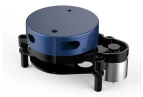
Application Note

YDlidar setup and documentation

Pratham Sharma



ABSTRACT

This project aims to design and implement a robust 360-degree LiDAR-based sensor fusion system for automated reverse parallel parking in autonomous vehicles. Leveraging a LiDAR sensor (or multiple sensors) positioned to provide panoramic environmental scanning around the vehicle platform, the system captures comprehensive spatial data for reliable obstacle detection in all directions. Improving the accuracy and reliability of environmental perception in parking scenarios is essential for fully autonomous parking maneuvers and overall vehicle safety.

Table of Contents

List of Tables

Overview (*From the documentation*)

The YDLIDAR X2 Lidar is a 360-degree two-dimensional distance measurement product (hereinafter referred to as X2) developed by the YDLIDAR team. This product is basedontheprinciple of triangulation, and is equipped with relevant optics, electricity, and algorithmdesign to realize high-frequency and high-precision distance measurement. At the sametimeas the distance measurement, 360 degrees of scanning distance measurement is achieved by continuously obtaining the angle information through the 360 degree rotation of themotor.

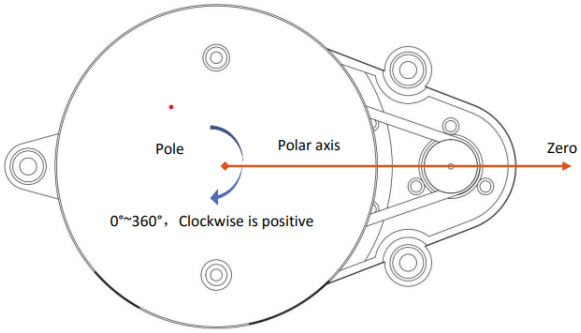


Figure 1: Caption

What is the need of lidar in reverse parallel parking?

LiDAR is essential in reverse parallel parking for autonomous vehicles because it provides highly accurate, real-time 360-degree environmental scanning. The key reasons for the need of LiDAR in this context are:

- **Precise Obstacle Detection:** LiDAR generates a high-resolution, panoramic point cloud that accurately detects the distance and shape of obstacles (such as other vehicles, curbs, and pedestrians) all around the vehicle, not just directly behind or beside it.
- **Reliability in Complex Scenarios:** LiDAR excels at identifying parking spaces and the boundaries of parking slots, even in scenarios where the space might be tight or the environmental conditions are less than ideal (such as nighttime or high glare).
- **Continuous Monitoring During Maneuver:** As the autonomous vehicle executes the reverse parallel parking maneuver, LiDAR allows the system to continuously track obstacles and ensure the vehicle follows a safe and optimal path, avoiding collisions—including during close-quarters adjustments.
- **Resilience Across Environments:** Compared to other sensors (like cameras or ultrasonic), LiDAR is less affected by lighting conditions and provides consistent detection capability in sunlight, darkness, and even in adverse weather.
- **Supports Odometry-Free Navigation:** Some strategies use LiDAR alone to localize the vehicle and execute the full parking maneuver without the need for wheel odometry or additional mapping, reducing system complexity while maintaining safety and robustness.

Overall, LiDAR enables autonomous vehicles to safely and accurately assess parking spaces, detect all surrounding obstacles, and perform complex maneuvers required for reverse parallel parking in a wide variety of real-world conditions.

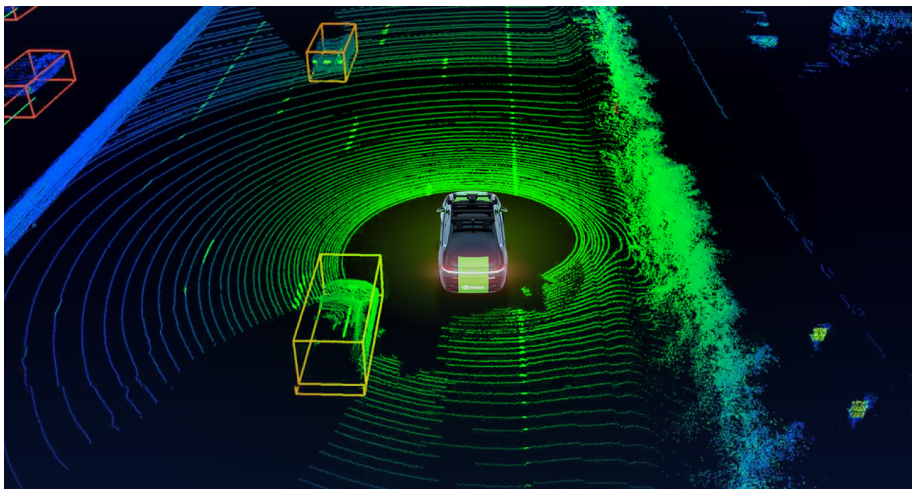


Figure 2: Caption

Requirements

- YDLidar X2
- Raspberry Pi 5
- Video Capture Device
- HDMI to microHMDI cable

YDLidar Specification

- 360-degree scanning distance measurement
- Small distance error; stable distance measurement and high accuracy
- Wide ranging range, not less than 8 m
- Strong resistance to ambient light interference
- Low power consumption, small size, stable performance and long life
- Laser power meets Class I laser safety standards
- Ranging frequency up to 3000 Hz (support customization)

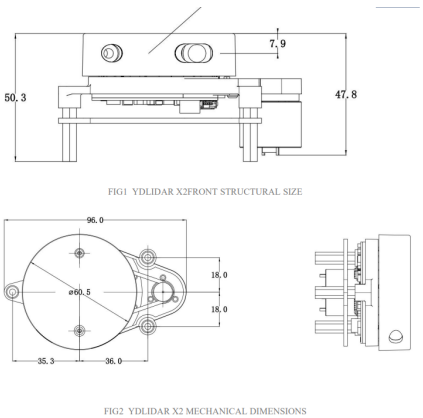


Figure 3: Caption

Product Parameter

CHART1 YDLIDAR X2 PRODUCT PARAMETER

Item	Min	Typical	Max	Unit	Remarks
Ranging frequency	-	3000	-	Hz	3000 times per second
Motor frequency	-	7	-	Hz	PWM or Voltage Regulation
Ranging distance	0.10	-	>8	m	Indoor
Scanning angle	-	0~360	-	Deg	-
Absolute error	-	2	-	cm	Distance≤0.5m
Relative error	-	1.5%	-	-	0.5m<Distance≤6m
	-	2.0%	-	-	6m<Distance≤8m
Angle resolution	0.82	0.84	0.86	Deg	Scanning frequency=7

Note 1: The measurement range and relative accuracy above are the factory inspection standard value;

Note 2: The relative error value indicates the accuracy of the Lidar measurement.

*Relative error = (Measuring distance - Actual distance) / Actual distance * 100%.*

Please avoid using Lidar under high-temperature, high-low temperature or strong vibration situation, which might cause a 3% relative error.

Figure 4: Caption

Polar coordinate system definition

In order to facilitate secondary development, X2 internally defines a polar coordinate system.

Pole: the center of the rotating core of the X2;

Positive direction: clockwise;

Zero angle: directly in front of the X2 motor;

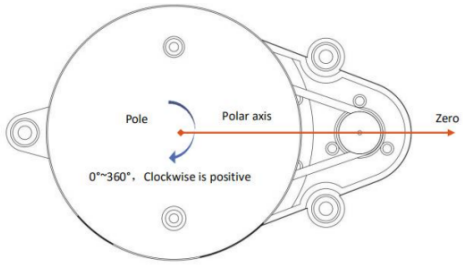


FIG4 YDLIDAR X2 POLAR COORDINATE SYSTEM DEFINITION

Others

CHART7 YDLIDAR X2 OTHERS

Item	Min	Typical	Max	Unit	Remarks
Operating temperature	0	20	50	℃	High temperature environment will reduce life expectancy
Lighting environment	0	550	2000	Lux	For reference only
weight	-	200	-	g	N.W.

Figure 5: Caption

Electrical Parameter

CHART2 YDLIDAR X2 ELECTRICAL PARAMETER

Item	Min	Typical	Max	Unit	Remarks
Supply voltage	4.8	5	5.2	V	Excessive voltage might damage the Lidar while low affect normal performance
Voltage ripple	0	50	100	mV	Excessive ripple affect normal performance
Starting current	300	400	500	mA	Higher current required at start-up
Working current	200	350	380	mA	Normal working

Interface Definition

X2 provides a PH1.25-4P female connector with functional interfaces for system power, data communication and motor control.

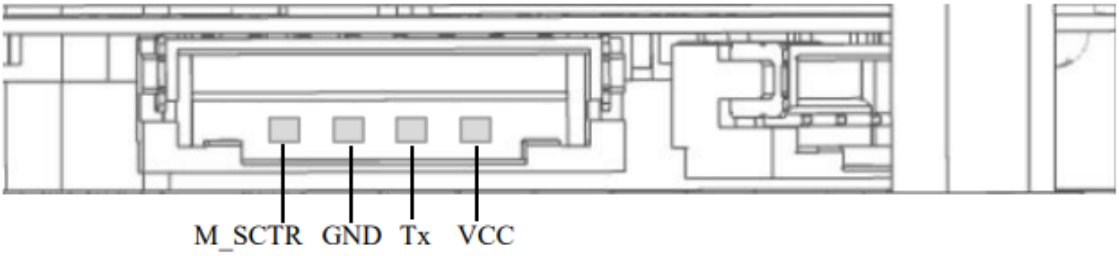


FIG3 YDLIDAR X2 INTERFACES

CHART3 YDLIDAR X2 INTERFACE DEFINITION

Pin	Type	Description	Defults	Range	Remarks
VCC	Power Supply	Positive	5V	4.8V~5.2V	-
Tx	Output	System serial output	-	-	Data stream: Lidar➔Peripherals
GND	Power Supply	negative	0V	0V	-
M_SCTR	Input	Motor speed control terminal	1.8V	0V~3.3V	Voltage or PWM speed regulation

Figure 6: Caption

Data communication

With a 3.3V level serial port (UART), users can connect the external system and the product through the physical interface. After that, you can obtain the real-time scanned point cloud data, device information as well as device status. The communication protocol of parameters are as follows:

CHART4 YDLIDAR X2 SERIAL SPECIFICATION

Item	Min	Typical	Max	Unit	Remarks
Baud rate	-	115200	-	bps	8-bit data bit,1 stop bit,no parity
High Signal Level	1.8	3.3	3.5	V	Signal voltage>1.8V
Low signal Level	0	0	0.5	V	Signal voltage<0.5V

Motor control

X2's motor driver supports speed control function and can be adjusted by the M_SCTR pin.

The lower the voltage / the smaller the PWM duty cycle, the higher the motor speed.

For example:

The maximum speed is 0V/duty cycle 0% means when the M_SCTP input 0V voltage, the motor rotates at the highest speed.

Following is the PWM signal requirements of M_SCTP:

CHART5 YDLIDAR X2 MOTOR PWM SIGNAL SPECIFICATION

Item	Min	Typical	Max	Unit	Remarks
PWM Frequency	-	10	-	KHz	PWM is the wave signal
Duty cycle range	50%	85%	100%		The smaller the duty cycle, the faster the speed

Optical Characteristic

X2 uses an infrared point pulsed laser that meets FDA Class I laser safety standards. The laser and optical lens are used for the transmission and reception of the laser signal to achieve high- frequency ranging. To ensure performance, please keep the laser and optical lens clean. The detailed optical parameters are as follows:

Figure 7: Caption

d

Ubuntu 24.04 Installation in Raspberry Pi 5

We are using Raspberry Pi 5 with 8 GB RAM. In addition, we have installed an active cooler.

How to Install Ubuntu 24.04 LTS on Raspberry Pi Through Raspberry Pi Imager

You can also download and install the **Raspberry Pi Imager** tool's latest version on your system to directly load Ubuntu 24.04 on the SD Card. The advantage of using the **Raspberry Pi Imager** tool is that you will not be required to download the Ubuntu 24.04 image file. To install Ubuntu 24.04 on a Raspberry Pi device using **Raspberry Pi Imager**, follow the below-given steps:

Step 1: Download Raspberry Pi Imager

First of all, navigate to the Raspberry Pi website, download the **Raspberry Pi Imager** stable release on the system:

Step 2: Create a Raspberry Pi Image on an SD Card

- Now, first, insert your SD Card into your computer and open **Raspberry Pi Imager** on your desktop:

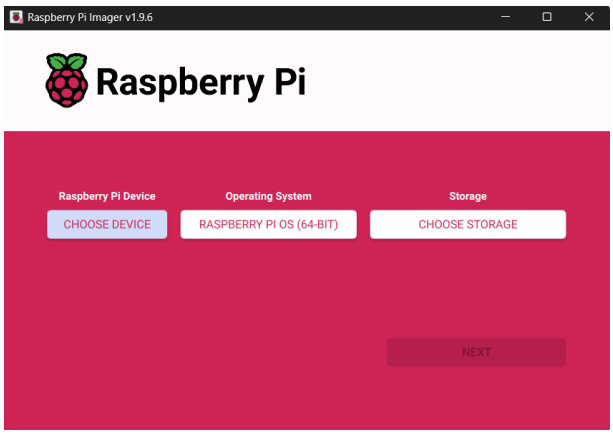


Figure 8: Caption

- Then select the Raspberry Pi device model using the **CHOOSE DEVICE** option:

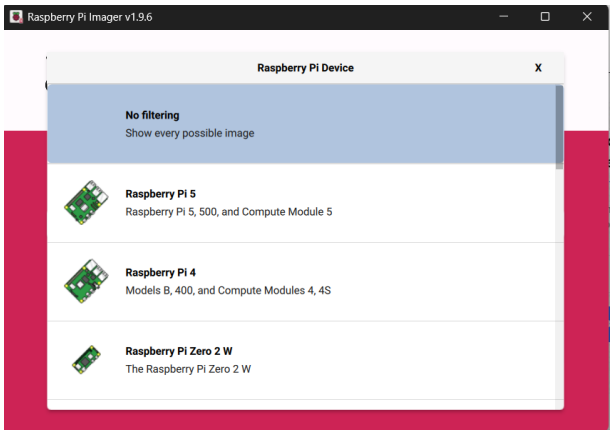


Figure 9: Caption

- Then select **CHOOSE OS**:
- Navigate towards the **Other-general-purpose OS:**
- Select **Ubuntu**:

- Choose **Ubuntu Desktop 24.04 LTS (64-bit)**:

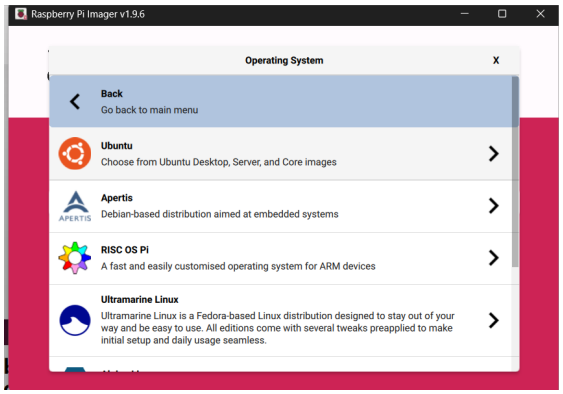


Figure 10: Caption

- Then click the **CHOOSE STORAGE** option later on:
- Select USB Storage option:
- Then click the **Next** button once everything is done according to the above steps:
- Finally, click the **YES** option to allow Raspberry Pi Imager to format your SD Card:

After this phase, the **Raspberry Pi Imager** will start writing the image of Ubuntu 24.04 onto the SD Card: Once the Flash process is done, simply remove the SD Card and insert it into the Raspberry Pi's SD Card port. Power on your Raspberry Pi device and wait for a few seconds until the device loads the Ubuntu 24.04 system.

Note: Ensure you have set up a complete Raspberry Pi desktop with a display monitor, mouse, and keyboard.

After a while, you will be able to see the Ubuntu 24.04 system configuration, simply complete the steps which include, keyboard layout, connecting to a Wi-Fi network, selecting the location, and adding your computer details. It include the name of computer, username, as well as password. You have to do this on your own and once it is done, you will be able to use the Ubuntu 24.04 desktop on Raspberry Pi:

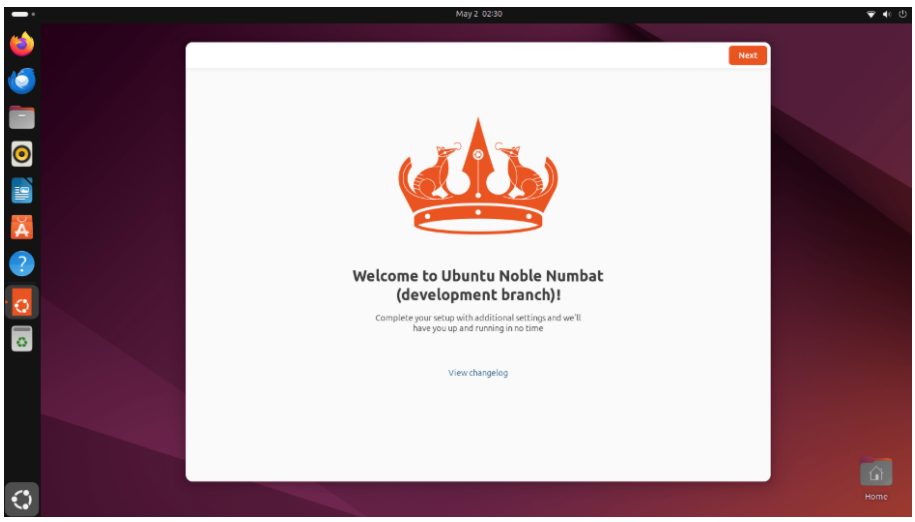


Figure 11: Caption

ROS2 Jazzy installation in Ubuntu 24.04

Set locale

Make sure you have a locale which supports `UTF-8`. If you are in a minimal environment (such as a docker container), the locale may be something minimal like `POSIX`. We test with the following settings. However, it should be fine if you're using a different UTF-8 supported locale.

```
locale # check for UTF-8

sudo apt update && sudo apt install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8

locale # verify settings
```

Enable required repositories

You will need to add the ROS 2 apt repository to your system. First ensure that the Ubuntu Universe repository is enabled.

```
sudo apt install software-properties-common
sudo add-apt-repository universe
```

Now add the ROS 2 GPG key with apt.

```
sudo apt update && sudo apt install curl -y
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg
```

Then add the repository to your sources list.

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
```

Install development tools (optional)

If you are going to build ROS packages or otherwise do development, you can also install the development tools:

```
sudo apt update && sudo apt install ros-dev-tools
```

Install ROS 2

Update your apt repository caches after setting up the repositories.

```
sudo apt update
```

ROS 2 packages are built on frequently updated Ubuntu systems. It is always recommended that you ensure your system is up to date before installing new packages.

```
sudo apt upgrade
```

Desktop Install (Recommended)

ROS, RViz, demos, tutorials.

```
sudo apt install ros-jazzy-desktop
```

ROS-Base Install (Bare Bones)

Communication libraries, message packages, command line tools. No GUI tools.

```
sudo apt install ros-jazzy-ros-base
```

Install additional RMW implementations (optional)

The default middleware that ROS 2 uses is `Fast DDS`, but the middleware (RMW) can be replaced at runtime. See the guide on how to work with multiple RMWs.

Setup environment

Set up your environment by sourcing the following file:

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

Note: Replace `.bash` with your shell if you're not using bash. Possible values are: `setup.bash`, `setup.sh`, `setup.zsh`.

Automatic setup (recommended)

To make it permanent (auto-load on terminal start):

If you are using **bash** (default in Ubuntu), add the following line to the end of your `~/.bashrc` file:

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

Reload your shell for the changes to take effect:

```
source ~/.bashrc
```

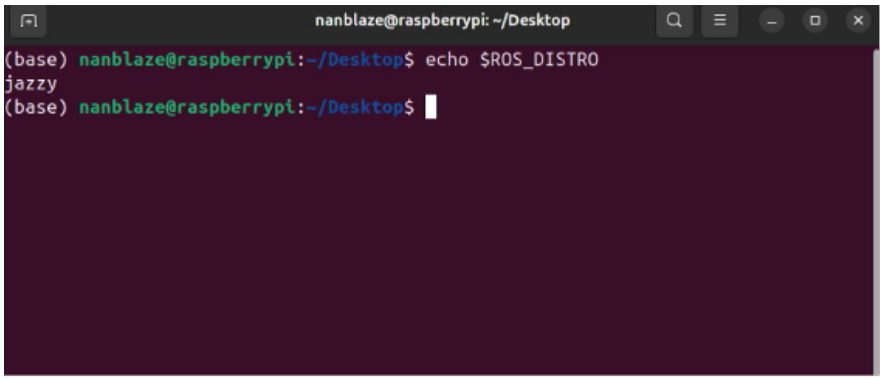


Figure 12: Caption

1 Download YDLidar SDK

ROS2 SLAM Toolbox and Odometry for 2D LiDAR + IMU

SLAM Toolbox Odometry Requirements

SLAM Toolbox requires some form of odometry (from wheels, scan matcher, or an external source) for real-time map updates in ROS2. It does **not** natively generate odometry from 2D LiDAR + IMU alone. This limitation is confirmed by both the repository maintainers and user reports.

- Community recommendation: plug in a scan matcher node or use external LiDAR odometry.
- Most current ROS2 packages either require odometry or focus on 3D LiDAR, not pure 2D LiDAR + IMU without odometry.

Confirmed Limitations

- SLAM Toolbox does not generate odometry itself.
- Expects an external odometry source: wheels, scan matcher, visual, etc.
- Without odometry, the map fails to update or updates sporadically.
- IMU support improves rotational accuracy (yaw) but cannot substitute proper odometry.

Known Workarounds or Alternatives

- Integrate a separate ROS2 scan matcher package to produce odometry messages from laser scans (many scan matcher nodes are outdated or ROS1-based).
- Leverage IMU + LiDAR fusion algorithms such as LIO-SAM or LINS; most implementations are 3D-focused or require modification for pure 2D.
- Use scan matcher outputs as simulated odometry (confirmed to work for 2D LiDAR + IMU without odometry).

Experimental ROS2 Packages

- MOLA claims ROS2 support for LiDAR odometry and provides pipelines for 2D LiDARs without odometry. Manual configuration may be required.

References

- SLAM Toolbox Issue #221: “Must provide odometry from wheels, scan matcher, or other external node; not in scope of the SLAM library itself.”
- User reports: “Map won’t update in real time without odometry, despite valid LiDAR and TF chain.”
- MOLA-LO: Open-source LiDAR odometry (ROS2) with dedicated support for 2D LiDAR workflows.

Importance of Odometry for Mapping

Odometry is like a robot’s way of keeping track of how it moves from one place to another. It uses data from sensors that measure motion, such as wheel rotations or movement sensors, to estimate distance traveled and rotations.

Why Odometry Matters

Imagine drawing a map while walking blindfolded. To know where to put each part of the map, you need to estimate where you have stepped. Odometry tells the robot: "you moved this far forward and turned this much," so it knows where to place new observations.

Simple Analogy

Counting steps and turns while walking in a dark room to estimate your position. Odometry acts as the robot's "step counter" for real-time navigation and mapping.

Summary

- Odometry enables continuous position updates, which are essential for building accurate maps.
- Without odometry, the robot cannot determine its position relative to previous locations, making mapping from sensor data alone difficult.
- Odometry provides a rough position estimate, refined later by sensor data like laser scans or camera images.

Using Hall Effect Sensors for Odometry

Hall effect sensors can be used as speed sensors for odometry in robotics.

How Hall Effect Sensors Work

- Output pulses when a magnet passes by the sensor.
- Counting pulses calculates wheel rotations and angular displacement.
- Combined with wheel size and gear ratio, distance traveled can be calculated.

Advantages for Odometry

- Durable and contactless, reducing wear compared to mechanical switches.
- Provide precise and reliable signals for estimating robot movement.

A Hall effect sensor acts as a rotary encoder, producing data necessary for odometry. This helps robots

Decoding Occupancy Grid Mapping

LINK