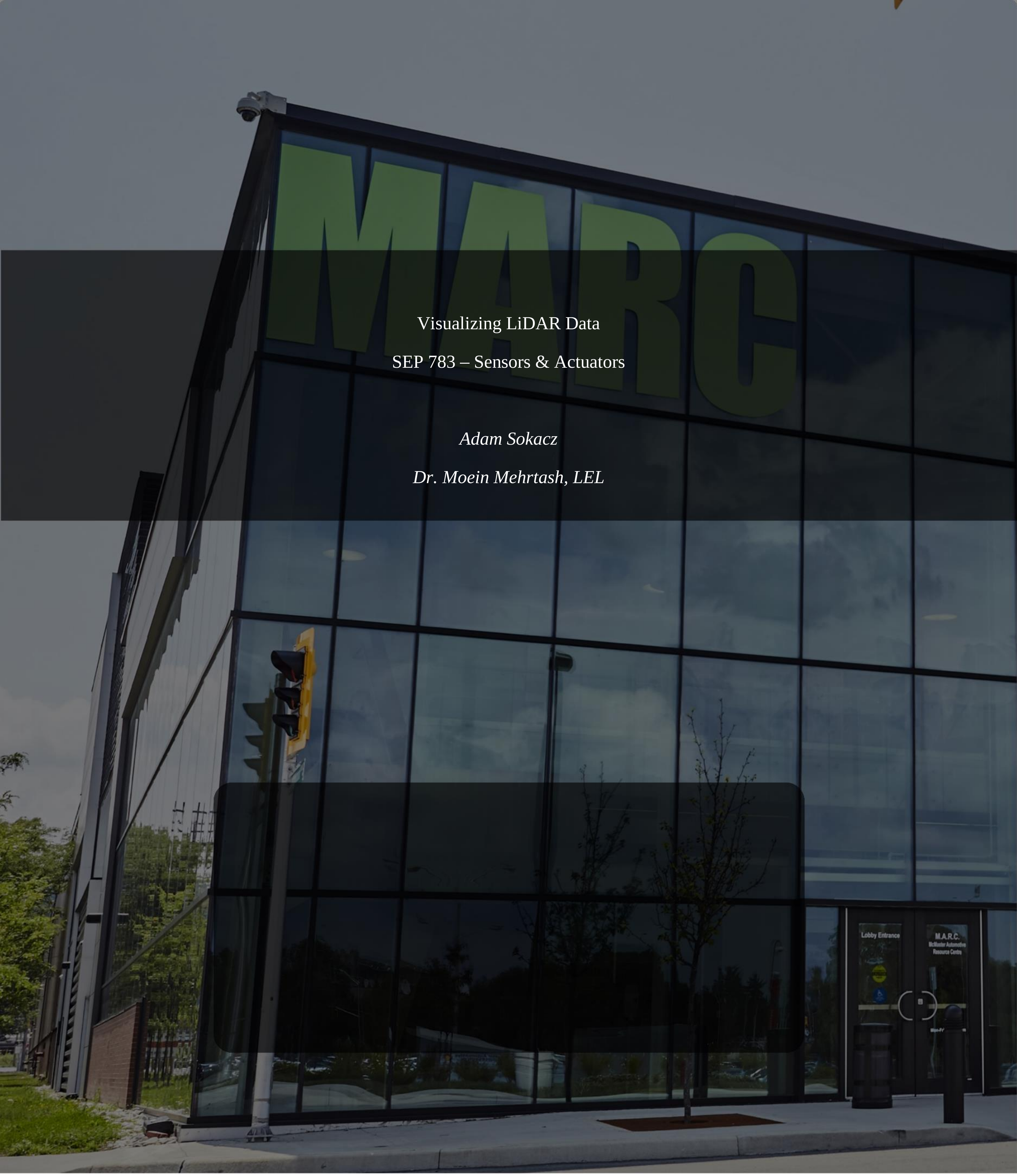


Visualizing LiDAR Data

SEP 783 – Sensors & Actuators

Adam Sokacz

Dr. Moein Mehrtash, LEL

W BOOTH SCHOOL OF ENGINEERING PRACTICE AND TECHNOLOGY
McMaster-Mohawk Bachelor of Technology Partnership
McMaster University ENGINEERING
mohawk

Objective

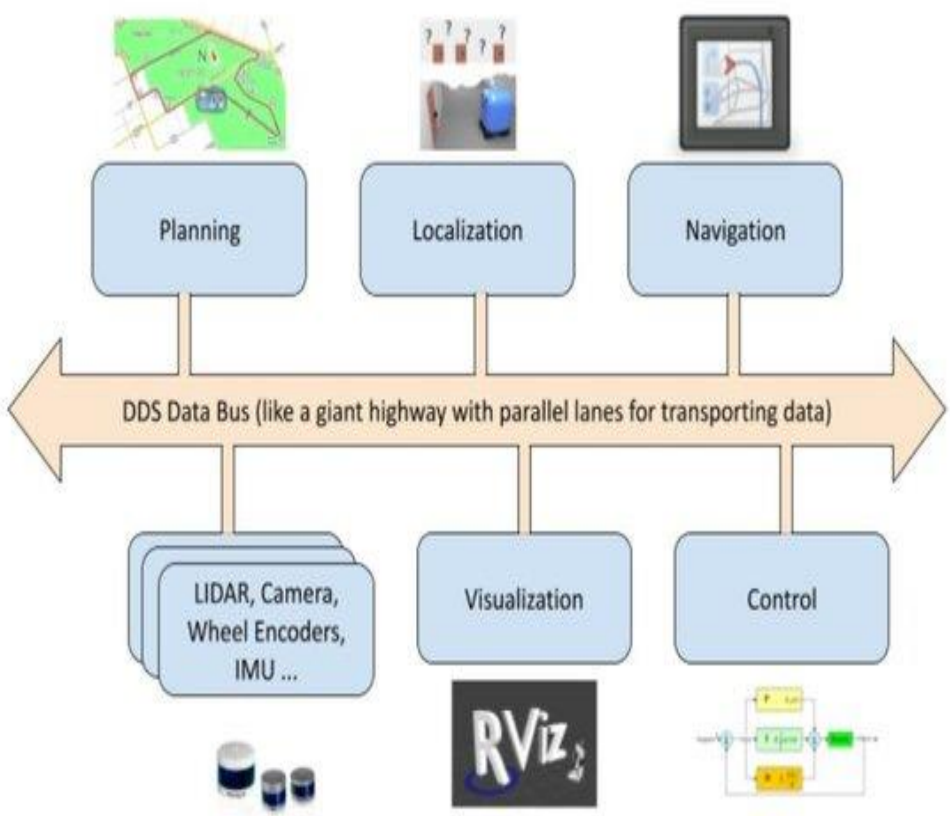
To familiarize yourselves with Linux, the basics of ROS middleware, and be comfortable reading and visualizing LiDAR sensor data.

Table of Contents

Overview to ROS

Sometime before 2007, the first pieces of what eventually would become Robot Operating System (ROS or ros) began coalescing at Stanford University. Eric Berger and Keenan Wyrobek, PhD students working in Kenneth Salisbury's robotics laboratory at Stanford, were leading the Personal Robotics Program. In their first steps towards this unifying system, the two built the PR1 as a hardware prototype and began to work on software from it, borrowing the best practices from other early open-source robotic software frameworks, particularly switchyard, a system that Morgan Quigley, another Stanford PhD student, had been working on in support of the STanford Artificial Intelligence Robot (STAIR) by the Stanford Artificial Intelligence Laboratory. The main ROS client libraries are geared toward a Unix-like system, mostly because of their dependence on large sets of open-source software dependencies.

On 1 September 2014, NASA announced the first robot to run ROS in space: Robotnaut 2, on the International Space Station. In 2017, the OSRF changed its name to Open Robotics. Tech giants Amazon and Microsoft began to take an interest in ROS during this time, with Microsoft porting core ROS to Windows in September 2018, followed by Amazon Web Services releasing RoboMaker in November 2018. Autoware is the world's leading open-source software project for autonomous driving. Autoware is built on Robot Operating System (ROS) and enables commercial deployment of autonomous driving in a broad range of vehicles and applications.

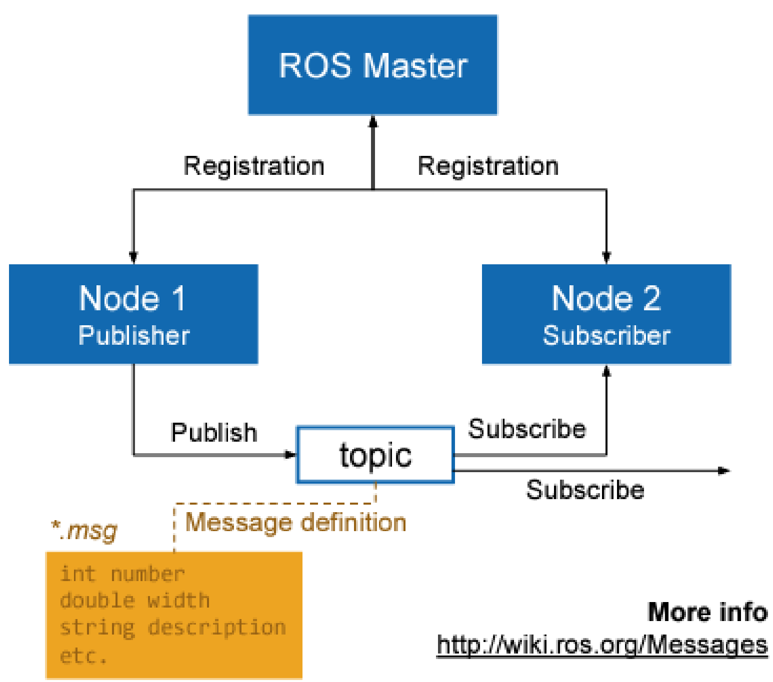


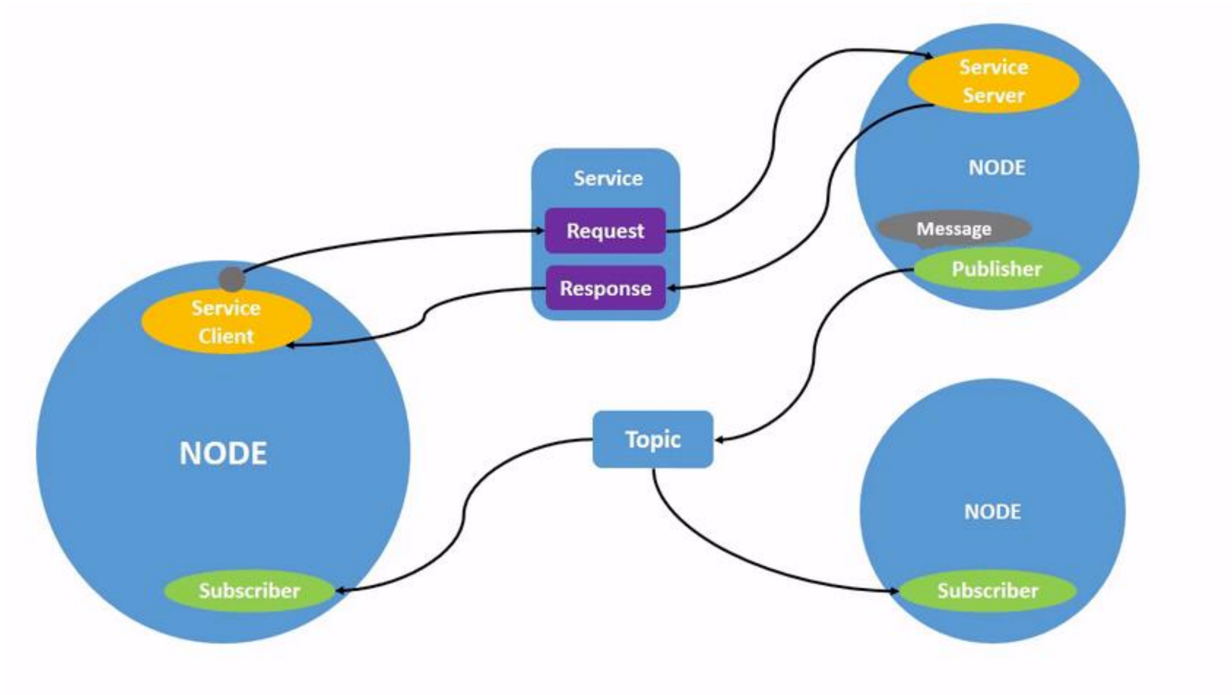
ROS philosophy can be summarized as,

- Peer to peer: Individual programs communicate over defined API (ROS messages, services, etc.)
- Distributed: Programs can be run on multiple computers and communicate over the network.
- Multi-lingual: ROS modules can be written in any language for which a client library exists (C++, Python, MATLAB, Java, etc.).
- Light-weight: Stand-alone libraries are wrapped around with a thin ROS layer
- Free and open-source

Roscore is a collection of nodes and programs that are prerequisites of a ROS-based system. You must have a roscore running in order for ROS nodes to communicate. It is launched using the roscore command.

- ROS Master manages the communication between nodes (processes). Every node registers at startup with the master
- ROS Nodes are Single-purpose, executable programs that are individually compiled, executed, and managed. They are organized in "Packages"
- ROS Topics: Nodes communicate over topics
- ROS Messages: Data structure defining the type of a topic



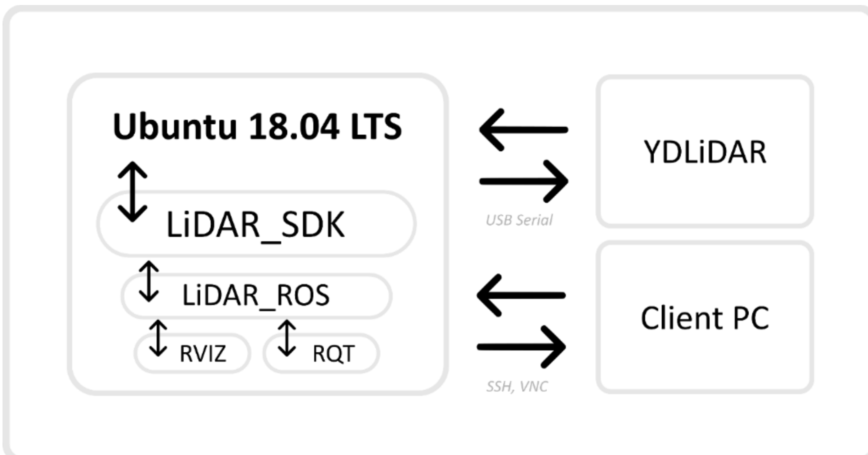

Service
Server
NODE
Service
Request
Response
Message
Publisher
Service
Client
NODE
Topic
NODE
Subscriber
Subscriber

-
-

Intro to Lidar as a Node

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 an excellent introduction to ROS, and sensor visualization using ROS. I hope you enjoy it!



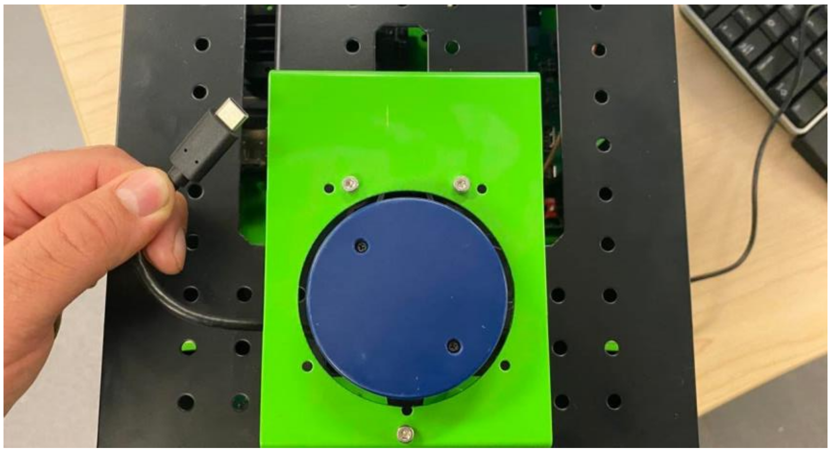
Procedure of today's lab in brief:

- Connection to MacBot
- Setup Lidar Driver (sdk) cmake
- Install Lidar workspace (Catkin)
- Install Lidar on ROS Catkin
- Mater Node Rosecore
- Launch and rviz

MacBot Setup

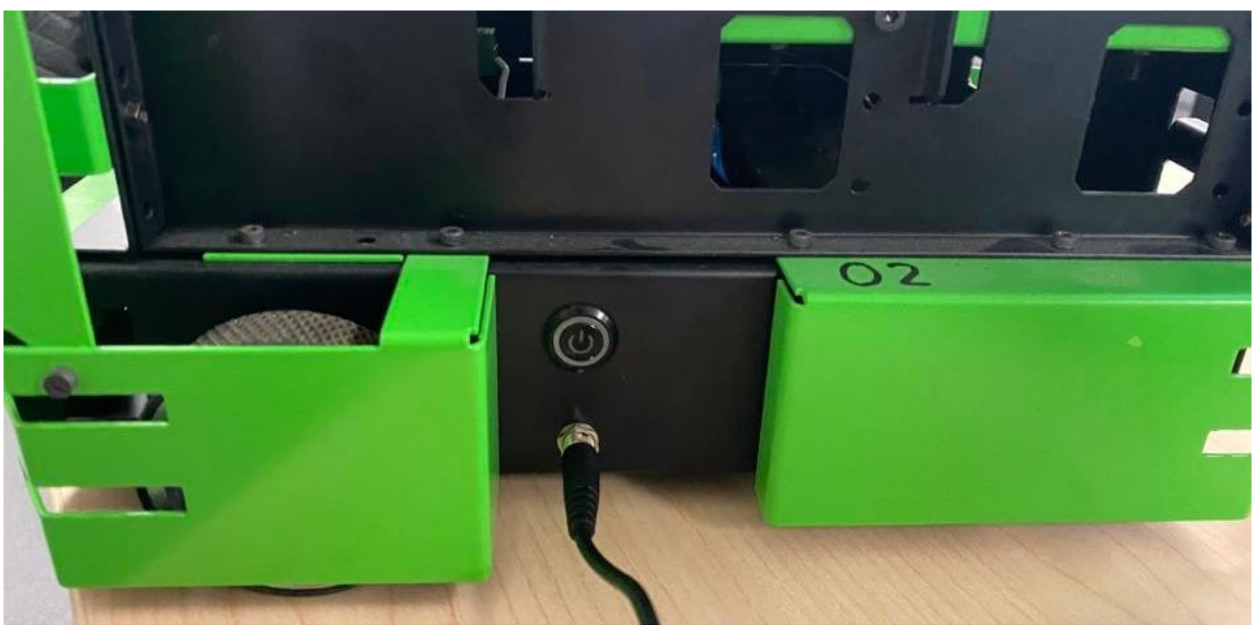
Disconnecting the LiDAR

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.

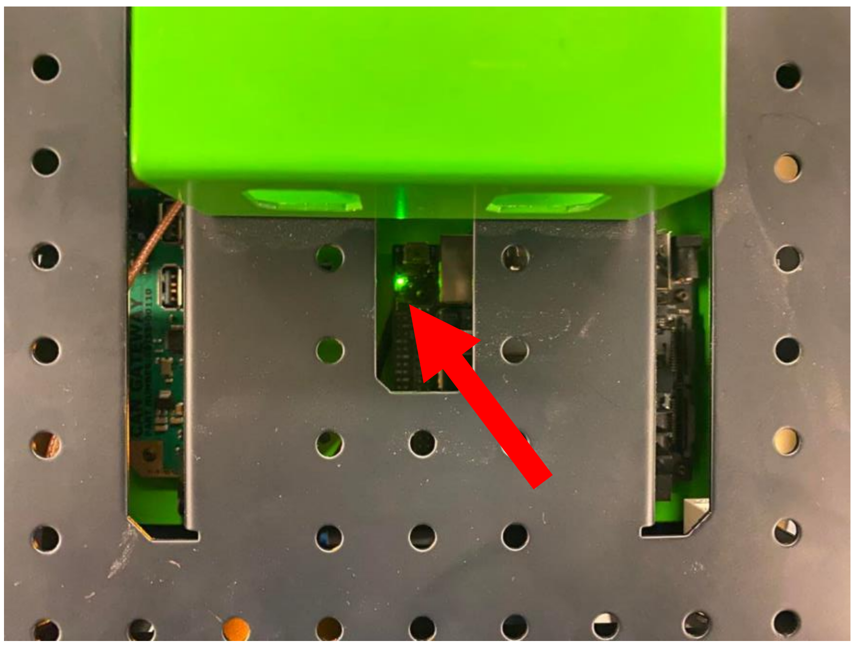


Powering on the MacBot

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 using wifi

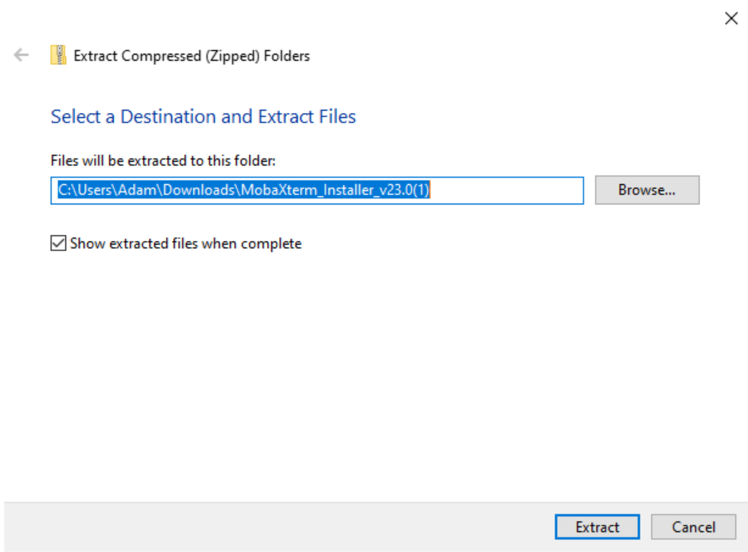
MobaXTerm allows accessing multiple sessions on remote servers with a secure SSH connection. It helps to copy the file from server to host with just drag and … . We can use MobaXTerm to connect to Macbots with Ubunto operating system. Download Install MobaXTerm from:

https://download.mobatek.net/2302023012231703/MobaXterm_Installer_v23.0.zip

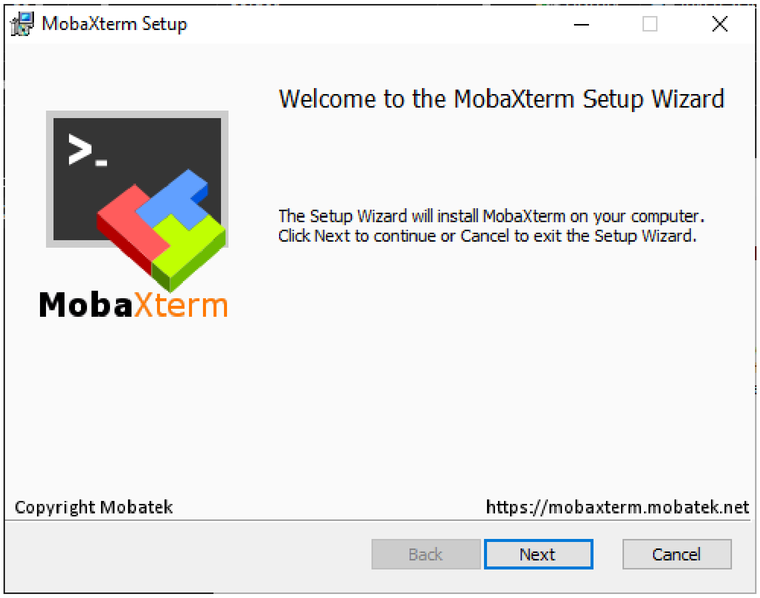
Or from Avenue to learn.

Follow the instructions: (You can also follow these steps from https://adam-36.gitbook.io/macbot/)

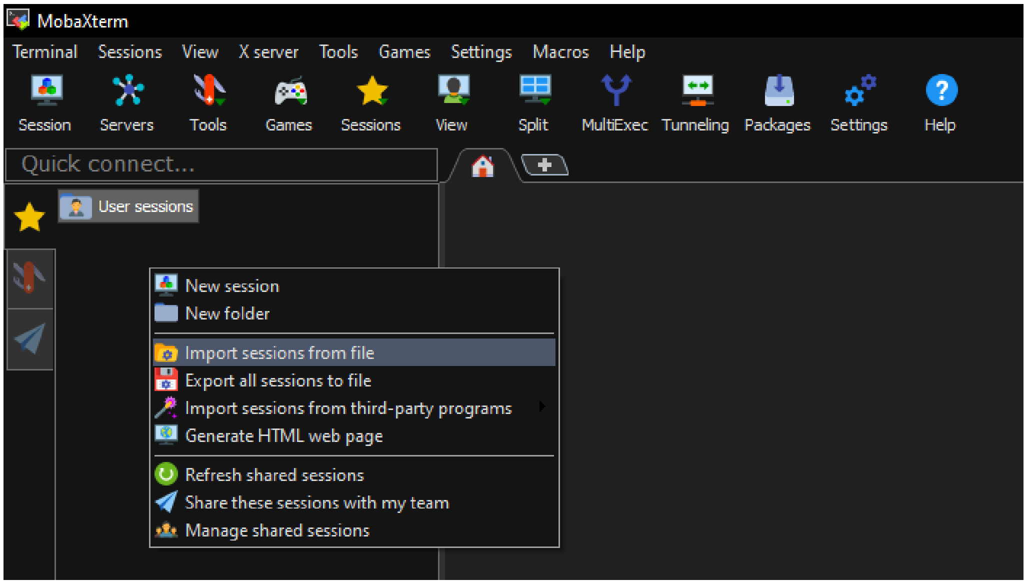
1. Use 7Zip or Windows Zip Extractor to extract the compressed folder into the Downloads/ folder

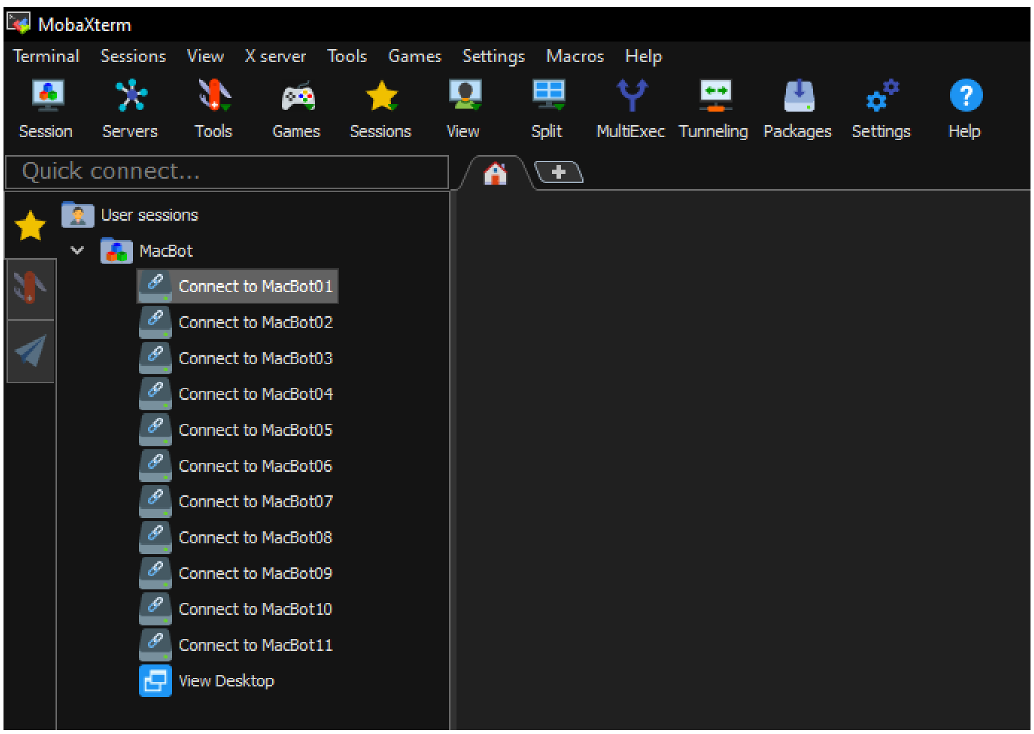


2. Run the MobaXTerm MSI installer with the default install configuration.

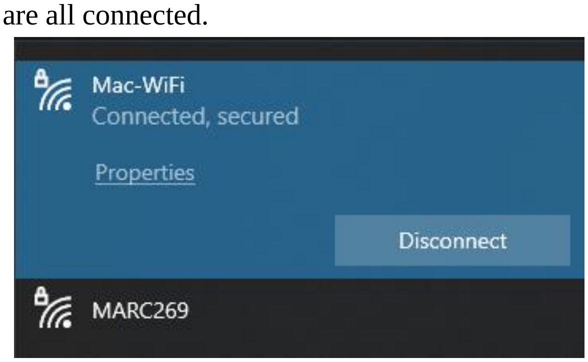


3. Download the MobaXT session: “MacBot.mxtsessions” file to your Downloads/ folder, it is avilable on the Avenue to Learn. It contains the pre-configured environment (IP address) for establishing an SSH tunnel to and streaming a graphical interface from each MacBot.
4. Open MobaXTerm and load the downloaded session file.

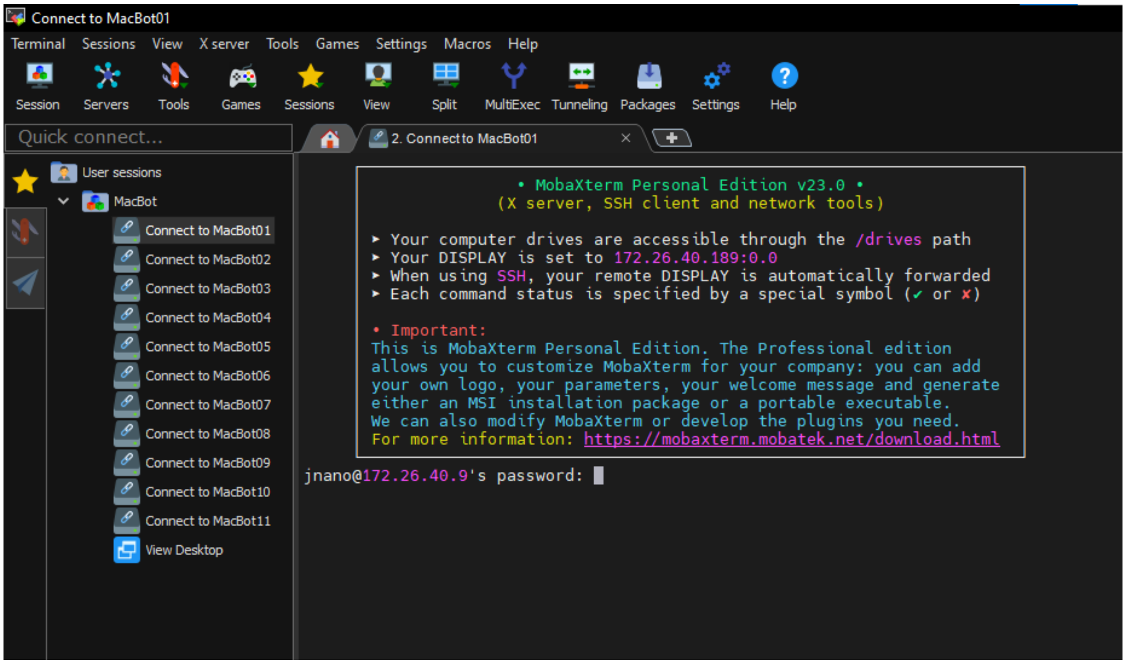




5. If you are using your own Laptop, make sure to connect to Mac-WiFi. If you are using the PCs in the computer Lab, they are all connected.



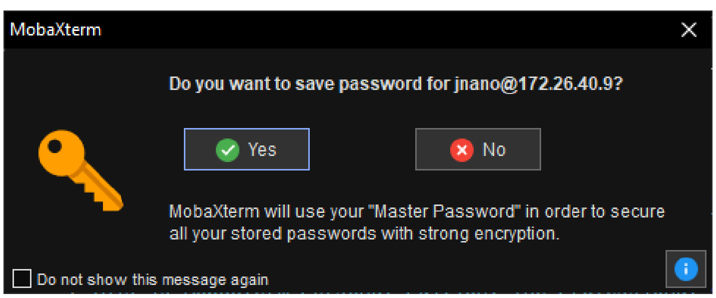
6. Ensure that your MacBot is powered ON and booted by waiting 2 minutes.
7. Click Connect to MacBot for your group (there is MacBot # on each device)



Enter the password for your MacBot, which can be found here:

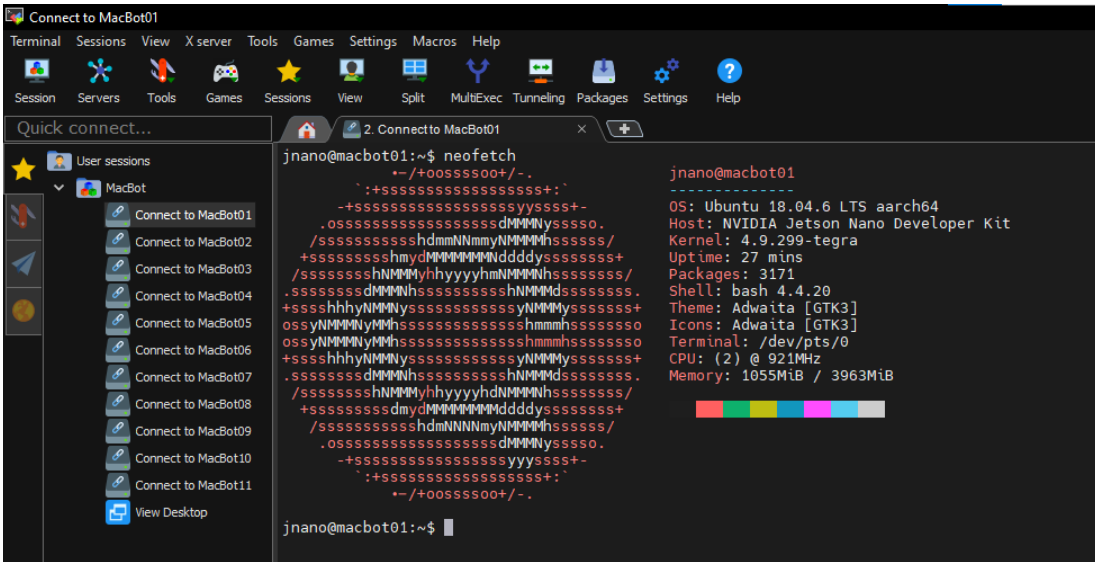
User: jnano
Pass: 9055259140

When prompted to save the password, always select No.



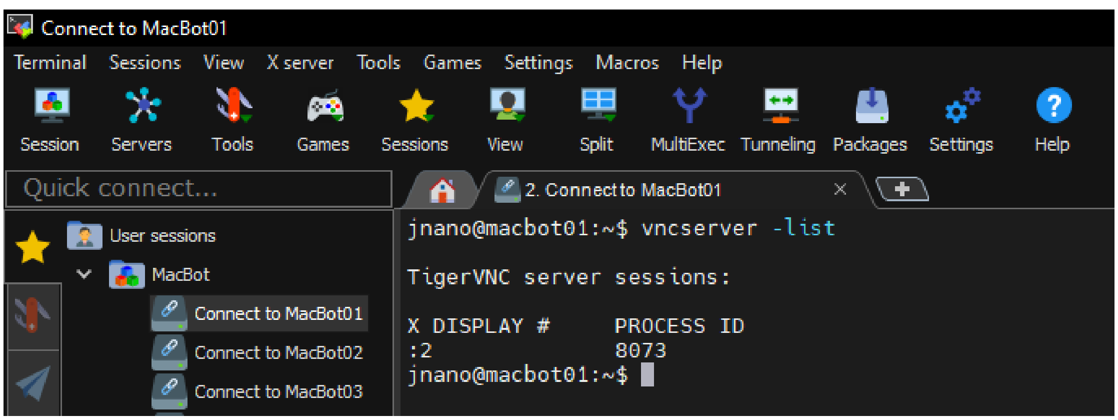
8. Run neofetch to verify that you are connected to the correct machine. It shows the machine properties:

```
neofetch
```

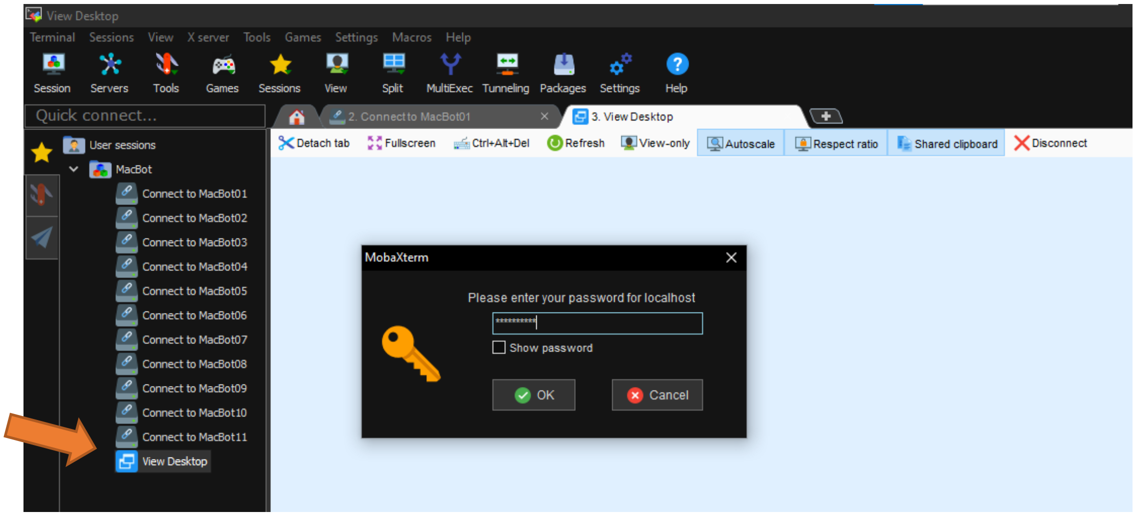


9. Ensure that the VNC server is running on port 5902 using the following command:

```
vncserver -list
```



10. Connect to the MacBot by pressing the View Desktop button.



Pass: 9055259140

Ensure that you see a remote desktop connection in MobaXTerm. You can make it “Fullscreen” and unckeck “Stay on top”. You can use “Alt +Tab” to switch between windows.

Setting Up the LiDAR Drivers

You must use MobaXTerm to connect to Macbot and install Lidar packages. You need to perform this section on the installed Linux on Macbot.

Downloading the YDLiDAR SDK

You need to download software development kit (SDK) for the X2 lidar (https://www.ydlidar.com/products/view/6.html) . A software development kit (SDK) is a collection of software development tools in one installable package.

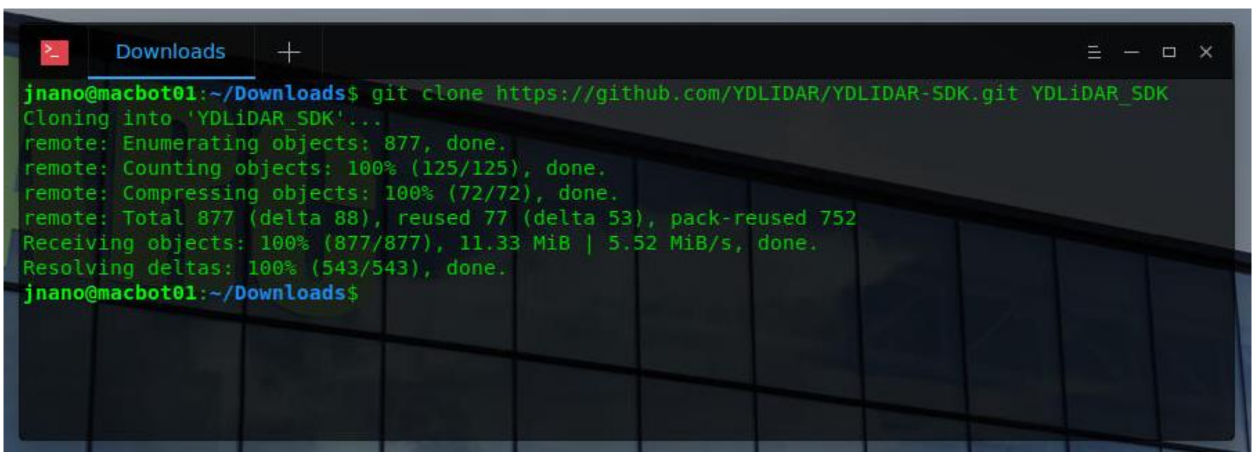
Navigate to your ~/Downloads/ folder in the command Terminal (LXTerminal):

```
cd ~/Downloads
```

“git” clone is a Git command line utility which is used to target an existing repository and create a clone, or copy of the target repository. 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
```



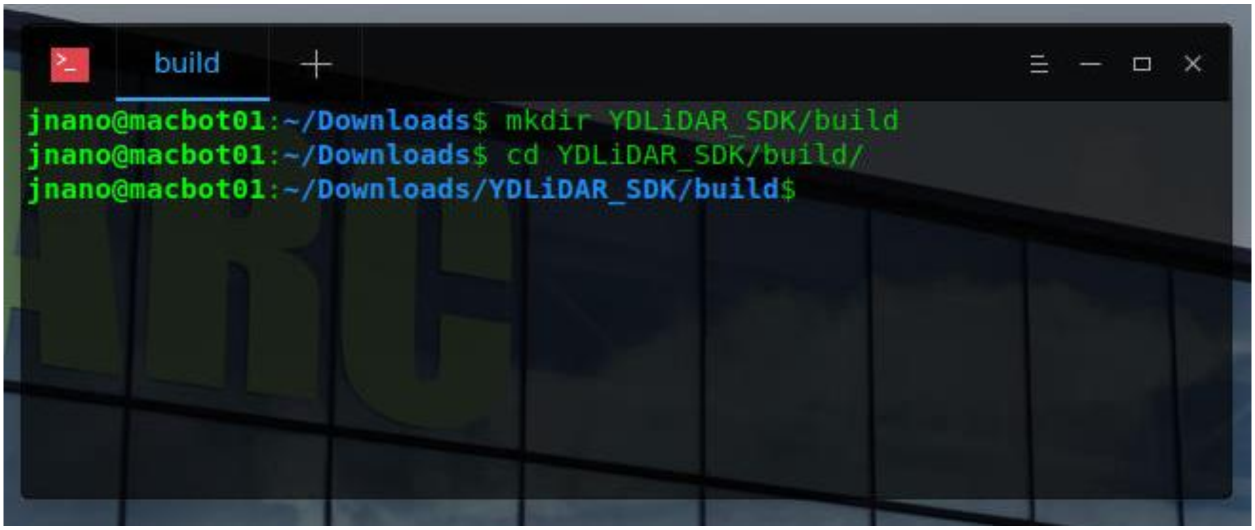
Building the YDLiDAR SDK

Create a folder (directory) inside of YDLiDAR_SDK/ called build/.

```
mkdir YDLiDAR_SDK/build
```

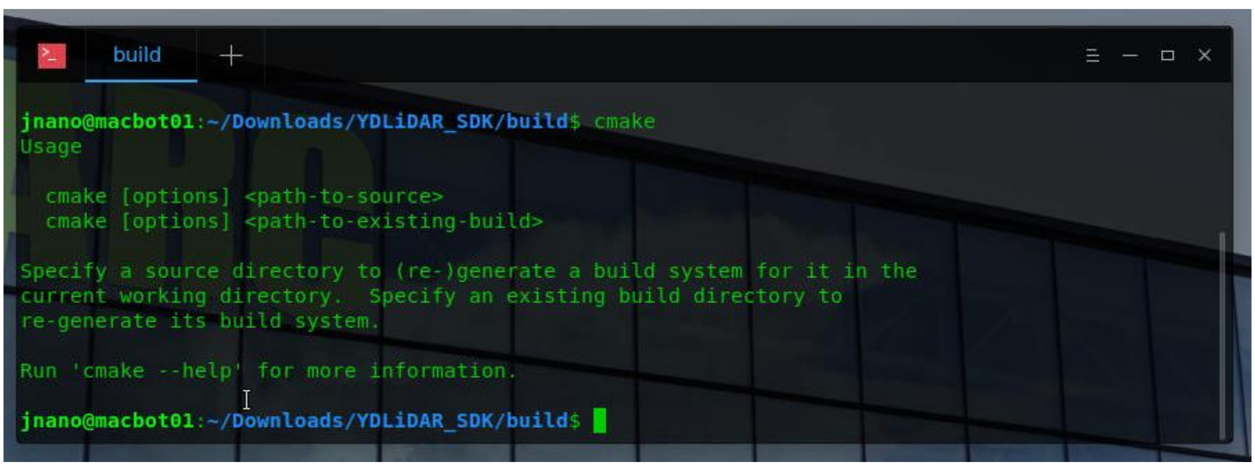
Navigate to it

```
cd YDLiDAR_SDK/build/
```



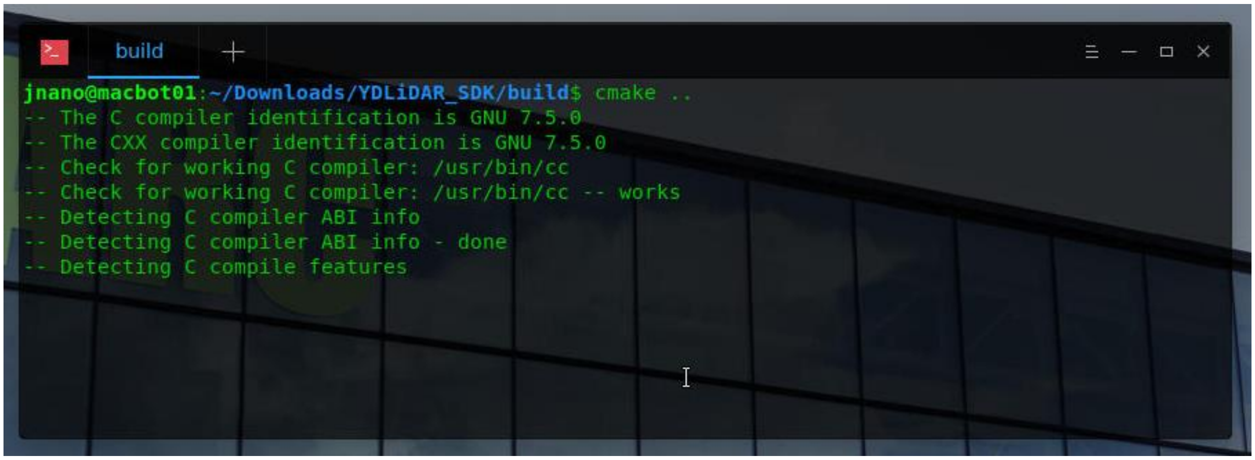
CMake is an open-source, cross-platform family of tools designed to build, test, and package software. CMake controls the software compilation process using simple platform and compiler-independent configuration files, and generates native makefiles and workspaces that can be used in the compiler environment of your choice. Run cmake and see what paths it requires.

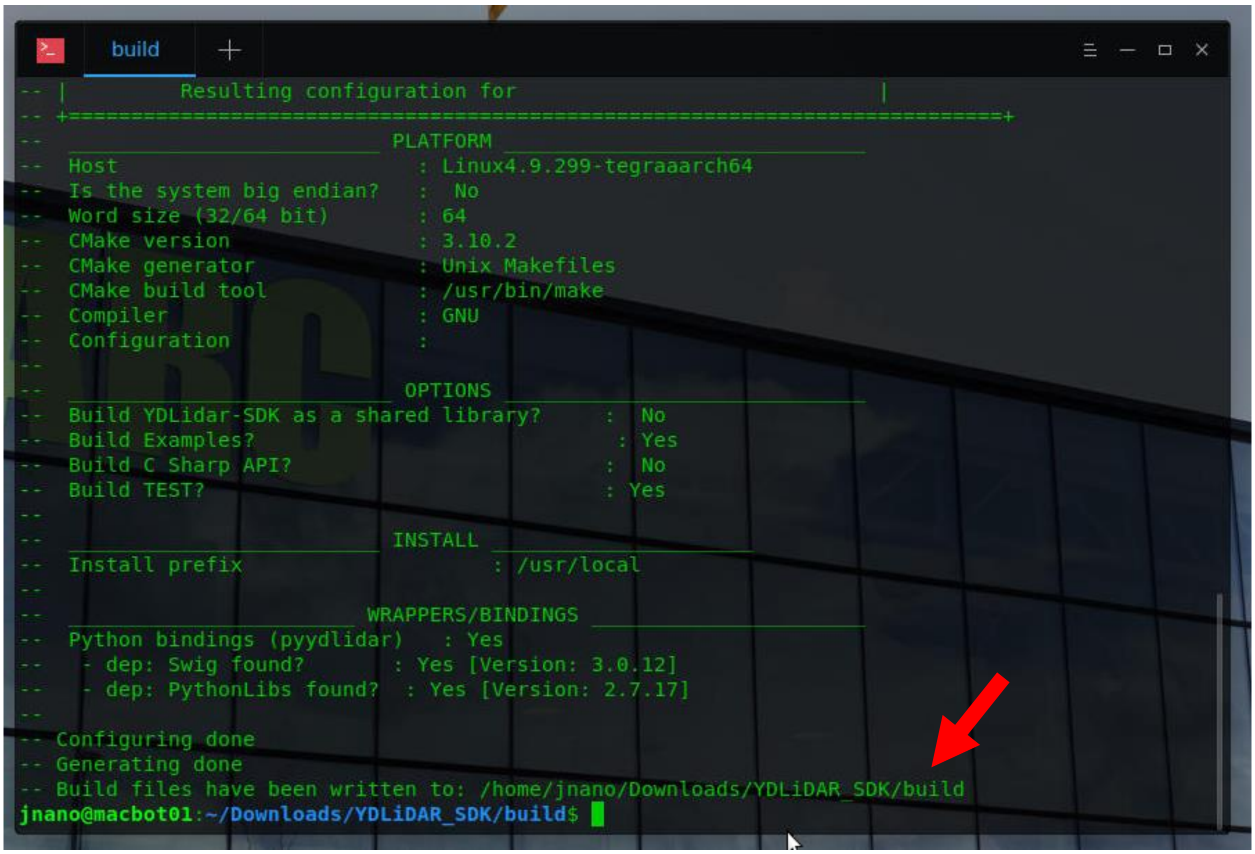
cmake



Run cmake on the project.

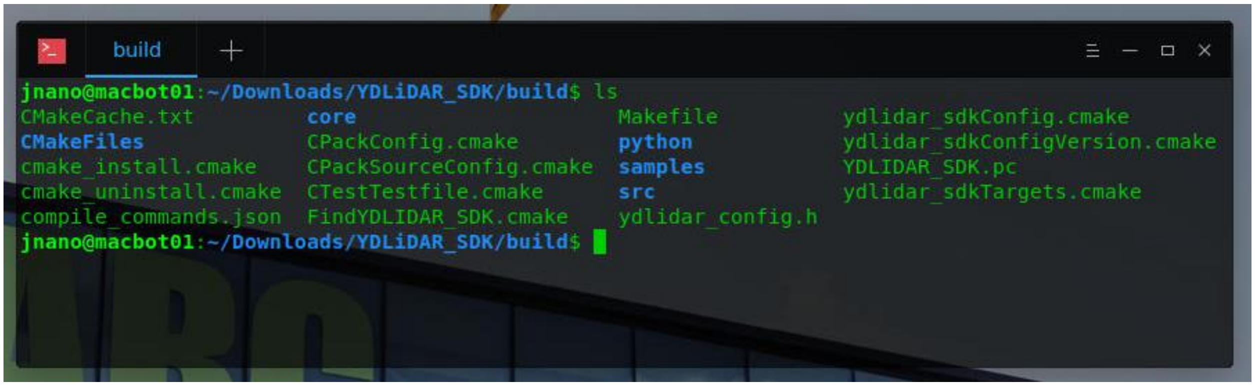
cmake ..





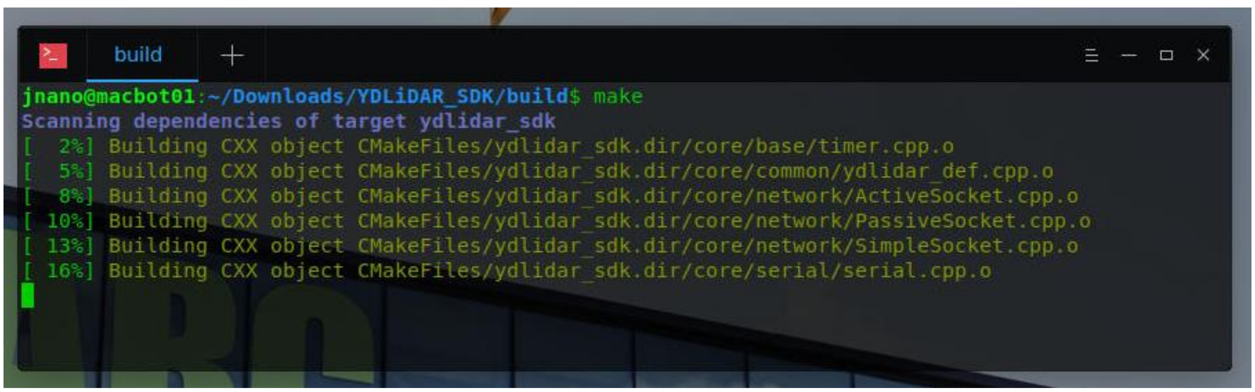
Notice that build files have been created. Use the ls command to display the contents of a directory.

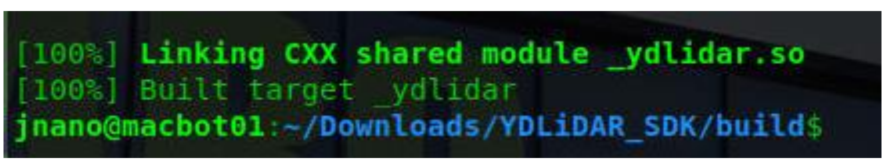
ls



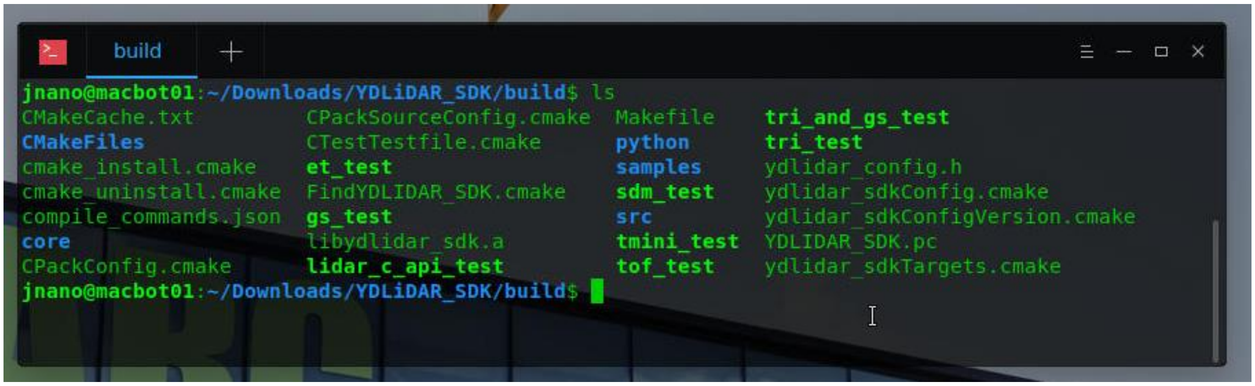
Make creates executables from the source files, which have to include a Makefile. In contrast, when using CMake, a CMakeLists. txt file is provided, which is used to create a Makefile. Run make on the project.

Make





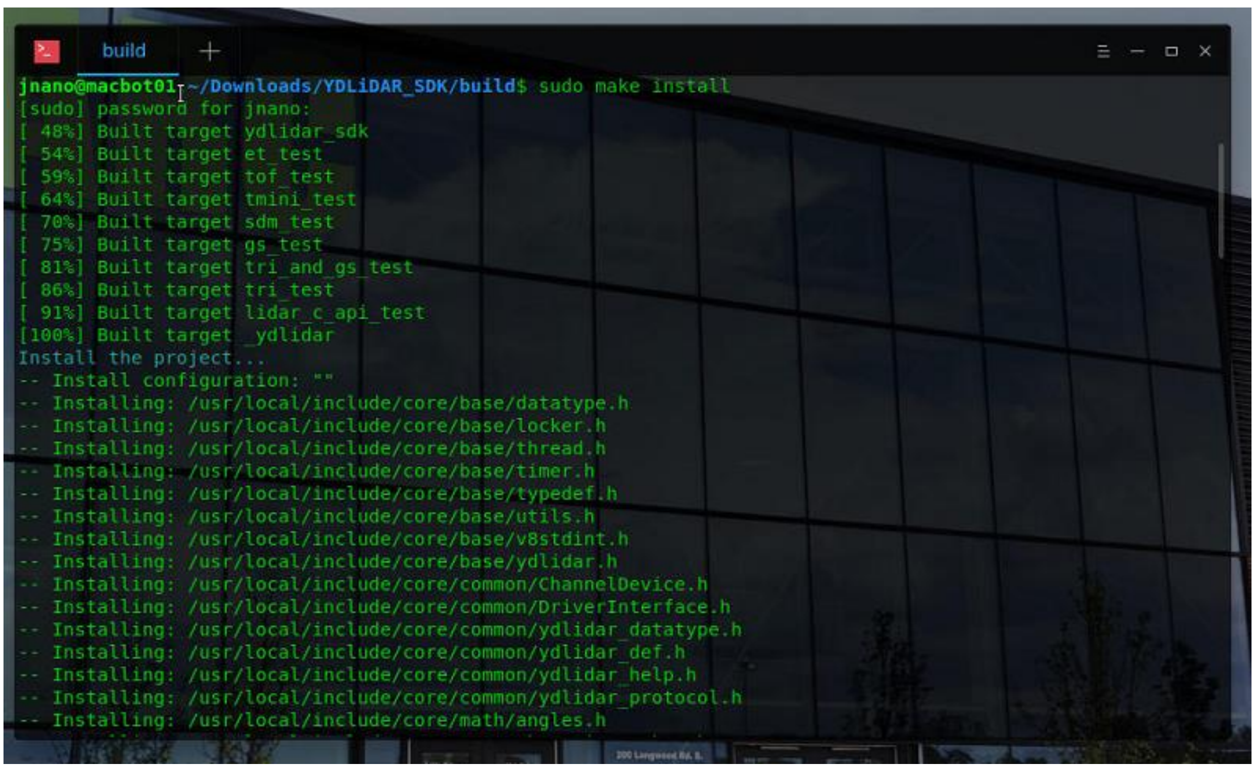
Notice that the project has been built.



Installing YDLiDAR_SDK onto the MacBot

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

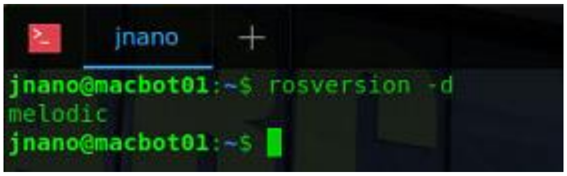
sudo make install



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



Creating and Sourcing a ROS Workspace

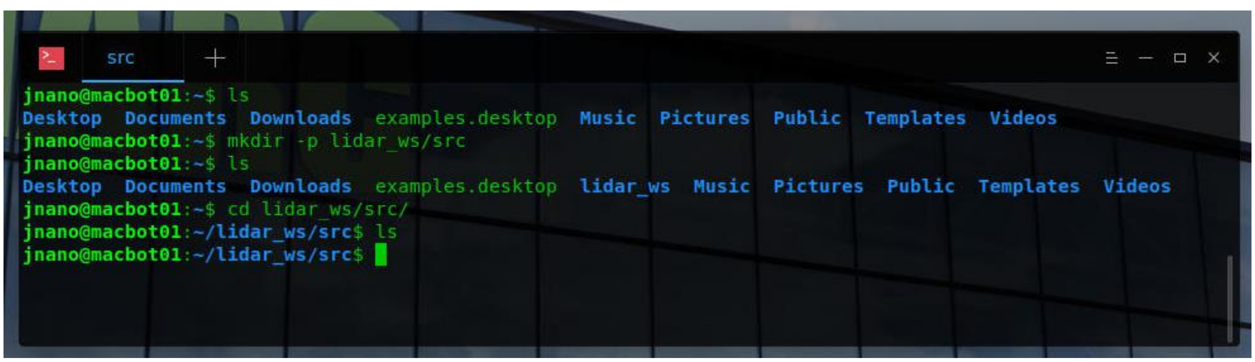
A workspace is a set of directories (or folders) where you store related pieces of ROS code. The official name for workspaces in ROS is catkin workspaces. catkin packages can be built as a standalone project, in the same way that normal cmake projects can be built, but catkin also provides the concept of workspaces, where you can build multiple, interdependent packages together all at once. Navigate to the home/ directory using the following command:

```
cd ~
```

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

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

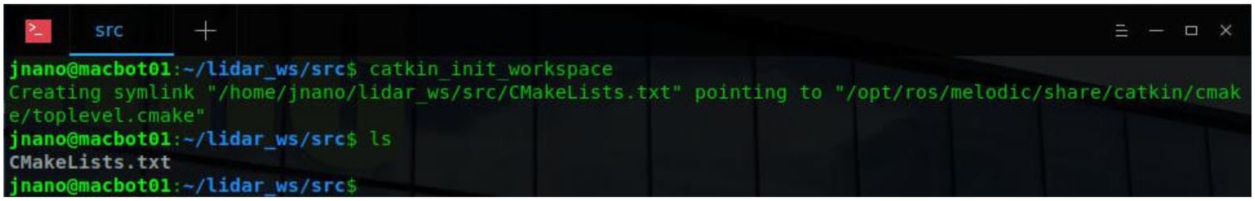
```
cd lidar_ws/src
```



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

```
catkin_init_workspace
```

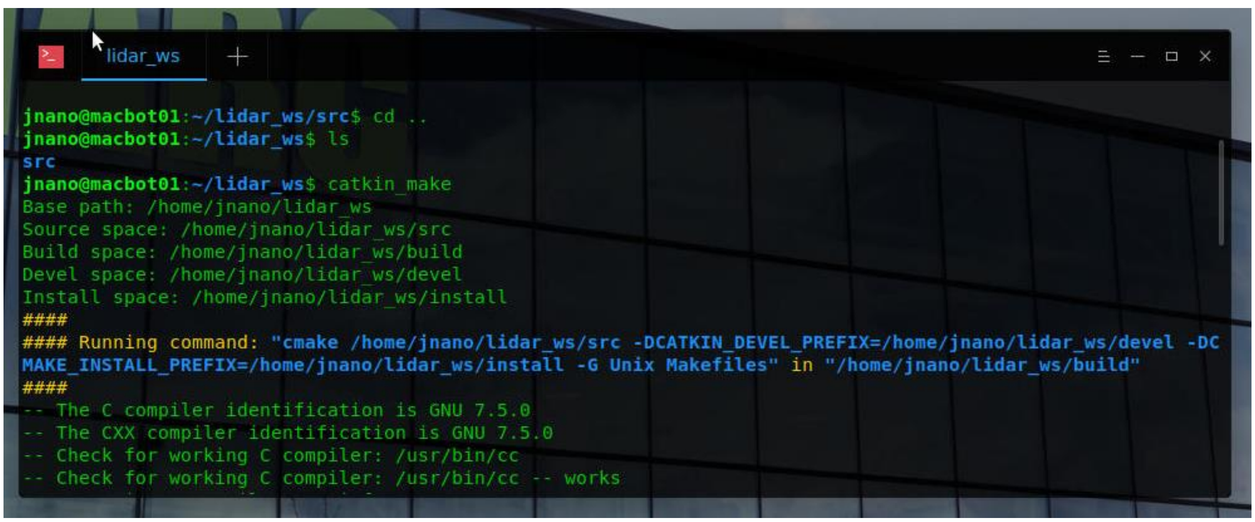
A new file called **CMakeLists.txt** should have been generated.



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

```
cd ~/lidar_ws
```

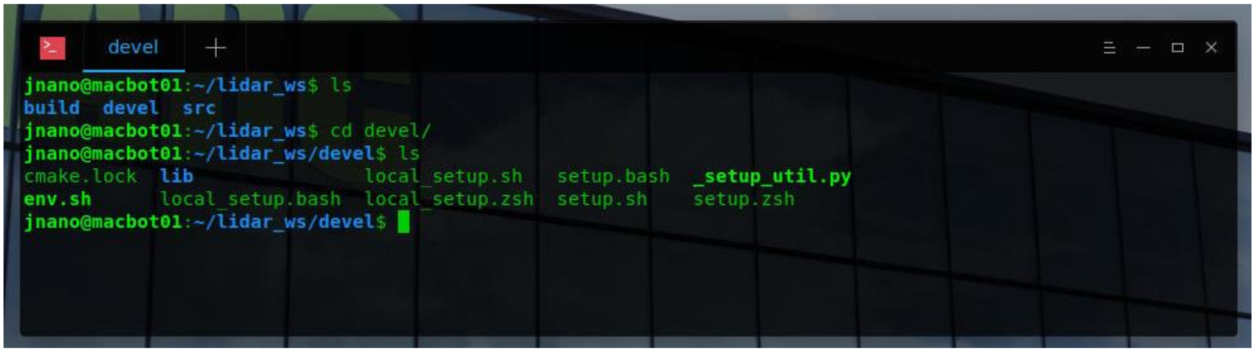
```
catkin_make
```



Notice that some new directories have been generated after the build is successful. 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.

```
ls
```

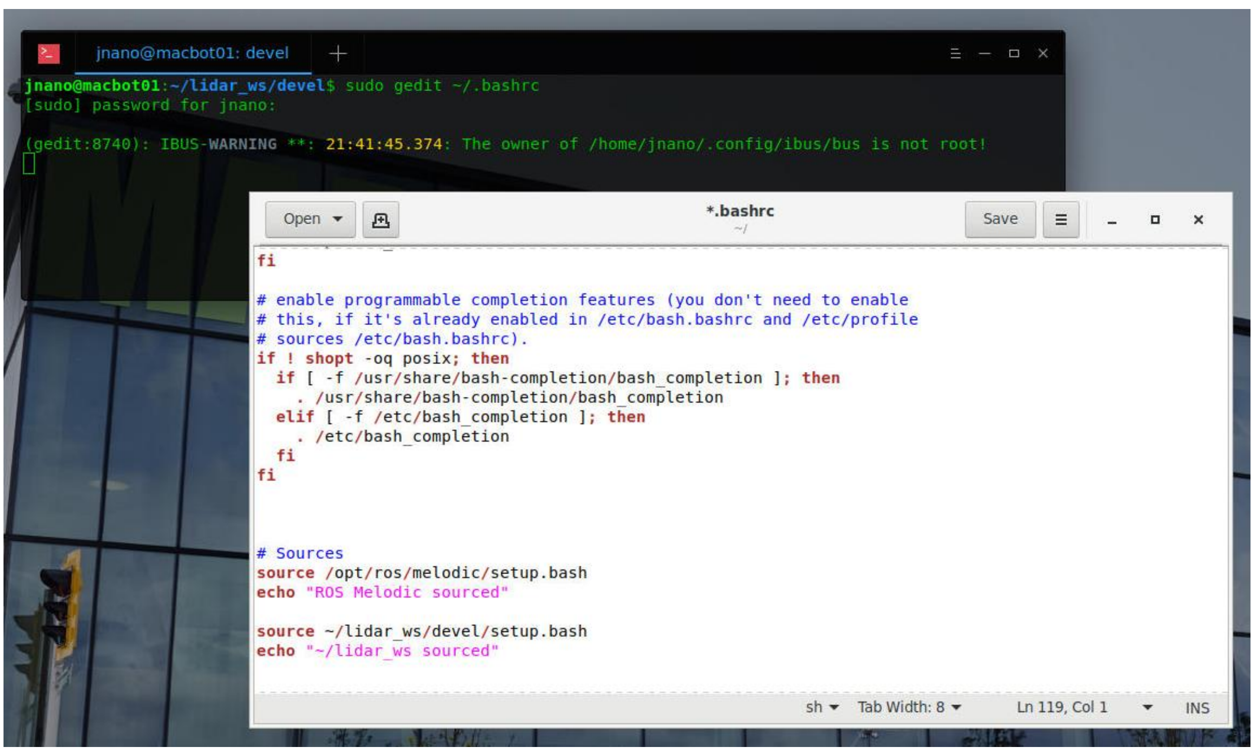
```
cd devel/
```



The .bashrc file is a script file that’s executed when a user logs in. The file itself contains a series of configurations for the terminal session. 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*> if needed



Notice the #Sources section that was previously added. Ensure that you have these two paths sourced and echoed to the terminal window. If you do not have them, copy-paste it.

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

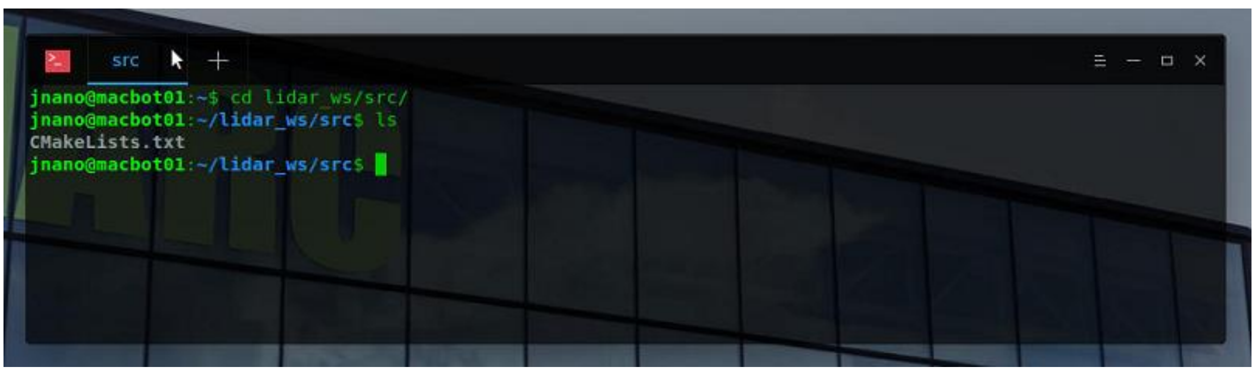
Save and close the file and Open a new teminal window.

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

Building the YDLiDAR ROS Driver

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

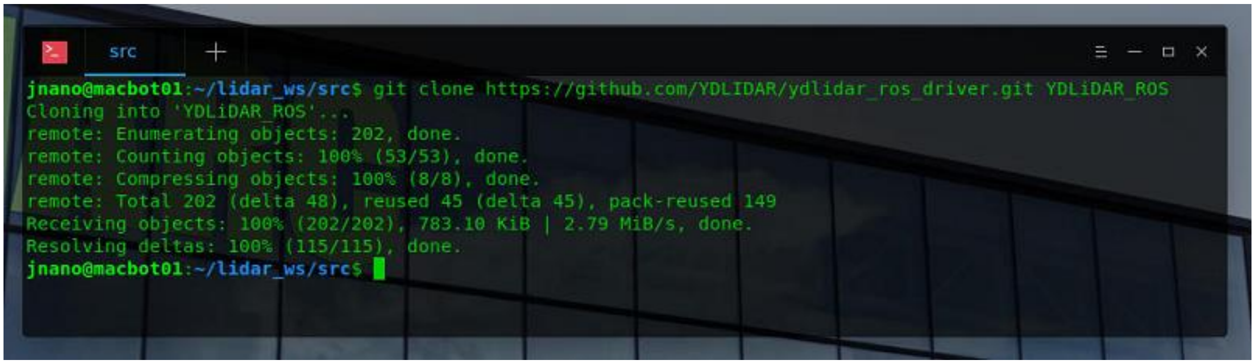
cd ~/lidar_ws/src



Clone the following YDLiDAR ROS driver into your workspace:

https://github.com/YDLIDAR/ydlidar_ros_driver

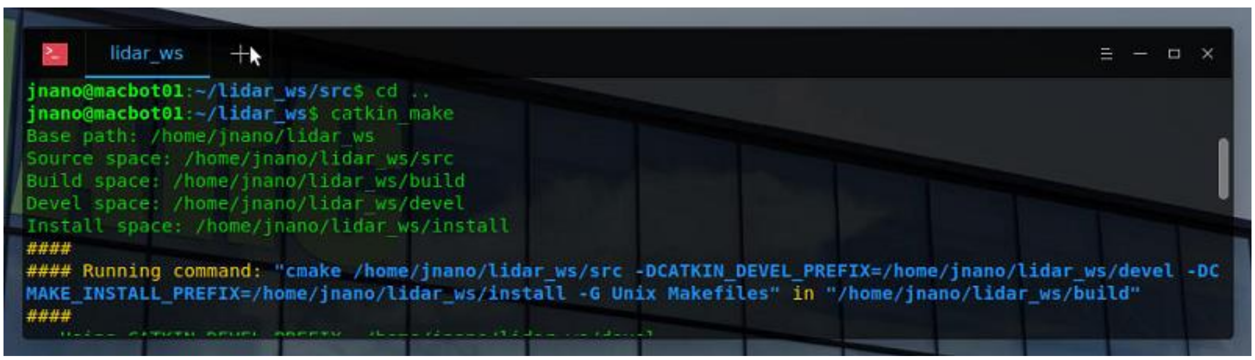
git clone https://github.com/YDLIDAR/ydlidar_ros_driver.git YDLiDAR_ROS



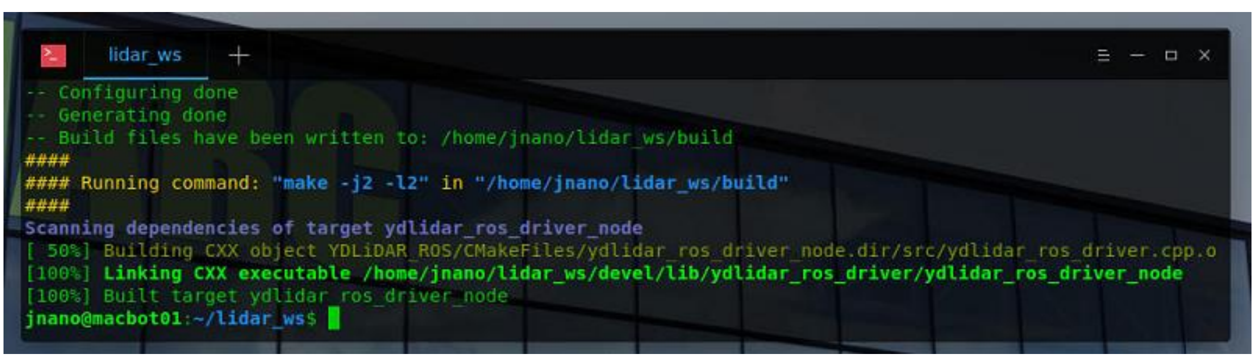
Build the workspace using **catkin_make**.

cd ..

catkin_make



Ensure that the build is successful.



Setting Permissions for the ROS LiDAR Driver

Navigate to your built LiDAR ROS package.

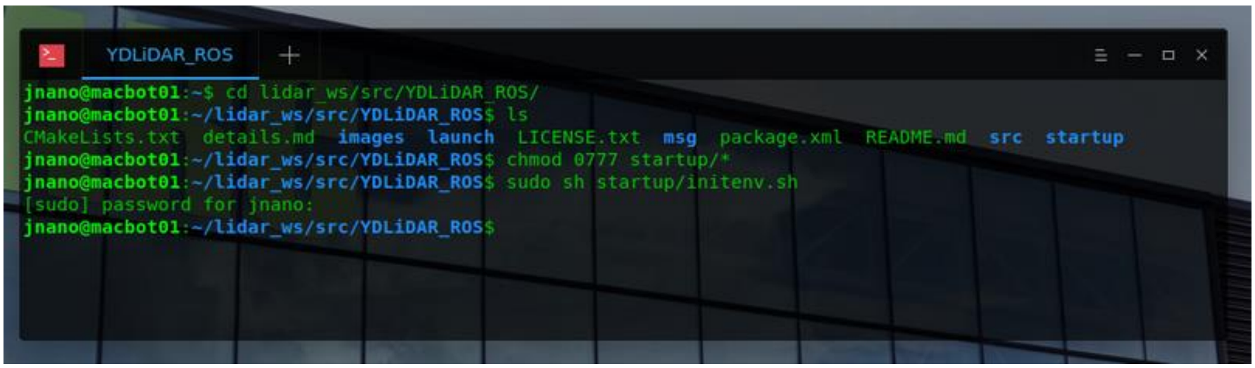
cd lidar_ws/src/YDLiDAR_ROS/

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

*chmod 0777 startup/**

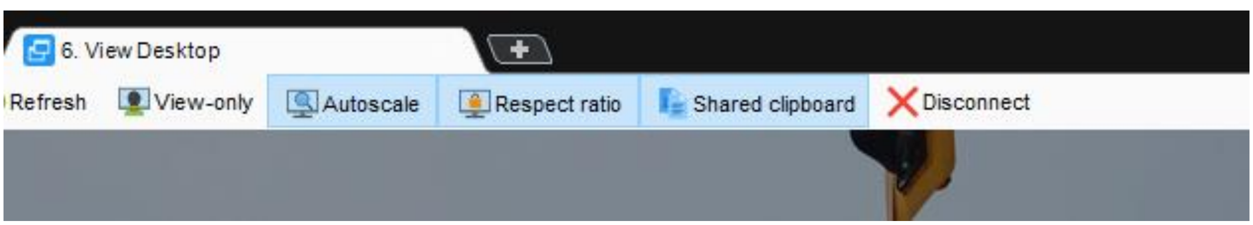
Run the environment initialization script inside of the startup/ directory. This script modifies the USB kernel device module to be able to communicate with the LiDAR. It then restarts the Linux device daemon/service.

sudo sh startup/initenv.sh



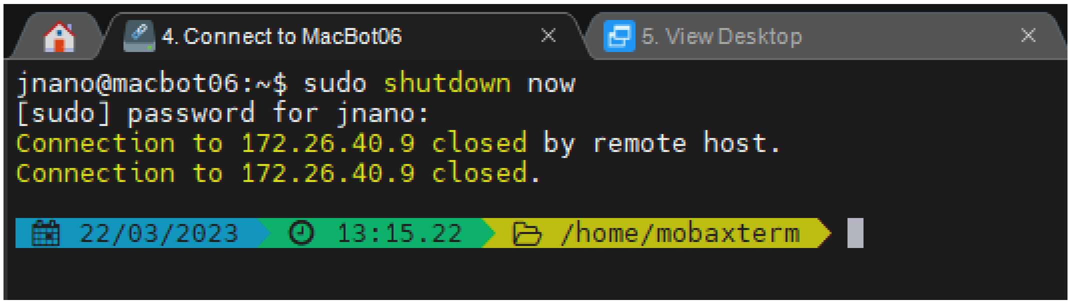
Now, we need to reboot the MacBot to ensure that the changes take effect.

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

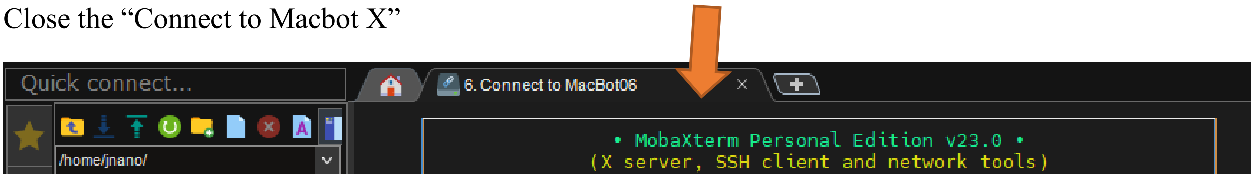


In the connection bash terminal, type the following command:

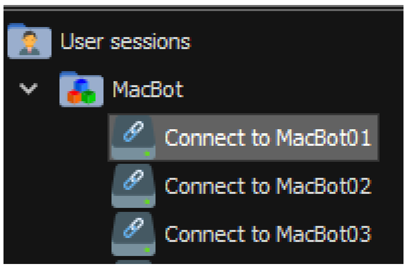
sudo shutdown now

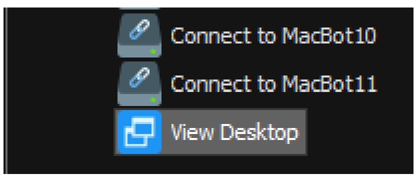


Close the “Connect to Macbot X”



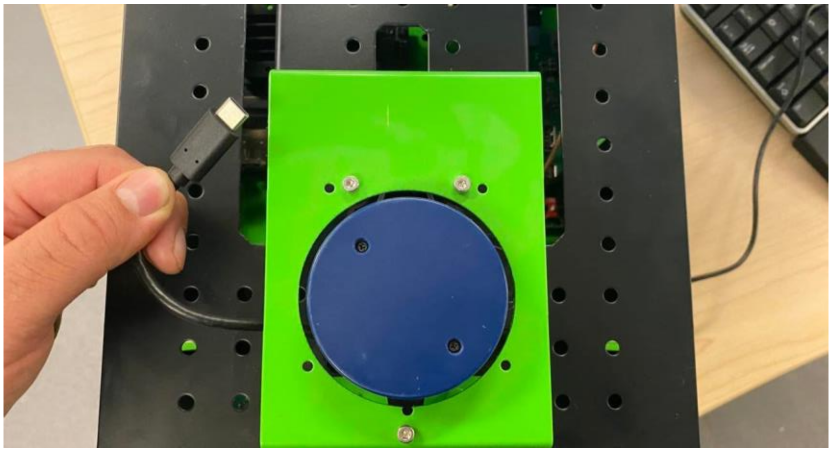
Wait 2 minutes before reconnecting to the MacBotX.





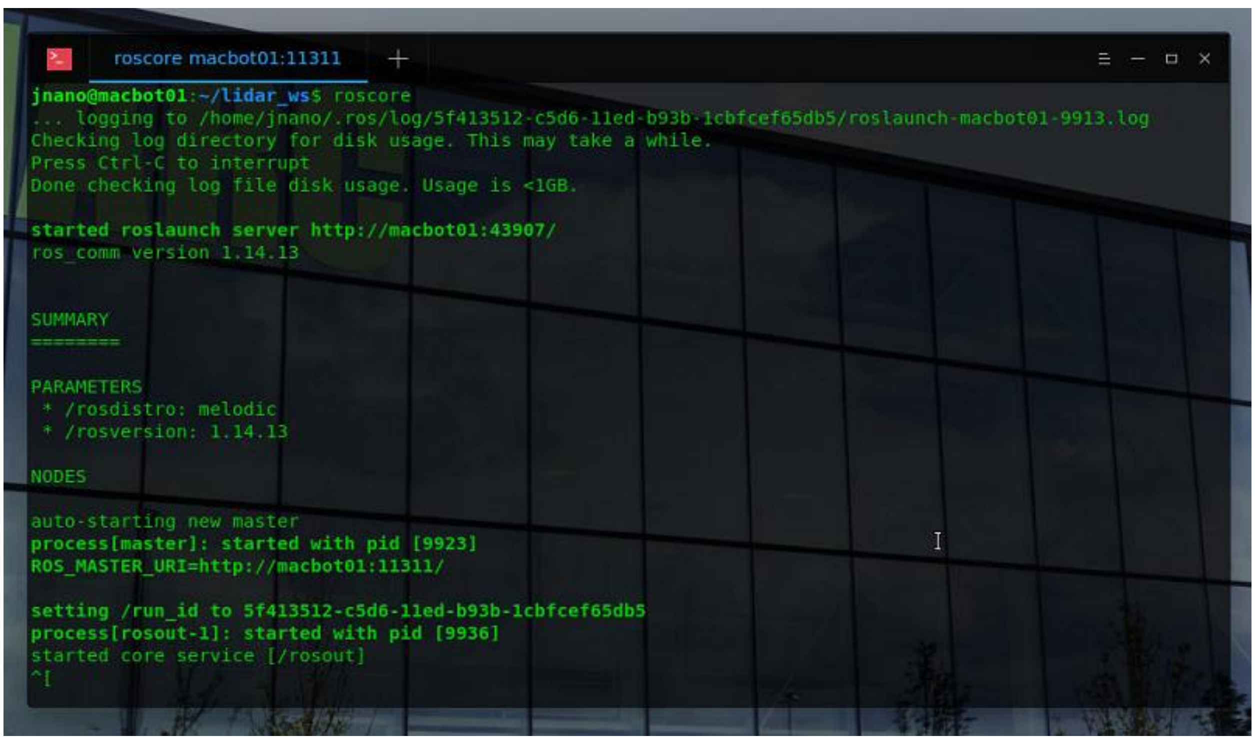
Visualizing LiDAR Point Cloud Data

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



Start ROSCore in a **new terminal window**.

roscore

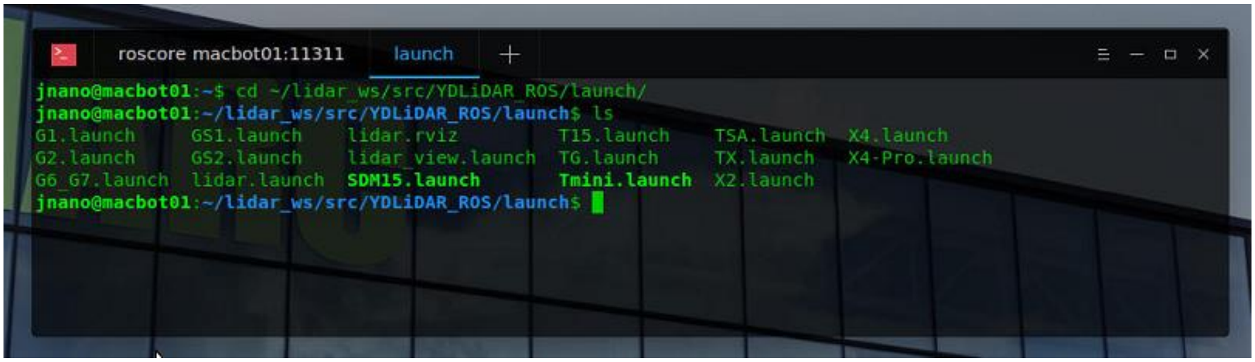


Open another terminal window (or tab, or split screen) and navigate to ~/lidar_ws/src/YDLiDAR/launch.

List out the different launch files available to run.

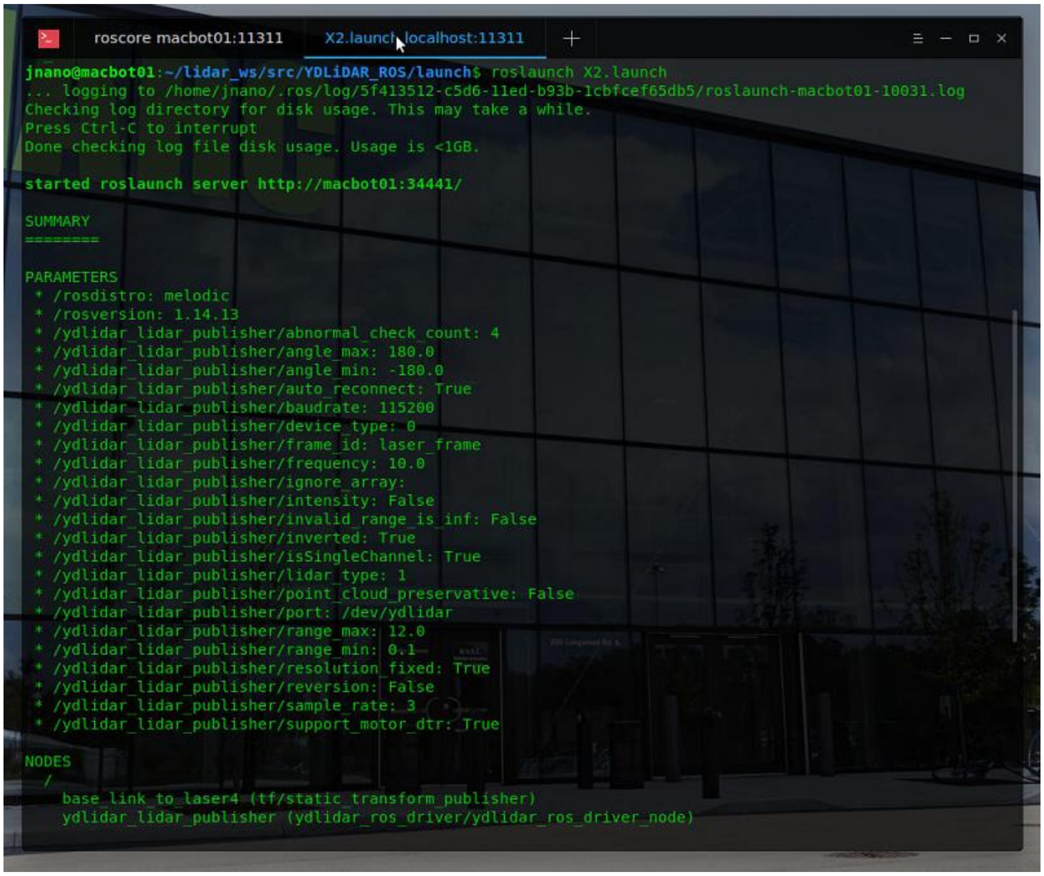
cd ~/lidar_ws/src/YDLiDAR_ROS/launch

ls



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.

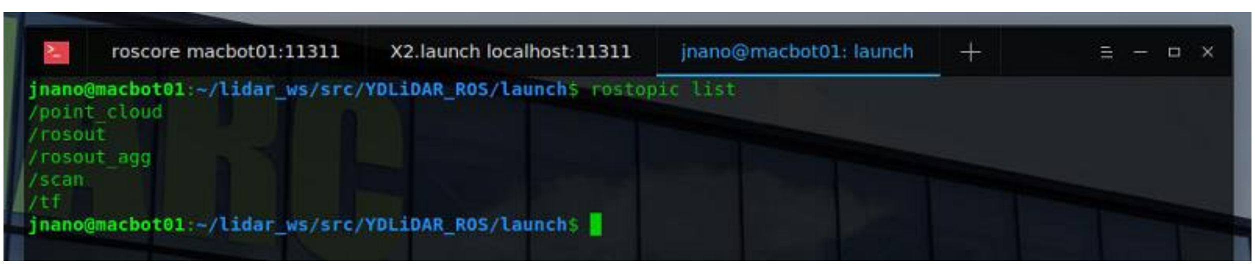
roslaunch X2.launch



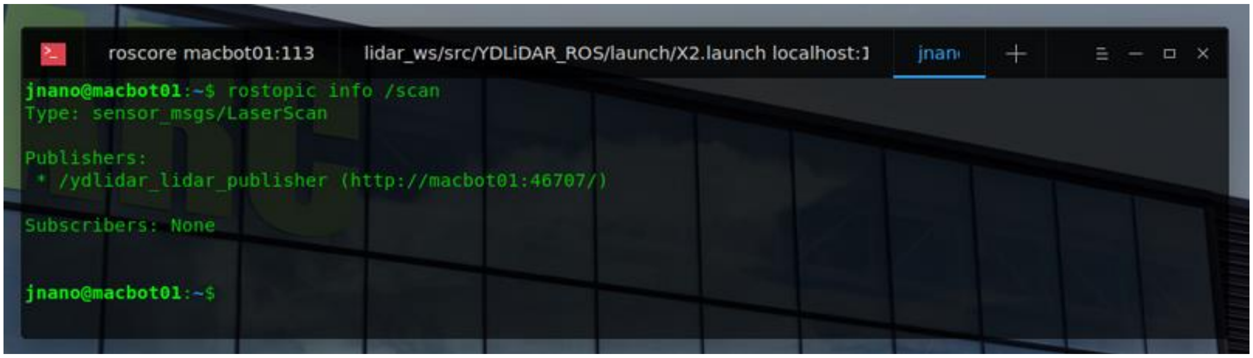
Next, open a new terminal window (or tab, or split-screen).

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

rostopic list

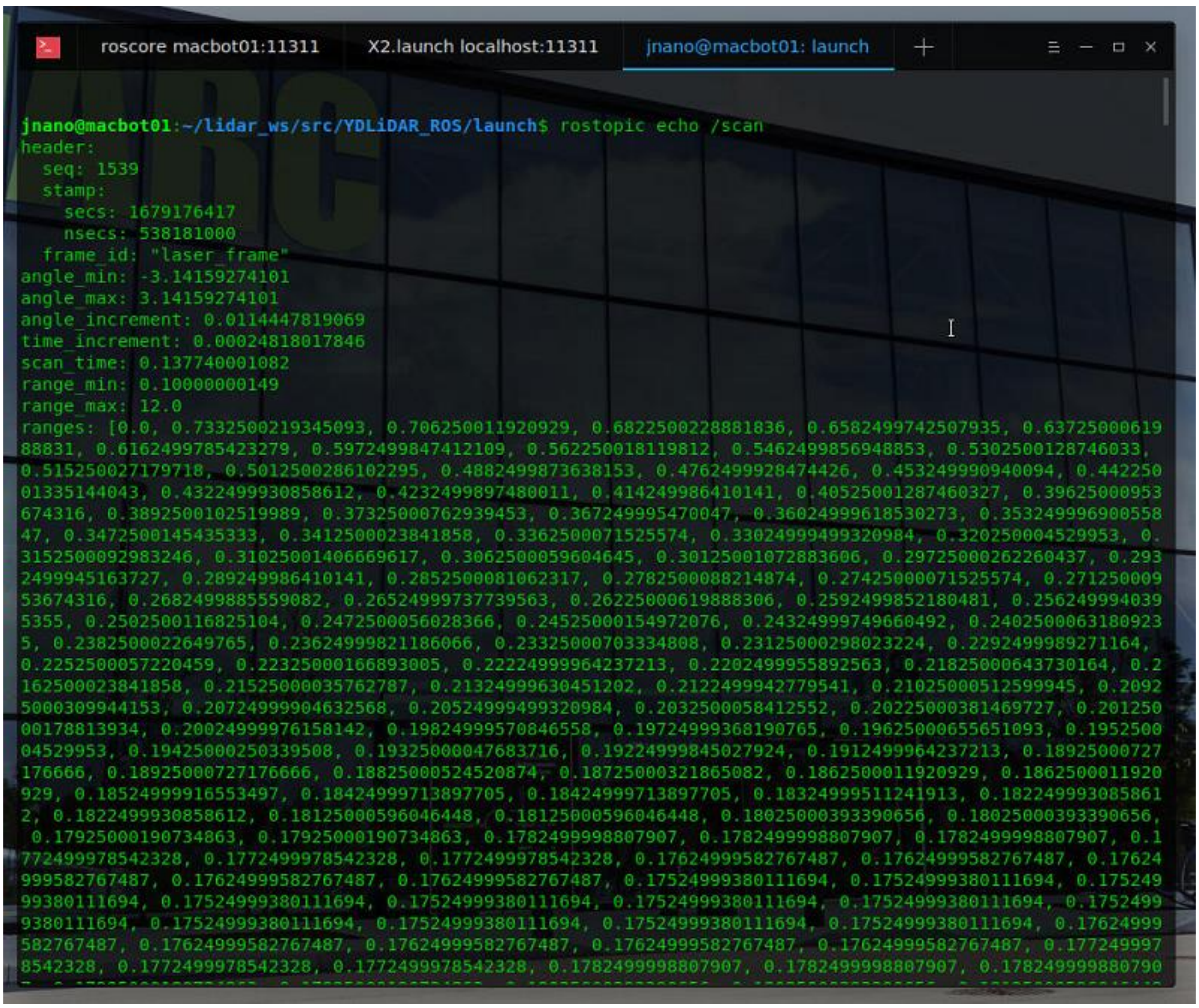


Find more information about the /scan topic including the datatype and port its hosted on.



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

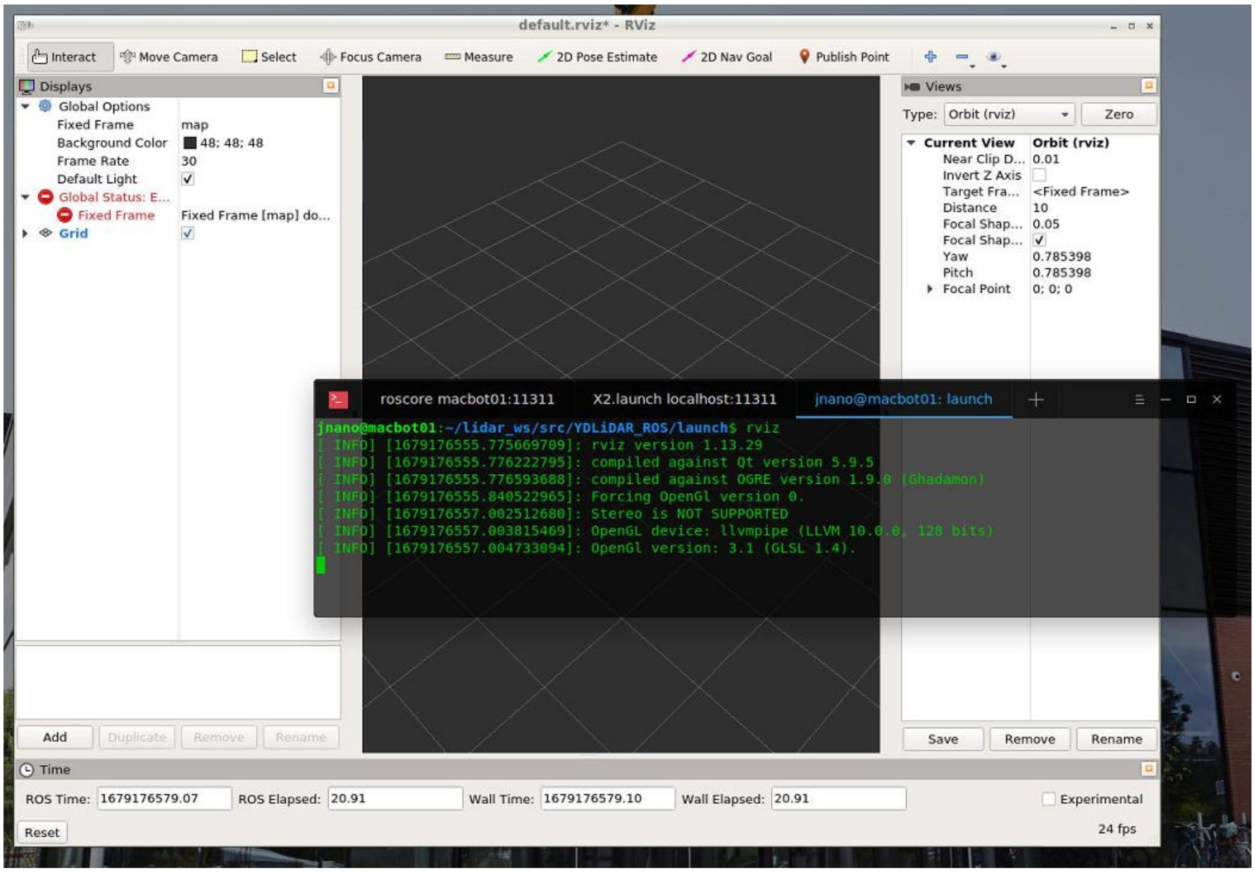
rostopic echo /scan



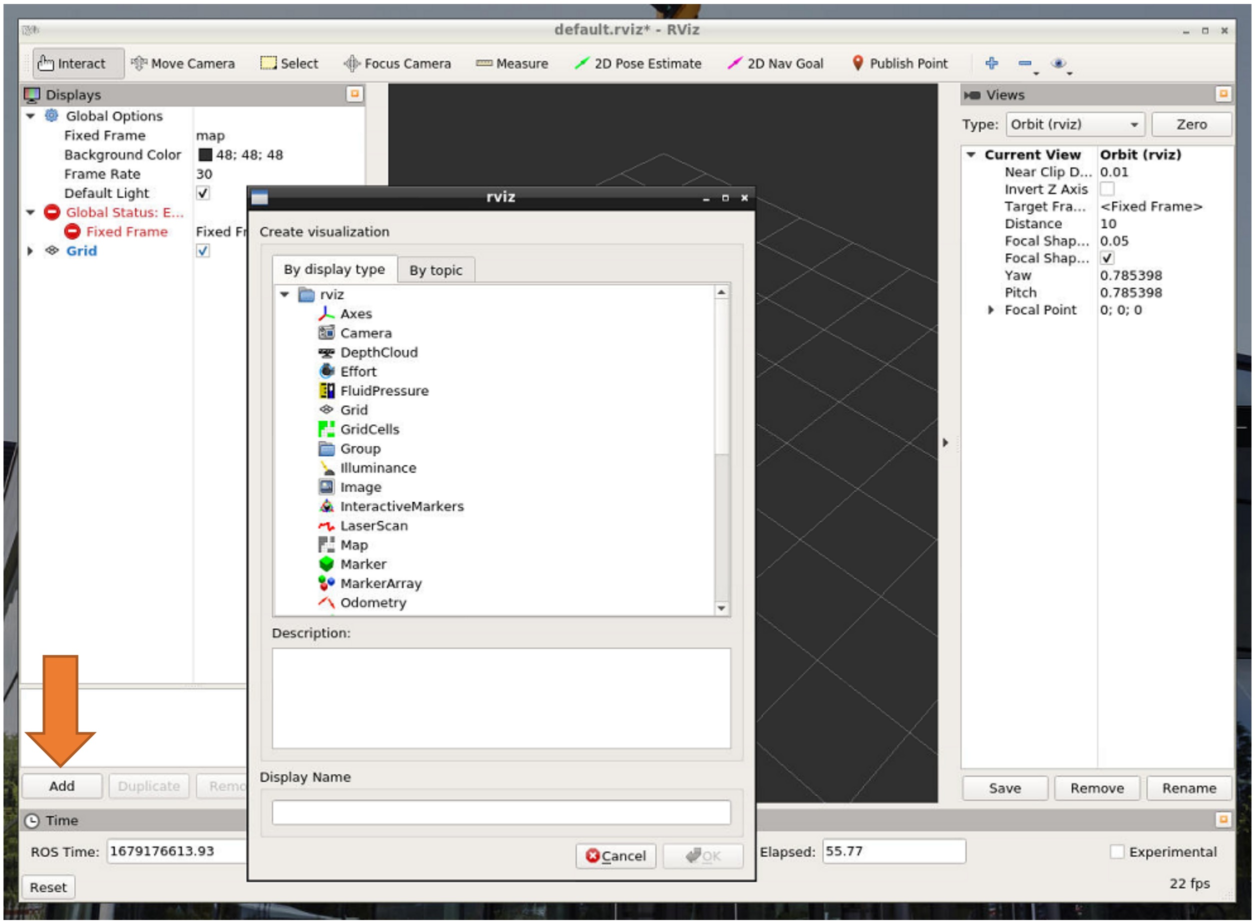
Press **CTRL + C** to stop viewing the data stream.

Next, open the ROS Visualization tool. A graphical utility will launch.

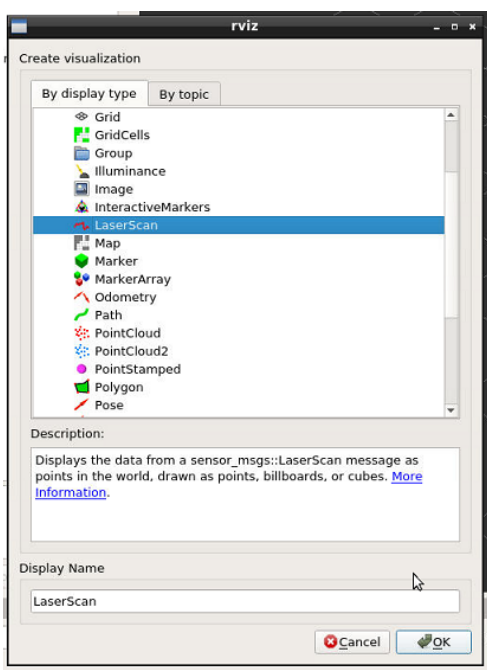
```
rviz
```



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



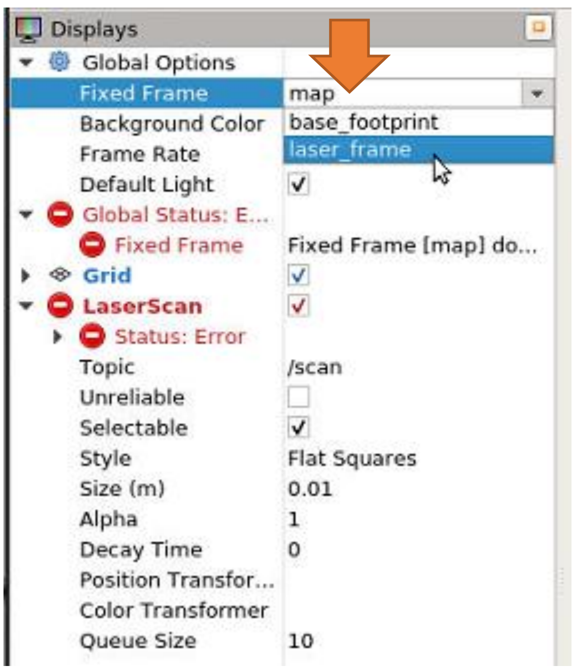
Choose laserscan and press OK.



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

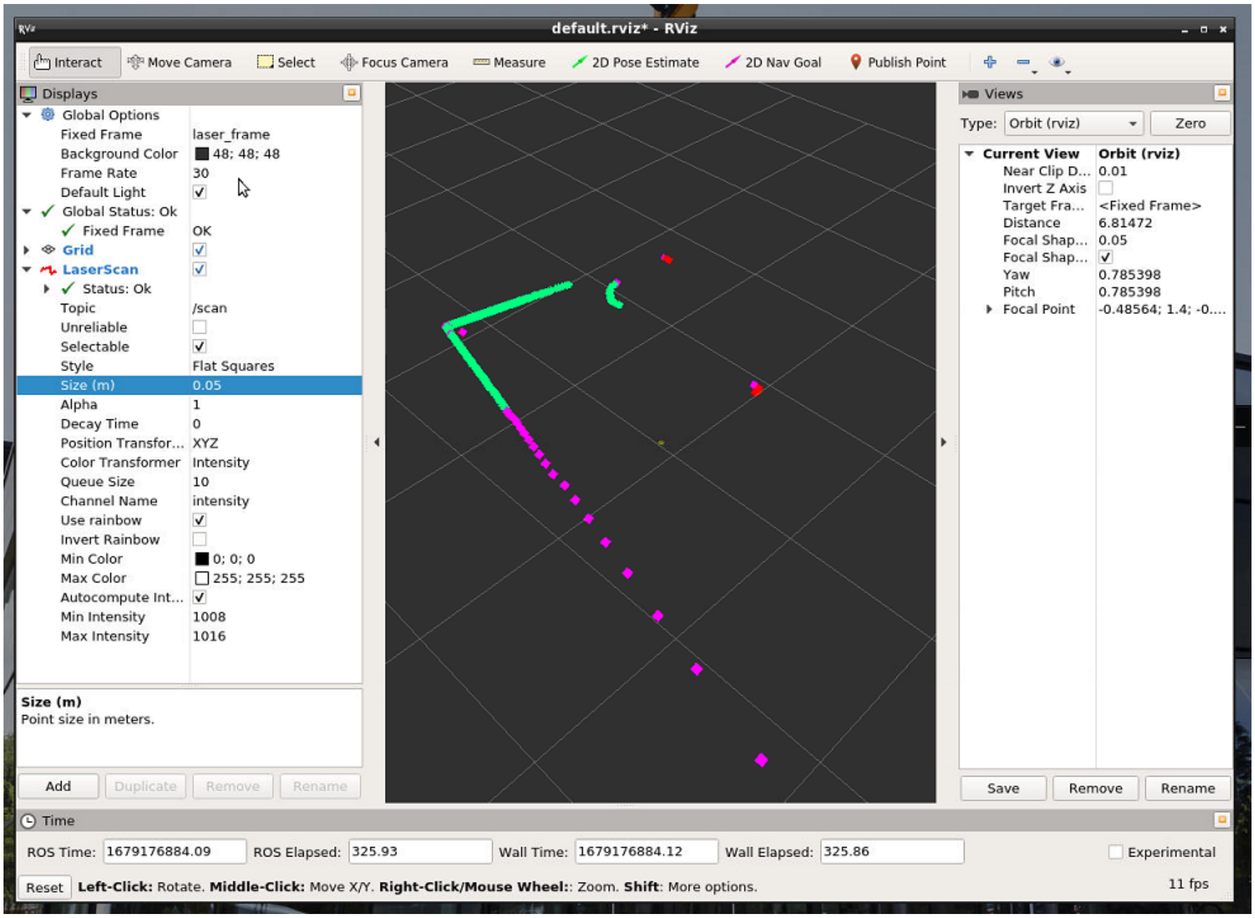
You will notice that RVIZ is in error state. This is because it does not have a reference point to plot the pointcloud data against.

Under **Display**, change the Fixed Frame to laser_frame.

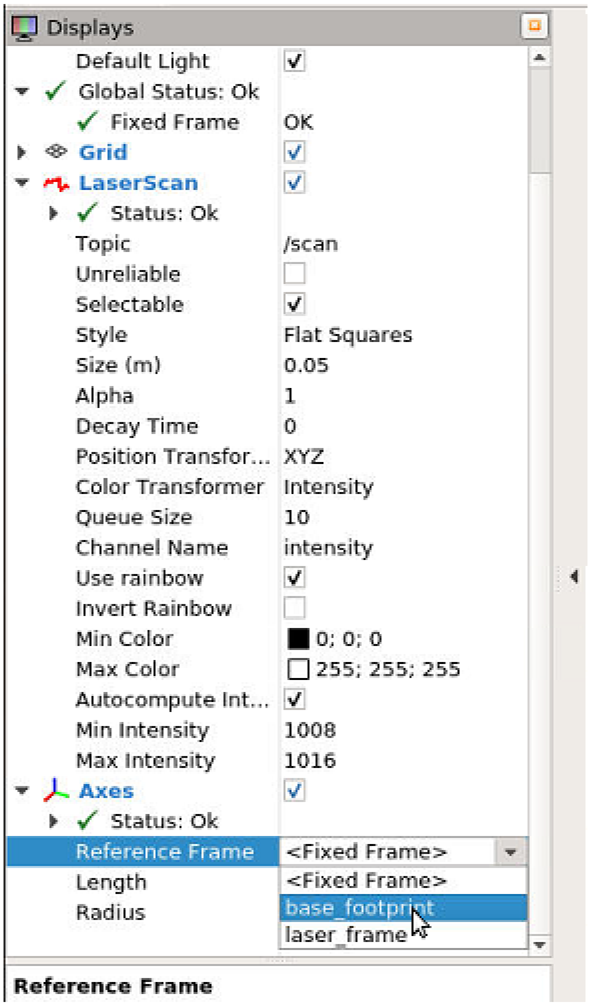


You should now observe data being visualized in RVIZ.

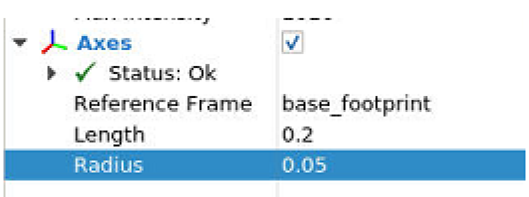
Modify Laser Scan > Size (m) to 0.05 to make the points a bit larger.

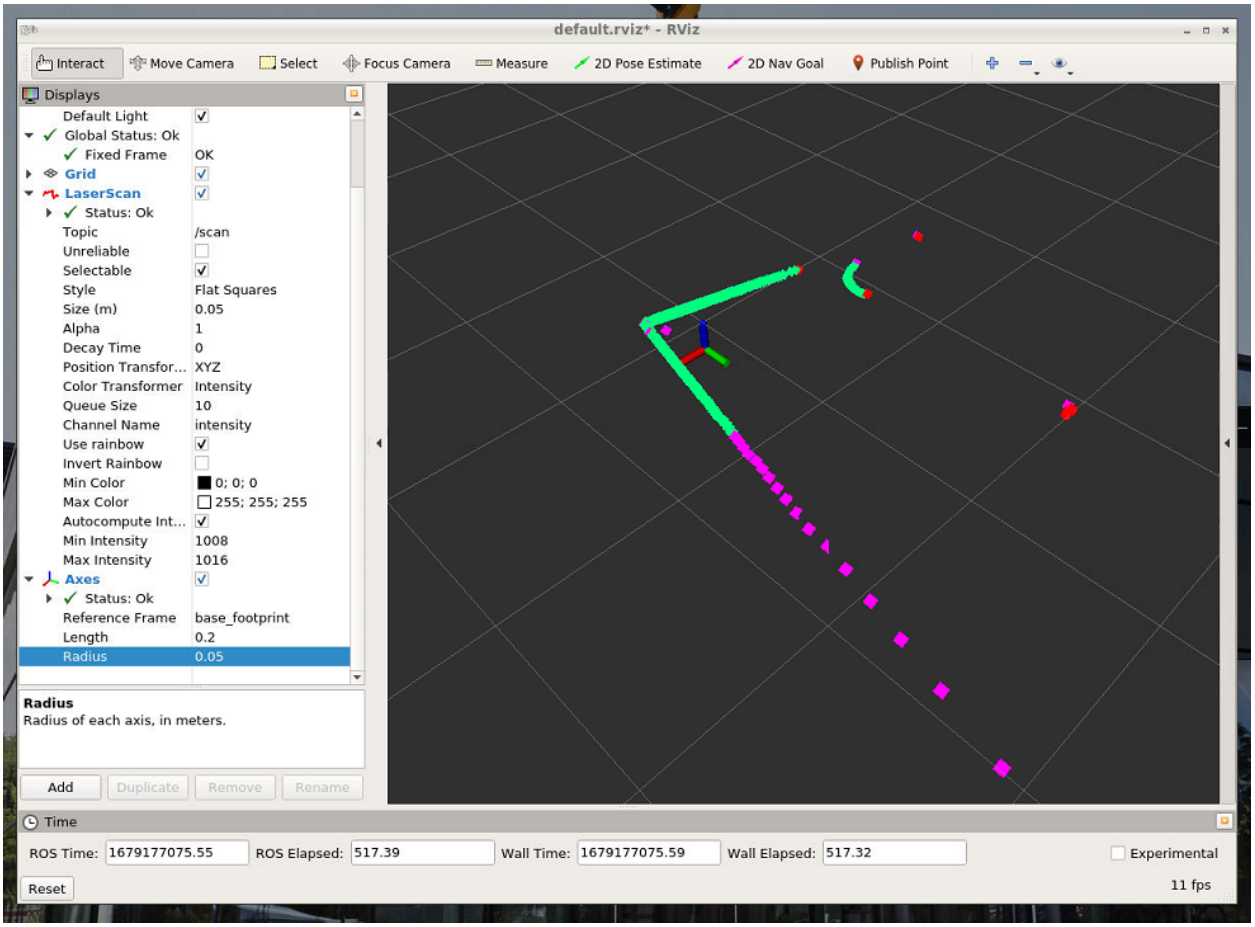


Lastly, lets input a marker for where the LiDAR is located. Insert an Axis display and set it to the base_footprint frame.



Modify the Length and Radius values.





Built-in display types:

Name	Description	Messages Used
Axes	Displays a set of Axes	
Effort	Shows the effort being put into each revolute joint of a robot.	sensor_msgs/JointStates
Camera	Creates a new rendering window from the perspective of a camera, and overlays the image on top of it.	sensor_msgs/Image, sensor_msgs/CameraInfo
Grid	Displays a 2D or 3D grid along a plane	
Grid Cells	Draws cells from a grid, usually	nav_msgs/GridCells

	obstacles from a costmap from the navigation stack.	
Image	Creates a new rendering window with an Image. Unlike the Camera display, this display does not use a CameraInfo. *Version: Diamondback+*	sensor_msgs/Image
InteractiveMarker	Displays 3D objects from one or multiple Interactive Marker servers and allows mouse interaction with them. *Version: Electric+*	visualization_msgs/InteractiveMarker
Laser Scan	Shows data from a laser scan, with different options for rendering modes, accumulation, etc.	sensor_msgs/LaserScan
Map	Displays a map on the ground plane.	nav_msgs/OccupancyGrid
Markers	Allows programmers to display arbitrary primitive shapes through a topic	visualization_msgs/Marker, visualization_msgs/MarkerArray
Path	Shows a path from the navigation stack.	nav_msgs/Path
Point	Draws a point as a small sphere.	geometry_msgs/PointStamped
Pose	Draws a pose as either an arrow or axes.	geometry_msgs/PoseStamped
Pose Array	Draws a "cloud" of arrows, one for each pose in a pose array	geometry_msgs/PoseArray
Point Cloud(2)	Shows data from a point cloud, with different options for	sensor_msgs/PointCloud, sensor_msgs/PointCloud2

	rendering modes, accumulation, etc.	
Polygon	Draws the outline of a polygon as lines.	geometry_msgs/Polygon
Odometry	Accumulates odometry poses from over time.	nav_msgs/Odometry
Range	Displays cones representing range measurements from sonar or IR range sensors. *Version: Electric+*	sensor_msgs/Range
RobotModel	Shows a visual representation of a robot in the correct pose (as defined by the current TF transforms).	
TF	Displays the tf transform hierarchy.	
Wrench	Draws a wrench as arrow (force) and arrow + circle (torque)	geometry_msgs/WrenchStamped
Twist	Draws a twist as arrow (linear) and arrow + circle (angular)	geometry_msgs/TwistStamped
Oculus	Renders the RViz scene to an Oculus headset	

Exercise A

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

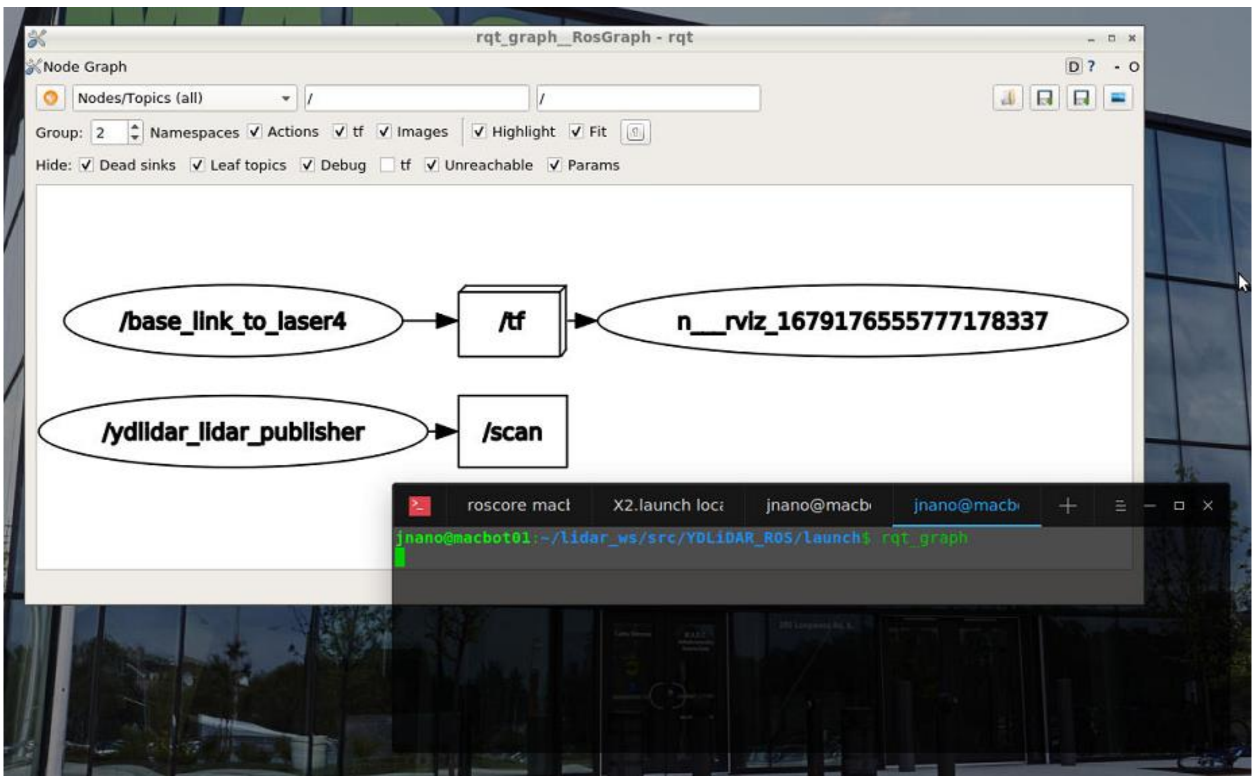
Generating a Diagram

Lastly, lets use the rqt_graph tool to generate a live-updated diagram of our ROS system.

In a new terminal window (or tab, or splt-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 submission.

Exercise C

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

(Suggested: 3 sentences)

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

(Suggested: 1 sentence)

Q4 - What is *Robot Operating System (ROS)* in your own words? Search for and list *3 applications* of ROS in industry.

(Suggested: Short paragraph)

Q5 - What role does *ROSCore* play in a functioning ROS system?

(Suggested: 1 sentence)

Q6 - Which command can be used to get *information* on a particular topic?

(Suggested: 1 sentence)

Q7 - What ROS tool can be used to *visualize* a stream of data?

(Suggested: 1 sentence)