

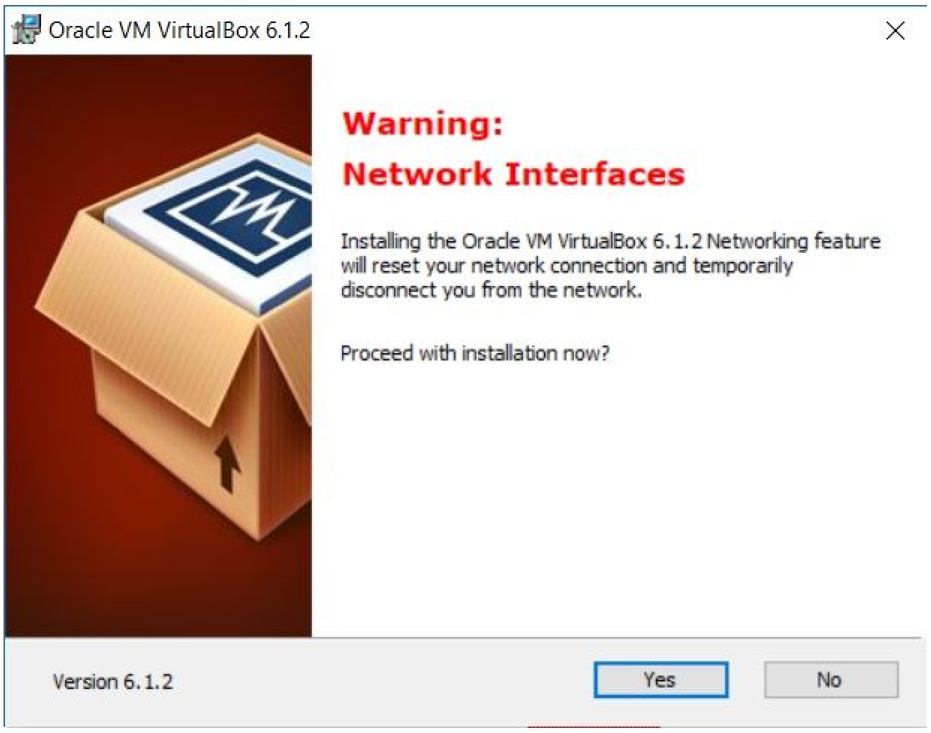
Oracle VM VirtualBox 6.1.2 application.

This feature requires 218MB on your hard drive. It has 3 of 3 subfeatures selected. The subfeatures require 928KB on yo...

Location: C:\Program Files\Oracle\VirtualBox\

Version 6.1.2

4. This dialog box warns you about setting up a Network Interface. what this means that VirtualBox will install network interfaces that will interact with the installed virtual machines and the host operating system which in our case is windows. This will temporarily disconnect you from the internet but that is OK, nothing to worry.


Oracle VM VirtualBox 6.1.2

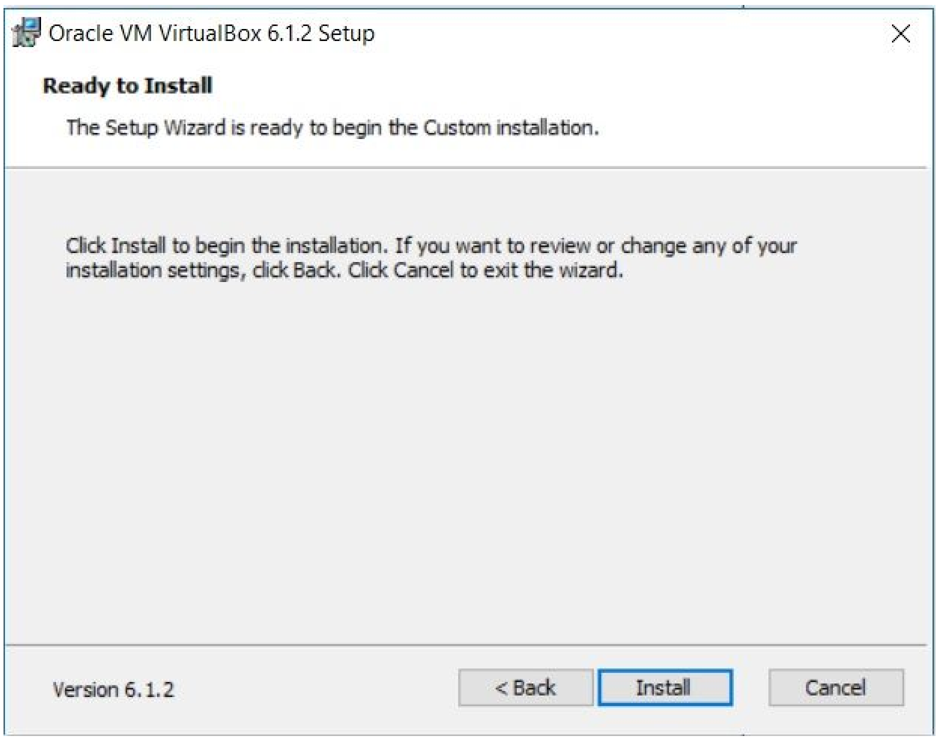
Warning:
Network Interfaces

Installing the Oracle VM VirtualBox 6.1.2 Networking feature will reset your network connection and temporarily disconnect you from the network.

Proceed with installation now?

Version 6.1.2

5. You will see ready to install a dialog box. Click Install to continue.


Oracle VM VirtualBox 6.1.2 Setup

Ready to Install

The Setup Wizard is ready to begin the Custom installation.

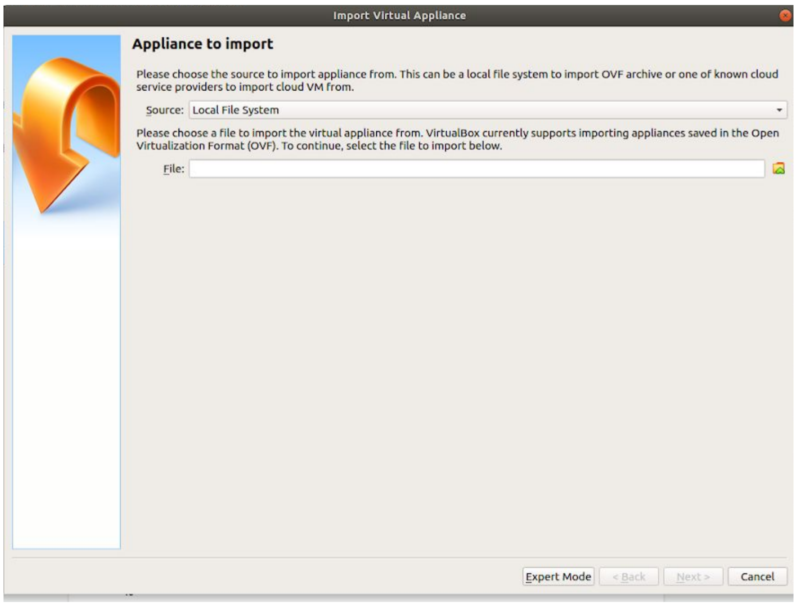
Click Install to begin the installation. If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Version 6.1.2

6. After installation, launch the application.

1.1.2 Import the Ubuntu Image in Oracle VM virtualbox

1. Download Ubuntu 18.04 LTS (Bionic) OS Image with pre installed ROS Melodic and Gazebo9 from here: https://drive.google.com/file/d/1SOYZSb0Od4PvqqrhlRA48Mv3pNMqVw2E/view?usp=sharing
2. After launching the application, go to "File" and click on "import appliance".



Appliance to import

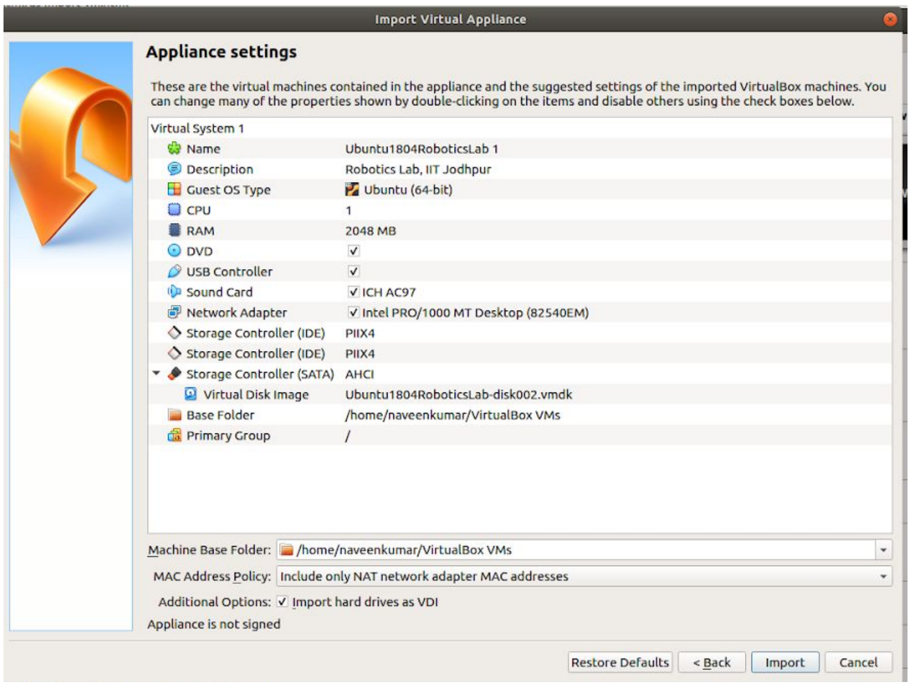
Please choose the source to import appliance from. This can be a local file system to import OVF archive or one of known cloud service providers to import cloud VM from.

Source: Local File System

Please choose a file to import the virtual appliance from. VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

File:

3. Choose the path of the file with its name by clicking on the file manager icon which has an arrow sign on it.
4. Click on next and then on the next window click on import.



Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

Virtual System 1	
Name	Ubuntu1804RoboticsLab 1
Description	Robotics Lab, IIT Jodhpur
Guest OS Type	Ubuntu (64-bit)
CPU	1
RAM	2048 MB
DVD	✓
USB Controller	✓
Sound Card	✓ ICH AC97
Network Adapter	✓ Intel PRO/1000 MT Desktop (82540EM)
Storage Controller (IDE)	PIIX4
Storage Controller (IDE)	PIIX4
Storage Controller (SATA)	AHCI
Virtual Disk Image	Ubuntu1804RoboticsLab-disk002.vmdk
Base Folder	/home/naveenkumar/VirtualBox VMs
Primary Group	/

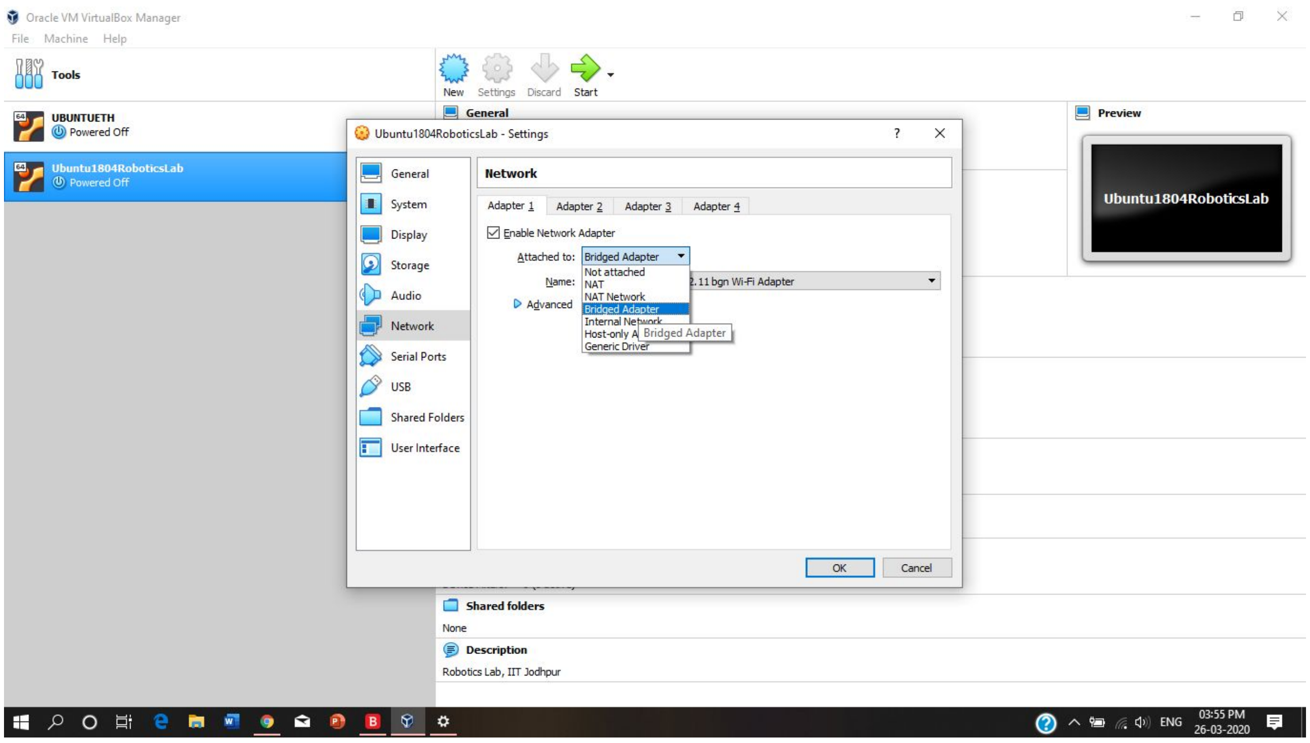
Machine Base Folder: /home/naveenkumar/VirtualBox VMs

MAC Address Policy: Include only NAT network adapter MAC addresses

Additional Options: ✓ Import hard drives as VDI

Appliance is not signed

5. Before you launch the virtual machine, you have to change the network adapter settings to make it compatible with your network for communication with your Android Mobile device.
6. Click on settings and go to Network options.



☑ Enable Network Adapter

Attached to: Bridged Adapter

Name: 802.11 bgn Wi-Fi Adapter

Shared folders: None

Description: Robotics Lab, IIT Jodhpur

7. Change the settings from "NAT" to "Bridged Adapter"
8. Click on OK and launch the machine by clicking on the start button.

Troubleshooting :

1. The USB device doesn't mount automatically in the case of VirtualBox, install VirtualBox extensions package for using the USB port.
2. Setup BIOS settings properly because some PC's don't allow virtual machines.
   For that you have to enable virtualization in the BIOS menu for that you have to check the documentation of your PC and Laptop.
3. You might have to disable secure boot if you are loading the virtual machine on a linux platform. Disabling secure boot is different according to your PC or Laptop. So, for that, you have to look at the documentation of your PC.

After installing the virtual machine, you have to install some important elements into your virtual machine.

Step 1: Open terminal in Ubuntu by pressing Ctrl + Alt + T

Step 2: Type below command to install .

```
sudo apt install net-tools
```

```
sudo apt install openssh-server
```

Chapter 2: Getting started with ROS

What is ROS

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

ROS is an open-source, meta-operating system for robots. It provides the services one would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

Why ROS

Distributed computation

Many modern robot systems rely on software that spans many different processes and runs across several different computers. For example:

1. Some robots carry multiple computers, each of which controls a subset of the robot's sensors and actuators.
2. Even within a single computer. it's often a good idea to divide the robot's software into small, stand-alone parts that cooperate to achieve the overall goal. This approach is sometimes called "complexity via composition."
3. When multiple robots attempt to cooperate on a shared task, they often need to communicate with one another to coordinate their efforts.
4. Human users often send commands to a robot from a laptop, a desktop computer, or mobile device. We can think of this human interface as an extension of the robot's software.

Software reuse

The rapid progress of robotics research has resulted in a growing collection of good algorithms for common tasks such as navigation, motion planning, mapping, and many others. Of course, the existence of these algorithms is only truly useful if there is a way to apply them in new contexts, without the need to reimplement each algorithm for each new system. ROS can help to prevent this kind of pain in at least two important ways.

1. ROS's standard packages provide stable, debugged implementations of many important robotics algorithms.
2. ROS's message passing interface is becoming a de facto standard for robot software interoperability, which means that ROS interfaces to both the latest hardware and to implementations of cutting edge algorithms are quite often available. For example, the ROS website lists hundreds of publicly-available ROS packages. This sort of uniform interface greatly reduces the need to write "glue" code to connect existing parts.

As a result, developers that use ROS can expect—after, of course, climbing ROS's initial learning curve—to focus more time on experimenting with new ideas, and less time reinventing wheels.

Rapid testing One of the reasons that software development for robots is often more challenging than other kinds of development is that testing can be time consuming and error-prone. Physical robots may not always be available to work with, and when they are, the process is sometimes slow and finicky. Working with ROS provides two effective workarounds to this problem.

1. Well-designed ROS systems separate the low-level direct control of the hardware and high-level processing and decision making into separate programs. Because of this separation, we can temporarily replace those low-level programs (and their corresponding hardware) with a simulator, to test the behavior of the high-level part of the system.
2. ROS also provides a simple way to record and play back sensor data and other kinds of messages. This facility means that we can obtain more leverage from the time we do spend operating a physical robot. By recording the robot's sensor data, we can replay it many times to test different ways of processing that same data. In ROS, these recordings are called "bags" and a tool called rosbag is used to record and replay them.

A crucial point for both of these features is that the change is seamless. Because the real robot, the simulator, and the bag playback mechanism can all provide identical (or at least very similar) interfaces, your software does not need to be modified to operate in these distinct scenarios, and indeed need not even “know” whether it is talking to a real robot or to something else.

2.1 Configuring your account

Whether you install ROS yourself or use a computer on which ROS is already installed, there are two important configuration steps that must be done within the account of every user that wants to use ROS.

Setting up rosdep in a user account: First, you must initialize the rosdep system in your

```
rosdep update
```

This command stores some files in your home directory, in a subdirectory called *.ros* . It generally only needs to be done once.

Setting environment variables: ROS relies on a few environment variables to locate the files it needs. To set these environment variables, you’ll need to execute the setup.bash script that ROS provides, using this command:

```
source /opt/ros/melodic/setup.bash
```

You can then confirm that the environment variables are set correctly using a command like this:

```
export | grep ROS
```

If everything has worked correctly, you should see a handful of values (showing values for environment variables like ROS_DISTRO and ROS_PACKAGE_PATH) as the out- put from this command. If setup.bash has not been run, then the output of this command will usually be empty.

Note, however, that the steps listed above apply only to the current shell. It would work perfectly well to simply execute that source command each time you start a new shell in which you’d like to execute ROS commands. However, this is both annoying and remarkably easy to forget, especially when you consider that the modular design of many

ROS systems often calls for several different commands to execute concurrently, each in a separate terminal.

Thus, you'll want to configure your account to run the setup.bash script automatically, each time you start a new shell. To do this, edit the file named .bashrc in your home directory, and put the above source command at the bottom, i.e.,

```
gedit ~/.bashrc
```

and provide the path of our workspace and at the end of the .bashrc file by writing the following line and save the file:

source /opt/ros/melodic/setup.bash

This will execute the above line automatically every time shell/terminal is opened.

2.2 Turtlesim Example

Before we begin to examine the details of how ROS works, let's start with an example. This quick exercise will serve a few different purposes: It will help you confirm that ROS is installed correctly, it will introduce the turtlesim simulator that is used in many online tutorials

Starting turtlesim

In three separate terminals (*press ctr* + *shift* + *t* for opening a new terminal), execute these three commands:

```
roscore
rosrun turtlesim turtlesim_node
rosrun turtlesim turtle_teleop_key
```

2.3 Packages:

All ROS software is organized into packages. A ROS package is a coherent collection of files, generally including both executables and supporting files, that serves a specific purpose.

Listing and locating packages: You can obtain a list of all of the installed ROS packages

```
rospack list
```

To find the directory of a single package, use the rospack find command:

```
rospack find package-name
```

Of course, there may be times when you don't know (or can't remember) the complete name of the package that you're interested in. In these cases, it's quite convenient that rospack supports tab completion for package names. For example, you could type

```
rospack find turtlesim
```

Inspecting a package: To view the files in a package directory, use a command like this:

```
rosls package-name
```

e.g.

```
rosls turtlesim
```

If you'd like to "go to" a package directory, you can change the current directory to a particular package, using a command like this:

```
roscd package-name
```

e.g.

```
roscd turtlesim
```

2.4 The master

One of the basic goals of ROS is to enable roboticists to design software as a collection of small, mostly independent programs called nodes that all run at the same time. For this to work, those nodes must be able to communicate with one another. The part of ROS that facilitates this communication is called the *ROS master*. To start the master, use this command:

```
roscore
```

We've already seen this in action in the turtlesim example. You should allow the master to continue running for the entire time that you're using ROS.

2.5 Nodes

Once you've started roscore , you can run programs that use ROS. A running instance of a ROS program is called a node.

In the turtlesim example, we created two nodes. One node is an instance of an executable called *turtlesim_node*. This node is responsible for creating the turtlesim window and simulating the motion of the turtle. The second node is an instance of an executable called *turtle_teleop_key*. The abbreviation teleop is a shortened form of the word teleoperation, which refers to situations in which a human controls a robot remotely by giving direct movement commands. This node waits for an arrow key to be pressed, converts that key press to a movement command, and sends that command to the *turtlesim_node* node.

Starting nodes: The basic command to create a node (also known as running a ROS program) is *rosrun*

```
rosrun package-name executable-name
```

For example in *rosrun turtlesim turtlesim_node,* *turtlesim* is package name and *turtlesim_noden* is executable-name

Listing nodes: ROS provides a few ways to get information about the nodes that are running at any particular time. To get a list of running nodes, try this command:

```
rosnode list
```

If you do this after executing the commands from Section 2.1, you'll see a list of three nodes:

/rosout

/teleop_turtle

/turtlesim

The */rosout* node is a special node that is started automatically by roscore .

Inspecting a node You can get some information about a particular node using this com-

```
rosnode info node-name
```

Killing a node: To kill a node you can use this command:

```
rosnode kill node-name
```

2.6 Topics and messages

In our turtlesim example, it's clear that the teleoperation node and the simulator node must be talking to each other somehow. Otherwise, how would the turtle, which lives in the latter node, know when to move in response to your key presses, which are collected by the former node?

The primary mechanism that ROS nodes use to communicate is to send *messages*. Messages in ROS are organized into named *topics*. The idea is that a node that wants to share information will *publish* messages on the appropriate topic or topics; a node that wants to receive information will *subscribe* to the topic or topics that it's interested in. The ROS master takes care of ensuring that publishers and subscribers can find each other; the messages themselves are sent directly from publisher to subscriber.

2.6.1 Viewing the graph

This idea is probably easiest to see graphically, and the easiest way to visualize the publish-subscribe relationships between ROS nodes is to use this command:

```
rqt_graph
```

In this name, the r is for ROS, and the qt refers to the Qt GUI toolkit used to implement the program. You should see a GUI, most of which is devoted to showing the nodes in the current system.

Running this after turtle sim example, you will see something like Figure 1. In this graph, the *ovals represent nodes*, and the *directed edges represent publisher-subscriber relationships*. The graph tells us that the node named */teleop_turtle* publishes messages on a topic called */turtle1/cmd_vel* , and that the node named */turtlesim* subscribes to those messages. (In this context, the name " *cmd_vel* " is short for "*command velocity*.")

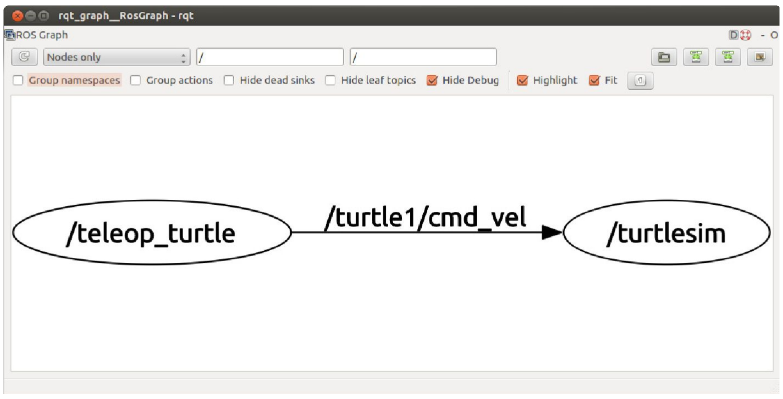


Figure 2.1. The rqt_graph interface, showing the graph for the turtlesim example. Debug nodes, including rosout , are omitted by default.

You might notice that the *rosout* node that we saw earlier is missing from this view. By default, rqt_graph hides nodes that it thinks exist only for debugging. You can disable this feature by unchecking the "*Hide debug*" box from Figure 1. The resulting graph will be similar to Figure 2.

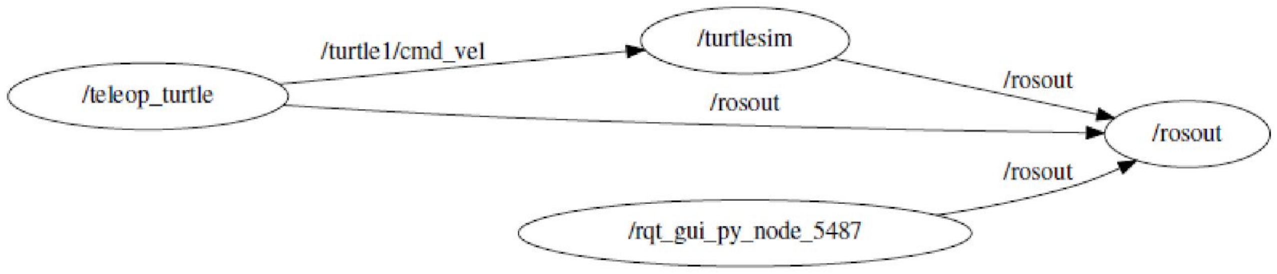


Figure 2.2: The complete turtlesim graph, including nodes that rqt_graph classifies as debug nodes.

Notice that *rqt_graph* itself appears as a node. All of these nodes publish messages on a topic called */rosout* , to which the node named */rosout* subscribes. This topic is one mechanism through which nodes can generate textual log messages.

The *rqt_graph* tool has several other options for tweaking the way that it shows the graph. For example, one can change the dropdown from "Nodes only" to "Nodes/Topics (all)", and to disable all of the checkboxes except "Hide Debug." This setup, whose results are shown in Figure 3, has the advantage that all of the topics are shown in rectangles, separate from the nodes.One can see, for example, that the turtlesim node, in addition to subscribing to velocity commands, also publishes both its *current pose* and *data from a simulated color sensor*. When you're exploring a new ROS system, rqt_graph , especially with these options, can be a useful way to discover what topics are available for your programs to use to communicate with the existing nodes.

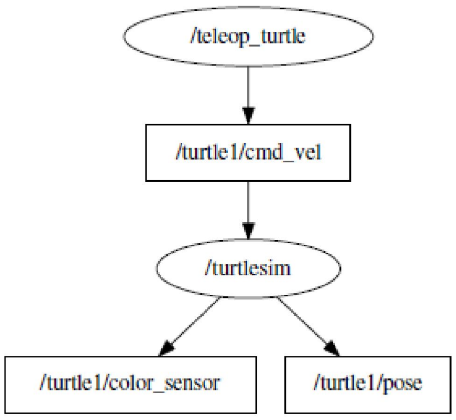


Figure 2.3. The turtlesim graph, showing all topics, including those with no publishers or no subscribers, as distinct objects.

The important points here are:

- The simulator doesn't care (or even know) which program publishes those cmd_vel messages. Any program that publishes on that topic can control the turtle.
- The teleoperation program doesn't care (or even know) which program subscribes to the cmd_vel messages it publishes. Any program that subscribes to that topic is free to respond to those commands.

2.6.2 Messages and message types

So far we've talked about the idea that nodes can send messages to each other, but we've been quite vague about what information is actually contained in those messages. Let's take a closer look at the topics and messages themselves.

Listing topics

To get a list of active topics, use this command:

```
rostopic list
```

In our example, this shows a list of five topics:

```
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

The topic list should, of course, be the same as the set of topics viewable in rqt_graph, but might be more convenient to see in text form.

Echoing messages

You can see the actual messages that are being published on a single topic using the rostopic command:

```
rostopic echo topic-name
```

This command will dump any messages published on the given topic to the terminal. For example, running the following command in new terminal after performing steps in 2.1

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

shows some example output at a time when /teleop_turtle was receiving keystrokes.

Measuring publication rates

There are also two commands for measuring the speed at which messages are published and the bandwidth consumed by those messages:

```
rostopic hz topic-name
rostopic bw topic-name
```

These commands subscribe to the given topic and output statistics in units of messages per second and bytes per second, respectively.

Inspecting a topic

You can learn more about a topic using the rostopic info command:

```
rostopic info topic-name
```

For example, from this command:

```
rostopic info /turtle1/cmd_vel
```

you should see output similar to this:

```
Type: geometry_msgs/Twist
Publishers:
 * /teleop_turtle (http://robotics:43701/)
Subscribers:
* /turtlesim (http://robotics:36053/)
```

The most important part of this output is the very first line, which shows the message type of that topic. In the case of /turtle1/cmd_vel , the message type is geometry_msgs/Twist .The word "type" in this context is referring to the concept of a data type.

Inspecting a message type

To see details about a message type, use a command like

```
rosmsg show message-type-name
```

Let's try using it on the message type for /turtle1/cmd_vel that we found above:

```
rosmsg show geometry_msgs/Twist
```

The output is:

```
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

In this case, both linear and angular are composite fields whose data type is geometry_msgs/Vector3 . The indentation shows that fields named x , y , and z are members within those two top-level fields. That is, a message with type geometry_msgs/Twist contains exactly six numbers, organized into two vectors called linear and angular . Each of these numbers has the built-in type float64 , which means, naturally, that each is a 64-bit floating point number.

Publishing messages from the command line

Most of the time, the work of publishing messages is done by specialized programs. However, you may find it useful at times to publish messages by hand. To do this, use rostopic :

```
rostopic pub -r rate-in-hz topic-name message-type message-content
```

This command repeatedly publishes the given message on the given topic at the given rate. The final message content parameter should provide values for all of the fields in the message type, in order. Here's an example:

```
rostopic pub -r 1 /turtle1/cmd_vel geometry_msgs/Twist '[2, 0, 0]' '[0, 0, 0.5]'
```

When this step is performed after 2.1, the robot can be seen moving with the above rates. The values are assigned to message fields in the same order that they are shown by rosmsg show .In the case, the first three numbers denote the desired linear velocity and the final three numbers denote the desired angular velocity . We use single quotes ('. . . ') and square brackets ([. . .]) to group the individual subfields into the two top-level composite fields. As you might guess, the messages generated by this example is a circle of radius 4 units.

Understanding message type names

Like everything else in ROS, every message type belongs to a specific package. Message type names always contain a slash, and the part before the slash is the name of the containing package:

package-name/type-name

For example, the *turtlesim/Color message* type breaks down this way:

$$\underbrace{\text{turtlesim}}_{\text{package name}} + \underbrace{\text{Color}}_{\text{type name}} \Rightarrow \underbrace{\text{turtlesim/Color}}_{\text{message data type}}$$

2.7 Second example

So far in this chapter, we've seen how to start the ROS master, how to start ROS nodes, and how to investigate the topics those nodes use to communicate with one another. This section wraps up our introduction with another example intended to illustrate a bit more fully the way topics and messages work.

First, stop any nodes that might be currently running. You can use "Ctr +C" or "rosnode kill" Start roscore if it's not already active. Then, in four separate terminals, run these four commands:

```
rosrun turtlesim turtlesim_node __name:=A
rosrun turtlesim turtlesim_node __name:=B
rosrun turtlesim turtle_teleop_key __name:=C
rosrun turtlesim turtle_teleop_key __name:=D
```

This should start two instances of the turtlesim simulator—These should appear in two separate windows—and two instances of the turtlesim teleoperation node.The only element in the example that might be unfamiliar is the _ _name parameter to rosrun . These parameters override the default name that each node tries to assign to itself. **They're needed because the ROS master does not allow multiple nodes with the same name.**

Before we discuss this four-node example, you may wish to take a moment to think about how the system will behave. What would the graph, as displayed by rqt_graph , look like? Which turtles would move in response to which teleoperation nodes? Hopefully, you predicted that the graph would look like Figure 4, and that both turtles would make the same movements in response to key presses sent to either teleoperation node. Let's see why.

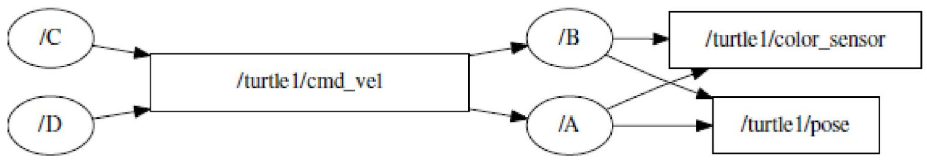


Figure 2.4: A slightly more complex ROS graph, with two turtlesim nodes named A and B and two teleoperation nodes named C and D .

2.7.1 Communication via topics is many-to-many.

You might have expected each teleoperation node to connect to one simulator, creating two independently controllable simulations. Note, however, that these two kinds of nodes publish and subscribe, respectively, on the /turtle1/cmd_vel topic. Messages published on this topic, regardless of which node publishes them, are delivered to every subscriber of that topic.

In this example, every message published by teleoperation node C is delivered to both simulation nodes, namely A and B . Likewise, messages published by D are delivered to both A and B . When these messages arrive, the turtles move accordingly, regardless of which node published them. The main idea here is that topics and messages are used for many-to-many communication. Many publishers and many subscribers can share a single topic.

Now let us try the following command in a new terminal

```
rostopic pub -r 1 /turtle1/cmd_vel geometry_msgs/Twist '[2,0, 0]' '[0, 0, 0.5]'
```

When this step is performed both robots can be seen moving with the above rates.

2.7.2 Nodes are loosely coupled.

No doubt you have noticed that we did not need to reprogram the turtlesim simulator to accept movement commands from multiple sources, nor did the teleoperation node need to be designed to drive multiple instances of the simulator at once. In fact, it would be an easy exercise to extend this example to arbitrarily many nodes of either type.

At the other extreme, consider what would happen if the turtlesim simulator were started in isolation, without any other nodes. In that situation, the simulator would wait for messages on /turtle1/cmd_vel , happily oblivious to the fact that there are no publishers for that topic.

The punchline is that our turtlesim nodes specifically—and most well-designed ROS nodes generally—are loosely coupled. None of the nodes explicitly know about any of the others; their only interactions are indirect, at the level of topics and messages. This independence of nodes, along with the decomposition it facilitates of larger tasks into smaller reusable pieces, is one of the key design features of ROS.

2.8 Checking for problems

One final (for now) command line tool, which can be helpful when ROS is not behaving the way you expect, is roswtf , which can be run with no arguments:

```
roswtf
```

This command performs a broad variety of sanity checks, including examinations of your environment variables, installed files, and running nodes. For example, roswtf checks whether the rosdep portions of the install process have been completed, whether any nodes appear to have hung or died unexpectedly, and whether the active nodes are correctly connected to each other. A complete list of checks performed by roswtf seems to exist, unfortunately, only in the Python source code for that tool

Chapter 3: Setting up ROS Workspace for running external packages

First remove any previously created catkin_ws by the following command

```
rm -r  ~/catkin_ ws
```

Creating a catkin workspace: Creating a workspace Packages that you create should live together in a directory called a workspace. Use the normal mkdir command to create a directory named catkin_ws

```
mkdir -p ~/catkin_ws/src
```

Now change the existing directory to catkin_ws using the following command:

```
cd ~/catkin_ws/
```

We'll refer to this new directory as your workspace directory.

Building the workspace: Next you can build your work-space including compiling all of the executables in all of its packages using this command

```
catkin_make
```

Because it's designed to build all of the packages in your workspace, this command must be run from your workspace directory. It will perform several configuration steps (especially the first time you run it) and create subdirectories called devel and build within your work- space. These two new directories contain build-related files like automatically-generated makefiles, object code, and the executables themselves. If you like, the devel and build subdirectories can safely be deleted when you've finished working on your package. If there are compile errors, you'll see them here. After correcting them, you can catkin_make again to complete the build process. Running the catkin_make command for the first time will also create *CMakeLists.txt* link in your *src* folder.

Sourcing setup.bash :The final step is to execute a script called setup.bash , which is created by catkin_make inside the devel subdirectory of your workspace:

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

This automatically-generated script sets several environment variables that enable ROS to find your package and its newly-generated executables. It is analogous to the global setup.bash from Section 2.1, but tailored specifically to your workspace. Unless the directory structure changes, you only need to do this only once in each terminal, even if you modify the code and recompile with catkin_make ..

Alternatively one can configure account to run the setup.bash script automatically, each time you start a new shell. To do this, edit the file named .bashrc in your home directory, and put the above source command at the bottom, i.e.,

```
gedit ~/.bashrc
```

and provide the path of our workspace and at the end of the .bashrc file by writing the following line and save the file:

source ~/catkin_ws/devel/setup.bash

This will execute the above line automatically every time shell/terminal is opened.

To check if your workspace is properly overlayed enter the following in the terminal

```
echo $ROS_PACKAGE_PATH
```

You should see:

```
/home/username/catkin_ws/src:/opt/ros/melodic/share
```

Chapter 4: Setting up and controlling the P3dx robot in Gazebo

Pioneer 3dx robot is a modular, differential drive robot that can be used for several applications like mapping, reconnaissance, teleoperation, localization, monitoring, etc. In this chapter Pioneer 3dx (P3dx) will be controlled using ROS in a simulated environment.

4.1 Cloning of p3dx files

First change to the directory name src by the following command

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

You can clone the following file through terminal using git clone command in to SRC folder, i.e.,

```
git clone https://github.com/JenJenChung/pioneer_gazebo_ros.git
git clone https://github.com/naveenkumar2208/pioneer_description.git
git clone https://github.com/naveenkumar2208/pioneer_2dnav.git
git clone https://github.com/JenJenChung/nav_bundle.git
git clone https://github.com/JenJenChung/simple_navigation_goals.git
git clone https://github.com/naveenkumar2208/ua_ros_p3dx.git
```

Next go to the directory catkin_ws

```
cd ~/catkin_ws/
```

build the workspace using

```
catkin_make
```

and Source setup.bash

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

4.2 Moving P3dx in empty world

Launch the file *gazebo.launch* of the package *p3dx_gazebo:*

```
roslaunch p3dx_gazebo gazebo.launch
```

roslaunch provides a mechanism for starting the master and many nodes all at once, using a file called a launch file.

Now run *teleop_twist_keyboard.py executable of* the package *teleop_twist_keyboard* to control the robot through your keyboard in Gazebo environment:

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/pioneer/cmd_vel
```

cmd_vel:=/pioneer/cmd_vel will remap cmd_vel topic to /pioneer/cmd_vel

You can see the actual messages that are being published on the topic using the rostopic command:

```
rostopic echo /pioneer/cmd_vel
```

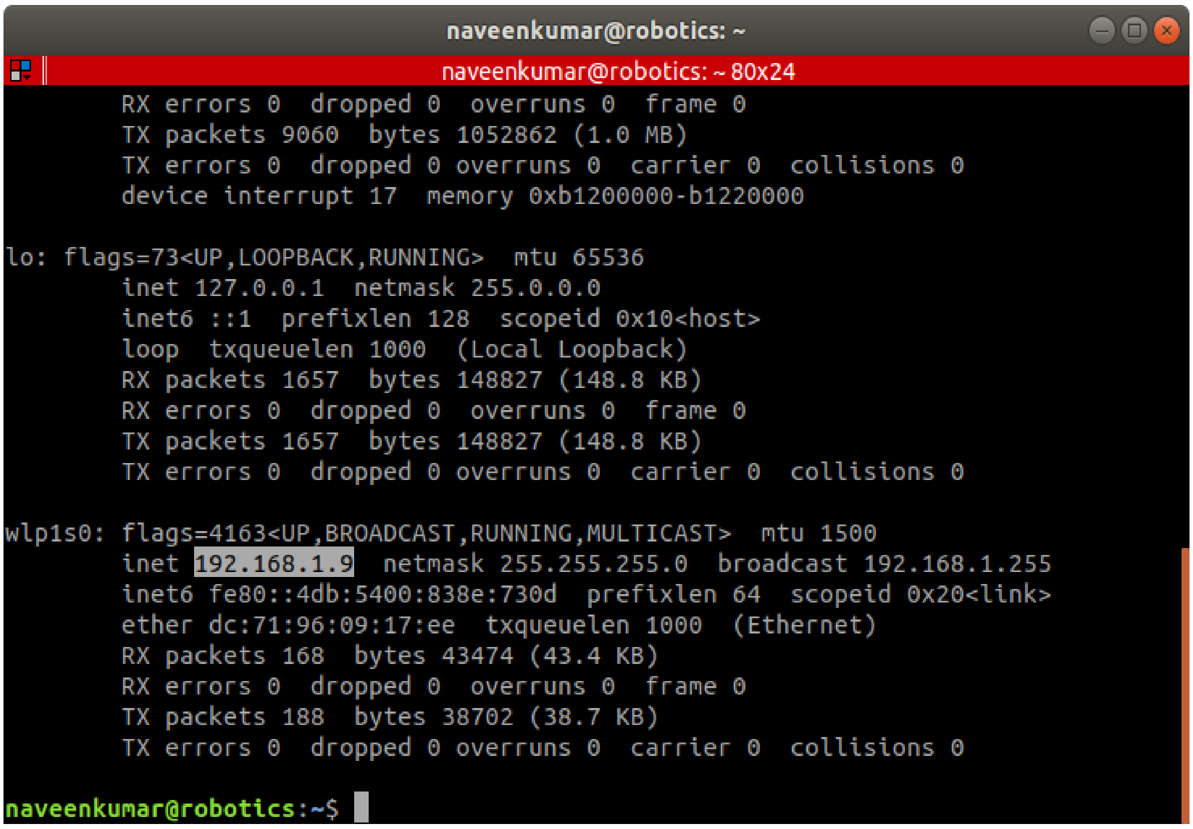
Now let us try the following command in a new terminal

```
rostopic pub -r 1 /pioneer/cmd_vel geometry_msgs/Twist '[2,0, 0]' '[0, 0, 0.5]'
```

When this step is performed the robot can be seen moving with the above rates.

4.3 Commanding Robot using your mobile phone

1. First of all, download the ROS Control application in your phone from the given link:https://play.google.com/store/apps/details?id=com.robotca.ControlApp&hl=en_IN
2. Next, connect your mobile and robot from the same wi-fi network.
3. Note the IP address of the robot's pc by typing "*ifconfig*" in the terminal. The inet address is the IP address of the computer as highlighted in the figure below.



```
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9060  bytes 1052862 (1.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 17  memory 0xb1200000-b1220000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1657  bytes 148827 (148.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1657  bytes 148827 (148.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.9  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::4db:5400:838e:730d  prefixlen 64  scopeid 0x20<link>
        ether dc:71:96:09:17:ee  txqueuelen 1000  (Ethernet)
        RX packets 168  bytes 43474 (43.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 188  bytes 38702 (38.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

naveenkumar@robotics:~$ 
```

Fig. 4.1 Screenshot of the terminal after executing ifconfig command

4. Then type the following commands in the terminal of your PC.

    export ROS_IP=**Enter IP of PC**

    roslaunch p3dx_gazebo gazebo.launch

    The command *export ROS_IP* sets the declared network address of a ROS Node. The same IP is also required to set up under ROS_MASTER_URI in the mobile phone so that it can locate the master.

5. Now open the application and click on "+" icon and fill up the details given below.

    The MASTER_URI: http://**Enter IP of PC**:11311/

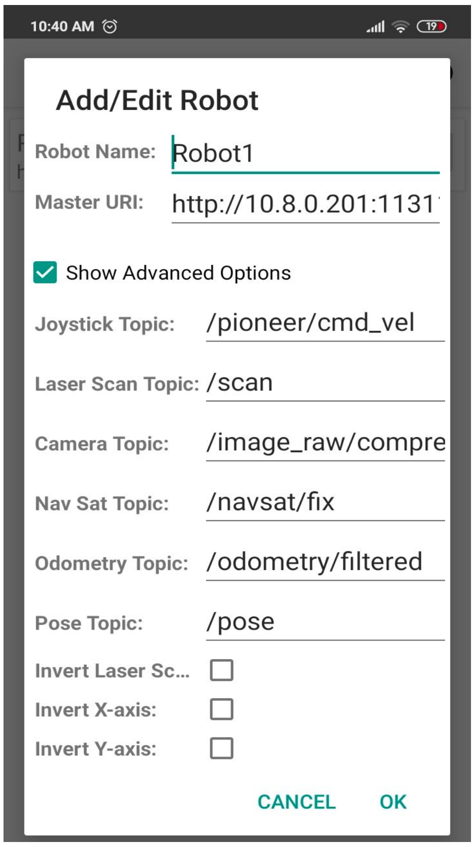
    Joystick topic : /pioneer/cmd_vel

    Pose topic : /pose

    Now, you can control the robot using a joystick.

    Warning: DON'T USE THE TILT MODE IN THE APPLICATION.

Sample details can be seen in the figure below:


10:40 AM

Add/Edit Robot

Robot Name: Robot1

Master URI: http://10.8.0.201:1131

☑ Show Advanced Options

Joystick Topic: /pioneer/cmd_vel

Laser Scan Topic: /scan

Camera Topic: /image_raw/compre

Nav Sat Topic: /navsat/fix

Odometry Topic: /odometry/filtered

Pose Topic: /pose

Invert Laser Sc... ☐

Invert X-axis: ☐

Invert Y-axis: ☐

4.4 Moving P3dx in cluttered environment and SLAM

First change the directory to *catkin_ws/src/pioneer_gazebo_ros*

```
cd ~/catkin_ws/src/pioneer_gazebo_ros
```

Run the script *run_pioneer_gazebo* using the command (use "./" for running a script):

```
./run_pioneer_gazebo
```

Now, run the executable *teleop_twist_keyboard.py*

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/cmd_vel
```

Messages that are being published on the topic can be seen using the rostopic command:

```
rostopic echo /cmd_vel
```

To move robot with user specified inputs use the following command with the above rate

```
rostopic pub -r 1 /cmd_vel geometry_msgs/Twist '[1, 0, 0]' '[0, 0, 0.05]'
```

4.5 Robot model description in Gazebo

The robot is modelled using a URDF (Unified Robot Description Format) file. This is a kinematic and basic description of a robot. Every link has three tags: <visual>, <collision>, and <inertial>.

<visual> tags describe the robot's appearance, position, etc. it has tags like <geometry> and <pose>, meshes of the robot are the files of the robot from any modelling software like SolidWorks. Basic shapes can be used in geometry tags like a box, cylinder, etc.

<collision> tags describe the physics of the body of the robot. They can be similar to the visual tags.

<inertial> tags describe the inertia of the link of the robot in every axis like I_{xx}, Iyy, I_{ZZ}, etc.

<joint> tags describe the position and type of the joint between a parent and a child link.

To enable actuation, <transmission> properties must be included in the model. The transmission block defines the actuator properties at a specified joint and sets up the interface for various Gazebo controller plugins.

A URDF file uses XML macro language which has many benefits like increasing modularity, reducing redundancy and permitting parametrization. This language can generate a URDF file instantly.

Gazebo plugins give your URDF models greater functionality and can tie in ROS messages and service calls for sensor output and motor input. In this tutorial, we explain how to setup pre-existing plugins and how to create your own custom plugins that can work with ROS.

Information on gazebo plugins is available here

http://gazebosim.org/tutorials/?tut=plugins_hello_world

Chapter 5: Interfacing P3dx with ROSARIA

The Pioneer 3-DX robot is an all-purpose mobile robot, used for research and applications involving mapping, teleoperation, localization, monitoring, reconnaissance and other behaviors. Pioneer 3-DX is characterized by a set of features listed below

characteristics	**values**
length	485 mm
width	381 mm
height	217 mm
weight	9 kgs

The Pioneer 3-DX platform is assembled with motors featuring 500-tick encoders, 19 cm wheels, tough aluminum body, 8 forward-facing ultrasonic (sonar) sensors and 1, 2 or 3 hot-swappable batteries, and a complete software development kit. The base Pioneer 3-DX platform can reach speeds of 1.6 meters per second and carry a payload of up to 23 kg.

The RosAria node provides a ROS interface for most Adept MobileRobots, MobileRobots Inc., and ActivMedia mobile robot bases including Pioneer 2, Pioneer 3, and any other past, current or future robot base supported by Adept MobileRobots open source ARIA library. This node implements the process of gathering information from the robot base, and controlling the velocity and acceleration of the robot. It publishes the topics providing data received from the robot's embedded controller and sets desired velocity, acceleration and other commands when new commands are received from command topic.

5.1 Installing the ROSARIA client interface

1. RosAria uses the Aria library from Adept MobileRobots to communicate with the robot. To download the latest version of RosAria for Ubuntu 16.04 (64-bit) or later, click on the following link. Install the .deb package by double clicking on the file. https://web.archive.org/web/20181103135921/http://robots.mobilerobots.com/ARIA/download/current/libaria_2.9.4+ubuntu16_amd64.deb (This step is already done at CC and in the provided virtual machine)
2. Next, you should create a workspace as discussed in Chapter 3, if not created yet.
3. Download the rosaria source code from its repository and bring into the workspace

```
cd ~/catkin_ws/src
git clone https://github.com/amor-ros-pkg/rosaria.git
cd ~/catkin_ws/
catkin_make
```

4. ROSARIA client interface can also be used to control the robot, to install the client interface enter the following commands, in the PC that will be used to control the robot.

```
cd ~/catkin_ws/src
git clone https://github.com/pengtang/rosaria_client.git
cd ~/catkin_ws/
catkin_make
source devel/setup.bash
```

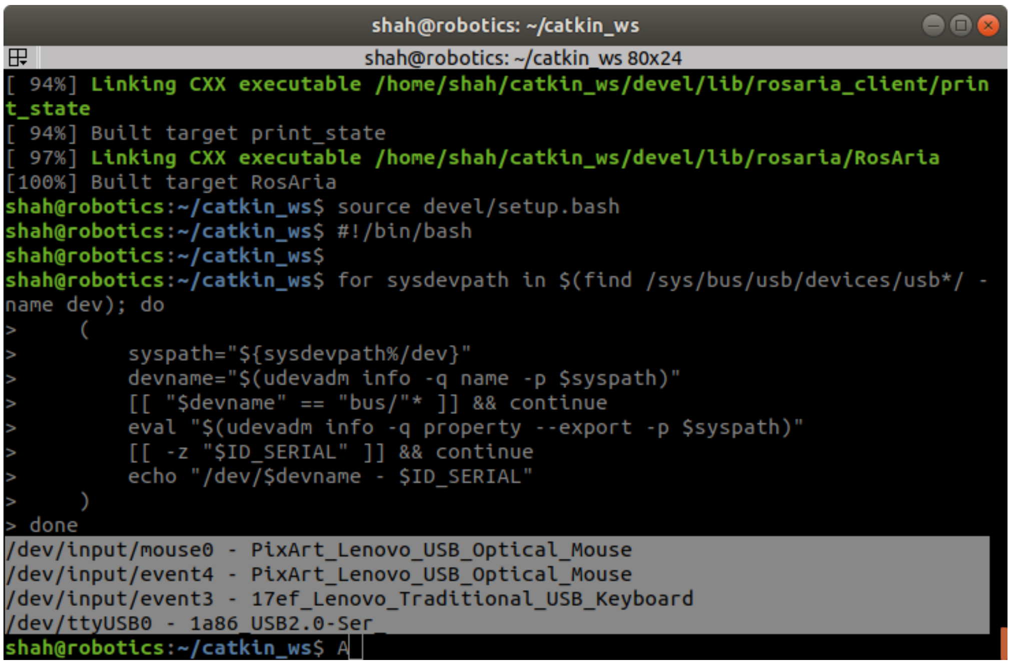
5.2 Connecting the host computer directly to robot through USB RS 232 cable.(Off-Board)

1. First of all, switch on the controller side of the motor where the serial port is located.
2. Connect the robot through your computer via a USB RS 232 cable.
3. Type this in the terminal to check that your PC is connected to the robot or not.

```
#!/bin/bash
```

```
for sysdevpath in $(find /sys/bus/usb/devices/usb*/ -name dev); do
   (
      syspath="${sysdevpath%/dev}"
      devname="$(udevadm info -q name -p $syspath)"
      [[ "$devname" == "bus/"* ]] && continue
      eval "$(udevadm info -q property --export -p $syspath)"
      [[ -z "$ID_SERIAL" ]] && continue
      echo "/dev/$devname - $ID_SERIAL"
   )
done
```

The highlighted portion in the figure below shows connected devices. Note down the dev path for the device. For example, the dev path of the robot is /dev/ttyUSB0.



```
shah@robotics: ~/catkin_ws
shah@robotics: ~/catkin_ws 80x24
[ 94%] Linking CXX executable /home/shah/catkin_ws/devel/lib/rosaria_client/prin
t_state
[ 94%] Built target print_state
[ 97%] Linking CXX executable /home/shah/catkin_ws/devel/lib/rosaria/RosAria
[100%] Built target RosAria
shah@robotics:~/catkin_ws$ source devel/setup.bash
shah@robotics:~/catkin_ws$ #!/bin/bash
shah@robotics:~/catkin_ws$
shah@robotics:~/catkin_ws$ for sysdevpath in $(find /sys/bus/usb/devices/usb*/ -
name dev); do
>     (
>         syspath="${sysdevpath%/dev}"
>         devname="$(udevadm info -q name -p $syspath)"
>         [[ "$devname" == "bus/"* ]] && continue
>         eval "$(udevadm info -q property --export -p $syspath)"
>         [[ -z "$ID_SERIAL" ]] && continue
>         echo "/dev/$devname - $ID_SERIAL"
>     )
> done
/dev/input/mouse0 - PixArt_Lenovo_USB_Optical_Mouse
/dev/input/event4 - PixArt_Lenovo_USB_Optical_Mouse
/dev/input/event3 - 17ef_Lenovo_Traditional_USB_Keyboard
/dev/ttyUSB0 - 1a86_USB2.0-Ser_
shah@robotics:~/catkin_ws$ A
```

4. Run the following command in the existing terminal

```
roscore
```

5. Open one more terminal in and run the following two commands:

```
sudo chmod a+rw /dev/ttyUSB0
```

This gives permission to read, write and execute. This will add your user to change the permissions of the serial port device to allow all users to access it.

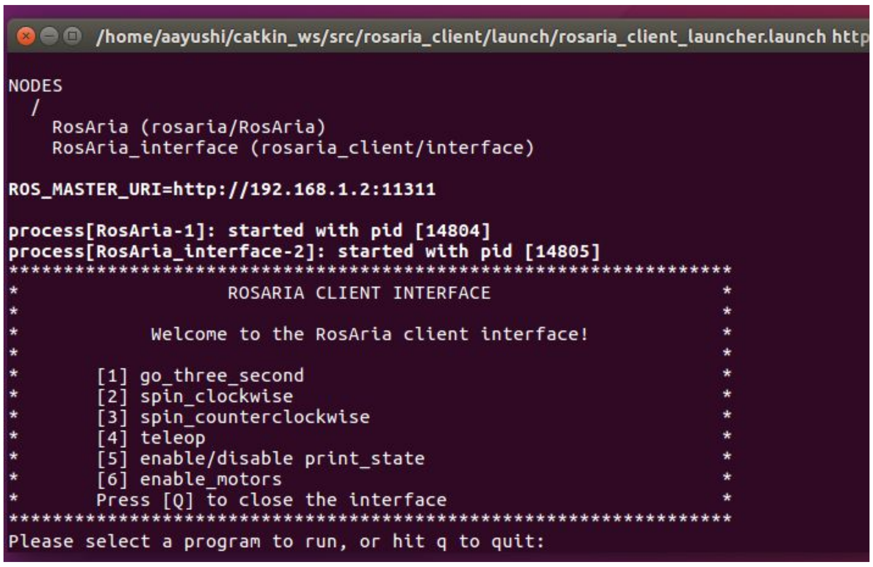
The default port is /dev/ttyUSB0, the following command activates RosAria node of package rosaria and connects Linux laptop or other computer to the robot via a USB-serial adapter, and also on the Pioneer LX which uses USB-serial adapters:

```
rosrun rosaria RosAria _port:=/dev/ttyUSB0
```

6. Open third terminal and type the following commands

```
rosrun rosaria_client interface
```

The robot is now connected to the PC and can be controlled using rosaria.



```
/home/aayushi/catkin_ws/src/rosaria_client/launch/rosaria_client_launcher.launch http

NODES
  /
    RosAria (rosaria/RosAria)
    RosAria_interface (rosaria_client/interface)

ROS_MASTER_URI=http://192.168.1.2:11311

process[RosAria-1]: started with pid [14804]
process[RosAria_interface-2]: started with pid [14805]
**********************************************************************
*                      ROSARIA CLIENT INTERFACE                      *
*                                                                    *
*              Welcome to the RosAria client interface!              *
*                                                                    *
*        [1] go_three_second                                         *
*        [2] spin_clockwise                                          *
*        [3] spin_counterclockwise                                   *
*        [4] teleop                                                  *
*        [5] enable/disable print_state                              *
*        [6] enable_motors                                           *
*        Press [Q] to close the interface                            *
**********************************************************************
Please select a program to run, or hit q to quit:
```

Then a set of options will appear in the terminal as shown above, choose the option wanted. You may select teleop for control using arrow keys. The following keys can be used:

- Up Arrow: Move forward
- Down Arrow: Move backward
- Left Arrow: Turn counter-clockwise
- Right Arrow: Turn clockwise
- Spacebar: Stop moving
- A: Increase linear velocity
- Z: Decrease linear velocity
- S: Increase angular velocity
- X: Decrease angular velocity

- Q: Quit teleop

Alternatively one can use teleop_twist_keyboard using the following command

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
cmd_vel:=/RosAria/cmd_vel
```

to control the robot through the keyboard.

5.3 Control the robot by its own computer

1. Connect the robot with mouse, monitor and keyboard.
2. Open terminal 1 and run roscore

   ```
   roscore
   ```

3. Open second terminal and give permission to the Robot PC to read and write on Robot controller

   ```
   sudo chmod a+rw /dev/ttyS0
   ```

   run RosAria node of package rosaria to connect to the robot in the same terminal

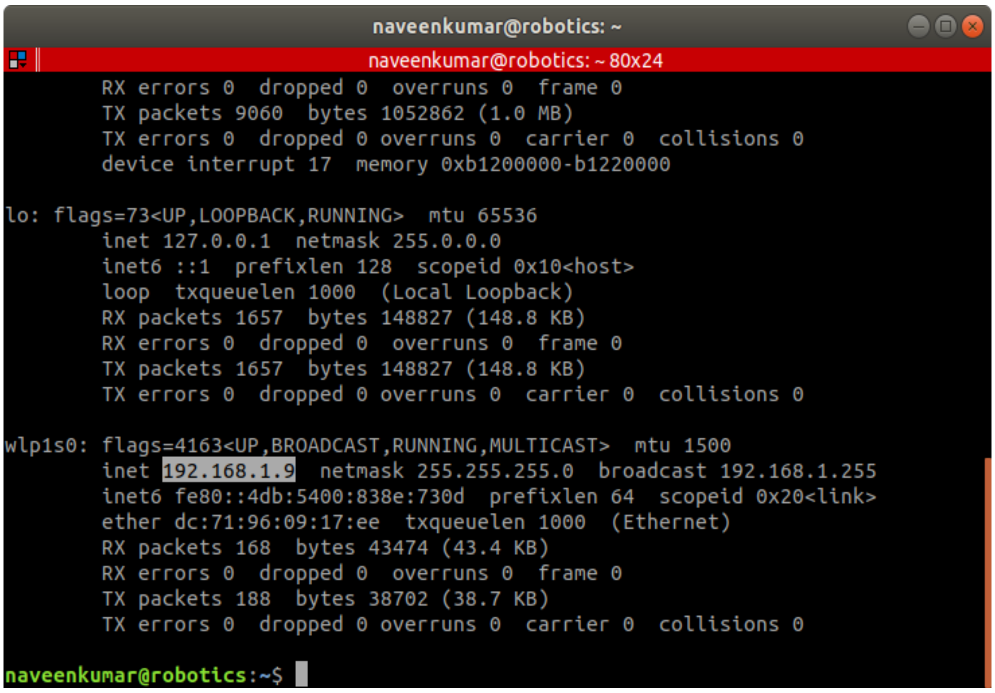
   ```
   rosrun rosaria RosAria _port:=/dev/ttyS0
   ```

4. Open third terminal and enter the following command to control the robot

   ```
   rosrun rosaria_client interface
   ```

5.4 Connect P3dx through on board computer in wireless manner using SSH

1. Connect the robot with mouse, monitor and keyboard.
2. Check the ip address of the robot's pc by typing "ifconfig" in the terminal.We need The inet address is the ip address of the computer.



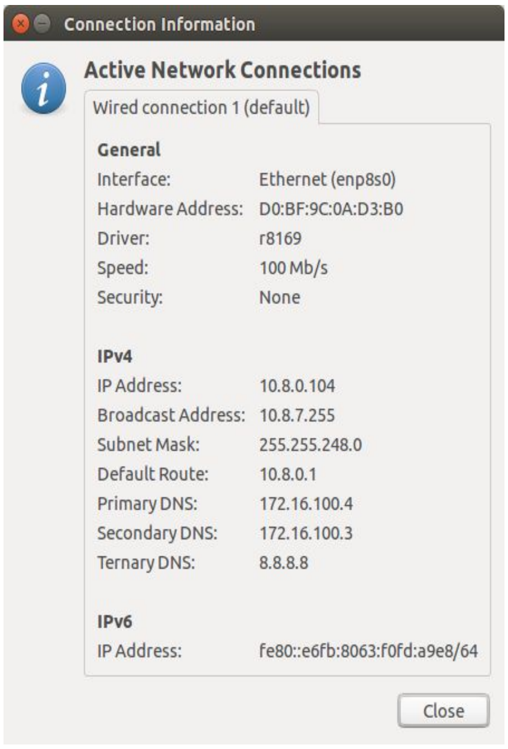
```
naveenkumar@robotics: ~
naveenkumar@robotics: ~ 80x24
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9060  bytes 1052862 (1.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 17  memory 0xb1200000-b1220000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1657  bytes 148827 (148.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1657  bytes 148827 (148.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.9  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::4db:5400:838e:730d  prefixlen 64  scopeid 0x20<link>
        ether dc:71:96:09:17:ee  txqueuelen 1000  (Ethernet)
        RX packets 168  bytes 43474 (43.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 188  bytes 38702 (38.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

naveenkumar@robotics:~$ 
```

Alternatively you can see the ip address of the robot by going to connection information by clicking on either the wifi sign or LAN sign on the top right corner. Check IPv4 IP address.



Connection Information

Active Network Connections

Wired connection 1 (default)

General

Interface:	Ethernet (enp8s0)
Hardware Address:	D0:BF:9C:0A:D3:B0
Driver:	r8169
Speed:	100 Mb/s
Security:	None

IPv4

IP Address:	10.8.0.104
Broadcast Address:	10.8.7.255
Subnet Mask:	255.255.248.0
Default Route:	10.8.0.1
Primary DNS:	172.16.100.4
Secondary DNS:	172.16.100.3
Ternary DNS:	8.8.8.8

IPv6

IP Address:	fe80::e6fb:8063:f0fd:a9e8/64

3. Open the terminal window in 3 tabs in client's PC in each tab type

```
ssh name of the robot's pc@ROBOT IP
```

   for robot name r28 and IP: 10.8.0.104, *"ssh r28@10.8.0.104"*. (for reference you can watch this video https://youtu.be/-9NHupBPC6Y)
4. type the password of the robot in each terminal.
5. Open terminal 1 and run roscore

```
roscore
```

6. Open second terminal and give permission to the Robot PC to read and write on Robot controller

```
sudo chmod a+rw /dev/ttyS0
```

   run RosAria node of package rosaria to connect to the robot in the same terminal

```
rosrun rosaria RosAria _port:=/dev/ttyS0
```

7. Open third terminal and enter the following command to control the robot

```
rosrun rosaria_client interface
```

5.5 Connect P3dx through on board computer in wireless manner through ROS

1. The client PC and the robot to the same network.
2. Connect the robot with a monitor, a mouse, and a keyboard. Start the terminal
3. Check the ip of the robot and computer, type *ifconfig* or check connection properties at the right hand corner where you can see either a wifi sign or up-down arrow sign.
4. Perform the following step in client PC Export client PC IP to ROS

```
export ROS_IP=Client IP
```

   and run roscore

```
roscore
```

5. Next, Perform the following steps in Robot's PC:
   Export Robot PC to ROS

```
export ROS_IP=Robot IP
```

   Connect the robot to the master (client) PC

```
export ROS_MASTER_URI=http://Client IP:11311
```

Give permission to the master (client) PC to read and write on Robot controller

```
sudo chmod a+rw /dev/ttyS0
```

run RosAria node of package rosaria to connect to the robot

```
rosrun rosaria RosAria _port:=/dev/ttyS0
```

6. In the client PC start new terminal and enter the following commands to control robot through your keyboard

```
export ROS_IP=Client IP
```

and

```
rosrun rosaria_client interface
```

5.6 Topics and Commands

RosAria provides us with various commands and topics for publishing data received from the robot.

Pose

RosAria publishes the current pose estimation received from the robot in /RosAria/pose. The type of this data is nav_msgs/Odometry. (This pose estimation is calculated by the robot firmware as ARIA based on wheel odometry, and if present, the optional internal gyroscope.) This pose estimation is received from the robot every 100ms (by default), and is then published on the /RosAria/pose topic.

To view /RosAria/pose using rostopic use:

```
rostopic echo /RosAria/pose
```

Velocity Commands

The desired velocity of the robot is set via the /RosAria/cmd_vel topic.You do not need to send commands at any fixed rate to maintain robot velocity, ARIA and the robot controller maintain the velocities. Velocity commands only need to be sent to change the desired velocities.

A standard type for robot motion commands in ROS is geometry_msgs/Twist, which contains many components (dimensions). The relevant twist components for Pioneer mobile robots are linear x (forward/back translational velocity) and angular z (rotational velocity).
The syntax for specifying twist data using the rostopic command is as follows (this example sets a linear velocity of 0.1 meters/s):

```
rostopic pub -1 /RosAria/cmd_vel geometry_msgs/Twist '[0.1, 0.0, 0.0]' '[0.0, 0.0, 0.0]'
```

Enable/Disable Motors

Whether the robot's motors are enabled or disabled is published as a boolean value via the /RosAria/motors_state topic.
To enable the motors, use

```
/RosAria/enable_motors service
```

To disable the motors, use

```
/RosAria/disable_motors service
```

5.6 Control the robot using Android application

1. First of all, switch on the controller side of the motor where the serial port is located.
2. Connect the robot through your computer via a USB RS 232 cable.
3. Download the ROS Control application in your phone from the given link. https://play.google.com/store/apps/details?id=com.robotca.ControlApp&hl=en_IN
4. Connect your mobile and PC from the same wi-fi network. IP can be checked using *ifconfig* command.
5. In PC, type the following commands.

```
export ROS_IP=IP of PC
roscore
```

In second terminal, type following commands

```
export ROS_IP=IP of PC
```

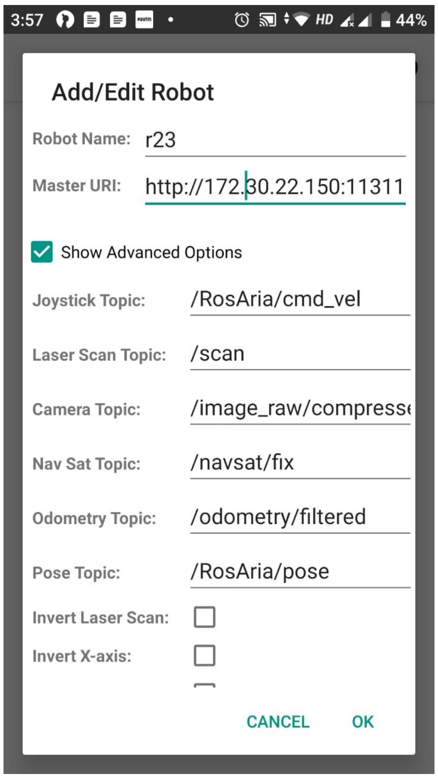
Next enter

```
sudo chmod a+rw /dev/ttyUSB0
```

Enter password when prompted

```
rosrun rosaria RosAria _port:=/dev/ttyUSB0
```

6. Next open the application and click on "+" icon and Now fill the details as shown in picture below.



3:57 44%

Add/Edit Robot

Robot Name: r23

Master URI: http://172.30.22.150:11311

☑ Show Advanced Options

Joystick Topic: /RosAria/cmd_vel

Laser Scan Topic: /scan

Camera Topic: /image_raw/compresse

Nav Sat Topic: /navsat/fix

Odometry Topic: /odometry/filtered

Pose Topic: /RosAria/pose

Invert Laser Scan: ☐

Invert X-axis: ☐

CANCEL OK

The MASTER_URI: http:// **IP of PC**:11311/

Joystick topic : /RosAria/cmd_vel

Pose topic : /RosAria/pose

Now, you can control the robot using joystick.

Warning: DON'T USE THE TILT MODE IN THE APPLICATION.

Chapter 6: Mapping with Microsoft Kinect

Microsoft Kinect is primarily used for motion sensing as a gaming accessory and to capture the motion of a human subject and that person's voice commands, but its applications also lie in robotics as it is frequently used as a motion sensor, mapping device, etc. It has the following components depicted in fig.4.1:

- An infrared sensor that transmits and receives infrared beams.
- A 3D depth camera which can sense the distance of the objects in the room, from it
- An RGB sensor which receives visible light from the subjects.
- A monochrome sensor only receives a black color.
- An array of four microphones which can accurately compute and isolate the source of the sound (for example voice commands from a human). This allows the person to be a bit far from the Kinect and still give voice commands.
- A color VGA video camera detects features in three colors: Red, Blue and Green.

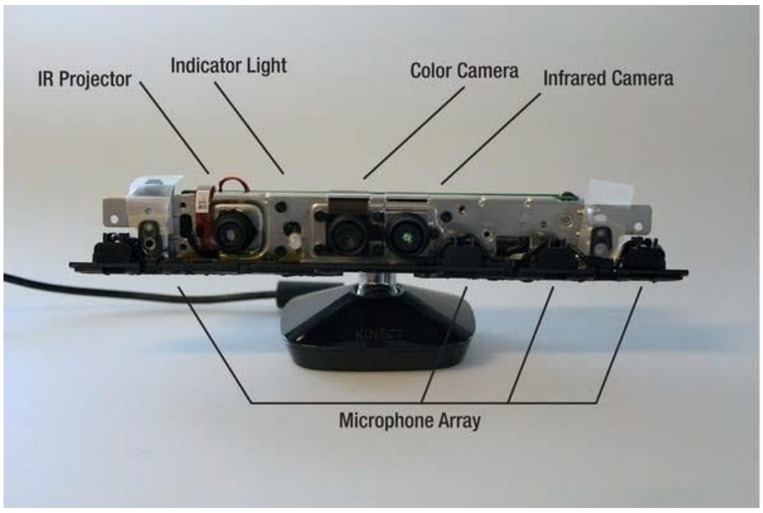


Fig. 4.1 Picture showing the hardware of the Kinect

6.1 Connecting Laptop, Mobile and Robot

For mapping we would be using a laptop for map creation whereas the motion of the rob ot would be commanded through mobile. For this the first step is to connect mobile and laptop and robot.

For connecting laptop and mobile follow the procedure given under **Section 5.6**.

6.2. Installation of libfreenect and freenect stack in the catkin_ws

Open new terminal and clone of Libfreenect and freenect

```
cd ~/catkin_ws/src
git clone https://github.com/ros-drivers/libfreenect.git
git clone https://github.com/ros-drivers/freenect_stack.git
```

Installing Libfreenect and freenect

```
mkdir -p ~/catkin_ws/devel/include/libfreenect
cp ~/catkin_ws/src/libfreenect/include/libfreenect_audio.h
~/catkin_ws/devel/include/libfreenect/
cp ~/catkin_ws/src/libfreenect/include/libfreenect.h
~/catkin_ws/devel/include/libfreenect/
cp ~/catkin_ws/src/libfreenect/include/libfreenect_registration.h
~/catkin_ws/devel/include/libfreenect/
cd ~/catkin_ws
catkin_make
source devel/setup.bash
```

The above commands will create "libfreenect" folder in /home/catkin_ws/devel/include and copy "libfreenect.h", "libfreenect_audio.h", "libfreenect_registration.h" from /home/catkin_ws/src/libfreenect/include/ to /home/catkin_ws/devel/include/libfreenect/ .

6.3 Mapping using Kinect

Before we use kinect we have to install the kinect driver from the source in our catkin_ws but we also required some openni drivers to be installed in our machine. For detailed installation procedure for Openni drivers please refer to the Annexure-2.

Note:- Openni drivers are already installed in the Computer Center and the virtual machine provided with this course

Open a new terminal(ctr+al+t) and create four sub terminals using (split on right click on mouse). Now enter the following commands in four terminals.

Subterminal 1 (Connecting Camera)

```
roslaunch freenect_launch freenect.launch depth_registration:=true
```

Subterminal 2 (Making pseudo laser image from depth image)

```
rosrun depthimage_to_laserscan depthimage_to_laserscan
image:=/camera/depth_registered/image_raw
```

Subterminal 3 (Establishes pose of camera with reference to the robot)

```
rosrun tf static_transform_publisher 0 0 0 0 0 0 base_link camera_link 100
```

Subterminal 4 (Launching Rviz and RTAB Map)

```
roslaunch rtabmap_ros rtabmap.launch rtabmap_args:="--delete_db_on_start" rviz:=true
rtabmapviz:=true
```

The “delete_dbs_onstart” argument clears the database whenever rtabmap is started, to continue mapping on the previous map, remove the “delete_dbs_onstart” argument in the command. you move the robot using your mobile and may see map creation

If you are closing the terminal with incomplete mapping the following command can be executed instead of the above to start mapping from where you have left.

```
roslaunch rtabmap_ros rtabmap.launch ratbmap_args:= rviz:=true rtabmapviz:=true
```

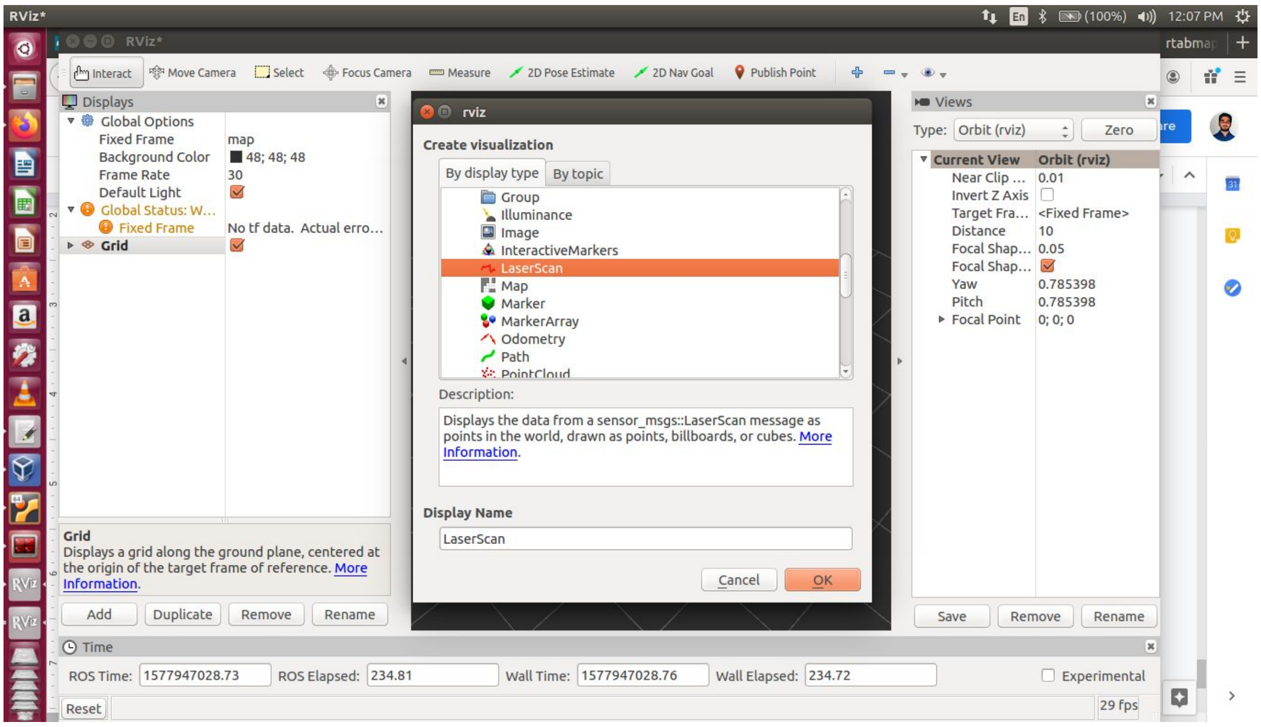
Note that the position and orientation of the robot should be the same.

6.4 Saving the map

Now Rviz must start and a visual of what is captured by Kinect can be seen in point cloud images, the robot must be moved around to map the area.

In order to get the 2D map of the environment, perform the following steps.

(i) Click on add (left side at the bottom) in rviz and choose LaserScan from the list



Create visualization

Description:

Displays the data from a sensor_msgs::LaserScan message as points in the world, drawn as points, billboards, or cubes. More Information.

Display Name: LaserScan

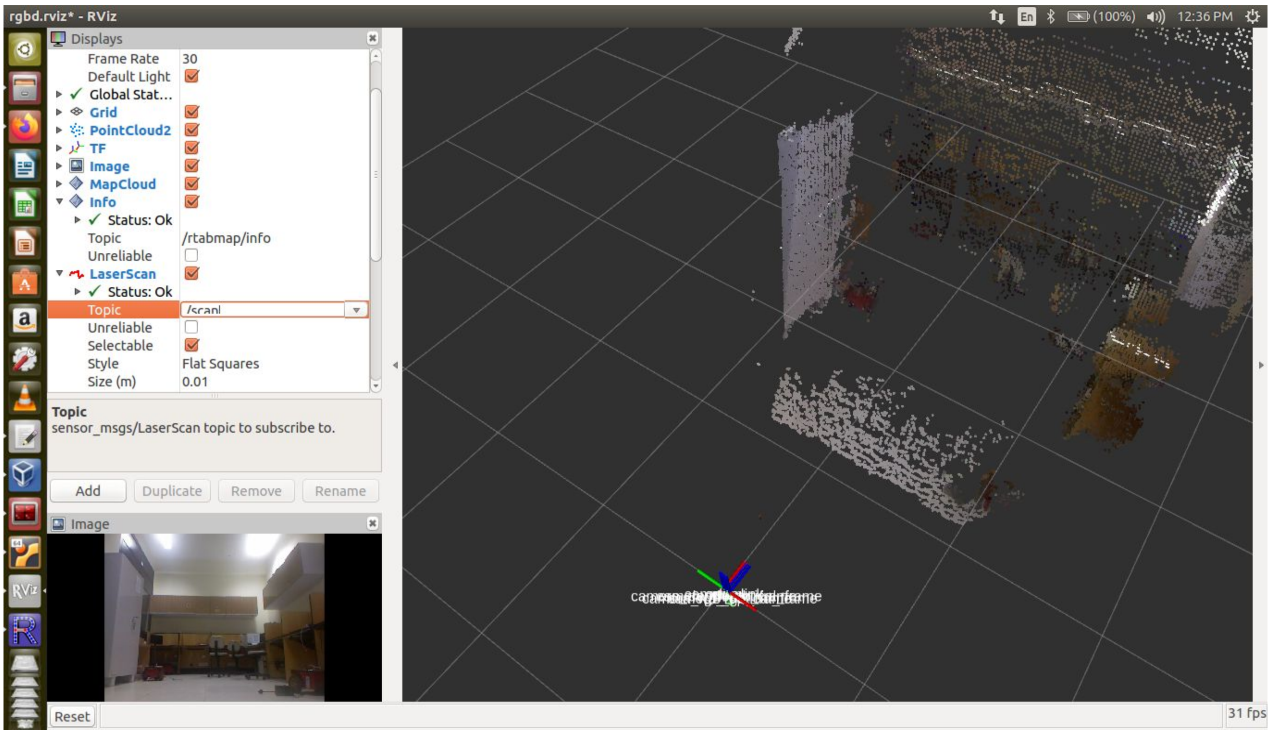
Global Options
Fixed Frame: map
Background Color: 48; 48; 48
Frame Rate: 30
Global Status: W... Fixed Frame: No tf data. Actual erro...

Grid
Displays a grid along the ground plane, centered at the origin of the target frame of reference. More Information.

Current View: Orbit (rviz)
Near Clip ...: 0.01
Target Fra...: <Fixed Frame>
Distance: 10
Focal Shap...: 0.05
Yaw: 0.785398
Pitch: 0.785398
Focal Point: 0; 0; 0

ROS Time: 1577947028.73 ROS Elapsed: 234.81 Wall Time: 1577947028.76 Wall Elapsed: 234.72

(ii) In LaserScan topic choose /scan

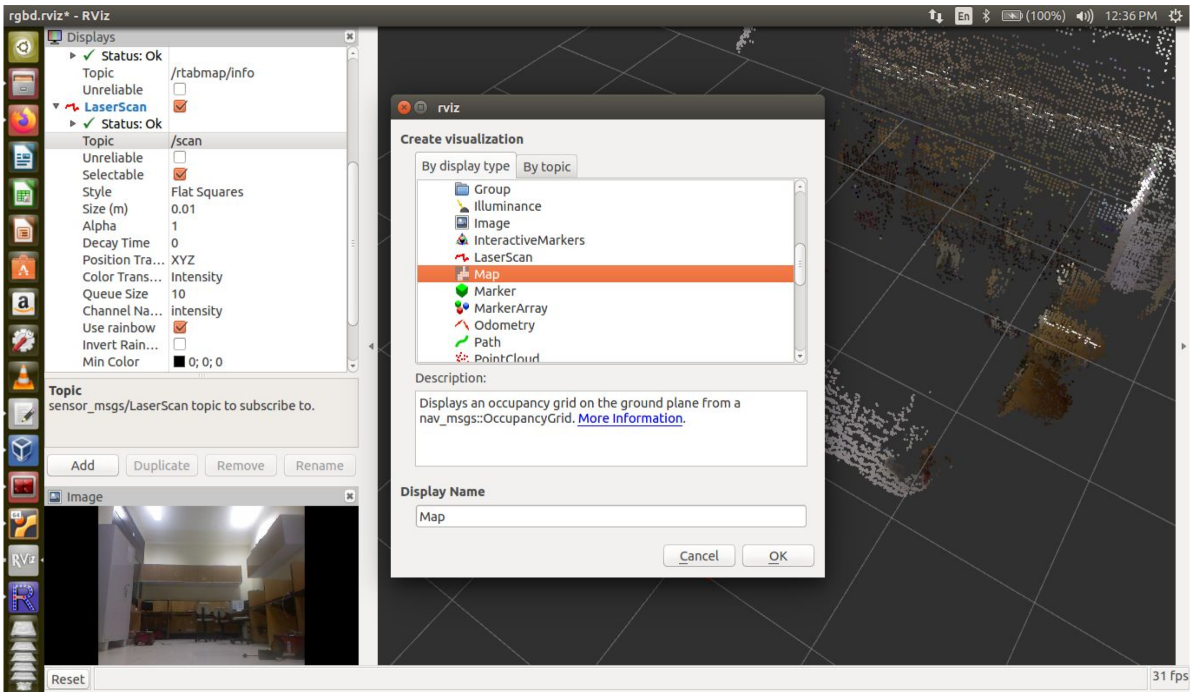


Frame Rate: 30
Info Topic: /rtabmap/info
LaserScan
Status: Ok
Topic: /scan
Style: Flat Squares
Size (m): 0.01

Topic
sensor_msgs/LaserScan topic to subscribe to.

(iii) Click on add again and choose map.

(iv)in map, choose /rtabmap/grid_map as a topic in it.

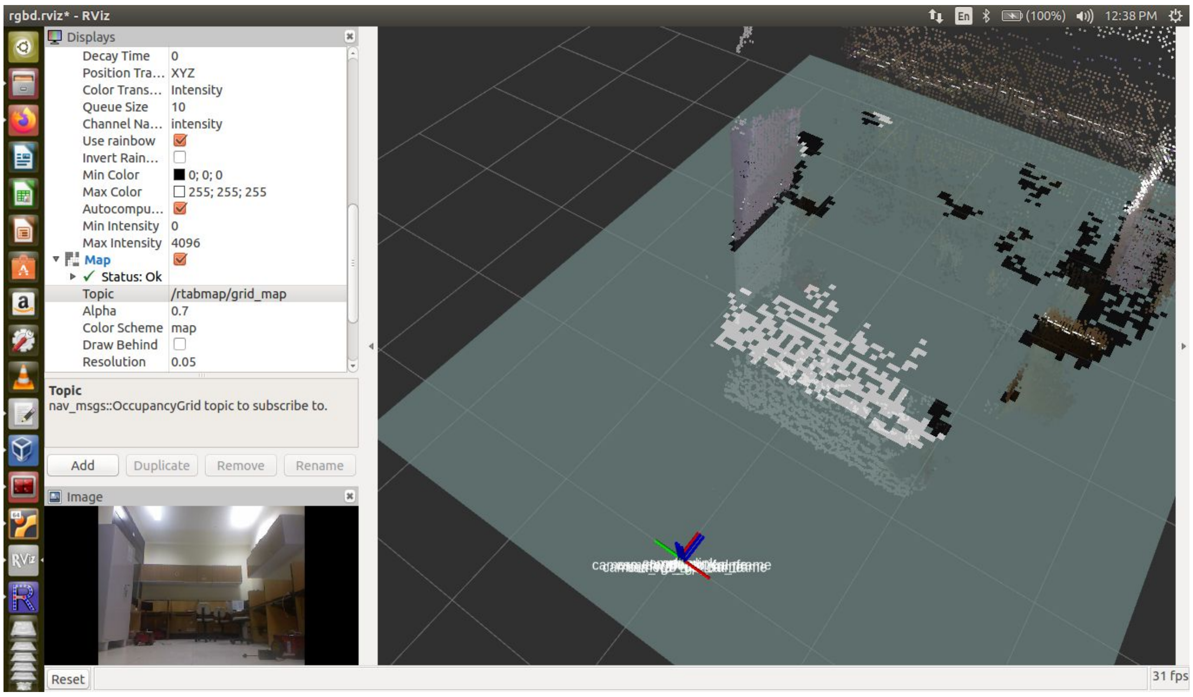


Status: Ok
Topic /rtabmap/info
Unreliable ☐
LaserScan ✓
Status: Ok
Topic /scan
Unreliable ☐
Selectable ✓
Style Flat Squares
Size (m) 0.01
Alpha 1
Decay Time 0
Position Tra... XYZ
Color Trans... Intensity
Queue Size 10
Channel Na... intensity
Use rainbow ✓
Invert Rain... ☐
Min Color ■ 0; 0; 0

Topic
sensor_msgs/LaserScan topic to subscribe to.

Description:
Displays an occupancy grid on the ground plane from a nav_msgs::OccupancyGrid. More Information.

Display Name
Map



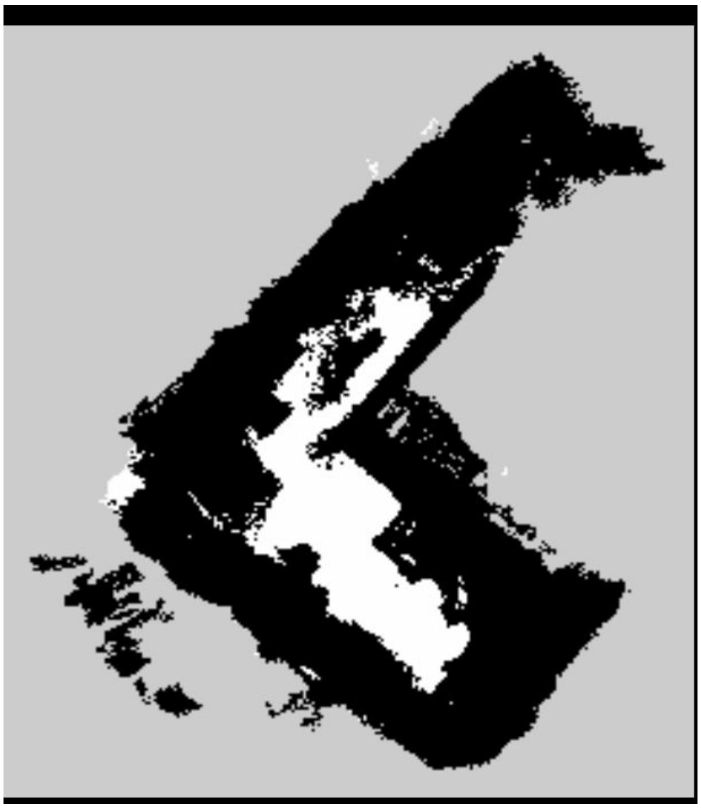
Decay Time 0
Position Tra... XYZ
Color Trans... Intensity
Queue Size 10
Channel Na... intensity
Use rainbow ✓
Invert Rain... ☐
Min Color ■ 0; 0; 0
Max Color ☐ 255; 255; 255
Autocompu... ✓
Min Intensity 0
Max Intensity 4096
Map ✓
Status: Ok
Topic /rtabmap/grid_map
Alpha 0.7
Color Scheme map
Draw Behind ☐
Resolution 0.05

Topic
nav_msgs::OccupancyGrid topic to subscribe to.

After completing the mapping it needs to be saved enter the command below to save it.

```
rosrun map_server map_saver map:=/rtabmap/grid_map
```

For more information one may visitr http://wiki.ros.org/map_server



Map saved in .pgm format

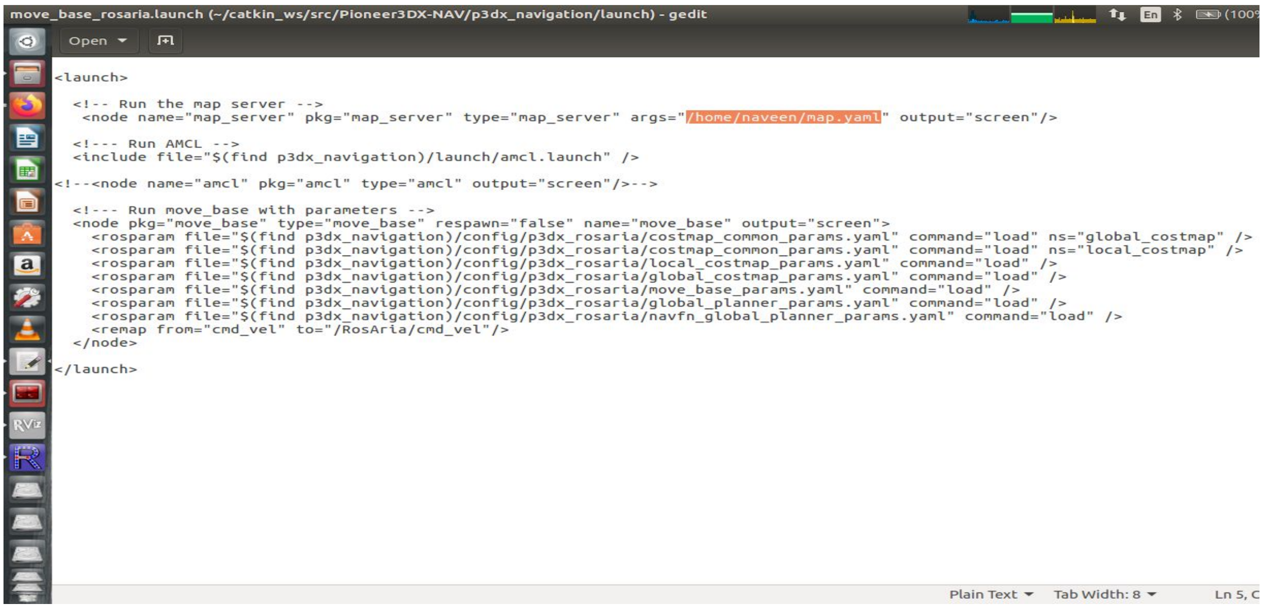
Chapter 7 Navigation in ROS

7.1 Setting up of the navigation stack for the robot

Now, we have the map of our surroundings, we want our robot to move around autonomously, avoiding obstacles on its way, once we point it to it's goal. ROS helps us achieve that by using the navigation stack. But first we need to set our robot up for it. Type following commands in the terminal to clone the navigation package:

```
cd ~/catkin_ws/src
git clone https://github.com/naveenkumar2208/Pioneer3DX-NAV
cd ~/catkin_ws/
catkin_make
source devel/setup.bash
```

When the package is ready, we have to modify the file "move_base_rosaria.launch" located in /home/your_user_name/catkin_ws/src/Pioneer3DX-NAV/p3dx_navigation/launch. Replace "Naveen" in the path of the map highlighted below with username of your pc


move_base_rosaria.launch (~/catkin_ws/src/Pioneer3DX-NAV/p3dx_navigation/launch) - gedit
Open
Plain Text
Tab Width: 8
Ln 5, C

```
<launch>

  <!-- Run the map server -->
   <node name="map_server" pkg="map_server" type="map_server" args="/home/naveen/map.yaml" output="screen"/>

  <!--- Run AMCL -->
  <include file="$(find p3dx_navigation)/launch/amcl.launch" />

<!--<node name="amcl" pkg="amcl" type="amcl" output="screen"/>-->

  <!--- Run move_base with parameters -->
  <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
    <rosparam file="$(find p3dx_navigation)/config/p3dx_rosaria/costmap_common_params.yaml" command="load" ns="global_costmap" />
    <rosparam file="$(find p3dx_navigation)/config/p3dx_rosaria/costmap_common_params.yaml" command="load" ns="local_costmap" />
    <rosparam file="$(find p3dx_navigation)/config/p3dx_rosaria/local_costmap_params.yaml" command="load" />
    <rosparam file="$(find p3dx_navigation)/config/p3dx_rosaria/global_costmap_params.yaml" command="load" />
    <rosparam file="$(find p3dx_navigation)/config/p3dx_rosaria/move_base_params.yaml" command="load" />
    <rosparam file="$(find p3dx_navigation)/config/p3dx_rosaria/global_planner_params.yaml" command="load" />
    <rosparam file="$(find p3dx_navigation)/config/p3dx_rosaria/navfn_global_planner_params.yaml" command="load" />
    <remap from="cmd_vel" to="/RosAria/cmd_vel"/>
  </node>

</launch>
```

7.2 Connecting laptop with robot

The procedure to connect the robot with the laptop is similar to Section 5.2, i.e., run the following command in the existing terminal

```
roscore
```

Open one more terminal in and run the following two commands:

```
sudo chmod a+rw /dev/ttyUSB0
```

This gives permission to read, write and execute. This will add your user to change the permissions of the serial port device to allow all users to access it.

The default port is /dev/ttyUSB0, the following command activates RosAria node of package rosaria and connects Linux laptop or other computer to the robot via a USB-serial adapter, and also on the Pioneer LX which uses USB-serial adapters:

```
rosrun rosaria RosAria _port:=/dev/ttyUSB0
```

7.2 Loading Camerta and Map in Rviz ROS

First go to folder /home/your_user_name/catkin_ws/src/Pioneer3DX-NAV/ and copy files **map.pgm** and **map.yaml** to /home/your_user_name. Alternatively, you can use the map created by you in Chapter 6.

Open a new terminal(ctr+al+t) and create five sub terminals using (split on right click on mouse). Now enter the following commands in five terminals.

Subterminal 1 (Connecting Camera)

```
roslaunch freenect_launch freenect.launch depth_registration:=true
```

Subterminal 2 (Making pseudo laser image from depth image)

```
rosrun depthimage_to_laserscan depthimage_to_laserscan
image:=/camera/depth_registered/image_raw
```

Subterminal 3 (Establishes pose of camera with reference to the robot)

```
rosrun tf static_transform_publisher 0 0 0 0 0 0 base_link camera_link 100
```

Subterminal 4 (Opening Rviz)

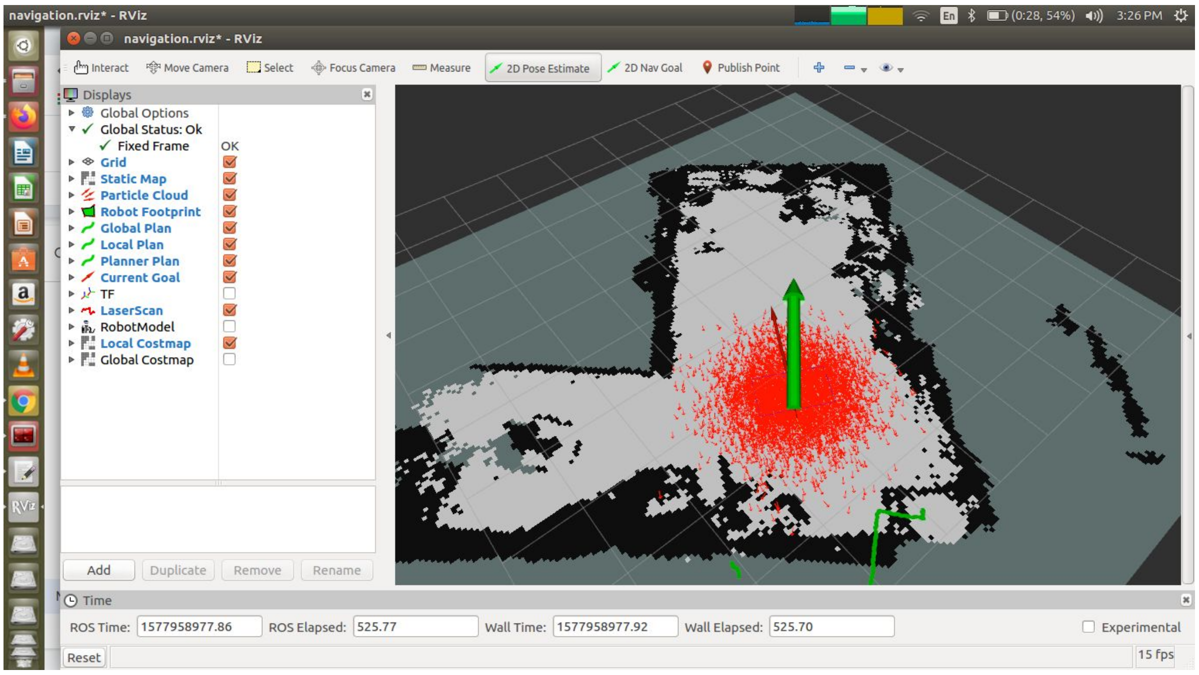
```
roslaunch p3dx_navigation rviz_p3dx.launch
```

Subterminal 5 (Connect robot to Rviz and load the map)

```
roslaunch p3dx_navigation move_base_rosaria.launch
```

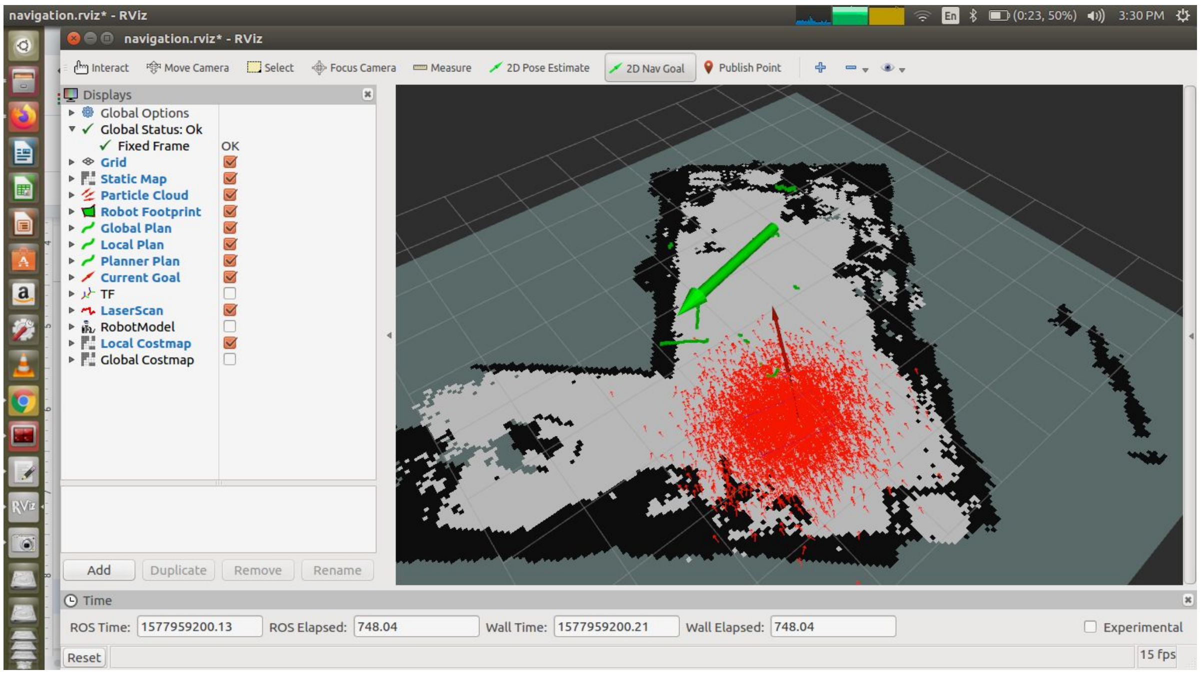
7.3 Navigation

Click on 2D pose estimate in rviz window to set the location and orientation of the robot in the current map. Green arrow represents the pose location and orientation of the robot. Please verify the prescribed location with the LAser data shown with green blinking boundaries



ROS Time: 1577958977.86 ROS Elapsed: 525.77 Wall Time: 1577958977.92 Wall Elapsed: 525.70

Click on 2D Nav Goal to set the goal location and orientation of the robot.



ROS Time: 1577959200.13 ROS Elapsed: 748.04 Wall Time: 1577959200.21 Wall Elapsed: 748.04

(the green arrow represents the goal location and orientation of the robot the red arrow represents the current location and orientation of the robot.)

Chapter 8 How to get video feed from webcam or USB cam connected to PC to Smartphone

We can do this by two methods.

1. **Using ros usb_cam driver.**

(i). First of all install usb_cam driver by typing following command.

```
sudo apt install ros-melodic-usb-cam
```

(ii). Download and install the app ROS Teleop Controller AuTURBO from playstore from this link:
https://play.google.com/store/apps/details?id=org.ros.android.controllerSample&hl=en_us

(iii). Connect both PC and phone with the same wifi network.

(iv). Run the following commands in your PC's terminal. IP can be checked using *ifconfig* command.

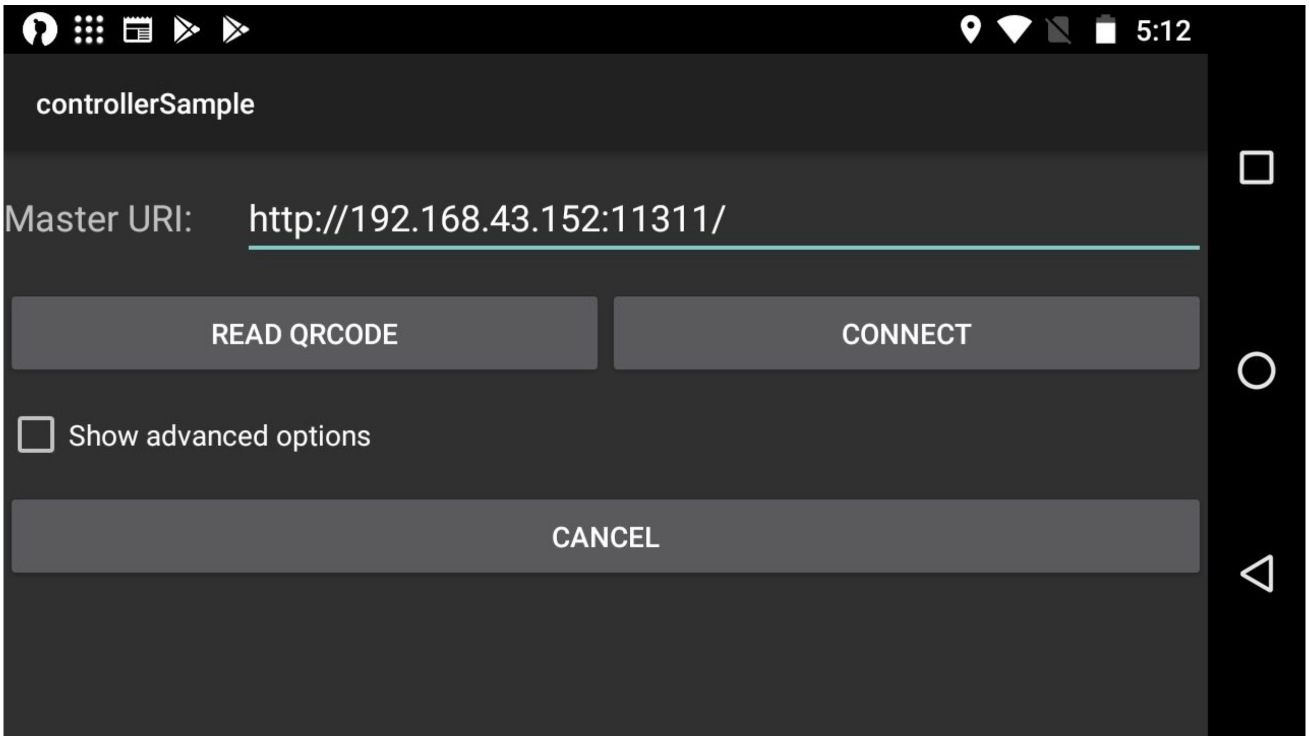
```
export ROS_IP=IPofPC
roscore
```

(v). In another terminal, run these commands:

```
export ROS_IP=IPofPC
rosrun usb_cam usb_cam_node
```

(vi) Open the mobile application and put the Master URI according to the PC.

Master URI: http://**IPofPC**:11311/

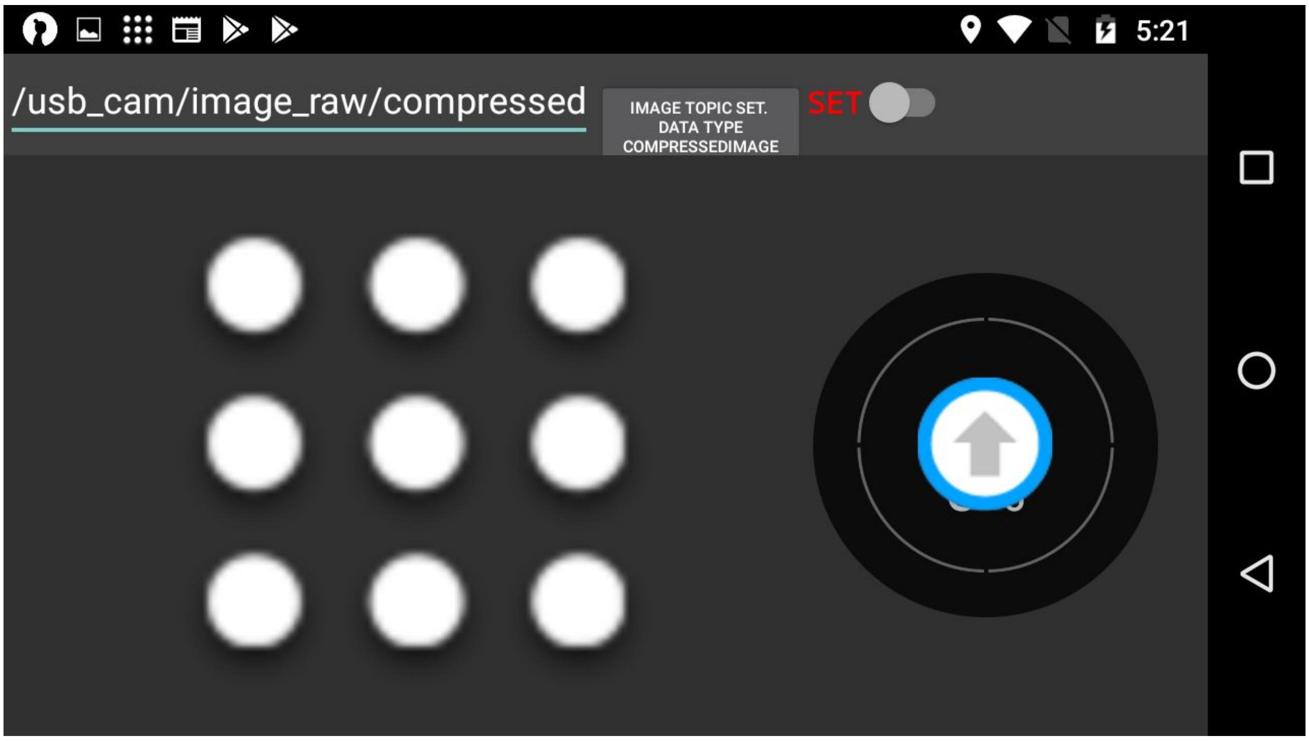


Master URI: http://192.168.43.152:11311/

☐ Show advanced options

(vii). Click on "connect" on mobile application.

(viii). Now, type the compressed image topic in the place of the upper left corner in the mobile screen.



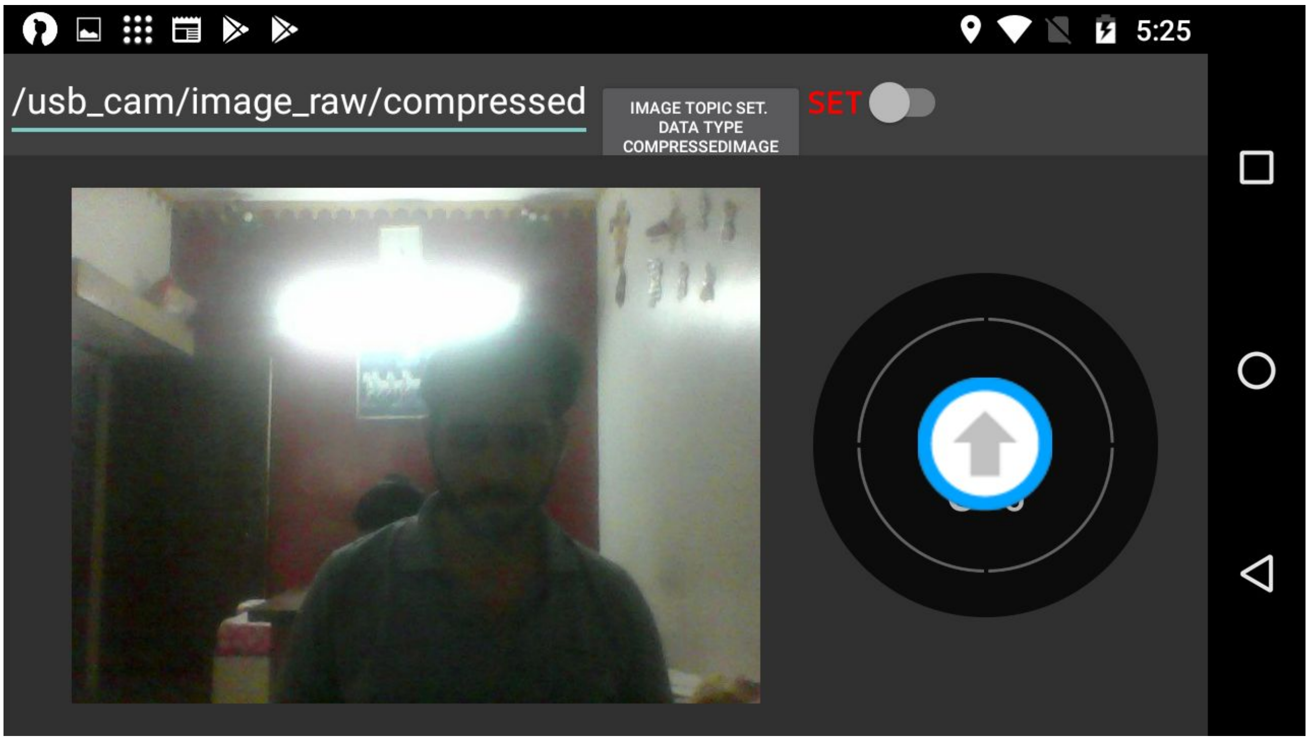
/usb_cam/image_raw/compressed

IMAGE TOPIC SET.
DATA TYPE
COMPRESSEDIMAGE

Name of the topic: /usb_cam/image_raw/compressed

(ix) Now, click on the button "IMAGE TOPIC SET DATA TYPE COMPRESSED IMAGE"next to the typed topic.

(x) Now, you will be able to see the live stream captured by the webcam on the screen.


5:25
/usb_cam/image_raw/compressed
IMAGE TOPIC SET. DATA TYPE COMPRESSEDIMAGE
SET

2. **Using video_stream_opencv**

(i). First of all install video_stream_opencv by typing following command.

```
sudo apt install ros-melodic-video-stream-opencv
```

(ii). Download and install the app ROS Teleop Controller AuTURBO from playstore from this link:
https://play.google.com/store/apps/details?id=org.ros.android.controllerSample&hl=en_us

(iii). Connect both PC and phone with the same wifi network.

(iv). Run the following commands in your PC's terminal. IP can be checked using *ifconfig* command.

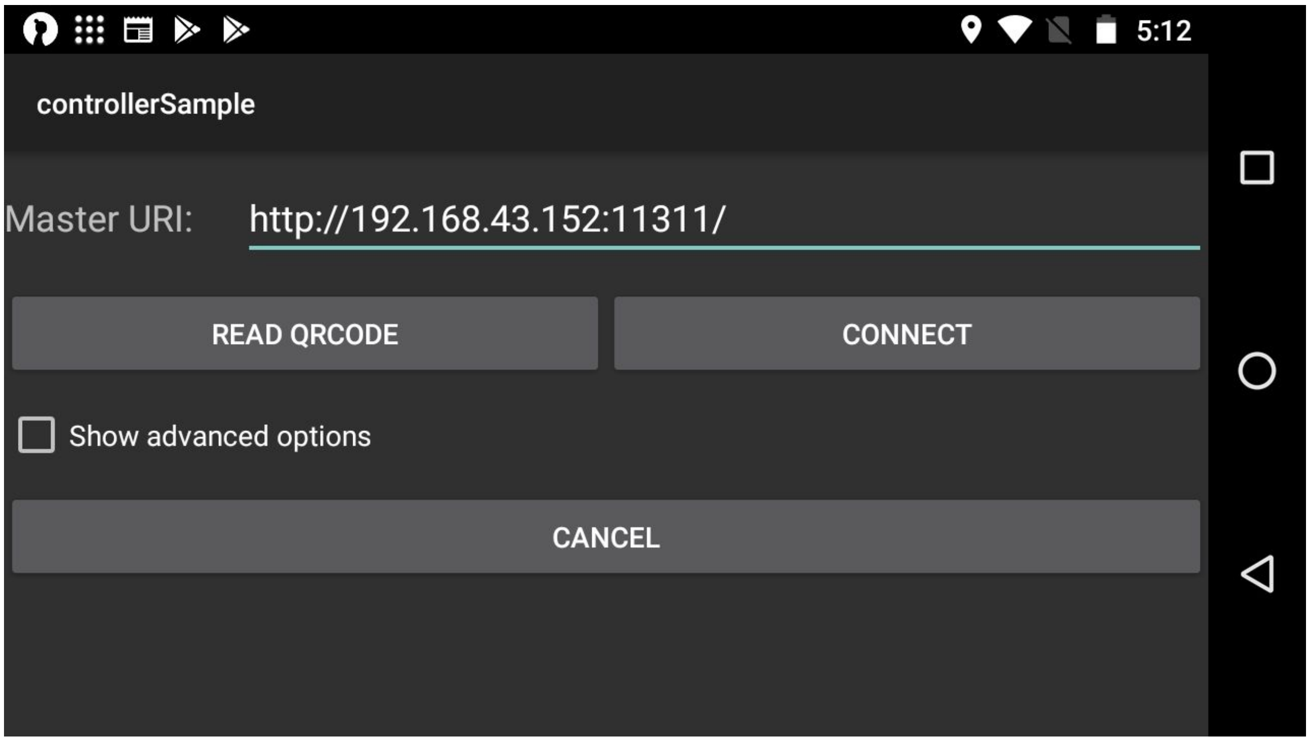
```
export ROS_IP=IPofPC
roscore
```

(v). In another terminal, run these commands:

```
export ROS_IP=IPofPC
roslaunch video_stream_opencv webcam.launch
```

(vi) Open the mobile application and put the Master URI according to the PC.

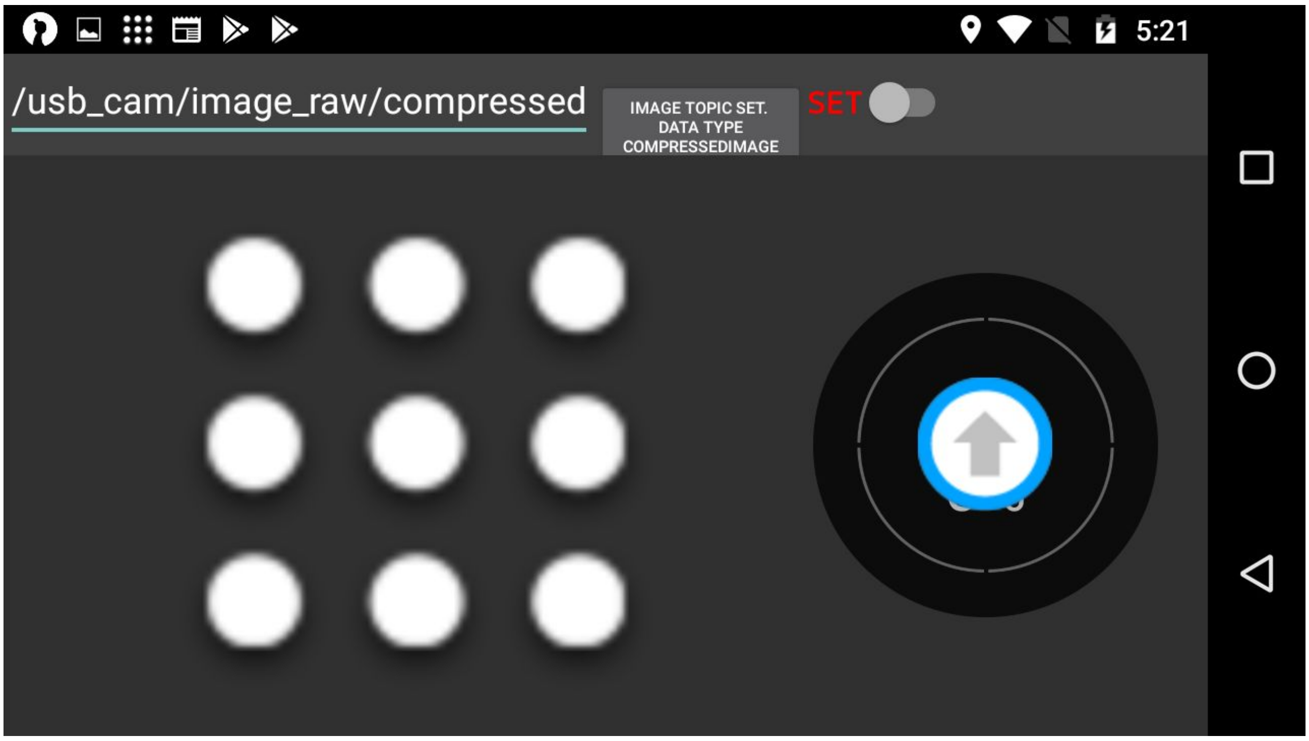
Master URI: http://**IPofPC**:11311/



Master URI: http://192.168.43.152:11311/

(vii). Click on “connect” on mobile application.

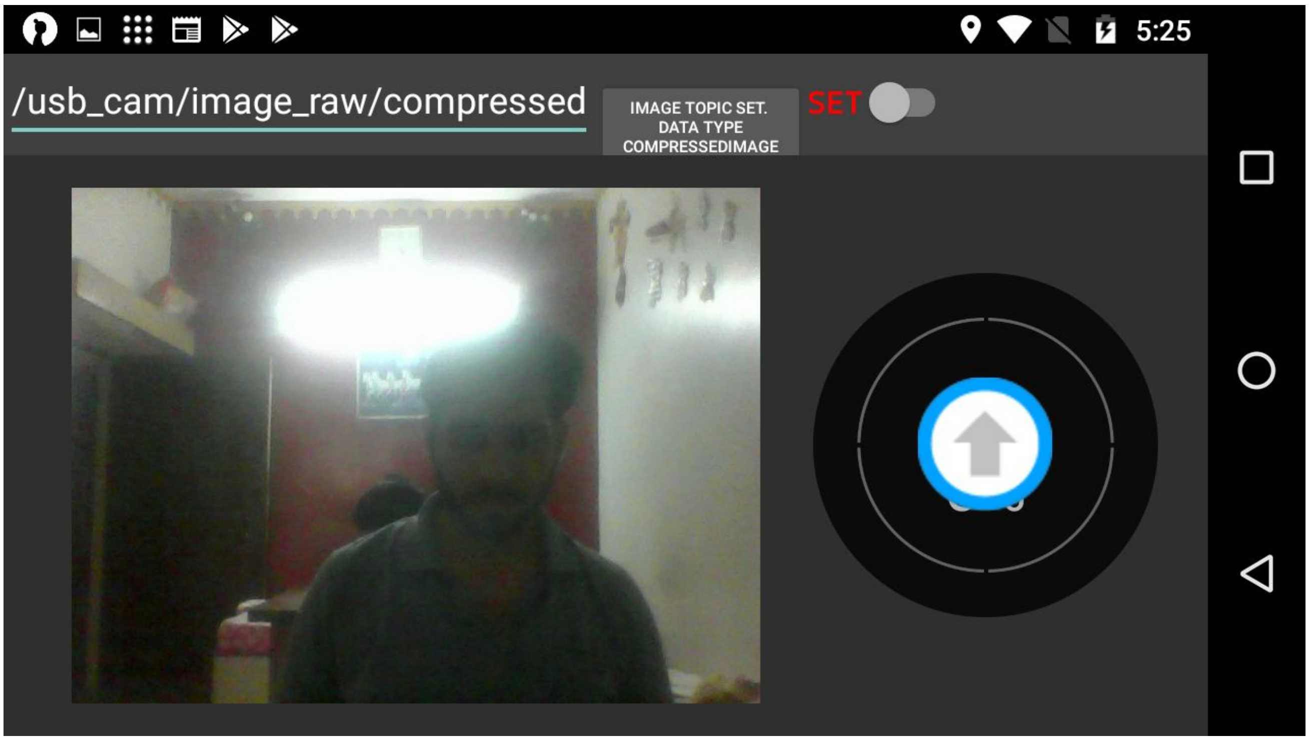
(viii). Now, type the compressed image topic in the place of the upper left corner in the mobile screen.


5:21
/usb_cam/image_raw/compressed
IMAGE TOPIC SET. DATA TYPE COMPRESSEDIMAGE
SET

Name of the topic: /webcam/image_raw/compressed

(ix) Now, click on the button "IMAGE TOPIC SET DATA TYPE COMPRESSED IMAGE"next to the typed topic.

(x) Now, you will be able to see the live stream captured by the webcam on the smartphone screen.


5:25
/usb_cam/image_raw/compressed
IMAGE TOPIC SET. DATA TYPE COMPRESSEDIMAGE
SET

This document is reviewed up to previous section

Annexure 1: Dual Booting PC with Ubuntu 18.04

Sometimes, virtual machines perform slowly so it becomes difficult to get our work done through the computer. Sometimes, we face incompatibility between hardware and software. 8jtSo, dual-boot is a good option for installing Ubuntu.

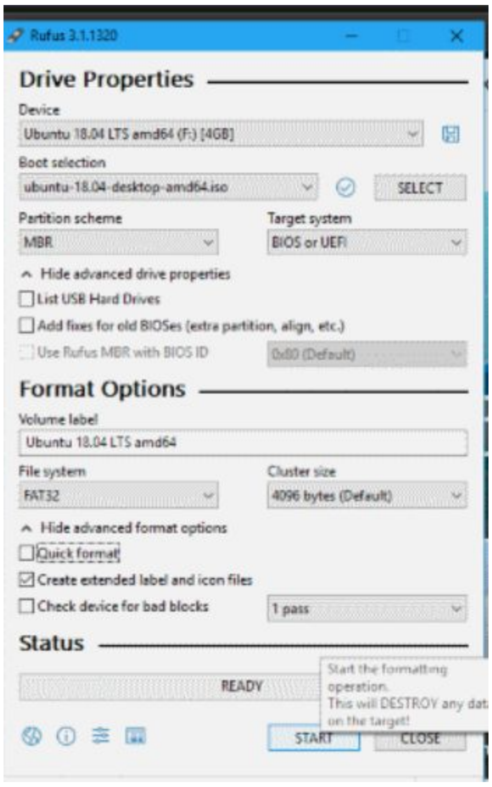
Note: this procedure is for the Windows PCs the ubuntu installation procedure is same for other Linux PCs, not for MAC

A1.1. Making Bootable pendrive of Ubuntu.

a) Download Rufus, which is most suited for your PC from the link: https://rufus.akeo.ie/.

b) Insert the USB Device (minimum capacity 4gb and empty) and start the application as **Administrator.**

c) Choose the device that you prefer from the drop-down menu under **Devices** (fig. 1.4).



Rufus 3.1.1320

Drive Properties

Device: Ubuntu 18.04 LTS amd64 (F:) [4GB]

Boot selection: ubuntu-18.04-desktop-amd64.iso

Partition scheme: MBR

Target system: BIOS or UEFI

Hide advanced drive properties

☐ List USB Hard Drives

☐ Add fixes for old BIOSes (extra partition, align, etc.)

☐ Use Rufus MBR with BIOS ID: 0x80 (Default)

Format Options

Volume label: Ubuntu 18.04 LTS amd64

File system: FAT32

Cluster size: 4096 bytes (Default)

Hide advanced format options

☐ Quick format

☑ Create extended label and icon files

☐ Check device for bad blocks: 1 pass

Status

READY

Start the formatting operation. This will DESTROY any dat on the target!

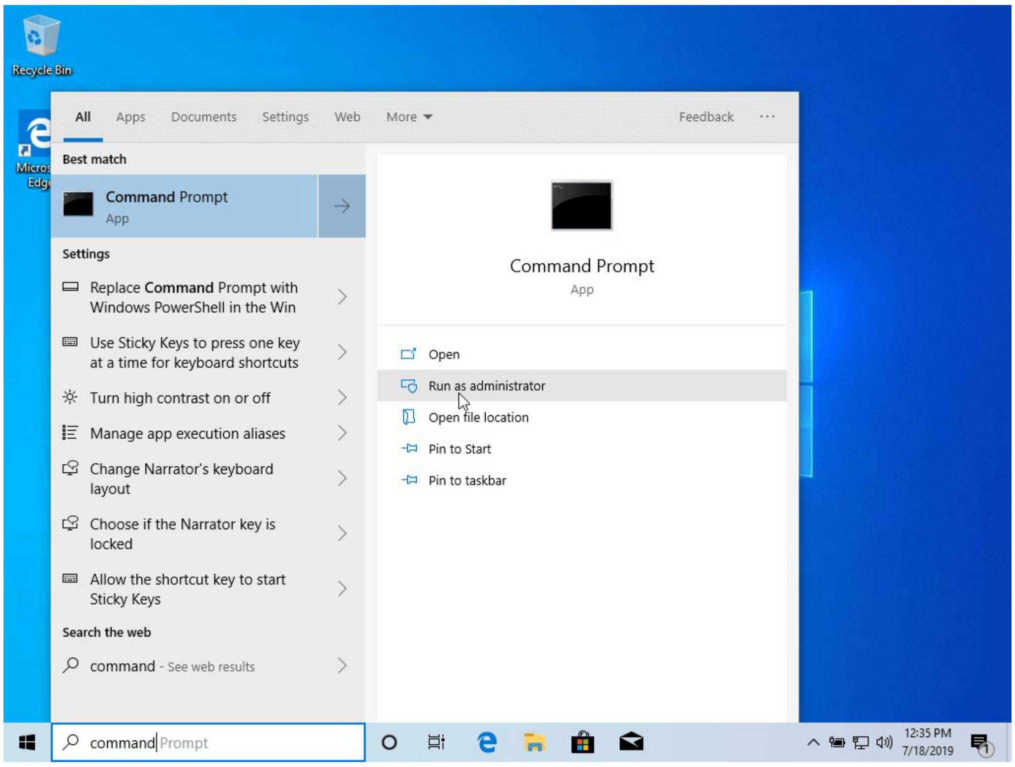
d) Next, browse to the Ubuntu 18.04 file as ISO, this can be done under **Boot selection**.

e) Then, Click on **Start** and in the subsequent window popup, click on **OK** to confirm erasing the content and format the drive. Probably, the process will take less than a few minutes.

Finally, when everything is complete, you will have your Boot device ready.

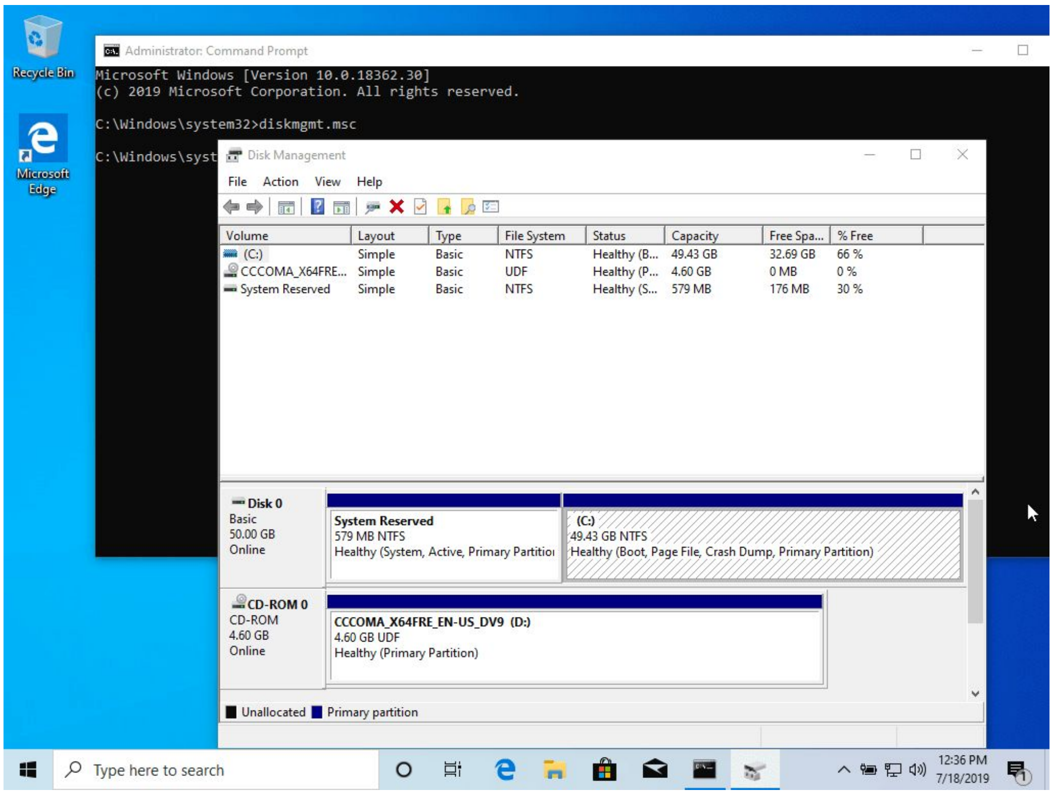
A1.2. Create free space for Ubuntu on Windows PC

1. The first thing you need to take care of is to create a free space on the computer hard disk in case the system is installed on a single partition.
2. Login to your Windows machine with an administrative account and right click on the Start Menu -> Command Prompt (Admin) in order to enter Windows Command Line.


Recycle Bin
All
Apps
Documents
Settings
Web
More
Feedback
Best match
Command Prompt
App
Settings
Replace Command Prompt with Windows PowerShell in the Win
Use Sticky Keys to press one key at a time for keyboard shortcuts
Turn high contrast on or off
Manage app execution aliases
Change Narrator's keyboard layout
Choose if the Narrator key is locked
Allow the shortcut key to start Sticky Keys
Search the web
command - See web results
Command Prompt
App
Open
Run as administrator
Open file location
Pin to Start
Pin to taskbar
command Prompt
12:35 PM
7/18/2019

3. Type diskmgmt.msc on prompt and the Disk Management utility should open. From here, right click on C: partition (for example) and select Shrink Volume in order to resize the partition.

   C:\Windows\system32\>diskmgmt.msc



```
Microsoft Windows [Version 10.0.18362.30]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>diskmgmt.msc

C:\Windows\syst
```

Volume	Layout	Type	File System	Status	Capacity	Free Spa...	% Free
(C:)	Simple	Basic	NTFS	Healthy (B...	49.43 GB	32.69 GB	66 %
CCCOMA_X64FRE...	Simple	Basic	UDF	Healthy (P...	4.60 GB	0 MB	0 %
System Reserved	Simple	Basic	NTFS	Healthy (S...	579 MB	176 MB	30 %

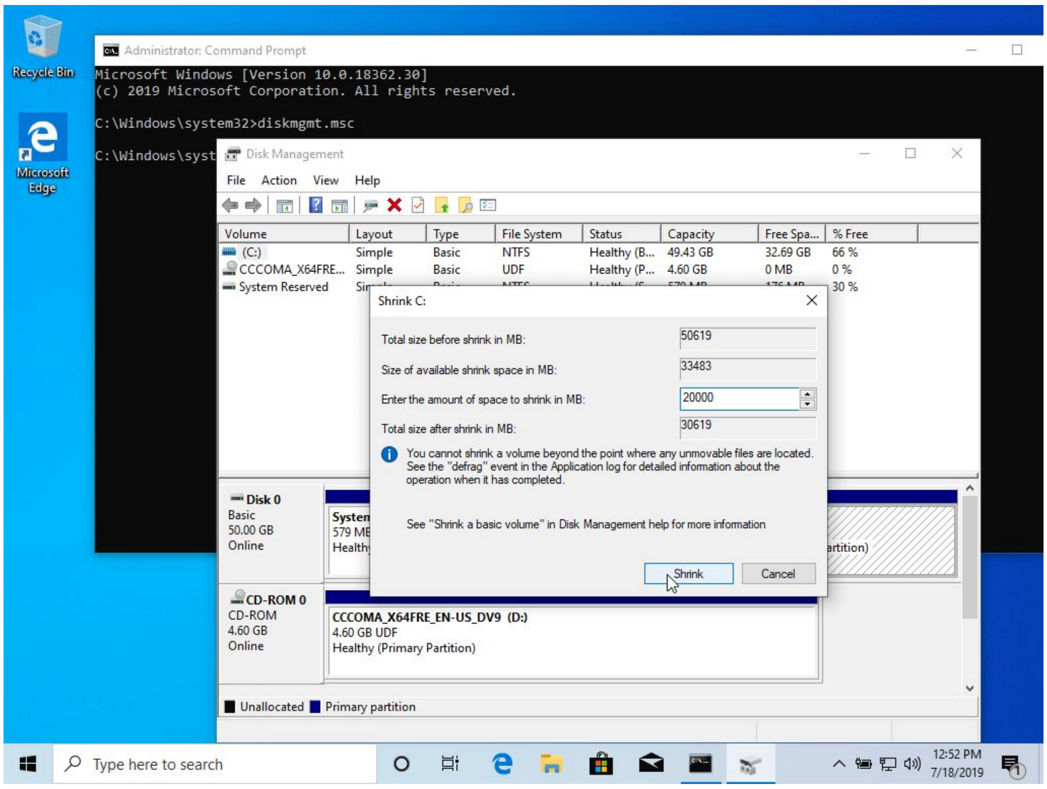
Disk 0
Basic
50.00 GB
Online

System Reserved
579 MB NTFS
Healthy (System, Active, Primary Partitio

(C:)
49.43 GB NTFS
Healthy (Boot, Page File, Crash Dump, Primary Partition)

CD-ROM 0
CD-ROM
4.60 GB
Online

CCCOMA_X64FRE_EN-US_DV9 (D:)
4.60 GB UDF
Healthy (Primary Partition)



```
Microsoft Windows [Version 10.0.18362.30]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>diskmgmt.msc

C:\Windows\syst
```

Volume	Layout	Type	File System	Status	Capacity	Free Spa...	% Free
(C:)	Simple	Basic	NTFS	Healthy (B...	49.43 GB	32.69 GB	66 %
CCCOMA_X64FRE...	Simple	Basic	UDF	Healthy (P...	4.60 GB	0 MB	0 %
System Reserved	Si						30 %

Shrink C:

Total size before shrink in MB:	50619
Size of available shrink space in MB:	33483
Enter the amount of space to shrink in MB:	20000
Total size after shrink in MB:	30619

You cannot shrink a volume beyond the point where any unmovable files are located. See the "defrag" event in the Application log for detailed information about the operation when it has completed.

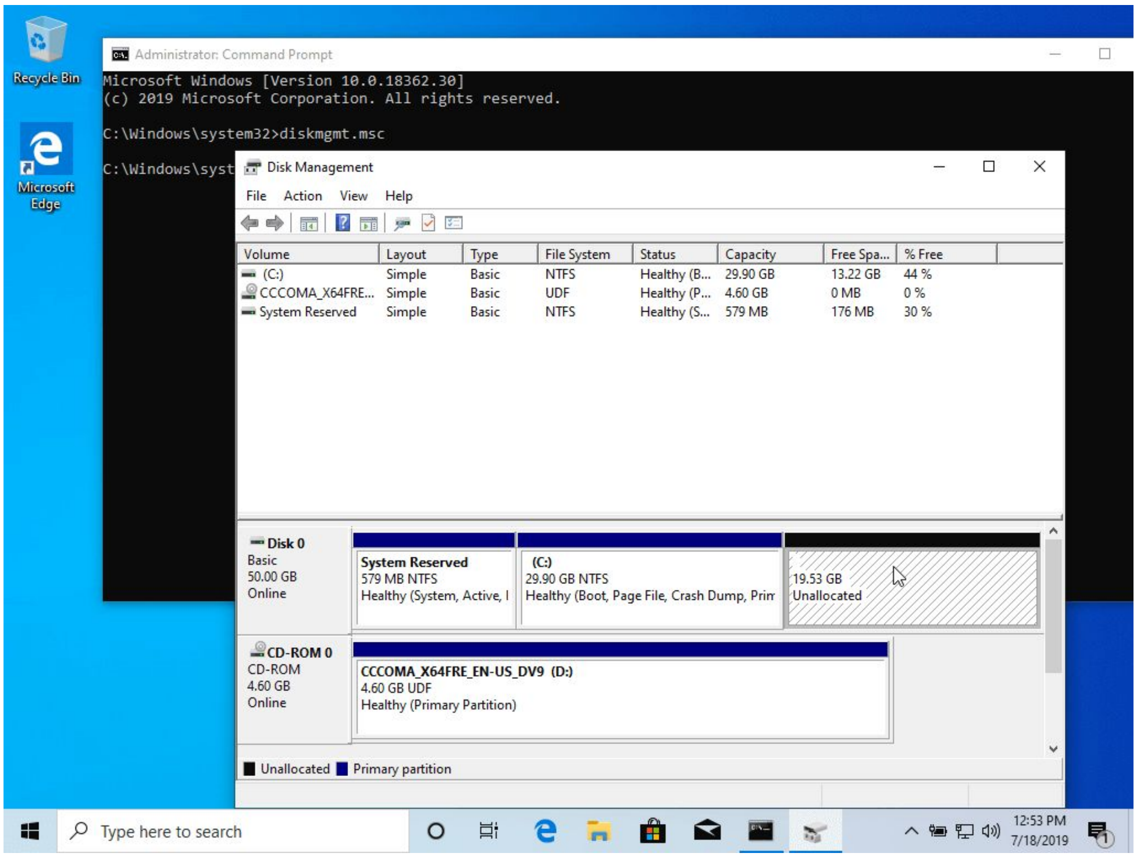
See "Shrink a basic volume" in Disk Management help for more information

Disk 0
Basic
50.00 GB
Online

CD-ROM 0
CD-ROM
4.60 GB
Online

CCCOMA_X64FRE_EN-US_DV9 (D:)
4.60 GB UDF
Healthy (Primary Partition)

■ Unallocated ■ Primary partition



```
Microsoft Windows [Version 10.0.18362.30]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>diskmgmt.msc

C:\Windows\syst
```

Volume	Layout	Type	File System	Status	Capacity	Free Spa...	% Free
(C:)	Simple	Basic	NTFS	Healthy (B...	29.90 GB	13.22 GB	44 %
CCCOMA_X64FRE...	Simple	Basic	UDF	Healthy (P...	4.60 GB	0 MB	0 %
System Reserved	Simple	Basic	NTFS	Healthy (S...	579 MB	176 MB	30 %

Disk 0
Basic
50.00 GB
Online

System Reserved
579 MB NTFS
Healthy (System, Active, I

(C:)
29.90 GB NTFS
Healthy (Boot, Page File, Crash Dump, Prir

19.53 GB
Unallocated

CD-ROM 0
CD-ROM
4.60 GB
Online

CCCOMA_X64FRE_EN-US_DV9 (D:)
4.60 GB UDF
Healthy (Primary Partition)

■ Unallocated ■ Primary partition

4. On Shrink C: enter a value on space to shrink in MB (use at least 25600 MB depending on the C: partition size) and hit Shrink to start partition resize as illustrated below.

Once the space has been resized you will see a new unallocated space on the hard drive. Leave it as default and reboot the computer in order to proceed with Ubuntu installation.

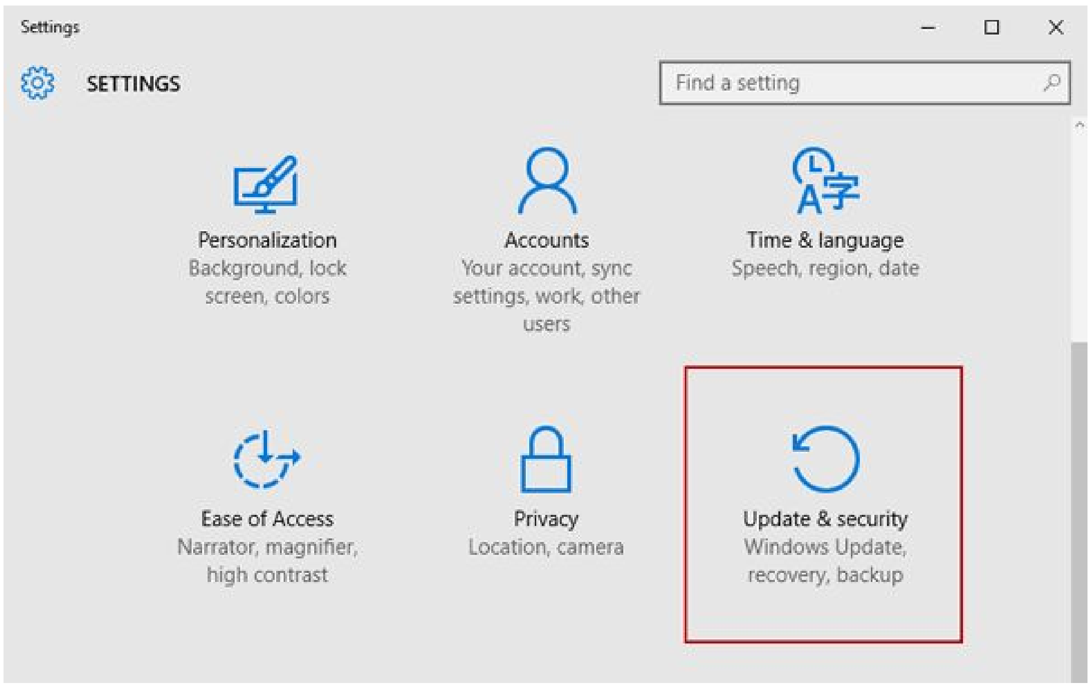
A1.3. Disable secure boot in Windows 10

This is the most important step. The new **secure boot** feature of Windows 8, originally intended for security features for rootkit viruses, prevents dual booting of Windows with Linux. To dual boot Windows 8 with Linux, we must disable secure boot in UEFI.

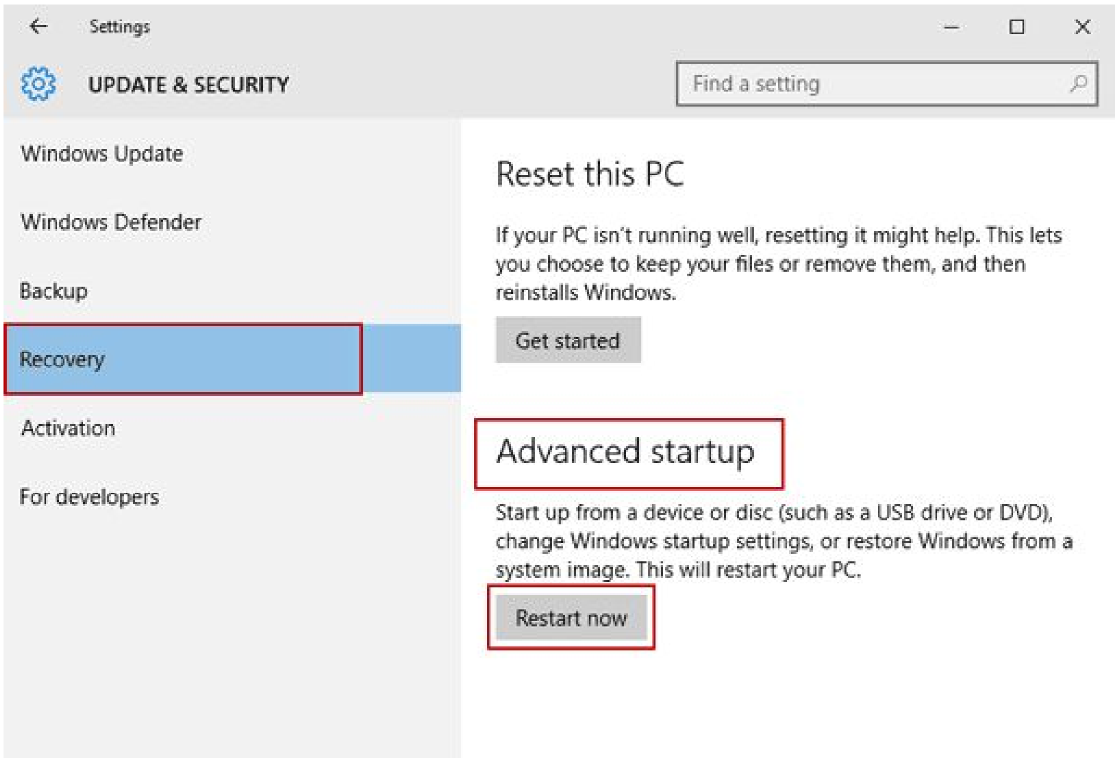
For **Windows 8 & 8.1,** click here

For **Windows 10**,

1. In PC settings, go to **Update & Security**


Settings
SETTINGS
Find a setting
Personalization
Background, lock screen, colors
Accounts
Your account, sync settings, work, other users
Time & language
Speech, region, date
Ease of Access
Narrator, magnifier, high contrast
Privacy
Location, camera
Update & security
Windows Update, recovery, backup

2. Next, select Recovery from the left menu and you can see Advanced startup at the right side.



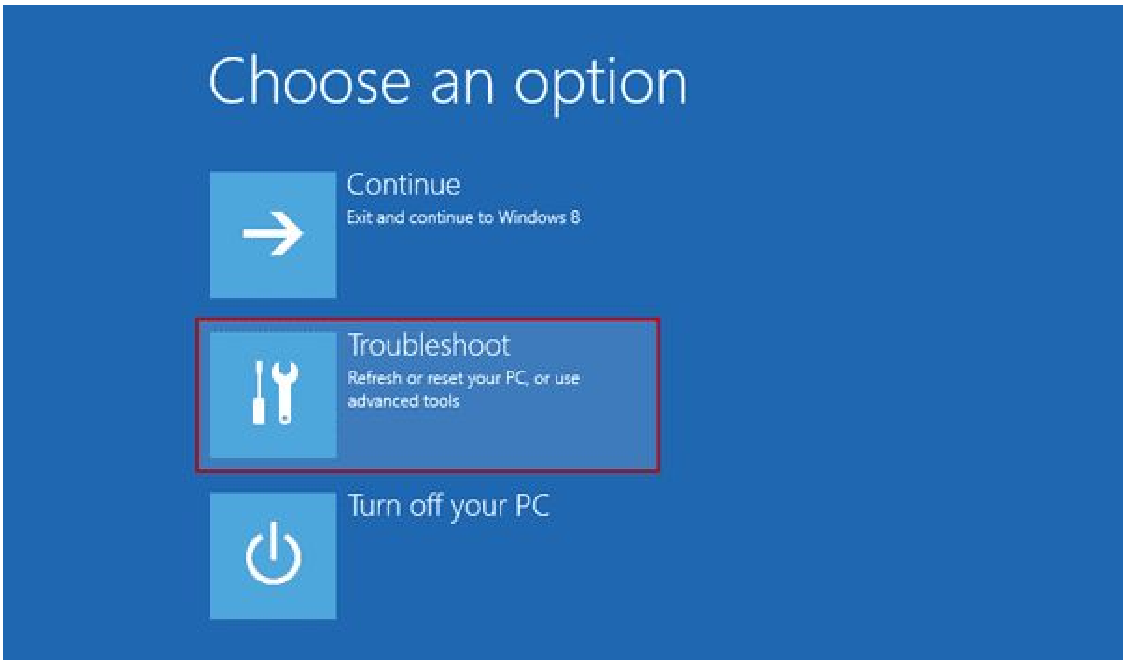
Reset this PC

If your PC isn't running well, resetting it might help. This lets you choose to keep your files or remove them, and then reinstalls Windows.

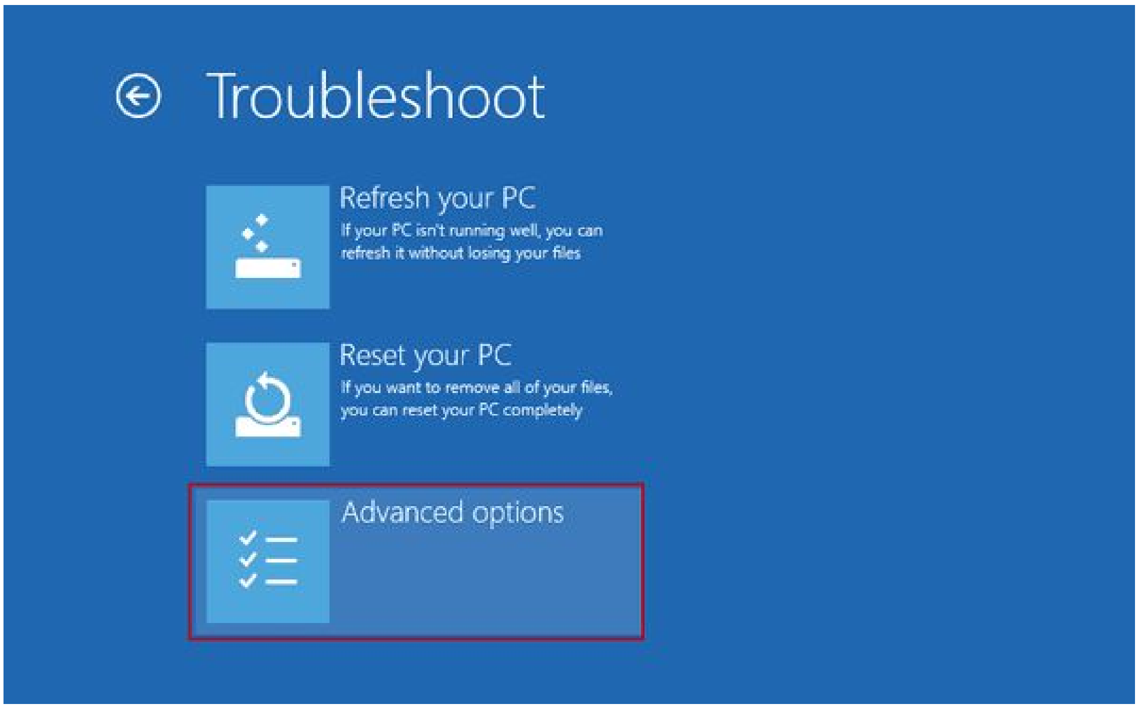
Advanced startup

Start up from a device or disc (such as a USB drive or DVD), change Windows startup settings, or restore Windows from a system image. This will restart your PC.

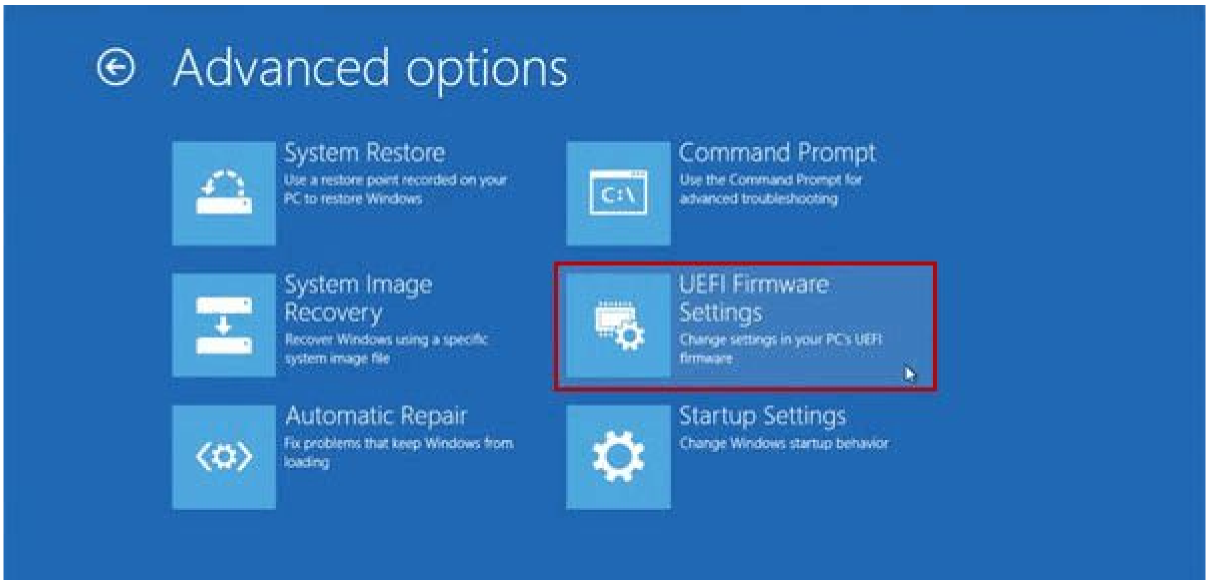
3. Click Restart Now under Advanced startup option. The comptuer will reboot to a special menu.
4. Now you will be presented with the Choose an option screen, simply select Troubleshoot.


Choose an option
Continue
Exit and continue to Windows 8
Troubleshoot
Refresh or reset your PC, or use advanced tools
Turn off your PC

5. Next select Advanced options.


Troubleshoot
Refresh your PC
If your PC isn't running well, you can refresh it without losing your files
Reset your PC
If you want to remove all of your files, you can reset your PC completely
Advanced options

6. Next you select UEFI Firmware Settings.


Advanced options
System Restore
Use a restore point recorded on your PC to restore Windows
Command Prompt
Use the Command Prompt for advanced troubleshooting
System Image Recovery
Recover Windows using a specific system image file
UEFI Firmware Settings
Change settings in your PC's UEFI firmware
Automatic Repair
Fix problems that keep Windows from loading
Startup Settings
Change Windows startup behavior

7. Click the Restart button. Your system will restart and take you to the UEFI BIOS.

Note: Disable secure boot according to your computer. (different for different computers)

A1.4. Disable Fast Startup in Windows, So Linux can work smoothly.

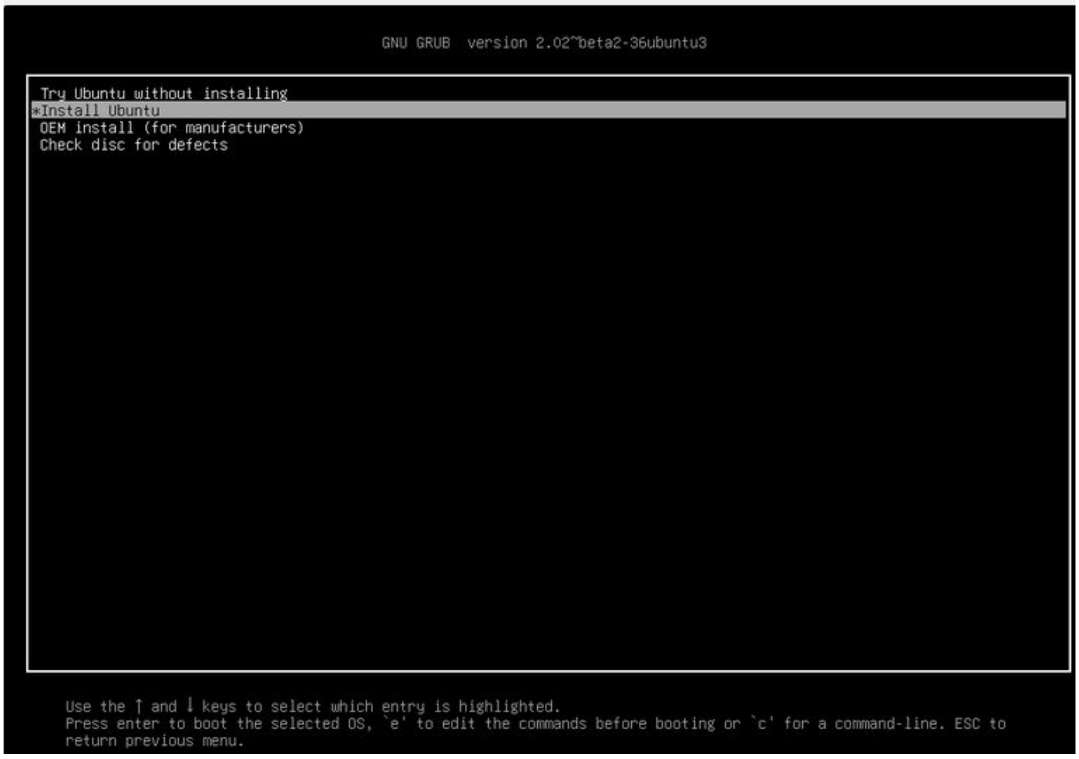
1. Press the Windows key on your keyboard, type in Power Options, and then press Enter.
2. From the left menu, select Choose what the power buttons do.

3. Under the Shutdown settings section, uncheck the box next to Turn on fast startup (recommended).
4. Click the Save changes button.

A1.5. Ubuntu Installation

a) Place the bootable USB device in the appropriate drive, reboot the machine and press the special function key (usually F12, F10 or F2 depending on the vendor specifications) instruct the BIOS/UEFI to boot-up from the DVD/USB. You can use this tutorial: https://youtu.be/u5QyjHIYwTQ.

b) Once the media boot-up occurs a new grub screen should appear on your monitor. From the menu select **Install Ubuntu** and press **Enter** to continue (fig.1.5).



```
GNU GRUB  version 2.02~beta2-36ubuntu3

 Try Ubuntu without installing
*Install Ubuntu
 OEM install (for manufacturers)
 Check disc for defects

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, `e' to edit the commands before booting or `c' for a command-line. ESC to
return previous menu.
```

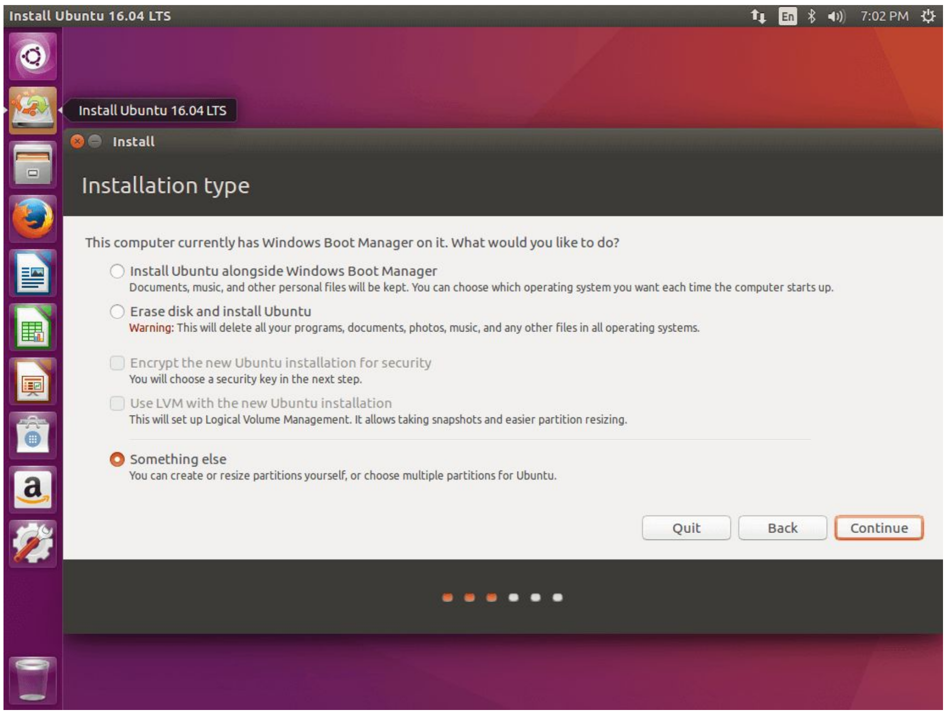
c) After the boot media finishes loading into RAM you will have a completely functional Ubuntu system running in live mode. On the launcher click on the second icon from top, **Install Ubuntu 18.04 LTS**, and the installer utility will start. Choose the language you wish to perform the installation and click on the **Continue** button to proceed further.

d) Next, leave both options from **Preparing to Install Ubuntu** unchecked and hit on the **Continue** button again.

e) Now it's time to select an Installation Type. You can choose to **Install Ubuntu** alongside **Windows Boot Manager**, which will automatically take care of all the partition steps. As shown in fig. 1.6.

Note:- **<u>The option Erase disk and install Ubuntu should be avoided on dual-boot because is hazardous and will wipe out your disk.</u>**

f) Choose something else if you want to do partitions according to your needs.



Installation type

This computer currently has Windows Boot Manager on it. What would you like to do?

( ) Install Ubuntu alongside Windows Boot Manager
Documents, music, and other personal files will be kept. You can choose which operating system you want each time the computer starts up.

( ) Erase disk and install Ubuntu
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.

[ ] Encrypt the new Ubuntu installation for security
You will choose a security key in the next step.

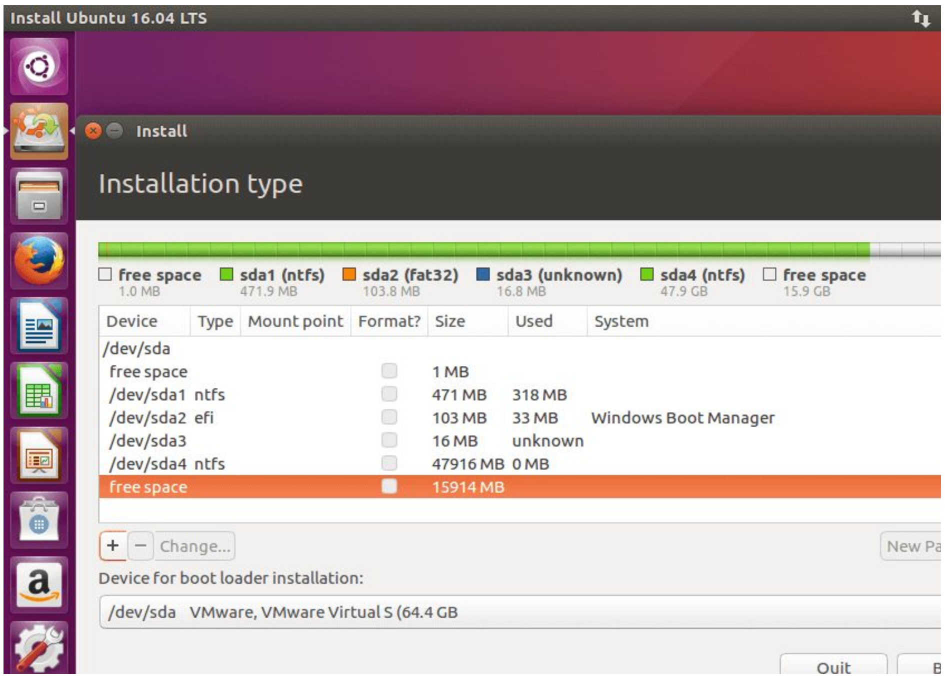
[ ] Use LVM with the new Ubuntu installation
This will set up Logical Volume Management. It allows taking snapshots and easier partition resizing.

[✓ Something else]
You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.

g) On this step, we'll create our custom partition layout for **Ubuntu 18.04** as in fig. 1.7.

h) In order to create a partition, select free space and click on + icon.

i) Make "/", swap and "/home" partition. Use this tutorial https://www.itzgeek.com/how-tos/linux/ubuntu-how-tos/how-to-install-ubuntu-18-04-alongside-with-windows-10-or-8-in-dual-boot.html


Install Ubuntu 16.04 LTS

Install

Installation type

☐ free space 1.0 MB · 🟩 sda1 (ntfs) 471.9 MB · 🟧 sda2 (fat32) 103.8 MB · 🟦 sda3 (unknown) 16.8 MB · 🟩 sda4 (ntfs) 47.9 GB · ☐ free space 15.9 GB

Device	Type	Mount point	Format?	Size	Used	System
/dev/sda						
free space			☐	1 MB		
/dev/sda1	ntfs		☐	471 MB	318 MB	
/dev/sda2	efi		☐	103 MB	33 MB	Windows Boot Manager
/dev/sda3			☐	16 MB	unknown	
/dev/sda4	ntfs		☐	47916 MB	0 MB	
free space			☐	15914 MB		

Device for boot loader installation:

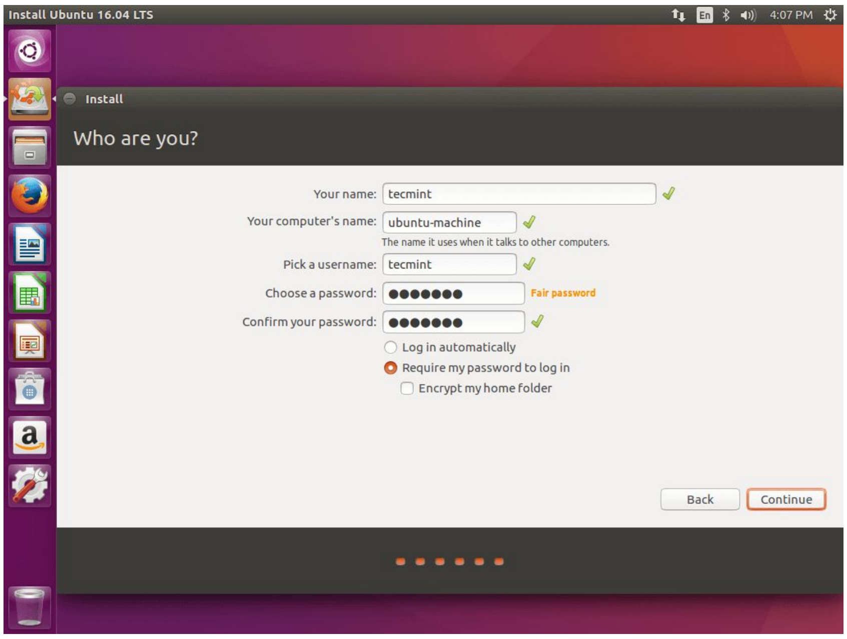
/dev/sda VMware, VMware Virtual S (64.4 GB

j) When finished creating the partition, hit the **Install Now** button in order to apply changes to the disk and start the installation process. A pop-up window will ask you if you agree with committing changes to disk. Hit **Continue** to write changes to disk and the installation process will now start.

k) On the next screen adjust your machine's physical location by selecting a city nearby from the map. When done hit **Continue** to move ahead.

l) Next, select your **keyboard** layout and click the **Continue** button.

m) Pick a username and password as shown in fig. 1.8 for your administrative **sudo** account, enter a descriptive name for your computer and hit **Continue** to finalize the installation.


Install Ubuntu 16.04 LTS

Install

Who are you?

Your name: tecmint ✓

Your computer's name: ubuntu-machine ✓

The name it uses when it talks to other computers.

Pick a username: tecmint ✓

Choose a password: ●●●●●●● Fair password

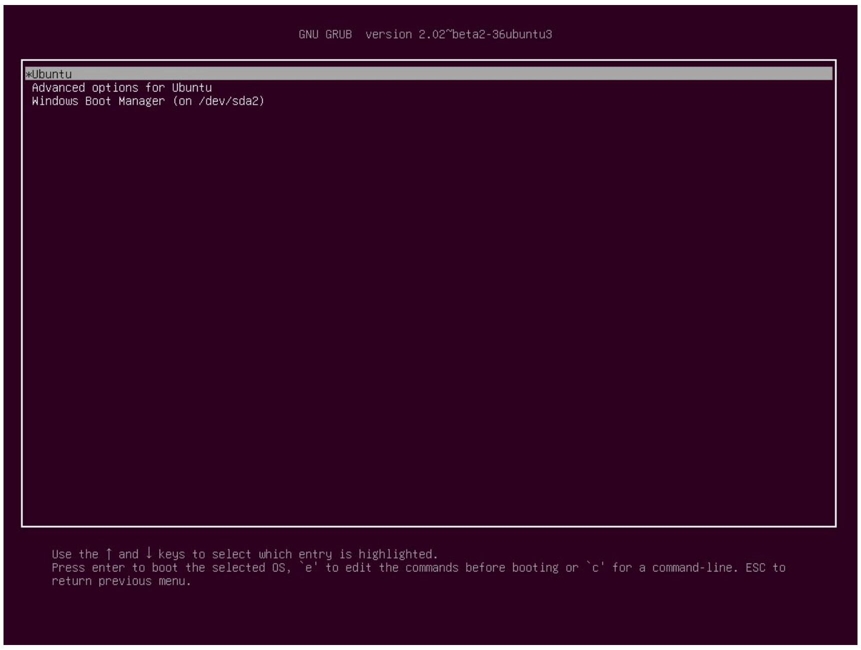
Confirm your password: ●●●●●●● ✓

○ Log in automatically

● Require my password to log in

☐ Encrypt my home folder

n) After the installation process reaches its end hit on the **Restart Now** button in order to complete the installation. The machine will reboot into the **Grub** menu, where for ten seconds, you will be presented to choose what OS you wish to use further: **Ubuntu 18.04** or **Microsoft Windows** (fig. 1.8).


```
GNU GRUB  version 2.02~beta2-36ubuntu3

*Ubuntu
 Advanced options for Ubuntu
 Windows Boot Manager (on /dev/sda2)

   Use the ↑ and ↓ keys to select which entry is highlighted.
   Press enter to boot the selected OS, `e' to edit the commands before booting or `c' for a command-line. ESC to
   return previous menu.
```

Annexure 2: ROS installation

A2.1 Introduction

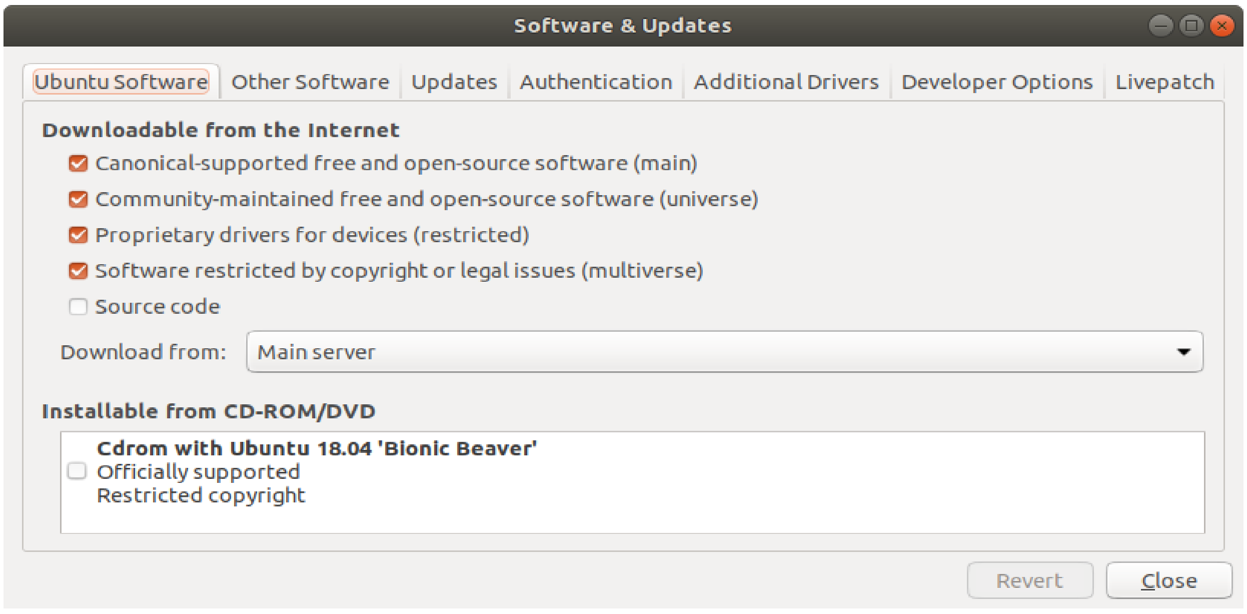
ROS(robot operating system) is an open-source, meta-operating system for your robot. It provides the services that a normal operating system does. It provides services such as hardware-abstraction, low-level device control, message passing between the processes, and package management. Currently, ROS runs only on Unix-based platforms. Software for ROS is primarily tested on Ubuntu and Mac OS X systems. A port to Microsoft Windows for ROS is possible, it hasn't been explored fully yet.

A2.2 Configure your Ubuntu repositories

Configure your Ubuntu repositories to allow "restricted," "universe," and "multiverse."

Step a: go to "software and updates" search it on ubuntu.

Step b: set all settings as shown fig. 3.11.


Software & Updates

Downloadable from the Internet

- [✓] Canonical-supported free and open-source software (main)
- [✓] Community-maintained free and open-source software (universe)
- [✓] Proprietary drivers for devices (restricted)
- [✓] Software restricted by copyright or legal issues (multiverse)
- [ ] Source code

Download from: Main server

Installable from CD-ROM/DVD

- [ ] **Cdrom with Ubuntu 18.04 'Bionic Beaver'**
  Officially supported
  Restricted copyright

A2.3 Setup your sources.list

Open a new terminal (Ctrl+Alt+t) and type the following command in it.

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

A2.4 Set up your keys

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

If above command don't work then install curl by

```
sudo apt install curl
```

After installing curl, run this

```
curl -sSL
'http://keyserver.ubuntu.com/pks/lookup?op=get&search=0xC1CF6E31E6BADE8868B1
72B4F42ED6FBAB17C654' | sudo apt-key add -
```

A2.5 Installation

Step a: First, make sure your Debian package index is up-to-date by typing the following command.

```
sudo apt update
sudo apt upgrade
```

Step b: Now, install ROS Melodic Desktop full version

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

A2.6 Initialize rosdep

Before you can use ROS, you will need to initialize rosdep. rosdep enables you to easily install system dependencies for a source you want to compile and is required to run some core components in ROS.

```
sudo rosdep init
rosdep update
```

A2.6 Environment setup

It's convenient if the ROS environment variables are automatically added to your bash session every time a new shell is launched:

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

A2.7 Dependencies for building packages

To create and manage your own ROS workspaces, there are various tools and requirements that are distributed separately. For example, rosinstall is a frequently used command-line tool that enables you to easily download many source trees for ROS packages with one command. To install this tool and other dependencies for building ROS packages, run:

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

To see all the dependencies write the following command and press the tab button on the keyboard twice.

```
sudo apt-get install ros-melodic-
```

Testing ROS Installation -

Test the following command to verify the ROS Installation.

```
roscore
```

A2.8 Troubleshooting:

If you face any problems while running p3dx in gazebo do the following steps

Copy the following command into the terminal.

```
sudo apt-get remove gazebo9
```

Enter the following commands in the terminal

```
sudo sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu-stable
`lsb_release -cs` main" > /etc/apt/sources.list.d/gazebo-stable.list'
```

to set up your computer to accept software from packages.org

```
wget http://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add -
```

to setup keys

```
sudo apt update
```

to update Debian database

```
sudo apt install gazebo9
```

Remove it again

```
sudo apt remove gazebo9
```

Then type this

```
sudo apt-get install ros-melodic-desktop-full
```

After that run

```
cd ~/catkin_ws/src/pioneer_gazebo_ros
./run\_pioneer\_gazebo
```

A2.9 Additional ROS packages

Install additional ROS packages which may be useful further, they may be installed in the full desktop installation, by entering the following command in the terminal

```
sudo apt-get install ros-melodic-navigation ros-melodic-slam-gmapping
ros-melodic-ros-control ros-melodic-ros-controllers ros-melodic-rviz
```

```
sudo apt-get install ros-melodic-joy ros-melodic-teleop-twist-joy
ros-melodic-teleop-twist-keyboard ros-melodic-laser-proc ros-melodic-rgbd-launch
ros-melodic-depthimage-to-laserscan ros-melodic-rosserial-arduino
ros-melodic-rosserial-python ros-melodic-rosserial-server ros-melodic-rosserial-client
ros-melodic-rosserial-msgs ros-melodic-amcl ros-melodic-map-server
ros-melodic-move-base ros-melodic-urdf ros-melodic-xacro
ros-melodic-compressed-image-transport ros-melodic-rqt-image-view
ros-melodic-gmapping ros-melodic-navigation ros-melodic-interactive-markers
ros-melodic-slam-gmapping ros-melodic-ros-control ros-melodic-ros-controllers
ros-melodic-rviz
```

A2.10 Additional Tools

Annexure 4: Additional Tools

Install terminator

```
sudo apt install terminator
```

Download Libaria and install it by double clicking on the file.

https://web.archive.org/web/20181103135921/http://robots.mobilerobots.com/ARIA/download/current/libaria_2.9.4+ubuntu16_amd64.deb

or else you can type below commands to perform the same tasks.

```
curl -O
https://web.archive.org/web/20181103135921/http://robots.mobilerobots.com/ARIA/
download/current/libaria_2.9.4+ubuntu16_amd64.deb

sudo dpkg -i libaria_2.9.4+ubuntu16_amd64.deb
```

Installation of ROS drivers for Kinect camera and some additional components.

```
sudo apt install libfreenect-dev
sudo apt install ros-melodic-openni-camera
sudo apt install ros-melodic-openni-launch
sudo apt install ros-melodic-openni-camera-dbgsym
sudo apt install ros-melodic-openni2-camera
```

```
sudo apt-get install ros-melodic-depthimage-to-laserscan
sudo apt-get install ros-melodic-tf
sudo apt-get install ros-melodic-rtabmap-ros
```

Download Virtual studio code for Ubuntu from here.

For the installation, double click on the file and click on install.

Go to extensions at the left hand side in the application.

search “python” and download the python package provided by microsoft.

Search “ROS” and download the ROS package provided by microsoft.

Install Tuttlebot 3

In terminal

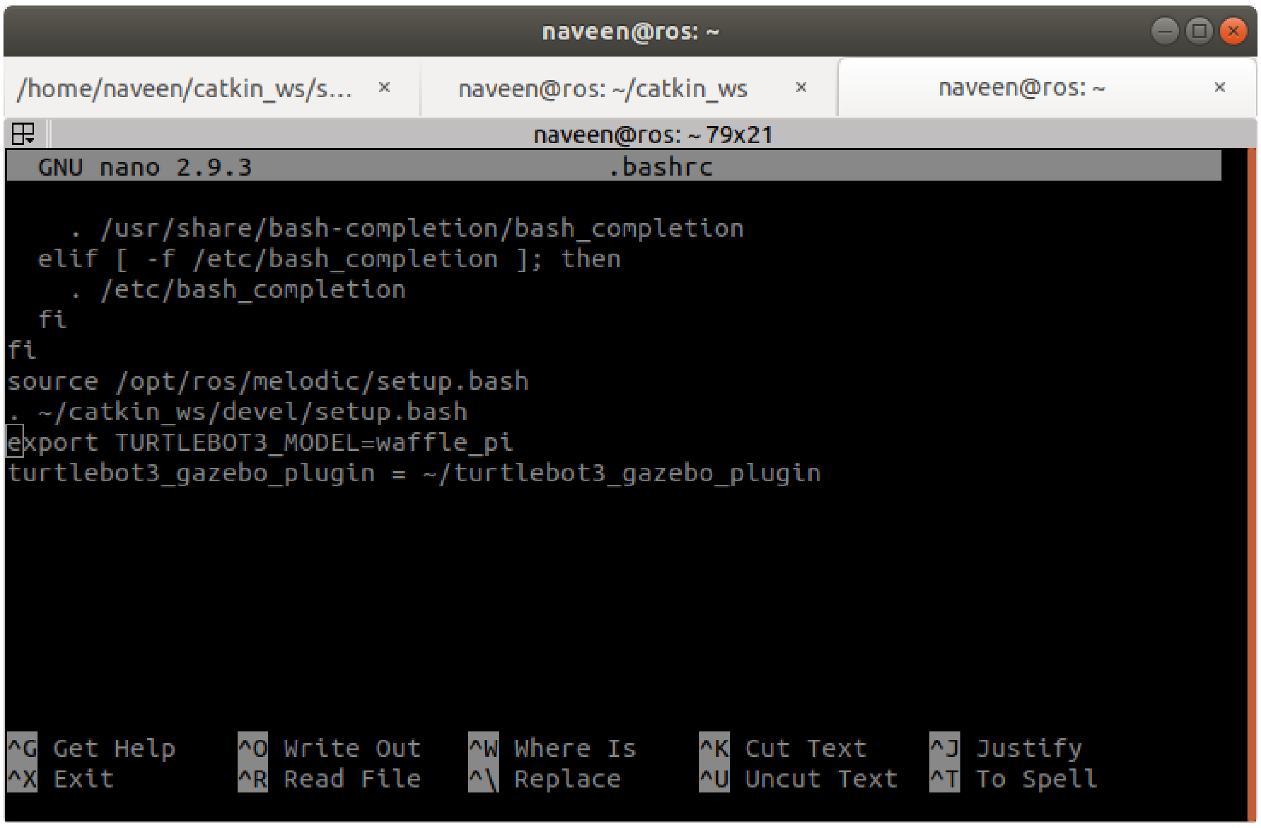
```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src/
git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
cd ~/catkin_ws && catkin_make
```

Open the .bashrc file

```
nano ~/.bashrc
```

Put the following statement at the end of the .bashrc file.

```
. ~/catkin_ws/devel/setup.bash
export TURTLEBOT3_MODEL=waffle_pi
```



```
naveen@ros: ~
/home/naveen/catkin_ws/s...   naveen@ros: ~/catkin_ws   naveen@ros: ~
naveen@ros: ~ 79x21
  GNU nano 2.9.3                    .bashrc

    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
source /opt/ros/melodic/setup.bash
. ~/catkin_ws/devel/setup.bash
export TURTLEBOT3_MODEL=waffle_pi
turtlebot3_gazebo_plugin = ~/turtlebot3_gazebo_plugin

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell
```

After writing this command save the file by pressing ctrl+O and ctrl+X

```
cd ~/catkin_ws/src/
git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
cd ~/catkin_ws && catkin_make
```

Now, turtlebot3 is installed. To check your installation

Run following commands in different terminals to run tb3 in Rviz

```
roslaunch turtlebot3_fake turtlebot3_fake.launch
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

Now you can run the tb3 in Gazebo

```
roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

```
roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

```
roslaunch turtlebot3_gazebo turtlebot3_house.launch
```

NOTE : If TurtleBot3 House is executed for the first time, downloading the map file takes a couple of minutes or more depending on download speed.

Install P3-DX files

```
cd ~/catkin_ws/src
git clone https://github.com/JenJenChung/pioneer_gazebo_ros.git
git clone https://github.com/JenJenChung/pioneer_description.git
git clone https://github.com/naveenkumar2208/pioneer_2dnav.git
git clone https://github.com/JenJenChung/nav_bundle.git
git clone https://github.com/JenJenChung/simple_navigation_goals.git
git clone https://github.com/naveenkumar2208/ua_ros_p3dx.git

cd ~/catkin_ws
catkin_make
```

Check the installation by typing following commands

```
cd ~/catkin_ws/src/pioneer_gazebo_ros
```

```
./run\_pioneer\_gazebo
```

In other window

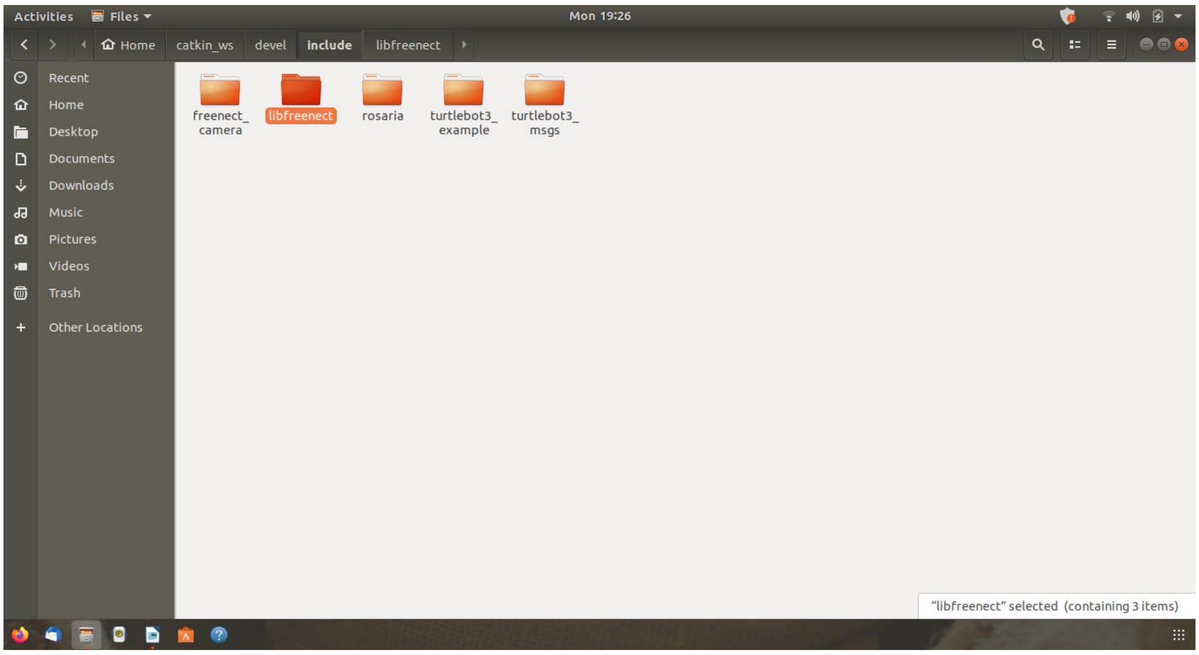
```
roslaunch p3dx_gazebo gazebo.launch
```

```

cd ~/catkin_ws/src
git clone https://github.com/amor-ros-pkg/rosaria.git
cd ..
catkin_make
cd ~/catkin_ws/src
git clone https://github.com/pengtang/rosaria_client.git
cd ~/catkin_ws/
catkin_make
source devel/setup.bash
```

Installing Libfreenect and freenect

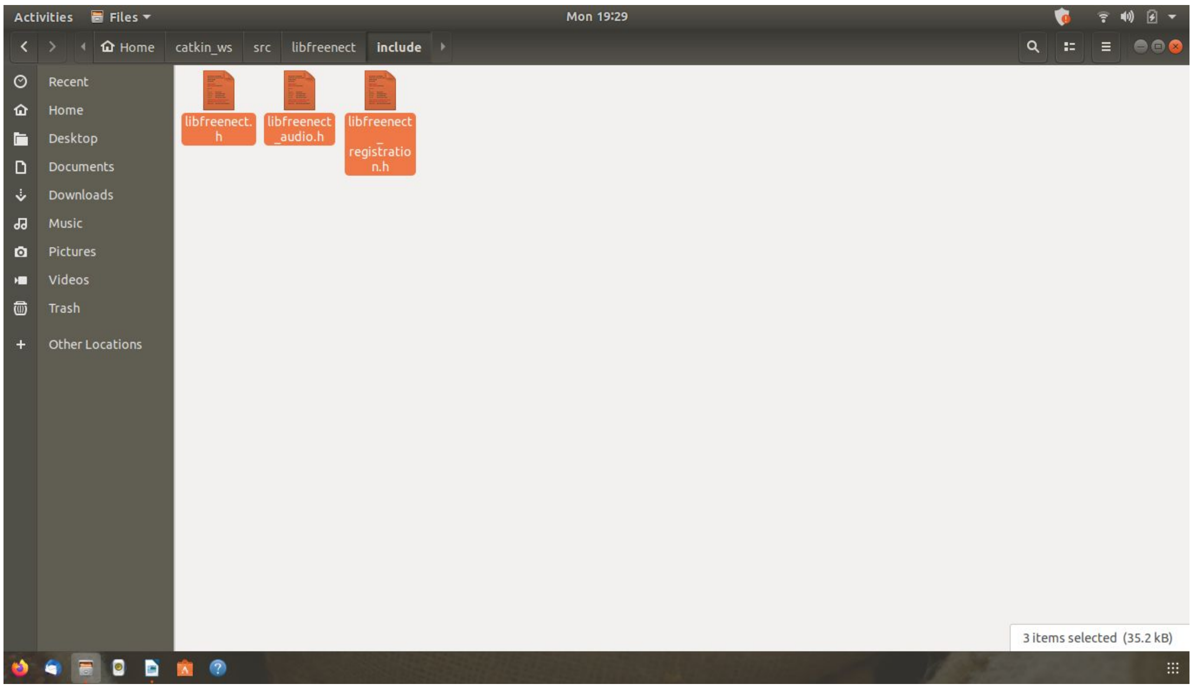
```
cd ~/catkin_ws/src
git clone https://github.com/ros-drivers/libfreenect.git
git clone https://github.com/ros-drivers/freenect_stack.git
```



Go to /home/catkin_ws/devel/include

Create a new folder "libfreenect" in it.

Copy "libfreenect.h", "libfreenect_audio.h", "libfreenect_registration.h" from /home/catkin_ws/src/libfreenect/include/ to /home/catkin_ws/devel/include/libfreenect/



```
cd ~/catkin_ws/
```

```
catkin_make
```

```
cd ~/catkin_ws/src
git clone https://github.com/naveenkumar2208/Pioneer3DX-NAV.git
cd ..
catkin_make
```

Useful web references

- https://www.tecmint.com/install-ubuntu-16-04-alongside-with-windows-10-or-8-in-dual-boot/ for ubuntu installation
- https://wiki.ros.org/ROS/tutorials (all ROS tutorials)
- https://gazebosim.org/tutorials (all gazebo tutorials)
- http://wiki.ros.org/kinetic/Installation for ros installation
- https://maythecodebewithu.wordpress.com/2013/07/28/the-hardware-of-kinect-sensor/ for fig. 4.1
- https://medium.com/@tushar0618/install-ubuntu-16-04-lts-on-virtual-box-desktop-version-30dc6f1958d0
- https://www.appgeeker.com/recovery/disable-uefi-secure-boot-in-windows-10.html

- http://wiki.ros.org/catkin/Tutorials/CreatingPackage
- https://www.choitek.com/uploads/5/0/8/4/50842795/ros_kinect.pdf
- http://wiki.ros.org/navigation
- http://wiki.ros.org/gmapping
- http://wiki.ros.org/depthimage_to_laserscan
- http://wiki.ros.org/amcl
- https://github.com/Choitek/mmmros-tutorials/tree/master/mmm_kinect