**LIDAR 360 to Raspberry pi 4**

**🛠️ Step 1: Install Dependencies on Raspberry Pi**

Since you are using **Raspberry Pi 4**, we need to install the necessary dependencies to communicate with **RPLIDAR A1M8** and process its data.

**🔹 1️. Update Raspberry Pi Packages**

First, update your Raspberry Pi’s package list and upgrade existing packages:

bash

CopyEdit

sudo apt update && sudo apt upgrade -y

**🔹 2️. Install Python & Required Libraries**

Your code will run using **Python 3**, so install necessary Python libraries:

bash

CopyEdit

sudo apt install python3-pip python3-numpy python3-opencv -y

Additionally, install libraries required to interface with **RPLIDAR**:

bash

CopyEdit

pip3 install rplidar-roboticia matplotlib

**🔹 3️. Check USB Connection for LiDAR**

If you are using **USB to connect RPLIDAR to Raspberry Pi**, check which port it is connected to:

bash

CopyEdit

ls /dev | grep ttyUSB

This will output something like:

nginx

CopyEdit

ttyUSB0

This means your LiDAR is connected at /dev/ttyUSB0.  
  
chat gpt :

Your **Raspberry Pi OS** has system-wide package management protection (PEP 668). This prevents **pip** from installing packages directly into the system Python environment.

To fix this, use **one of these methods**:

**🔹 Solution 1: Use a Virtual Environment (Recommended)**

1️. **Create a virtual environment:**

bash

CopyEdit

python3 -m venv lidar\_env

2️. **Activate the virtual environment:**

bash

CopyEdit

source lidar\_env/bin/activate

3️. **Now install the required packages inside the virtual environment:**

bash

CopyEdit

pip install rplidar-roboticia matplotlib

4️. R**un your Python scripts while the virtual environment is active.**  
Whenever you restart the Pi, reactivate it using:

bash

CopyEdit

source lidar\_env/bin/activate

**Run a Python Test Script**

Now, let's write a Python script to check if LiDAR is scanning and returning distance data.

**📜 Test Script: Read and Print LiDAR Data**

1️. **Create a Python file**:

bash

CopyEdit

nano test\_lidar.py

2️. **Copy & Paste the following code**:

python

CopyEdit

from rplidar import RPLidar

# Change the port if needed (for GPIO use '/dev/ttyS0')

LIDAR\_PORT = '/dev/ttyUSB0'

# Initialize RPLIDAR

lidar = RPLidar(LIDAR\_PORT)

print("Starting RPLIDAR scan... Press Ctrl+C to stop.")

try:

for i, scan in enumerate(lidar.iter\_scans()):

print(f'Scan {i}: {scan}')

if i > 10: # Stop after 10 scans

break

except KeyboardInterrupt:

print("Stopping...")

finally:

lidar.stop()

lidar.disconnect()

3️. **Save and exit** (press **Ctrl + X**, then **Y**, then **Enter**)

4️. **Run the script**:

bash

CopyEdit

python3 test\_lidar.py

📌 If everything is working correctly, it will print **distance data** for **each angle** in real time.

**🔹 Step 3: Visualize LiDAR Data**

Now, let's plot a **real-time LiDAR scan** using matplotlib.

**📜 Script to Visualize LiDAR Data**

1️. **Create a new Python file**:

bash

CopyEdit

nano visualize\_lidar.py

2️. **Copy & Paste the following code**:

python

CopyEdit

import numpy as np

import matplotlib.pyplot as plt

from rplidar import RPLidar

# Define LiDAR port

LIDAR\_PORT = '/dev/ttyUSB0'

# Initialize RPLidar

lidar = RPLidar(LIDAR\_PORT)

def update\_plot():

fig, ax = plt.subplots(subplot\_kw={'projection': 'polar'})

ax.set\_ylim(0, 6000) # Set max range to 6 meters

try:

for \_, scan in enumerate(lidar.iter\_scans()):

angles = [np.deg2rad(meas[1]) for meas in scan] # Convert angles to radians

distances = [meas[2] for meas in scan]

ax.clear()

ax.scatter(angles, distances, c='b', alpha=0.75) # Plot points

ax.set\_ylim(0, 6000) # Keep range fixed

plt.pause(0.1) # Update plot

except KeyboardInterrupt:

print("Stopping visualization...")

finally:

lidar.stop()

lidar.disconnect()

update\_plot()

3️. **Save and exit** (press **Ctrl + X**, then **Y**, then **Enter**)

4️. **Run the visualization script**:

bash

CopyEdit

python3 visualize\_lidar.py

Output:  
Scan 0: [(15, 324.21875, 2654.25), (15, 325.484375, 2625.5), (15, 326.71875, 2595.0), (13, 343.140625, 2047.25), (15, 344.375, 2048.25), (15, 349.390625, 1982.75)] Scan 1: [(15, 351.859375, 2083.75), (15, 353.125, 2062.75), (15, 354.34375, 2046.5), (15, 355.578125, 2056.0), (15, 358.015625, 2326.0), (15, 359.21875, 2359.5), (15, 4.234375, 2471.25), (15, 5.46875, 2462.75), (15, 7.9375, 2569.0), (15, 9.1875, 2582.0), (15, 10.40625, 2613.75), (15, 11.65625, 2657.5), (15, 12.875, 2670.75), (15, 15.421875, 2570.5), (15, 16.640625, 2580.75), (15, 17.875, 2601.0), (15, 19.125, 2619.0), (15, 25.34375, 2765.0), (15, 26.578125, 2805.5), (15, 27.8125, 2851.75), (15, 38.0625, 129.0), (15, 72.015625, 1983.0), (15, 73.296875, 1976.0), (15, 74.515625, 1965.75), (15, 75.765625, 1968.75), (15, 77.046875, 1974.5), (15, 78.3125, 1964.75), (15, 79.515625, 1962.25), (15, 80.8125, 1969.5), (15, 82.03125, 1968.75), (15, 84.703125, 1556.0), (15, 85.921875, 1516.75), (15, 87.203125, 1500.5), (15, 88.484375, 1511.25), (15, 89.71875, 1525.5), (15, 90.921875, 1525.75), (15, 92.25, 1513.5), (15, 93.453125, 1505.25), (15, 94.703125, 1521.75), (15, 107.5625, 1029.25), (15, 108.890625, 1030.25), (15, 110.140625, 1040.5), (15, 118.8125, 319.75), (15, 120.96875, 300.75), (15, 122.265625, 296.0), (15, 124.328125, 292.5), (15, 125.203125, 290.0), (15, 126.78125, 287.25), (15, 127.609375, 287.0), (15, 132.78125, 303.25), (15, 134.125, 303.5), (15, 136.515625, 293.0), (15, 145.171875, 311.0), (15, 145.59375, 309.25), (15, 147.34375, 309.75), (15, 148.84375, 310.5), (15, 150.0, 312.0), (15, 151.09375, 314.25), (15, 152.171875, 316.25), (15, 152.984375, 318.5), (15, 154.671875, 321.5), (15, 156.0625, 324.25), (15, 156.640625, 327.25), (15, 158.296875, 330.75), (15, 159.578125, 334.25), (15, 160.5625, 338.5), (15, 161.859375, 342.75), (15, 162.953125, 347.25), (15, 163.71875, 352.0), (15, 165.09375, 357.0), (15, 166.078125, 362.5), (15, 167.4375, 368.75), (15, 168.90625, 375.75), (15, 174.828125, 560.5), (15, 176.015625, 561.0), (15, 177.359375, 562.5), (15, 178.625, 565.25), (15, 179.8125, 567.5), (15, 181.03125, 570.25), (15, 182.265625, 573.5), (15, 183.390625, 576.0), (15, 184.796875, 579.0), (15, 185.859375, 582.5), (15, 187.25, 586.5), (15, 188.328125, 590.25), (15, 189.5625, 594.0), (15, 190.703125, 599.0), (15, 192.0, 604.25), (15, 193.3125, 609.5), (15, 194.484375, 615.0), (15, 195.78125, 621.5), (15, 196.8125, 628.25), (15, 198.203125, 635.5), (15, 199.421875, 642.75), (15, 200.578125, 651.25), (15, 201.875, 660.0), (15, 203.046875, 669.75), (15, 204.171875, 680.25), (15, 205.40625, 690.25), (15, 206.609375, 702.0), (15, 207.90625, 713.5), (15, 209.1875, 724.75), (15, 210.3125, 738.75), (15, 211.484375, 754.75), (15, 212.6875, 770.75), (15, 213.921875, 785.25), (15, 215.140625, 801.75), (15, 216.359375, 822.25), (15, 217.546875, 842.75), (15, 218.796875, 862.5), (15, 220.0625, 883.5), (15, 221.234375, 908.75), (15, 222.375, 936.25), (15, 223.546875, 964.5), (15, 224.84375, 993.75), (15, 229.828125, 1112.0), (15, 231.09375, 1094.0), (15, 232.390625, 1083.0), (15, 233.625, 1073.0), (15, 234.859375, 1060.25), (15, 236.171875, 1047.25), (15, 237.34375, 1039.5), (15, 238.703125, 1036.0), (15, 239.9375, 1027.5), (15, 241.25, 1017.5), (15, 242.484375, 1012.0), (15, 243.765625, 1009.25), (15, 245.0, 1003.5), (15, 246.28125, 995.75), (15, 247.515625, 990.75), (15, 248.78125, 987.5), (15, 249.96875, 985.25), (15, 251.234375, 981.75), (15, 252.46875, 978.5), (15, 253.78125, 976.0), (15, 254.984375, 975.75), (15, 256.203125, 975.25), (15, 257.53125, 974.25), (15, 258.6875, 975.0), (15, 260.0, 976.0), (15, 261.171875, 977.5), (15, 262.453125, 979.0), (15, 263.703125, 980.25), (15, 264.953125, 981.0), (15, 266.15625, 983.25), (15, 267.453125, 986.5), (15, 268.6875, 988.5), (15, 269.984375, 994.75), (15, 271.109375, 1001.0), (15, 272.375, 1005.0), (15, 277.625, 882.0), (15, 278.8125, 869.5), (15, 280.078125, 854.75), (15, 281.265625, 851.75), (15, 283.78125, 922.5), (15, 284.90625, 936.0), (15, 286.125, 943.75), (15, 287.390625, 950.25), (15, 289.8125, 1012.25), (15, 291.578125, 2034.25), (15, 292.8125, 2058.0), (15, 294.078125, 2092.5), (15, 295.296875, 2140.0), (15, 296.5, 2190.5), (15, 297.765625, 2231.75), (15, 298.984375, 2273.25), (15, 300.265625, 2320.0), (15, 302.8125, 2158.25), (15, 304.078125, 2116.75), (15, 305.3125, 2069.0), (11, 306.5625, 2033.75), (15, 307.8125, 2004.75), (14, 309.109375, 1968.5), (10, 310.34375, 1942.0), (15, 311.640625, 1911.5), (15, 322.65625, 2643.75), (15, 323.90625, 2661.25), (15, 325.140625, 2637.75), (15, 326.375, 2607.5), (15, 329.8125, 996.5), (15, 340.28125, 2057.0), (15, 341.53125, 2048.75), (15, 345.28125, 2021.5), (15, 346.484375, 2052.0), (15, 348.984375, 1982.5)]

It looks like your LiDAR is successfully scanning and returning data points in the format:

scss

CopyEdit

(distance, angle, intensity)

Visualized\_code :  
  
Here is the **modified script** with improvements to clear the buffer before scanning and handle errors properly:

**✅ Modified visualize\_lidar.py**

python

CopyEdit

import time

import sys

from rplidar import RPLidar, RPLidarException

# Define the LiDAR USB Port

LIDAR\_PORT = "/dev/ttyUSB0"

# Initialize LiDAR

try:

lidar = RPLidar(LIDAR\_PORT, baudrate=115200)

print("Lidar initialized successfully!")

except RPLidarException as e:

print(f"Error initializing LiDAR: {e}")

sys.exit(1)

time.sleep(2) # Wait for LiDAR to initialize

# Clear Input Buffer

try:

lidar.clean\_input() # Ensure old data is cleared

print("Buffer cleaned!")

except AttributeError:

print("Warning: `clean\_input` not found! Trying alternative method...")

lidar.stop()

lidar.disconnect()

time.sleep(2)

lidar = RPLidar(LIDAR\_PORT, baudrate=115200)

# Function to update LiDAR scan

def update\_plot():

try:

for i, scan in enumerate(lidar.iter\_scans()):

print(f"Scan {i+1}: {scan}") # Display scan data

if i > 5:

break # Stop after 5 scans

except RPLidarException as e:

print(f"Error during scan: {e}")

except KeyboardInterrupt:

print("\nUser interrupted! Stopping LiDAR...")

finally:

lidar.stop()

lidar.disconnect()

print("LiDAR stopped and disconnected.")

# Start scanning

update\_plot()

**🔥 What’s Improved?**

1. ✅ **Handles "Incorrect descriptor" errors** by clearing the buffer.
2. ✅ **Cleans input before scanning** to avoid old data.
3. ✅ **Stops LiDAR gracefully** when an error or Ctrl+C occurs.
4. ✅ **Handles missing clean\_input() method** by restarting the LiDAR.
5. ✅ **Limits scans to 5** to avoid infinite loops while debugging.

**How to Run It**

1. Save this script as visualize\_lidar.py.
2. Run:

bash

CopyEdit

python3 visualize\_lidar.py

1. If errors persist, check the USB port:

bash

CopyEdit

ls /dev/ttyUSB\*

**Output :**

(15, 281.875, 879.25), (15, 284.453125, 920.0), (15, 285.875, 923.0), (15, 287.171875, 924.25), (15, 288.40625, 926.75), (15, 289.6875, 938.5), (15, 290.96875, 958.25), (15, 295.65625, 1710.5), (15, 298.34375, 1636.25), (15, 299.65625, 1677.75), (15, 301.015625, 1725.5), (15, 302.25, 1771.75), (15, 303.59375, 1825.0), (15, 304.875, 1878.75), (15, 308.765625, 2214.25), (15, 311.453125, 2072.75), (15, 312.8125, 2027.5), (10, 314.125, 1973.5), (15, 318.171875, 1850.0), (15, 319.546875, 1813.0), (15, 320.875, 1777.5), (15, 335.3125, 2492.5), (15, 336.640625, 2470.75), (15, 337.96875, 2430.5), (15, 339.296875, 2397.25), (15, 340.625, 2380.5), (15, 341.96875, 2362.5), (15, 343.28125, 2349.25), (15, 344.625, 2335.5), (15, 348.5625, 2297.25), (15, 349.90625, 2268.75)] LiDAR stopped and disconnected.

**Code of Testing:**  
  
import sys

from rplidar import RPLidar

# Replace with the correct port for your system

PORT\_NAME = '/dev/ttyUSB0'

# Initialize the LIDAR (without baudrate argument)

lidar = RPLidar(PORT\_NAME)

def get\_lidar\_data():

try:

print("Starting RPLidar A1 scanning...")

for scan in lidar.iter\_scans():

for (\_, angle, distance) in scan:

print(f"Angle: {angle:.2f} degrees, Distance: {distance/1000:.2f} meters")

except KeyboardInterrupt:

print("\nScan stopped by user.")

except Exception as e:

print(f"Error: {e}")

finally:

print("Stopping and disconnecting LiDAR...")

lidar.stop()

lidar.disconnect()

if \_\_name\_\_ == "\_\_main\_\_":

get\_lidar\_data()  
  
**Output :**Angle: 145.64 degrees, Distance: 0.30 meters

Angle: 146.19 degrees, Distance: 0.30 meters

Angle: 148.25 degrees, Distance: 0.30 meters

Angle: 149.16 degrees, Distance: 0.29 meters

Angle: 150.22 degrees, Distance: 0.29 meters

Angle: 151.88 degrees, Distance: 0.30 meters