

3. Get distance information_ROS

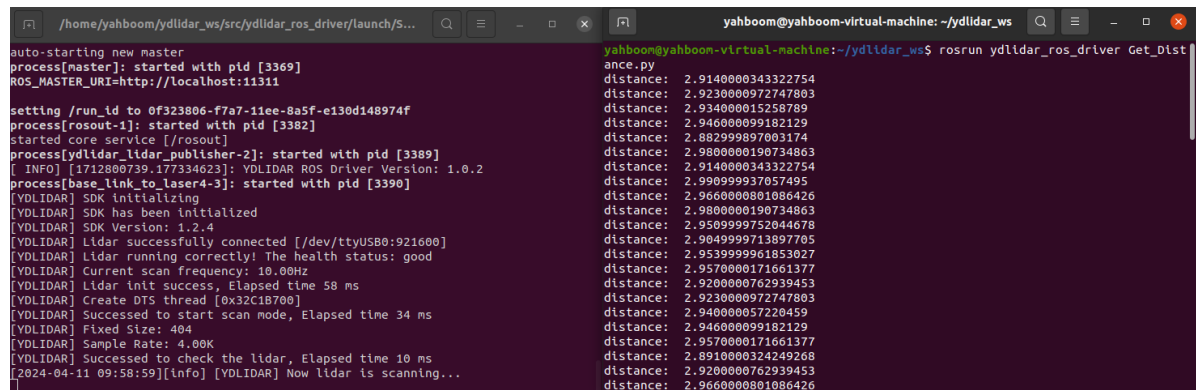
1. Start-up

ROS1

Terminal input,

```
roslaunch ydlidar_ros_driver SDM18.launch
roswrun ydlidar_ros_driver Get_Distance.py
```

After successful operation, the distance will be printed on the terminal,



The screenshot shows two terminal windows. The left window displays the output of `roslaunch ydlidar_ros_driver SDM18.launch`, showing the initialization of the YDLIDAR ROS driver. The right window displays the output of `roswrun ydlidar_ros_driver Get_Distance.py`, which prints a series of distance measurements in meters.

```
auto-starting new master
process[master]: started with pid [3369]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 0f323806-f7a7-11ee-8a5f-e130d148974f
process[rosout-1]: started with pid [3382]
started core service [/rosout]
process[ydlidar_lidar_publisher-2]: started with pid [3389]
[ INFO ] [1712800739.177334623]: YDLIDAR ROS Driver Version: 1.0.2
process[base_link_to_laser4-3]: started with pid [3390]
[YDLIDAR] SDK Initializing
[YDLIDAR] SDK has been initialized
[YDLIDAR] SDK Version: 1.2.4
[YDLIDAR] Lidar successfully connected [/dev/ttyUSB0:921600]
[YDLIDAR] Lidar running correctly! The health status: good
[YDLIDAR] Current scan frequency: 10.00Hz
[YDLIDAR] Lidar init success, Elapsed time 58 ms
[YDLIDAR] Create DTS thread [0x32C1B700]
[YDLIDAR] Successfully to start scan mode, Elapsed time 34 ms
[YDLIDAR] Fixed Size: 404
[YDLIDAR] Sample Rate: 4.00K
[YDLIDAR] Successfully to check the lidar, Elapsed time 10 ms
[2024-04-11 09:58:59][info] [YDLIDAR] Now lidar is scanning...

distance: 2.9140000343322754
distance: 2.9230000972747803
distance: 2.934000015258789
distance: 2.946000099182129
distance: 2.882999997003174
distance: 2.9880000190734863
distance: 2.9140000343322754
distance: 2.990999937057495
distance: 2.9660000801086426
distance: 2.9800000190734863
distance: 2.9509999752044678
distance: 2.9049999713897705
distance: 2.9539999961853027
distance: 2.9570000171661377
distance: 2.9200000762939453
distance: 2.9230000972747803
distance: 2.940000057220459
distance: 2.946000099182129
distance: 2.9570000171661377
distance: 2.8910000324249268
distance: 2.9200000762939453
distance: 2.9660000801086426
```

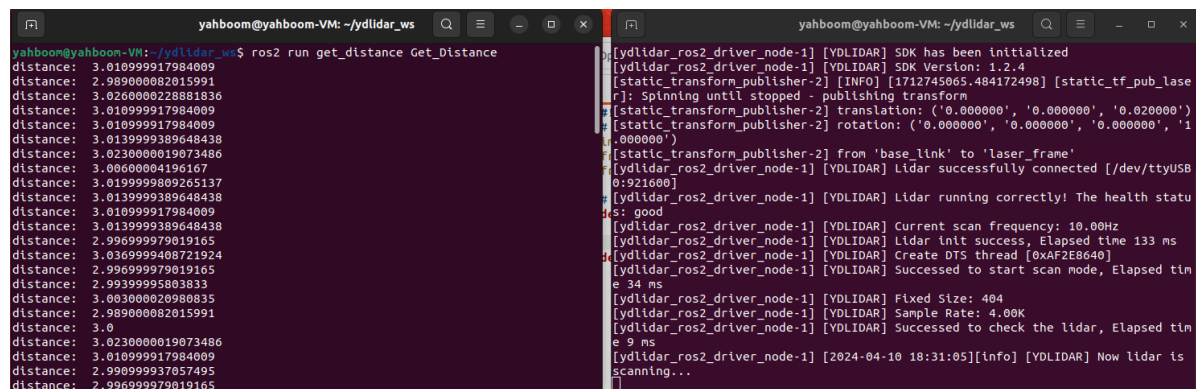
The distance here represents the distance measured by the module, in meters(m)

ROS2

Terminal input

```
ros2 launch ydlidar_ros2_driver ydlidar_launch.py
ros2 run get_distance Get_Distance
```

After successful operation, the distance will be printed on the terminal,



The screenshot shows two terminal windows. The left window displays the output of `ros2 launch ydlidar_ros2_driver ydlidar_launch.py`, showing the initialization of the YDLIDAR ROS2 driver. The right window displays the output of `ros2 run get_distance Get_Distance`, which prints a series of distance measurements in meters.

```
distance: 3.010999917984009
distance: 2.989000082015991
distance: 3.0260000228881836
distance: 3.010999917984009
distance: 3.010999917984009
distance: 3.0139999389648438
distance: 3.0230000019073486
distance: 3.00600004196167
distance: 3.0199999809265137
distance: 3.0139999389648438
distance: 3.010999917984009
distance: 3.0139999389648438
distance: 2.996999979019165
distance: 3.0369999408721924
distance: 2.996999979019165
distance: 2.99399995803833
distance: 3.003000020980835
distance: 2.989000082015991
distance: 3.0
distance: 3.0230000019073486
distance: 3.010999917984009
distance: 2.990999937057495
distance: 2.996999979019165

[ydlidar_ros2_driver_node-1] [YDLIDAR] SDK has been initialized
[ydlidar_ros2_driver_node-1] [YDLIDAR] SDK Version: 1.2.4
[static_transform_publisher-2] [INFO] [1712745065.484172498] [static_tf_pub_lase
r]: Spinning until stopped - publishing transform
[static_transform_publisher-2] translation: ('0.000000', '0.000000', '0.020000')
[static_transform_publisher-2] rotation: ('0.000000', '0.000000', '0.000000', '1
.000000')
[static_transform_publisher-2] from 'base_link' to 'laser_frame'
[ydlidar_ros2_driver_node-1] [YDLIDAR] Lidar successfully connected [/dev/ttyUSB
0:921600]
[ydlidar_ros2_driver_node-1] [YDLIDAR] Lidar running correctly! The health statu
s: good
[ydlidar_ros2_driver_node-1] [YDLIDAR] Current scan frequency: 10.00Hz
[ydlidar_ros2_driver_node-1] [YDLIDAR] Lidar init success, Elapsed time 133 ms
[ydlidar_ros2_driver_node-1] [YDLIDAR] Create DTS thread [0xAF2E8640]
[ydlidar_ros2_driver_node-1] [YDLIDAR] Successfully to start scan mode, Elapsed tim
e 34 ms
[ydlidar_ros2_driver_node-1] [YDLIDAR] Fixed Size: 404
[ydlidar_ros2_driver_node-1] [YDLIDAR] Sample Rate: 4.00K
[ydlidar_ros2_driver_node-1] [YDLIDAR] Successfully to check the lidar, Elapsed tim
e 9 ms
[ydlidar_ros2_driver_node-1] [2024-04-10 18:31:05][info] [YDLIDAR] Now lidar is
scanning...
```

The distance here represents the distance measured by the module, in meters.

2.code

Taking ROS2 as an example, ROS1 can refer to the source code path we provide

ROS2 source code location:

- Function package installation location: ~/ydlidar_ws/src/ydlidar_ros2_driver)
- Get data py file: ~/ydlidar_ws/src/get_distance/get_distance/Get_Distance.py

ROS1 source code location:

- Function package installation location: ~/ydlidar_ros_ws/src/ydlidar_ros_driver)
- Get data py file: ~/ydlidar_ros_ws/src/ydlidar_ros_driver/scripts/Get_Distance.py

Note: Search based on the installation location of your feature pack, which is located in the scripts section of the feature pack directory.

```
#!/usr/bin/env python
# coding:utf-8
import rclpy
from sensor_msgs.msg import LaserScan
from rclpy.qos import QoSProfile

#
def scancallback(scan_data):
    print("distance: ", scan_data.ranges[0])

def GetSDMData():
    rclpy.init() # ROS2 node init
    node = rclpy.create_node('Get_SDM_Data')
    qos_policy =
rclpy.qos.QoSProfile(reliability=rclpy.qos.ReliabilityPolicy.BEST_EFFORT,

    history=rclpy.qos.HistoryPolicy.KEEP_LAST,
                        depth=1)

    node.create_subscription(LaserScan, 'scan',
scancallback,qos_profile=qos_policy)
    rclpy.spin(node)
    rclpy.shutdown()

def main():
    GetSDMData()
```

The code is very short. After running the launch file, a topic data message of '/scan' will be published. We set the

Assign a subscriber to subscribe to this message,(The qos_policy here is because the QOS communication method of this radar has changed)

```
node.create_subscription(LaserScan, 'scan', scancallback,qos_profile=qos_policy)
```

Then, in the callback function, we print out the data for this message,

```
def scancallback(scan_data):  
    print("distance: ", scan_data.ranges[0])
```

We are in rostopic echo /scan knowable, The distance information is saved in the ranges list, and we can read the content of this list. The subscript is 0 because the list only has one data.