

Introduction to Robot Operating System (ROS)

SEP 783 – Sensors & Actuators

Adam Sokacz

Dr. Moein Mehrtash, LEL

Group #:

First name, Last name, Student #

Date:

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 do 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

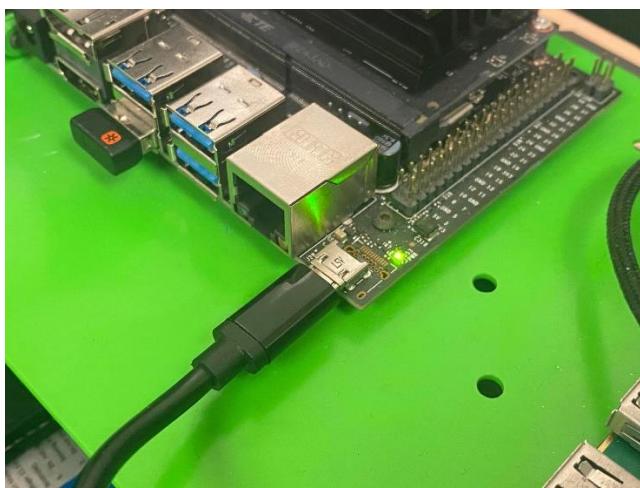
MacBot Setup

Powering the MacBot

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

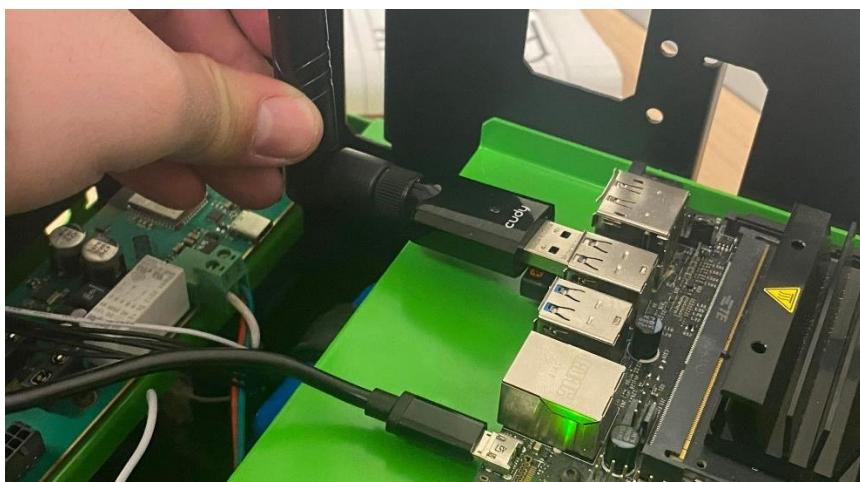
- MicroUSB cable
- 5 W barrel connector
- Internal battery

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.

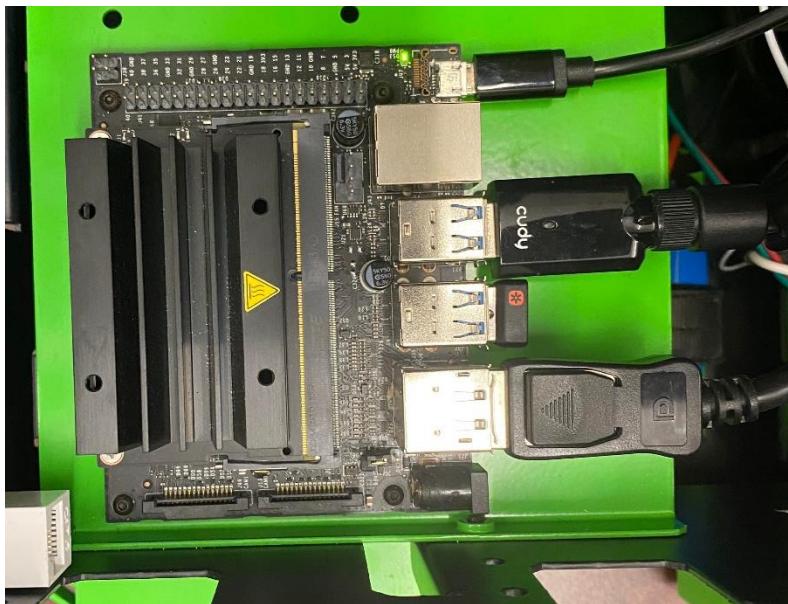


Connect Option #1 – Mouse and Keyboard

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

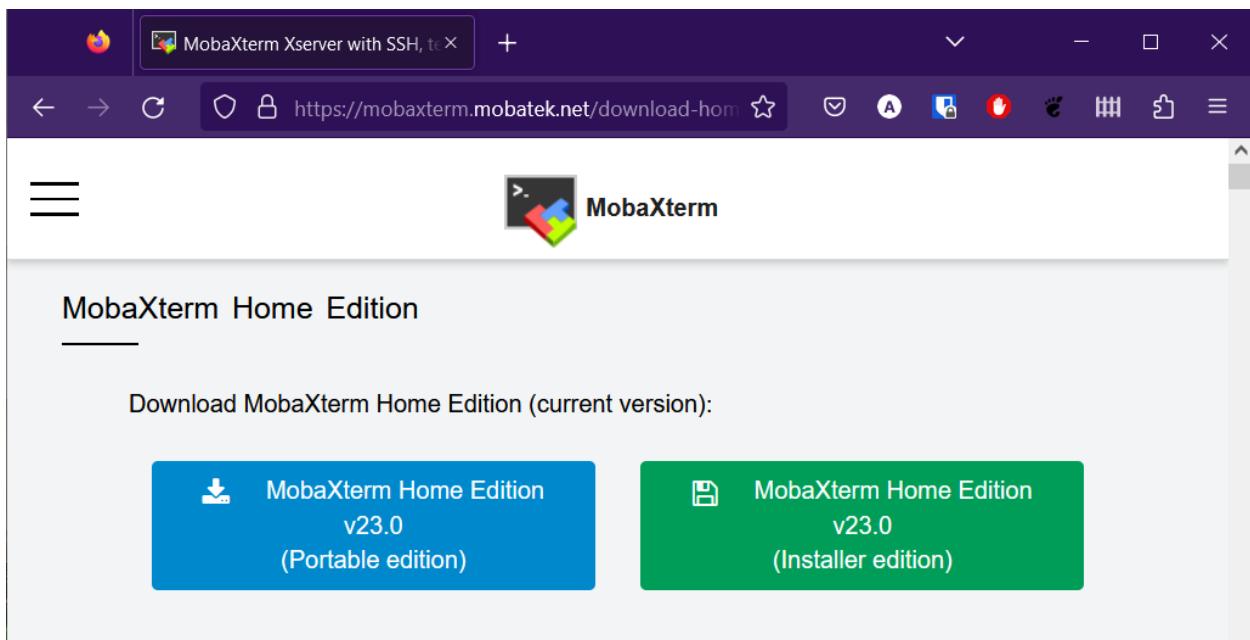
[Connect Option #2 – Remote Connection](#)

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:

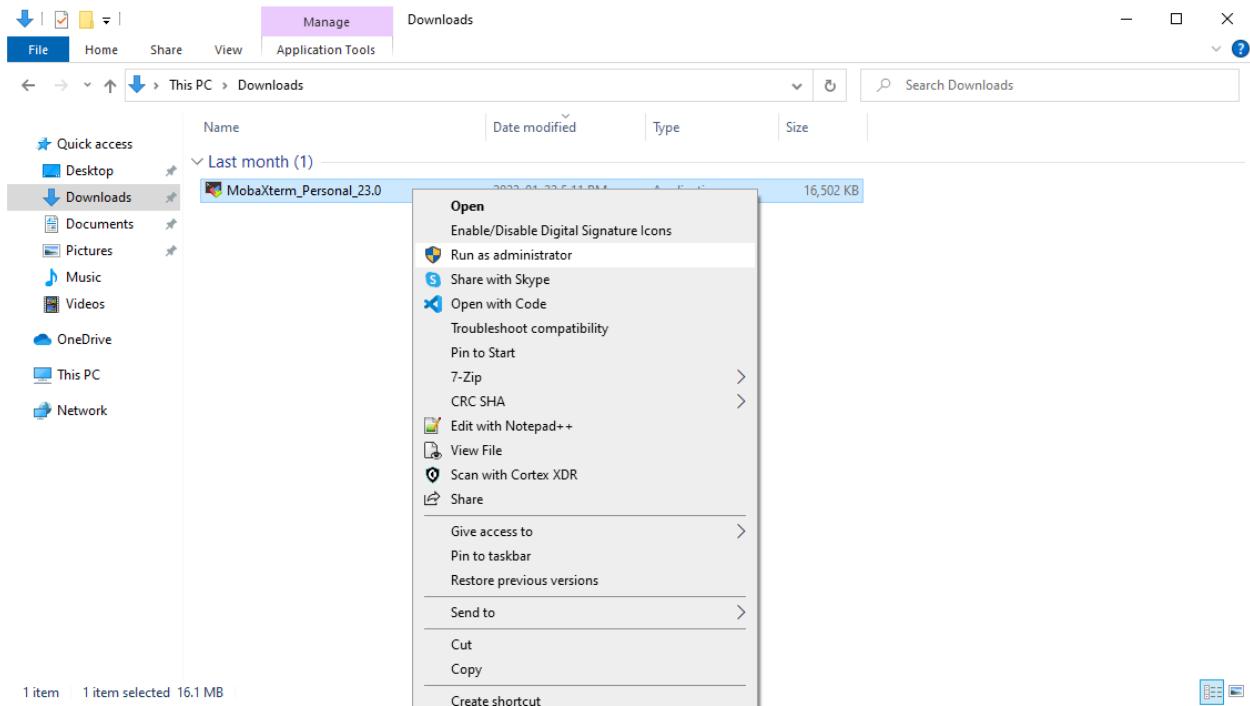
MobaXterm Home Edition Portable (<https://mobaxterm.mobatek.net/download-home-edition.html>)



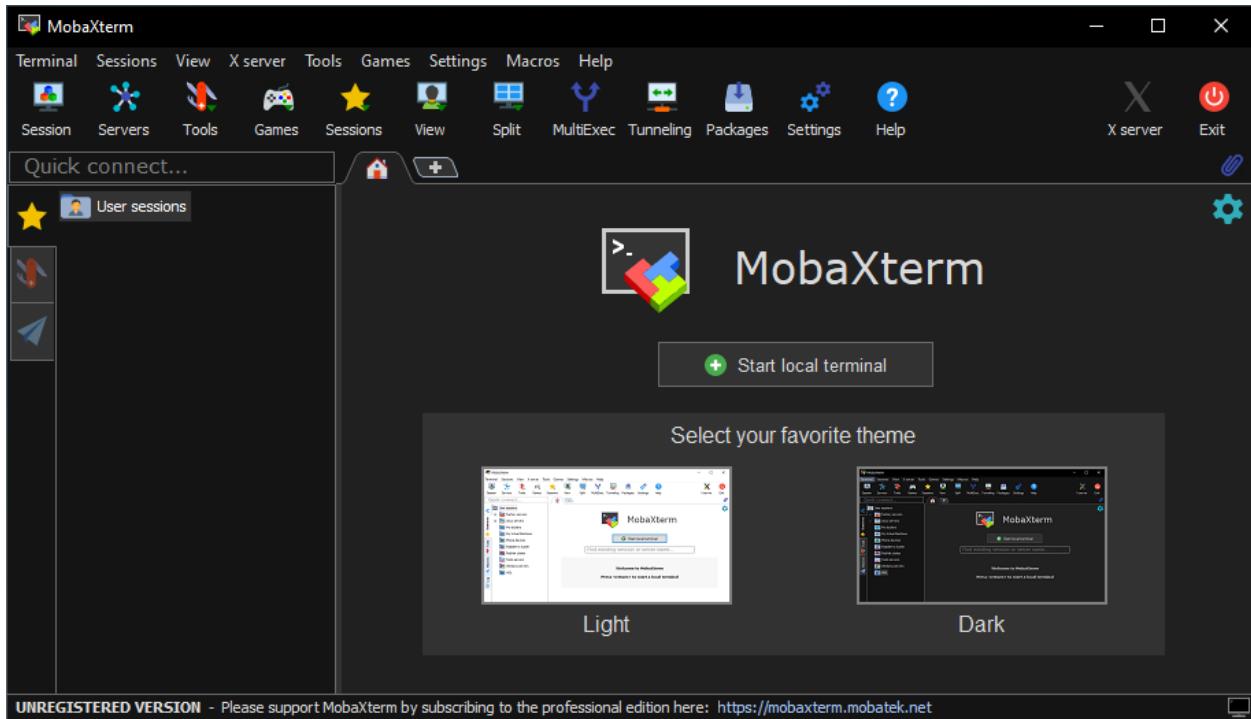
The screenshot shows a Microsoft Edge browser window with the URL <https://mobaxterm.mobatek.net/download-home-edition.html>. The page displays the MobaXterm logo and the title "MobaXterm Home Edition". Below this, there is a section titled "Download MobaXterm Home Edition (current version):" containing two download options:

- A blue button labeled "MobaXterm Home Edition v23.0 (Portable edition)" with a download icon.
- A green button labeled "MobaXterm Home Edition v23.0 (Installer edition)" with a download icon.

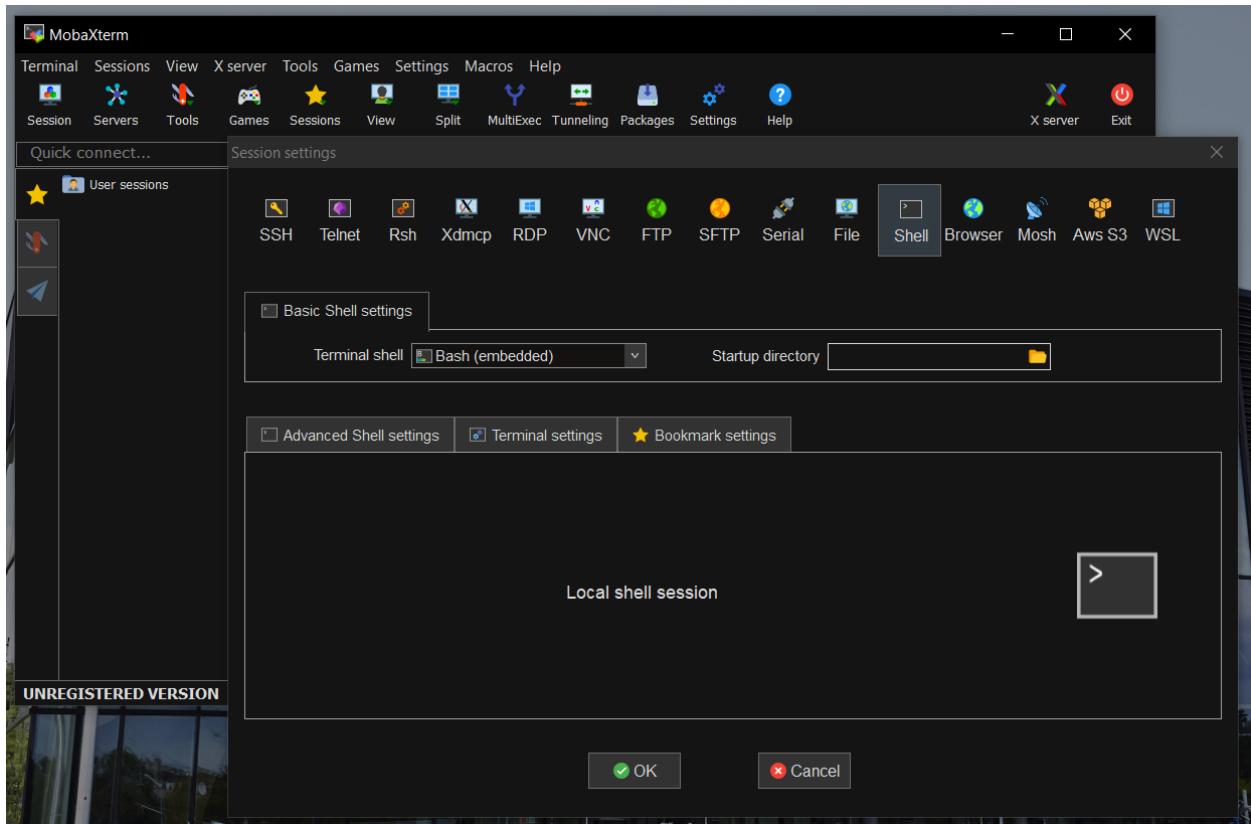
Launch **MobaXterm** as Administrator.



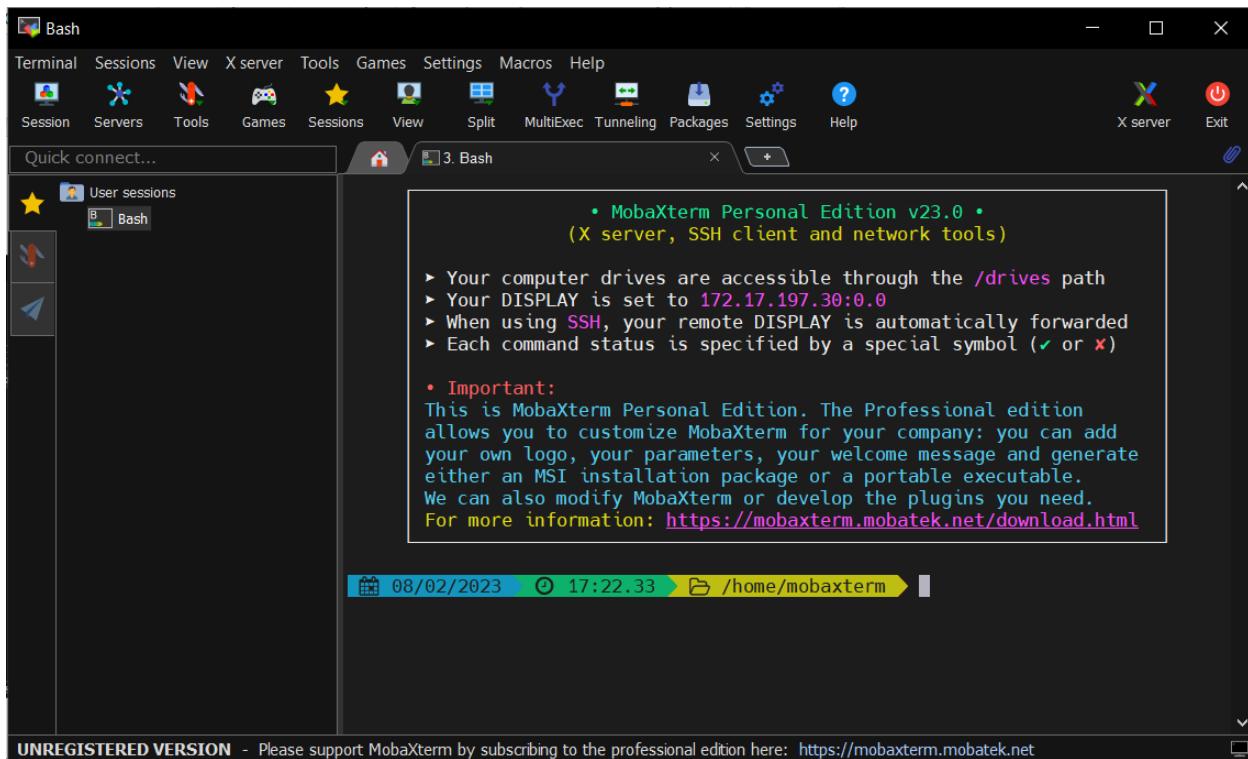
The screenshot shows a Windows File Explorer window with the "Downloads" folder selected. A context menu is open over a file named "MobaXterm_Personal_23.0". The menu includes options like "Open", "Run as administrator", "Share with Skype", "Open with Code", "Edit with Notepad++", "View File", "Scan with Cortex XDR", "Share", "Give access to", "Pin to taskbar", "Restore previous versions", "Send to", "Cut", "Copy", and "Create shortcut".



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.

```

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

```

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

```

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>
```

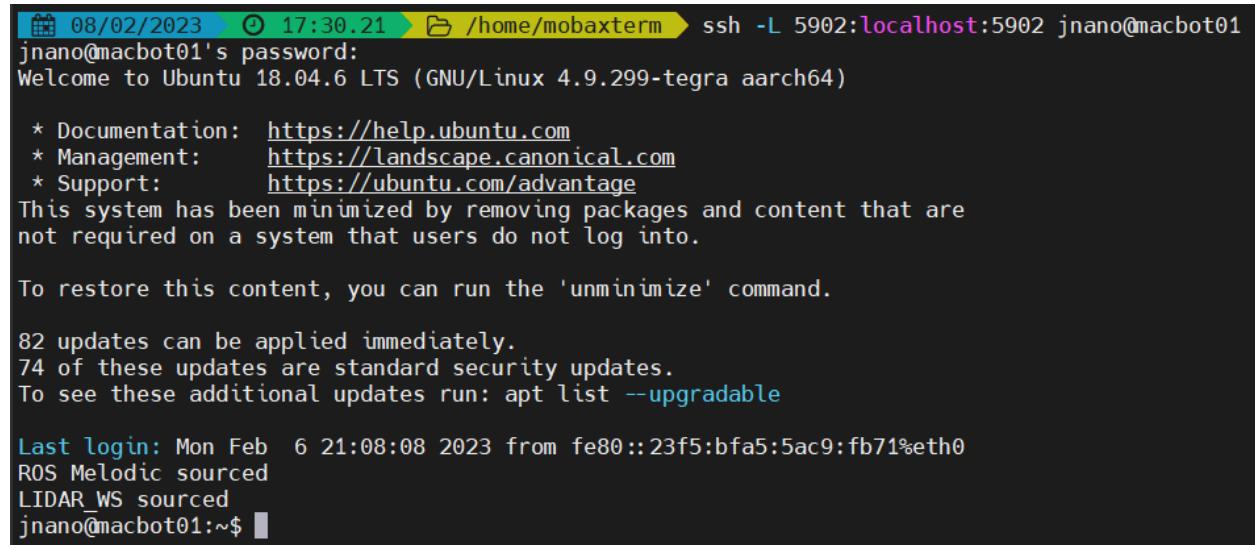
```
ssh jnano@macbot01
```

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.



A screenshot of a terminal window showing an SSH session. The title bar shows the date (08/02/2023), time (17:30, 21), and current directory (/home/mobaxterm). The command entered is ssh -L 5902:localhost:5902 jnano@macbot01. The password is entered, followed by a welcome message for Ubuntu 18.04.6 LTS (GNU/Linux 4.9.299-tegra aarch64). It provides documentation links and a note about system minimization. It also mentions 82 updates available, 74 of which are security updates, and provides a command to see additional updates. The last login information is displayed, followed by a prompt at the end of the session.

```
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.

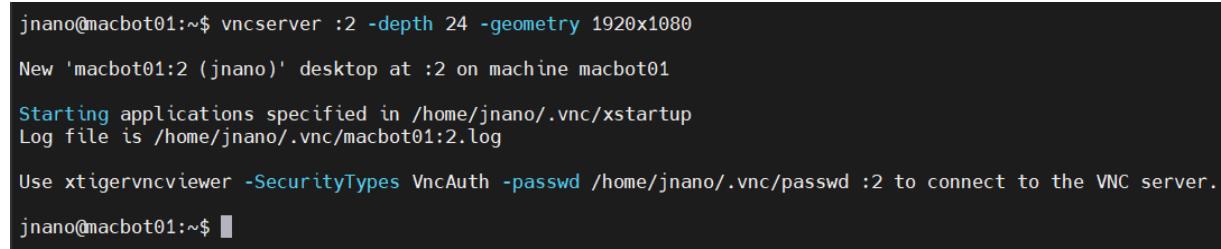
To restore this content, you can run the 'unminimize' command.

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
```



A screenshot of a terminal window showing the command vncserver :2 -depth 24 -geometry 1920x1080 being run. The output shows the creation of a new desktop at :2 on machine macbot01, starting applications from /home/jnano/.vnc/xstartup, and providing a log file path. It also gives instructions on how to connect using xtigervncviewer.

```
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:~$
```

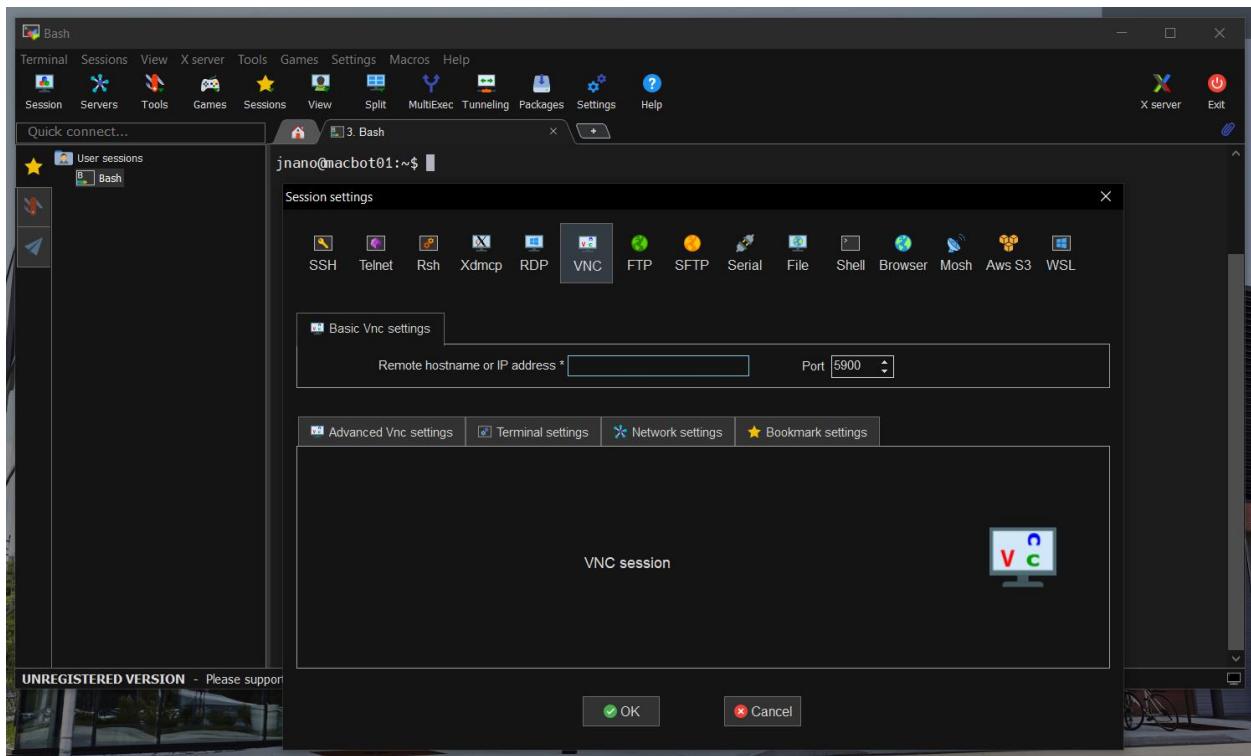
To ensure that the server is running, we can use the following command:

```
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.

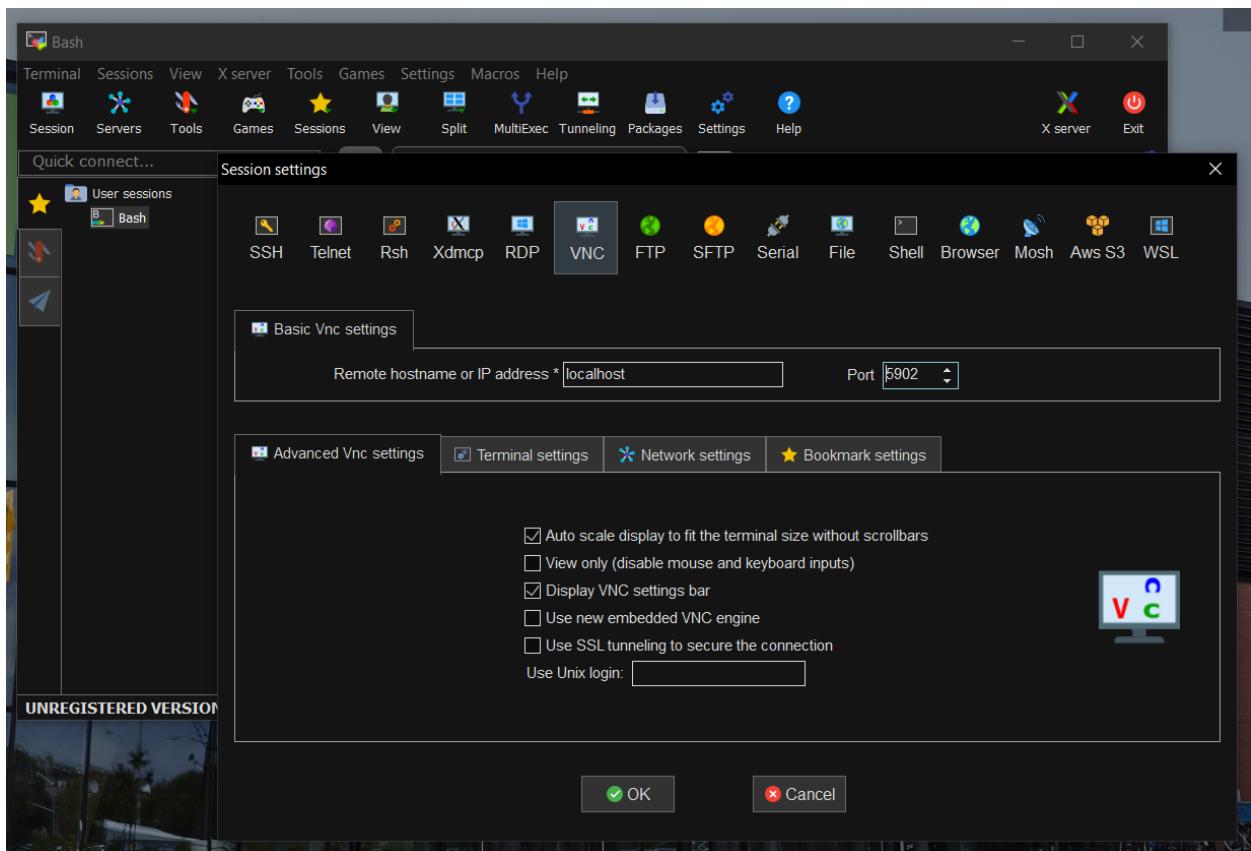


Set the following configuration:

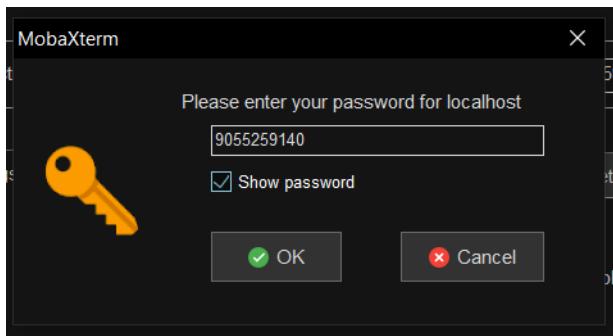
hostname: localhost

port: 5902

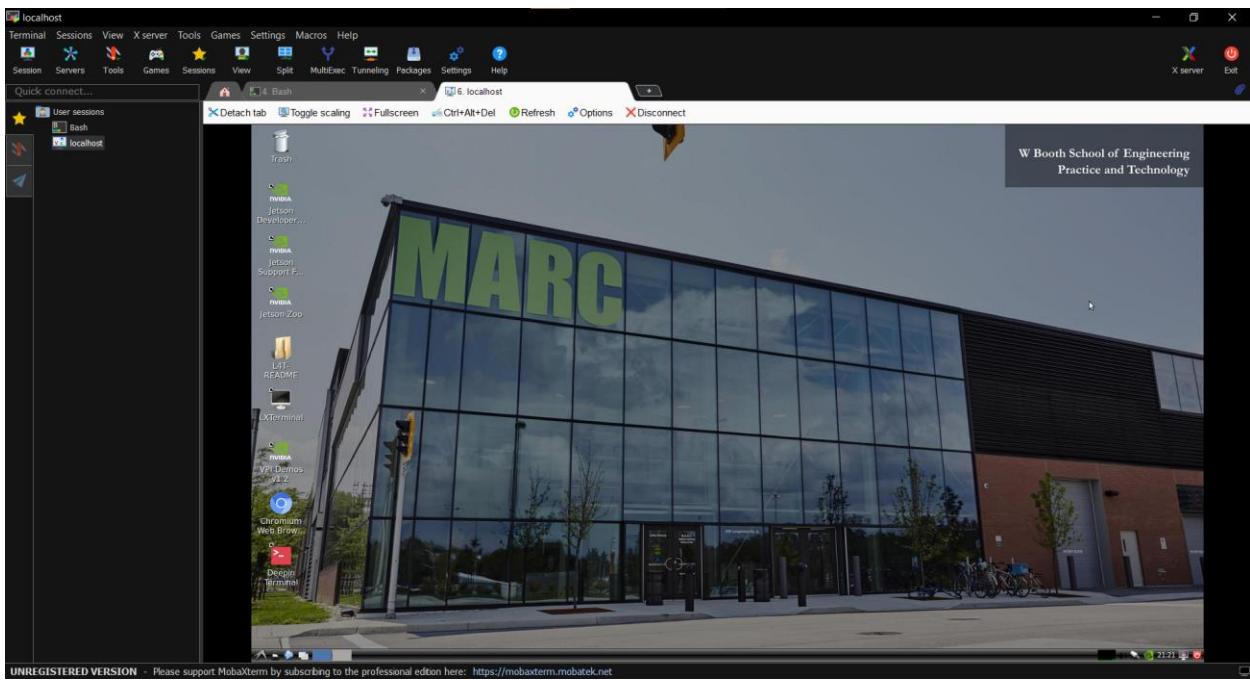
Uncheck: Use new embedded VNC engine.



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
```

```
jnano@macbot01:~$ ls
Desktop Documents Downloads examples.desktop Music Pictures Public Templates Videos
jnano@macbot01:~$ mkdir -p lidar_ws/src
jnano@macbot01:~$ ls
Desktop Documents Downloads examples.desktop lidar_ws Music Pictures Public Templates Videos
jnano@macbot01:~/lidar_ws/src$ cd lidar_ws/src/
jnano@macbot01:~/lidar_ws/src$ ls
jnano@macbot01:~/lidar_ws/src$
```

A screenshot of a terminal window titled 'src'. The terminal shows the following commands and output:

```
jnano@macbot01:~$ ls
Desktop Documents Downloads examples.desktop Music Pictures Public Templates Videos
jnano@macbot01:~$ mkdir -p lidar_ws/src
jnano@macbot01:~$ ls
Desktop Documents Downloads examples.desktop lidar_ws Music Pictures Public Templates Videos
jnano@macbot01:~/lidar_ws/src$ cd lidar_ws/src/
jnano@macbot01:~/lidar_ws/src$ ls
jnano@macbot01:~/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.

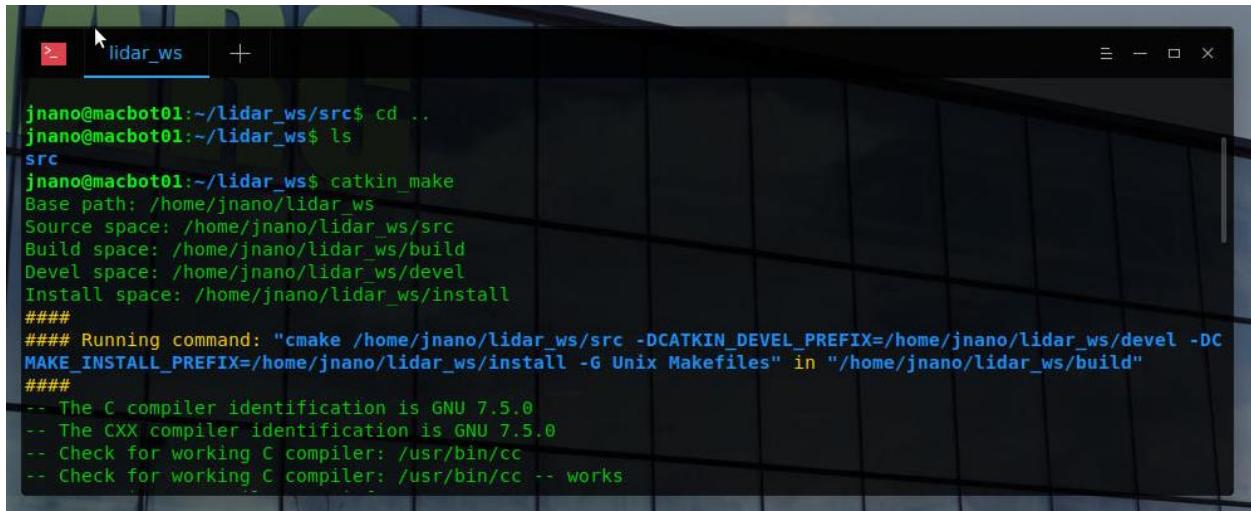


```
jnano@macbot01:~/lidar_ws/src$ catkin_init_workspace
Creating symlink "/home/jnano/lidar_ws/src/CMakeLists.txt" pointing to "/opt/ros/melodic/share/catkin/cmake/toplevel.cmake"
jnano@macbot01:~/lidar_ws/src$ ls
CMakeLists.txt
jnano@macbot01:~/lidar_ws/src$
```

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

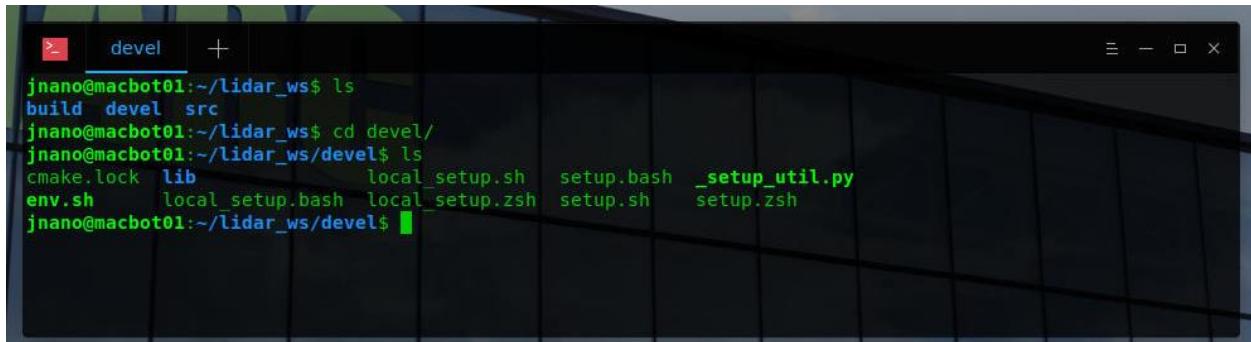
```
cd ~/lidar_ws
```

```
catkin_make
```



```
jnano@macbot01:~/lidar_ws/src$ cd ..
jnano@macbot01:~/lidar_ws$ ls
src
jnano@macbot01:~/lidar_ws$ catkin_make
Base path: /home/jnano/lidar_ws
Source space: /home/jnano/lidar_ws/src
Build space: /home/jnano/lidar_ws/build
Devel space: /home/jnano/lidar_ws/devel
Install space: /home/jnano/lidar_ws/install
#####
##### Running command: "cmake /home/jnano/lidar_ws/src -DCATKIN_DEVEL_PREFIX=/home/jnano/lidar_ws/devel -DCMAKE_INSTALL_PREFIX=/home/jnano/lidar_ws/install -G Unix Makefiles" in "/home/jnano/lidar_ws/build"
#####
-- The C compiler identification is GNU 7.5.0
-- The CXX compiler identification is GNU 7.5.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
```

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.

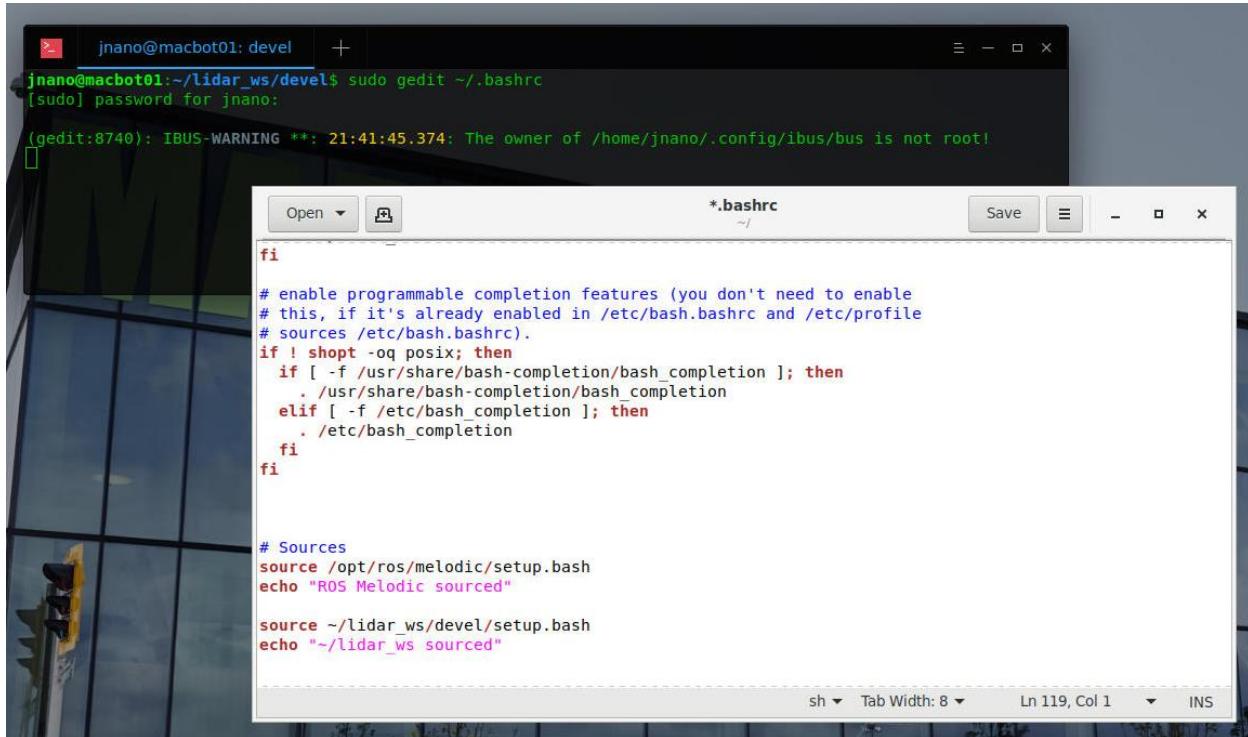


```
jnano@macbot01:~/lidar_ws$ ls
build  devel  src
jnano@macbot01:~/lidar_ws$ cd devel/
jnano@macbot01:~/lidar_ws/devel$ ls
cmake.lock  lib          local_setup.sh  setup.bash  _setup_util.py
env.sh      local_setup.bash  local_setup.zsh  setup.sh   setup.zsh
jnano@macbot01:~/lidar_ws/devel$
```

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>



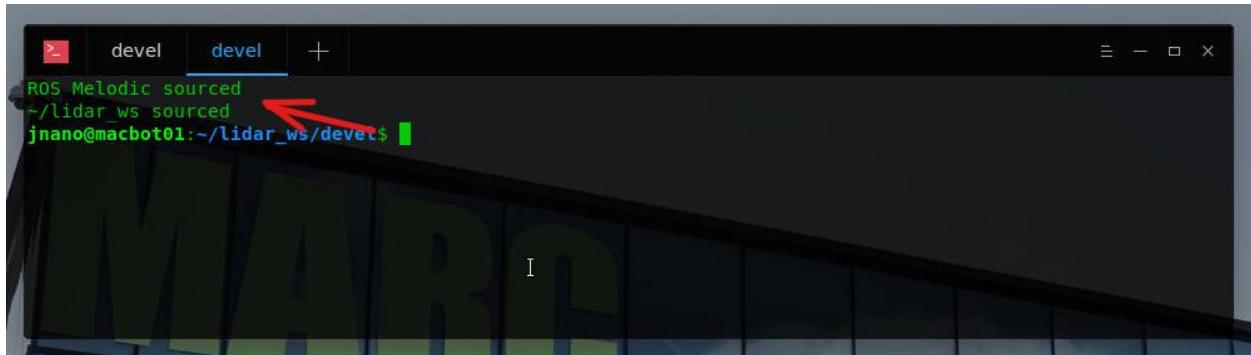
Notice the `#Sources` section that was previously added. Ensure that you have these two paths sourced and echoed to the terminal window.

`# 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 terminal window.



A screenshot of a terminal window titled "devel". The window shows two lines of text: "ROS Melodic sourced" and "~/lidar_ws sourced". A red arrow points to the second line of text.

```
ROS Melodic sourced
~/lidar_ws sourced
jnano@macbot01:~/lidar_ws/devel$
```

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

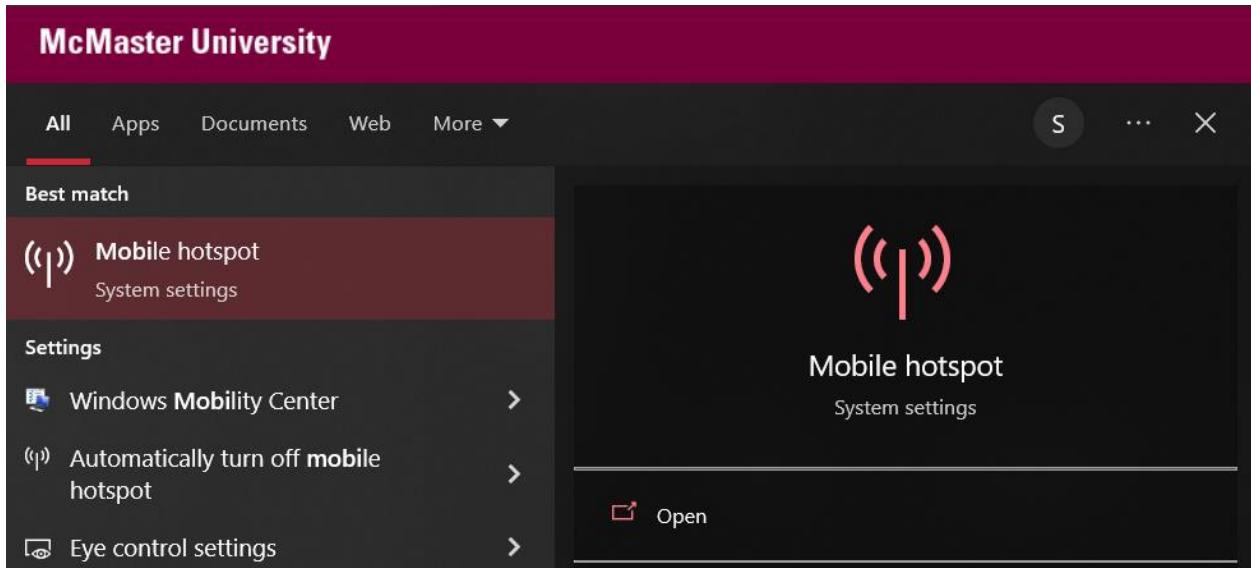
Installing The YDLidar SDK

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.

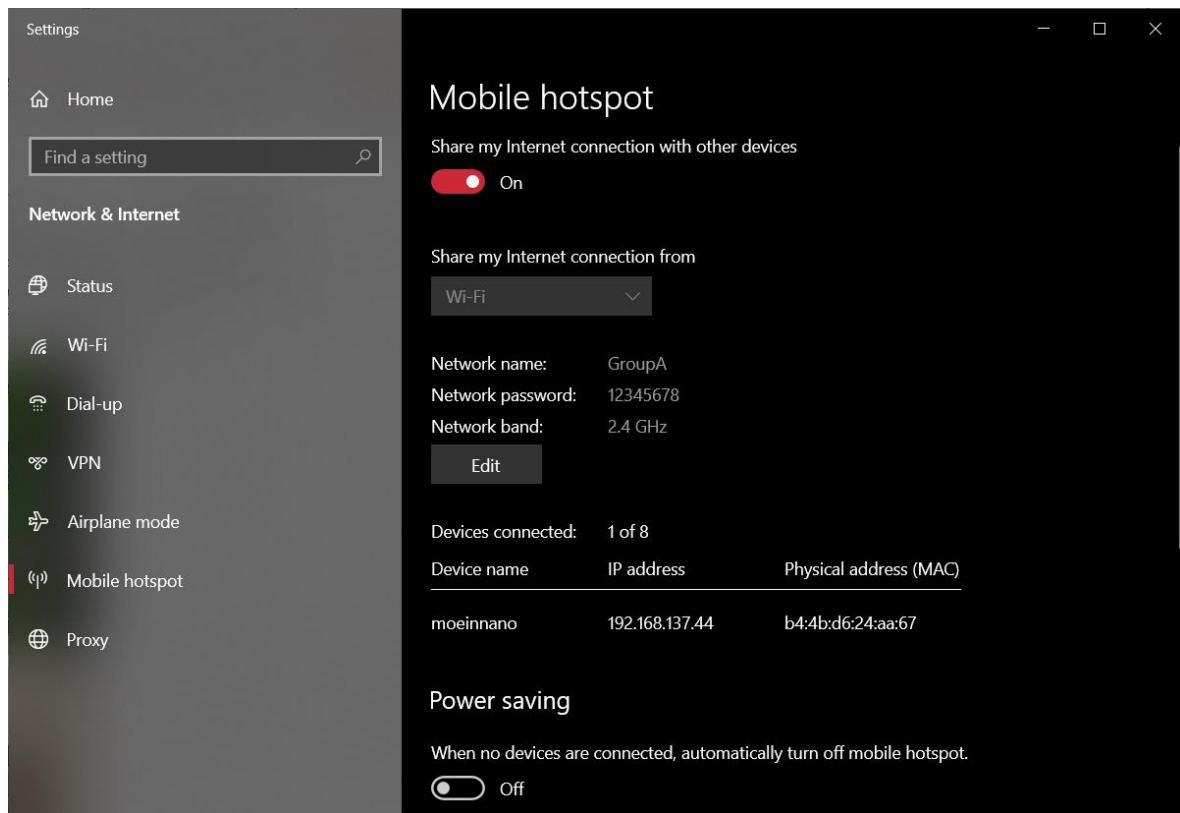
PC Hotspot

Open the Mobile Hotspot application on a personal laptop.

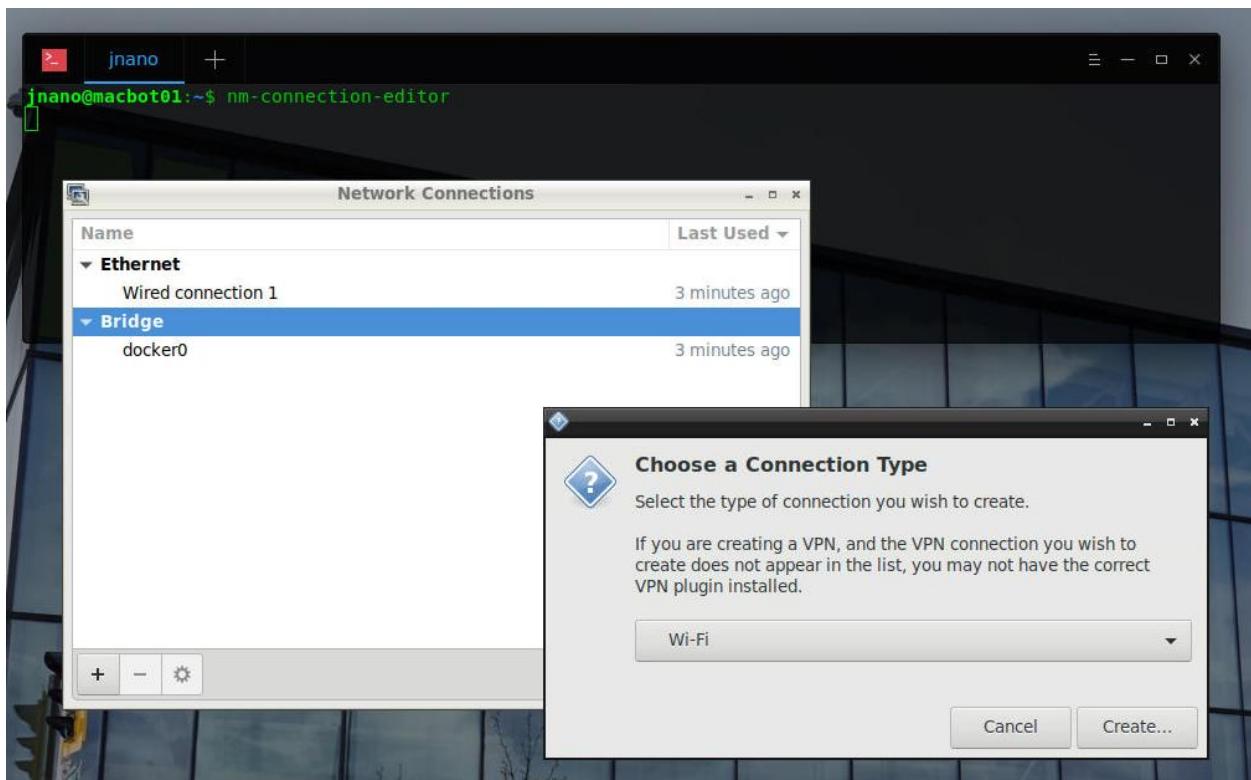


Configure a **unique SSID** and simple 8-digit password.

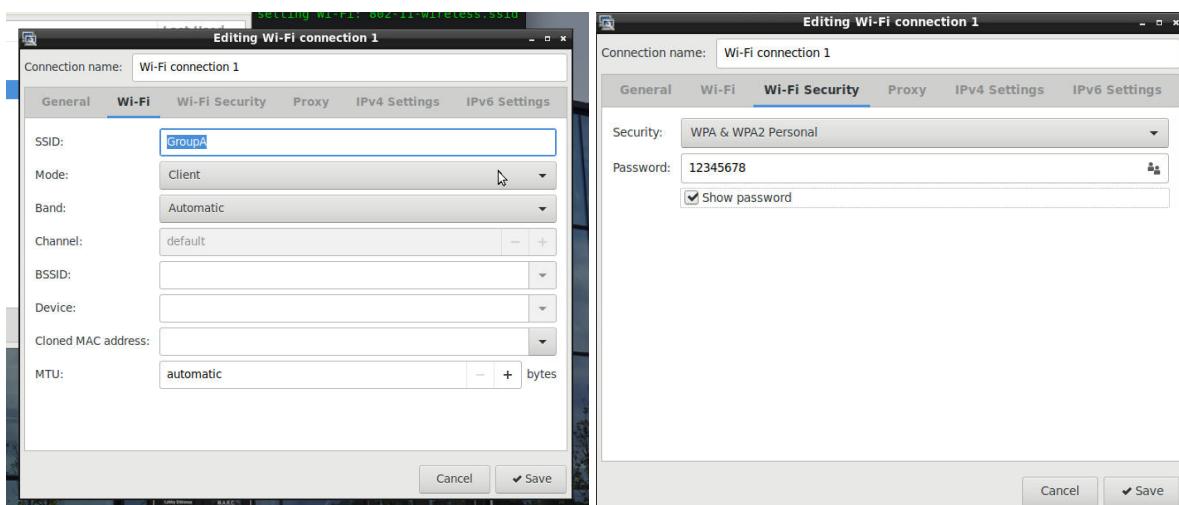
To avoid any compatibility issues, ensure that the **Network Band** is set to **2.4GHz**, and turn off **Power Saving** mode so the hotspot runs continuously. Toggle it **ON**.



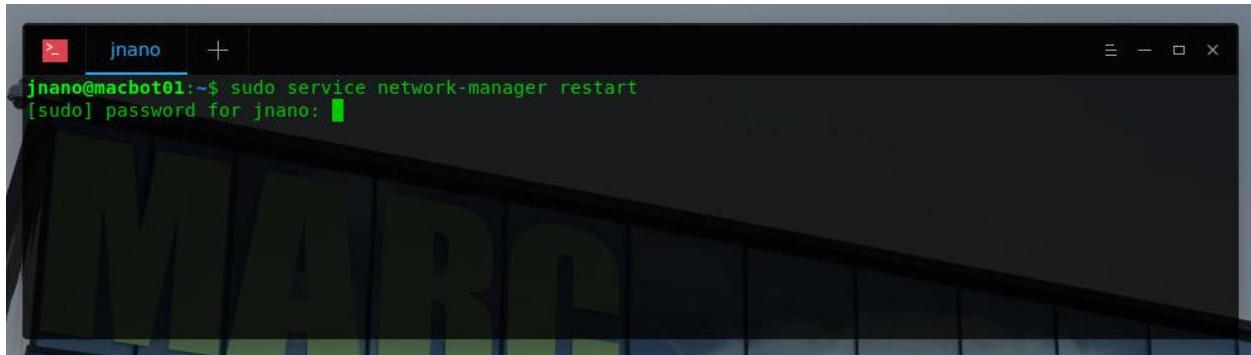
To connect to it, we can use the **nm-connection-editor**.



Set the SSID and password.



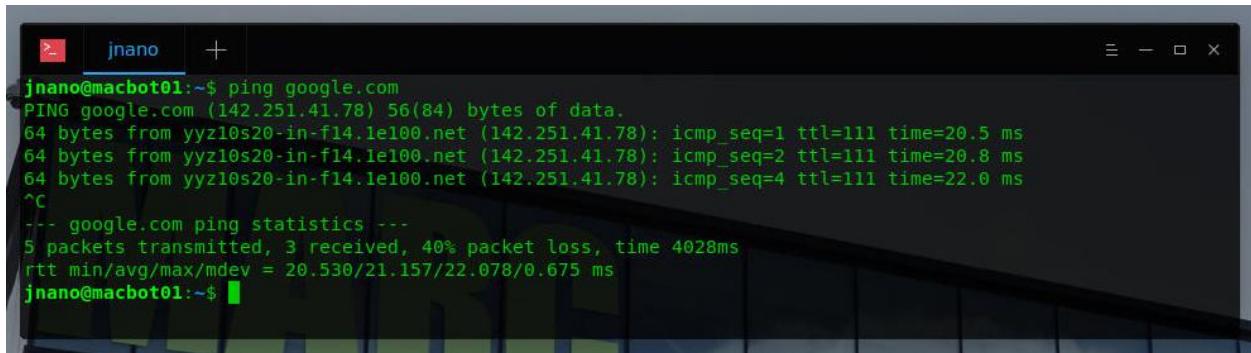
Restart the network manager.



```
jnano@macbot01:~$ sudo service network-manager restart
[sudo] password for jnano: [REDACTED]
```

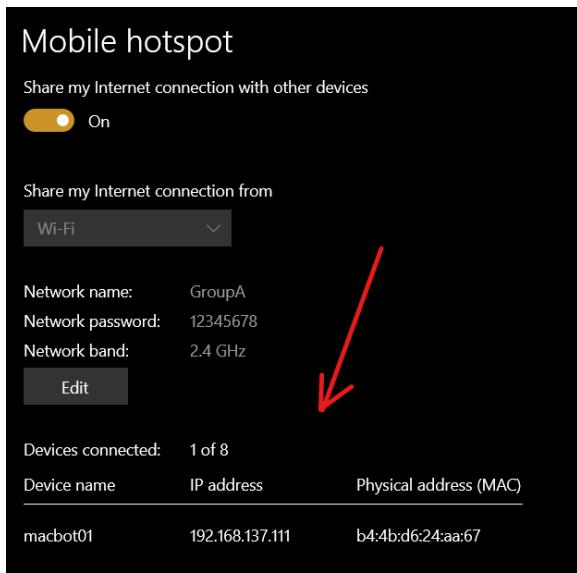
Wait **20 seconds** as your connection to the MacBot is temporarily interrupted.

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.



```
jnano@macbot01:~$ ping google.com
PING google.com (142.251.41.78) 56(84) bytes of data.
64 bytes from yyz10s20-in-f14.1e100.net (142.251.41.78): icmp_seq=1 ttl=111 time=20.5 ms
64 bytes from yyz10s20-in-f14.1e100.net (142.251.41.78): icmp_seq=2 ttl=111 time=20.8 ms
64 bytes from yyz10s20-in-f14.1e100.net (142.251.41.78): icmp_seq=3 ttl=111 time=22.0 ms
^C
-- google.com ping statistics ---
5 packets transmitted, 3 received, 40% packet loss, time 4028ms
rtt min/avg/max/mdev = 20.530/21.157/22.078/0.675 ms
jnano@macbot01:~$
```

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

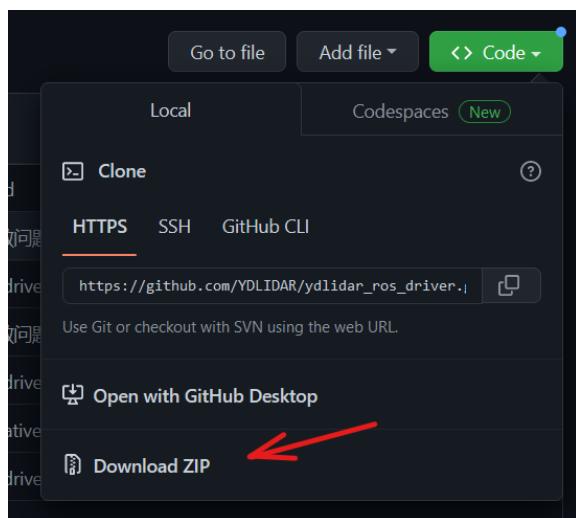
<https://github.com/YDLIDAR/YDLidar-SDK>

```
cd ~/lidar_ws/src  
git clone https://github.com/YDLIDAR/ydlidar\_ros\_driver  
cd ~/Documents  
git clone https://github.com/YDLIDAR/YDLidar-SDK
```

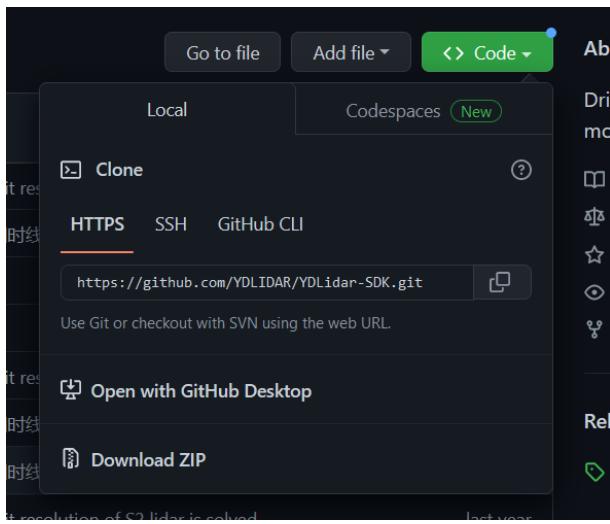
Option #2 – Sending from PC Over Ethernet

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

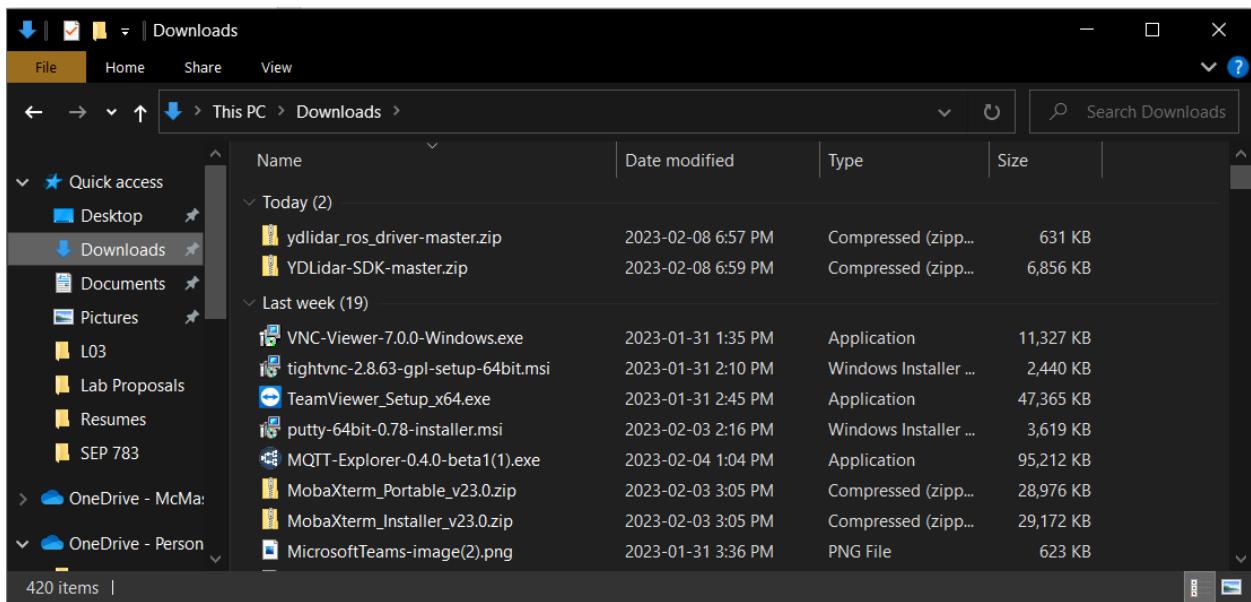
https://github.com/YDLIDAR/ydlidar_ros_driver



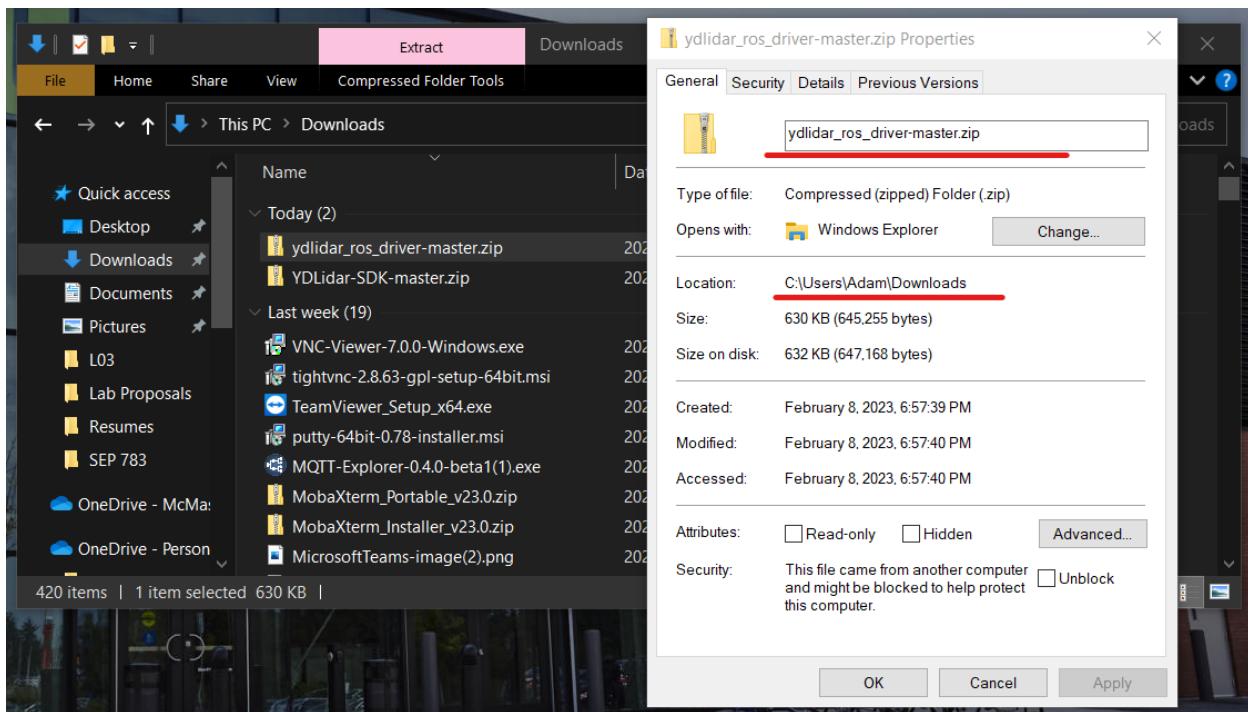
<https://github.com/YDLIDAR/YDLidar-SDK>



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.

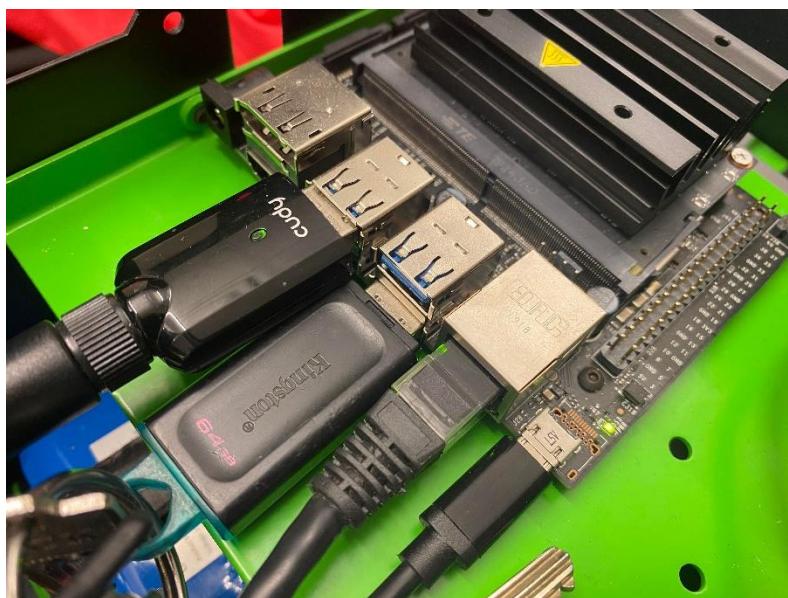


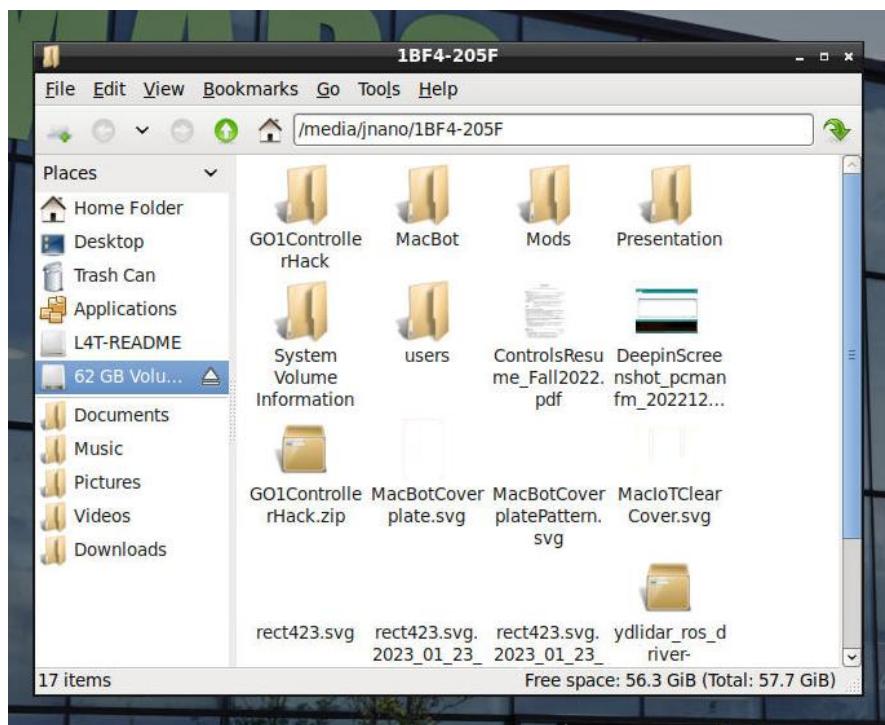
Here are the file transfers:

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

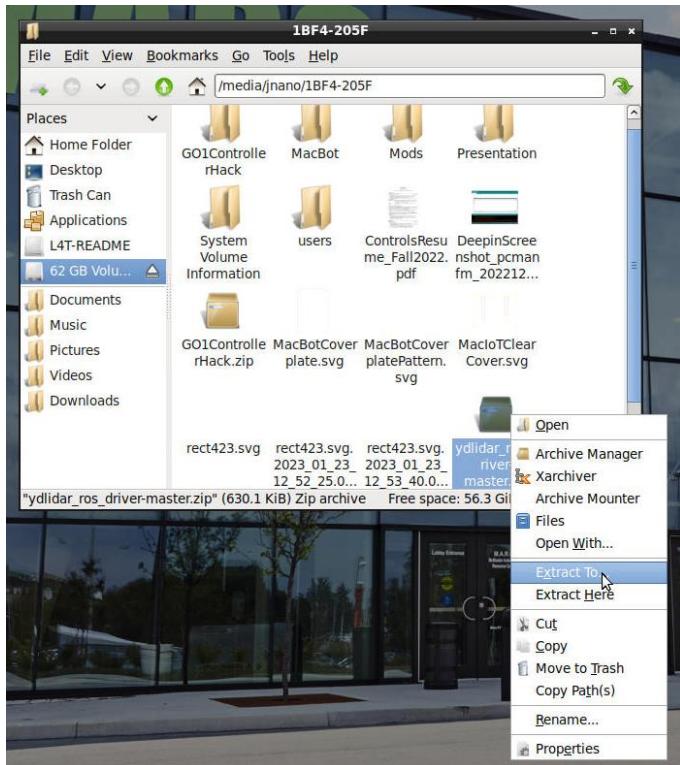
LiDAR ROS: "C:\Users\Adam\Downloads\ydlidar_ros_driver-master.zip" → /home/jnano/lidar_ws/src

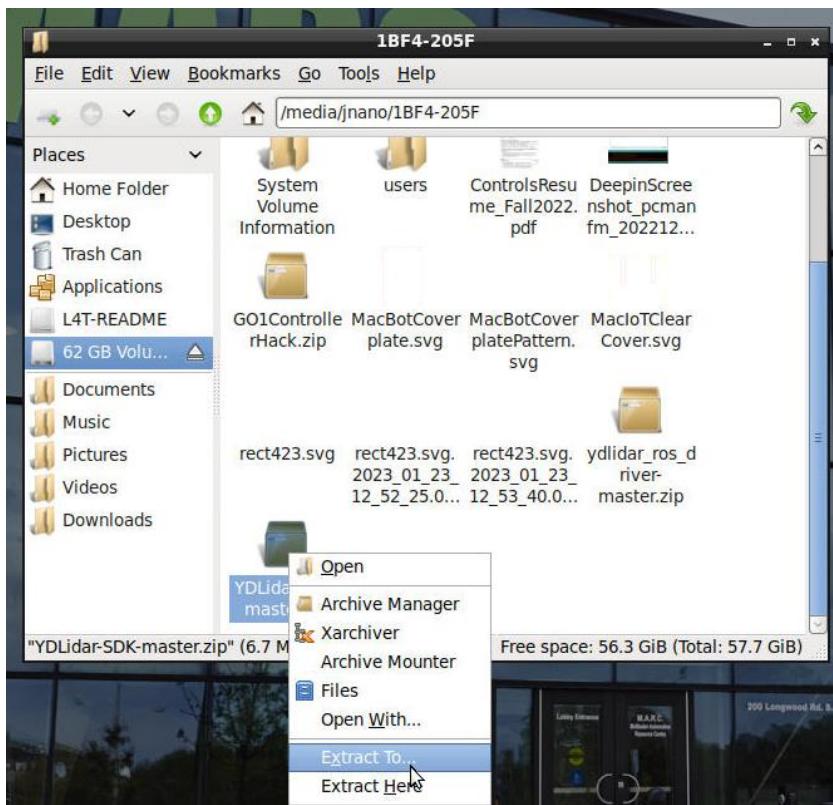
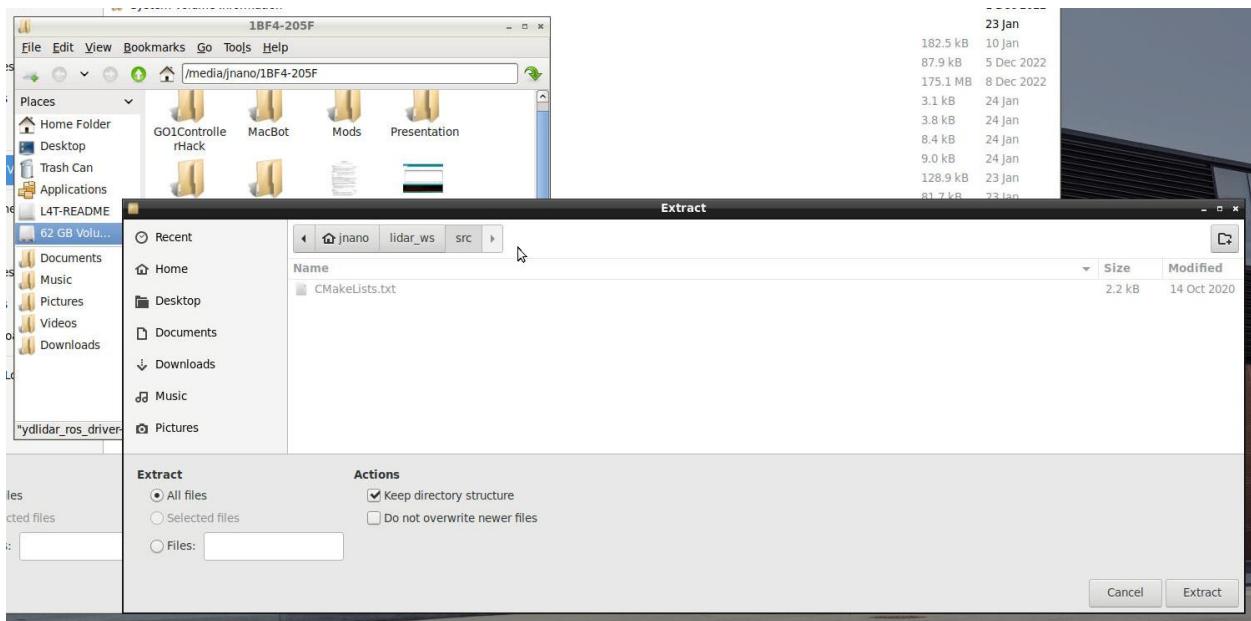
You can do this using SFTP from within MobaXTerm, but I think the least complicated way is to just use a USB thumb drive and copy the files to it.

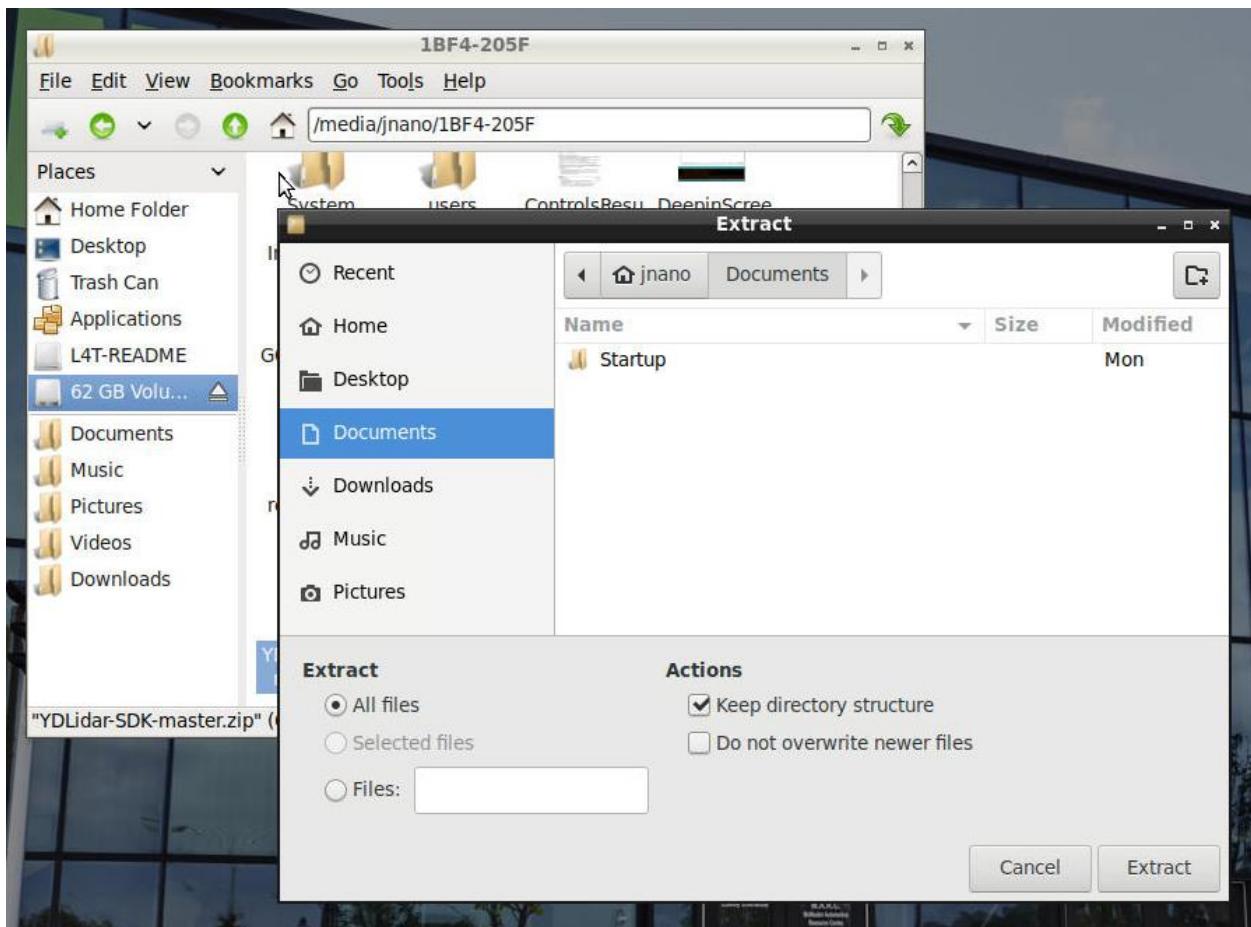




Drag and drop the files to their correct locations within VNC.

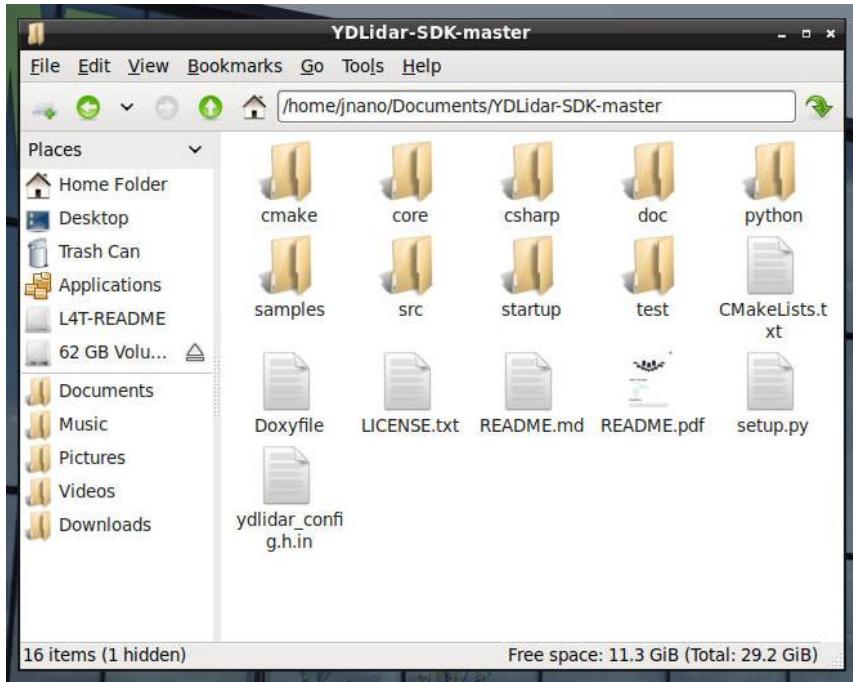




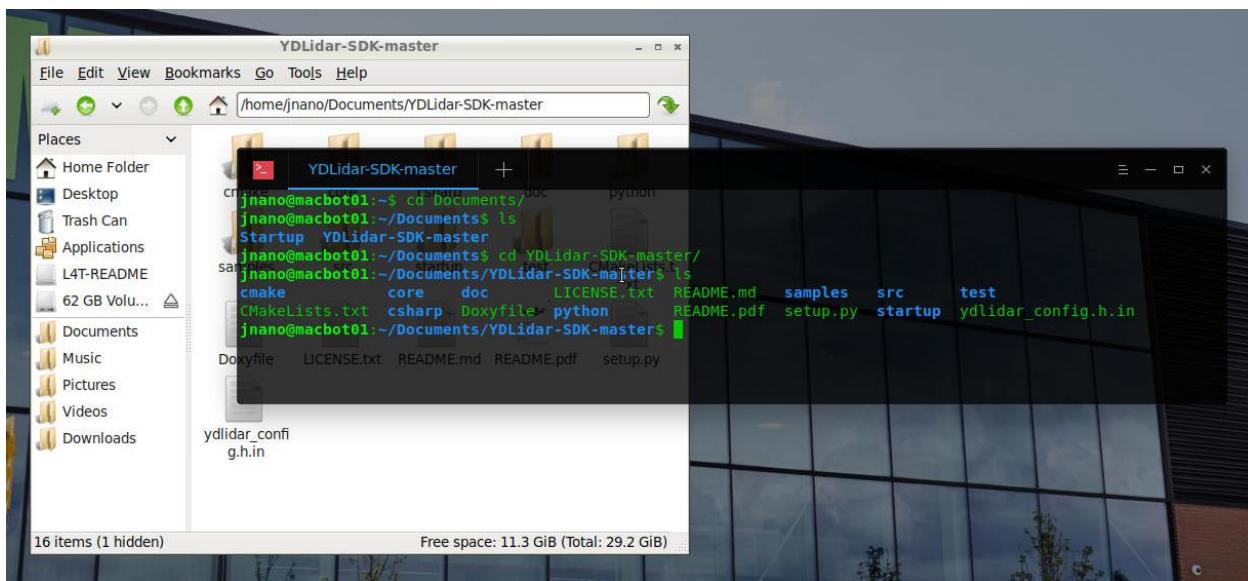


Building the YDLiDAR SDK

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



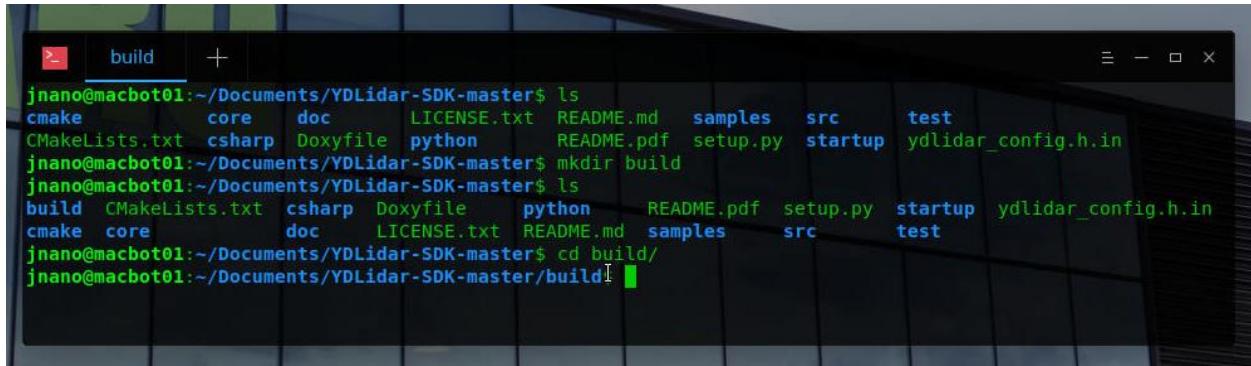
Open a terminal window and navigate to this Directory.



Make a directory called **build/**.

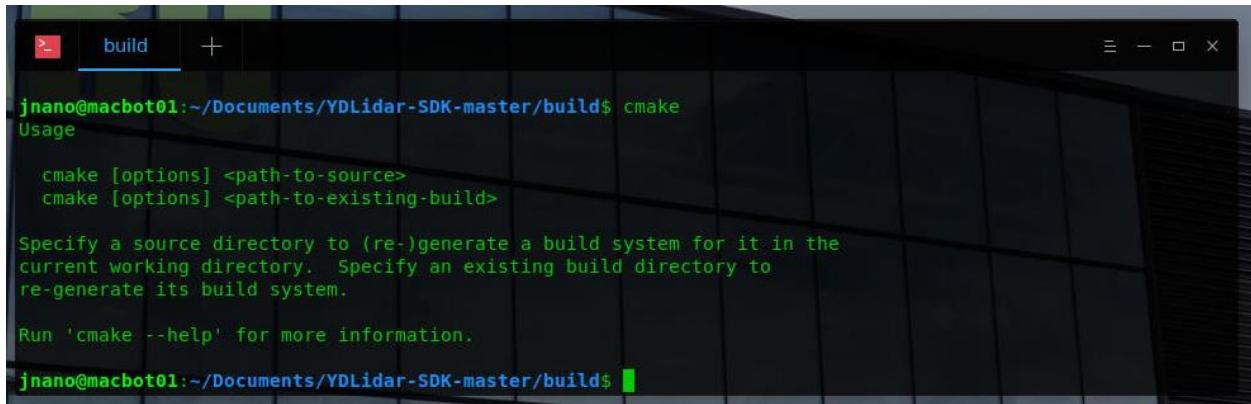
```
mkdir build
```

```
cd build
```



```
jnano@macbot01:~/Documents/YDLidar-SDK-master$ ls
cmake    core    doc    LICENSE.txt  README.md  samples  src    test
CMakeLists.txt  csharp  Doxyfile  python   README.pdf  setup.py  startup  ydlidar_config.h.in
jnano@macbot01:~/Documents/YDLidar-SDK-master$ mkdir build
jnano@macbot01:~/Documents/YDLidar-SDK-master$ ls
build  CMakeLists.txt  csharp  Doxyfile  python   README.pdf  setup.py  startup  ydlidar_config.h.in
cmake  core    doc    LICENSE.txt  README.md  samples  src    test
jnano@macbot01:~/Documents/YDLidar-SDK-master$ cd build/
jnano@macbot01:~/Documents/YDLidar-SDK-master/build$
```

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



```
jnano@macbot01:~/Documents/YDLidar-SDK-master/build$ cmake
Usage

  cmake [options] <path-to-source>
  cmake [options] <path-to-existing-build>

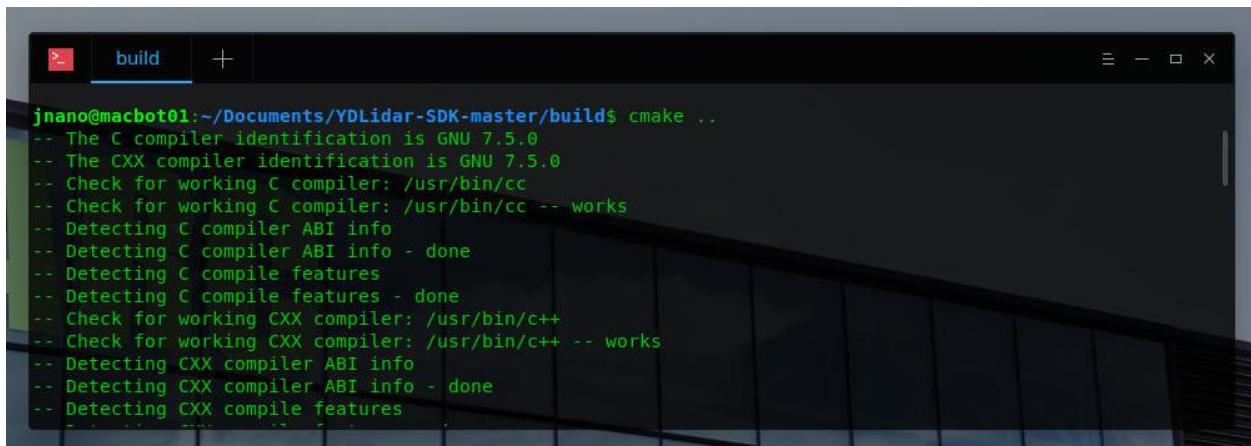
Specify a source directory to (re-)generate a build system for it in the
current working directory. Specify an existing build directory to
re-generate its build system.

Run 'cmake --help' for more information.

jnano@macbot01:~/Documents/YDLidar-SDK-master/build$
```

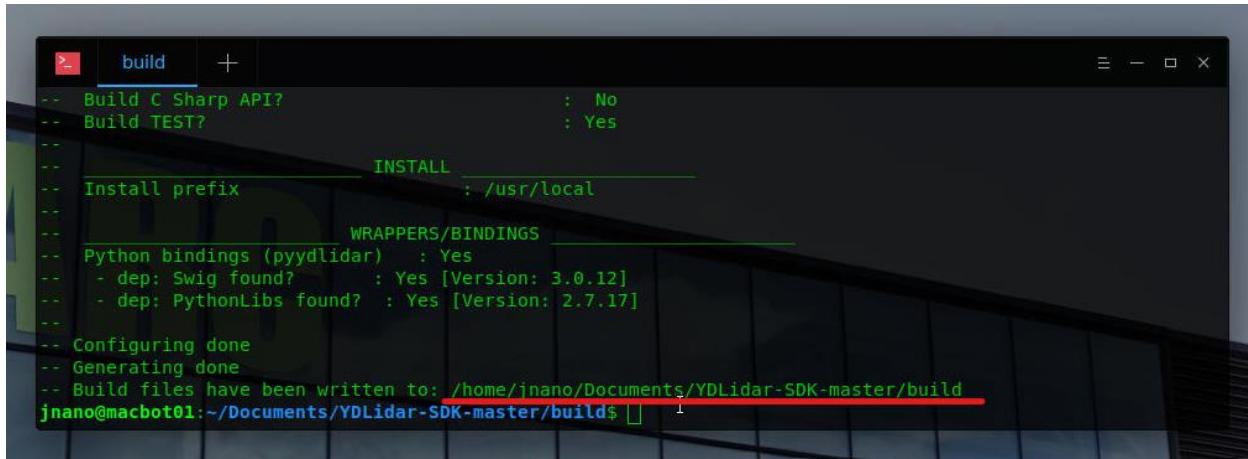
In our case, the **build/** directory is our target and the source is the **YDLidar-SDK-master** directory which is one level above.

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



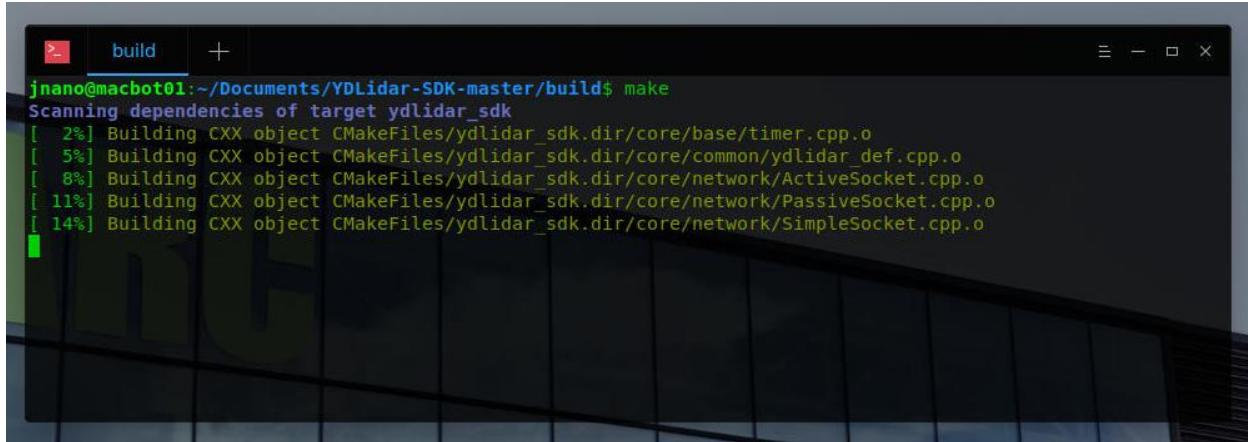
```
jnano@macbot01:~/Documents/YDLidar-SDK-master/build$ cmake ..
-- The C compiler identification is GNU 7.5.0
-- The CXX compiler identification is GNU 7.5.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
```

Ensure that build files have been generated correctly.



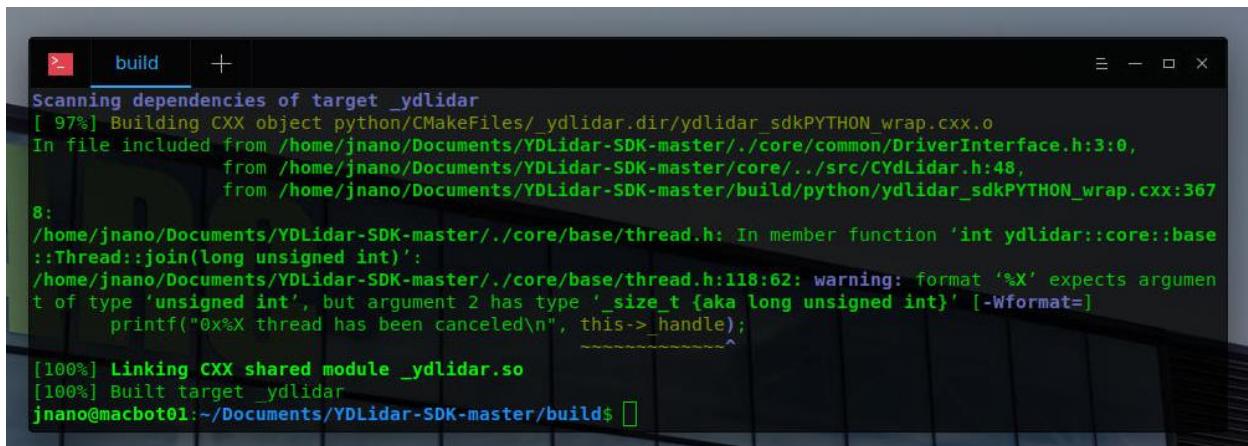
```
-- Build C Sharp API? : No
-- Build TEST? : Yes
-- INSTALL
-- Install prefix : /usr/local
-- WRAPPERS/BINDINGS
-- Python bindings (pyydlidar) : Yes
-- - dep: Swig found? : Yes [Version: 3.0.12]
-- - dep: PythonLibs found? : Yes [Version: 2.7.17]
-- Configuring done
-- Generating done
-- Build files have been written to: /home/jnano/Documents/YDLidar-SDK-master/build
jnano@macbot01:~/Documents/YDLidar-SDK-master/build$
```

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



```
jnano@macbot01:~/Documents/YDLidar-SDK-master/build$ make
Scanning dependencies of target ydlidar_sdk
[ 2%] Building CXX object CMakeFiles/ydlidar_sdk.dir/core/base/timer.cpp.o
[ 5%] Building CXX object CMakeFiles/ydlidar_sdk.dir/core/common/ydlidar_def.cpp.o
[ 8%] Building CXX object CMakeFiles/ydlidar_sdk.dir/core/network/ActiveSocket.cpp.o
[11%] Building CXX object CMakeFiles/ydlidar_sdk.dir/core/network/PassiveSocket.cpp.o
[14%] Building CXX object CMakeFiles/ydlidar_sdk.dir/core/network/SimpleSocket.cpp.o
```

Ensure it builds without fatal errors.

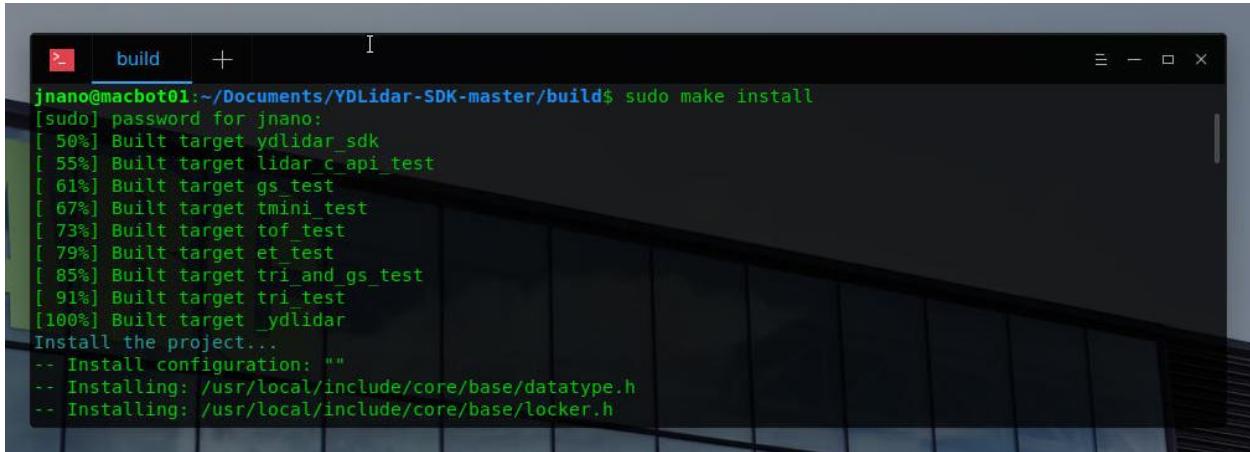


```
Scanning dependencies of target _ydlidar
[ 97%] Building CXX object python/CMakeFiles/_ydlidar.dir/ydlidar_sdkPYTHON_wrap.cxx.o
In file included from /home/jnano/Documents/YDLidar-SDK-master./core/common/DriverInterface.h:3:0,
                 from /home/jnano/Documents/YDLidar-SDK-master/core/../src/CYdLidar.h:48,
                 from /home/jnano/Documents/YDLidar-SDK-master/build/python/ydlidar_sdkPYTHON_wrap.cxx:367
8:
/home/jnano/Documents/YDLidar-SDK-master./core/base/thread.h: In member function 'int ydlidar::core::base
::Thread::join(Long unsigned int)':
/home/jnano/Documents/YDLidar-SDK-master./core/base/thread.h:118:62: warning: format '%X' expects argument
of type 'unsigned int', but argument 2 has type '_size_t {aka long unsigned int}' [-Wformat=]
    printf("0x%X thread has been canceled\n", this->_handle);
                                                               ^
[100%] Linking CXX shared module _ydlidar.so
[100%] Built target _ydlidar
jnano@macbot01:~/Documents/YDLidar-SDK-master/build$
```

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.

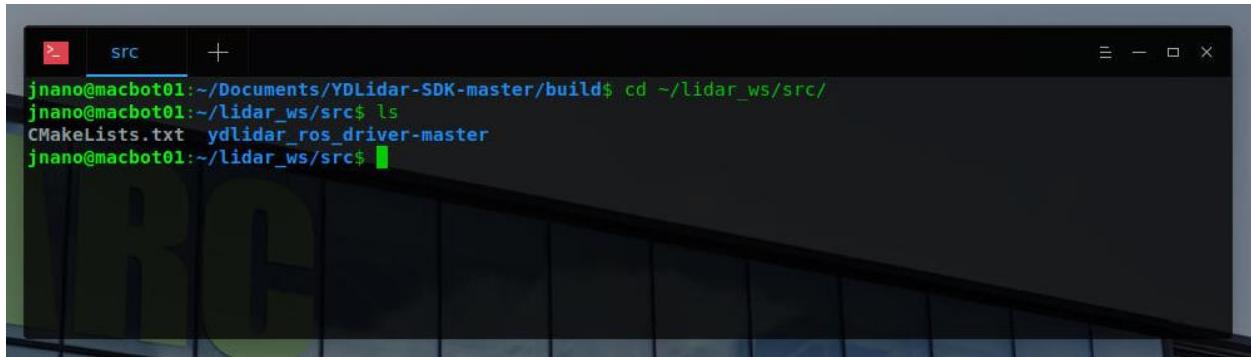


A screenshot of a terminal window titled "build". The window shows the command `sudo make install` being run in a directory under `~/Documents/YDLidar-SDK-master/build$`. The output of the command is displayed in green text, showing the progress of building various targets (ydlidar_sdk, lidar_c_api_test, gs_test, tmini_test, tof_test, et_test, tri_and_gs_test, tri_test, and _ydlidar) and the final installation step, which installs headers to `/usr/local/include/core/base`.

```
jnano@macbot01:~/Documents/YDLidar-SDK-master/build$ sudo make install
[sudo] password for jnano:
[ 50%] Built target ydlidar_sdk
[ 55%] Built target lidar_c_api_test
[ 61%] Built target gs_test
[ 67%] Built target tmini_test
[ 73%] Built target tof_test
[ 79%] Built target et_test
[ 85%] Built target tri_and_gs_test
[ 91%] Built target tri_test
[100%] Built target _ydlidar
Install the project...
-- Install configuration: ""
-- Installing: /usr/local/include/core/base/datatype.h
-- Installing: /usr/local/include/core/base/locker.h
```

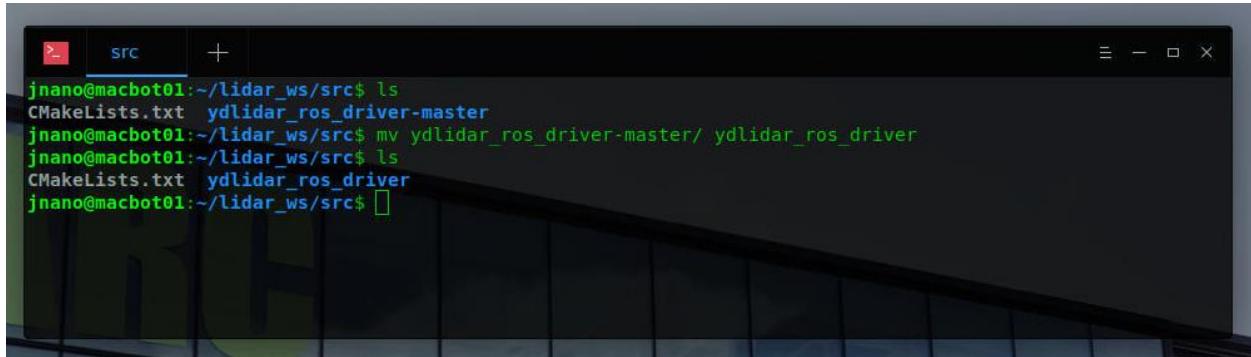
Building the ROS YDLiDAR Package

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



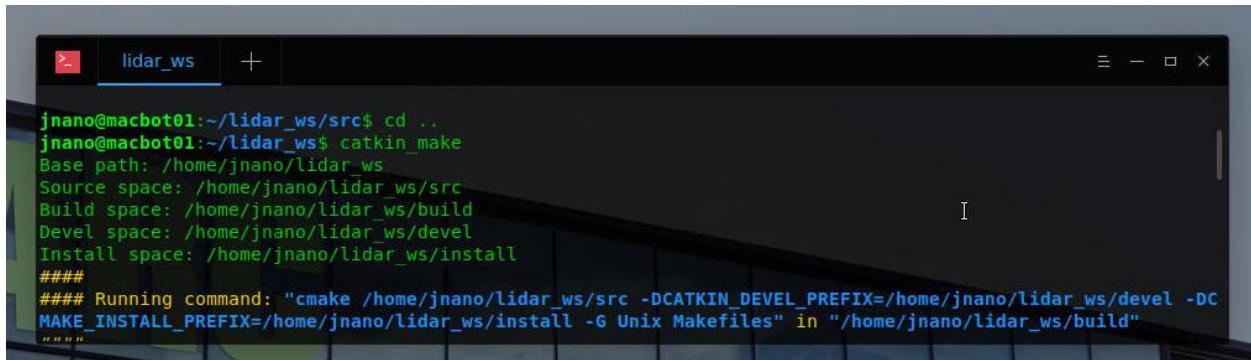
```
jnano@macbot01:~/Documents/YDLidar-SDK-master/build$ cd ~/lidar_ws/src/
jnano@macbot01:~/lidar_ws/src$ ls
CMakeLists.txt  ydlidar_ros_driver-master
jnano@macbot01:~/lidar_ws/src$
```

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



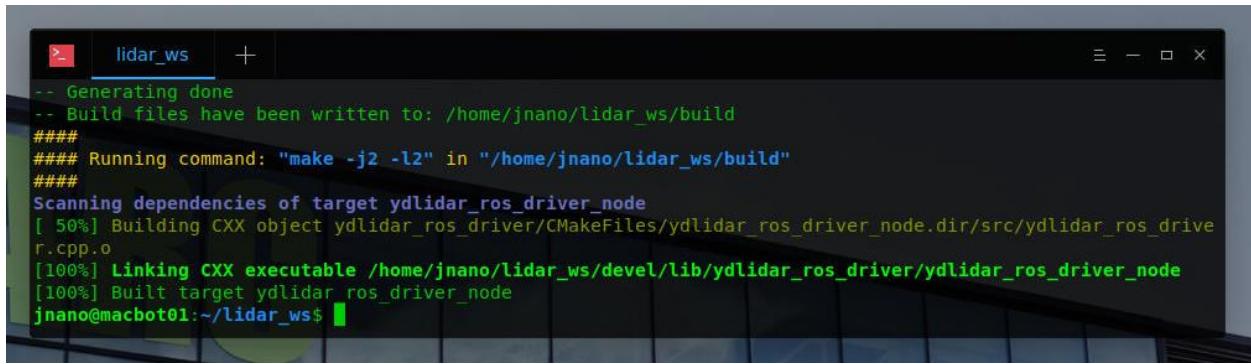
```
jnano@macbot01:~/lidar_ws/src$ ls
CMakeLists.txt  ydlidar_ros_driver-master
jnano@macbot01:~/lidar_ws/src$ mv ydlidar_ros_driver-master/ ydlidar_ros_driver
jnano@macbot01:~/lidar_ws/src$ ls
CMakeLists.txt  ydlidar_ros_driver
jnano@macbot01:~/lidar_ws/src$
```

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



```
jnano@macbot01:~/lidar_ws/src$ cd ..
jnano@macbot01:~/lidar_ws$ catkin_make
Base path: /home/jnano/lidar_ws
Source space: /home/jnano/lidar_ws/src
Build space: /home/jnano/lidar_ws/build
Devel space: /home/jnano/lidar_ws/devel
Install space: /home/jnano/lidar_ws/install
#####
#### Running command: "cmake /home/jnano/lidar_ws/src -DCATKIN_DEVEL_PREFIX=/home/jnano/lidar_ws/devel -DCMAKE_INSTALL_PREFIX=/home/jnano/lidar_ws/install -G Unix Makefiles" in "/home/jnano/lidar_ws/build"
####
```

Ensure that the build was successful.

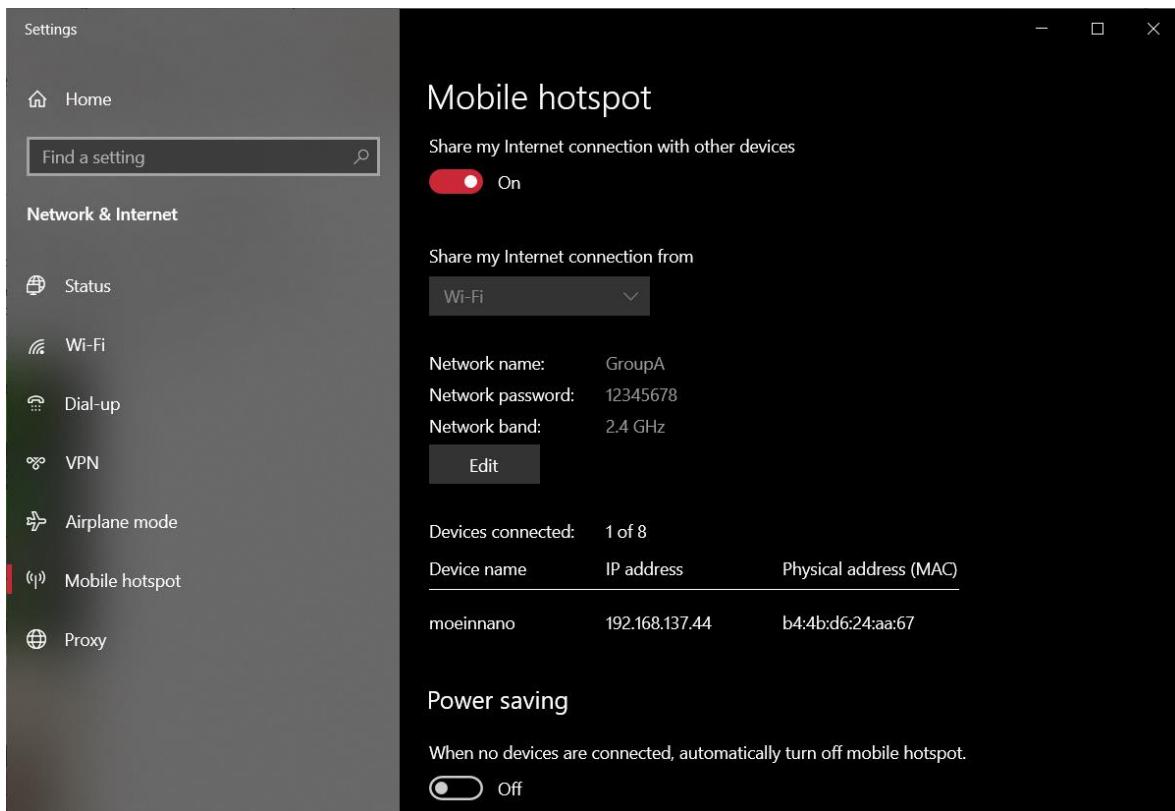


```
-- Generating done
-- Build files have been written to: /home/jnano/lidar_ws/build
#####
##### Running command: "make -j2 -l2" in "/home/jnano/lidar_ws/build"
#####
Scanning dependencies of target ydlidar_ros_driver_node
[ 50%] Building CXX object ydlidar_ros_driver/CMakeFiles/ydlidar_ros_driver_node.dir/src/ydlidar_ros_driver_node.cpp.o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

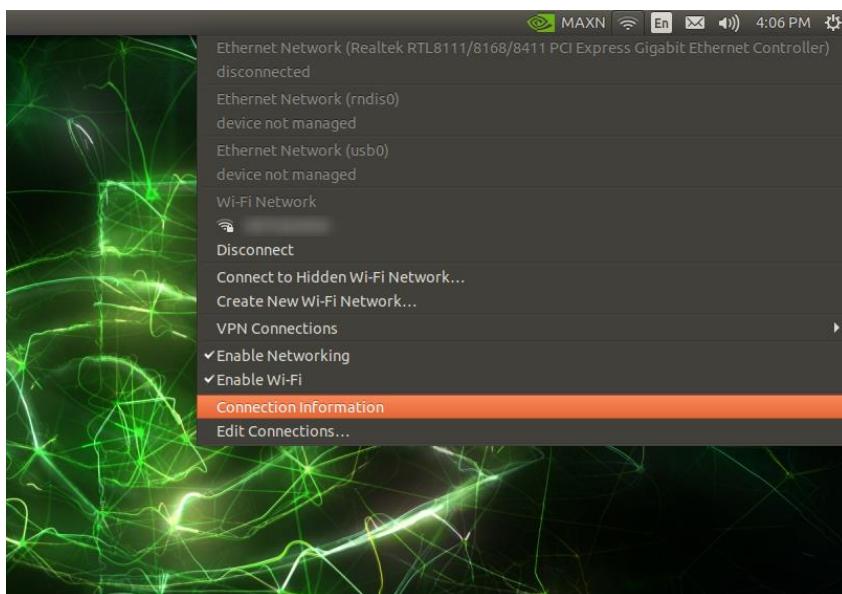
Capturing and Visualizing LiDAR Data

Connecting the YD LiDAR X2

Connecting to a Local Network

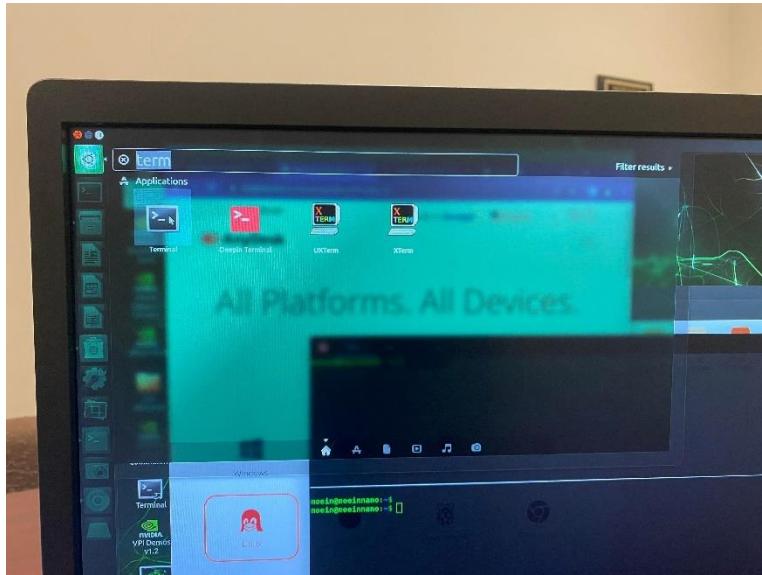


Back in Ubuntu, **select** the hosted WiFi network.



Updating Packages

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



Run the following command:

```
sudo apt update
```

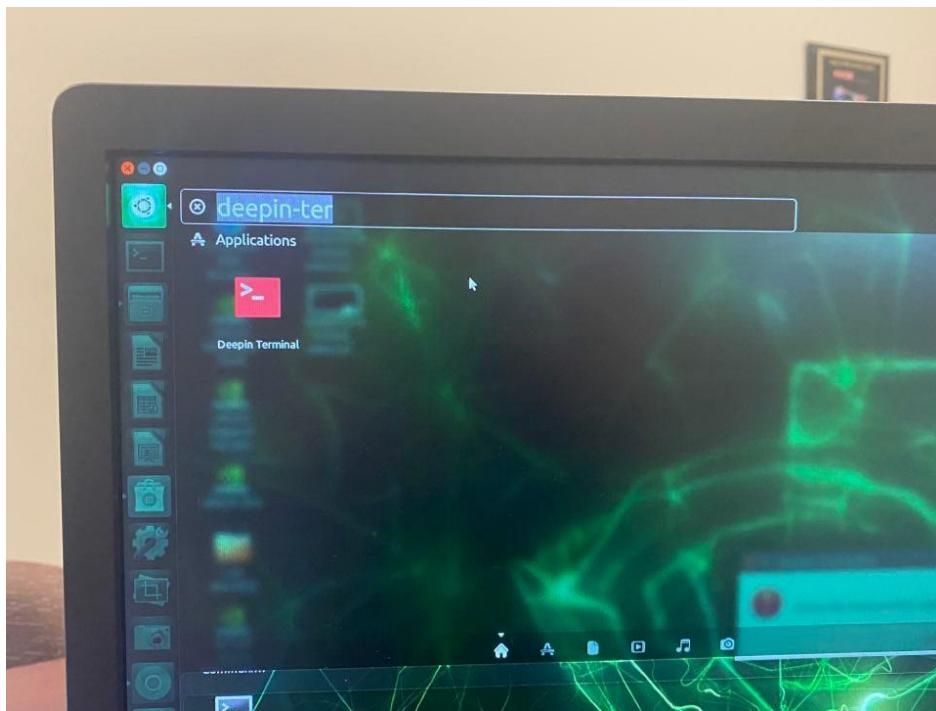
```
moein@moeinnano:~$ sudo apt update
Get:1 file:/var/cuda-repo-l4t-10-2-local InRelease
Ign:1 file:/var/cuda-repo-l4t-10-2-local InRelease
Get:2 file:/var/visionworks-repo InRelease
Ign:2 file:/var/visionworks-repo InRelease
Get:3 file:/var/visionworks-sfm-repo InRelease
Ign:3 file:/var/visionworks-sfm-repo 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 InRelease
```

Installing Deepin-Terminal

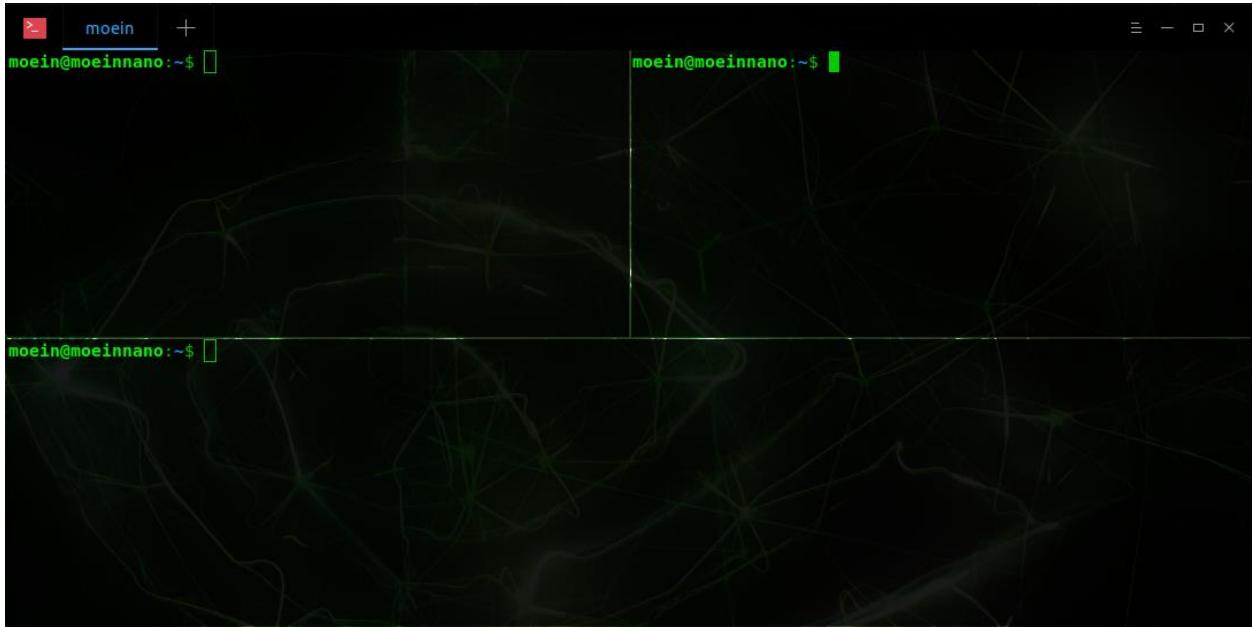
When working with ROS, it is useful to be able to split a terminal window into multiple horizontal and vertical windows or create new terminal tabs. This is because ROS requires many scripts to run concurrently and navigating between many open windows becomes tedious and difficult to navigate.

```
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

```
moein@moeinnano:~$ rosversion -d  
melodic  
moein@moeinnano:~$
```

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

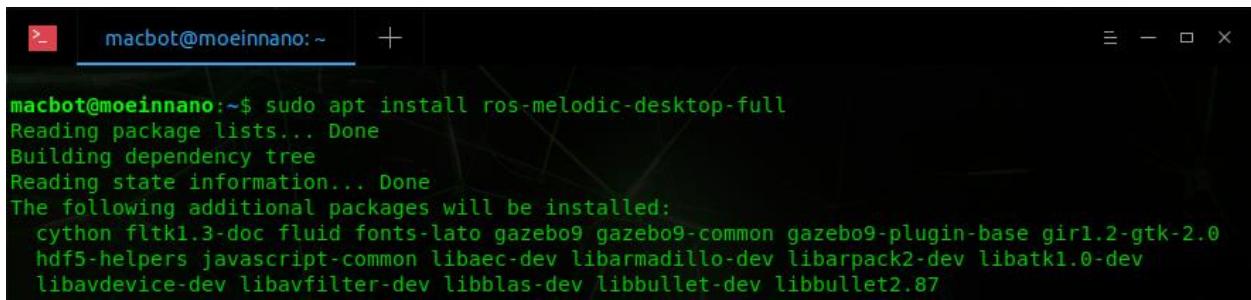
```
macbot@moeinnano:~$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
macbot@moeinnano:~$ sudo apt install curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  default-libmysqlclient-dev google-mock gtest libhddtemp0 libapr1 libapr1-dev libaprutil1
  libaprutil1-dev libassimp-dev libassimp4 libassuan libboost-all-dev libboost-atomic-dev
```

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

sudo apt update

```
macbot@moeinnano:~$ curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
OK
macbot@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
```

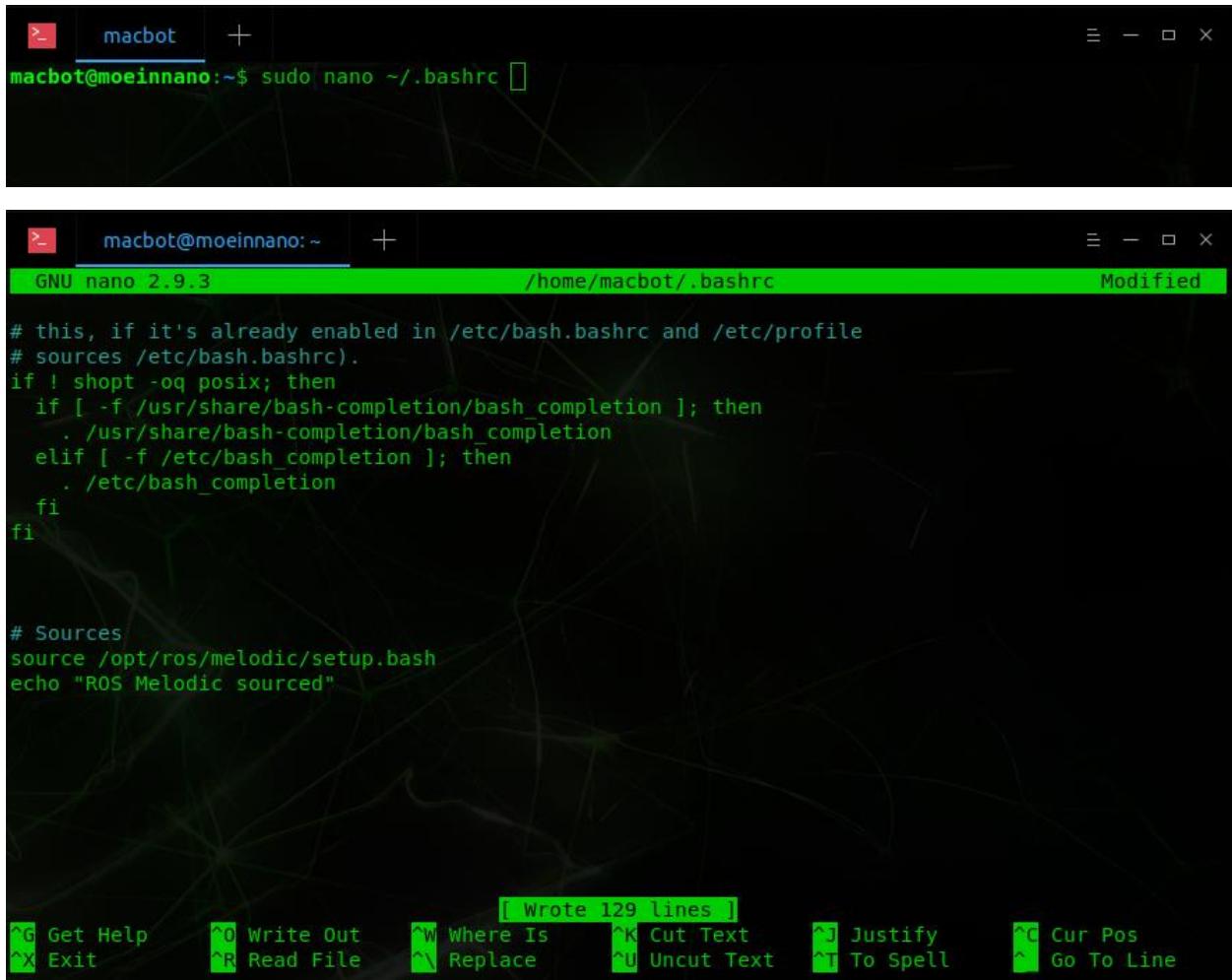
```
sudo apt install ros-melodic-desktop-full
```



```
macbot@moeinnano:~$ sudo apt install ros-melodic-desktop-full
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cython fltk1.3-doc fluid fonts-lato gazebo9 gazebo9-common gazebo9-plugin-base gir1.2-gtk-2.0
  hdf5-helpers javascript-common libaec-dev libarmadillo-dev libarpack2-dev libatk1.0-dev
  libavdevice-dev libavfilter-dev libblas-dev libbullet-dev libbullet2.87
```

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
```



```
macbot@moeinnano:~$ sudo nano ~/.bashrc
```

```
GNU nano 2.9.3          /home/macbot/.bashrc          Modified

# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# Sources
source /opt/ros/melodic/setup.bash
echo "ROS Melodic sourced"
```

[Wrote 129 lines]

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

Add the source for ROS melodic.

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

To exit nano, use the following commands:

CTRL + S

CTRL + X

Y

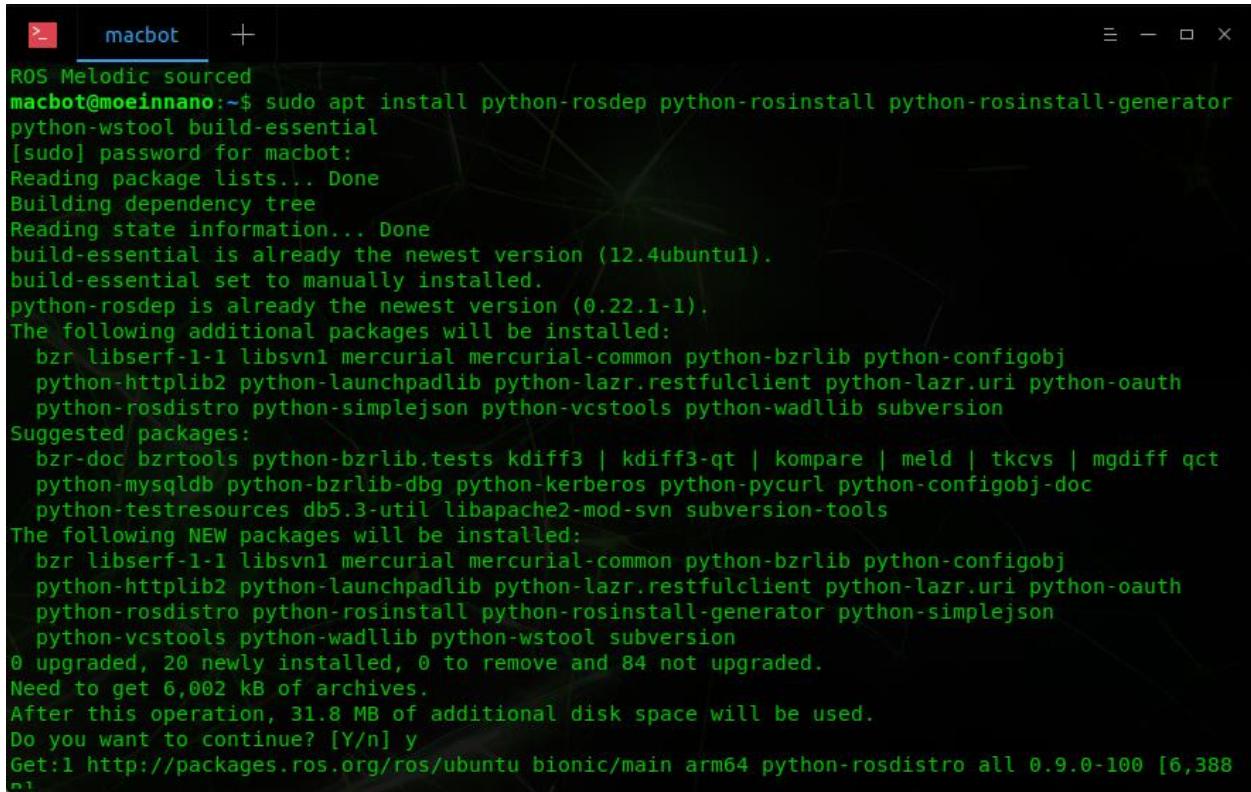
<Enter>

Open a new terminal window and see if any errors occur. If 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
```

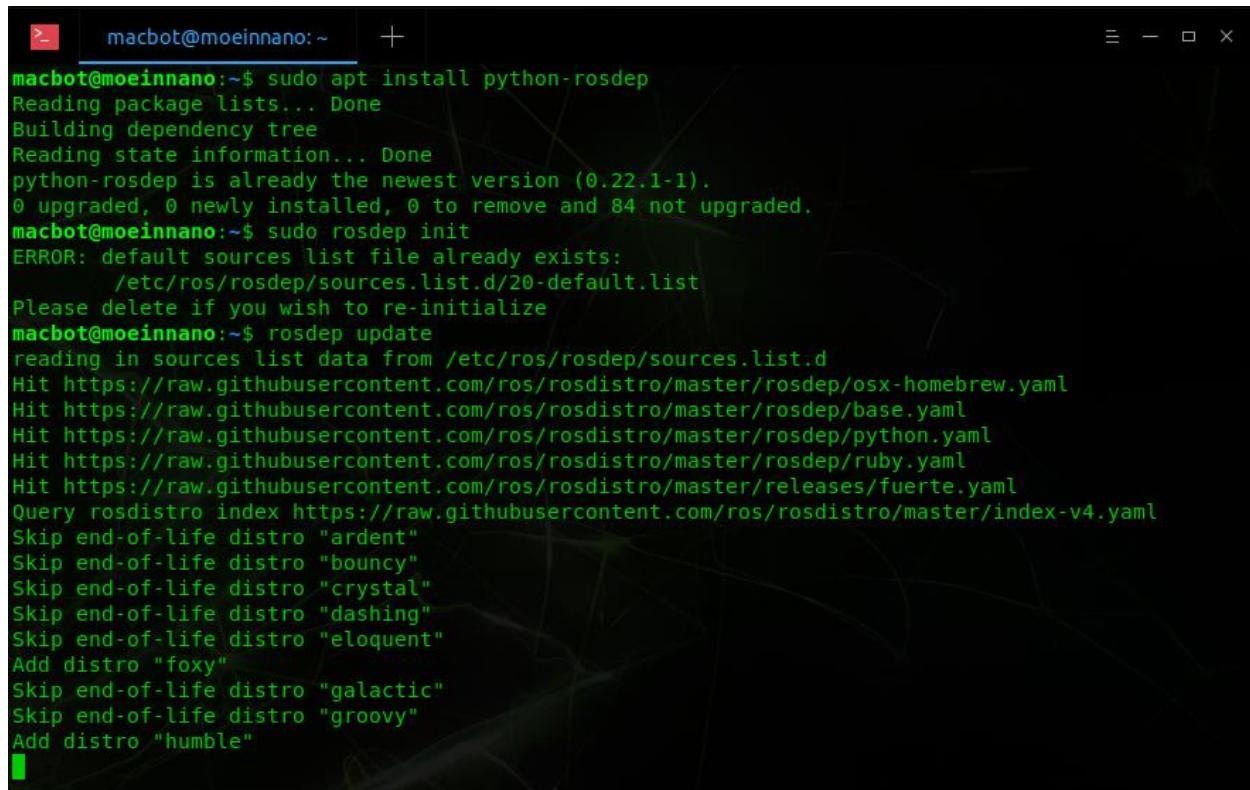


Lastly, initialize rosdep.

```
sudo apt install python-rosdep
```

```
sudo rosdep init
```

```
rosdep update
```



```
macbot@moeinnano:~$ sudo apt install python-rosdep
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-rosdep is already the newest version (0.22.1-1).
0 upgraded, 0 newly installed, 0 to remove and 84 not upgraded.
macbot@moeinnano:~$ sudo rosdep init
ERROR: default sources list file already exists:
/etc/ros/rosdep/sources.list.d/20-default.list
Please delete if you wish to re-initialize
macbot@moeinnano:~$ rosdep update
reading in sources list data from /etc/ros/rosdep/sources.list.d
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-homebrew.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/python.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/ruby.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/releases/fuerte.yaml
Query rosdistro index https://raw.githubusercontent.com/ros/rosdistro/master/index-v4.yaml
Skip end-of-life distro "ardent"
Skip end-of-life distro "bouncy"
Skip end-of-life distro "crystal"
Skip end-of-life distro "dashing"
Skip end-of-life distro "eloquent"
Add distro "foxy"
Skip end-of-life distro "galactic"
Skip end-of-life distro "groovy"
Add distro "humble"
```

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
```

```
lidar_ws +  
macbot@moeinnano:~$ ls  
Desktop Documents Downloads examples.desktop Music Pictures Public Templates Videos  
macbot@moeinnano:~$ mkdir lidar_ws  
macbot@moeinnano:~$ cd lidar_ws/  
macbot@moeinnano:~/lidar_ws$ mkdir src  
macbot@moeinnano:~/lidar_ws$ catkin_make  
Base path: /home/macbot/lidar_ws  
Source space: /home/macbot/lidar_ws/src  
Build space: /home/macbot/lidar_ws/build  
Devel space: /home/macbot/lidar_ws/devel  
Install space: /home/macbot/lidar_ws/install  
Creating symlink "/home/macbot/lidar_ws/src/CMakeLists.txt" pointing to "/opt/ros/melodic/share/catkin/cmake/toplevel.cmake"  
####  
### Running command: "cmake /home/macbot/lidar_ws/src -DCATKIN_DEVEL_PREFIX=/home/macbot/lidar_ws/devel -DCMAKE_INSTALL_PREFIX=/home/macbot/lidar_ws/install -G Unix Makefiles" in "/home/macbot/lidar_ws/build"  
####  
-- The C compiler identification is GNU 7.5.0  
-- The CXX compiler identification is GNU 7.5.0  
-- Check for working C compiler: /usr/bin/cc  
-- Check for working C compiler: /usr/bin/cc -- works  
-- Detecting C compiler ABI info  
-- Detecting C compiler ABI info - done  
-- Detecting C compile features  
-- Detecting C compile features - done  
-- Check for working CXX compiler: /usr/bin/c++  
-- Check for working CXX compiler: /usr/bin/c++ -- works
```

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

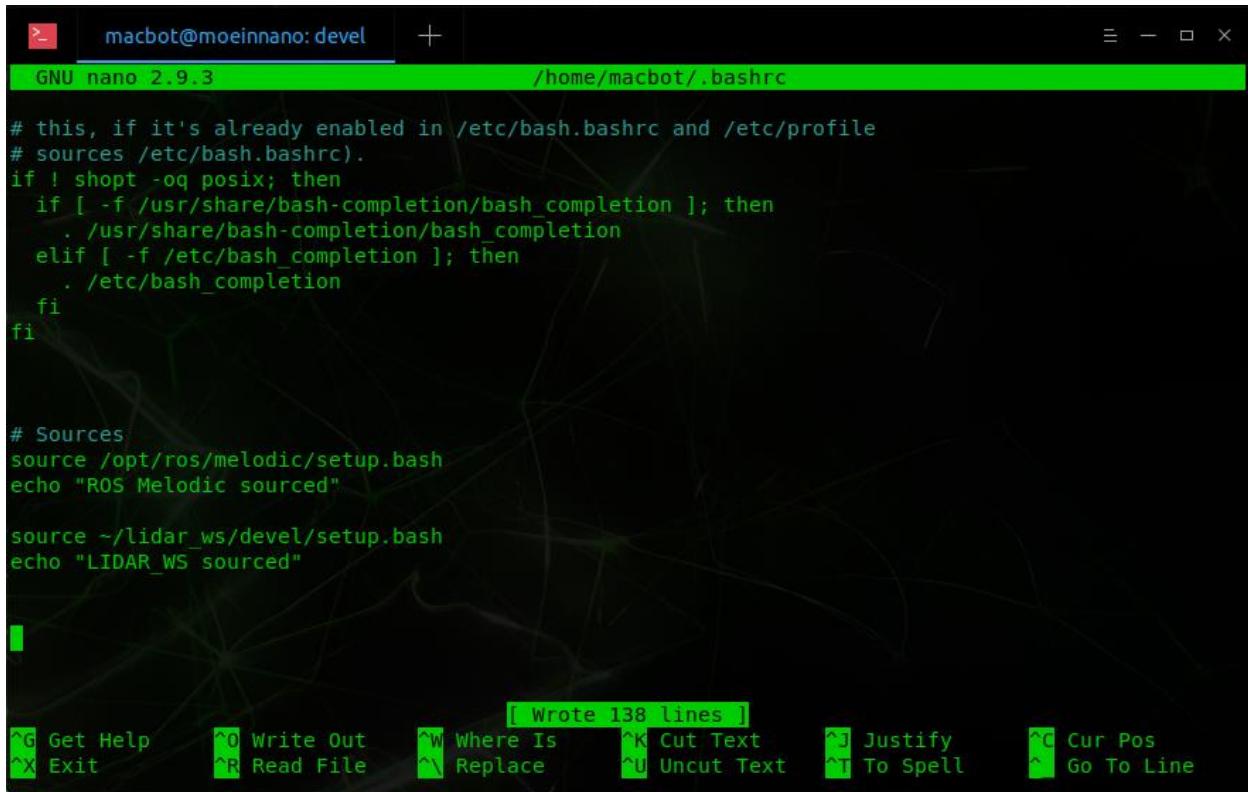
```
devel +  
macbot@moeinnano:~/lidar_ws$ ls  
build devel src  
macbot@moeinnano:~/lidar_ws$ cd devel/  
macbot@moeinnano:~/lidar_ws/devel$ ls  
cmake.lock lib local_setup.sh setup.bash _setup_util.py  
env.sh local_setup.bash local_setup.zsh setup.sh setup.zsh  
macbot@moeinnano:~/lidar_ws/devel$
```

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"
```



```
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

# Sources
source /opt/ros/melodic/setup.bash
echo "ROS Melodic sourced"

source ~/lidar_ws/devel/setup.bash
echo "LIDAR_WS sourced"
```

[Wrote 138 lines]

^G Get Help **^O** Write Out **^W** Where Is **^K** Cut Text **^J** Justify **^C** Cur Pos
^X Exit **^R** Read File **^** Replace **^U** Uncut Text **^T** To Spell **^** Go To Line

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.



```
ROS Melodic sourced
LIDAR_WS sourced
macbot@moeinnano:~/lidar_ws/devel$
```

Lidar Setup

Cloning the YDLidar ROS Package

```
cd ~/lidar_ws/src
```

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

YDLIDAR / ydlidar_ros_driver (Public)

Code Issues 6 Pull requests 2 Actions Projects Security Insights

master 5 branches 0 tags

YistXin 修改GS1.launch和GS2.launch的默认距离最小值为规格书中的最小量程 d487d79 on Dec 18, 2022 15 commits

File	Commit Message	Date
images	feat(*): Update RRADNE.md	2 years ago
launch	修改GS1.launch和GS2.launch的默认距离最小值为规格书中的最小量程	last month
msg	feature(*):add ydlidar_ros_driver package	2 years ago
src	1.Modified the problem that the fixed resolution cannot be set. resul...	last year
startup	feature(*):add ydlidar_ros_driver package	2 years ago
CMakeLists.txt	1.add point_cloud_preservative param to launch.	2 years ago
LICENSE.txt	feature(*):add ydlidar_ros_driver package	2 years ago
README.md	修改md文件	7 months ago
details.md	feat(*): Update RRADNE.md	2 years ago
package.xml	1.add point_cloud_preservative param to launch.	2 years ago

```
macbot@moeinnano:~$ cd lidar_ws/src/
macbot@moeinnano:~/lidar_ws/src$ git clone https://github.com/YDLIDAR/ydlidar_ros_driver.git
Cloning into 'ydlidar_ros_driver'...
remote: Enumerating objects: 181, done.
remote: Counting objects: 100% (181/181), done.
remote: Compressing objects: 100% (107/107), done.
remote: Total 181 (delta 101), reused 148 (delta 71), pack-reused 0
Receiving objects: 100% (181/181), 778.51 KiB | 2.27 MiB/s, done.
Resolving deltas: 100% (101/101), done.
macbot@moeinnano:~/lidar_ws/src$
```

Navigate back to `~/lidar_ws`.

```
lidar_ws +  
macbot@moeinnano:~/lidar_ws/src$ ls -la  
total 12  
drwxrwxr-x 3 macbot macbot 4096 Jan 31 17:12 .  
drwxrwxr-x 5 macbot macbot 4096 Jan 31 16:39 ..  
lrwxrwxrwx 1 macbot macbot 50 Jan 31 16:39 CMakeLists.txt -> /opt/ros/melodic/share/catkin/cmake/toplevel.cmake  
drwxrwxr-x 8 macbot macbot 4096 Jan 31 17:13 ydlidar_ros_driver  
macbot@moeinnano:~/lidar_ws/src$ cd ..  
macbot@moeinnano:~/lidar_ws$ ls  
build  devel  src  
macbot@moeinnano:~/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.

```
lidar_ws +  
-- ~ - ydlidar_ros_driver  
--  
-- +++ processing catkin package: 'ydlidar_ros_driver'  
-- ==> add subdirectory(ydlidar_ros_driver)  
CMake Error at ydlidar_ros_driver/CMakeLists.txt:9 (find_package):  
By not providing "Findydlidar_sdk.cmake" in CMAKE_MODULE_PATH this project  
has asked CMake to find a package configuration file provided by  
"ydlidar_sdk", but CMake did not find one.  
  
Could not find a package configuration file provided by "ydlidar_sdk" with  
any of the following names:  
  
    ydlidar_sdkConfig.cmake  
    ydlidar_sdk-config.cmake  
  
Add the installation prefix of "ydlidar_sdk" to CMAKE_PREFIX_PATH or set  
"ydlidar_sdk_DIR" to a directory containing one of the above files. If  
"ydlidar_sdk" provides a separate development package or SDK, be sure it  
has been installed.  
  
-- Configuring incomplete, errors occurred!  
See also "/home/macbot/lidar_ws/build/CMakeFiles/CMakeOutput.log".  
See also "/home/macbot/lidar_ws/build/CMakeFiles/CMakeError.log".  
Makefile:320: recipe for target 'cmake_check_build_system' failed  
make: *** [cmake_check_build_system] Error 1  
Invoking "make cmake_check_build_system" failed  
macbot@moeinnano:~/lidar_ws$
```

YDLiDAR SDK

Navigate to the following GitHub repo:

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

Copy its git URL.

YDLIDAR / YDLidar-SDK (Public)

Code Issues 19 Pull requests 4 Actions Projects Security Insights

master 2 branches 11 tags

zhanyaini 修改GS2和S2系列同时使用时线程异常问题

- cmake The problem of intensity bit resolution
- core 修改GS2和S2系列同时使用时线程异常问题
- csharp 解决了加密的问题。
- doc 修改适配T-mini Pro
- python The problem of intensity bit resolution
- samples 修改GS2和S2系列同时使用时线程异常问题
- src 修改GS2和S2系列同时使用时线程异常问题
- startup The problem of intensity bit resolution of S2 lidar is solved

Local Codespaces (New)

Clone

HTTPS SSH GitHub CLI

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

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

last year

HTTPS SSH GitHub CLI

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

Copied!

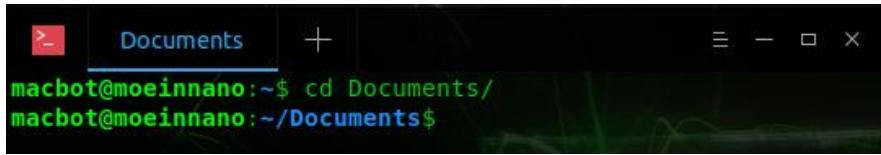
Use Git or checkout with SVN using the web URL.

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

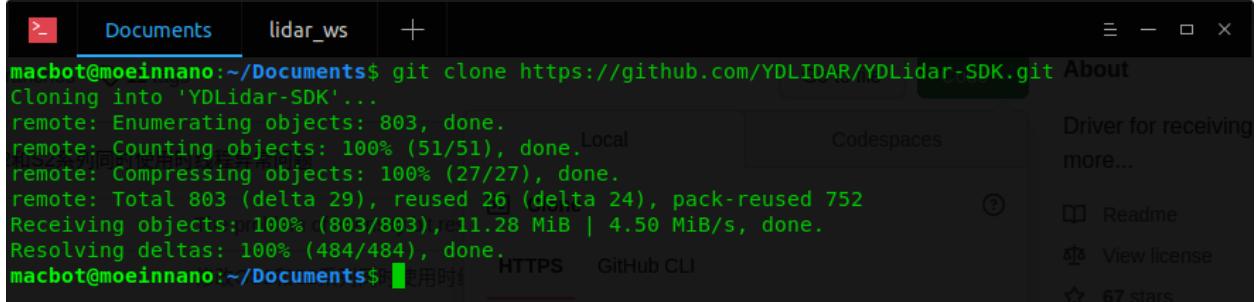
```
macbot@moeinnano:~$ sudo apt install cmake pkg-config
[sudo] password for macbot:
Reading package lists... Done
Building dependency tree...
Reading state information... Done
pkg-config is already the newest version (0.29.1-0ubuntu2).
cmake is already the newest version (3.10.2-1ubuntu2.18.04.2).
The following package was automatically installed and is no longer required:
  libostree-1-1
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 85 not upgraded.
macbot@moeinnano:~$
```

sudo apt install cmake pkg-config

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

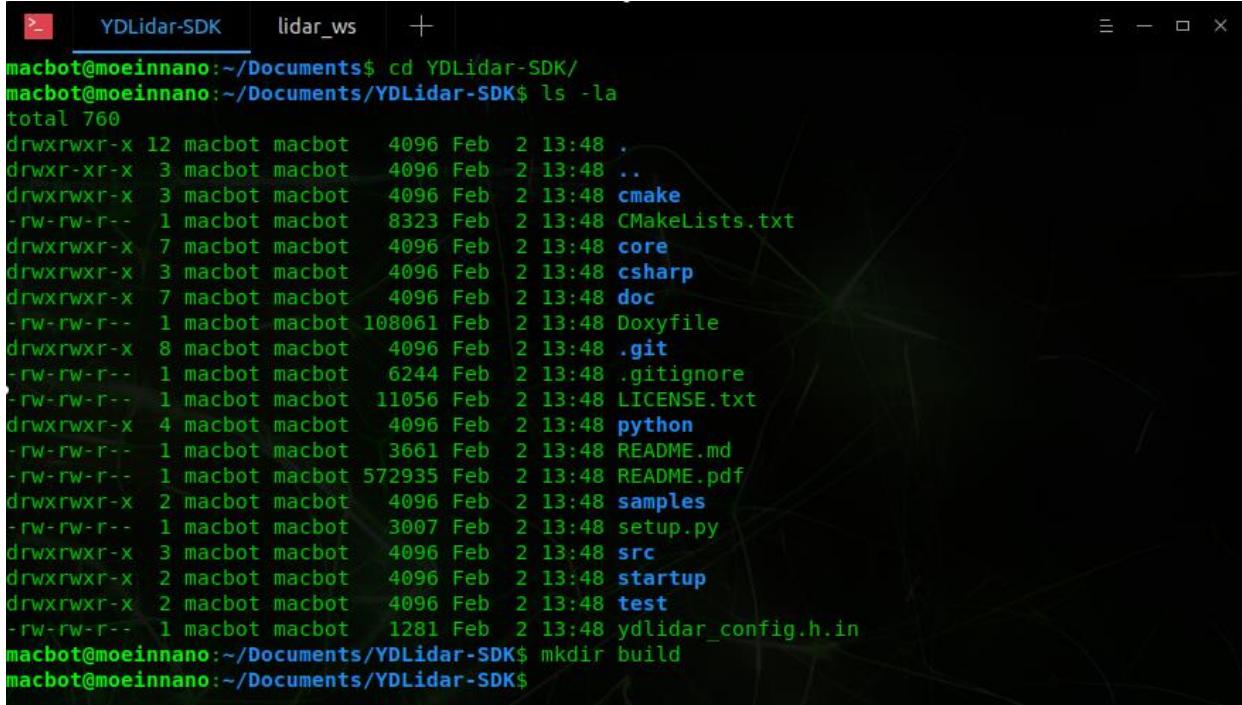


```
macbot@moeinnano:~$ cd Documents/
macbot@moeinnano:~/Documents$
```



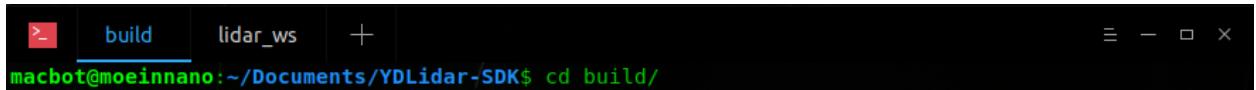
```
macbot@moeinnano:~/Documents$ git clone https://github.com/YDLIDAR/YDLidar-SDK.git
Cloning into 'YDLidar-SDK'...
remote: Enumerating objects: 803, done.
remote: Counting objects: 100% (51/51), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 803 (delta 29), reused 26 (delta 24), pack-reused 752
Receiving objects: 100% (803/803), 11.28 MiB | 4.50 MiB/s, done.
Resolving deltas: 100% (484/484), done.
macbot@moeinnano:~/Documents$
```

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



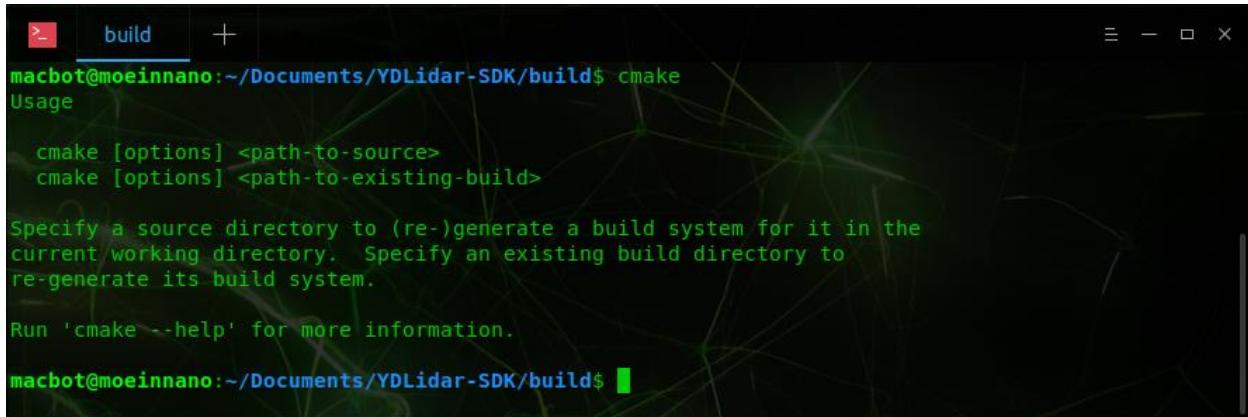
```
macbot@moeinnano:~/Documents$ cd YDLidar-SDK/
macbot@moeinnano:~/Documents/YDLidar-SDK$ ls -la
total 760
drwxrwxr-x 12 macbot macbot 4096 Feb  2 13:48 .
drwxr-xr-x  3 macbot macbot 4096 Feb  2 13:48 ..
drwxrwxr-x  3 macbot macbot 4096 Feb  2 13:48 cmake
-rw-rw-r--  1 macbot macbot 8323 Feb  2 13:48 CMakeLists.txt
drwxrwxr-x  7 macbot macbot 4096 Feb  2 13:48 core
drwxrwxr-x  3 macbot macbot 4096 Feb  2 13:48 csharp
drwxrwxr-x  7 macbot macbot 4096 Feb  2 13:48 doc
-rw-rw-r--  1 macbot macbot 108061 Feb  2 13:48 Doxyfile
drwxrwxr-x  8 macbot macbot 4096 Feb  2 13:48 .git
-rw-rw-r--  1 macbot macbot 6244 Feb  2 13:48 .gitignore
-rw-rw-r--  1 macbot macbot 11056 Feb  2 13:48 LICENSE.txt
drwxrwxr-x  4 macbot macbot 4096 Feb  2 13:48 python
-rw-rw-r--  1 macbot macbot 3661 Feb  2 13:48 README.md
-rw-rw-r--  1 macbot macbot 572935 Feb  2 13:48 README.pdf
drwxrwxr-x  2 macbot macbot 4096 Feb  2 13:48 samples
-rw-rw-r--  1 macbot macbot 3007 Feb  2 13:48 setup.py
drwxrwxr-x  3 macbot macbot 4096 Feb  2 13:48 src
drwxrwxr-x  2 macbot macbot 4096 Feb  2 13:48 startup
drwxrwxr-x  2 macbot macbot 4096 Feb  2 13:48 test
-rw-rw-r--  1 macbot macbot 1281 Feb  2 13:48 ydlidar_config.h.in
macbot@moeinnano:~/Documents/YDLidar-SDK$ mkdir build
macbot@moeinnano:~/Documents/YDLidar-SDK$
```

Navigate into the **build/** directory.



```
macbot@moeinnano:~/Documents/YDLidar-SDK$ cd build/
```

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



```
macbot@moeinnano:~/Documents/YDLidar-SDK/build$ cmake
Usage

  cmake [options] <path-to-source>
  cmake [options] <path-to-existing-build>

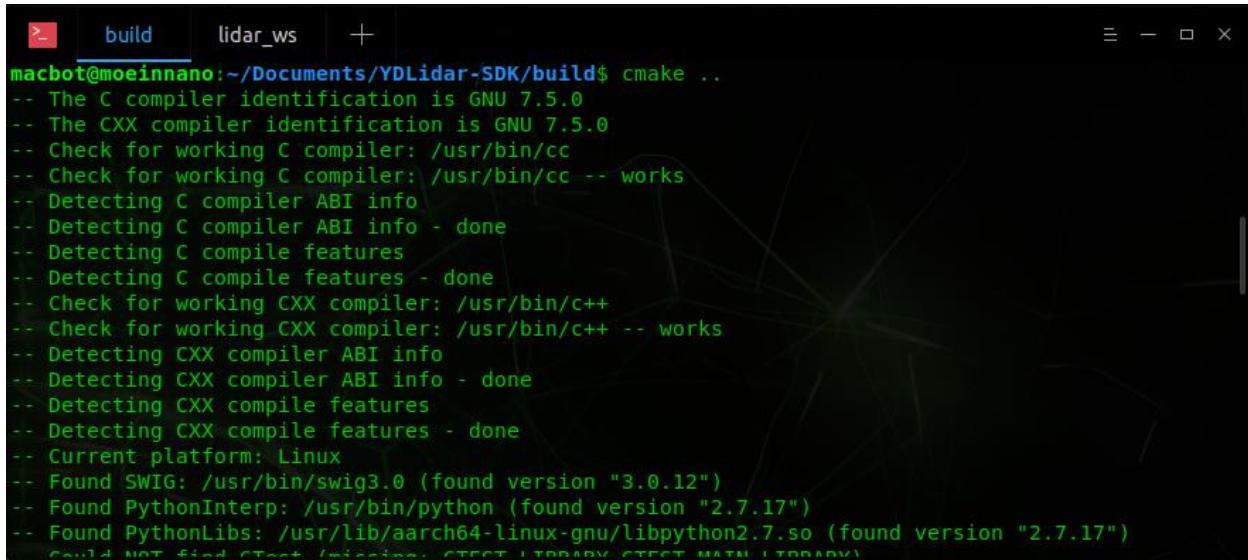
Specify a source directory to (re-)generate a build system for it in the
current working directory. Specify an existing build directory to
re-generate its build system.

Run 'cmake --help' for more information.

macbot@moeinnano:~/Documents/YDLidar-SDK/build$
```

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.



```
macbot@moeinnano:~/Documents/YDLidar-SDK/build$ cmake ..
-- The C compiler identification is GNU 7.5.0
-- The CXX compiler identification is GNU 7.5.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Current platform: Linux
-- Found SWIG: /usr/bin/swig3.0 (found version "3.0.12")
-- Found PythonInterp: /usr/bin/python (found version "2.7.17")
-- Found PythonLibs: /usr/lib/aarch64-linux-gnu/libpython2.7.so (found version "2.7.17")
-- Could NOT find CTEST (missing: CTEST_LIBRARY CTEST_MATH CTEST_PAPI)
```

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

```
build lidar_ws +  
-----+  
-- PLATFOR  
-- Host : Linux4.9.299-tegraaarch64  
-- Is the system big endian? : No  
-- Word size (32/64 bit) : 64  
-- CMake version : 3.10.2  
-- CMake generator : Unix Makefiles  
-- CMake build tool : /usr/bin/make  
-- Compiler : GNU  
-- Configuration :  
  
-- OPTIONS  
-- Build YDLidar-SDK as a shared library? : No  
-- Build Examples? : Yes  
-- Build C Sharp API? : No  
-- Build TEST? : Yes  
  
-- INSTALL  
-- Install prefix : /usr/local  
  
-- WRAPPERS/BINDINGS  
-- Python bindings (pyydlidar) : Yes  
- dep: Swig found? : Yes [Version: 3.0.12]  
- dep: PythonLibs found? : Yes [Version: 2.7.17]  
  
-- Configuring done  
-- Generating done  
-- Build files have been written to: /home/macbot/Documents/YDLidar-SDK/build  
macbot@moeinnano:~/Documents/YDLidar-SDK/build$
```

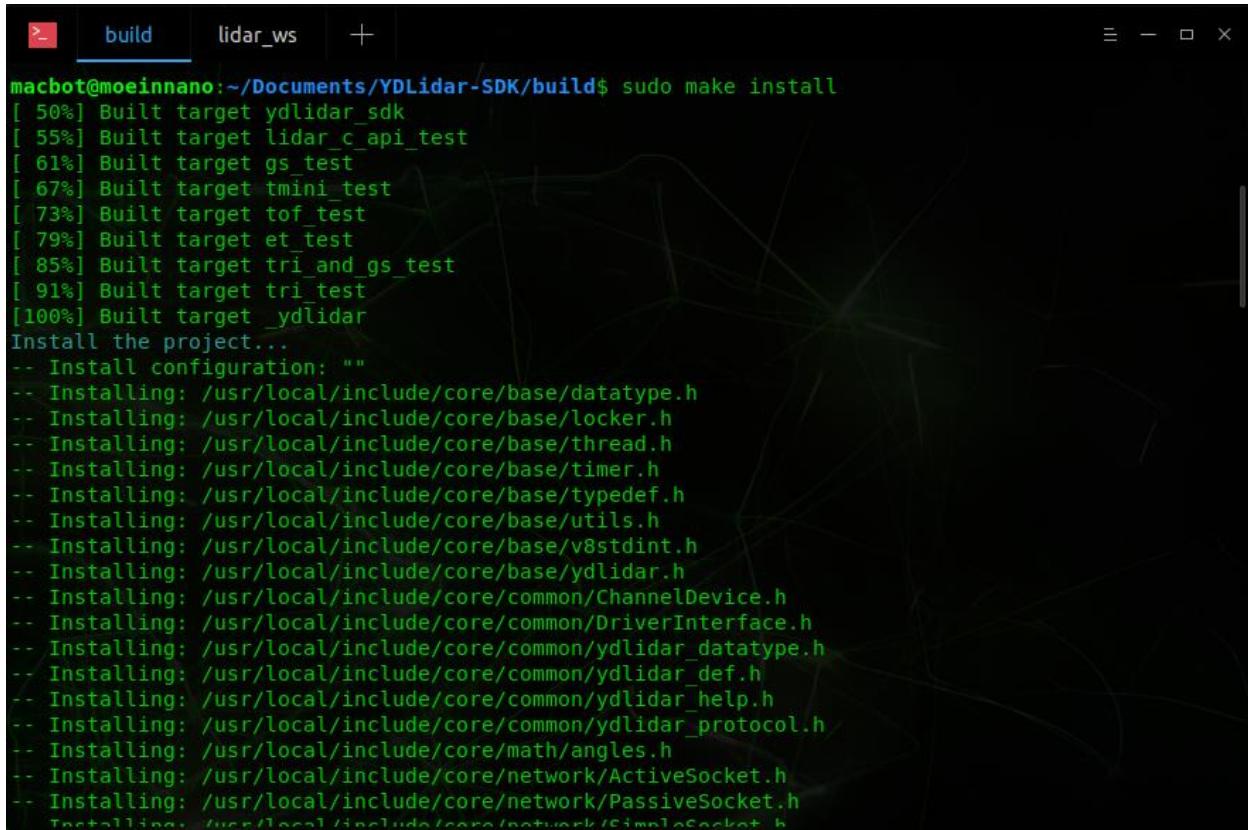
Next, make the SDK while being in the **build/** directory.

```
macbot@moeinnano:~/Documents/YDLidar-SDK/build$ make
Scanning dependencies of target ydlidar_sdk
[ 2%] Building CXX object CMakeFiles/ydlidar_sdk.dir/core/base/timer.cpp.o
[ 5%] Building CXX object CMakeFiles/ydlidar_sdk.dir/core/common/ydlidar_def.cpp.o
[ 8%] Building CXX object CMakeFiles/ydlidar_sdk.dir/core/network/ActiveSocket.cpp.o
[11%] Building CXX object CMakeFiles/ydlidar_sdk.dir/core/network/PassiveSocket.cpp.o
[14%] Building CXX object CMakeFiles/ydlidar_sdk.dir/core/network/SimpleSocket.cpp.o
[17%] Building CXX object CMakeFiles/ydlidar_sdk.dir/core/serial/serial.cpp.o
```

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

```
build +  
macbot@moeinnano:~/Documents/YDLidar-SDK/build$ ls -la  
total 6720  
drwxrwxr-x 7 macbot macbot 4096 Feb 2 13:59 .  
drwxrwxr-x 13 macbot macbot 4096 Feb 2 13:49 ..  
-rw-rw-r-- 1 macbot macbot 16010 Feb 2 13:53 CMakeCache.txt  
drwxrwxr-x 6 macbot macbot 4096 Feb 2 13:59 CMakeFiles  
-rw-rw-r-- 1 macbot macbot 28507 Feb 2 13:53 cmake_install.cmake  
-rw-rw-r-- 1 macbot macbot 765 Feb 2 13:53 cmake_uninstall.cmake  
-rw-rw-r-- 1 macbot macbot 13606 Feb 2 13:53 compile_commands.json  
drwxrwxr-x 8 macbot macbot 4096 Feb 2 13:53 core  
-rw-r--r-- 1 macbot macbot 3881 Feb 2 13:53 CPackConfig.cmake  
-rw-r--r-- 1 macbot macbot 4264 Feb 2 13:53 CPackSourceConfig.cmake  
-rw-rw-r-- 1 macbot macbot 355 Feb 2 13:53CTestfile.cmake  
-rwxrwxr-x 1 macbot macbot 665984 Feb 2 13:59 et_test  
-rw-rw-r-- 1 macbot macbot 452 Feb 2 13:53 FindYDLIDAR_SDK.cmake  
-rwxrwxr-x 1 macbot macbot 687120 Feb 2 13:59 gs_test  
-rw-r--r-- 1 root root 6407 Feb 2 13:59 install_manifest.txt  
-rw-rw-r-- 1 macbot macbot 1823612 Feb 2 13:59 libydlidar_sdk.a  
-rwxrwxr-x 1 macbot macbot 684688 Feb 2 13:59 lidar_c_api_test  
-rw-rw-r-- 1 macbot macbot 27506 Feb 2 13:53 Makefile  
drwxrwxr-x 4 macbot macbot 4096 Feb 2 13:59 python  
drwxrwxr-x 3 macbot macbot 4096 Feb 2 13:53 samples  
drwxrwxr-x 4 macbot macbot 4096 Feb 2 13:53 src  
-rwxrwxr-x 1 macbot macbot 692304 Feb 2 13:59 tmini_test  
-rwxrwxr-x 1 macbot macbot 772480 Feb 2 13:59 tof_test  
-rwxrwxr-x 1 macbot macbot 678240 Feb 2 13:59 tri_and_gs_test  
-rwxrwxr-x 1 macbot macbot 688104 Feb 2 13:59 tri_test  
-rw-rw-r-- 1 macbot macbot 1249 Feb 2 13:53 ydlidar_config.h  
-rw-rw-r-- 1 macbot macbot 1065 Feb 2 13:53 ydlidar_sdkConfig.cmake  
-rw-rw-r-- 1 macbot macbot 596 Feb 2 13:53 ydlidar_sdkConfigVersion.cmake  
-rw-rw-r-- 1 macbot macbot 248 Feb 2 13:53 YDLIDAR_SDK.pc  
-rw-rw-r-- 1 macbot macbot 1657 Feb 2 13:53 ydlidar_sdkTargets.cmake  
macbot@moeinnano:~/Documents/YDLidar-SDK/build$
```

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.



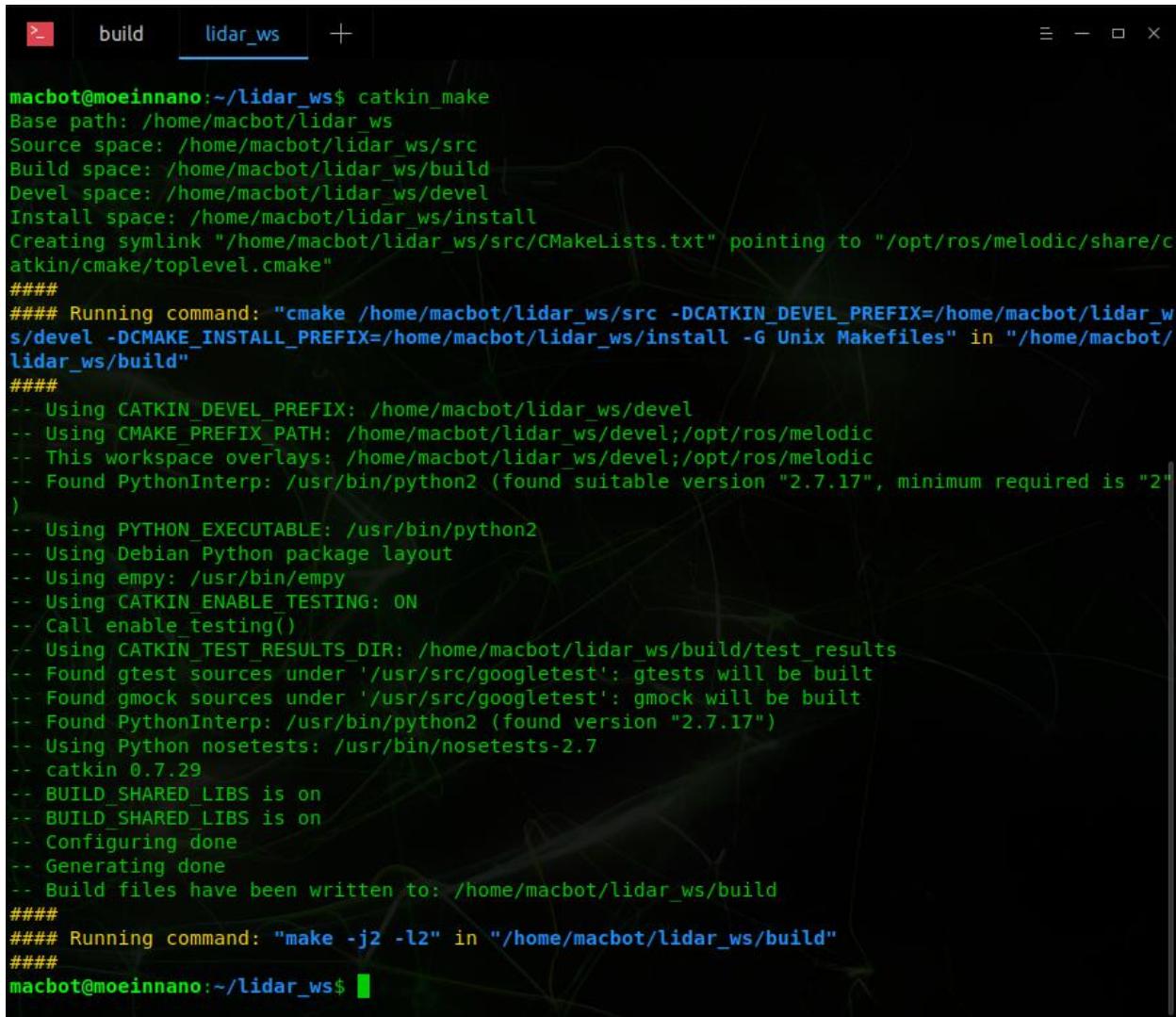
```
macbot@moeinnano:~/Documents/YDLidar-SDK/build$ sudo make install
[ 50%] Built target ydlidar_sdk
[ 55%] Built target lidar_c_api_test
[ 61%] Built target gs_test
[ 67%] Built target tmini_test
[ 73%] Built target tof_test
[ 79%] Built target et_test
[ 85%] Built target tri_and_gs_test
[ 91%] Built target tri_test
[100%] Built target _ydlidar
Install the project...
-- Install configuration: ""
-- Installing: /usr/local/include/core/base/datatype.h
-- Installing: /usr/local/include/core/base/locker.h
-- Installing: /usr/local/include/core/base/thread.h
-- Installing: /usr/local/include/core/base/timer.h
-- Installing: /usr/local/include/core/base/typedef.h
-- Installing: /usr/local/include/core/base/utils.h
-- Installing: /usr/local/include/core/base/v8stdint.h
-- Installing: /usr/local/include/core/base/ydlidar.h
-- Installing: /usr/local/include/core/common/ChannelDevice.h
-- Installing: /usr/local/include/core/common/DriverInterface.h
-- Installing: /usr/local/include/core/common/ydlidar_datatype.h
-- Installing: /usr/local/include/core/common/ydlidar_def.h
-- Installing: /usr/local/include/core/common/ydlidar_help.h
-- Installing: /usr/local/include/core/common/ydlidar_protocol.h
-- Installing: /usr/local/include/core/math/angles.h
-- Installing: /usr/local/include/core/network/ActiveSocket.h
-- Installing: /usr/local/include/core/network/PassiveSocket.h
-- Installing: /usr/local/include/core/network/SimpleSocket.h
```

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**.

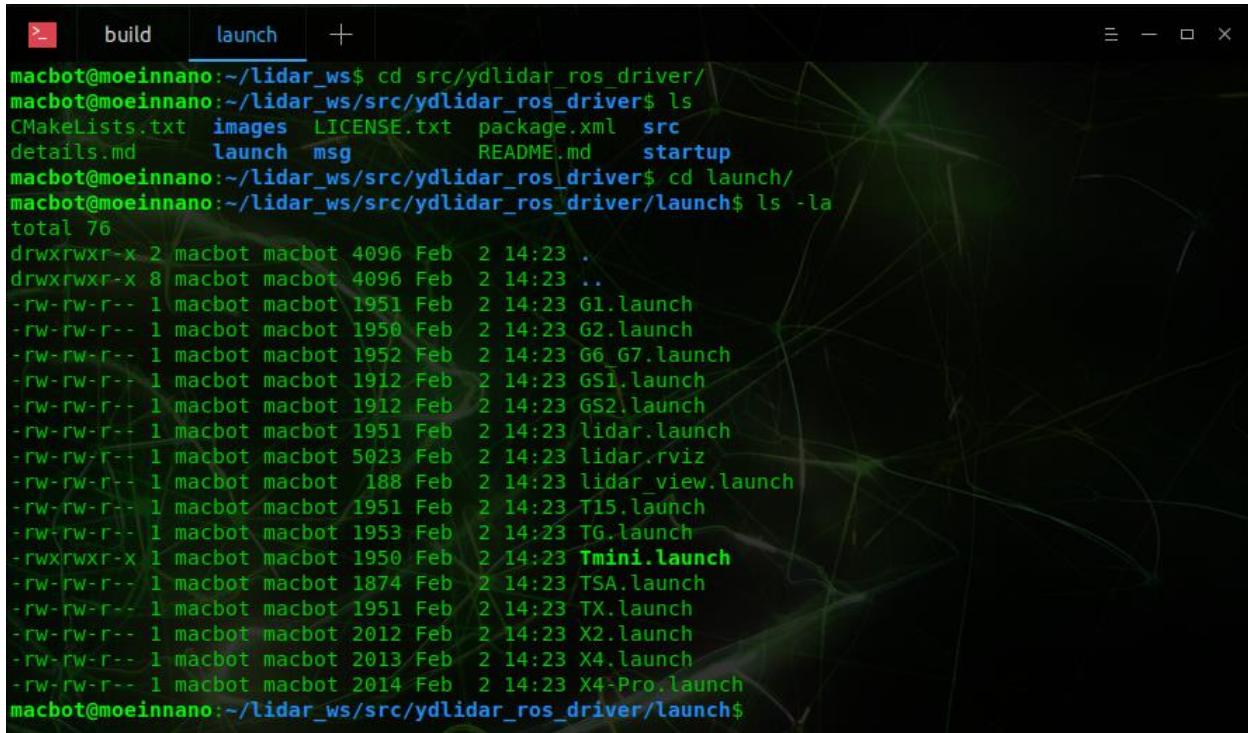


```
macbot@moeinnano:~/lidar_ws$ catkin_make
Base path: /home/macbot/lidar_ws
Source space: /home/macbot/lidar_ws/src
Build space: /home/macbot/lidar_ws/build
Devel space: /home/macbot/lidar_ws/devel
Install space: /home/macbot/lidar_ws/install
Creating symlink "/home/macbot/lidar_ws/src/CMakeLists.txt" pointing to "/opt/ros/melodic/share/catkin/cmake/toplevel.cmake"
#####
#### Running command: "cmake /home/macbot/lidar_ws/src -DCATKIN_DEVEL_PREFIX=/home/macbot/lidar_ws/devel -DCMAKE_INSTALL_PREFIX=/home/macbot/lidar_ws/install -G Unix Makefiles" in "/home/macbot/lidar_ws/build"
#####
-- Using CATKIN_DEVEL_PREFIX: /home/macbot/lidar_ws/devel
-- Using CMAKE_PREFIX_PATH: /home/macbot/lidar_ws/devel;/opt/ros/melodic
-- This workspace overlays: /home/macbot/lidar_ws/devel;/opt/ros/melodic
-- Found PythonInterp: /usr/bin/python2 (found suitable version "2.7.17", minimum required is "2")
)
-- Using PYTHON_EXECUTABLE: /usr/bin/python2
-- Using Debian Python package layout
-- Using empy: /usr/bin/empy
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/macbot/lidar_ws/build/test_results
-- Found gtest sources under '/usr/src/googletest': gtests will be built
-- Found gmock sources under '/usr/src/googletest': gmock will be built
-- Found PythonInterp: /usr/bin/python2 (found version "2.7.17")
-- Using Python nosetests: /usr/bin/nosetests-2.7
-- catkin 0.7.29
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
-- Configuring done
-- Generating done
-- Build files have been written to: /home/macbot/lidar_ws/build
#####
#### Running command: "make -j2 -l2" in "/home/macbot/lidar_ws/build"
#####
macbot@moeinnano:~/lidar_ws$
```

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.



```
build      launch +  
macbot@moeinnano:~/lidar_ws$ cd src/ydlidar_ros_driver/  
macbot@moeinnano:~/lidar_ws/src/ydlidar_ros_driver$ ls  
CMakeLists.txt  images  LICENSE.txt  package.xml  src  
details.md     launch  msg        README.md    startup  
macbot@moeinnano:~/lidar_ws/src/ydlidar_ros_driver$ cd launch/  
macbot@moeinnano:~/lidar_ws/src/ydlidar_ros_driver/launch$ ls -la  
total 76  
drwxrwxr-x  2 macbot macbot 4096 Feb  2 14:23 .  
drwxrwxr-x  8 macbot macbot 4096 Feb  2 14:23 ..  
-rw-rw-r--  1 macbot macbot 1951 Feb  2 14:23 G1.launch  
-rw-rw-r--  1 macbot macbot 1950 Feb  2 14:23 G2.launch  
-rw-rw-r--  1 macbot macbot 1952 Feb  2 14:23 G6_G7.launch  
-rw-rw-r--  1 macbot macbot 1912 Feb  2 14:23 GS1.launch  
-rw-rw-r--  1 macbot macbot 1912 Feb  2 14:23 GS2.launch  
-rw-rw-r--  1 macbot macbot 1951 Feb  2 14:23 lidar.launch  
-rw-rw-r--  1 macbot macbot 5023 Feb  2 14:23 lidar.rviz  
-rw-rw-r--  1 macbot macbot 188 Feb  2 14:23 lidar_view.launch  
-rw-rw-r--  1 macbot macbot 1951 Feb  2 14:23 T15.launch  
-rw-rw-r--  1 macbot macbot 1953 Feb  2 14:23 TG.launch  
-rwxrwxr-x  1 macbot macbot 1950 Feb  2 14:23 Tmini.launch  
-rw-rw-r--  1 macbot macbot 1874 Feb  2 14:23 TSA.launch  
-rw-rw-r--  1 macbot macbot 1951 Feb  2 14:23 TX.launch  
-rw-rw-r--  1 macbot macbot 2012 Feb  2 14:23 X2.launch  
-rw-rw-r--  1 macbot macbot 2013 Feb  2 14:23 X4.launch  
-rw-rw-r--  1 macbot macbot 2014 Feb  2 14:23 X4-Pro.launch  
macbot@moeinnano:~/lidar_ws/src/ydlidar_ros_driver/launch$
```

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.

```
macbot@moeinnano:~/lidar_ws/src/ydlidar_ros_driver/launch$ gedit X2.launch
x  build  launch  +
macbot@moeinnano:~/lidar_ws/src/ydlidar_ros_driver/launch$ gedit X2.launch
X2.launch (~/lidar_ws/src/ydlidar_ros_driver/launch) - gedit
Open Save
X2.launch
~/lidar_ws/src/ydlidar_ros_driver/launch
<launch>
  <node name="ydlidar_lidar_publisher"  pkg="ydlidar_ros_driver"  type="ydlidar_ros_driver_node"
output="screen" respawn="false" >
    <!-- string property -->
    <param name="port"      type="string" value="/dev/ydlidar"/>
    <param name="frame_id"   type="string" value="laser_frame"/>
    <param name="ignore_array" type="string" value="" />

    <!-- int property -->
    <param name="baudrate"     type="int" value="115200"/>
    <!-- 0:TYPE_TOF, 1:TYPE_TRIANGLE, 2:TYPE_TOF_NET -->
    <param name="lidar_type"   type="int" value="1"/>
    <!-- 0:YDLIDAR_TYPE_SERIAL, 1:YDLIDAR_TYPE_TCP -->
    <param name="device_type"  type="int" value="0"/>
    <param name="sample_rate"   type="int" value="3"/>
    <param name="abnormal_check_count" type="int" value="4"/>

    <!-- bool property -->
    <param name="resolution_fixed"  type="bool"  value="true"/>
    <param name="auto_reconnect"    type="bool"  value="true"/>
    <param name="reversion"        type="bool"  value="false"/>
    <param name="inverted"         type="bool"  value="true"/>
    <param name="isSingleChannel"  type="bool"  value="true"/>
    <param name="intensity"        type="bool"  value="false"/>
    <param name="support_motor_dtr" type="bool"  value="true"/>
    <param name="invalid_range_is_inf" type="bool"  value="false"/>
    <param name="point_cloud_preservative" type="bool"  value="false"/>

    <!-- float property -->
    <param name="angle_min"       type="double" value="-180" />
    <param name="angle_max"       type="double" value="180" />
    <param name="range_min"       type="double" value="0.1" />
    <param name="range_max"       type="double" value="12.0" />
    <!-- frequency is invalid, External PWM control speed -->
    <param name="frequency"      type="double" value="10.0"/>
  </node>
  <node pkg="tf" type="static_transform_publisher" name="base_link_to_laser4"
    args="0.0 0.0 0.2 0.0 0.0 0.0 /base_footprint /laser_frame 40" />
</launch>
```

Plain Text ▾ Tab Width: 8 ▾ Ln 22, Col 66 ▾ INS

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.

```
macbot@moeinnano:~$ roscore
... logging to /home/macbot/.ros/log/0a0b3d06-a33d-11ed-a4f8-0242433508ad/roslaunch-moeinnano-111
56.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://moeinnano:44691/
ros_comm version 1.14.13

SUMMARY
=====

PARAMETERS
* /rosdistro: melodic
* /rosversion: 1.14.13

NODES

auto-starting new master
process[master]: started with pid [11167]
ROS_MASTER_URI=http://moeinnano:11311/

setting /run_id to 0a0b3d06-a33d-11ed-a4f8-0242433508ad
process[rosout-1]: started with pid [11180]
started core service [/rosout]
```

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:

```
/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-11156.log  
Checking log directory for disk usage. This may take a while.  
Press Ctrl-C to interrupt  
Done checking log file disk usage. Usage is <1GB.  
  
started roslaunch server http://moeinnano:44691/  
ros_comm version 1.14.13  
  
SUMMARY  
=====
```

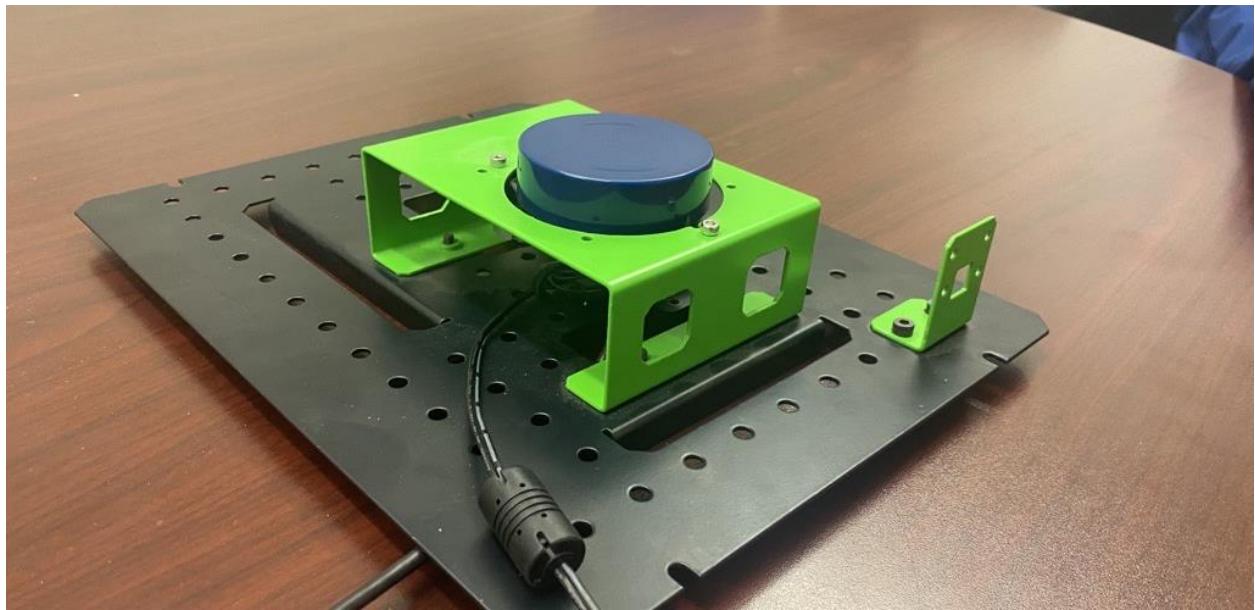
```
PARAMETERS
```

```
macbot@moeinnano:~$ roslaunch ydlidar_ros_driver X2.launch  
... logging to /home/macbot/.ros/log/0a0b3d06-a33d-11ed-a4f8-0242433508ad/roslaunch-moeinnano-11415.log  
Checking log directory for disk usage. This may take a while.  
Press Ctrl-C to interrupt  
Done checking log file disk usage. Usage is <1GB.  
  
started roslaunch server http://moeinnano:46731/  
  
SUMMARY  
=====
```

```
PARAMETERS
```

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 0x8FFFFF180  
[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.

```
/home/macbot/lidar_ws/src/ydilidar_ros_driver/launch/X2.launch | macbot@moeinnano + - x

Nodes:
/rosvout :
  Inbound:
    /ydilidar_lidar_publisher
    /base_link_to_laser4
  Outbound:
/base_link_to_laser4 :
  Inbound:
  Outbound:
    /rosvout
/ydilidar_lidar_publisher :
  Inbound:
  Outbound:
    /rosvout
Services:
/start_scan
/base_link_to_laser4/set_logger_level
/ydilidar_lidar_publisher/get_loggers
/rosvout/get_loggers
/rosvout/set_logger_level
/stop_scan
/base_link_to_laser4/get_loggers
/ydilidar_lidar_publisher/set_logger_level

Nodes:
/rosvout :
  Inbound:
    /ydilidar_lidar_publisher
    /base_link_to_laser4
  Outbound:
/base_link_to_laser4 :
  Inbound:
  Outbound:
    /rosvout
/ydilidar_lidar_publisher :
  Inbound:
  Outbound:
    /rosvout
Services:
/start_scan
/base_link_to_laser4/set_logger_level
/ydilidar_lidar_publisher/get_loggers
/rosvout/get_loggers
```

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

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

```
/home/macbot/lidar_ws/src/yc | macbot@moeinnano + - x

macbot@moeinnano:~$ rostopic list
/point_cloud
/rosout
/rosout_agg
/scan
/tf
macbot@moeinnano:~$
```

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

```
/home/macbot/lidar_ws/src/ydlidar  
macbot@moeinnano:~$ rostopic info /scan  
Type: sensor_msgs/LaserScan  
  
Publishers:  
* /ydlidar_lidar_publisher (http://moeinnano:40563/)  
  
Subscribers: None  
  
macbot@moeinnano:~$
```

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.

```
/home/macbot/lidar_ws/src/ydlidar  
macbot@moeinnano:~$ rostopic echo /scan  
header:  
  seq: 6458  
  stamp:  
    secs: 1675372995  
    nsecs: 503693000  
    frame_id: "laser_frame"  
  angle_min: -3.14159274101  
  angle_max: 3.14159274101  
  angle_increment: 0.0133969839662  
  time_increment: 0.000248031050432  
  scan_time: 0.119799003005  
  range_min: 0.10000000149  
  range_max: 12.0  
  ranges: [0.0, 1.1892499923706055, 1.190250039100647, 1.1902  
50039100647, 1.191249966621399, 1.1922500133514404, 1.19225  
00133514404, 1.1922500133514404, 1.1952500343322754, 1.1972  
500085830688, 1.1982500553131104, 1.2002500295639038, 1.203  
2500505447388, 1.2062499523162842, 1.2082500457763672, 1.21  
02500200271606, 1.2132500410079956, 1.216249942779541, 1.22  
02500104904175, 1.225250005722046, 1.2272499799728394, 1.23
```

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