1. Environment setup

1. Install sdk driver

Based on the specifications of the lidar you purchased, locate the compressed file labeled "YDLidar-SDK" in the provided source code package. Extract the "YDLidar-SDK" folder; this folder contains the SDK files for this lidar. Since using the ROS function package for this lidar requires the SDK to be installed beforehand, the "YDLidar-SDK" folder contains the lidar's driver files. Open a terminal in this folder and type:

```
mkdir build
cd build
cmake..
make -j4
sudo make install
```

If no errors are reported during operation, the driver is successfully installed.

2. Create a new workspace and compile function packages

- (Recommended) The first method is to unzip yahboomcar_ws in the source code to your own root directory, and then compile it directly using colcon build.

```
cd yahboomcar_ws
colcon build
```

After the compilation is passed, add the path of the workspace to .bashrc.

```
sudo gedit ~/.bashrc
```

Copy the following content to the end of the file,

```
source ~/yahboomcar_ws/devel/setup.bash --extend
```

- The second method is to create a self-named workspace. Take the name oradar_ws as an example and enter it in the terminal.

```
mkdir oradar_ws
cd oradar_ws
mkdir src
cd src
catkin_init_workspace
```

Then copy the decompressed source code yahboomcar_ws/src function package to the oradar_ws/src directory, and then use colcon build to compile in the oradar_ws directory.

```
cd oradar_ws
colcon build
```

After the compilation is passed, add the path of the workspace to .bashrc.

```
sudo gedit ~/.bashrc
```

Copy the following content to the end of the file,

```
source ~/oradar_ws/devel/setup.bash --extend
```

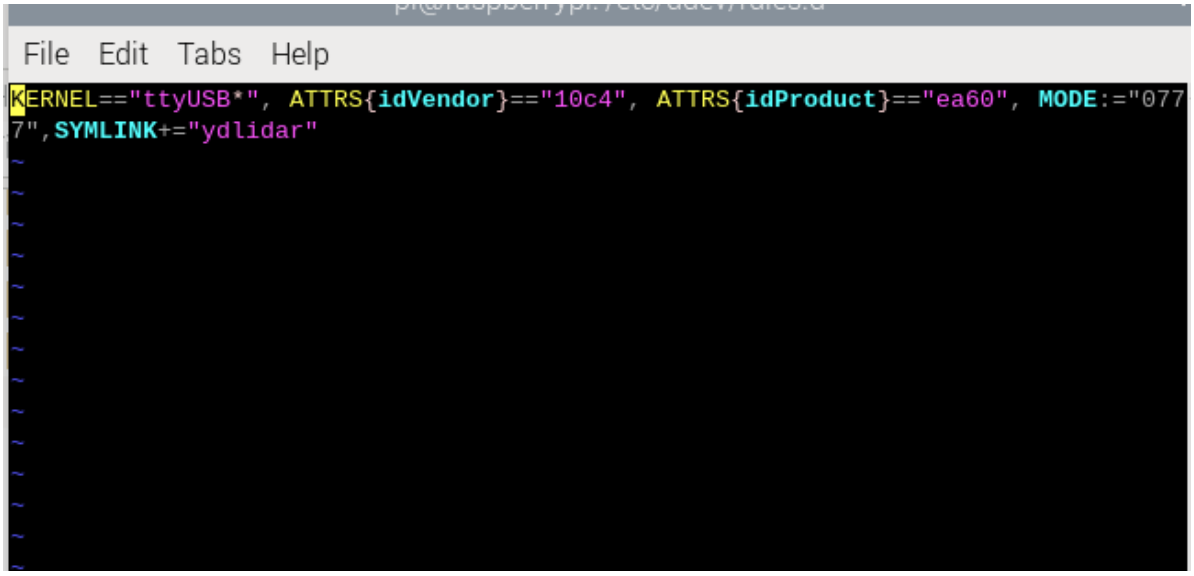
3. Bind lidar port name

Open the terminal in the root directory of the Raspberry Pi and enter the following command,

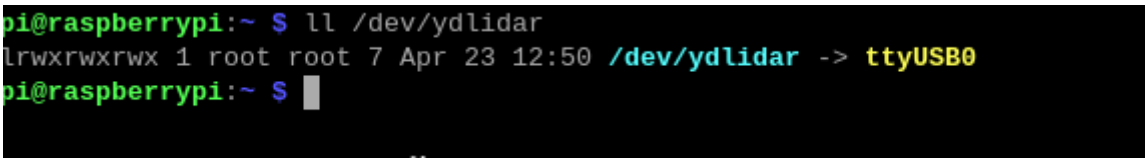
```
cd /etc/udev/rules.d/
sudo vi usb.rules
```

Then copy the following content into it,

```
KERNEL=="ttyUSB*", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60",
MODE:="0777", SYMLINK+="ydlidar"
```



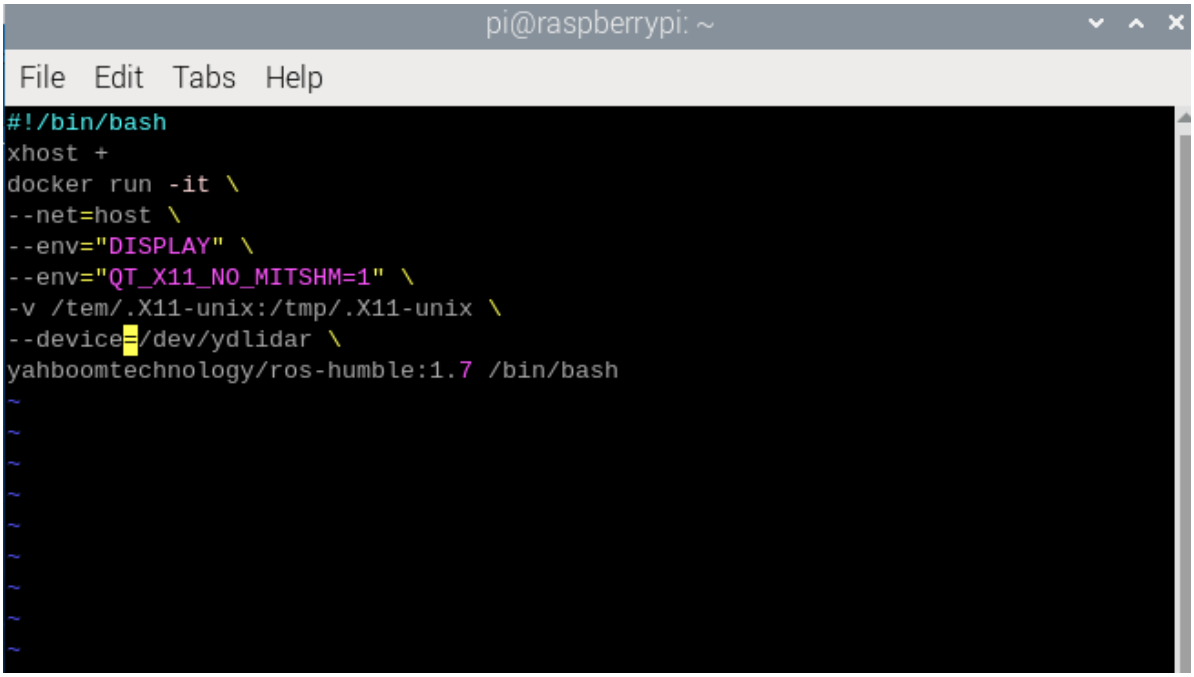
Then re-plug the radar serial port and enter ll /dev/ydlidar in the terminal.



The above content indicates that the binding is successful. The end is not necessarily 0 and changes according to the order in which the devices are inserted.

Then open our docker script and add it

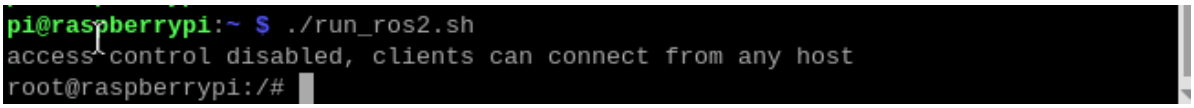
```
vi run_ros2.sh
```



4. Drive radar

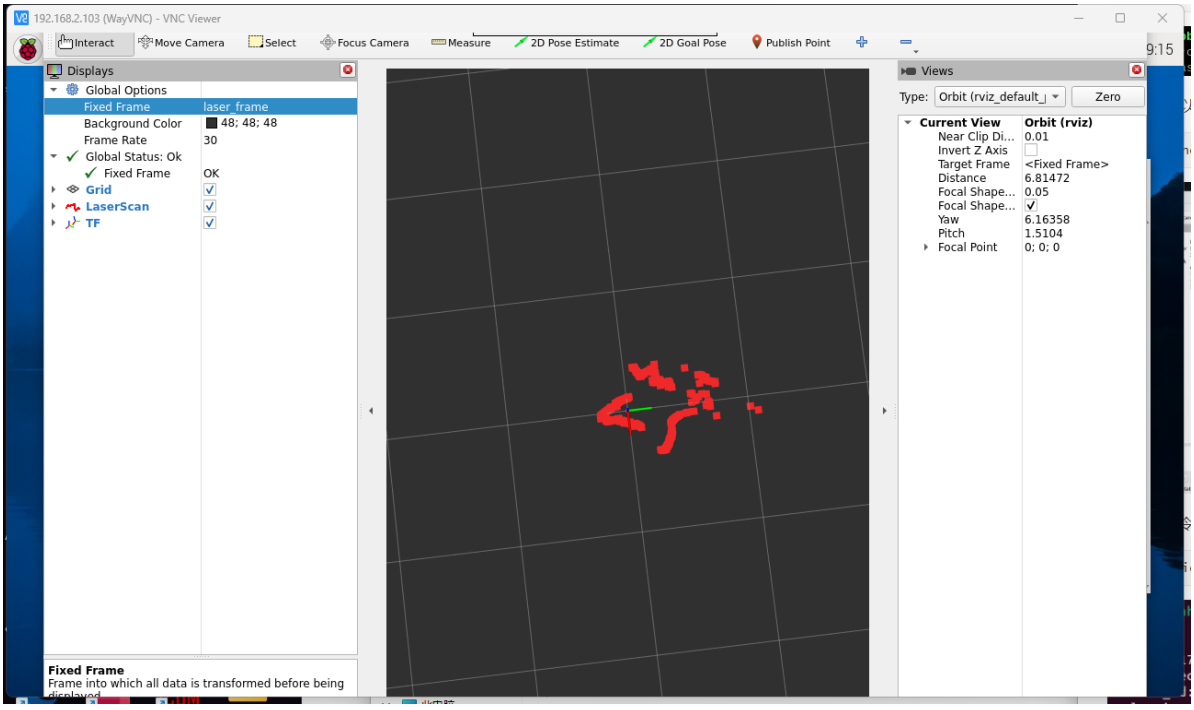
First go into the docker we provide, open a terminal in the Raspberry Pi directory and enter,

```
./run_ros2.sh
```



After that, enter the following statement to open the radar and display it in rviz,

```
ros2 launch ydlidar_ros2_driver ydlidar_launch_view.py
```



You can view radar node data through the following command, provided that you need to enter the same docker.

Enter in the Raspberry Pi terminal, which ID needs to be remembered?

```
docker ps
```

```
pi@raspberrypi:~ $ docker ps
CONTAINER ID   IMAGE                                COMMAND       CREATED
STATUS         PORTS     NAMES
b43de2c3c95f   yahboomtechnology/ros-humble:1.8   "/bin/bash"   2 minutes ago
Up 2 minutes             silly_lehmann
```

Then enter the command to log in to the same docker container

```
docker exec -it b43d /bin/bash
```

```
pi@raspberrypi:~ $ docker exec -it b43d /bin/bash
root@raspberrypi:/#
```

Enter the following command to view radar data,

```
ros2 topic echo /scan
```

```
header:
  stamp:
    sec: 1713946757
    nanosec: 181098000
  frame_id: laser_frame
angle_min: -3.1415927410125732
angle_max: 3.1415927410125732
angle_increment: 0.015591030940413475
time_increment: 0.00025105764507316053
scan_time: 0.1001719981431961
range_min: 0.029999999329447746
range_max: 12.0
ranges:
- 0.0
- 0.08900000154972076
- 0.0860000029206276
- 0.09000000357627869
- 0.08100000023841858
- 0.08100000023841858
- 0.08699999749660492
```