3.Get distance information_ROS

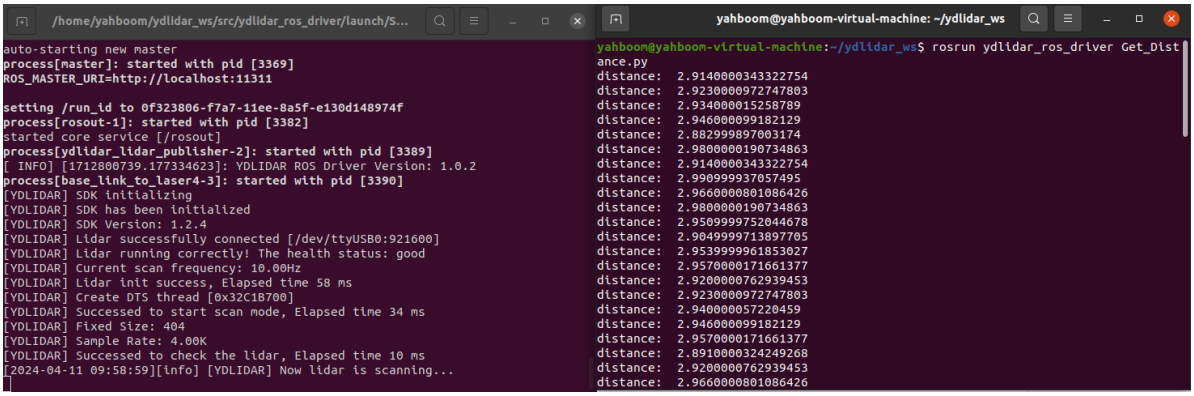
1.Start-up

ROS1

Terminal input,

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

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



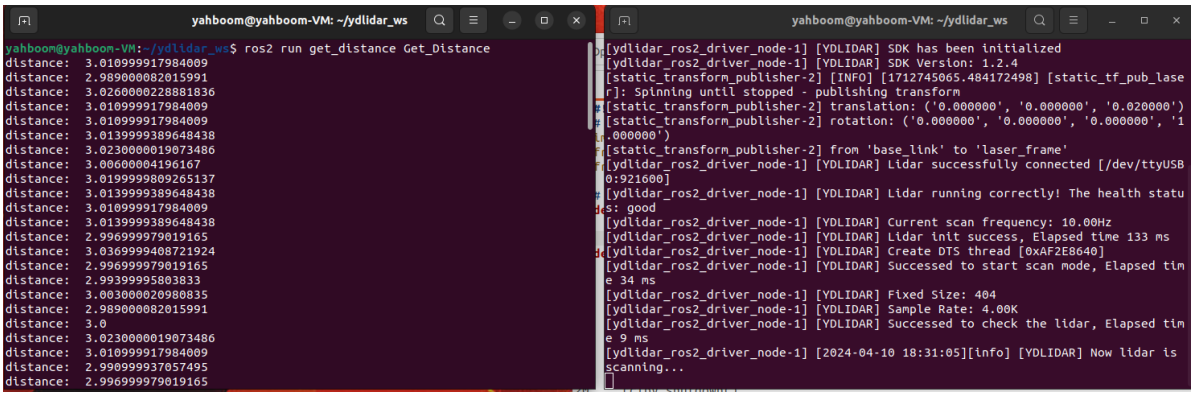
The distance here represents the distance measured by the module, in meters(m)

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 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.