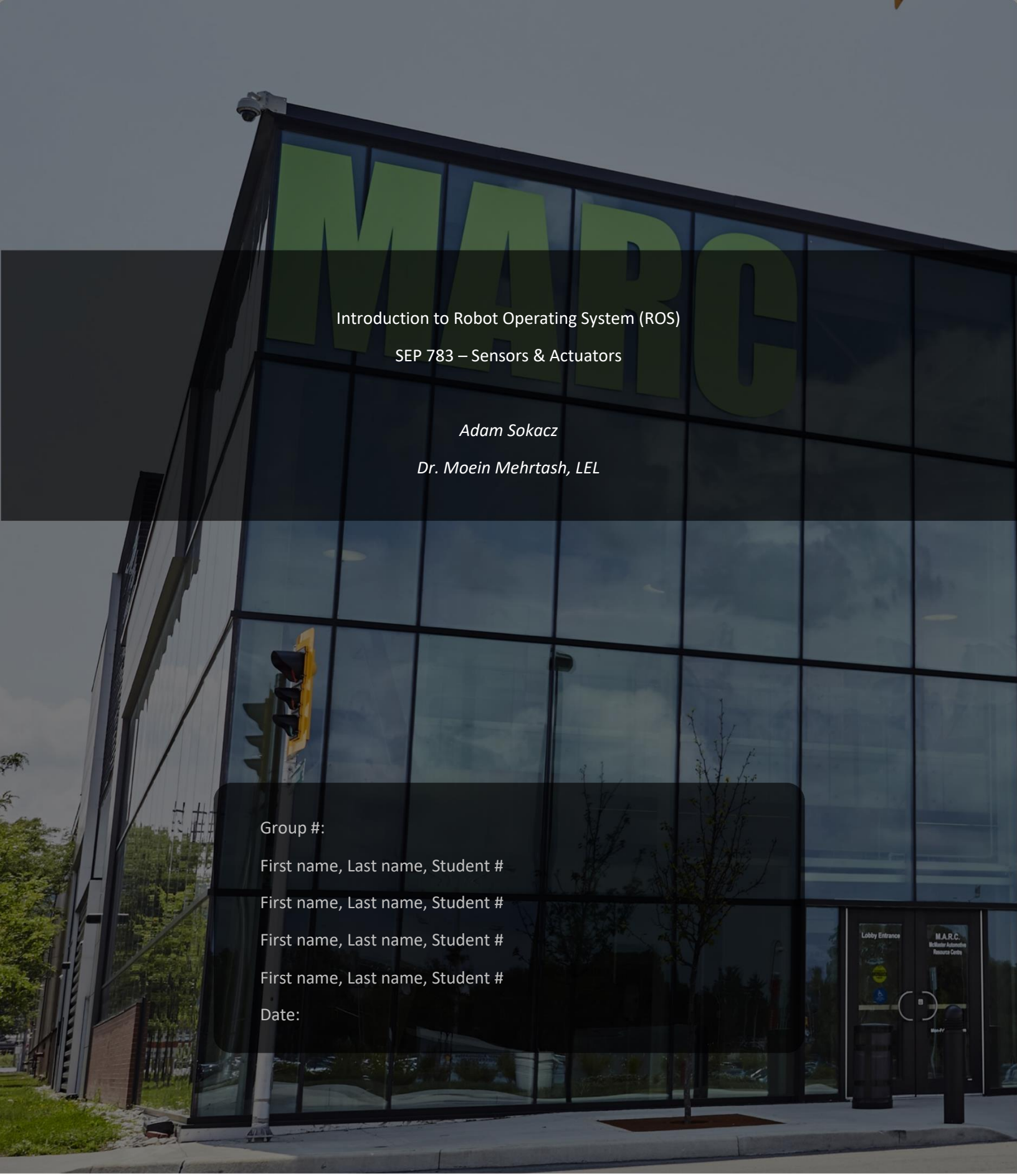


Introduction to Robot Operating System (ROS)

SEP 783 – Sensors & Actuators

Adam Sokacz

Dr. Moein Mehrtash, LEL

Group #:

First name, Last name, Student #

First name, Last name, Student #

First name, Last name, Student #

First name, Last name, Student #

Date:

W BOOTH SCHOOL OF ENGINEERING PRACTICE AND TECHNOLOGY

McMaster-Mohawk Bachelor of Technology Partnership

McMaster University ENGINEERING

mohawk

Objective

To familiarize yourselves with Linux, the basics of ROS middleware, and be comfortable reading and visualizing LiDAR sensor data.

Table of Contents

Objective 2

Pre-Lab Questions 4

Post-Lab Questions 4

Optional Assignment 5

Feedback 6

MacBot Setup 7

- Powering the MacBot 7
- Connecting to the Internet 7
- Connect Option #1 – Mouse and Keyboard 8
- Connect Option #2 – Remote Connection 9

Creating a New Workspace 17

Installing The YDLidar SDK 20

- Option #1 – Downloading from the Internet 20
  - PC Hotspot 20
  - Alternative – Connecting from the Terminal 24
  - Downloads 24
- Option #2 – Sending from PC Over Ethernet 24

Building the YDLiDAR SDK 30

Building the ROS YDLiDAR Package 34

Capturing and Visualizing LiDAR Data 36

- Connecting the YD LiDAR X2 36
- Connecting to a Local Network 42
- Updating Packages 43
- Installing Deepin-Terminal 44

ROS Setup 46

- Checking if ROS is Already Installed 46
- Installing ROS Melodic 46
- Creating a ROS Catkin Workspace 49

Lidar Setup 52
Cloning the YDLidar ROS Package 52
YDLiDAR SDK 53
Running the YDLidar ROS Package 60
Launch Files 60
Troubleshooting ROS Systems 66

Pre-Lab Questions

Q1 - What is *Ubuntu*? How is it different than *Windows* or *MacOS*? How is it similar?

(Suggested: 3 sentences)

Q2 - What is the difference between *sudo apt update* and *sudo apt upgrade*?

(Suggested: 1 sentence)

Q3 - What does the *sudo* keyword to when using it in front of a terminal command?

(Suggested: 1 sentence)

Q4 - What is the bash command to navigate into a *directory/folder*? How do we list the *file contents* of that folder?

(Suggested: 2 sentences)

Q5 - How do you create a *new file* from the bash terminal?

(Suggested: 1 sentence)

Post-Lab Questions

Q1 - What is *Robot Operating System (ROS)* in your own words? Search for and list *3 applications* of ROS in industry.

(Suggested: Short paragraph)

Q2 - What does the GREP command do in Linux? Use an example in your explanation.

(Suggested: Short paragraph)

Q3 - What does the WGET command do in Linux? Use an example in your explanation.

(Suggested: Short paragraph)

Q4 - What is an SSH tunnel? Use a Bash command example in your explanation.

(Suggested: Short paragraph)

Q5 - What is a Daemon in Linux? Use an example in your explanation.

(Suggested: Short paragraph)

Q6 - What role does *ROSCore* play in a functioning ROS system?

(Suggested: 1 sentence)

Q7 - What ROS command is used to *view* the current active topics? Take a screenshot of the *current active topics* on your system.

(Suggested: Screenshot)

Q8 - Which command was used to *echo* that topic to the terminal? Pick the *LiDAR topic* and echo it to the terminal.

(Suggested: Screenshot)

Q9 - Which command can be used to get *information* on a particular topic? Use that command on the *LiDAR data topic* and save a screenshot of your output.

(Suggested: Screenshot)

Q10 - What ROS tool can be used to *visualize* a stream of data? Take a screenshot of the visualization of your LiDAR data.

(Suggested: Screenshot)

Q11 – Write a brief LinkedIn post about key concepts that were learned in this lab.

(Suggested: Brief Paragraph)

Optional Assignment

Follow the tutorial below to program and build a 'Hello World' publish and subscribe node in either Python or C++. Take a screenshot that captures the data being sent and received.

Python: http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29

C++: http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29

Feedback

Q1 - What would you rate the difficulty of this lab?

(1 = easy, 5 = difficult)

1 **2** **3** **4** **5**

Comments about the difficulty of the lab:

Q2 - Did you have enough time to complete the lab within the designated lab time?

YES **NO**

Q3 - How easy were the lab instructions to understand?

(1 = easy, 5 = unclear)

1 **2** **3** **4** **5**

List any unclear steps:

Q4 - Could you see yourself using the skills learned in this lab to tackle future engineering challenges?

(1 = no, 5 = yes)

1 **2** **3** **4** **5**

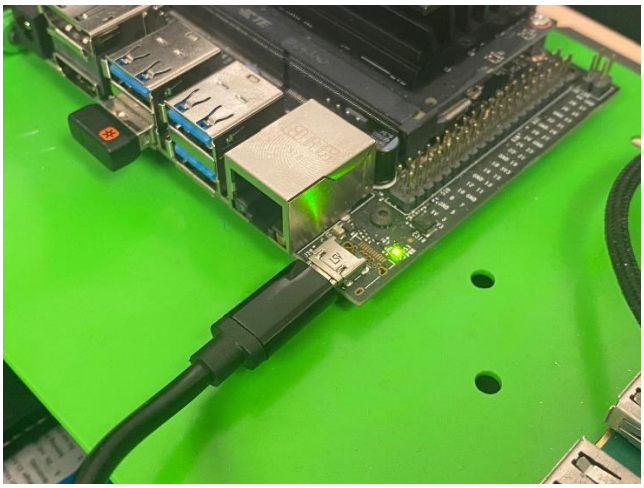
MacBot Setup

Powering the MacBot

There are 3 ways to power the Jetson Nano for this lab:

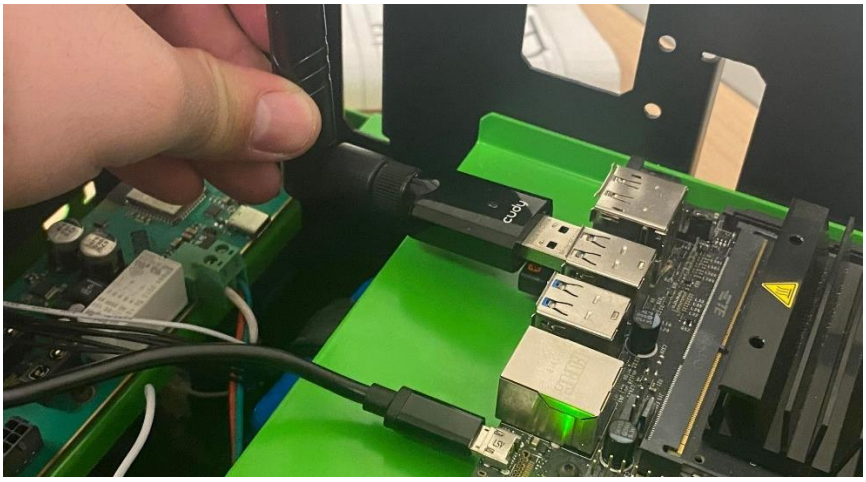
- MicroUSB cable
- 5 W barrel connector
- Internal battery

To keep things straightforward, we will be using either of the **wired** options.



Connecting to the Internet

The MacBot does not have a built-in network adapter. This means that an external one must be purchased in order to gain WiFi access. Ensure that the adapter being used is Linux-compatible.

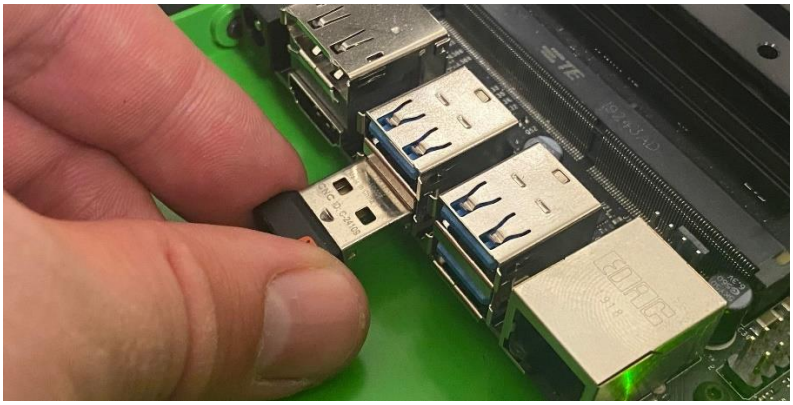


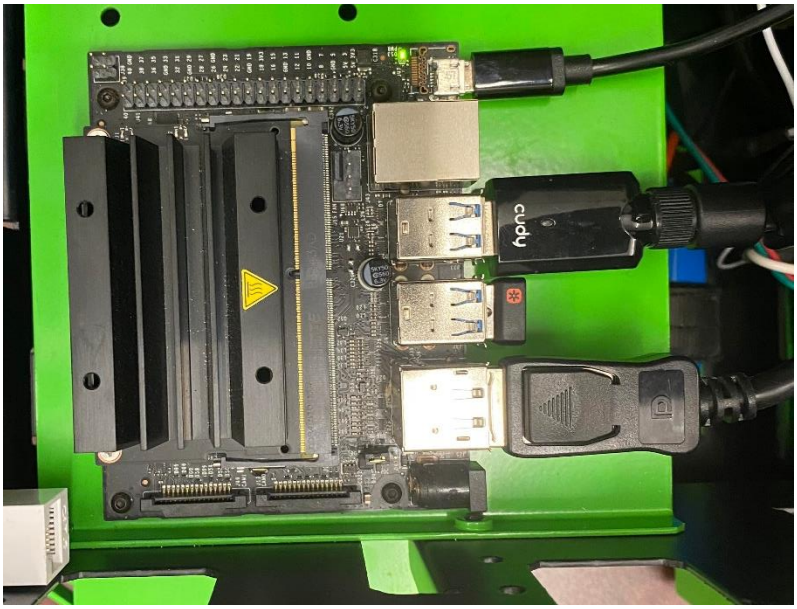
Connect Option #1 – Mouse and Keyboard

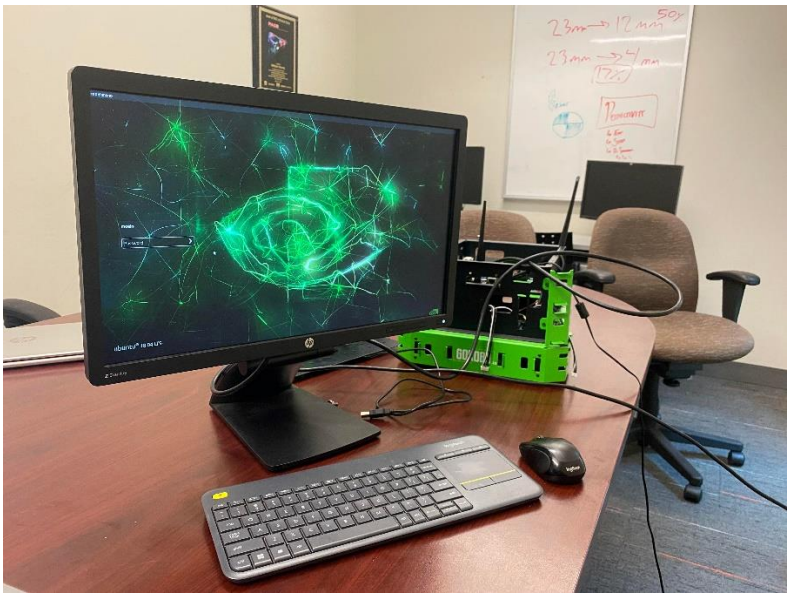
First, connect a monitor to the HDMI or **DisplayPort** port on the Jetson Nano.

Next, connect the power supply to the MacBot and watch it begin to boot into the graphical interface.

Lastly, connect a keyboard and mouse to the MacBot in order to interact with it.







Login Info:

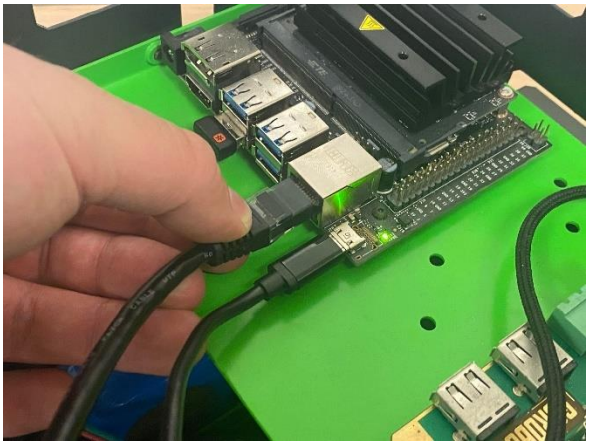
User: jnano

Password: 9055259140

Hint: The password is McMaster's general phone number

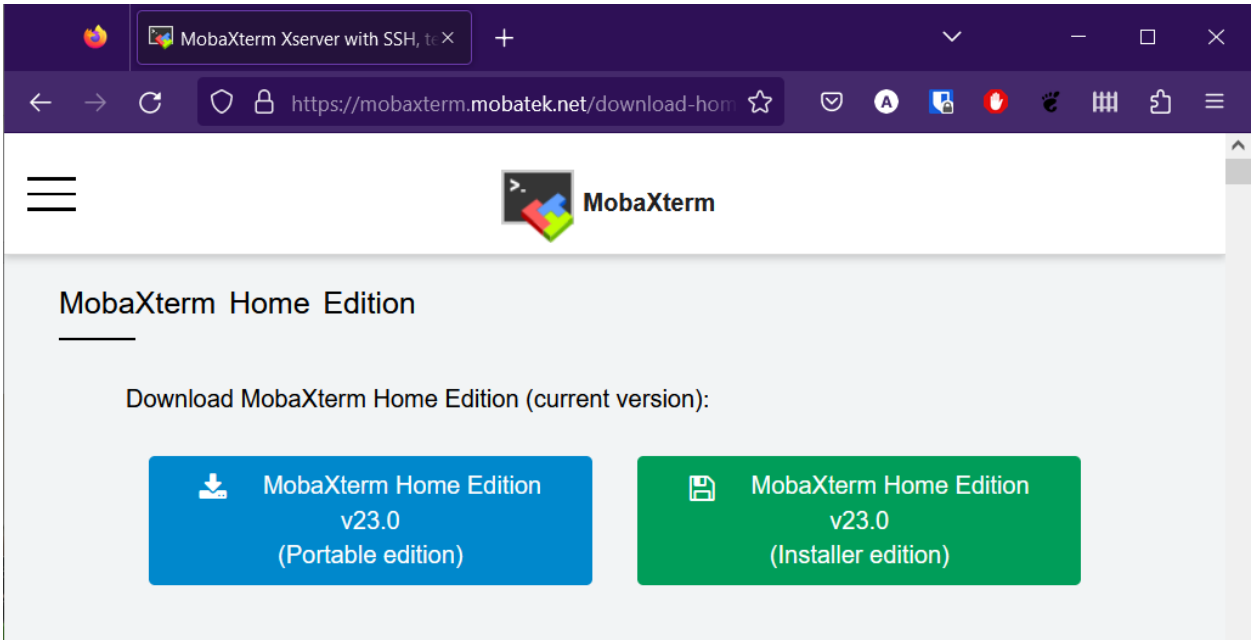
Connect Option #2 – Remote Connection

First, establish a network between the MacBot and your PC. The easiest way to accomplish this is through a wired ethernet connection using an **Ethernet-to-USB adapter**.

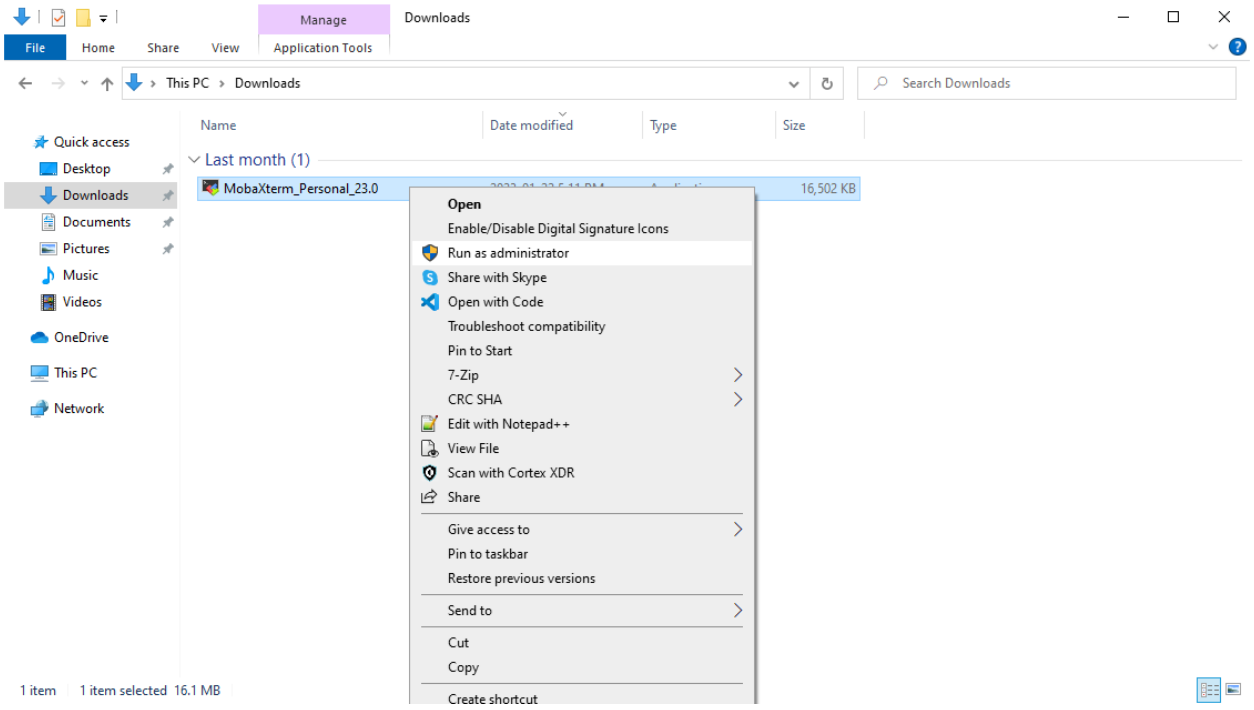


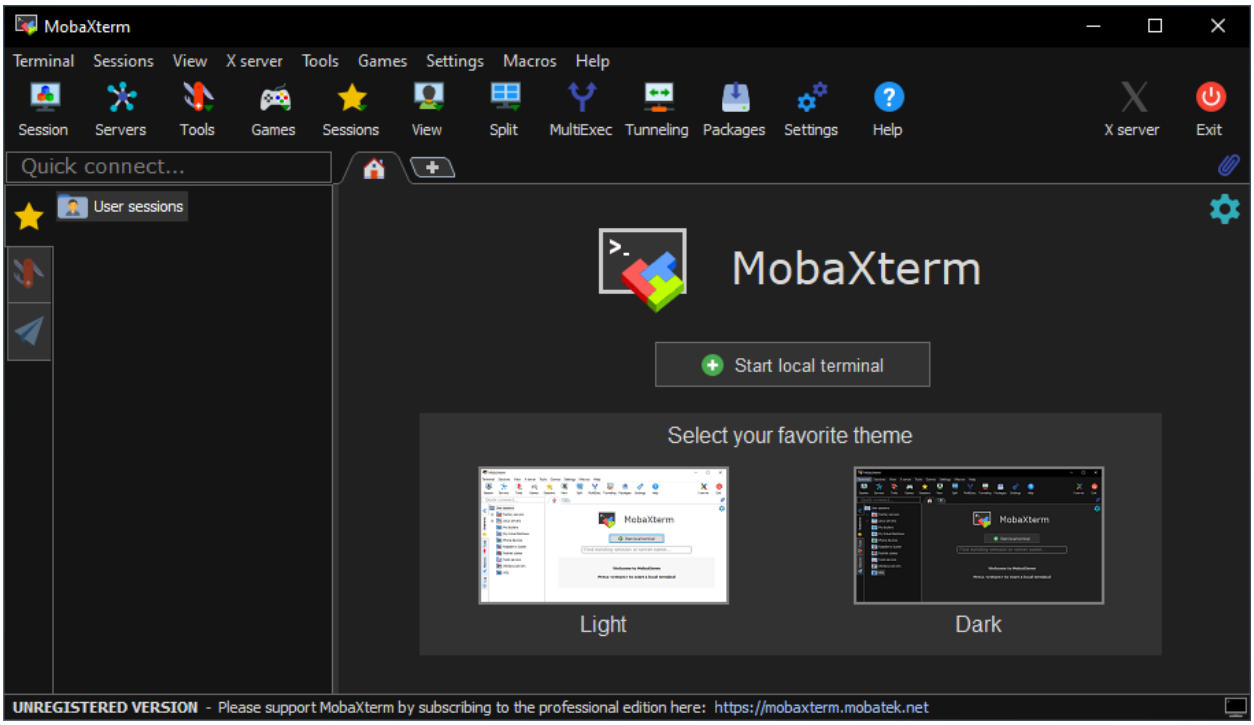
Next, install the following software:

MobaXTerm Home Edition Portable (https://mobaxterm.mobatek.net/download-home-edition.html)

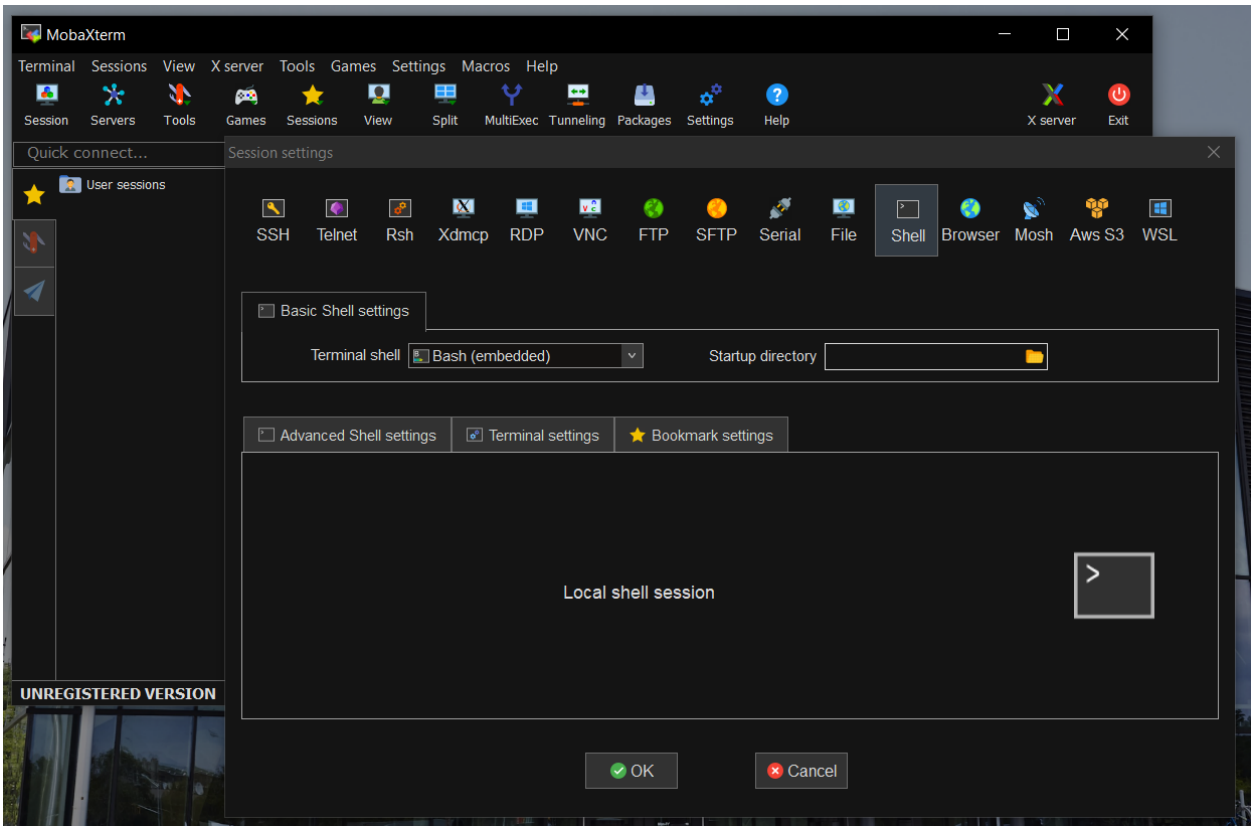


Launch **MobaXTerm** as **Administrator**.

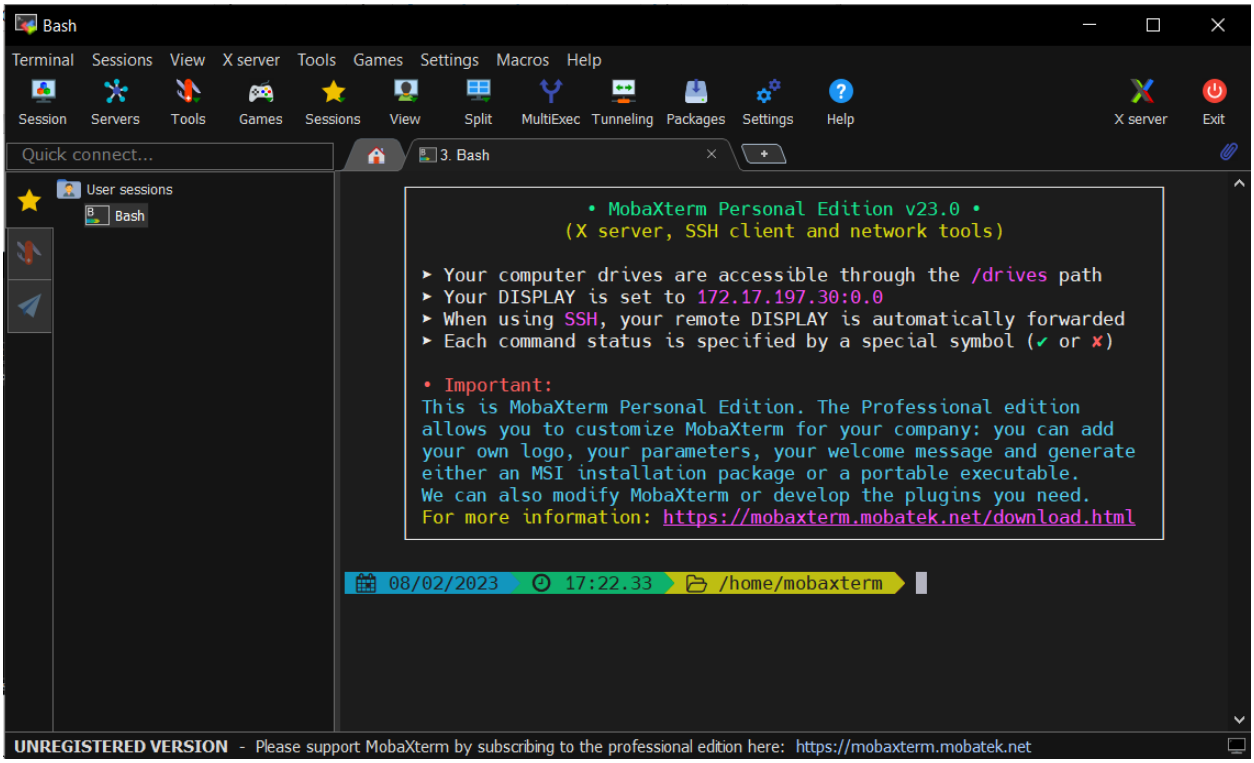




First, we need to make sure we can see the **MacBot**. To do this, navigate to **Session > Bash Terminal**.



Press **OK**. A new bash terminal tab should open.



Run the following command:

Ifconfig

This will show the available network interfaces.

```
08/02/2023  17:23.16  /home/mobaxterm  ifconfig

Software Loopback Interface 1
        Link encap: Local loopback
        inet addr:127.0.0.1 Mask: 255.0.0.0
        MTU: 1500 Speed:1073.74 Mbps
        Admin status:UP Oper status:OPERATIONAL
        RX packets:0 dropped:0 errors:0 unkown:0
        TX packets:0 dropped:0 errors:0 txqueuelen:0

Intel(R) Wireless-AC 9560 160MHz
        Link encap: IEEE 802.11 HWaddr: D0-AB-D5-6A-27-A0
        inet addr:172.17.197.30 Mask: 255.255.248.0
        MTU: 1500 Speed:144.40 Mbps
        Admin status:UP Oper status:OPERATIONAL
        RX packets:85032 dropped:0 errors:0 unkown:0
        TX packets:55552 dropped:0 errors:0 txqueuelen:0

Realtek USB GbE Family Controller
        Link encap: Ethernet HWaddr: 00-E0-4C-68-04-1F
        inet addr:169.254.36.242 Mask: 255.255.0.0
        MTU: 1500 Speed:1000.00 Mbps
        Admin status:UP Oper status:OPERATIONAL
        RX packets:101 dropped:0 errors:0 unkown:0
        TX packets:99 dropped:0 errors:0 txqueuelen:0
```

The *Realtek USB GbE Family Controller* is my USB-to-Ethernet adapter. The IP address listed **does not** represent the MacBot, it's the adapter itself.

To check if the MacBot is connected, attempt to ping your MacBot. The MacBot I am using is **macbot01**.

ping macbot01

```
08/02/2023  17:25.57  /home/mobaxterm  ping macbot01

Pinging macbot01.local [fe80::fde3:5968:aeae:5c20%10] with 32 bytes of data:
Reply from fe80::fde3:5968:aeae:5c20%10: time=2ms
Reply from fe80::fde3:5968:aeae:5c20%10: time=1ms
Reply from fe80::fde3:5968:aeae:5c20%10: time=1ms
Reply from fe80::fde3:5968:aeae:5c20%10: time=2ms

Ping statistics for fe80::fde3:5968:aeae:5c20%10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 2ms, Average = 1ms


08/02/2023  17:26.04  /home/mobaxterm
```

macbot01 is connected to my PC with the IP address of *fe80::fde3:5968:aeae:5c20%10.*

Next, we will attempt to connect to the MacBot using a secure protocol called **secure socket shell**.

ssh <username>@<ip_address_or_pc_name>

ssh jnano@macbot01

We will use the **-L** flag to create an SSH tunnel between port 5902 of our PC and port 5902 of the MacBot to allow us to connect to the visual interface securely.

ssh -L 5902:localhost:5902 jnano@macbot01

If prompted, type **yes** and **<enter>**.

The password is: **9055259140**, which is the general phone number for McMaster University.

```
08/02/2023  17:30.21  /home/mobaxterm  ssh -L 5902:localhost:5902 jnano@macbot01
jnano@macbot01's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.9.299-tegra aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

82 updates can be applied immediately.
74 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Mon Feb  6 21:08:08 2023 from fe80::23f5:bfa5:5ac9:fb71%eth0
ROS Melodic sourced
LIDAR_WS sourced
jnano@macbot01:~$
```

While connected, we will launch a VNC server using the following command:

vncserver :2 -depth 24 -geometry 1920x1080

```
jnano@macbot01:~$ vncserver :2 -depth 24 -geometry 1920x1080

New 'macbot01:2 (jnano)' desktop at :2 on machine macbot01

Starting applications specified in /home/jnano/.vnc/xstartup
Log file is /home/jnano/.vnc/macbot01:2.log

Use xtigervncviewer -SecurityTypes VncAuth -passwd /home/jnano/.vnc/passwd :2 to connect to the VNC server.

jnano@macbot01:~$
```

To ensure that the server is running, we can use the following command:

vncserver -list

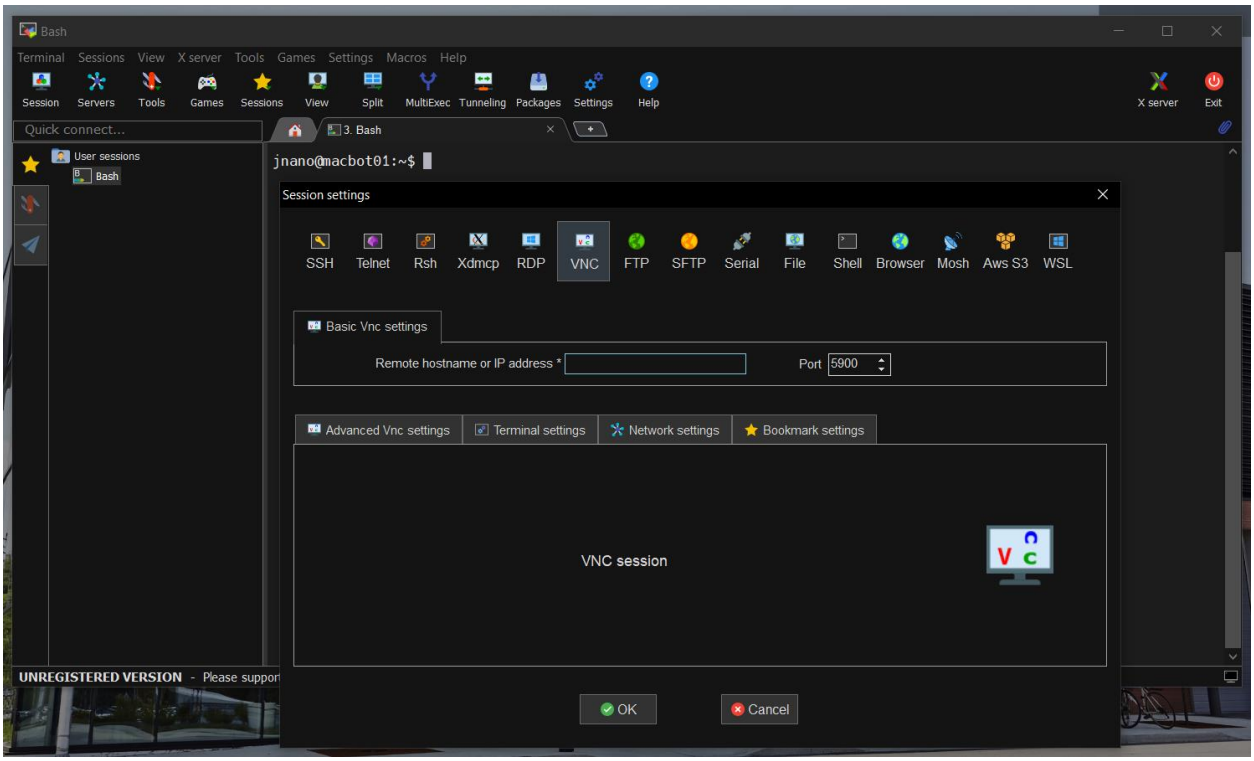
```
jnano@macbot01:~$ vncserver -list

TigerVNC server sessions:

X DISPLAY #     PROCESS ID
:2              7316
jnano@macbot01:~$
```

VNC port 2 is routed to 5900 + 2 = 5902, which is why we tunneled that port.

Without closing the ssh tunnel, open a new **VNC** tab in **MobaXTerm**.

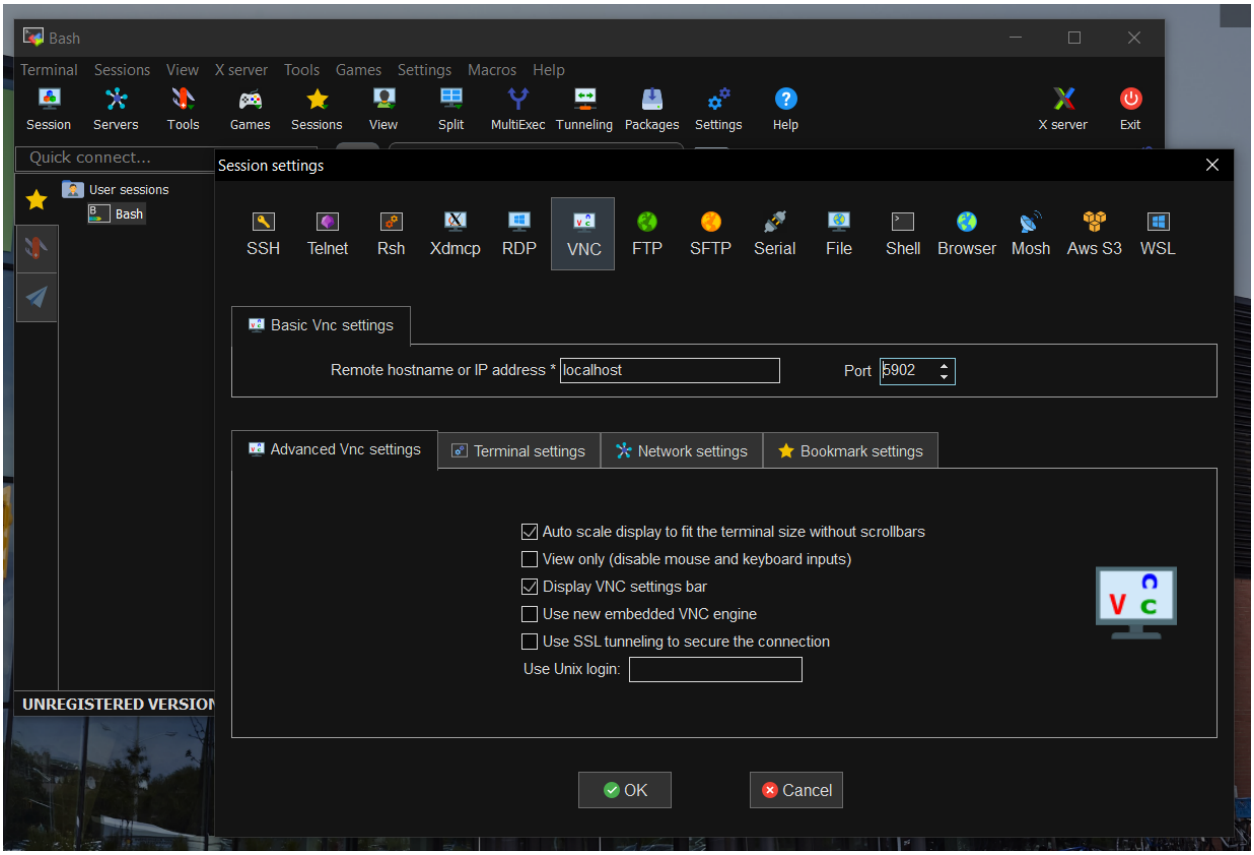


Set the following configuration:

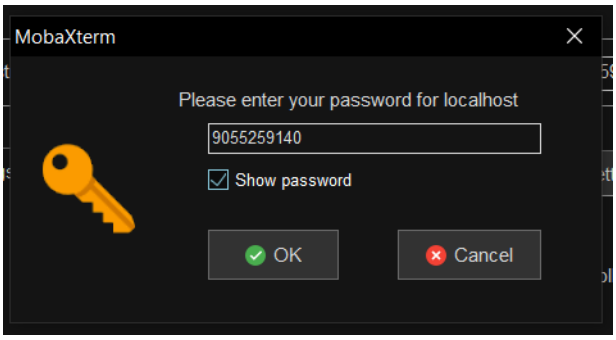
hostname: localhost

port: 5902

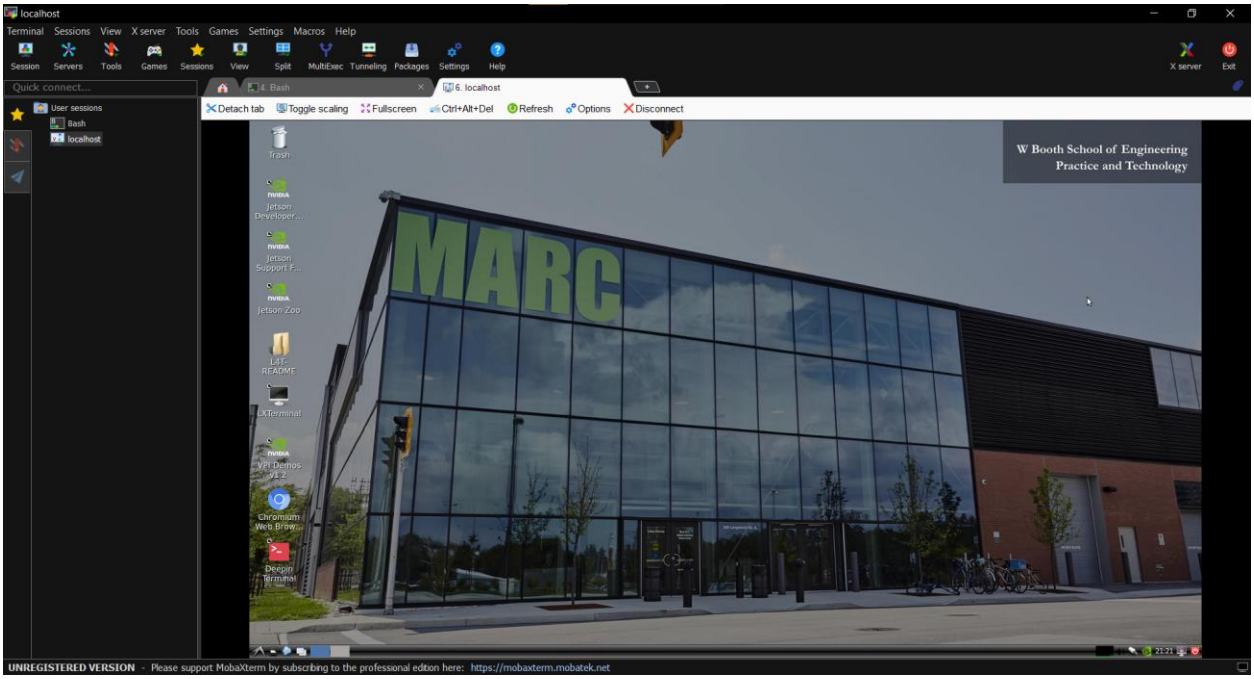
Uncheck: Use new embedded VNC engine.



Press **OK** and enter the password when prompted.



A full desktop environment should be visible after a few moments of waiting.



It is suggested to click **Fullscreen**, then unselect **Stay on Top**. You should now be able to cycle through your windows using either the Windows key or **Windows + Tab**.

Creating a New Workspace

Navigate to the home/ directory using the following command:

cd ~

Create a new folder called **lidar_ws/** with a subdirectory within it called **src/**.

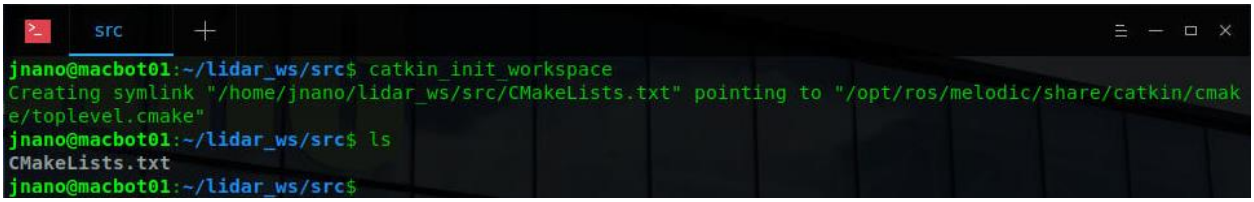
mkdir -p lidar_ws/src

cd lidar_ws/src

Initialize the workspace using the following command from the **~/lidar_ws/src** directory:

catkin_init_workspace

A new file called **CMakeLists.txt** should have been generated.

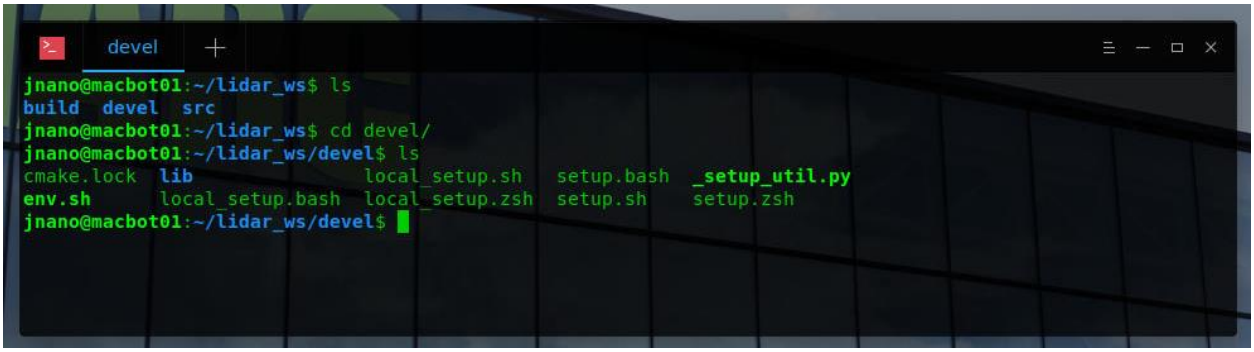


Navigate to the **~/lidar_ws** directory and build the empty workspace using the **catkin_make** command.

cd ~/lidar_ws

catkin_make

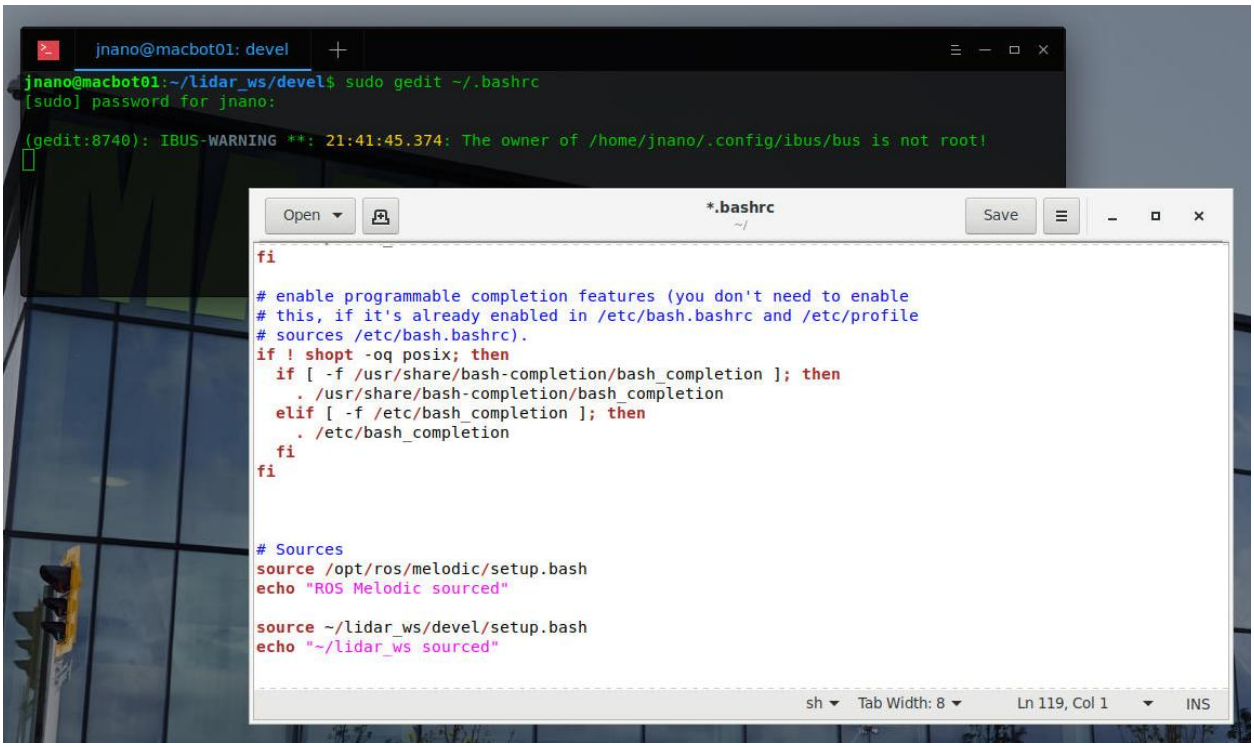
Notice that after the build is successful, some new directories have been generated. An important file is the **devel/setup.bash** script. We will need to load this script every time we open a new terminal emulator window in order to access workspace files when running ROS commands.



Use **gedit** as **superuser** to open the ~/**.bashrc** configuration script. This script runs each time a new terminal window is open. We will be appending commands to the **end** of bashrc to automatically source our ROS installation and workspace.

sudo gedit ~/.bashrc

<type_password>



Notice the #Sources section that was previously added. Ensure that you have these two paths sourced and echoed to the terminal window.

 # Sources

source /opt/ros/melodic/setup.bash

echo "ROS Melodic sourced"

source ~/lidar_ws/devel/setup.bash

echo "~/lidar_ws sourced"

Save and close the file.

Open a new teminal window.

You should notice the statements printing at the top of the window. **Close** the old window.

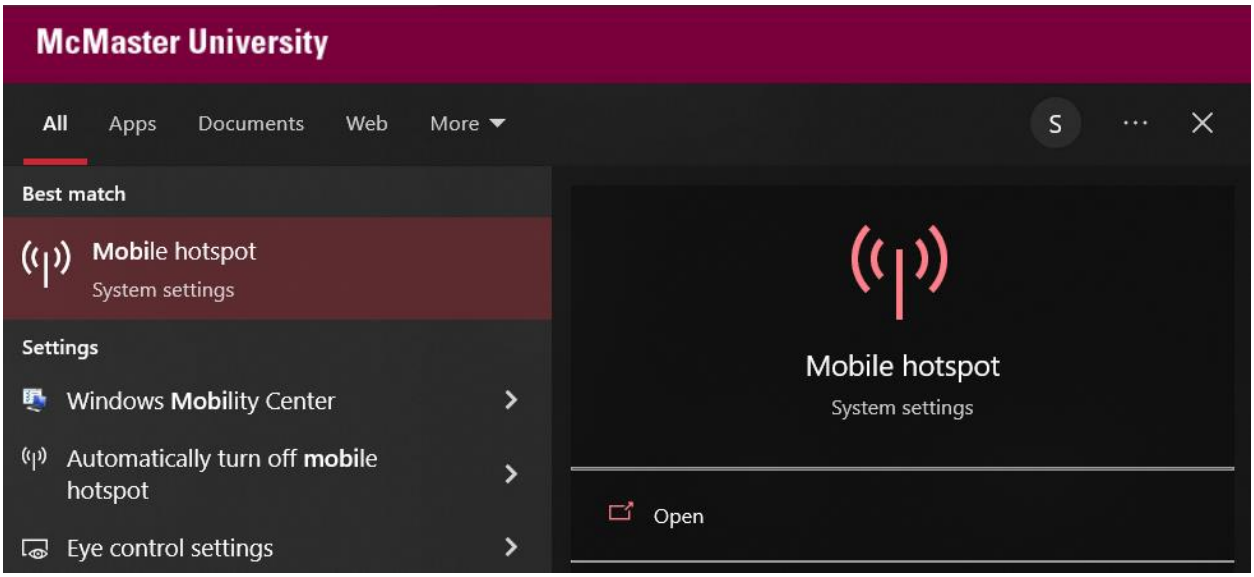
Installing The YDLidar SDK

Option #1 – Downloading from the Internet

If you have a WiFi adapter connected to your Jetson Nano, you can configure it to connect to either a PC hotspot or MAC_WIFI.

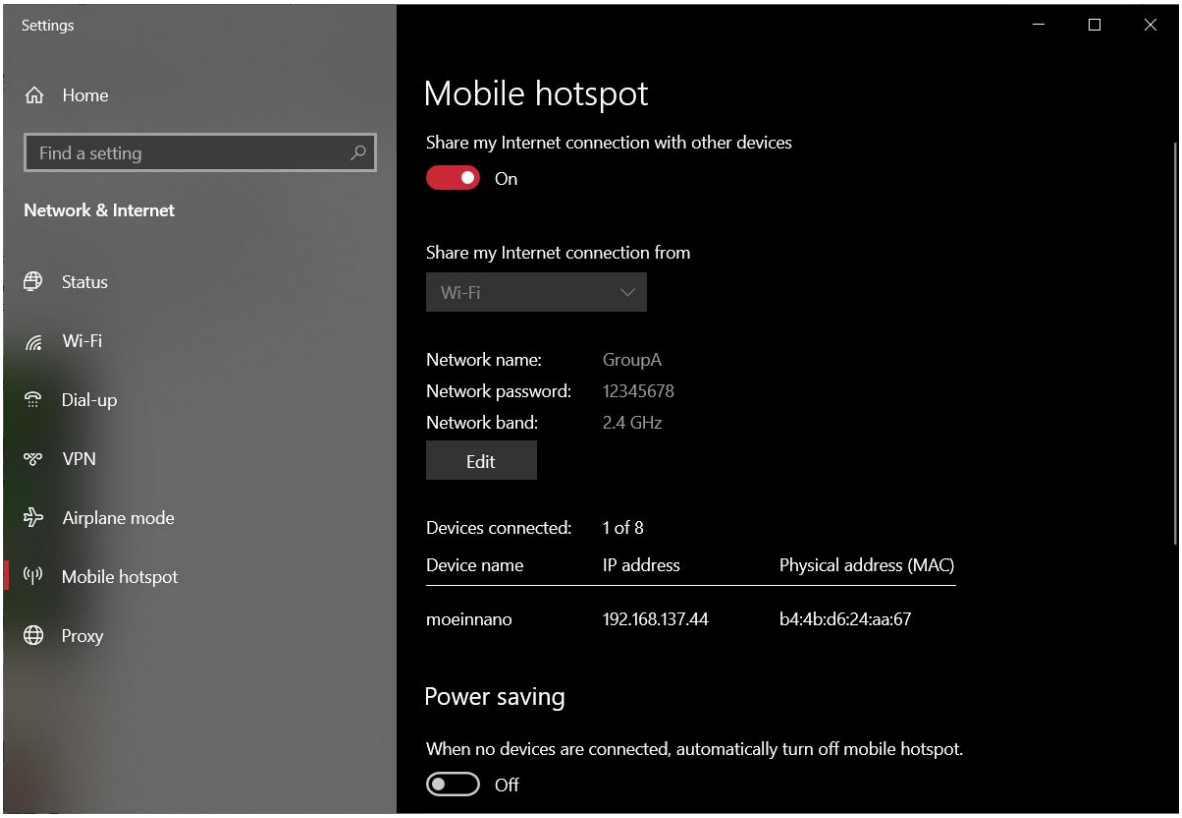
PC Hotspot

Open the Mobile Hotspot application on a personal laptop.

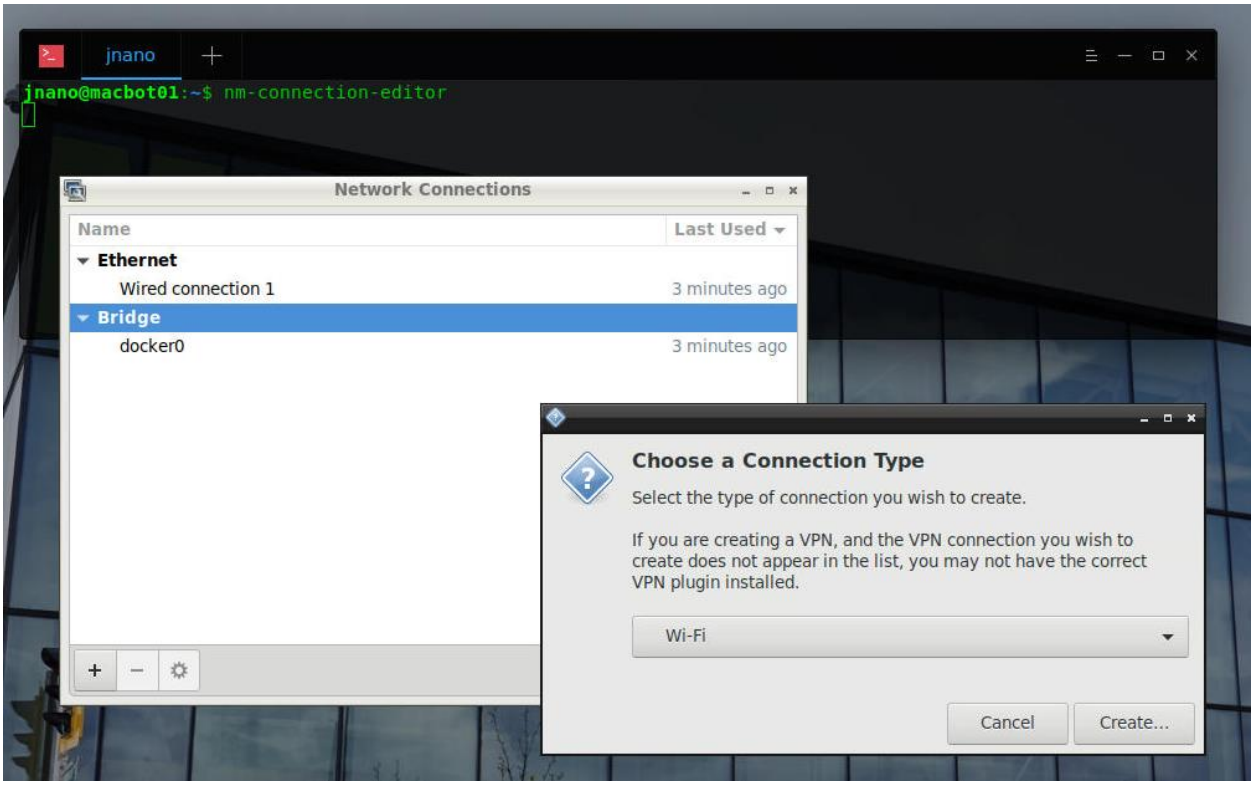


Configure a **unique SSID** and simple 8-digit password.

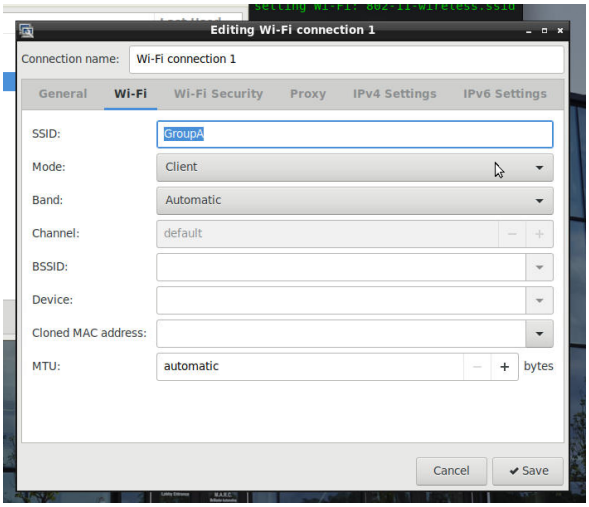
To avoid any compatibility issues, ensure that the **Network Band** is set to **2.4GHz**, and turn off **Power Saving** mode so the hotspot runs continuously. Toggle it **ON**.

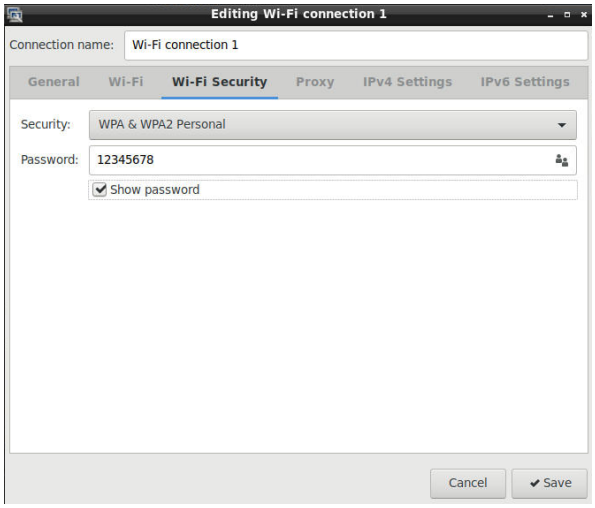


To connect to it, we can use the **nm-connection-editor.**

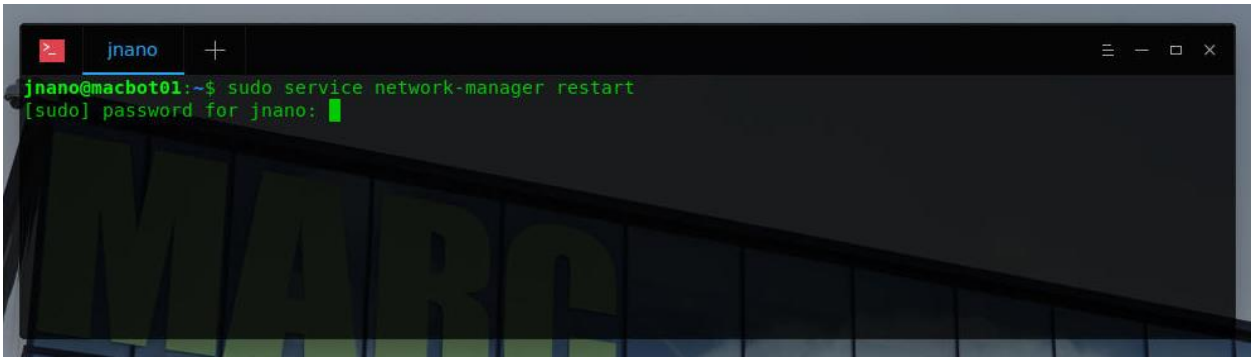


Set the SSID and password.



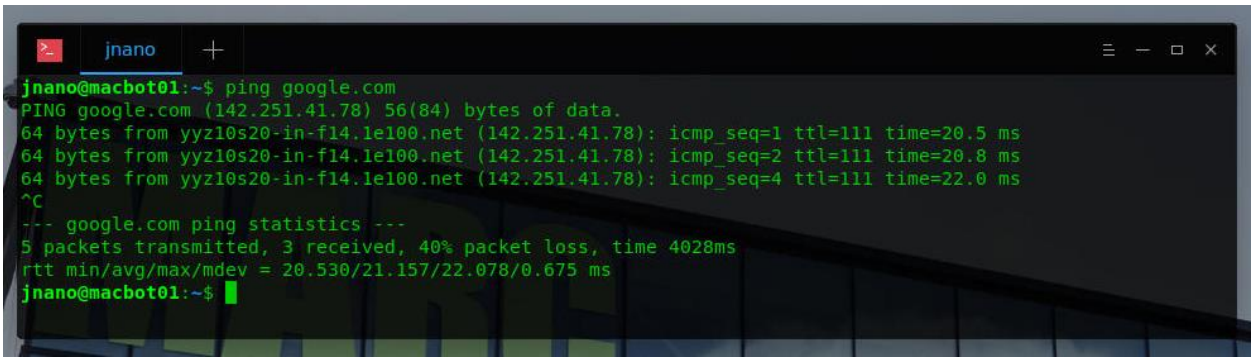


Restart the network manager.

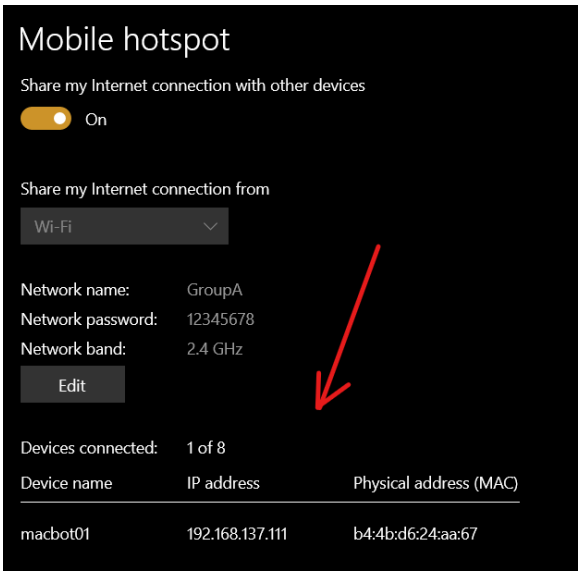


Wait **20 seconds** as your connection to the MacBot is temporarily interrupted.

Once the VNC starts responding again, attempt to **ping** google.com to verify that it has successfully connected to the internet. Give it a minute or two to connect if it fails the first time.



You should also see the connection on your mobile hotspot interface.



Alternative – Connecting from the Terminal

sudo nmcli dev wifi connect <ssid> password "<password>"

sudo nmcli dev wifi connect <GroupA> password "<12345678>"

sudo service network-manager restart

Downloads

Download the following packages onto the MacBot.

https://github.com/YDLIDAR/ydlidar_ros_driver

https://github.com/YDLIDAR/YDLidar-SDK

cd ~/lidar_ws/src

git clone https://github.com/YDLIDAR/ydlidar_ros_driver

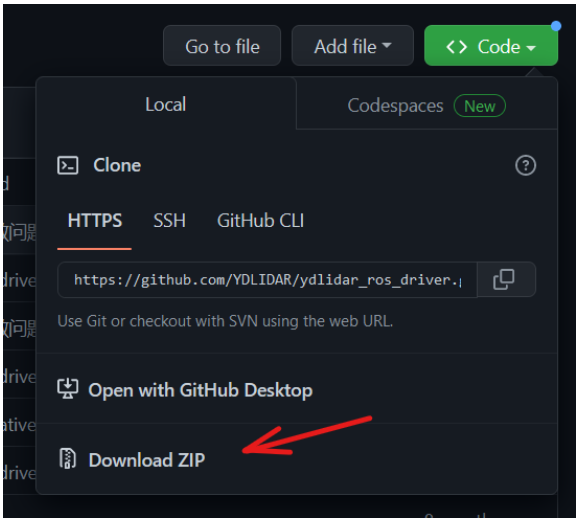
cd ~/Documents

git clone https://github.com/YDLIDAR/YDLidar-SDK

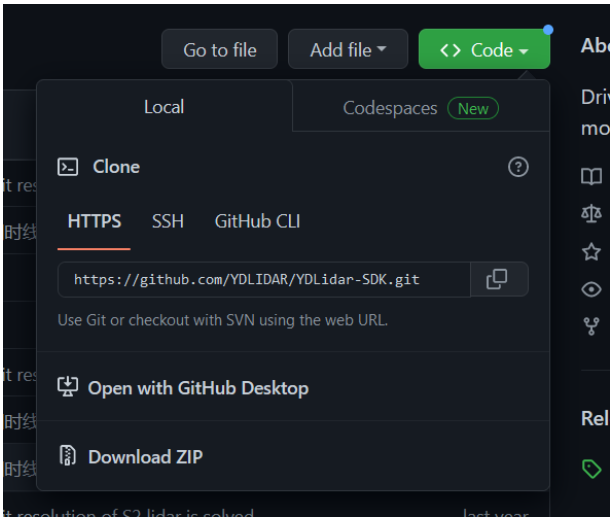
Option #2 – Sending from PC Over Ethernet

Navigate to the following GitHub repository and download the package as **ZIP.**

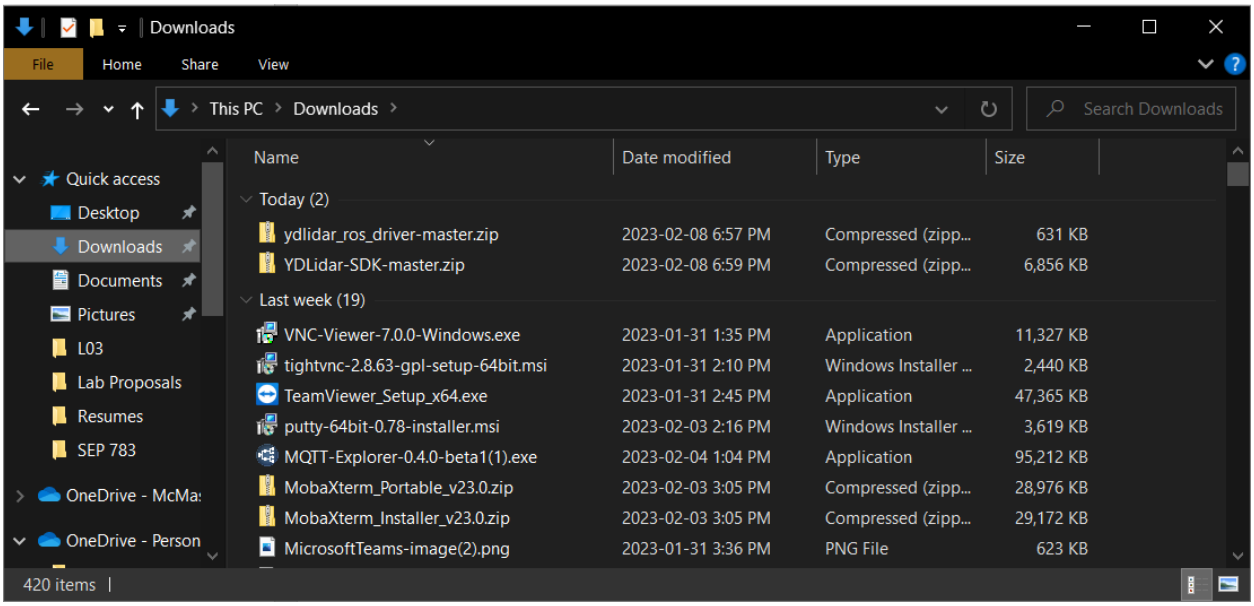
https://github.com/YDLIDAR/ydlidar_ros_driver



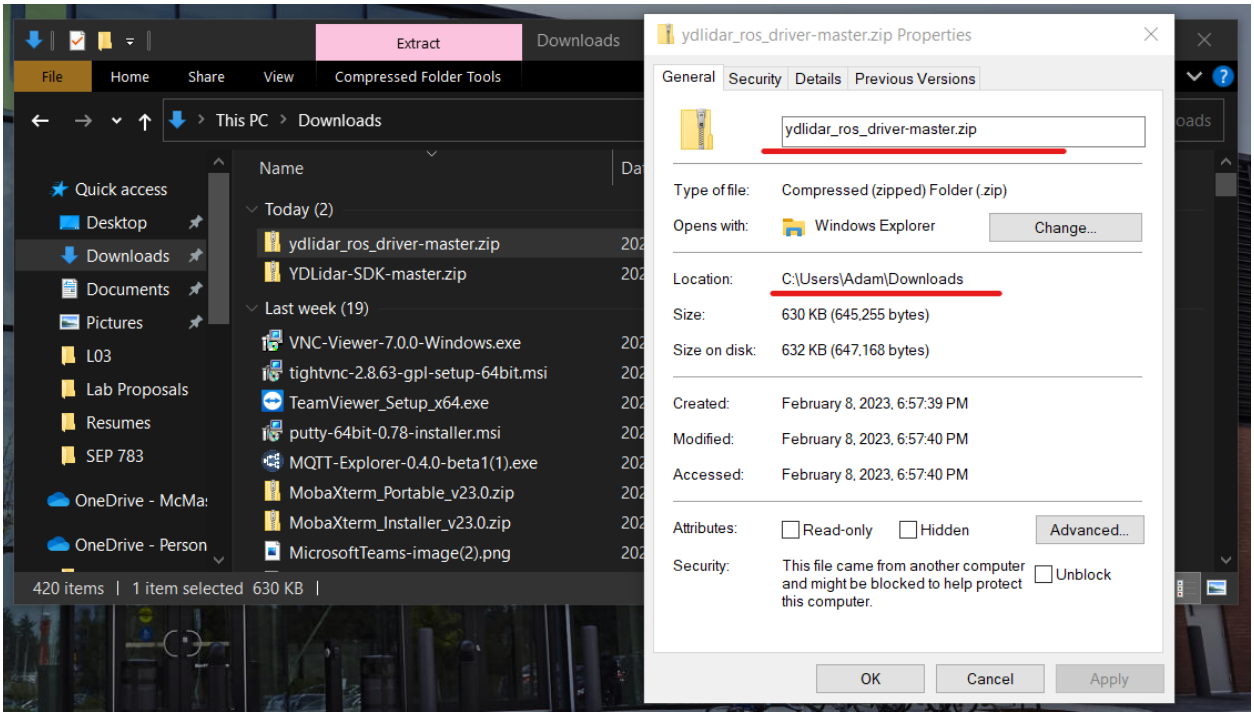
https://github.com/YDLIDAR/YDLidar-SDK



The files have both been downloaded into your PC's **Downloads/** folder.



View the path and the name of the files you wish to transfer over.

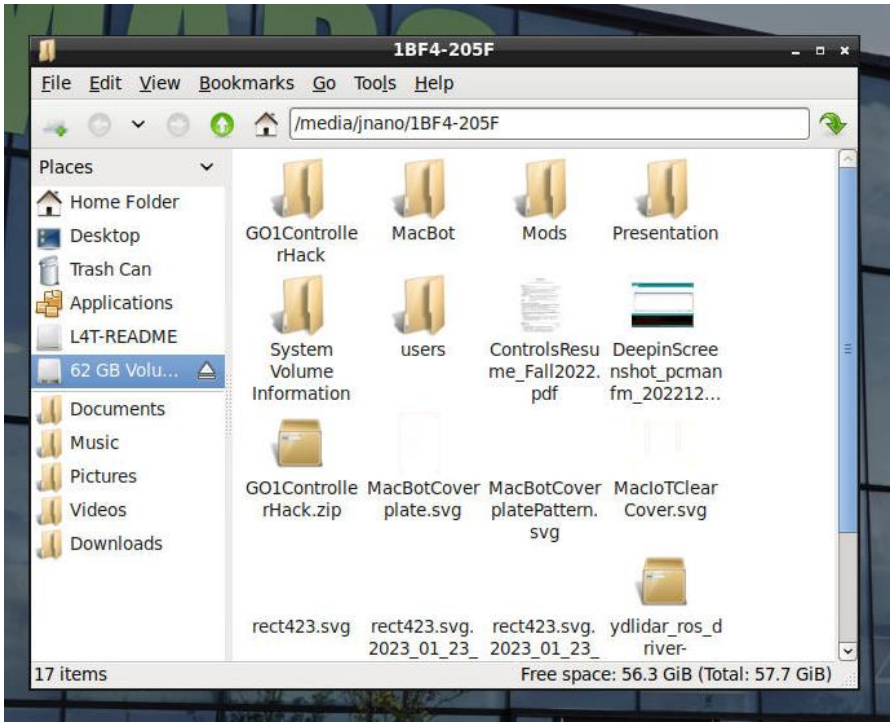


Here are the file transfers:

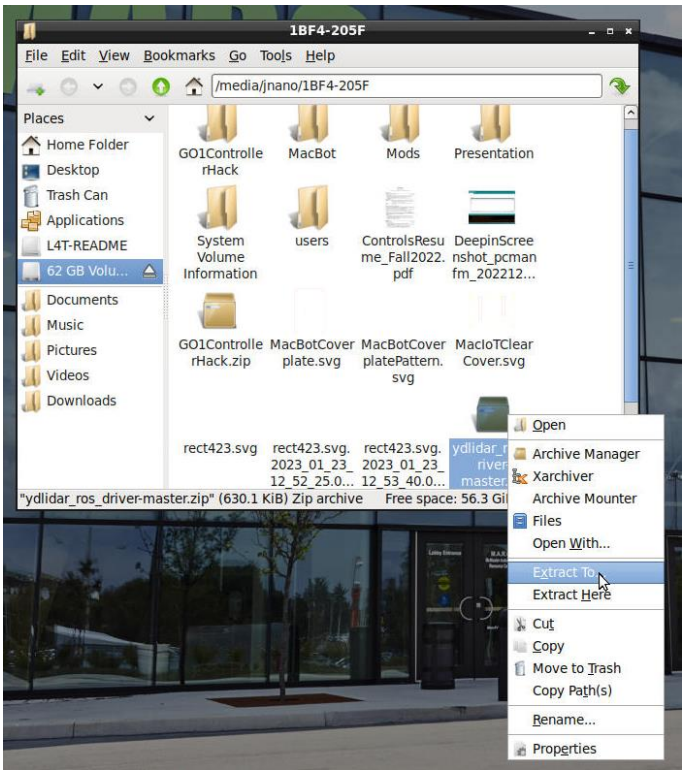
LiDAR SDK: *"C:\Users\Adam\Downloads\YDLidar-SDK-master.zip" → /home/jnano/Documents*

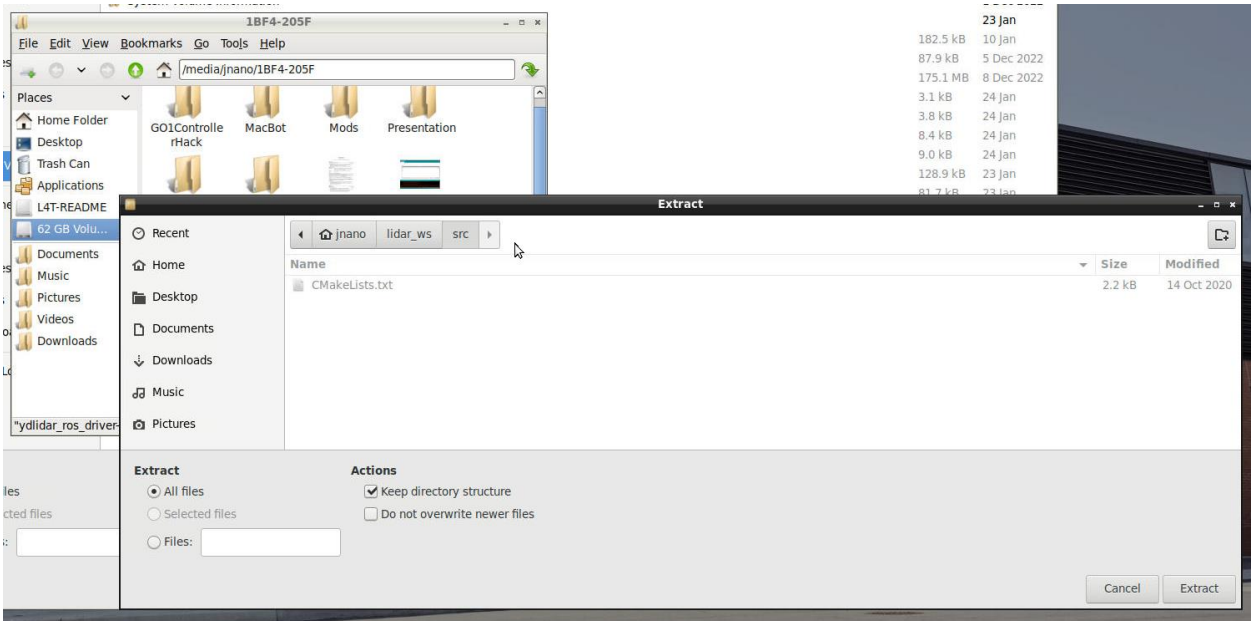
LiDAR ROS: *"C:\Users\Adam\Downloads\ydlidar_ros_driver-master.zip" → /home/jnano/lidar_ws/src*

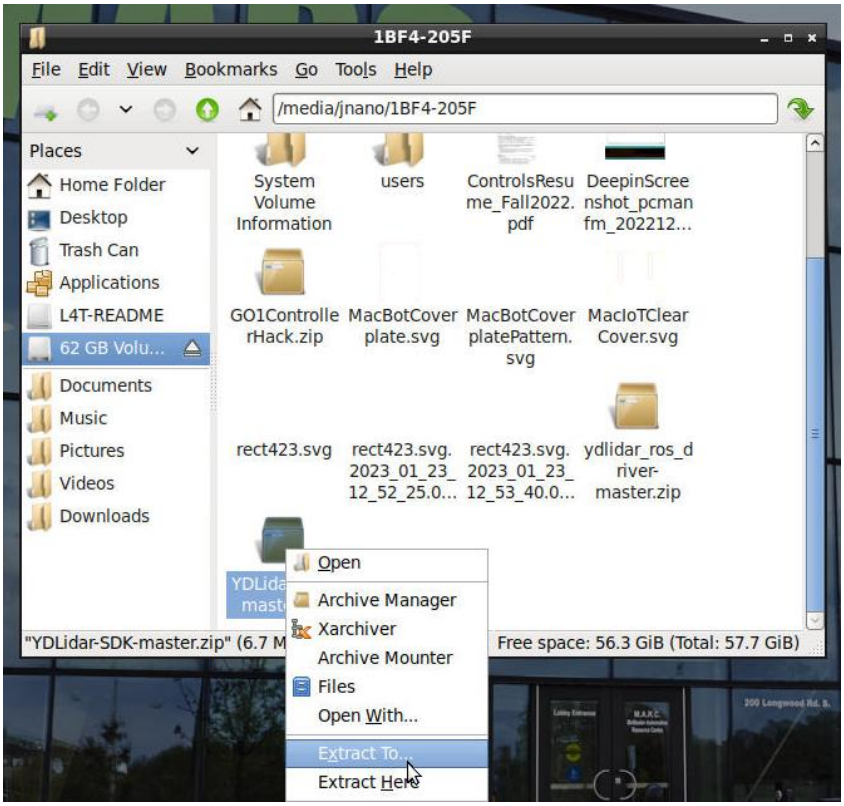
You can do this using SFTP from within MobaXTerm, but I think the least complicated way is to just use a USB thumb drive and copy the files to it.

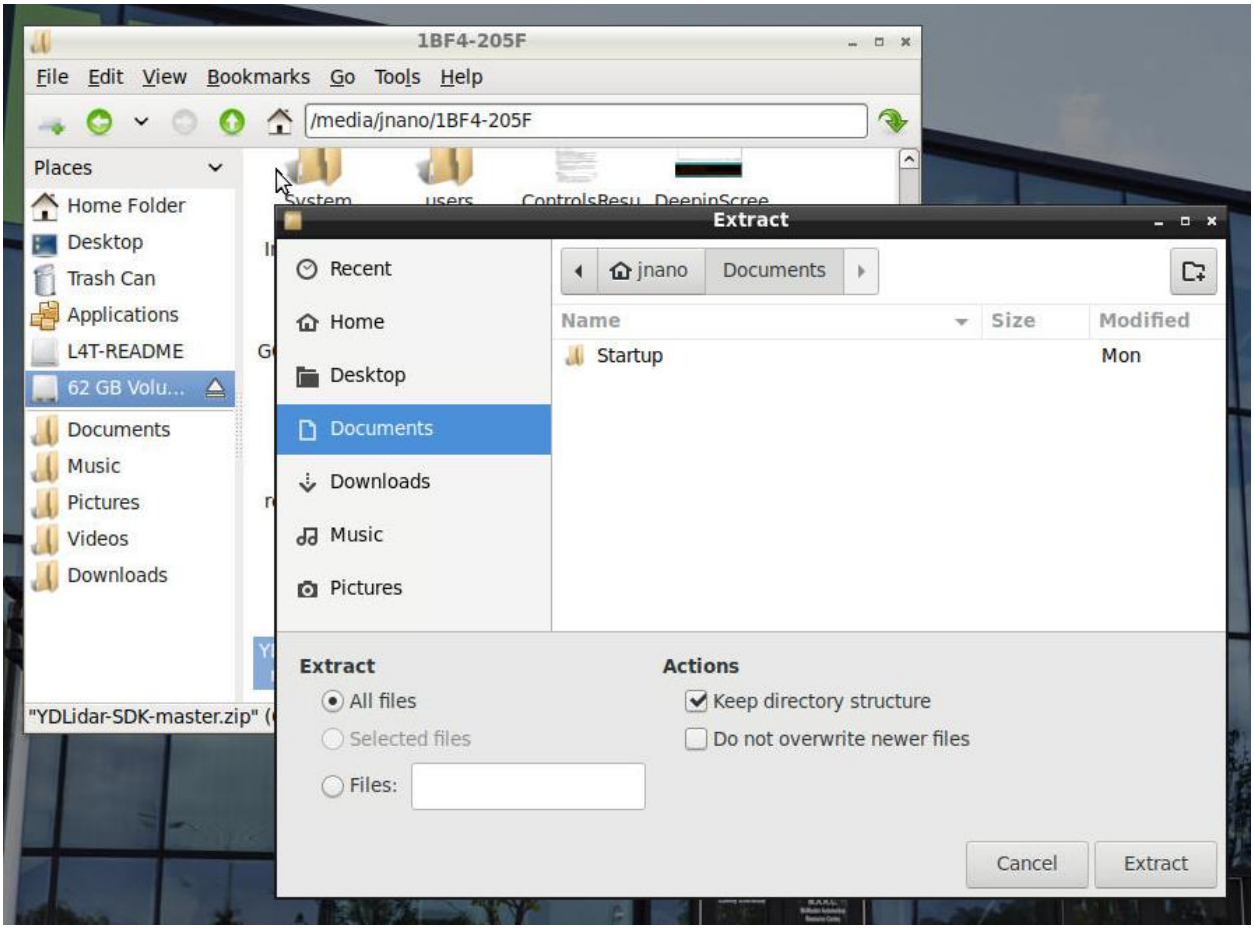


Drag and drop the files to their correct locations within VNC.



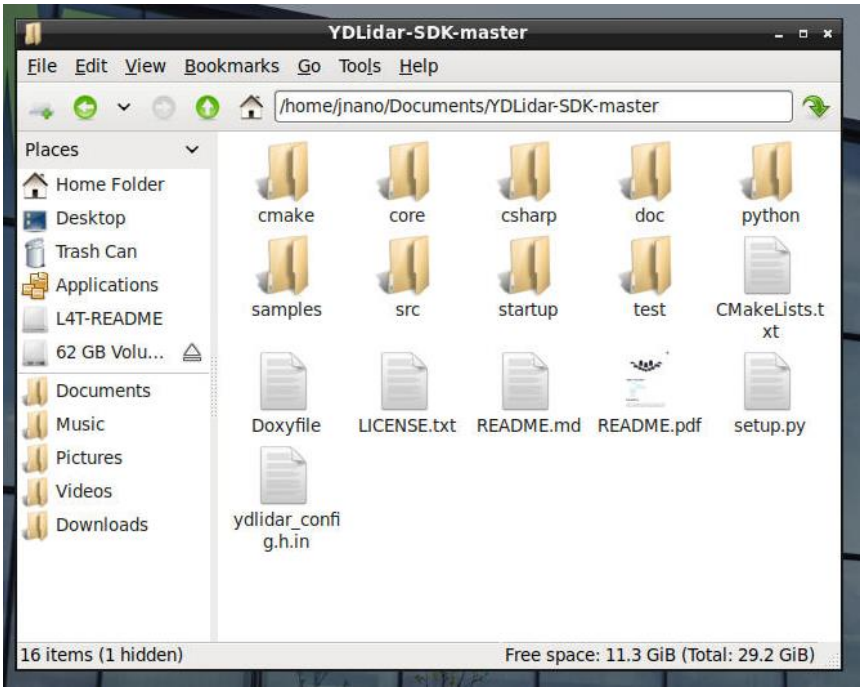

1BF4-205F
File Edit View Bookmarks Go Tools Help
/media/jnano/1BF4-205F
Places
Home Folder
Desktop
Trash Can
Applications
L4T-README
62 GB Volu...
Documents
Music
Pictures
Videos
Downloads
GO1ControlleRHack
MacBot
Mods
Presentation
Extract
jnano
lidar_ws
src
Name
Size
Modified
CMakeLists.txt
2.2 kB
14 Oct 2020
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Extract
All files
Selected files
Files:
Actions
Keep directory structure
Do not overwrite newer files
Cancel
Extract


1BF4-205F
File Edit View Bookmarks Go Tools Help
/media/jnano/1BF4-205F
Places
Home Folder
Desktop
Trash Can
Applications
L4T-README
62 GB Volu...
Documents
Music
Pictures
Videos
Downloads
System Volume Information
users
ControlsResume_Fall2022.pdf
DeepinScreenshot_pcmanfm_202212...
GO1ControlleRHack.zip
MacBotCoverplate.svg
MacBotCoverplatePattern.svg
MacIoTClearCover.svg
rect423.svg
rect423.svg.2023_01_23_12_52_25.0...
rect423.svg.2023_01_23_12_53_40.0...
ydlidar_ros_driver-master.zip
Open
Archive Manager
Xarchiver
Archive Mounter
Files
Open With...
Extract To...
Extract Here
"YDLidar-SDK-master.zip" (6.7 M
Free space: 56.3 GiB (Total: 57.7 GiB)

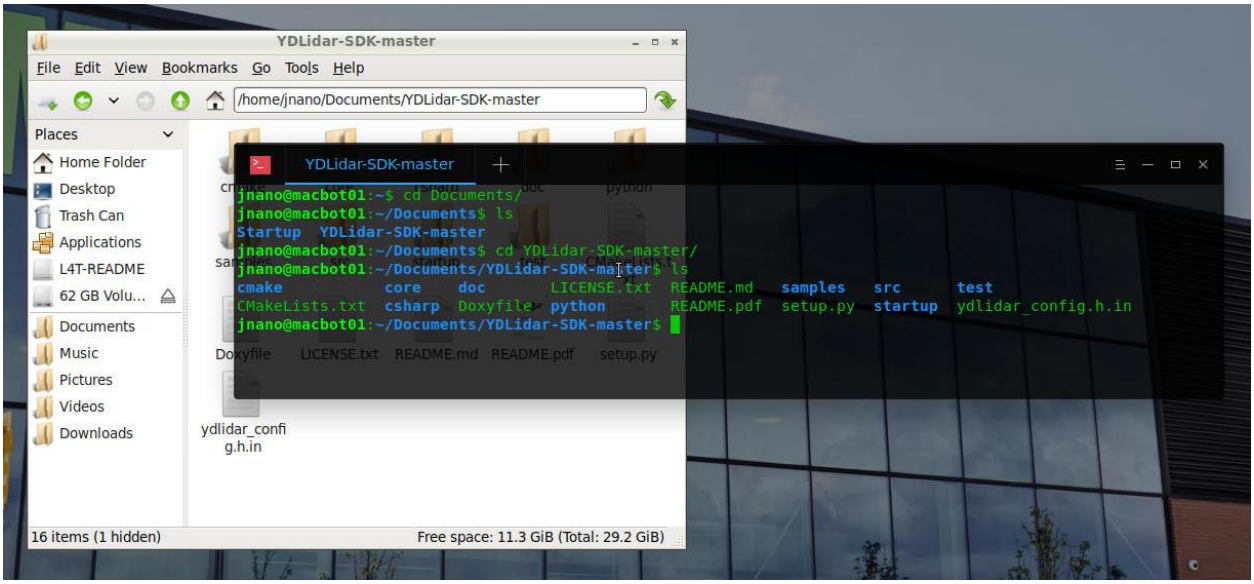

1BF4-205F
File Edit View Bookmarks Go Tools Help
/media/jnano/1BF4-205F
Places
Home Folder
Desktop
Trash Can
Applications
L4T-README
62 GB Volu...
Documents
Music
Pictures
Videos
Downloads
"YDLidar-SDK-master.zip"
Extract
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
jnano
Documents
Name
Size
Modified
Startup
Mon
Extract
All files
Selected files
Files:
Actions
Keep directory structure
Do not overwrite newer files
Cancel
Extract

Building the YDLiDAR SDK

Verify that the SDK has been successfully extracted into the **/home/jnano/Documents** folder.



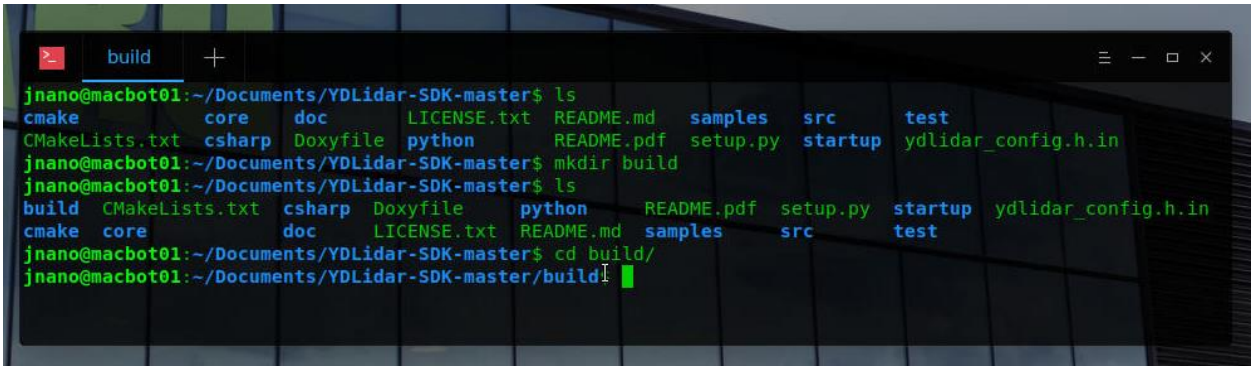
Open a terminal window and navigate to this Directory.



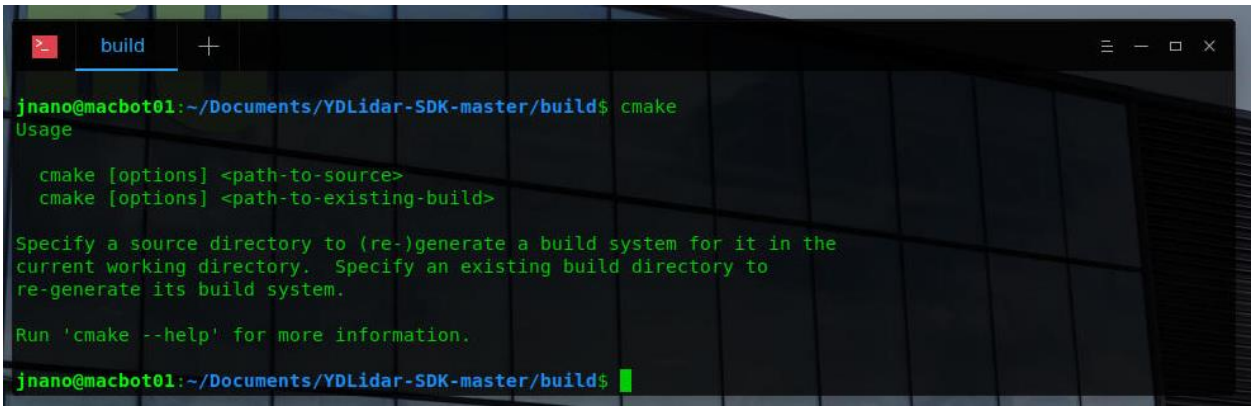
Make a directory called **build/**.

mkdir build

cd build

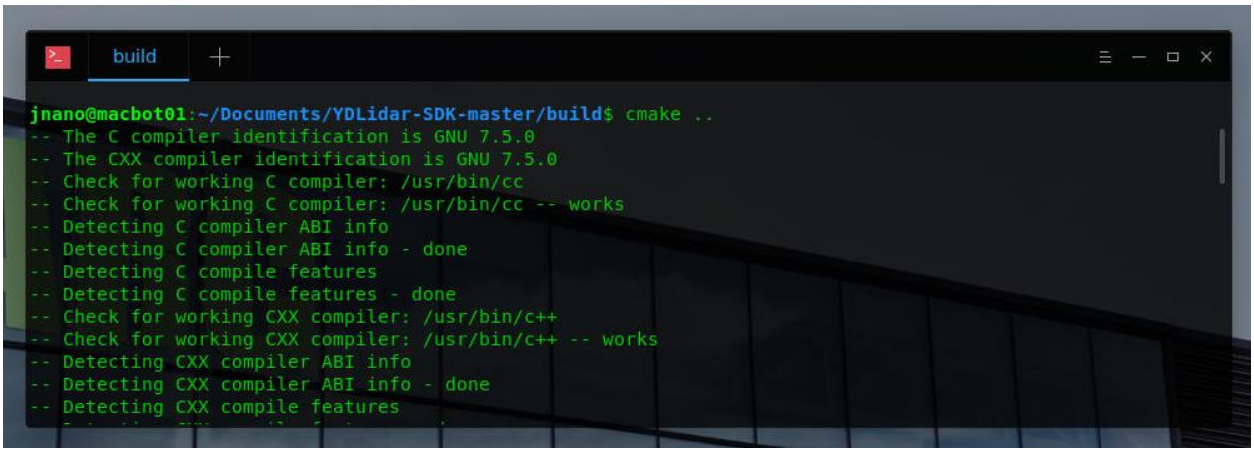


If you run **cmake** without any command line arguments, it will give your more information on what the program requires to prepare the build for a project.

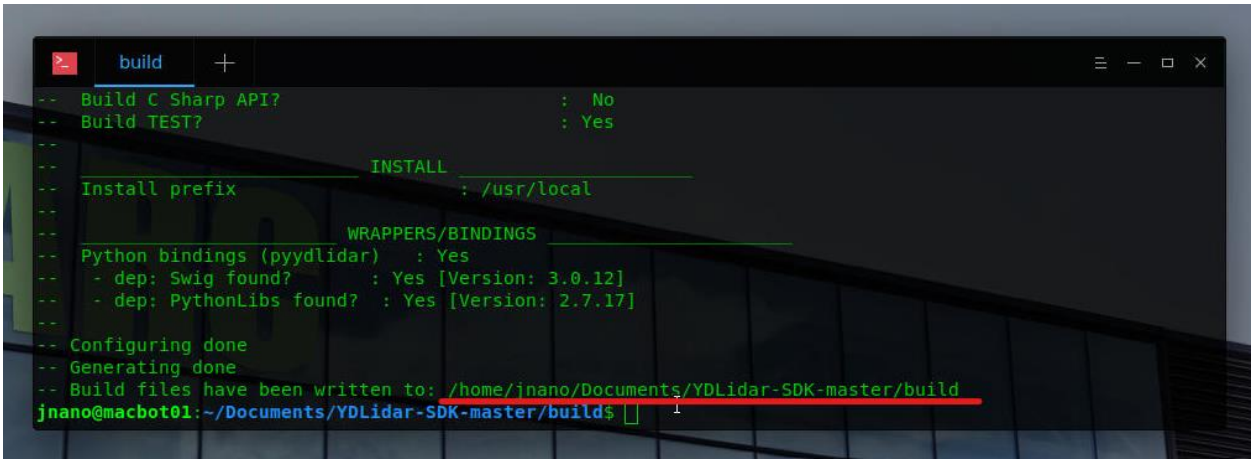


In our case, the **build/** directory is our target and the source is the **YDLidar-SDK-master** directory which is one level above.

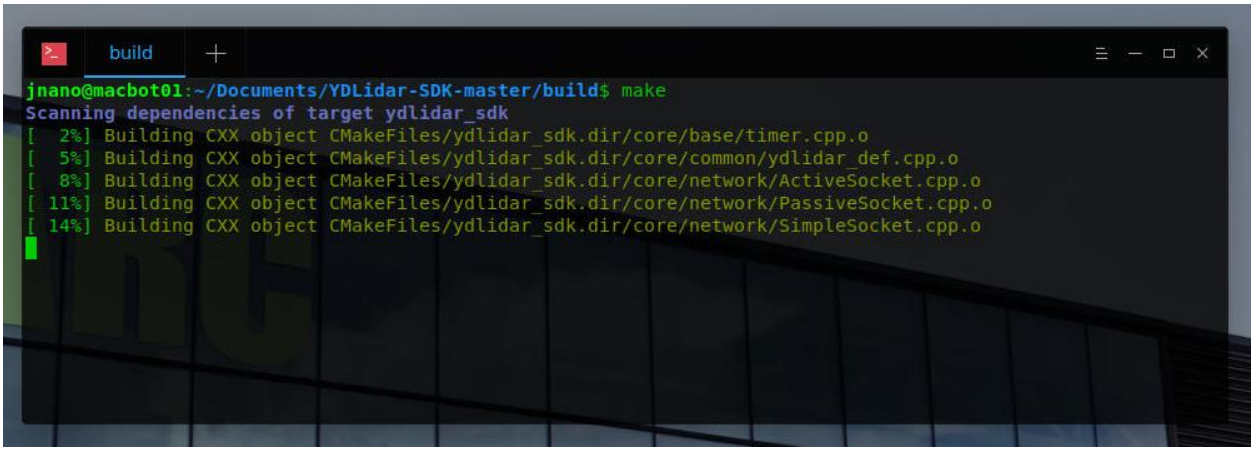
Run **CMake** to build the SDK into the **build/** directory.



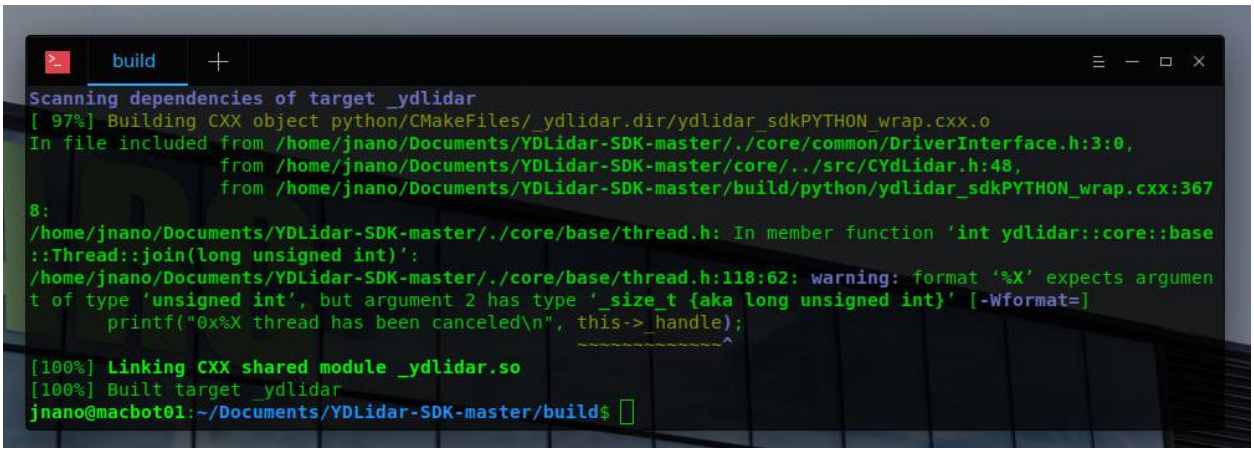
Ensure that build files have been generated correctly.



Next, run the **make** command to compile the project using those generated build files.



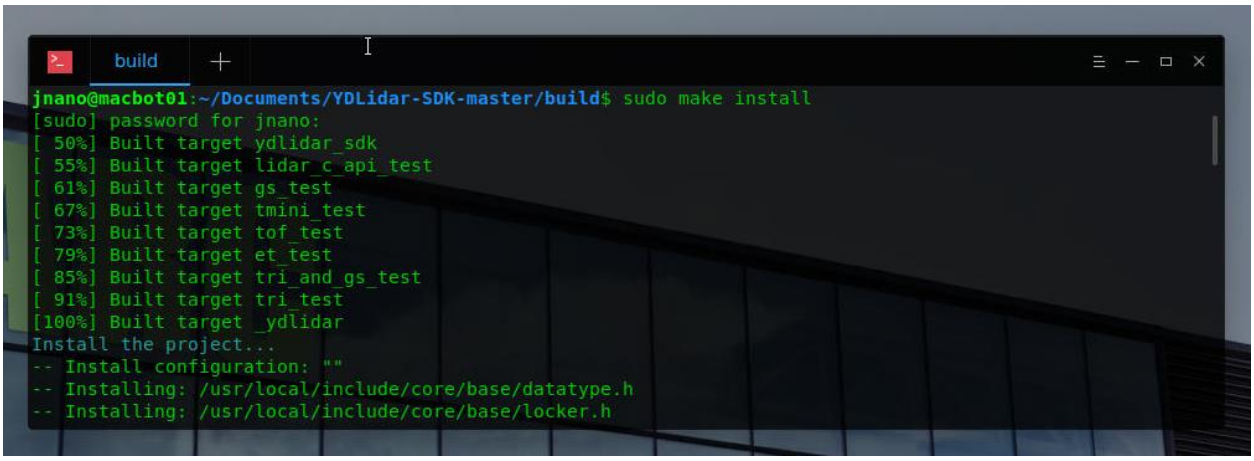
Ensure it builds without fatal errors.



But in our case, we want to use this SDK from our ROS workspace.

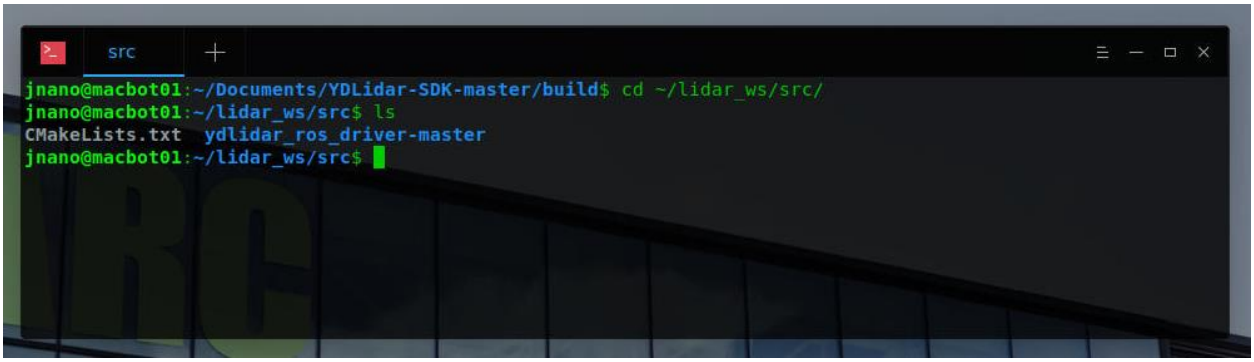
We need to give ROS access to this SDK.

This can be done using the **sudo make install** command, which copies all the binaries to appropriate locations in the operating system, where other programs can access them.

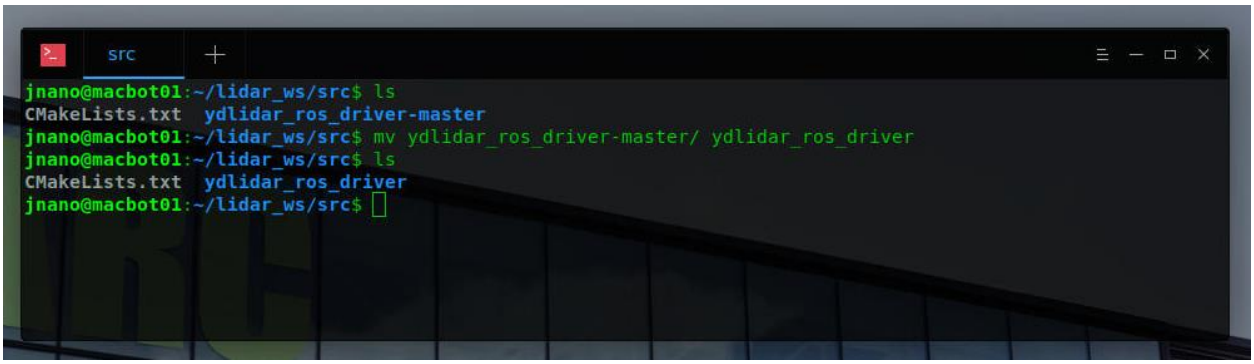


Building the ROS YDLiDAR Package

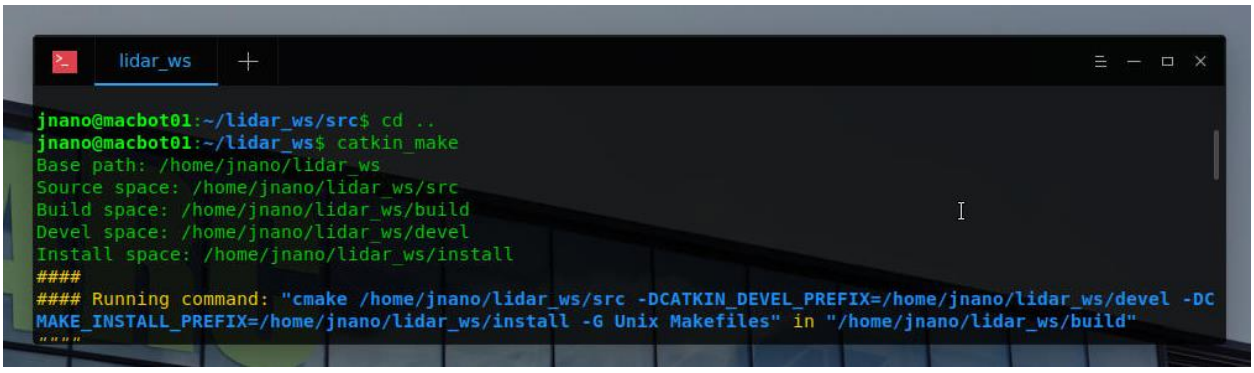
Navigate into your **~/lidar_ws/src** directory.



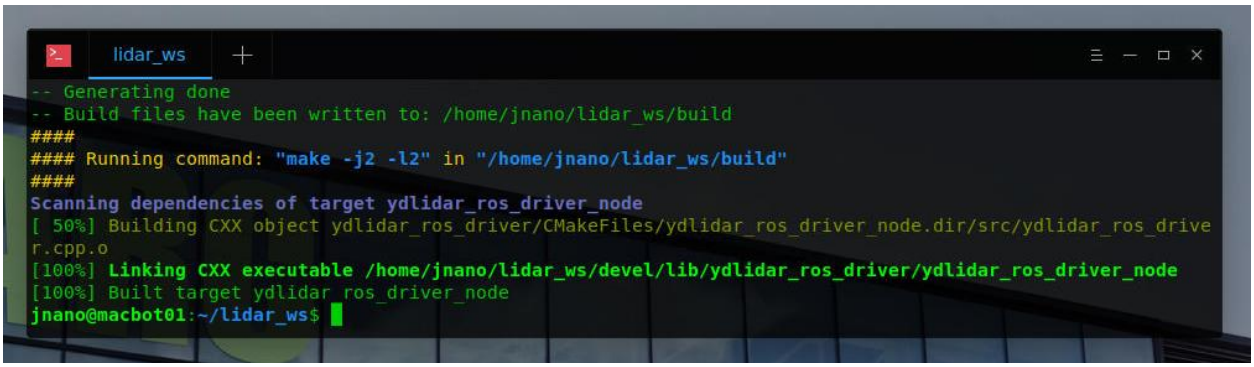
Use the **mv** command to rename the ROS LiDAR driver package to **ydlidar_ros_driver**.



Build the workspace using **catkin_make** from the **~/lidar_ws** directory.



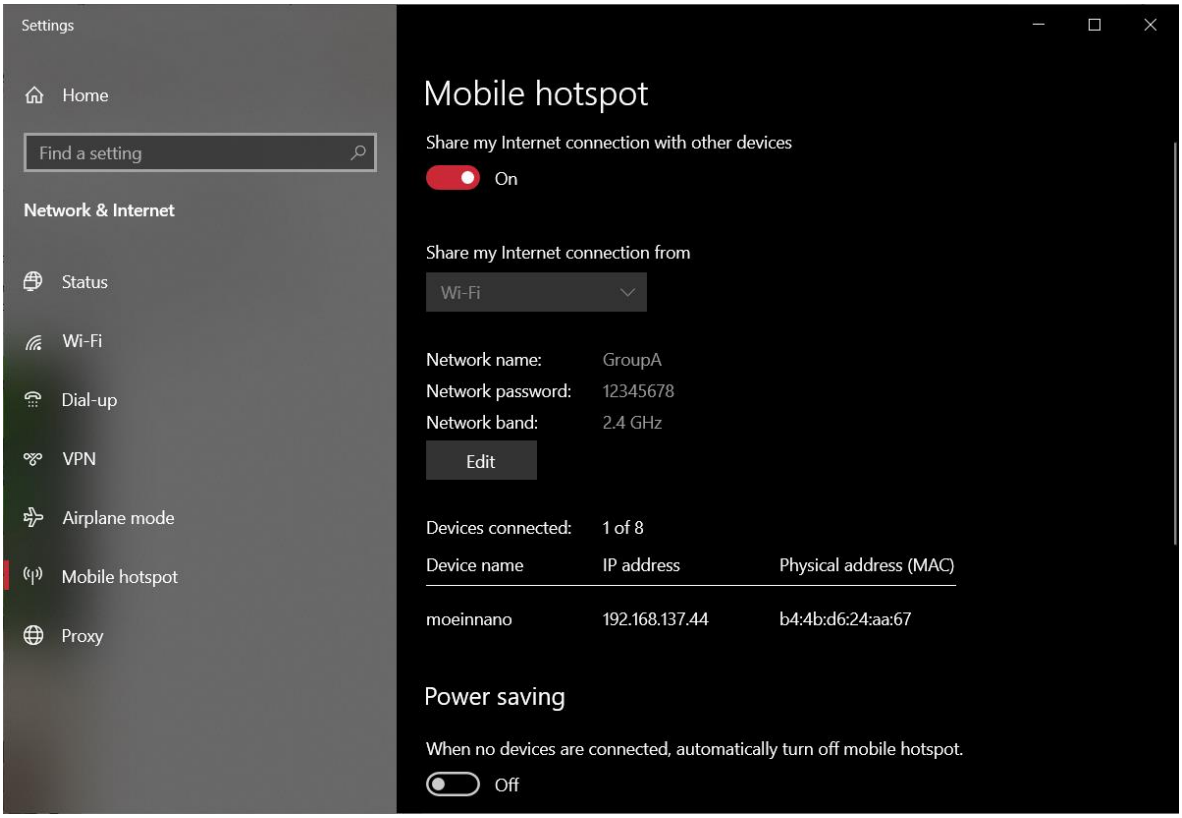
Ensure that the build was successful.


lidar_ws
-- Generating done
-- Build files have been written to: /home/jnano/lidar_ws/build
####
Running command: "make -j2 -l2" in "/home/jnano/lidar_ws/build"
####
Scanning dependencies of target ydlidar_ros_driver_node
[50%] Building CXX object ydlidar_ros_driver/CMakeFiles/ydlidar_ros_driver_node.dir/src/ydlidar_ros_driver.cpp.o
[100%] Linking CXX executable /home/jnano/lidar_ws/devel/lib/ydlidar_ros_driver/ydlidar_ros_driver_node
[100%] Built target ydlidar_ros_driver_node
jnano@macbot01:~/lidar_ws$

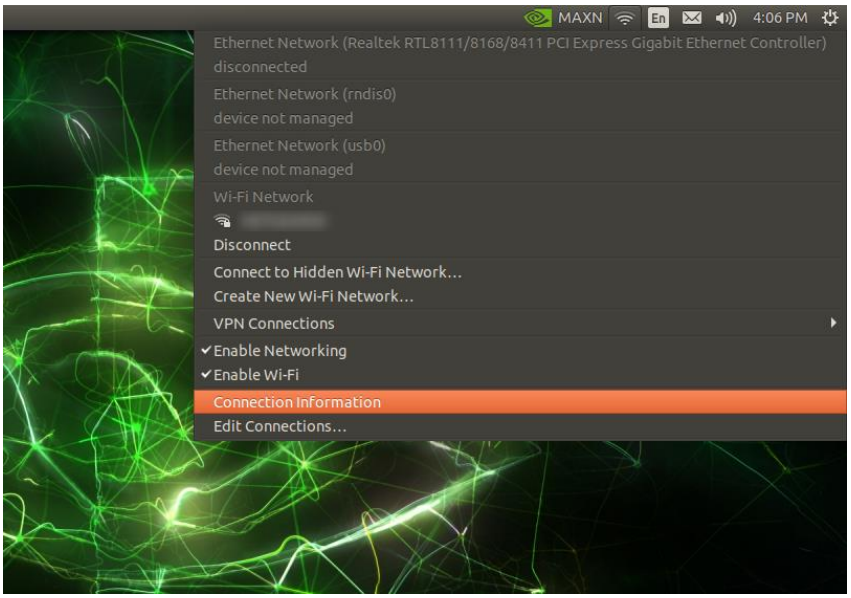
Capturing and Visualizing LiDAR Data

Connecting the YD LiDAR X2

Connecting to a Local Network

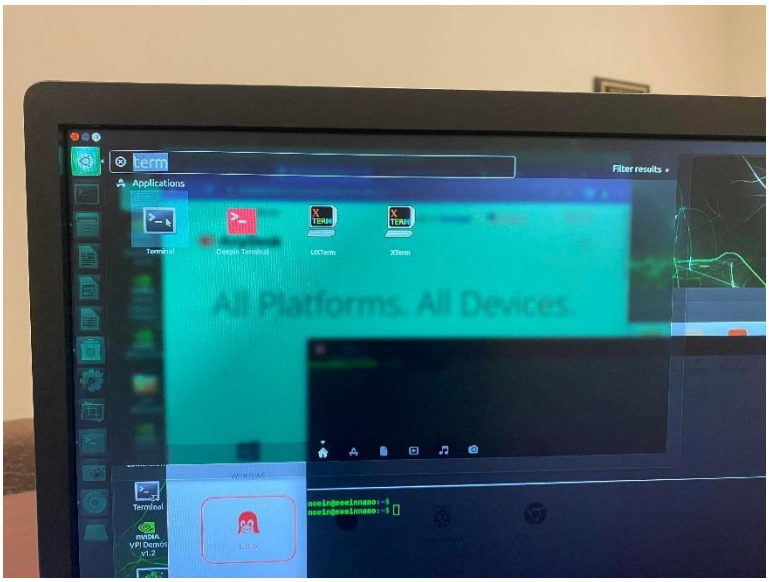


Back in Ubuntu, **select** the hosted WiFi network.



Updating Packages

Using the launch menu, open the **Terminal**.



Run the following command:

sudo apt update

```
moein@moeinnano:~$ sudo apt update
Get:1 file:/var/cuda-repo-l4t-10-2-local  InRelease
Ign:1 file:/var/cuda-repo-l4t-10-2-local  InRelease
Get:2 file:/var/visionworks-repo  InRelease
Ign:2 file:/var/visionworks-repo  InRelease
Get:3 file:/var/visionworks-sfm-repo  InRelease
Ign:3 file:/var/visionworks-sfm-repo  InRelease
Get:4 file:/var/visionworks-tracking-repo  InRelease
Ign:4 file:/var/visionworks-tracking-repo  InRelease
Get:5 file:/var/cuda-repo-l4t-10-2-local  Release [564 B]
Get:6 file:/var/visionworks-repo  Release [2,001 B]
Get:7 file:/var/visionworks-sfm-repo  Release [2,005 B]
Get:5 file:/var/cuda-repo-l4t-10-2-local  Release [564 B]
Get:8 file:/var/visionworks-tracking-repo  Release [2,010 B]
Get:6 file:/var/visionworks-repo  Release [2,001 B]
Get:7 file:/var/visionworks-sfm-repo  Release [2,005 B]
Get:8 file:/var/visionworks-tracking-repo  Release [2,010 B]
Hit:9 http://packages.ros.org/ros/ubuntu bionic InRelease
Hit:10 http://ports.ubuntu.com/ubuntu-ports bionic InRelease
Hit:11 http://packages.ros.org/ros2/ubuntu bionic InRelease
Hit:12 http://deb.anydesk.com all InRelease
Err:14 http://realsense-hw-public.s3.amazonaws.com/Debian/apt-repo bionic InRele
ase
```

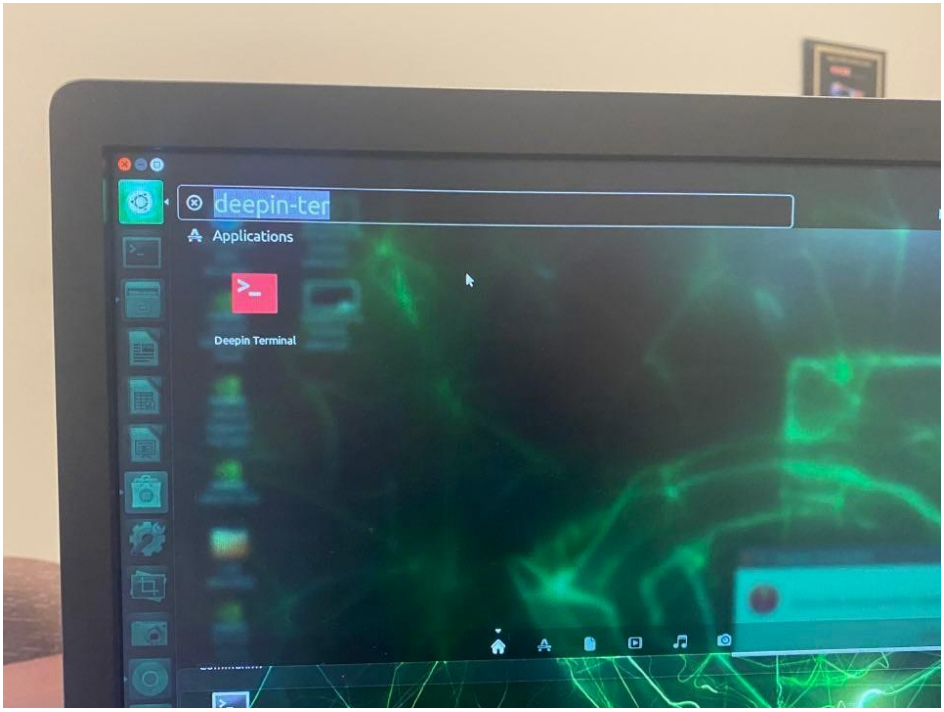
Installing Deepin-Terminal

When working with ROS, it is useful to be able to split a terminal window into multiple horizontal and vertical windows or create new terminal tabs. This is because ROS requires many scripts to run concurrently and navigating between many open windows becomes tedious and difficult to navigate.

sudo apt install deepin-terminal

```
moein@moeinnano:~$ sudo apt install deepin-terminal
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  deepin-terminal
0 upgraded, 1 newly installed, 0 to remove and 88 not upgraded.
Need to get 2,172 kB of archives.
After this operation, 4,901 kB of additional disk space will be used.
Get:1 http://ports.ubuntu.com/ubuntu-ports bionic/universe arm64 deepin-terminal
 arm64 2.9.2-1 [2,172 kB]
Fetched 2,172 kB in 1s (1,865 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package deepin-terminal.
(Reading database ... 213287 files and directories currently installed.)
Preparing to unpack .../deepin-terminal_2.9.2-1_arm64.deb ...
Unpacking deepin-terminal (2.9.2-1) ...
Setting up deepin-terminal (2.9.2-1) ...
Processing triggers for desktop-file-utils (0.23-1ubuntu3.18.04.2) ...
Processing triggers for bamfdaemon (0.5.3+18.04.20180207.2-0ubuntu1) ...
Rebuilding /usr/share/applications/bamf-2.index...

Progress: [ 83%] [##################################################.....]
```



To split deepin-terminal, **right-click** and select either **Horizontal split** or **Vertical** split. To create a new terminal tab, **click** on the **+** new tab button.

ROS Setup

Checking if ROS is Already Installed

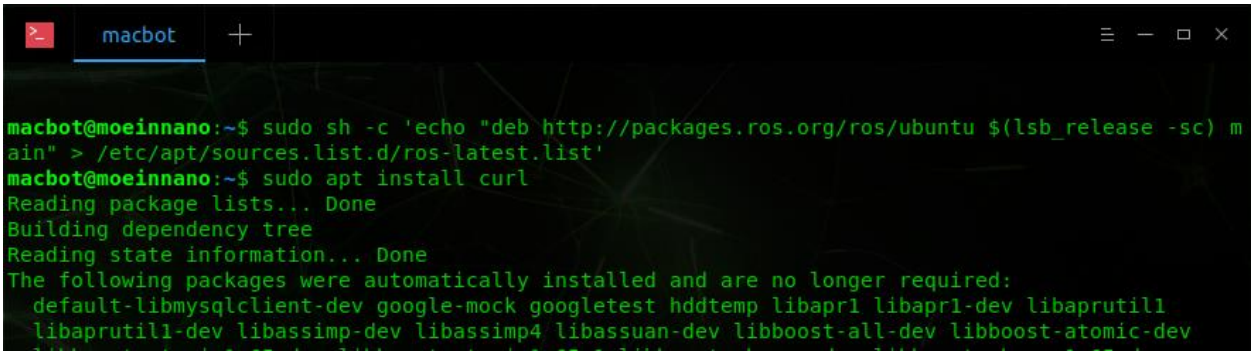
rosversion -d

If an error prints, ROS needs to be installed. Please continue with the installation instructions.

Installing ROS Melodic

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

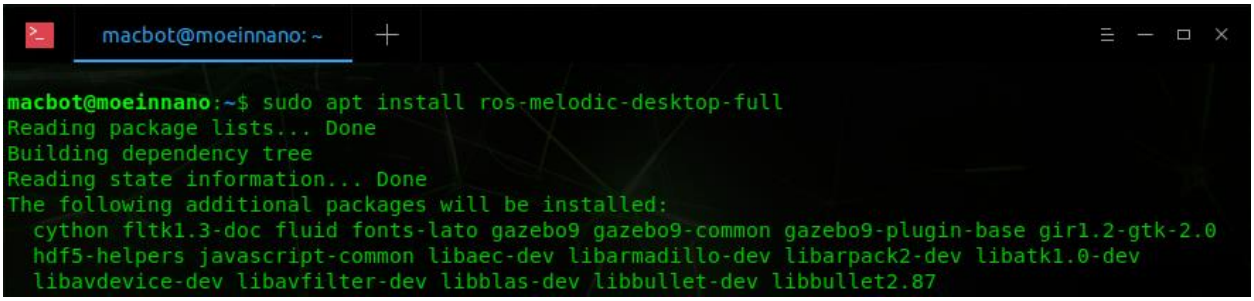
sudo apt install curl



curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add –

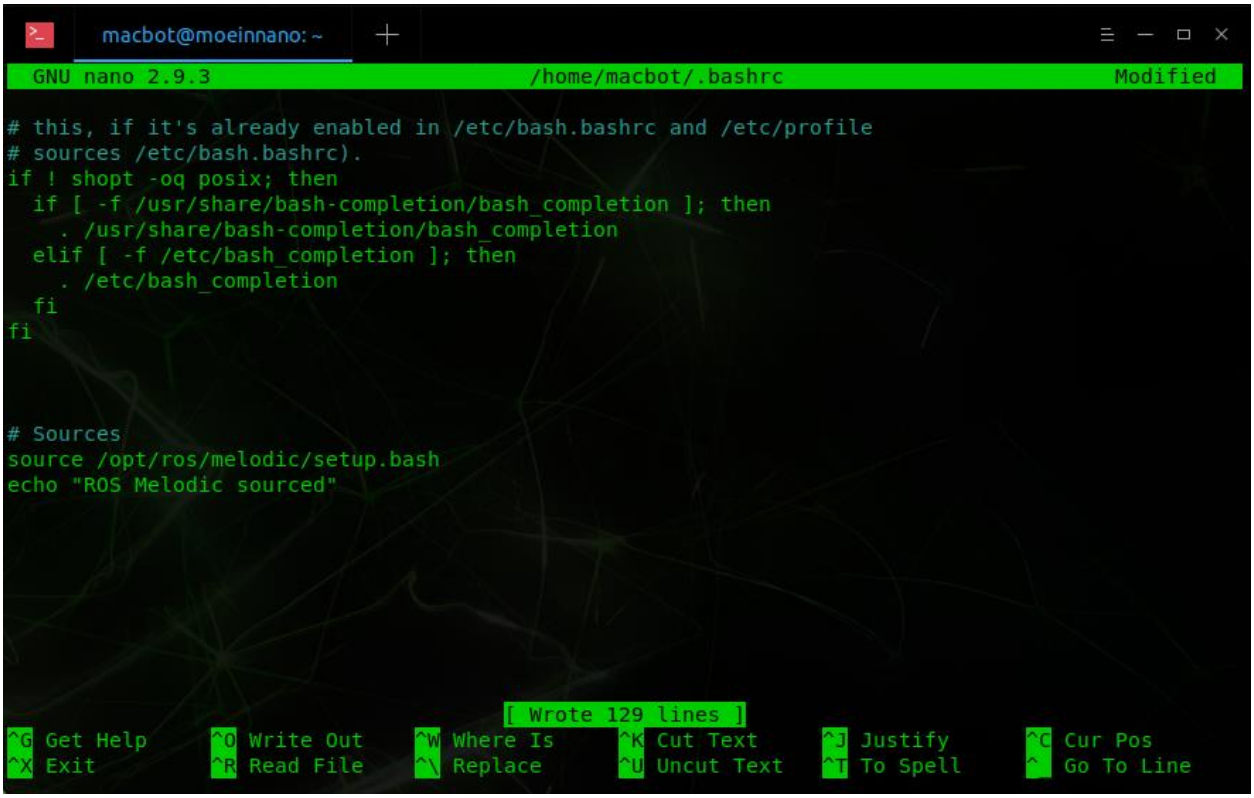
sudo apt update

sudo apt install ros-melodic-desktop-full



Now, in order to have access to ROS commands without needing to source the installation each time a terminal window is opened, we can add it to the **~/.bashrc** file.

sudo nano ~/.bashrc



Add the source for ROS melodic.

source /opt/ros/melodic/setup.bash

To exit nano, use the following commands:

CTRL + s

CTRL + x

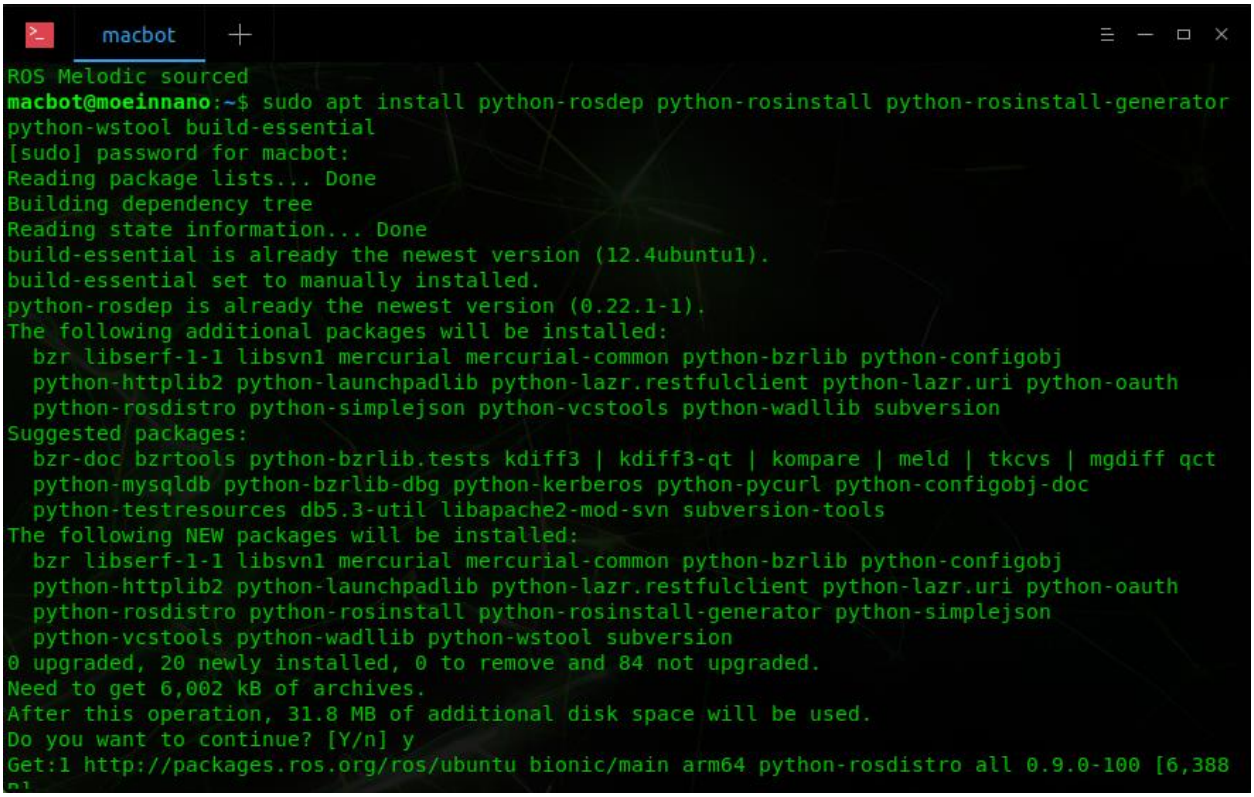
Y

<Enter>

Open a new terminal window and see if any errors occur. Is an error does occur, ensure that the correct path was added to the bashrc file.

Next, install useful dependencies for ROS Melodic.

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

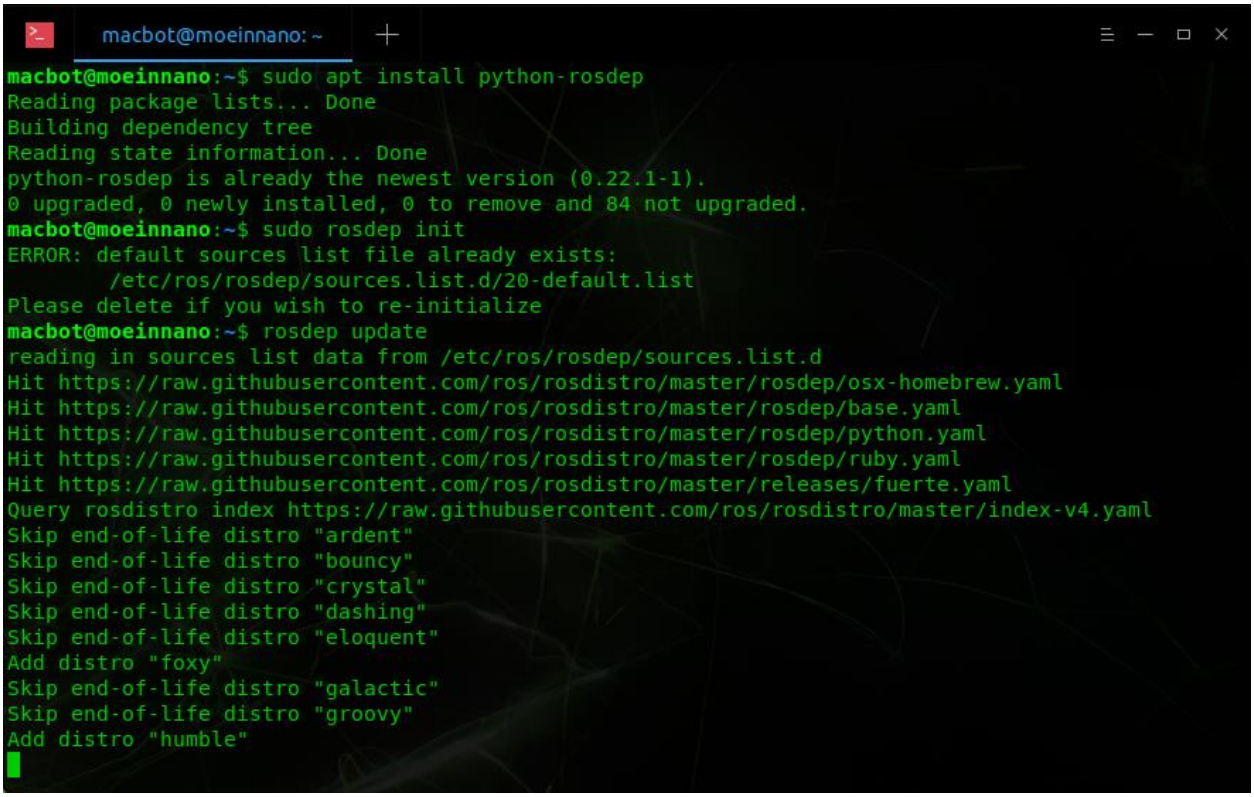


Lastly, initialize rosdep.

sudo apt install python-rosdep

sudo rosdep init

rosdep update



Creating a ROS Catkin Workspace

Create a new directory in your **home/** folder and initialise it as a catkin workspace.

cd ~/

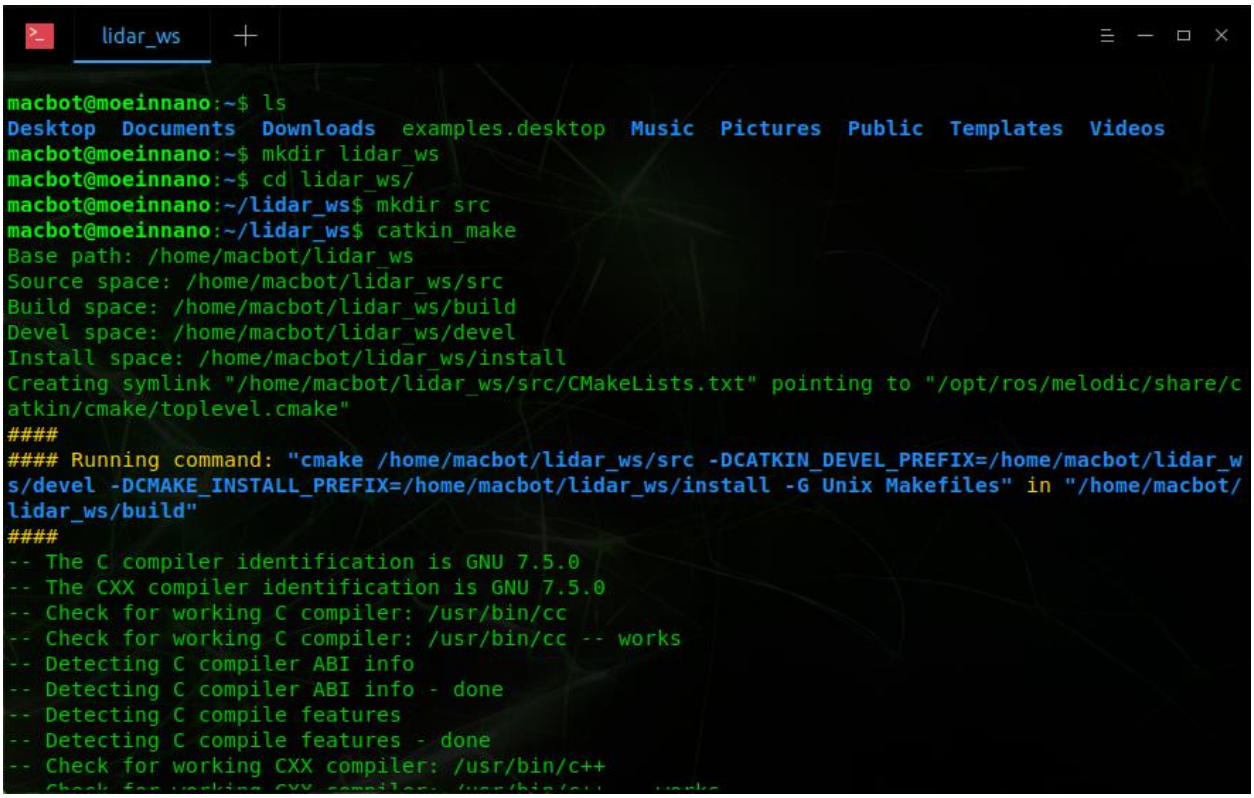
ls

mkdir lidar_ws

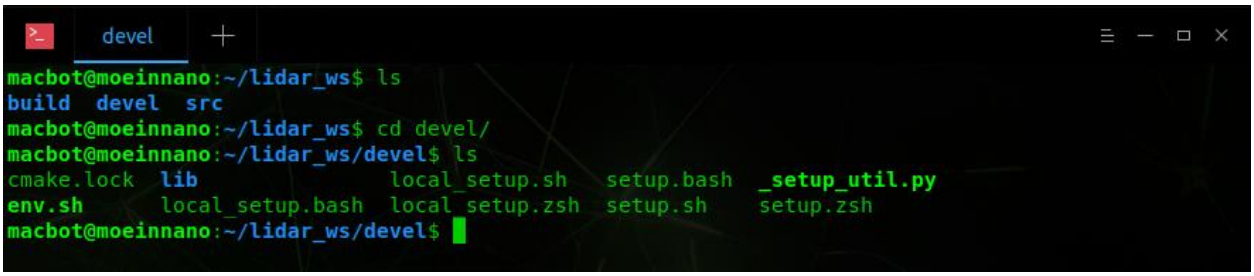
cd lidar_ws

mkdir src

catkin_make



If you look inside the generated **devel/** folder, you will find a **setup.bash** file. This file must be used to source the director.



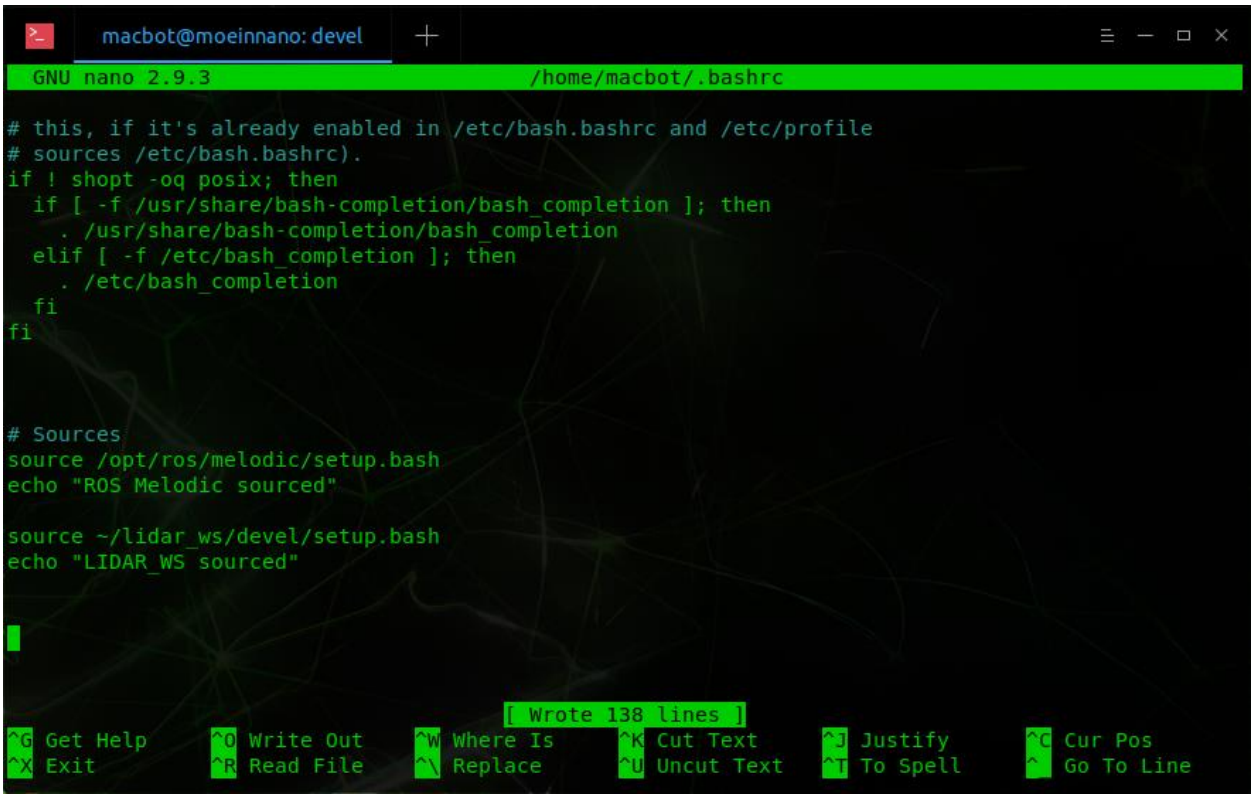
Open the ~/.**bashrc** folder, scroll to the bottom and source this file.

sudo nano ~/.bashrc

Add the following lines:

source ~/lidar_ws/devel/setup.bash

echo "LIDAR_WS sourced"



Save the file using the following commands:

CTRL + s

CTRL + x

Y

<Enter>

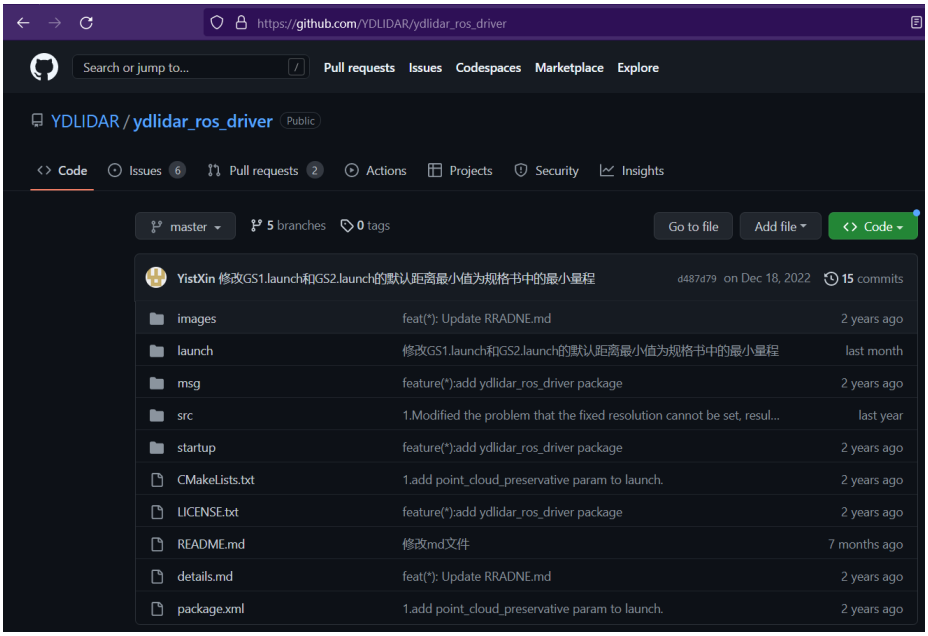
Open a new terminal window. Ensure that everything sources without any errors.

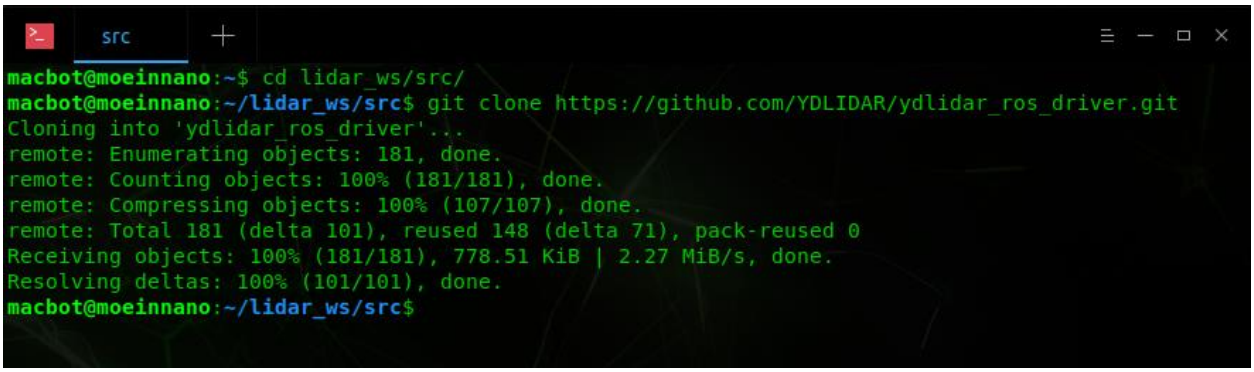
Lidar Setup

Cloning the YDLidar ROS Package

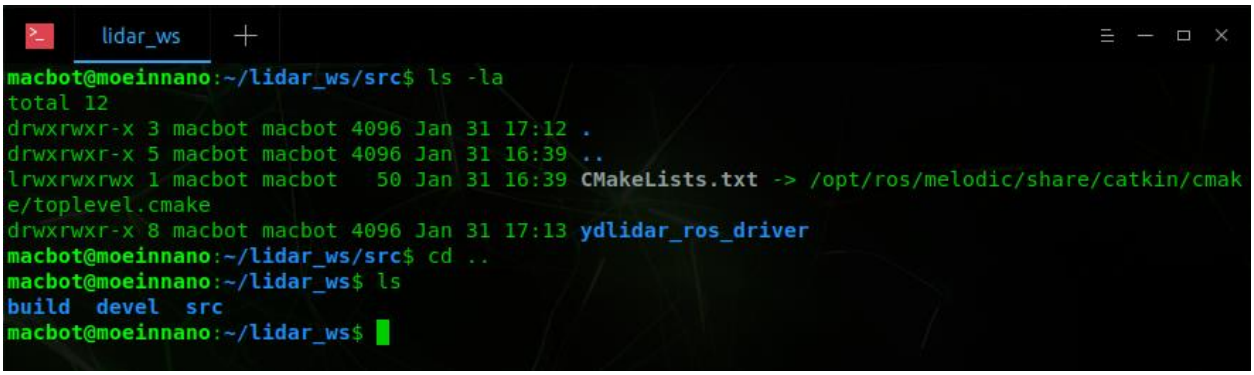
cd ~/lidar_ws/src

git clone https://github.com/YDLIDAR/ydlidar_ros_driver.git

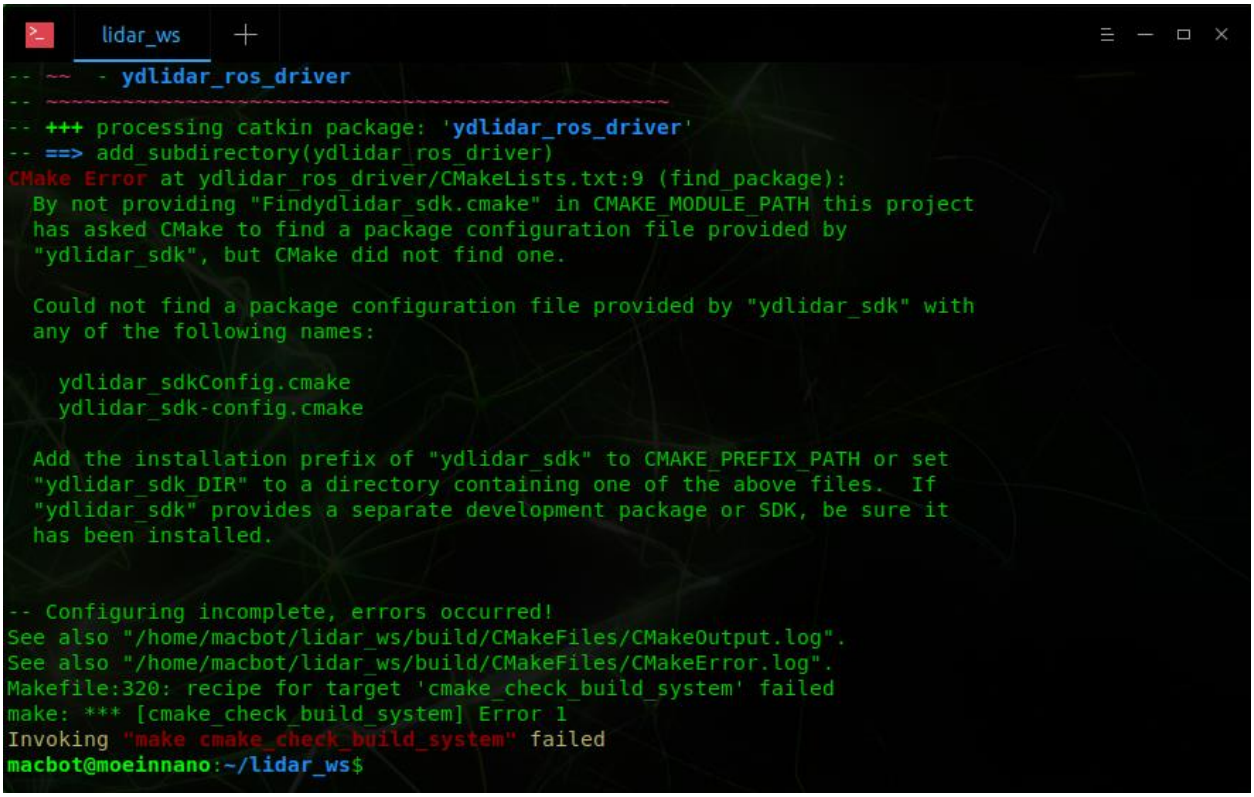




Navigate back to **~/lidar_ws.**



You will notice that if you build this project now, it will state that dependencies are missing. In order to resolve this, we must download and build the YDLiDAR SDK package.

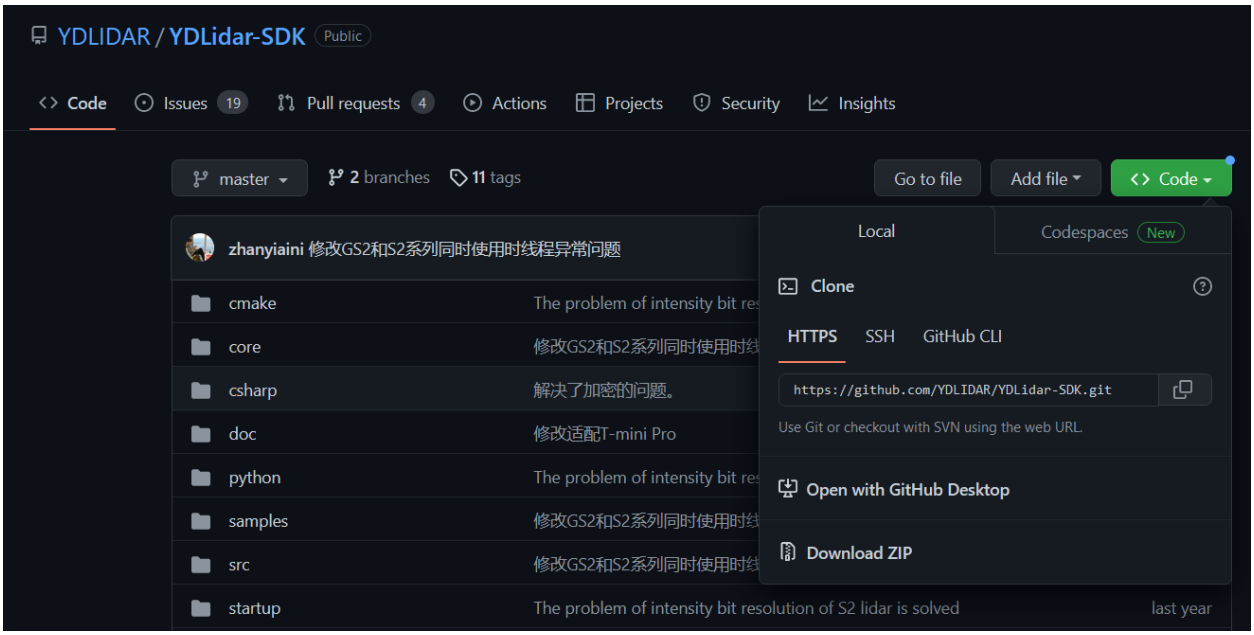


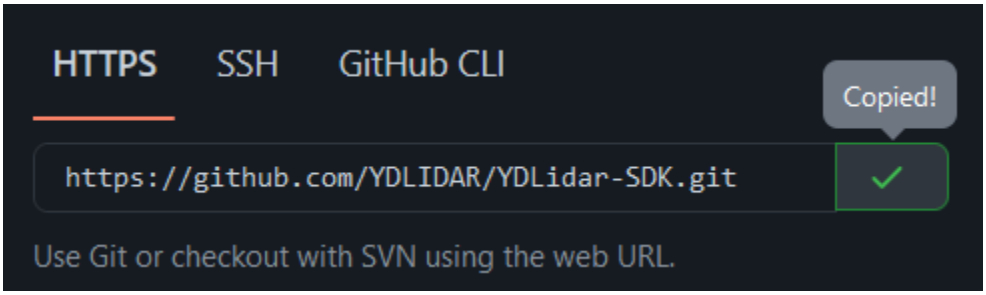
YDLiDAR SDK

Navigate to the following GitHub repo:

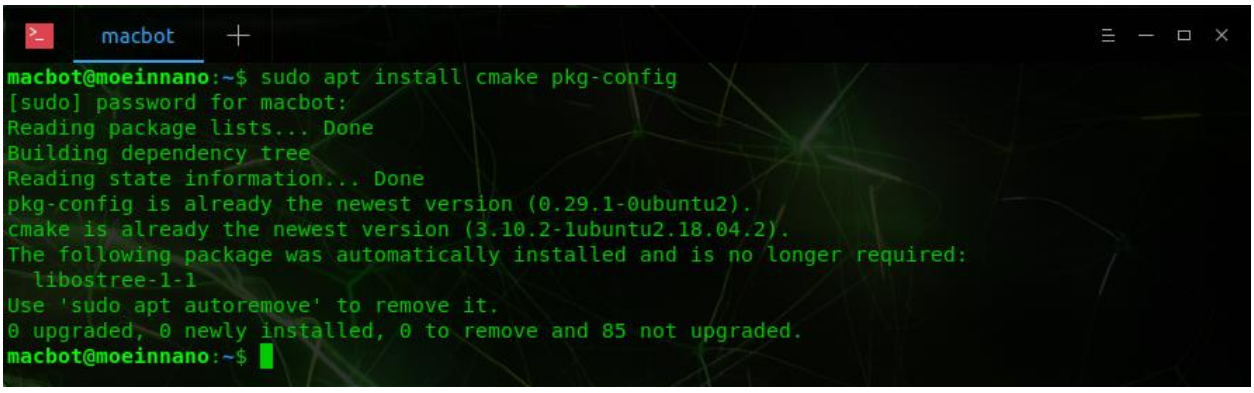
https://github.com/YDLIDAR/YDLidar-SDK

Copy its git URL.



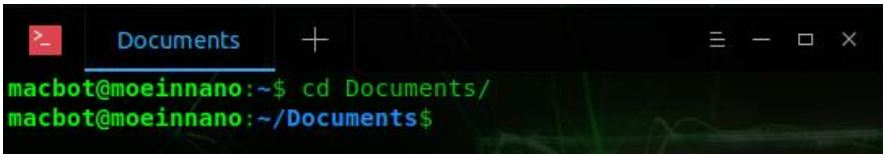


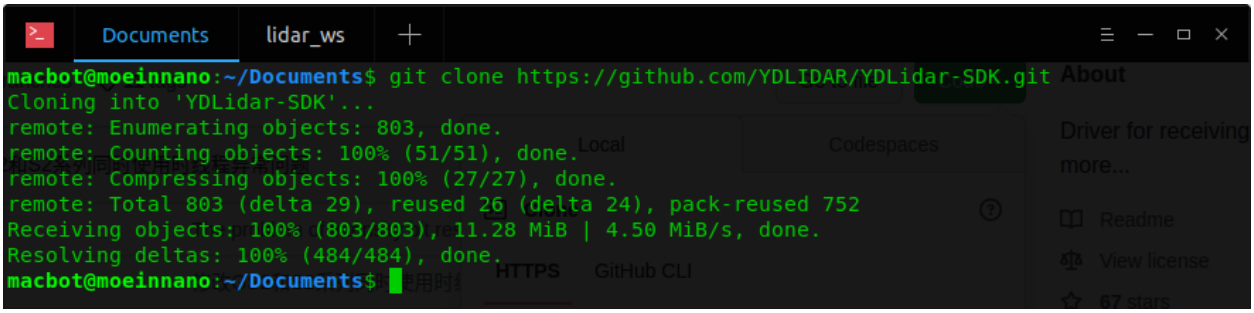
In order to build a C++ project, you will need to ensure cmake is installed.



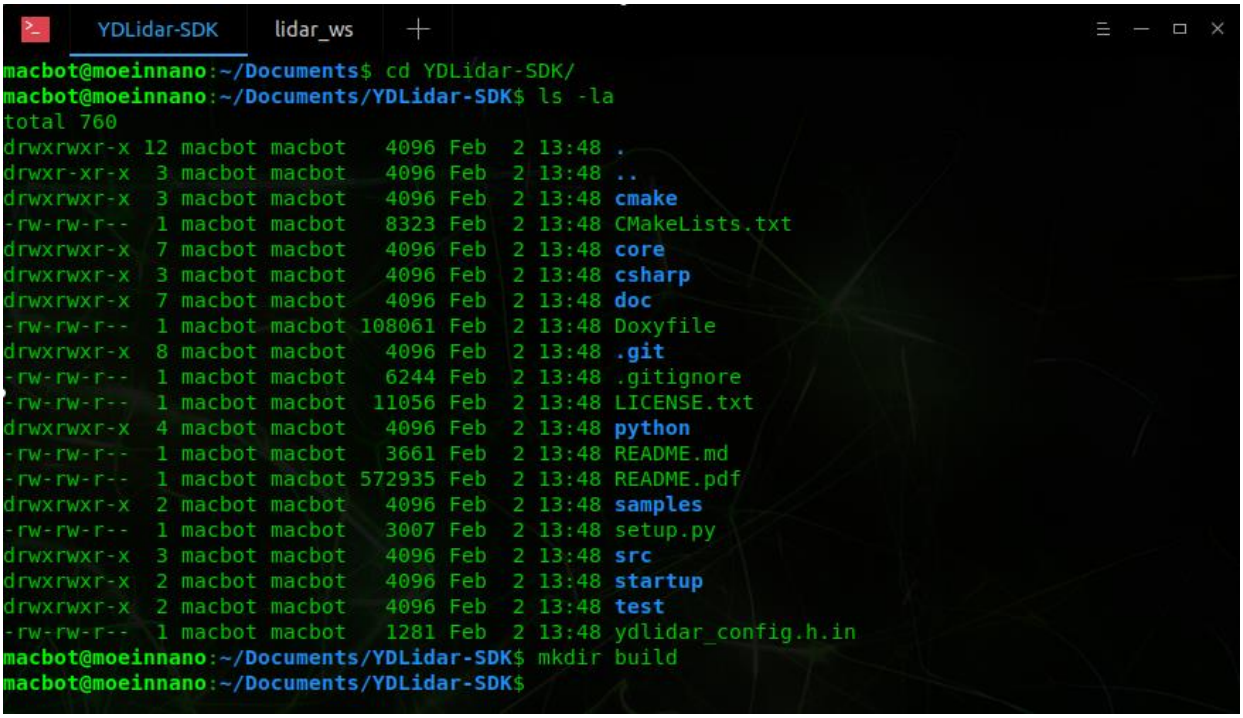
sudo apt install cmake pkg-config

Clone the repo into your **~/Documents** directory.

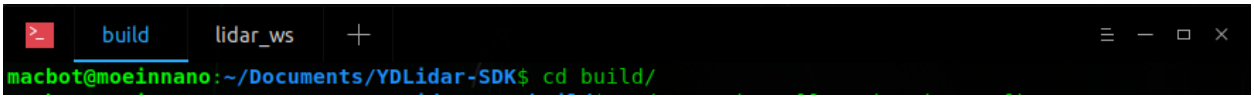




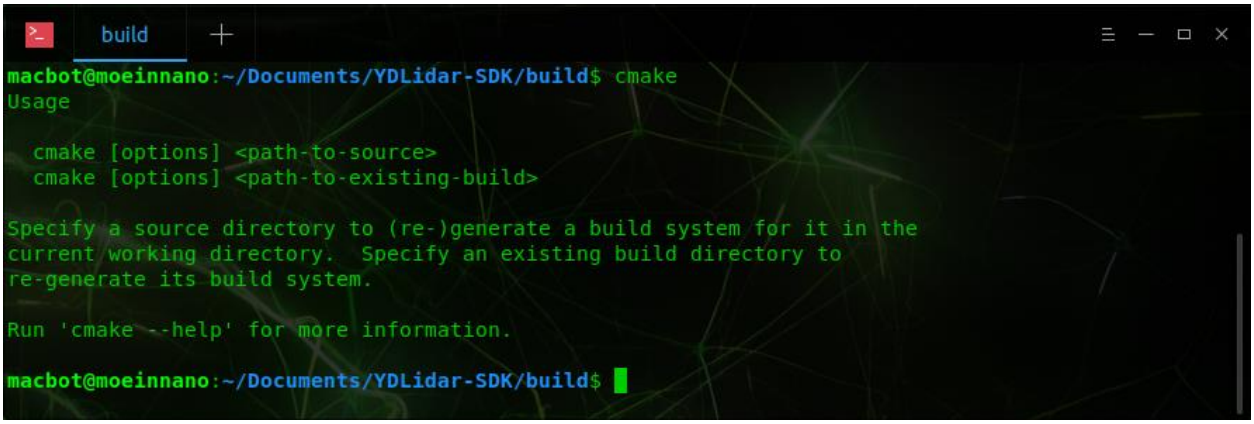
Navigate into the cloned repo and create a new folder called **build/**.



Navigate into the **build/** directory.

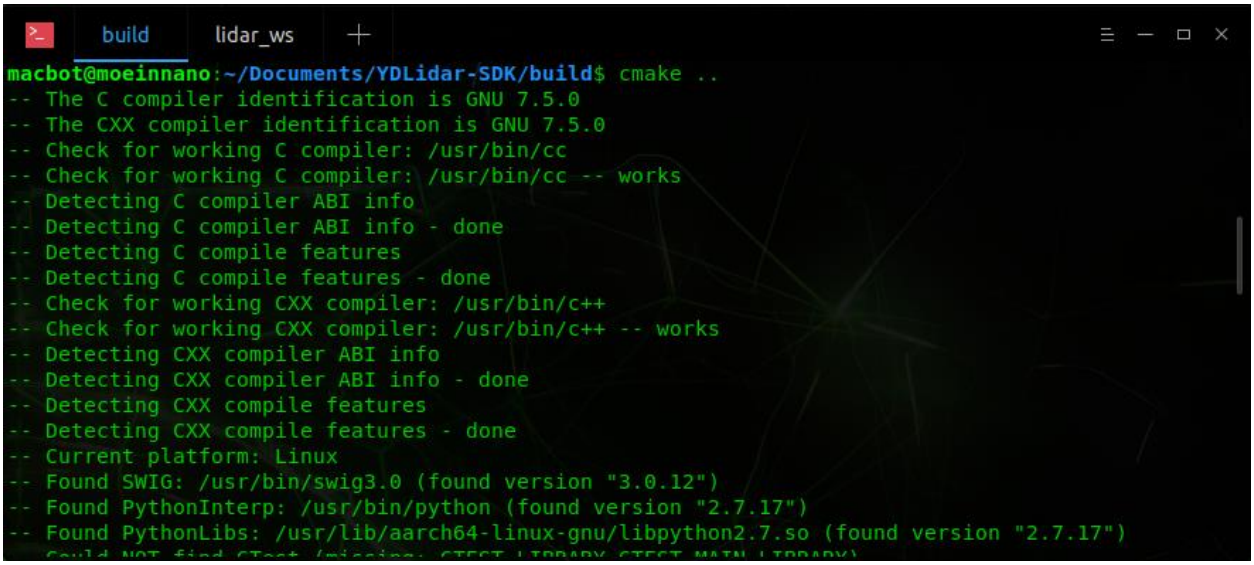


If you run **cmake** without any command line arguments, it will give your more information on what the program requires to prepare the build for a project.

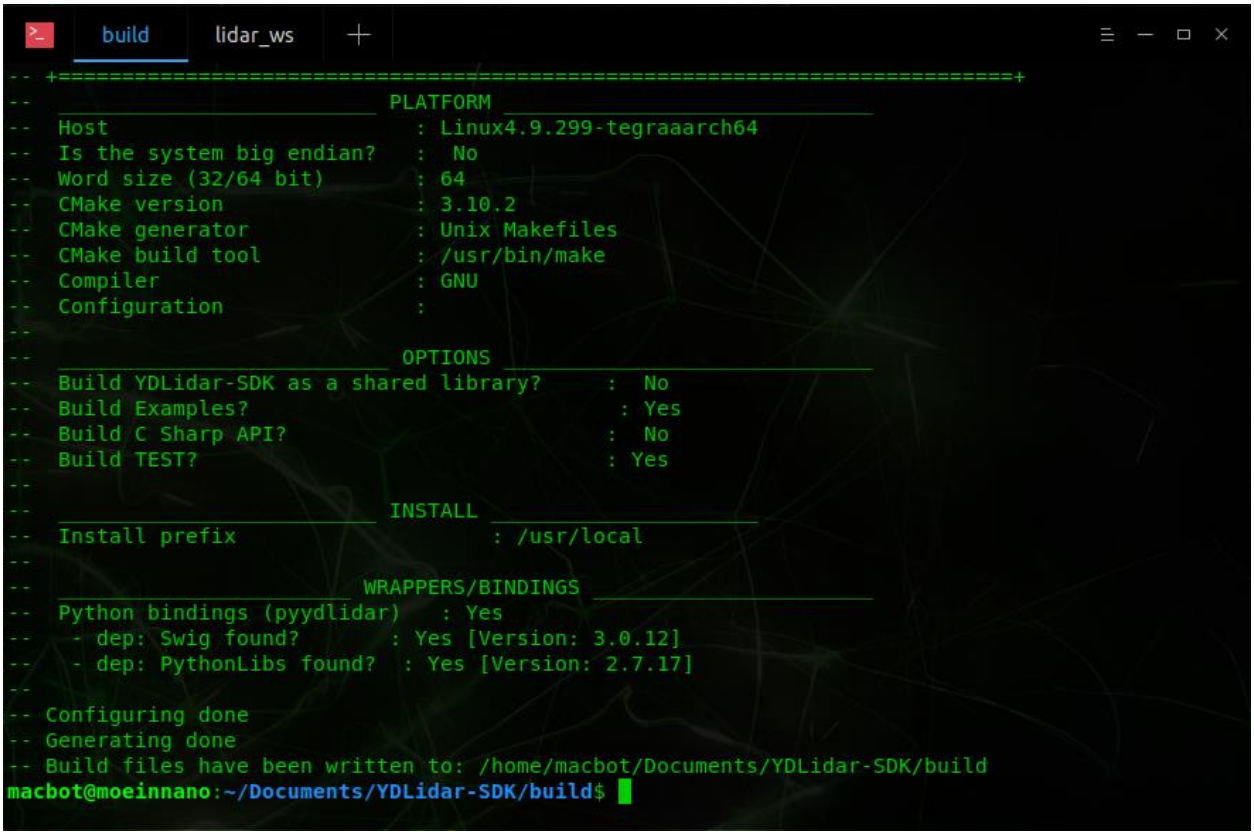


In our case, the **build/** directory is our target and the source is the **YDLidar-SDK** directory which is one level above.

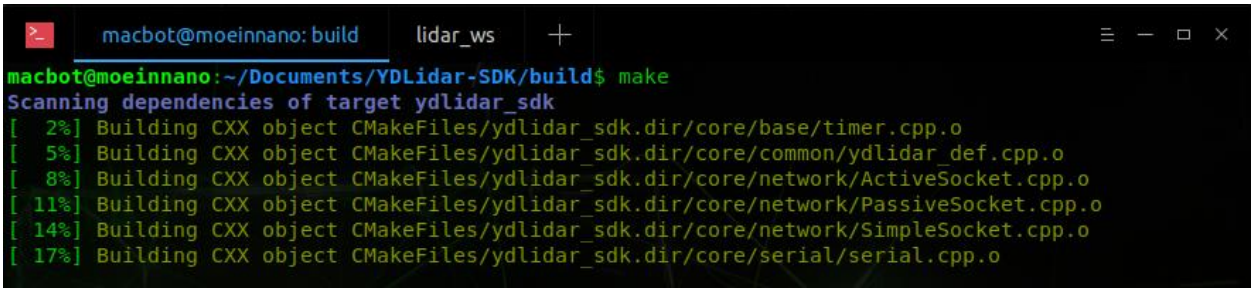
Run CMake to build the SDK into the build/ directory.



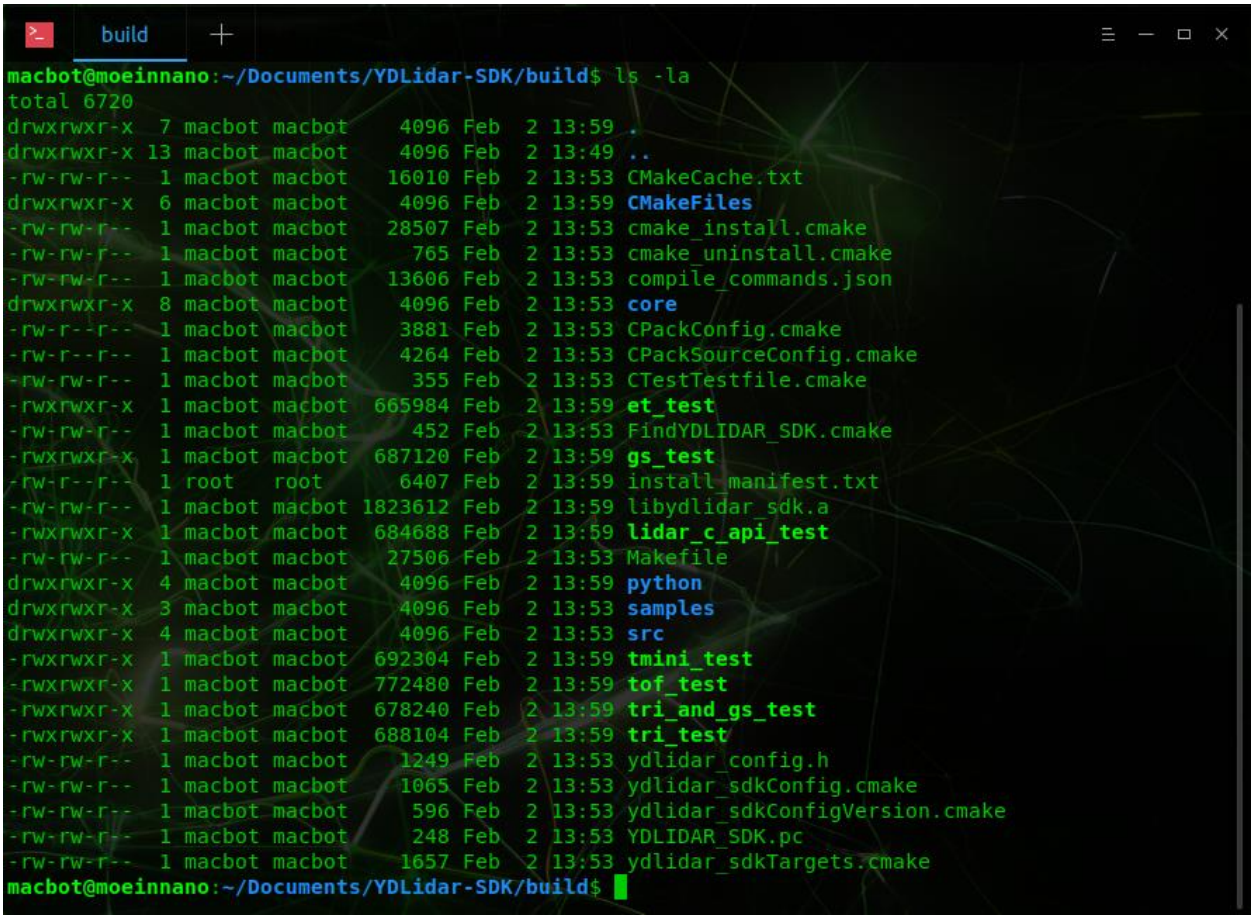
If it is able to traverse the project correctly, it should print an overview of the generated build configuration.



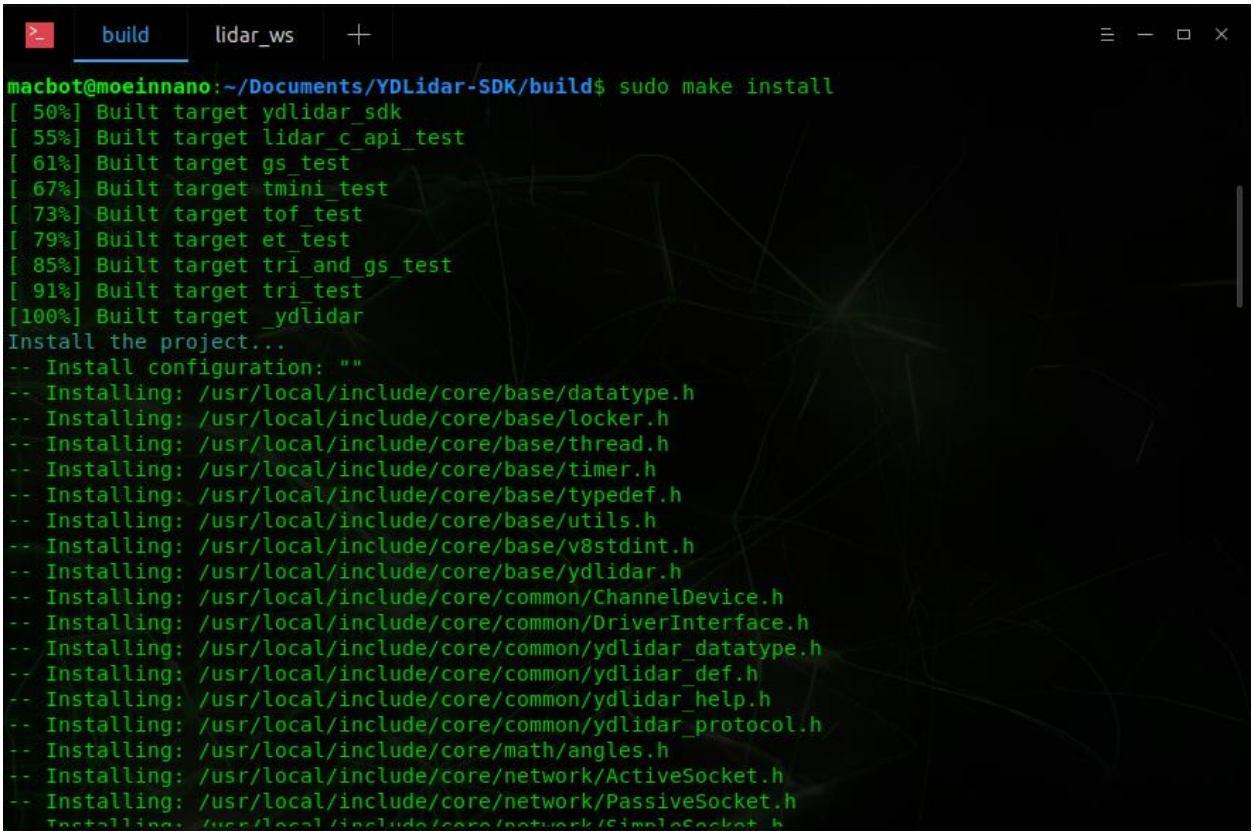
Next, **make** the SDK while being in the **build/** directory.



If the build is successful, many files should be generated including some test examples for working with the SDK.



But in our case, we want to use this SDK from our ROS system using the **ydlidar_ros_driver**. We need to give that ROS package access to this SDK. This can be done using the **make install** command, which copies all the binaries to appropriate locations in the operating system, where other programs can access them.

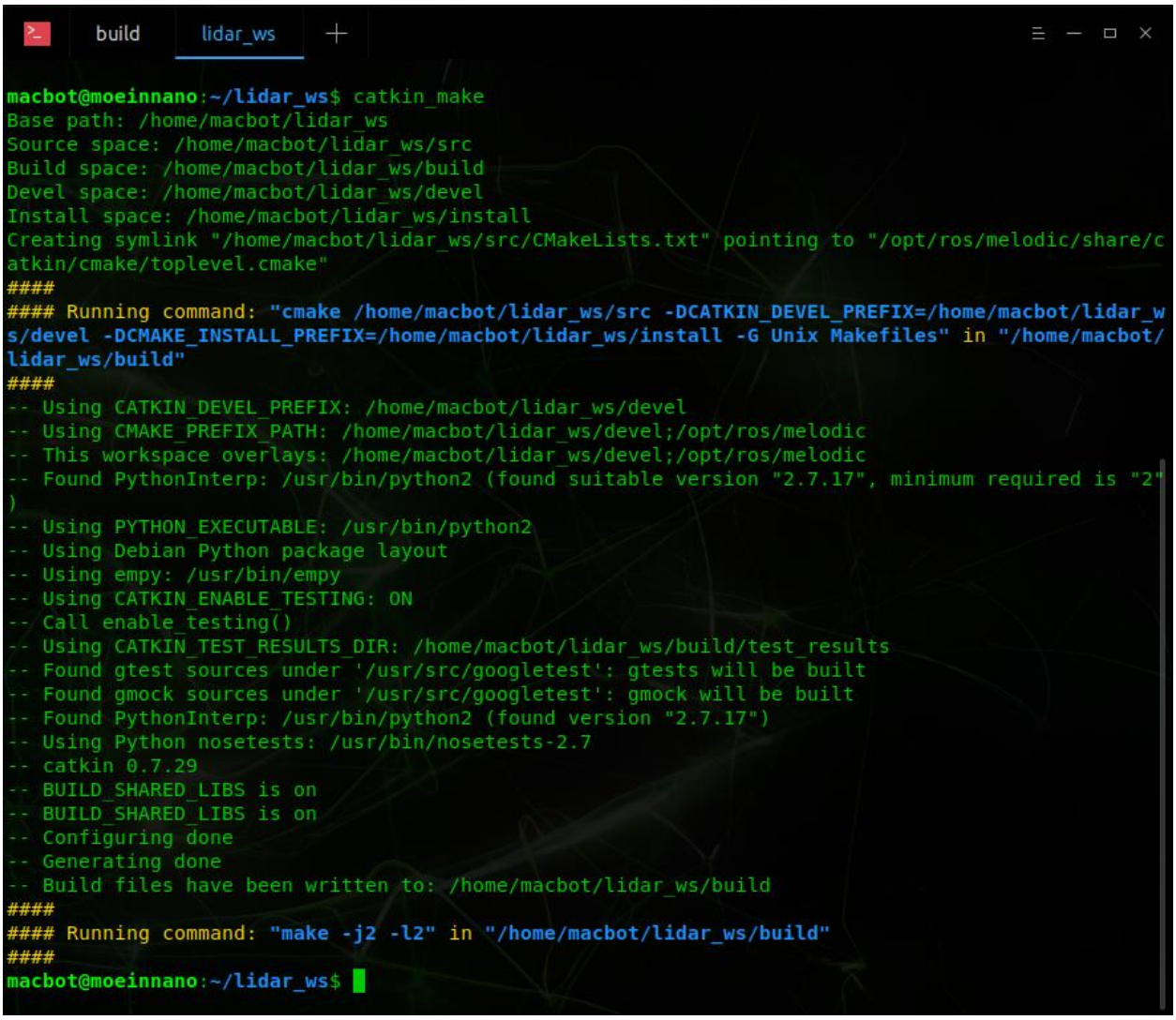


Once the SDK has been installed correctly, navigate back to your ROS workspace.

Running the YDLidar ROS Package

Now that the SDK has been installed, the ROS package should build correctly.

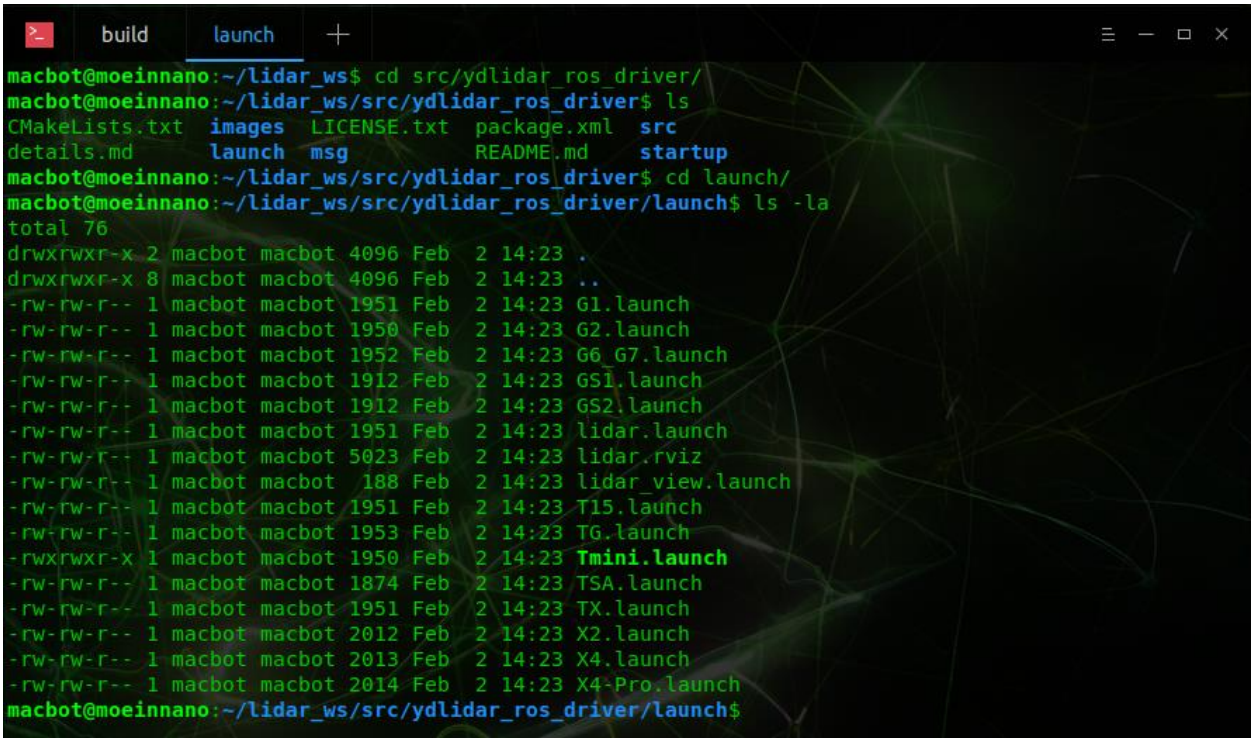
Navigate into the root of the **lidar_ws workspace** and run **catkin_make**.



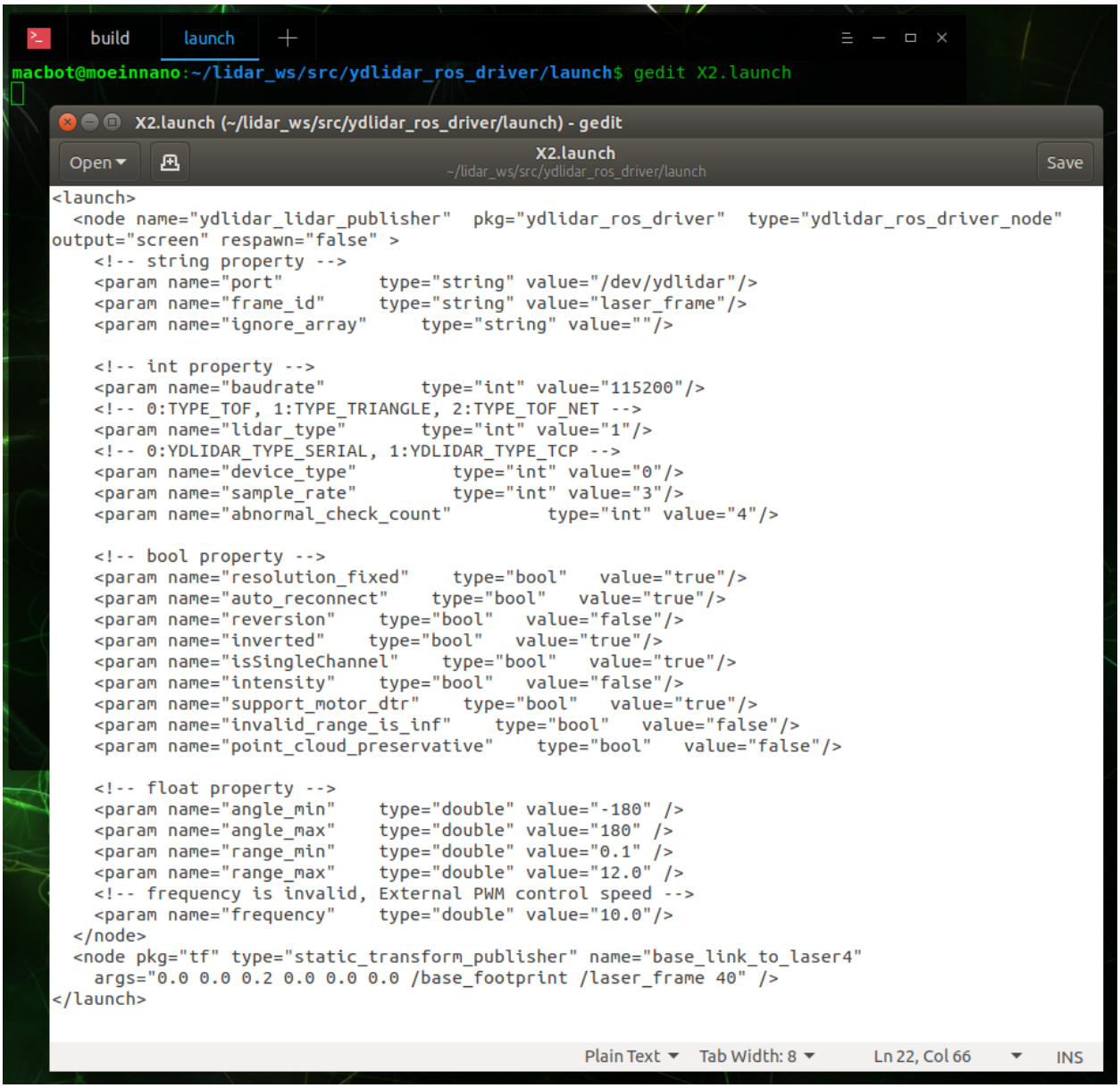
Launch Files

ROS projects can become very sophisticated, having many nodes communicating concurrently. A way to automate the startup of a program is to use a launch file.

Navigate to the **launch/** directory of the built **ydlidar_ros_driver** ROS package.

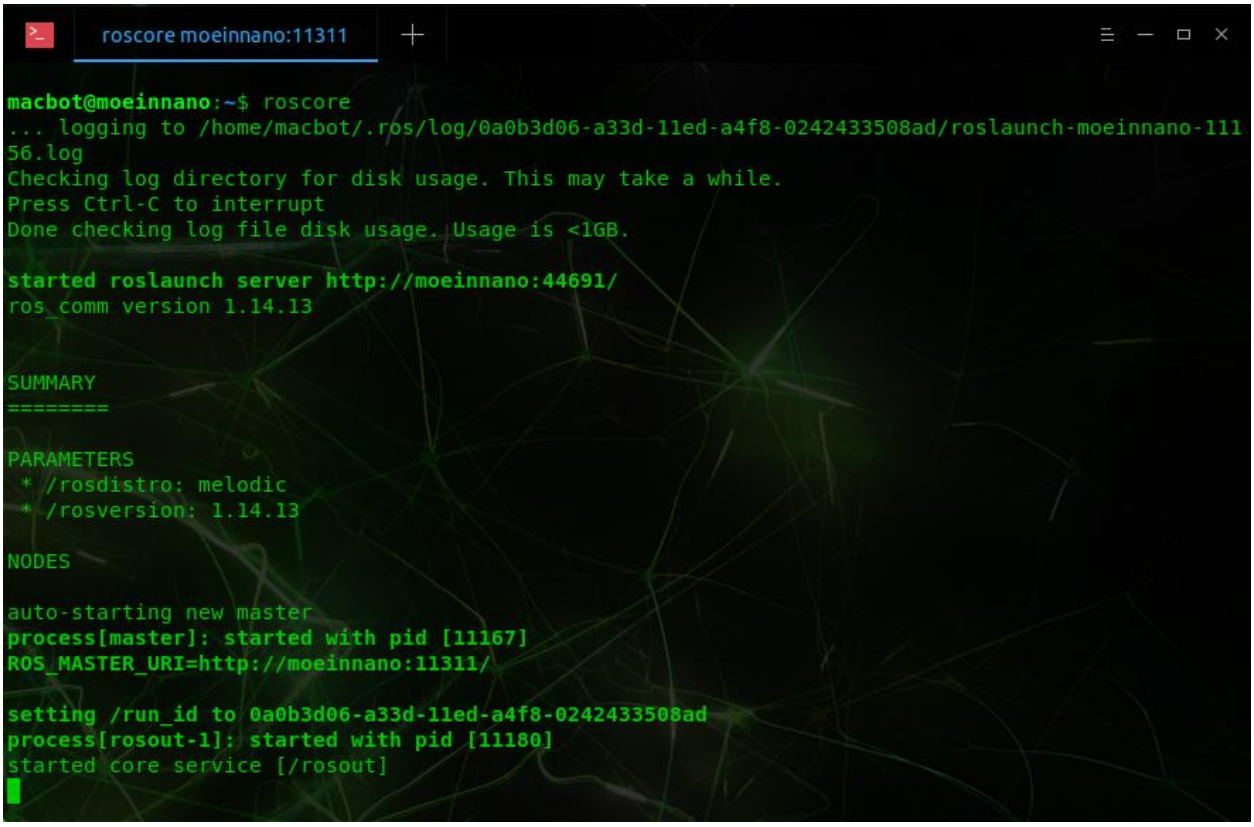


Lets open the **X2.launch** file in a text editor to view how it works. Use **nano** to view in-terminal or **gedit** to view in a graphical editor.



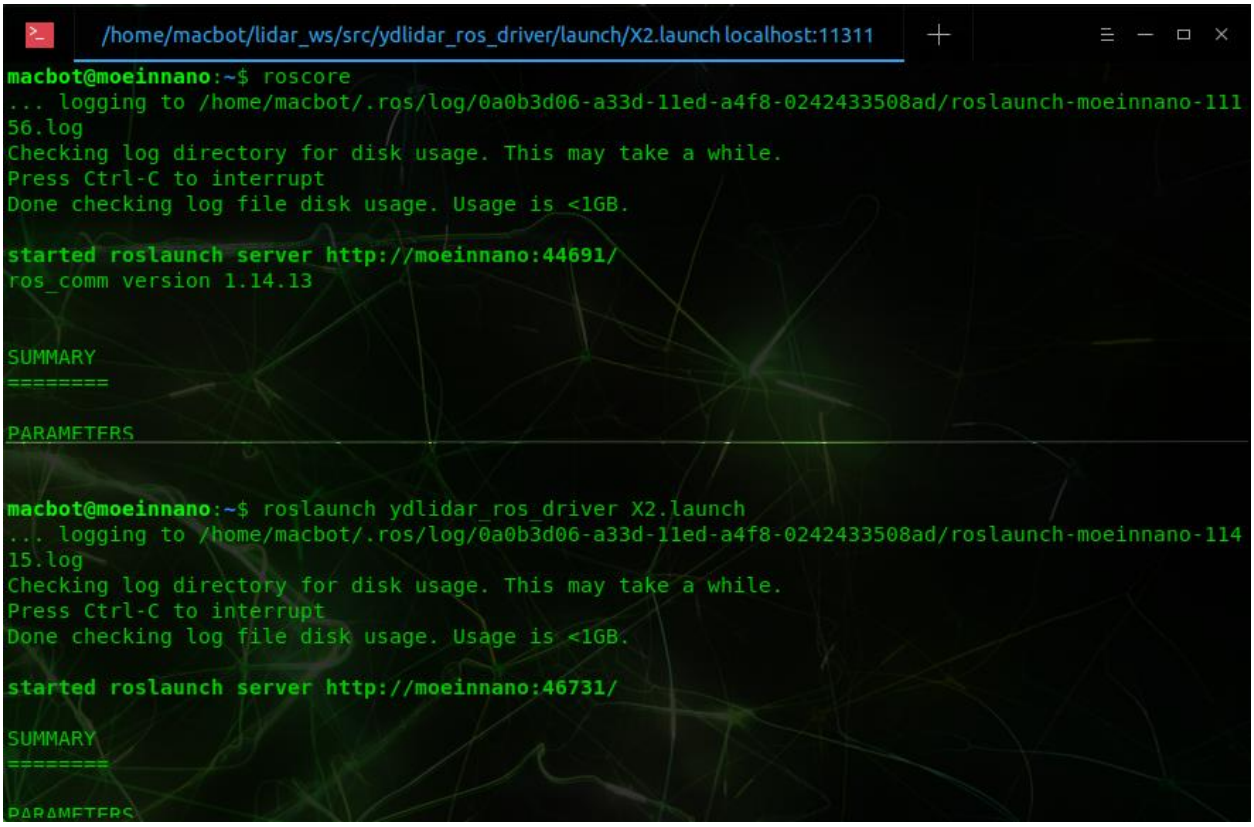
Notice that between the **<launch>** XML tags, the **ydlidar_lidar_publisher** node is started. Next, a lot of ROSParameters are set that contain the LiDAR communication configuration for the X2 lidar. Lastly, two transforms are created that will provide a reference point when visualizing the LiDAR point-cloud.

Before launching the **X2.launch** file, start ROSCore in a new terminal window.



ROSCore is a daemon that allows different ROS nodes to communicate with each other. It needs to be running for any ROS node to run. If one isn't already running, the system will begin ROSCore automatically in another thread.

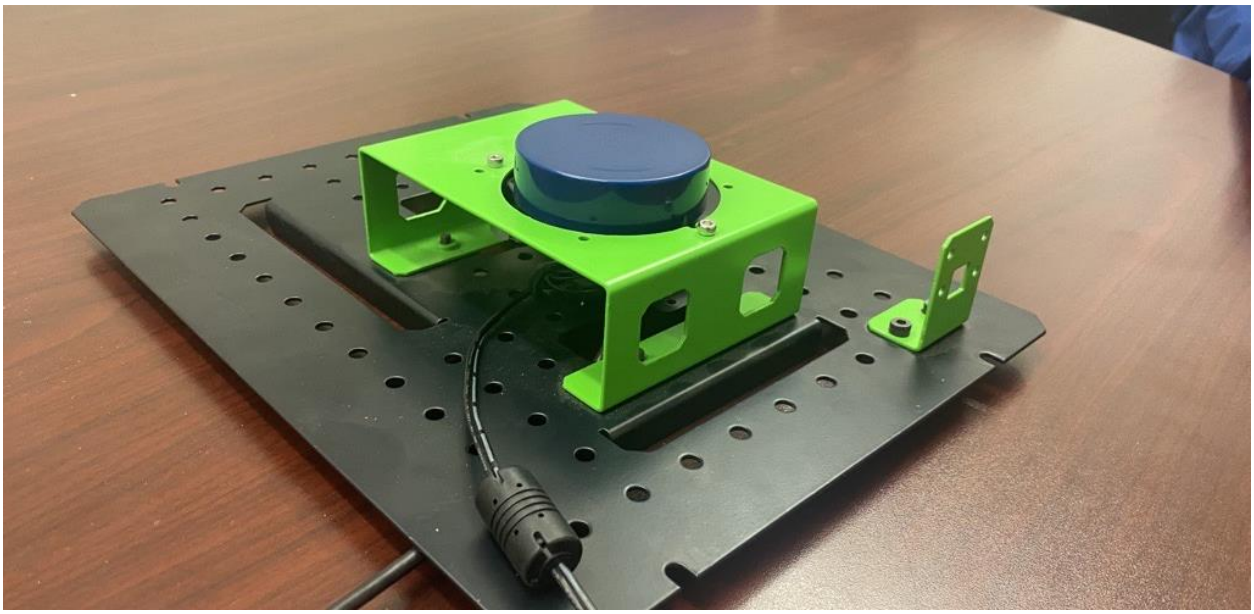
Launch the launch file. There are two ways to do this. If you are not in the same directory, you can use the following command:



Ensure that the LiDAR initializes correctly. You should hear it's fast spinning change to a more constant humming.

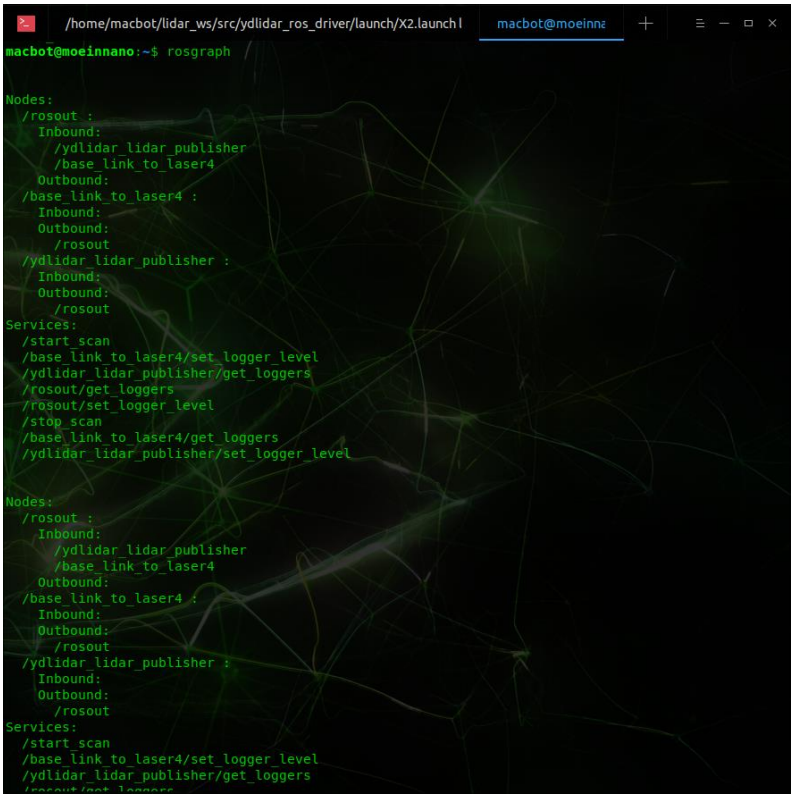
```
/home/macbot/lidar_ws/src/ydlidar_ros_driver/launch/X2.launch localhost:11311

macbot@moeinnano:~$ roscore
... logging to /home/macbot/.ros/log/0a0b3d06-a33d-11ed-a4f8-0242433508ad/roslaunch-moeinnano-111
56.log
process[ydlidar_lidar_publisher-1]: started with pid [11434]
[ INFO] [1675372217.270375650]: YDLIDAR ROS Driver Version: 1.0.2
process[base_link_to_laser4-2]: started with pid [11436]
YDLidar SDK initializing
YDLidar SDK has been initialized
[YDLIDAR]:SDK Version: 1.1.3
LiDAR successfully connected
[YDLIDAR]:Lidar running correctly ! The health status: good
LiDAR init success, Elapsed time 624 ms
Start to getting intensity flag
lastPos 266 currPos 278 offset 12
lastPos 1151 currPos 1163 offset 12
Auto set intensity 0
End to getting intensity flag
[YDlidar] Create thread 0x8FFFF180
[CYdLidar] Successed to start scan mode, Elapsed time 1077 ms
[YDLIDAR] Fixed Size: 720
[YDLIDAR] Sample Rate: 3K
[YDLIDAR] Fixed Size: 720
[YDLIDAR] Sample Rate: 3K
[YDLIDAR]:Single Fixed Size: 470
[YDLIDAR]:Sample Rate: 3K
[YDLIDAR INFO] Single Channel Current Sampling Rate: 3K
[YDLIDAR INFO] Now YDLIDAR is scanning ......
```



Troubleshooting ROS Systems

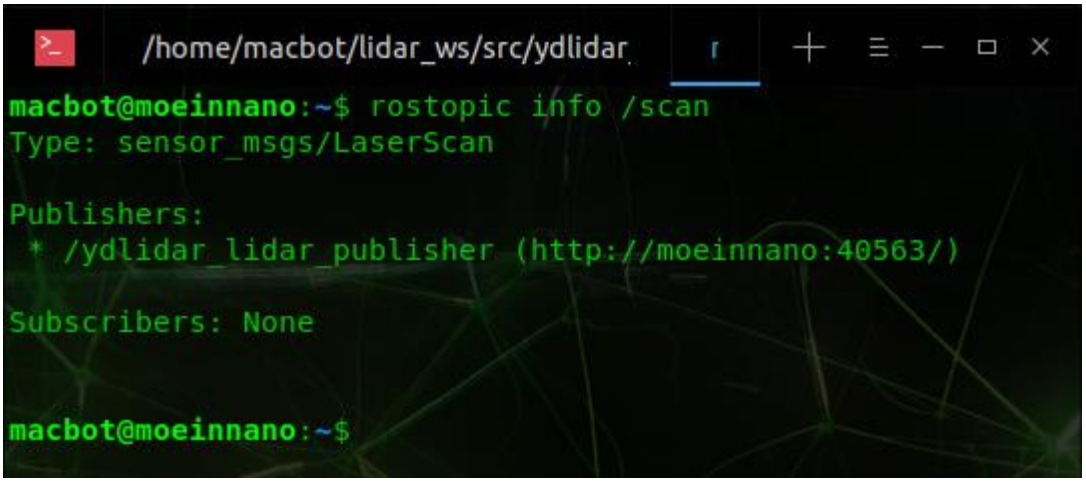
To get a current overview of the ROS system, use the **rosgraph** command in a new terminal window.



To list the active ROS topics, use the **rostopic list** command.

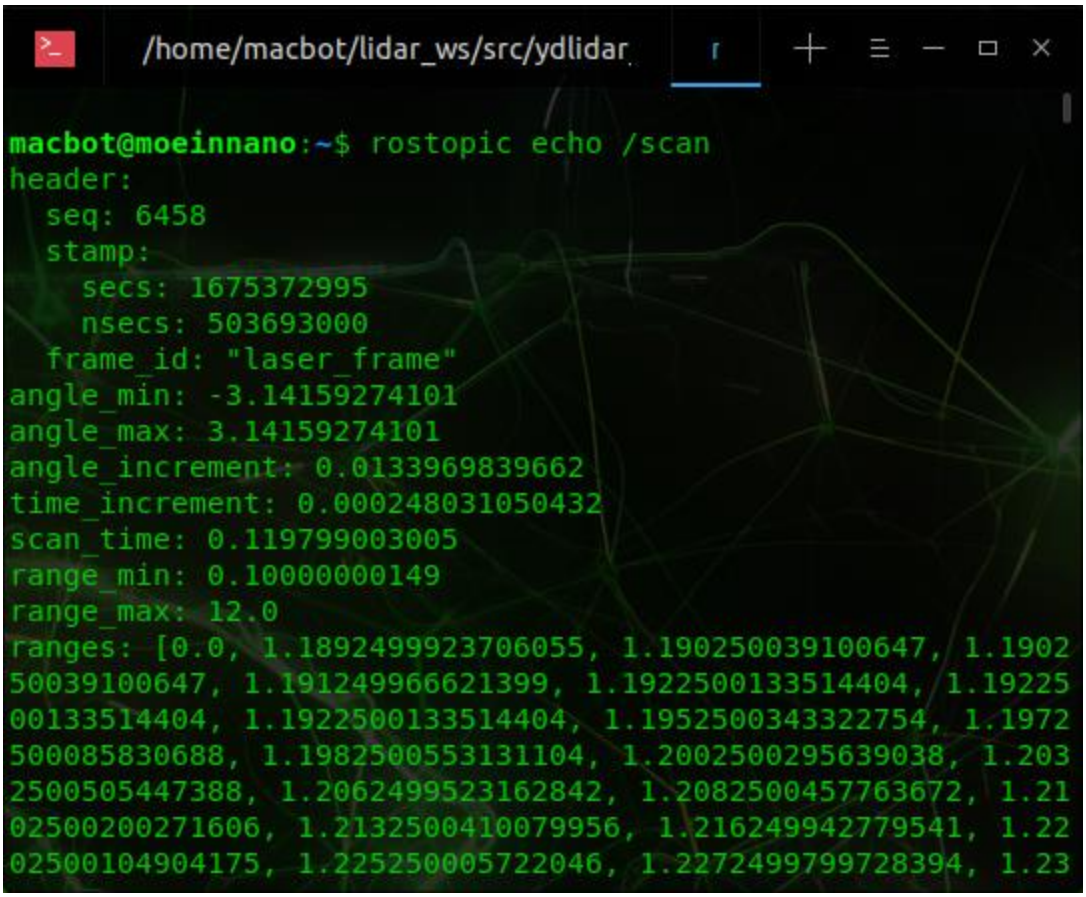
With the LiDAR running, you should see a **/scan** topic.

To find more info on the data being transmitted on this topic, use the **rostopic info** command.



The data structure used for this topic is **sensor_msgs/LaserScan** and the active publishers are **/ydlidar_lidar_publisher** on ROSCore port **40563**.

Next, lets see what raw data is being sent on this topic. This can be done using the **rostopic echo** command.



To exit the echo mode, press **CTRL + c** on your keyboard.

1.0 How is Linux installed?
2.0 Connecting and logging in
3.0 Using the Terminal to install update packages
4.0 Navigating the filesystem
5.0 Creating and editing files from the terminal, GEdit, and VSCode
6.0 Installing ROS
7.0 ROS Commands
8.0 Installing the YDLidar ROS package
9.0 Connecting the YDLidar
10.0 Visualizing the YDLidar