

# MARC

Visualizing LiDAR Data

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:



## 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.....	3
Post-Lab Questions .....	3
Optional Assignment.....	4
Feedback .....	5
Overview .....	6
Setup .....	7
Disconnecting the LiDAR.....	7
Powering on the MacBot .....	7
Connecting to the MacBot.....	8
Setting Up the LiDAR Drivers .....	10
Downloading the YDLiDAR SDK.....	10
Building the YDLiDAR SDK.....	10
Installing YDLiDAR_SDK onto the MacBot .....	14
Verifying that ROS Melodic is Installed.....	14
Creating and Sourcing a ROS Workspace.....	14
Building the YDLiDAR ROS Driver .....	17
Visualizing LiDAR Point Cloud Data.....	21
Exercise A.....	30
Generating a Diagram .....	30
Exercise B .....	31

## 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?

(Suggested: 1 sentence)

Q8 - Which command was used to *echo* that topic to the terminal?

(Suggested: 1 sentence)

Q9 - Which command can be used to get *information* on a particular topic?

(Suggested: 1 sentence)

Q10 - What ROS tool can be used to *visualize* a stream of data?

(Suggested: 1 sentence)

Q11 – Write a brief LinkedIn post on 4 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**

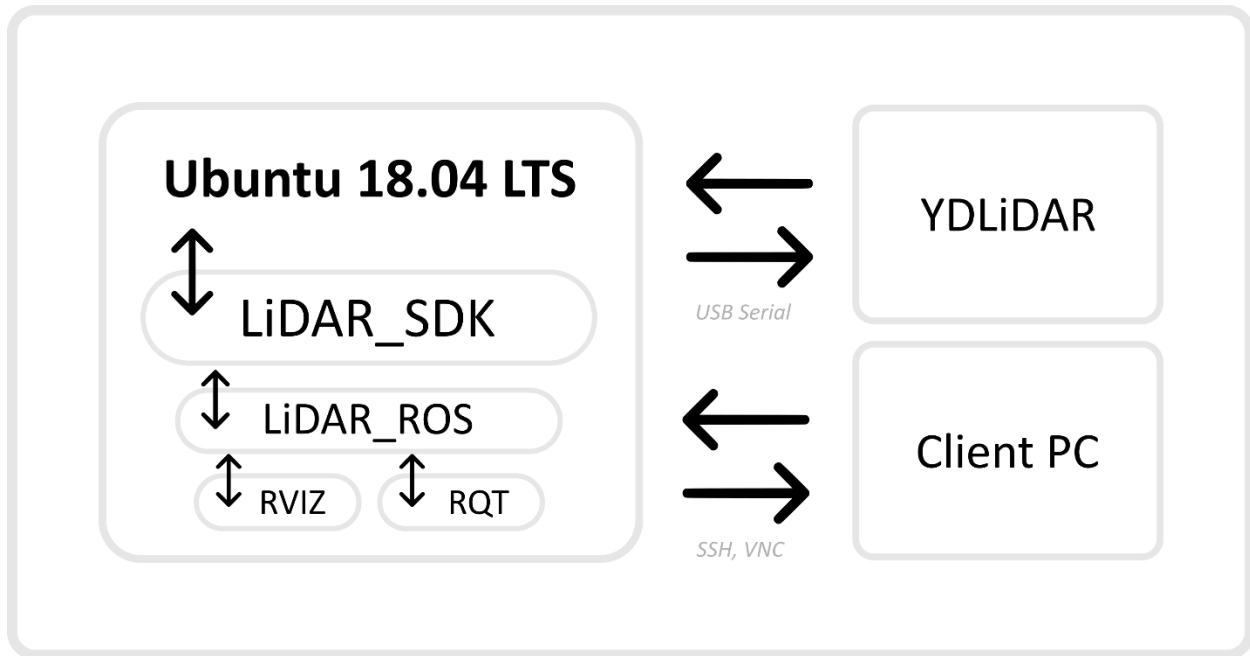
List any unclear steps:

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**

## Overview



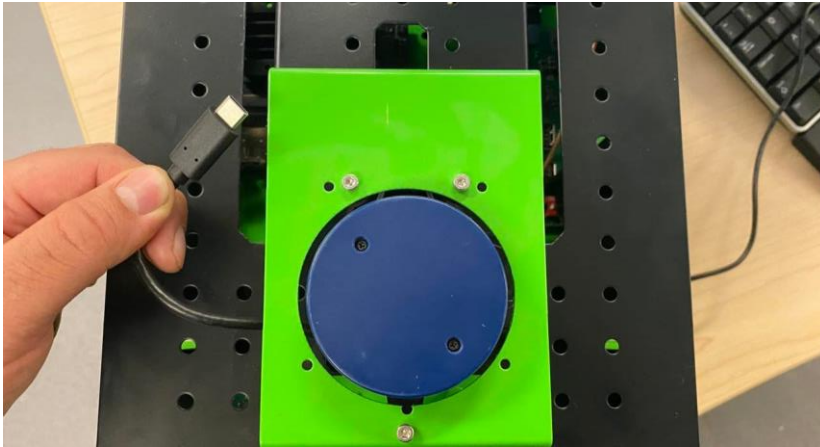
The goal of this lab is to visualize LiDAR data. The Jetson Nano microcontroller in the MacBot runs a distribution of Linux called Ubuntu version 18.04. In this lab, we will be installing the C++ & Python libraries for Linux to talk to the LiDAR. After, we use ROS, which is a middleware for creating robotic applications using Linux, to visualize the LiDAR data and create a chart outlining how data is flowing live in the MacBot. To make the raw libraries compatible with ROS, we build the LiDAR ROS driver package. Its job is to translate the C++ and Python interfaces to a standard node interface that ROS can understand and communicate with.

This lab is a great introduction to ROS, and sensor visualization using ROS. I hope you enjoy it!

## Setup

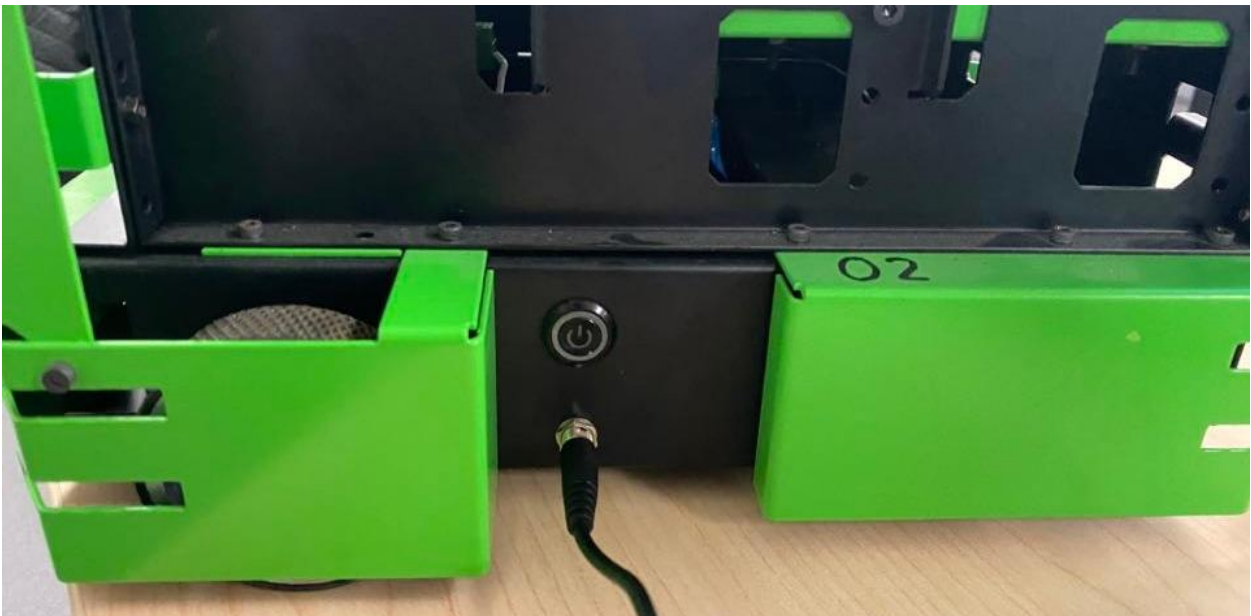
### Disconnecting the LiDAR

When the LiDAR is connected, it draws a considerable amount of current. This makes it difficult for the MacBot to complete its BOOT process. Before powering on the MacBot, ensure that you disconnect the LiDAR temporarily.



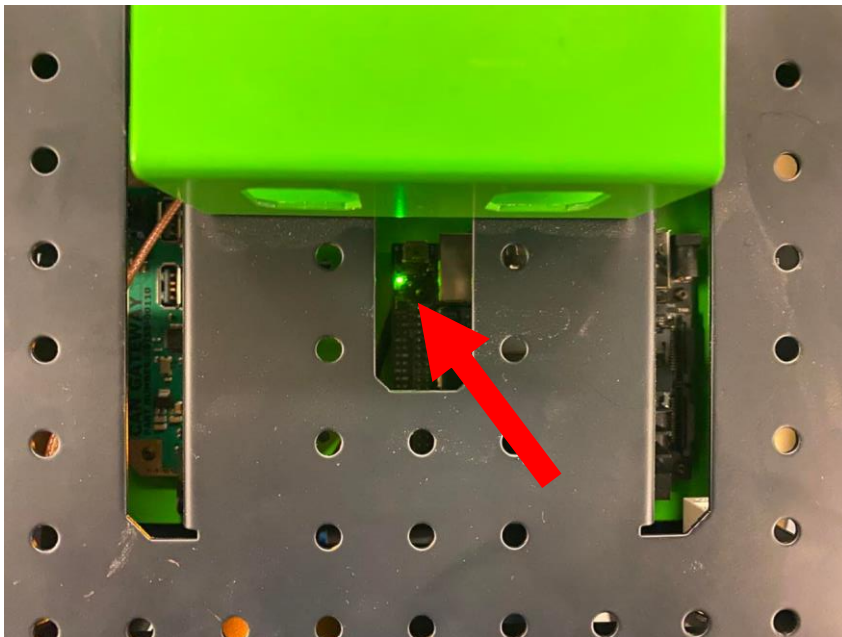
### Powering on the MacBot

Notice the power switch and charging port on the right-side of the MacBot chassis. Press the POWER button until it latches in the ON position and begins to glow.





Wait 10 seconds and ensure that the Jetson Nano POWER status LED remains lit. If not, your battery requires charging.



### Connecting to the MacBot

Follow the instructions on the **connect/** page of the macbot documentation to connect to your macbot unit:

<https://adam-36.gitbook.io/macbot/connect>

Ensure that you see a remote desktop connection in MobaXTerm.





## Setting Up the LiDAR Drivers

### Downloading the YDLiDAR SDK

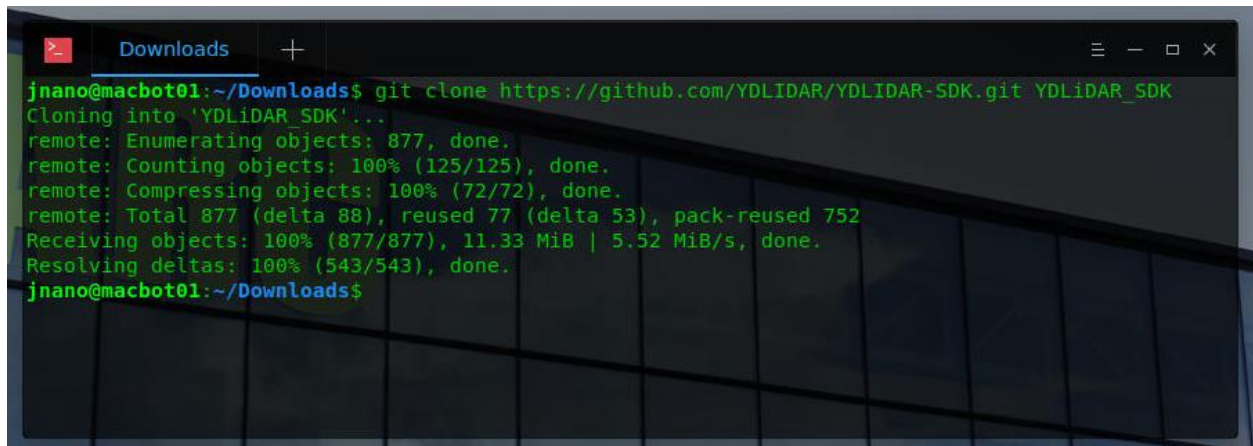
Navigate to your ~/Downloads/ folder.

```
cd ~/Downloads
```

Clone the following GitHub repo into your Downloads/ folder:

<https://github.com/YDLIDAR/sdk.git>

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

A terminal window titled 'Downloads' with standard window controls. The terminal shows the execution of the 'git clone' command to fetch the YDLiDAR SDK. The output displays progress for enumerating, counting, and compressing objects, followed by the total size and transfer rate. The process concludes with resolving deltas.

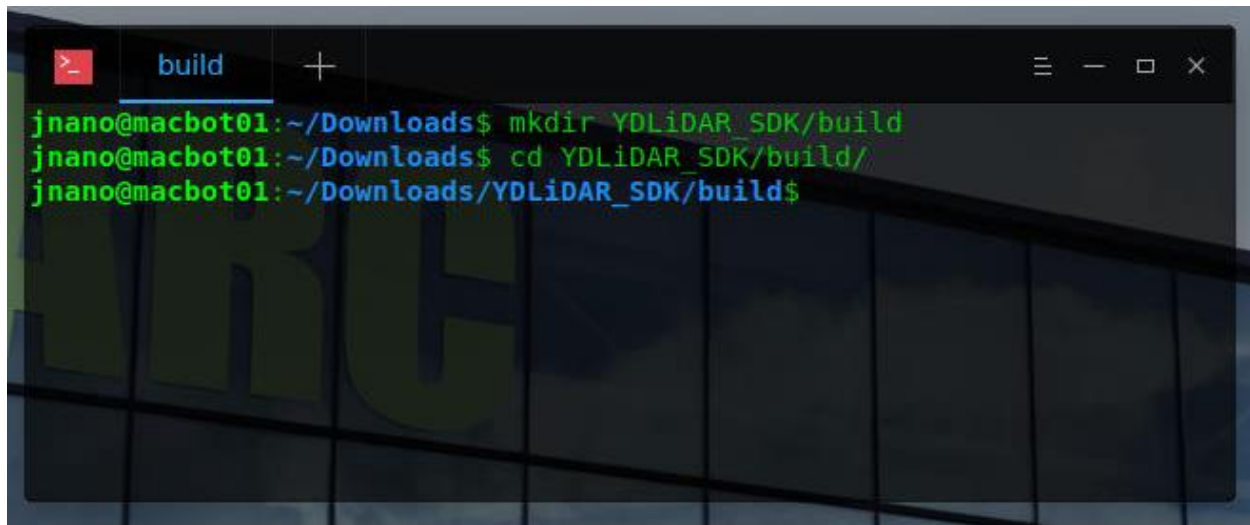
```
jnano@macbot01:~/Downloads$ git clone https://github.com/YDLIDAR/YDLidar-SDK YDLiDAR_SDK
Cloning into 'YDLiDAR_SDK'...
remote: Enumerating objects: 877, done.
remote: Counting objects: 100% (125/125), done.
remote: Compressing objects: 100% (72/72), done.
remote: Total 877 (delta 88), reused 77 (delta 53), pack-reused 752
Receiving objects: 100% (877/877), 11.33 MiB | 5.52 MiB/s, done.
Resolving deltas: 100% (543/543), done.
jnano@macbot01:~/Downloads$
```

### Building the YDLiDAR SDK

Create a folder inside of YDLiDAR\_SDK/ called build/.

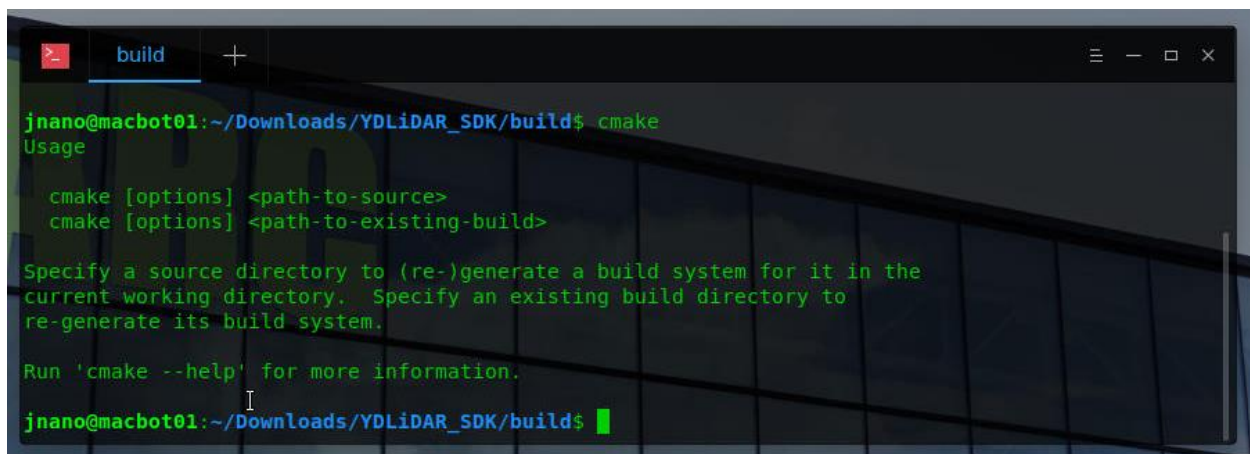
```
mkdir YDLiDAR_SDK/build
```

```
cd YDLiDAR_SDK/build/
```



```
jnano@macbot01:~/Downloads$ mkdir YDLiDAR_SDK/build
jnano@macbot01:~/Downloads$ cd YDLiDAR_SDK/build/
jnano@macbot01:~/Downloads/YDLiDAR_SDK/build$
```

Run cmake and see what paths it requires.



```
jnano@macbot01:~/Downloads/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.
jnano@macbot01:~/Downloads/YDLiDAR_SDK/build$
```

Run cmake on the project.

*cmake ..*

```
build +
jnano@macbot01:~/Downloads/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
```

```
build +
-- | Resulting configuration for |
-- +-----+
-- PLATFORM
-- 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/jnano/Downloads/YDLIDAR_SDK/build
jnano@macbot01:~/Downloads/YDLIDAR_SDK/build$
```

Notice that build files have been created.



```
build +
jnano@macbot01:~/Downloads/YDLiDAR_SDK/build$ ls
CMakeCache.txt      core                Makefile            ydlidar_sdkConfig.cmake
CMakeFiles          CPackConfig.cmake  python              ydlidar_sdkConfigVersion.cmake
cmake_install.cmake CPackSourceConfig.cmake samples             YDLIDAR_SDK.pc
cmake_uninstall.cmake CTestTestfile.cmake src                 ydlidar_sdkTargets.cmake
compile_commands.json FindYDLIDAR_SDK.cmake ydlidar_config.h
jnano@macbot01:~/Downloads/YDLiDAR_SDK/build$
```

Run make on the project.

*make*

```
build +
jnano@macbot01:~/Downloads/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
[10%] Building CXX object CMakeFiles/ydlidar_sdk.dir/core/network/PassiveSocket.cpp.o
[13%] Building CXX object CMakeFiles/ydlidar_sdk.dir/core/network/SimpleSocket.cpp.o
[16%] Building CXX object CMakeFiles/ydlidar_sdk.dir/core/serial/serial.cpp.o
[100%] Linking CXX shared module _ydlidar.so
[100%] Built target _ydlidar
jnano@macbot01:~/Downloads/YDLiDAR_SDK/build$
```

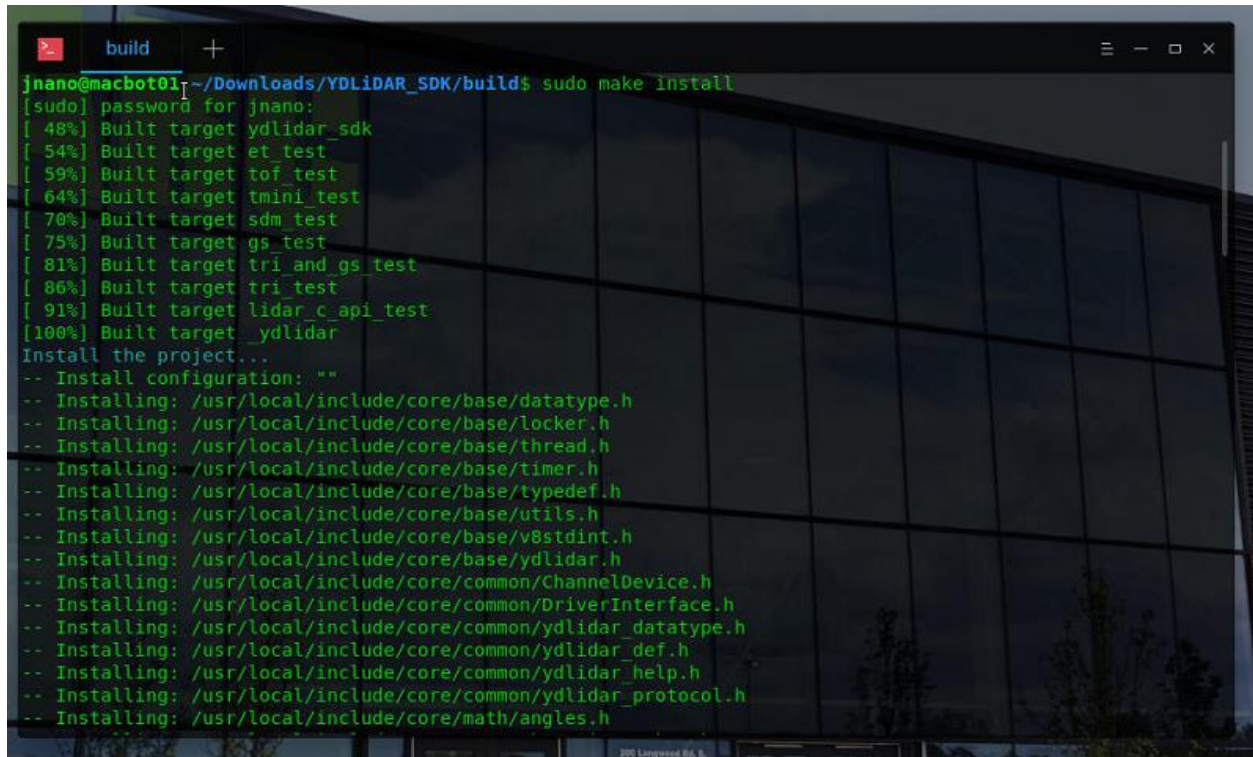
Notice that the project has been built.

```
build +
jnano@macbot01:~/Downloads/YDLiDAR_SDK/build$ ls
CMakeCache.txt      CPackSourceConfig.cmake  Makefile            tri_and_gs_test
CMakeFiles          CTestTestfile.cmake     python              tri_test
cmake_install.cmake et_test                  samples             ydlidar_config.h
cmake_uninstall.cmake FindYDLIDAR_SDK.cmake  sdm_test            ydlidar_sdkConfig.cmake
compile_commands.json gs_test                  src                 ydlidar_sdkConfigVersion.cmake
core                libydlidar_sdk.a        tmini_test         YDLIDAR_SDK.pc
CPackConfig.cmake  lidar_c_api_test        tof_test            ydlidar_sdkTargets.cmake
jnano@macbot01:~/Downloads/YDLiDAR_SDK/build$
```

## Installing YDLiDAR\_SDK onto the MacBot

Run the following command to install the SDK onto your system.

```
sudo make install
```



```
jnano@macbot01:~/Downloads/YDLiDAR_SDK/build$ sudo make install
[sudo] password for jnano:
[ 48%] Built target ydlidar_sdk
[ 54%] Built target et_test
[ 59%] Built target tof_test
[ 64%] Built target tmini_test
[ 70%] Built target sdm_test
[ 75%] Built target gs_test
[ 81%] Built target tri_and_gs_test
[ 86%] Built target tri_test
[ 91%] Built target lidar_c_api_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/utls.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
```

## Verifying that ROS Melodic is Installed

Open the Terminal and run the following command to verify that ROS Melodic has been correctly installed:

```
rosversion -d
```



```
jnano@macbot01:~$ rosversion -d
melodic
jnano@macbot01:~$
```

## Creating and Sourcing a ROS Workspace

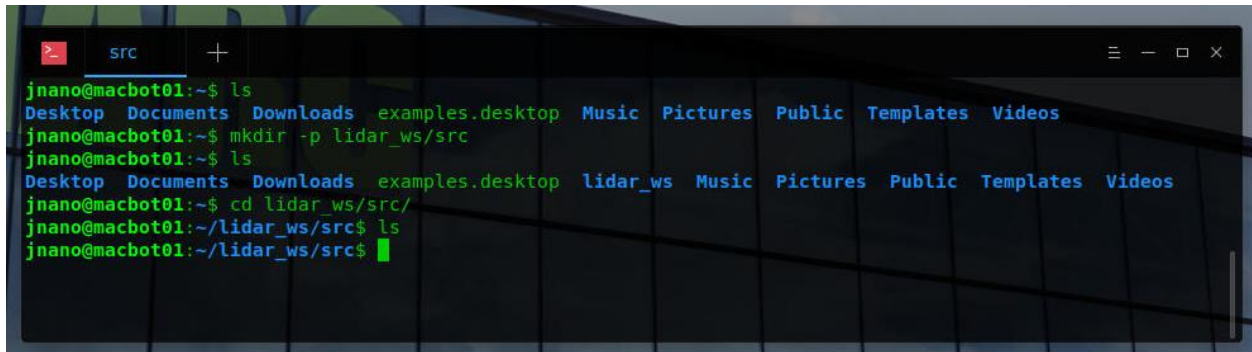
Navigate to the home/ directory using the following command:

```
cd ~
```

Create a new folder called **lidar\_ws/** with a subdirectory within it called **src/**.

```
mkdir -p lidar_ws/src
```

```
cd lidar_ws/src
```

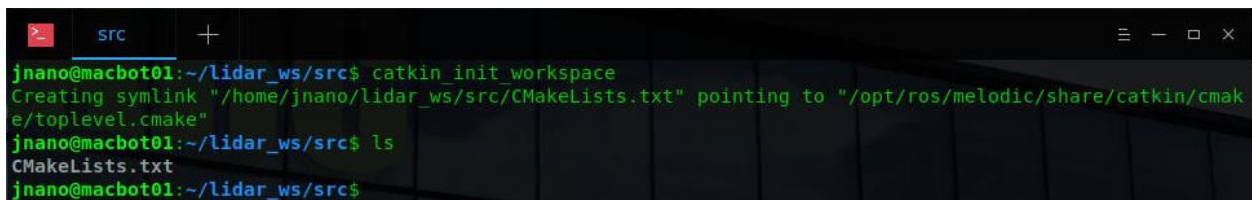


```
src +
jnano@macbot01:~$ ls
Desktop Documents Downloads examples.desktop Music Pictures Public Templates Videos
jnano@macbot01:~$ mkdir -p lidar_ws/src
jnano@macbot01:~$ ls
Desktop Documents Downloads examples.desktop lidar_ws Music Pictures Public Templates Videos
jnano@macbot01:~$ cd lidar_ws/src/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

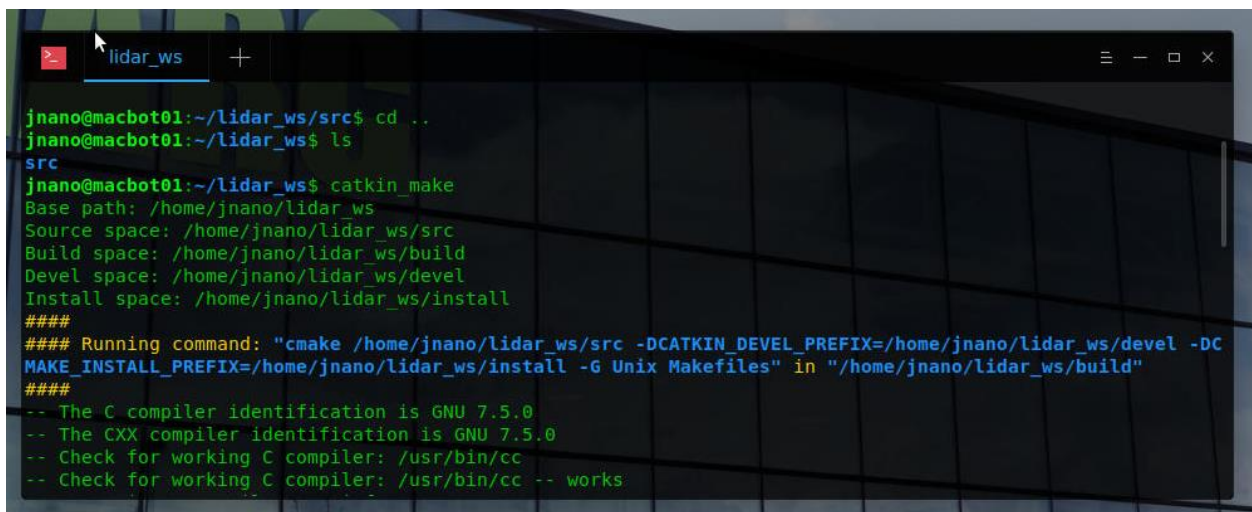


```
src +
jnano@macbot01:~/lidar_ws/src$ catkin_init_workspace
Creating symlink "/home/jnano/lidar_ws/src/CMakeLists.txt" pointing to "/opt/ros/melodic/share/catkin/cmake/toplevel.cmake"
jnano@macbot01:~/lidar_ws/src$ ls
CMakeLists.txt
jnano@macbot01:~/lidar_ws/src$
```

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

```
cd ~/lidar_ws
```

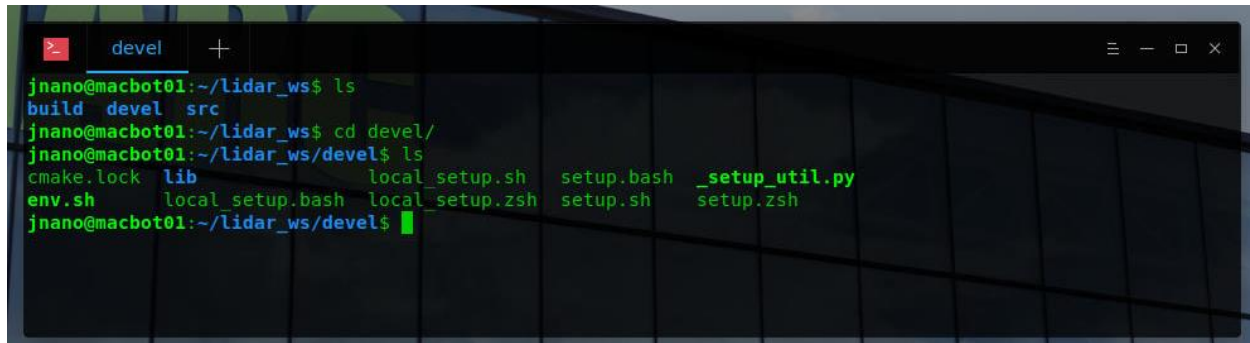
```
catkin_make
```



```
lidar_ws +
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.

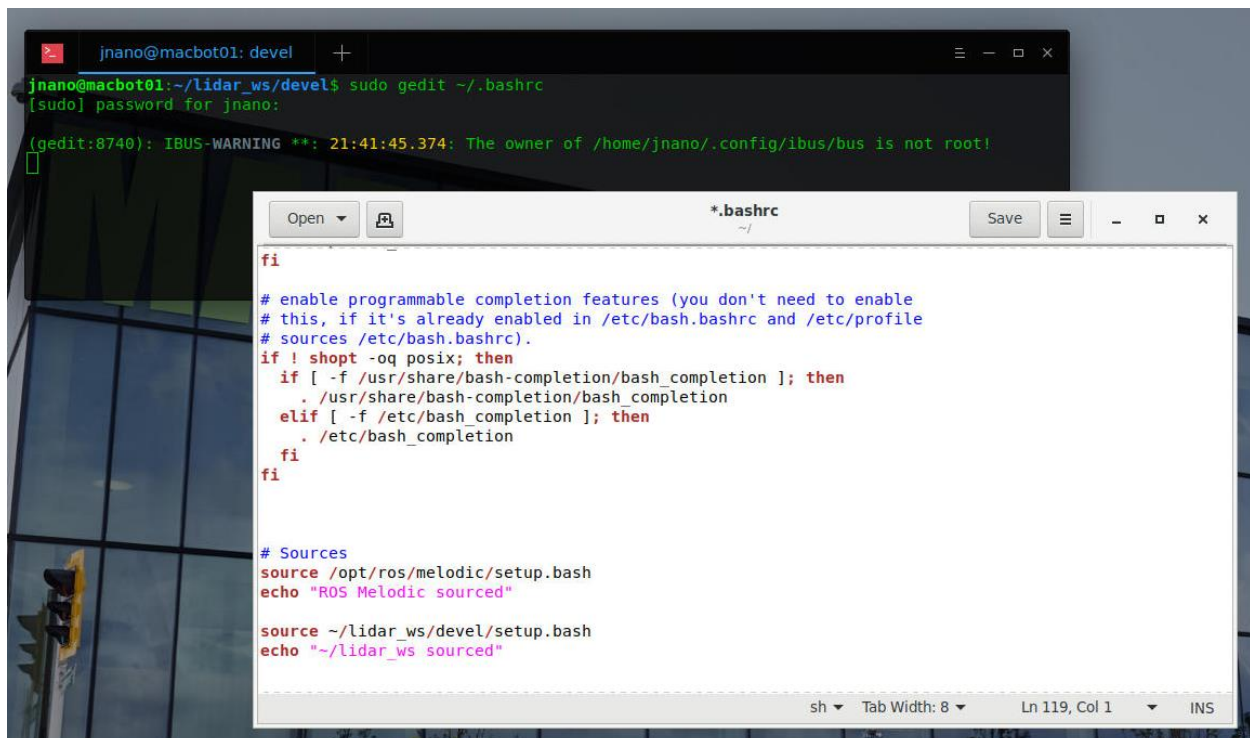


```
jnano@macbot01:~/lidar_ws$ ls
build  devel  src
jnano@macbot01:~/lidar_ws$ cd devel/
jnano@macbot01:~/lidar_ws/devel$ ls
cmake.lock  lib          local_setup.sh  setup.bash  _setup_util.py
env.sh      local_setup.bash  local_setup.zsh  setup.sh    setup.zsh
jnano@macbot01:~/lidar_ws/devel$
```

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>



```
jnano@macbot01:~/lidar_ws/devel$ sudo gedit ~/.bashrc
[sudo] password for jnano:

(gedit:8740): IBUS-WARNING **: 21:41:45.374: The owner of /home/jnano/.config/ibus/bus is not root!
```

```
fi

# enable programmable completion features (you don't need to enable
# 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"
```

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.

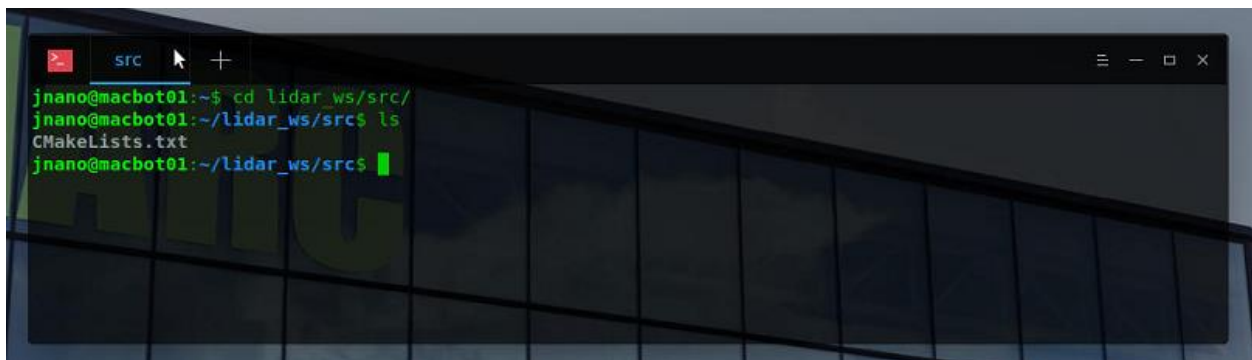
A terminal window with a dark background and green text. The window title bar shows two tabs, both labeled 'devel'. The terminal output shows two lines of green text: 'ROS Melodic sourced' and '~/lidar\_ws sourced'. Below these, the prompt 'jnano@macbot01:~/lidar\_ws/devel\$' is visible. A red arrow points to the second line of output. The window has standard Linux window controls (minimize, maximize, close) in the top right corner.

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

## Building the YDLiDAR ROS Driver

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

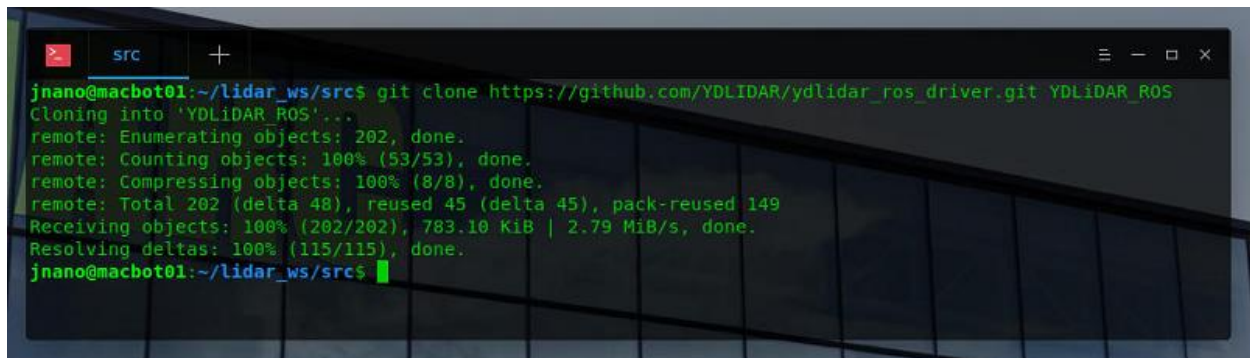
```
cd ~/ydlidar_ws/src
```

A terminal window with a dark background and green text. The window title bar shows a tab labeled 'src'. The terminal output shows the following commands and their outputs: 'cd lidar\_ws/src/' followed by a new prompt, 'ls' followed by 'CMakeLists.txt', and another prompt. The prompt is 'jnano@macbot01:~/lidar\_ws/src\$'. The window has standard Linux window controls in the top right corner.

Clone the following YDLiDAR ROS driver into your workspace:

[https://github.com/YDLIDAR/ydlidar\\_ros\\_driver](https://github.com/YDLIDAR/ydlidar_ros_driver)

git clone [https://github.com/YDLIDAR/ydlidar\\_ros\\_driver.git](https://github.com/YDLIDAR/ydlidar_ros_driver.git) YDLiDAR\_ROS

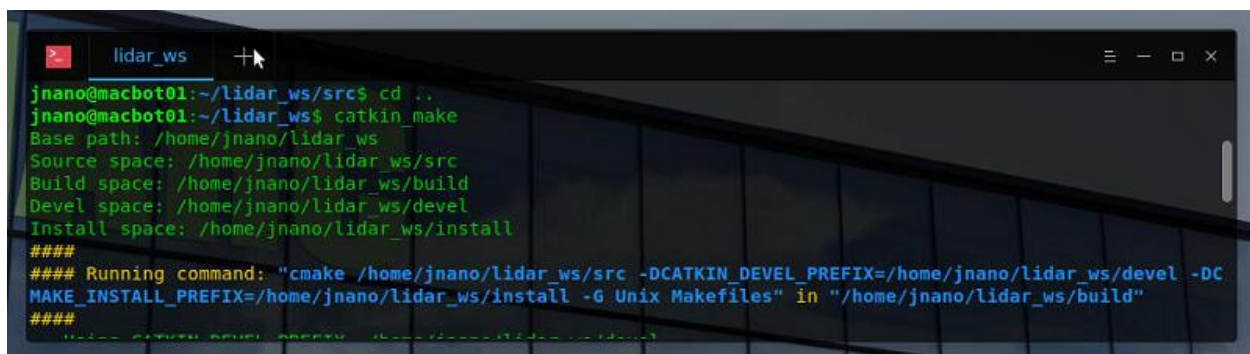


```
jnano@macbot01:~/lidar_ws/src$ git clone https://github.com/YDLIDAR/ydlidar_ros_driver.git YDLiDAR_ROS
Cloning into 'YDLiDAR_ROS'...
remote: Enumerating objects: 202, done.
remote: Counting objects: 100% (53/53), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 202 (delta 48), reused 45 (delta 45), pack-reused 149
Receiving objects: 100% (202/202), 783.10 KiB | 2.79 MiB/s, done.
Resolving deltas: 100% (115/115), done.
jnano@macbot01:~/lidar_ws/src$
```

Build the workspace using **catkin\_make**.

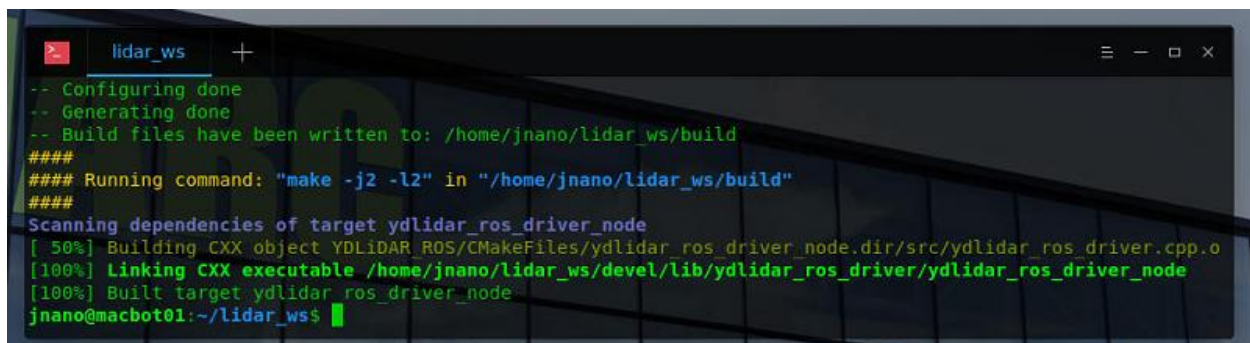
`cd ..`

`catkin_make`



```
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 is successful.



```
-- Configuring done
-- 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/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$
```

## Setting Permissions for the ROS LiDAR Driver

Navigate to your built LiDAR ROS package.

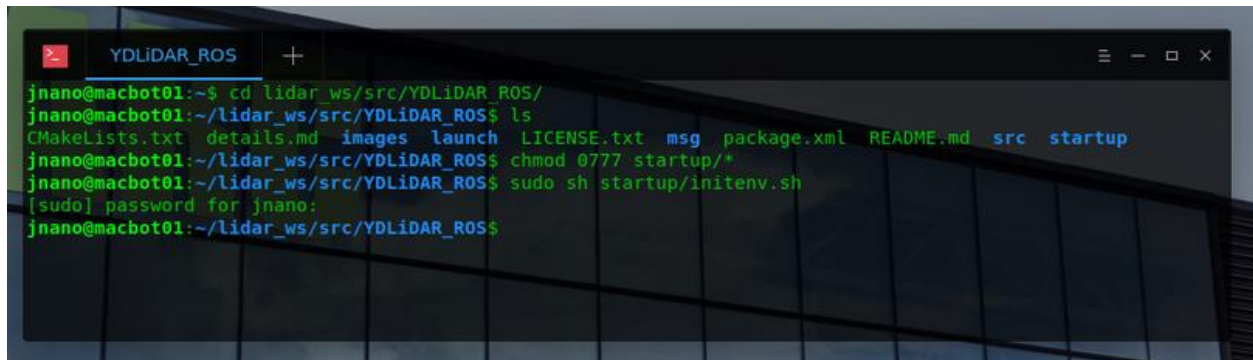
```
cd lidar_ws/src/YDLiDAR_ROS/
```

Give all files in the startup/ directory read, write, and executable permissions for all users.

```
chmod 0777 startup/*
```

Run the environment initialization script inside of the startup/ directory. This script modifies the USB kernel device module to be able to communicate with the LiDAR. It then restarts the Linux device daemon/service.

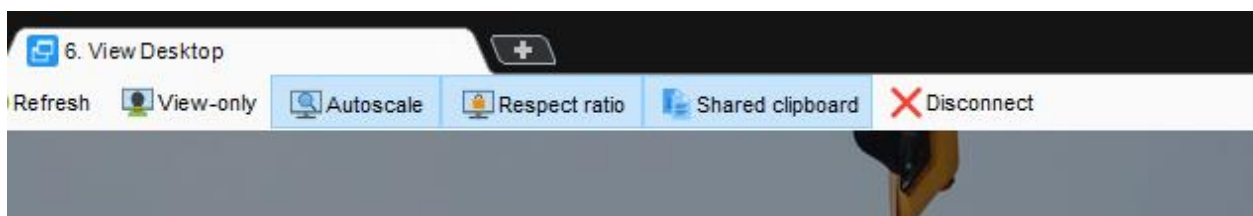
```
sudo sh startup/initenv.sh
```

A terminal window titled 'YDLIDAR\_ROS' with standard window controls. The terminal shows the following commands and output:

```
jnano@macbot01:~$ cd lidar_ws/src/YDLiDAR_ROS/
jnano@macbot01:~/lidar_ws/src/YDLiDAR_ROS$ ls
CMakeLists.txt  details.md  images  launch  LICENSE.txt  msg  package.xml  README.md  src  startup
jnano@macbot01:~/lidar_ws/src/YDLiDAR_ROS$ chmod 0777 startup/*
jnano@macbot01:~/lidar_ws/src/YDLiDAR_ROS$ sudo sh startup/initenv.sh
[sudo] password for jnano:
jnano@macbot01:~/lidar_ws/src/YDLiDAR_ROS$
```

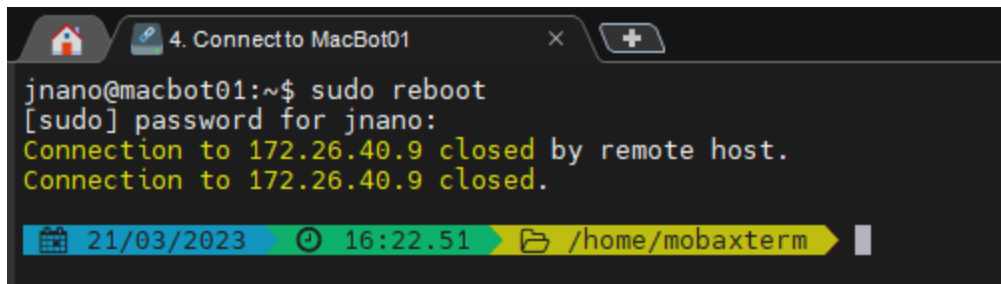
Now, we need to reboot the MacBot to ensure that the changes take effect.

Close the remote desktop connection by pressing **Disconnect**.



In the connection bash terminal, type the following command:

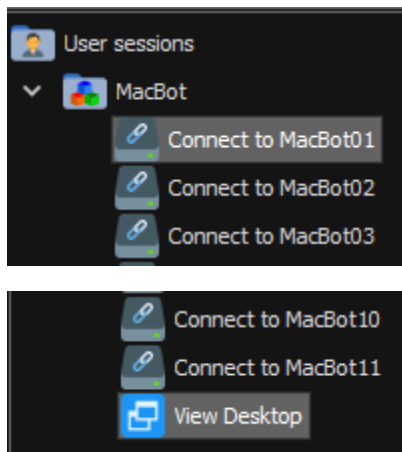
```
sudo reboot
```



```
jnano@macbot01:~$ sudo reboot
[sudo] password for jnano:
Connection to 172.26.40.9 closed by remote host.
Connection to 172.26.40.9 closed.
```

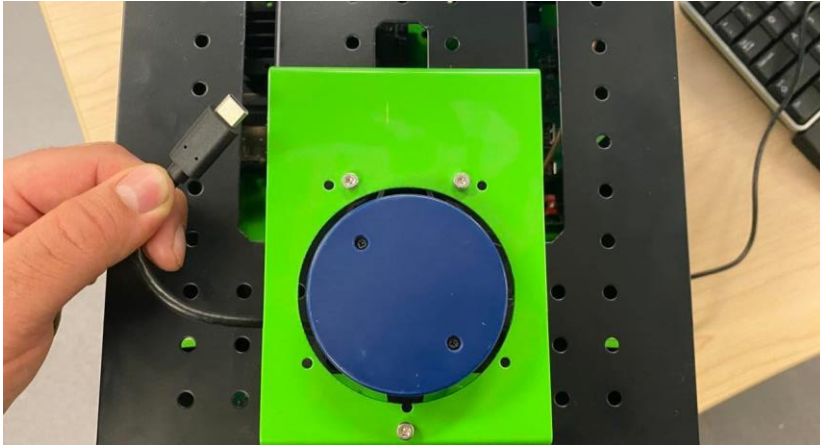
21/03/2023 16:22.51 /home/mobaxterm

Wait 2 minutes before reconnecting to the MacBot.



## Visualizing LiDAR Point Cloud Data

Connect the LiDAR to your MacBot using the USB-C cable.



Start ROSCore in a new terminal window.

```
roscore
```

A screenshot of a terminal window titled 'roscore macbot01:11311'. The terminal shows the command 'roscore' being executed in a shell. The output includes logging information, disk usage checks, and the successful start of the ROS master and roscout service. The background of the terminal window is a dark image of a modern building with large glass windows.

```
roscore macbot01:11311 +
jnano@macbot01:~/lidar_ws$ roscore
... logging to /home/jnano/.ros/log/5f413512-c5d6-11ed-b93b-1cbfcef65db5/roslaunch-macbot01-9913.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://macbot01:43907/
ros_comm version 1.14.13

SUMMARY
=====

PARAMETERS
* /rostdistro: melodic
* /rosversion: 1.14.13

NODES

auto-starting new master
process[master]: started with pid [9923]
ROS_MASTER_URI=http://macbot01:11311/

setting /run_id to 5f413512-c5d6-11ed-b93b-1cbfcef65db5
process[roscout-1]: started with pid [9936]
started core service [/roscout]
^I
```

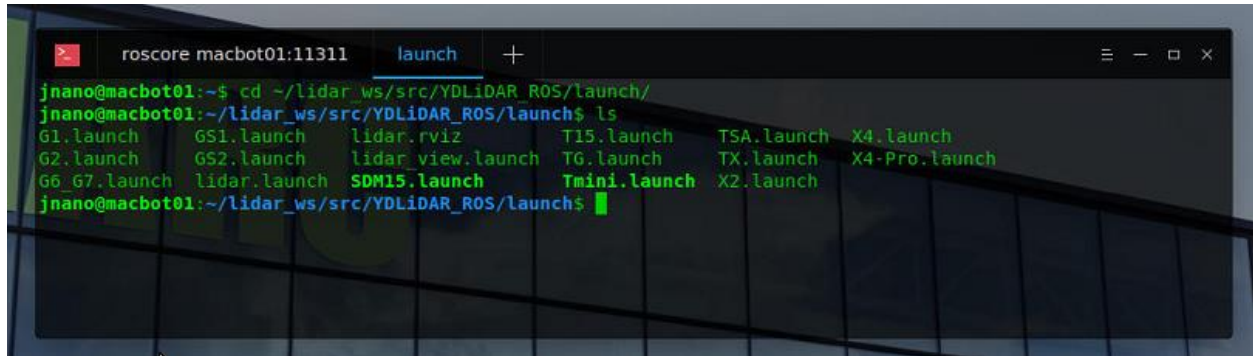
Open another terminal window (or tab, or split screen) and navigate to `~/lidar_ws/src/YDLiDAR/launch`.



List out the different launch files available to run.

```
cd ~/lidar_ws/src/YDLiDAR/launch
```

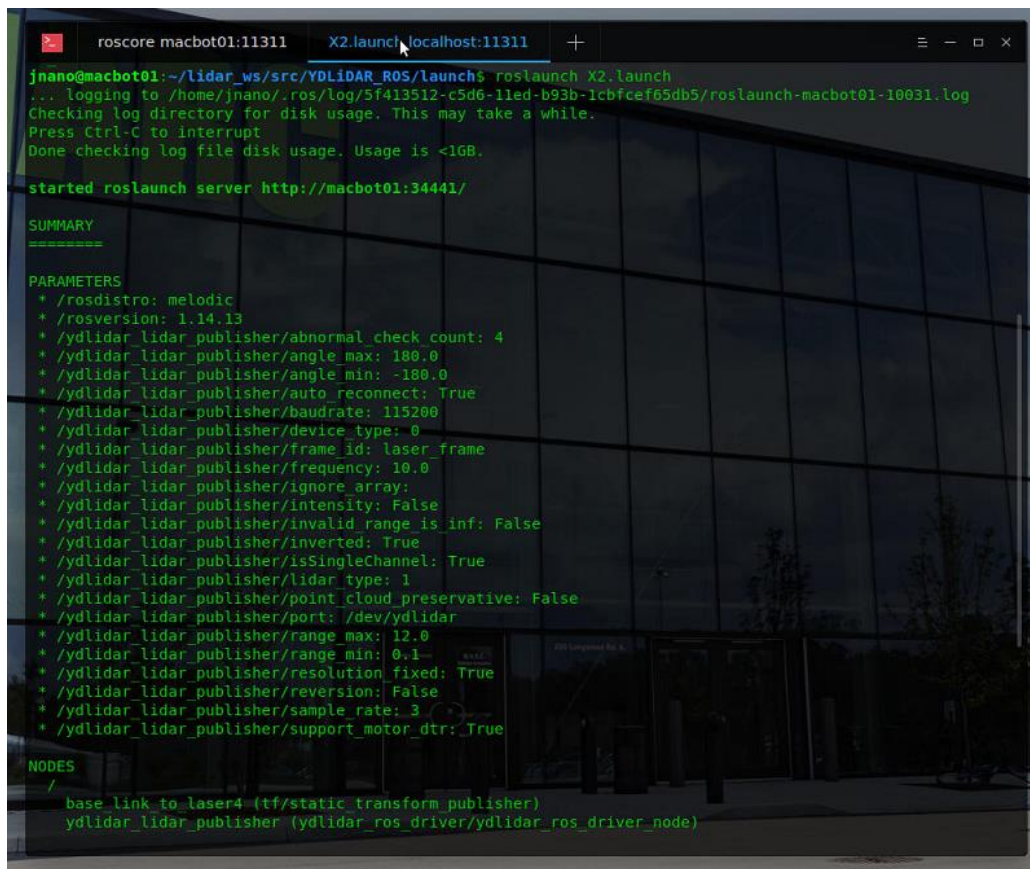
```
ls
```



```
roscore macbot01:11311 launch +
jnano@macbot01:~$ cd ~/lidar_ws/src/YDLiDAR_ROS/launch/
jnano@macbot01:~/lidar_ws/src/YDLiDAR_ROS/launch$ ls
G1.launch  GS1.launch  lidar.rviz  T15.launch  TSA.launch  X4.launch
G2.launch  GS2.launch  lidar_view.launch  TG.launch  TX.launch  X4-Pro.launch
G6 G7.launch  lidar.launch  SDM15.launch  Tmini.launch  X2.launch
jnano@macbot01:~/lidar_ws/src/YDLiDAR_ROS/launch$
```

Using the roslaunch command, launch the X2.launch file. Ensure that communication is established with the LiDAR. You will audibly notice a change in spin rotation speed.

```
roslaunch X2.launch
```



```
roscore macbot01:11311 X2.launch localhost:11311 +
jnano@macbot01:~/lidar_ws/src/YDLiDAR_ROS/launch$ roslaunch X2.launch
... logging to /home/jnano/.ros/log/5f413512-c5d6-11ed-b93b-1cbfcef65db5/roslaunch-macbot01-10031.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://macbot01:34441/

SUMMARY
=====

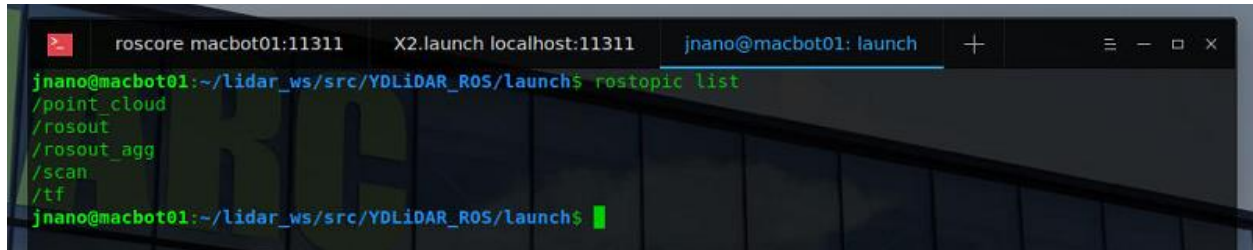
PARAMETERS
* /rostdistro: melodic
* /rosversion: 1.14.13
* /ydlidar_lidar_publisher/abnormal_check_count: 4
* /ydlidar_lidar_publisher/angle_max: 180.0
* /ydlidar_lidar_publisher/angle_min: -180.0
* /ydlidar_lidar_publisher/auto_reconnect: True
* /ydlidar_lidar_publisher/baudrate: 115200
* /ydlidar_lidar_publisher/device_type: 0
* /ydlidar_lidar_publisher/frame_id: laser_frame
* /ydlidar_lidar_publisher/frequency: 10.0
* /ydlidar_lidar_publisher/ignore_array:
* /ydlidar_lidar_publisher/intensity: False
* /ydlidar_lidar_publisher/invalid_range_is_inf: False
* /ydlidar_lidar_publisher/inverted: True
* /ydlidar_lidar_publisher/isSingleChannel: True
* /ydlidar_lidar_publisher/lidar_type: 1
* /ydlidar_lidar_publisher/point_cloud_preservative: False
* /ydlidar_lidar_publisher/port: /dev/ydlidar
* /ydlidar_lidar_publisher/range_max: 12.0
* /ydlidar_lidar_publisher/range_min: 0.1
* /ydlidar_lidar_publisher/resolution_fixed: True
* /ydlidar_lidar_publisher/reversion: False
* /ydlidar_lidar_publisher/sample_rate: 3
* /ydlidar_lidar_publisher/support_motor_dtr: True

NODES
/
  base_link_to_laser4 (tf/static_transform_publisher)
  ydlidar_lidar_publisher (ydlidar_ros_driver/ydlidar_ros_driver_node)
```

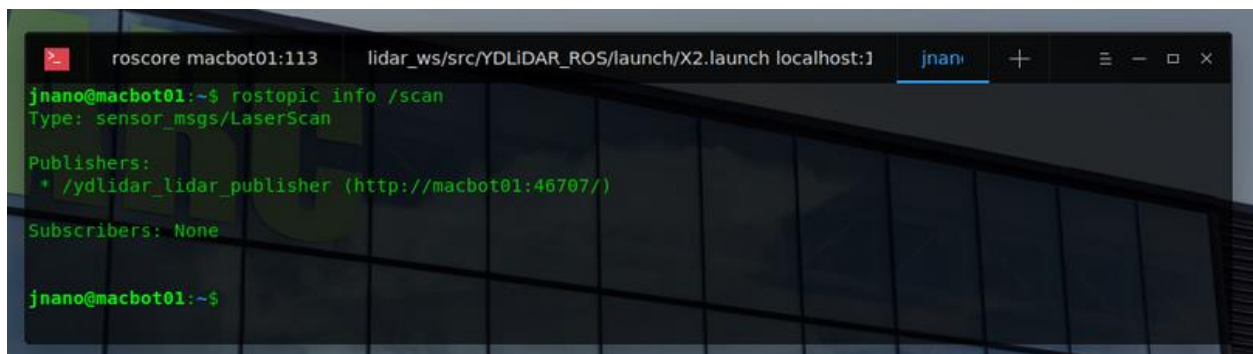
Next, open a new terminal window (or tab, or split-screen).

Use the `rostopic list` command to view all available streaming topics.

```
rostopic list
```

A terminal window with three tabs: 'roscore macbot01:11311', 'X2.launch localhost:11311', and 'jnano@macbot01: launch'. The active tab shows the command 'rostopic list' and its output: '/point\_cloud', '/rosout', '/rosout\_agg', '/scan', and '/tf'. The prompt is 'jnano@macbot01:~/lidar\_ws/src/YDLiDAR\_ROS/launch\$'.

Find more information about the `/scan` topic including the datatype and port its hosted on.

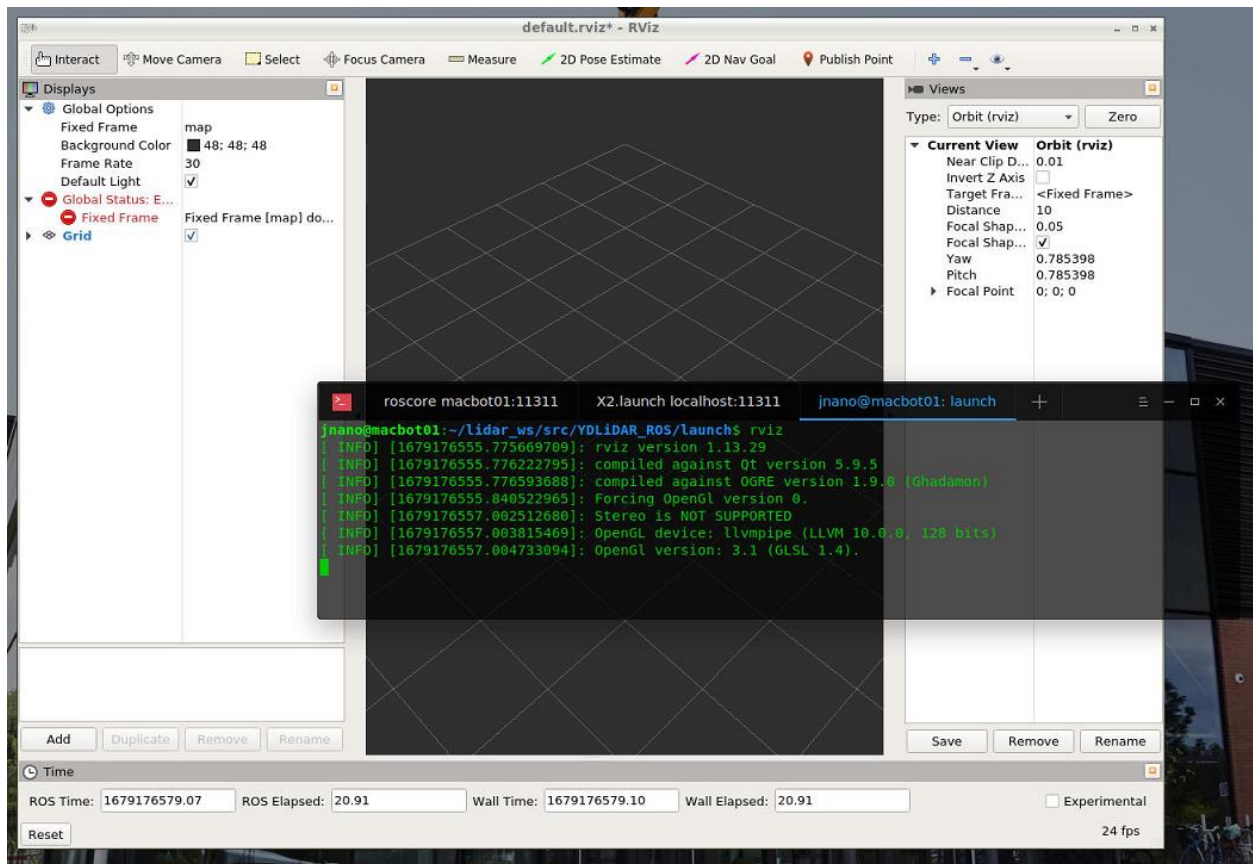
A terminal window with three tabs: 'roscore macbot01:113', 'lidar\_ws/src/YDLiDAR\_ROS/launch/X2.launch localhost:1', and 'jnano@macbot01: launch'. The active tab shows the command 'rostopic info /scan' and its output: 'Type: sensor\_msgs/LaserScan', 'Publishers: \* /ydlidar\_lidar\_publisher (http://macbot01:46707/)', and 'Subscribers: None'. The prompt is 'jnano@macbot01:~\$'.

Use the `rostopic echo` command to view the data being streamed on the `/scan` topic.

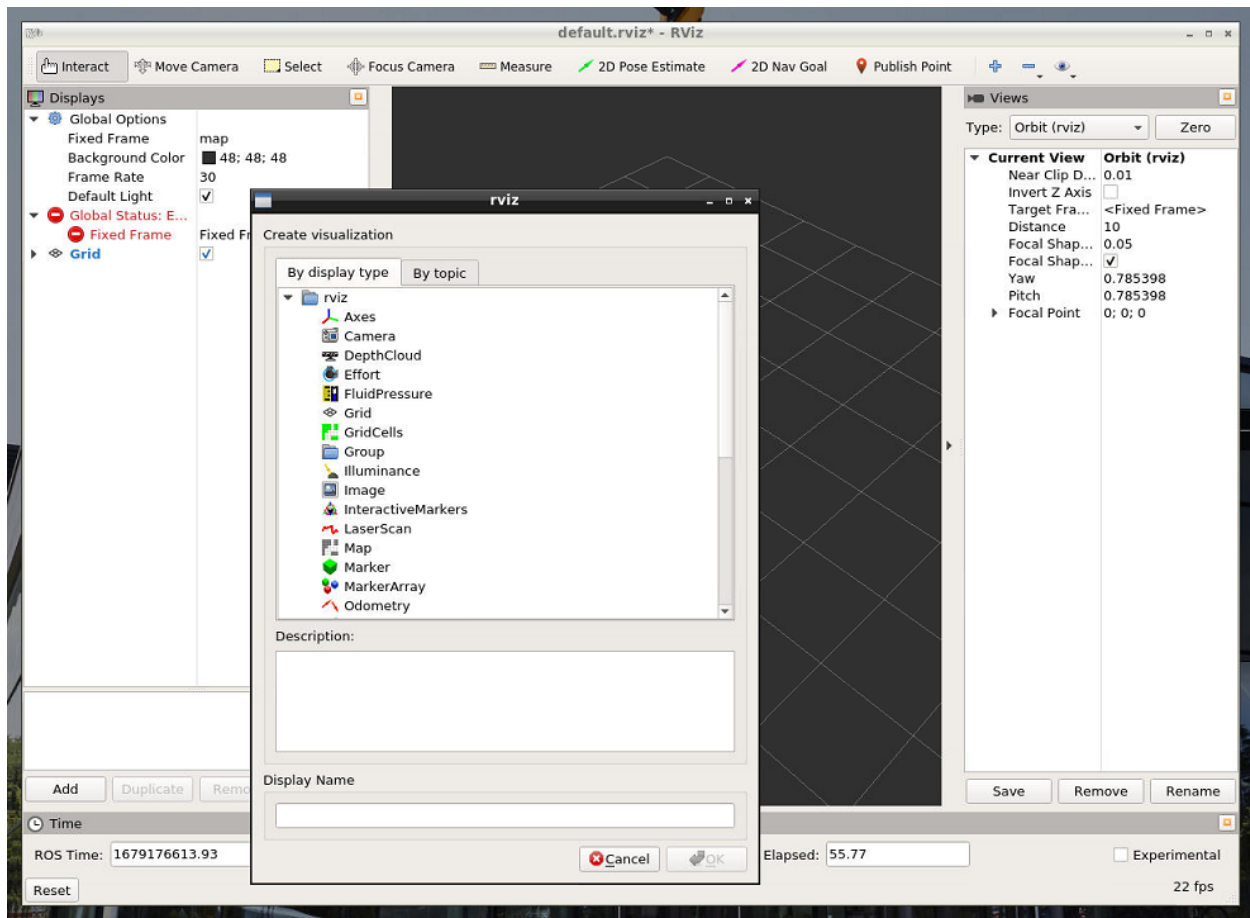
```
rostopic echo /scan
```



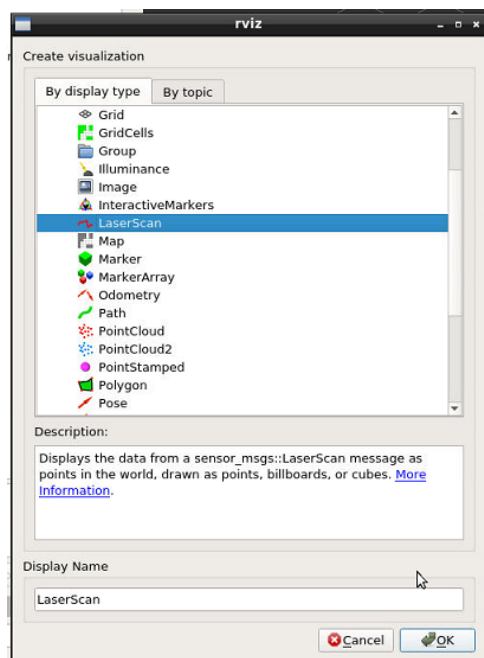




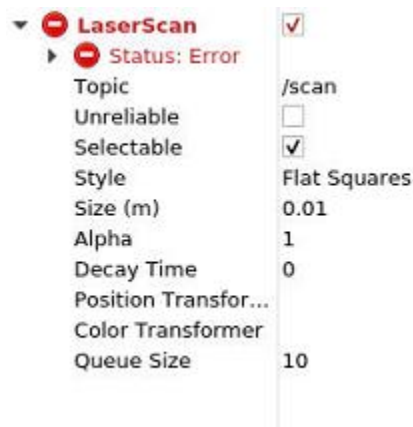
In the left pane, click **Add**. A window will appear to select the display type to add.



Choose laserscan and press OK.

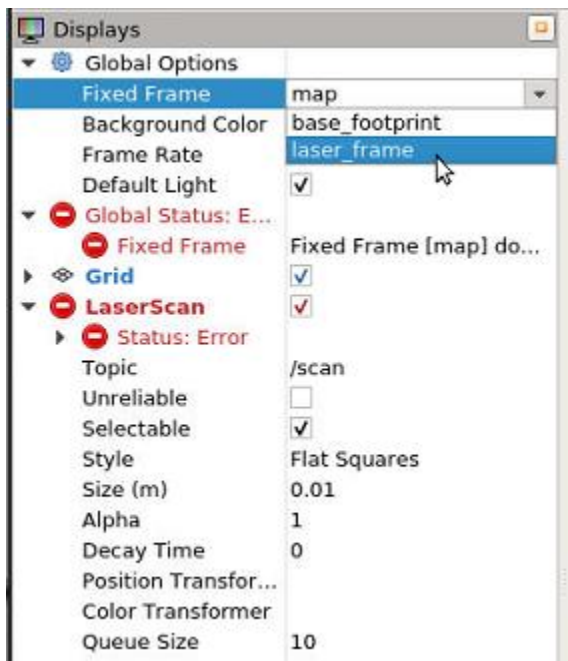


Back in the left pane, set the laserscan topic to /scan.



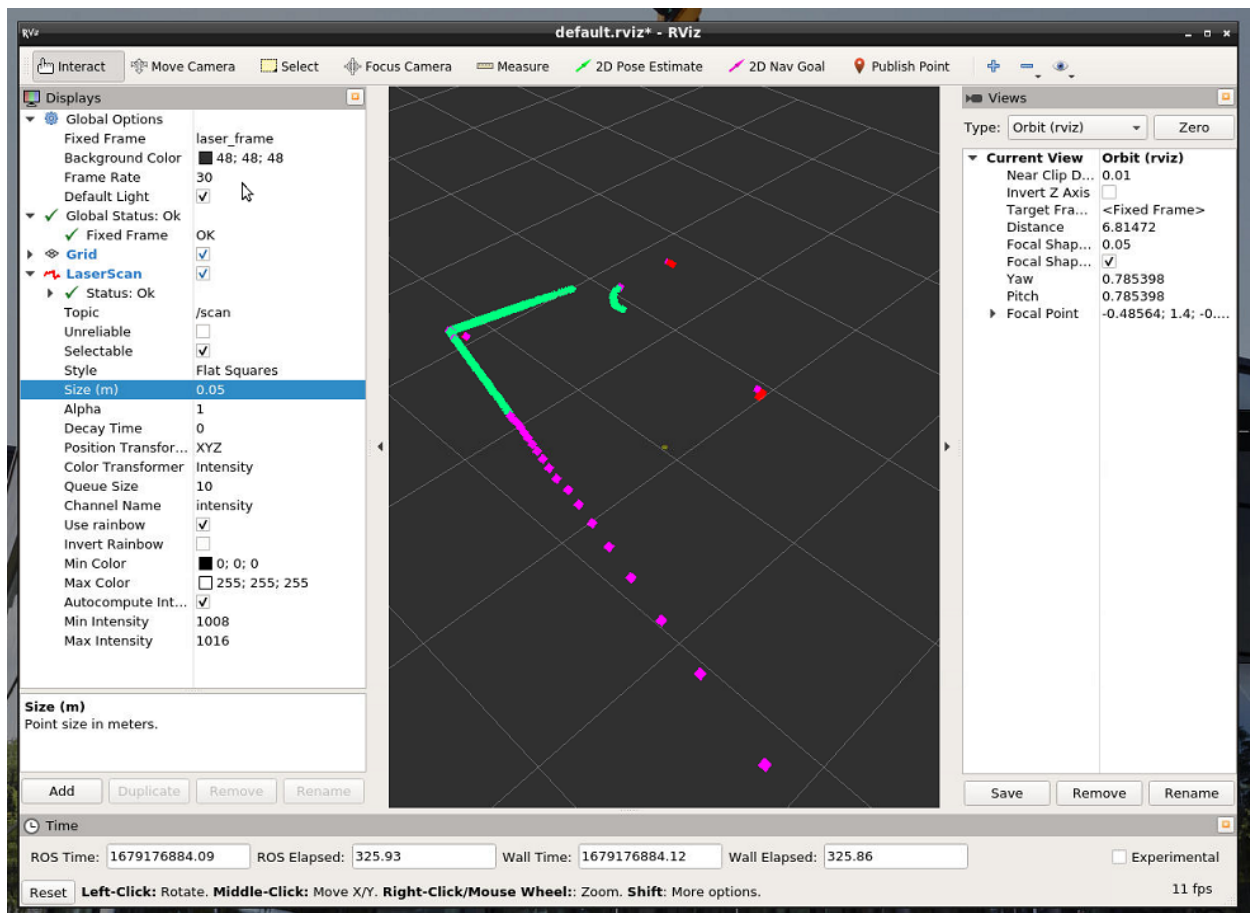
You will notice that RVIZ is in error state. This is because it does not have a reference point to plot the pointcloud data against.

Under **Display**, change the Fixed Frame to laser\_frame.

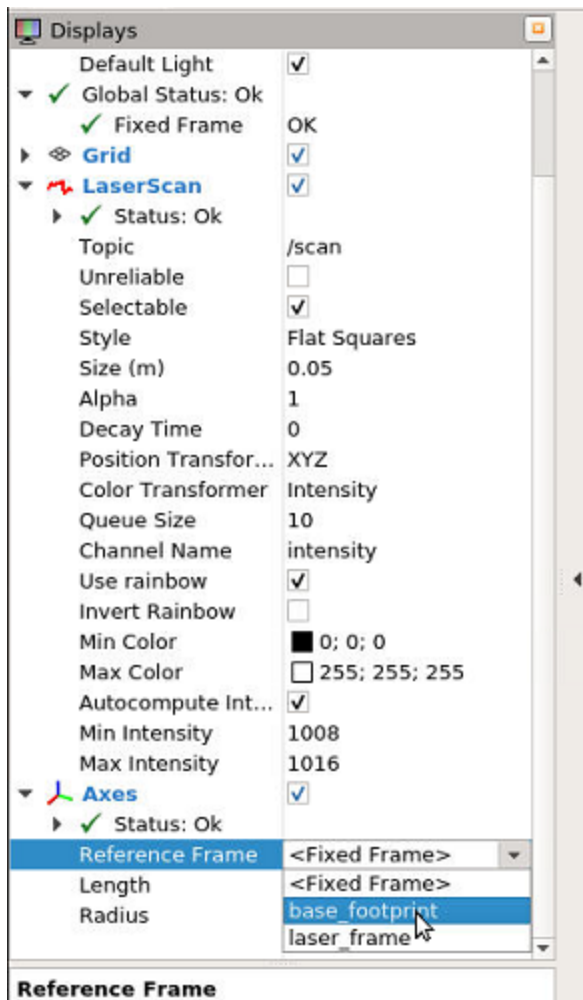


You should now observe data being visualized in RVIZ.

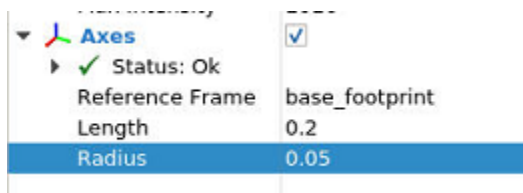
Modify Laser Scan > Size (m) to 0.05 to make the points a bit larger.

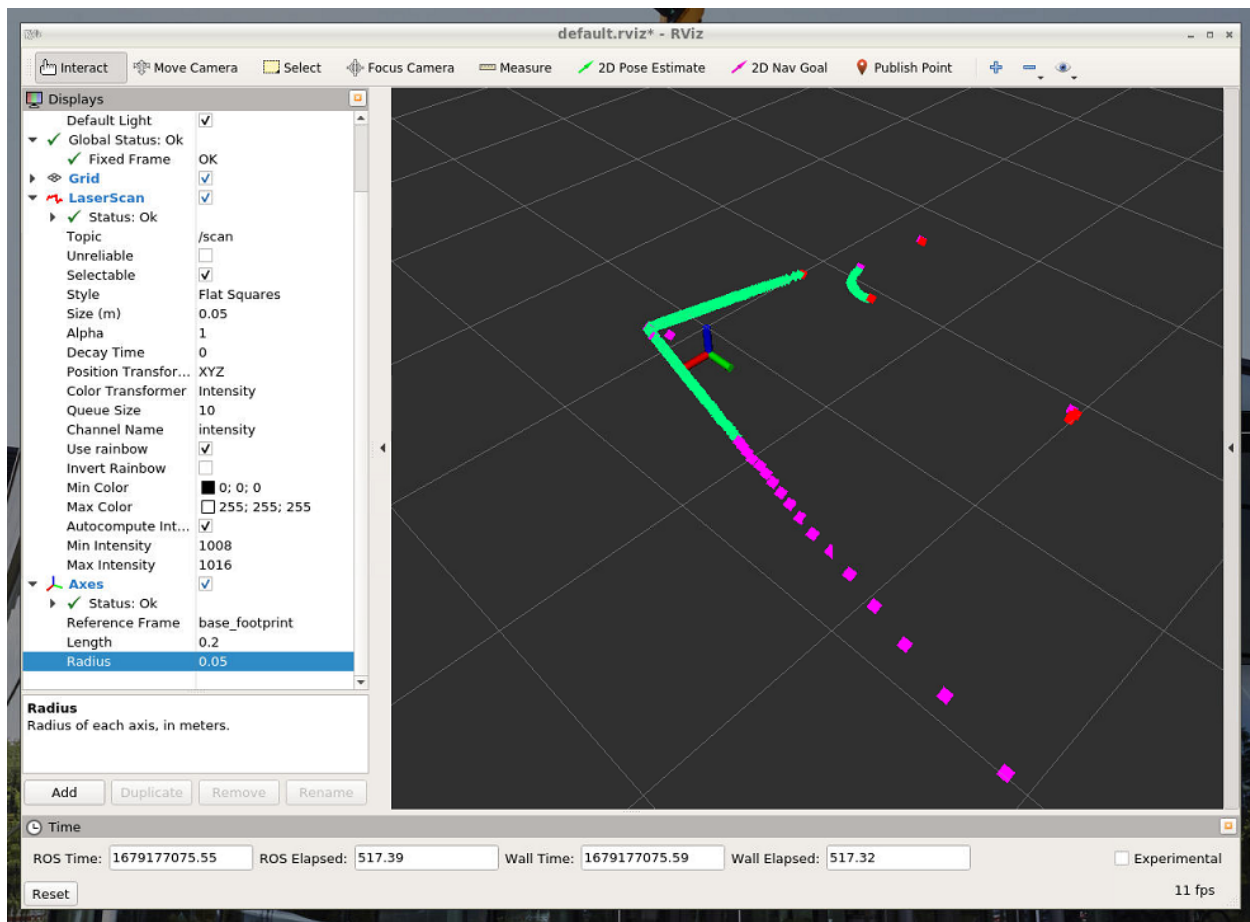


Lastly, let's input a marker for where the LiDAR is located. Insert an Axis display and set it to the base\_footprint frame.



Modify the Length and Radius values.





## Exercise A

You have successfully visualized LiDAR data. Take a screenshot of RVIZ and include it with your report.

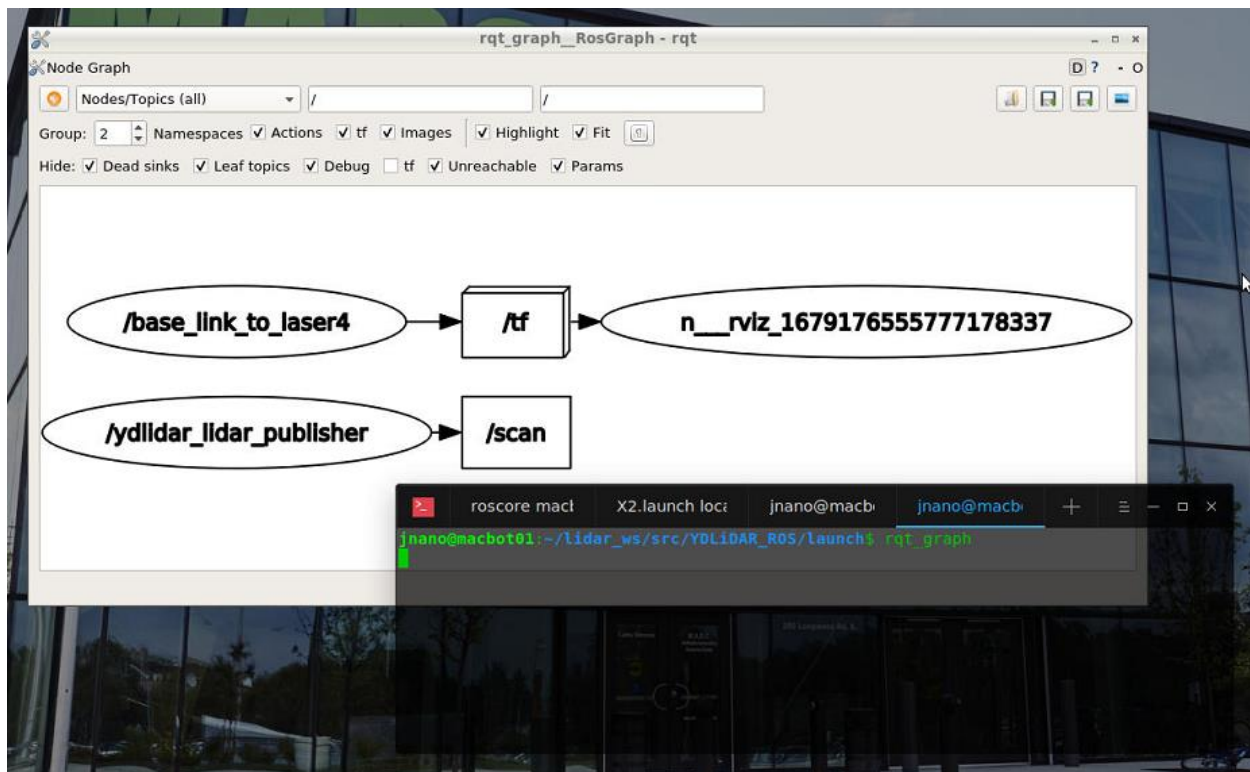
## Generating a Diagram

Lastly, let's use the `rqt_graph` tool to generate a live-updated diagram of our ROS system.

In a new terminal window (or tab, or split-screen), run the `rqt_graph` command.

```
rqt_graph
```

Set the graphical tool to display **Nodes/Topics (all)**



### Exercise B

You have successfully used the RQT\_Graph tool to generate a live diagram of your ROS project. Take a screenshot and include it with your report.