Basic distance measurement function

Note: The virtual machine needs to be in the same LAN as the car, and the ROS_DOMAIN_ID needs to be consistent. You can view [Expansion Board Parameter Configuration] to set the IP and ROS_DOMAIN_ID on the board.

1. Program function description

When the car is connected to the proxy, the distance information directly in front of the radar will be automatically displayed below the OELD screen. After running the program, the radar on the car scans the nearest object in the front range. If the object is closer to the radar than the set distance, the buzzer on the car will sound as a warning. The dynamic parameter regulator can be used to adjust the parameters such as the radar detection range and the distance of the ranging detection.

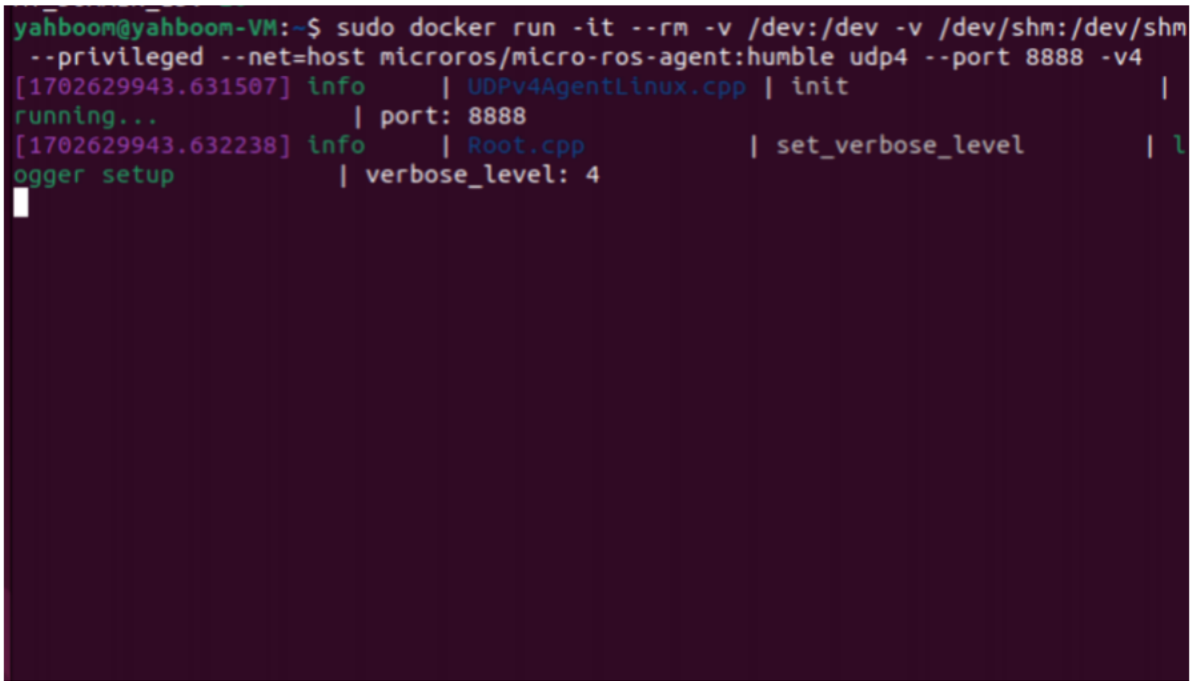
2. Start and connect the agent

If you are using the factory virtual machine system, you can enter in the terminal:

```
sh ~/start_agent_computer.sh
```

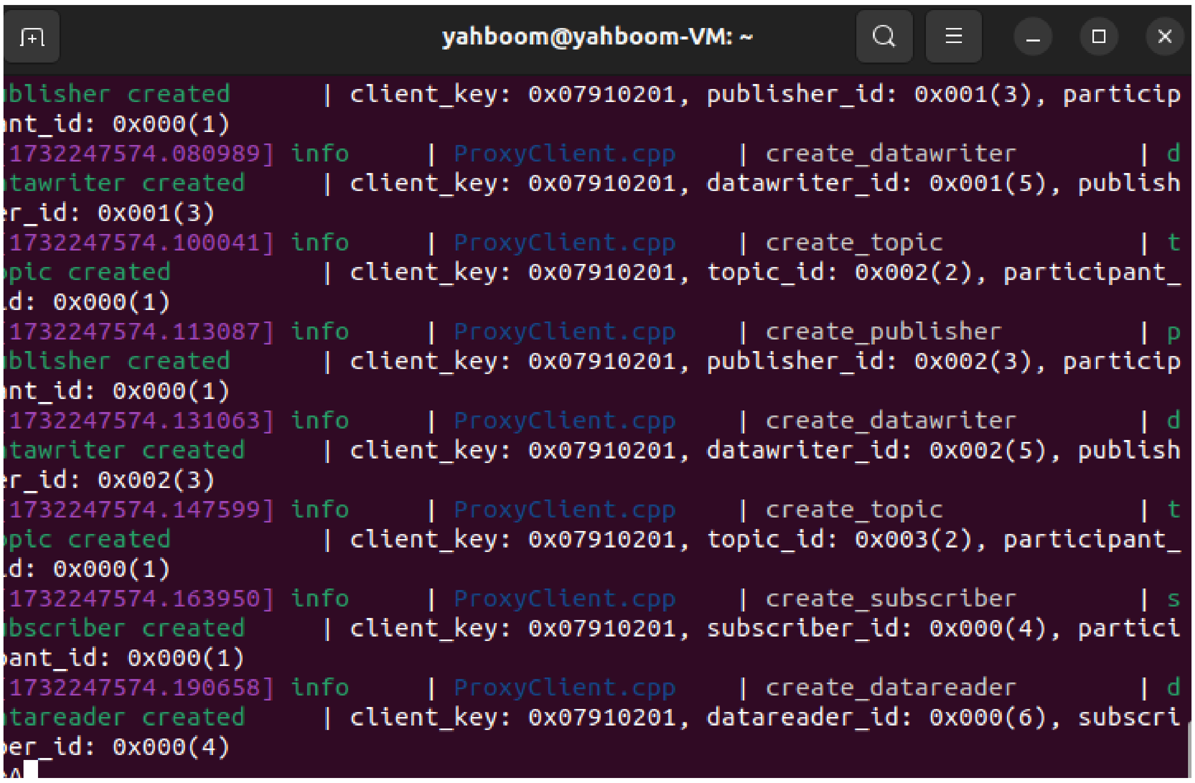
If you are using a third-party virtual machine system, you need to install the docker development environment first, and open the terminal to enter:

```
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --
net=host microros/micro-ros-agent:humble udp4 --port 8090 -v4
```



```
yahboom@yahboom-VM:~$ sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm
 --privileged --net=host microros/micro-ros-agent:humble udp4 --port 8888 -v4
[1702629943.631507] info     | UDPv4AgentLinux.cpp | init                     |
running...                 | port: 8888
[1702629943.632238] info     | Root.cpp            | set_verbose_level        | l
ogger setup               | verbose_level: 4
```

Then, turn on the car switch and wait for the car to connect to the agent. The connection is successful as shown in the figure below,



```
blisher created         | client_key: 0x07910201, publisher_id: 0x001(3), particip
nt_id: 0x000(1)
1732247574.080989] info     | ProxyClient.cpp     | create_datawriter        | d
tawriter created        | client_key: 0x07910201, datawriter_id: 0x001(5), publish
r_id: 0x001(3)
1732247574.100041] info     | ProxyClient.cpp     | create_topic             | t
pic created             | client_key: 0x07910201, topic_id: 0x002(2), participant_
d: 0x000(1)
1732247574.113087] info     | ProxyClient.cpp     | create_publisher         | p
blisher created         | client_key: 0x07910201, publisher_id: 0x002(3), particip
nt_id: 0x000(1)
1732247574.131063] info     | ProxyClient.cpp     | create_datawriter        | d
tawriter created        | client_key: 0x07910201, datawriter_id: 0x002(5), publish
r_id: 0x002(3)
1732247574.147599] info     | ProxyClient.cpp     | create_topic             | t
pic created             | client_key: 0x07910201, topic_id: 0x003(2), participant_
d: 0x000(1)
1732247574.163950] info     | ProxyClient.cpp     | create_subscriber        | s
bscriber created        | client_key: 0x07910201, subscriber_id: 0x000(4), partici
ant_id: 0x000(1)
1732247574.190658] info     | ProxyClient.cpp     | create_datareader        | d
tareader created        | client_key: 0x07910201, datareader_id: 0x000(6), subscri
er_id: 0x000(4)
```

3. OLED display distance

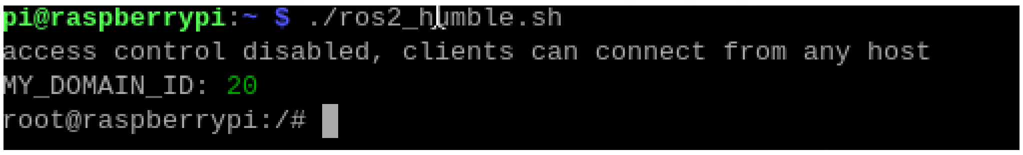
After selecting the warning level and radar model, the screen will display the selected radar, imu data, and the distance directly in front of the radar.

4. Run the program

If it is the desktop version of Raspberry Pi and the desktop version of Jetson Nano and X5, you need to enter Docker in advance and enter in the terminal,

```
sh ~/ros2_humble.sh
```

When the following interface appears, it means that you have successfully entered Docker,



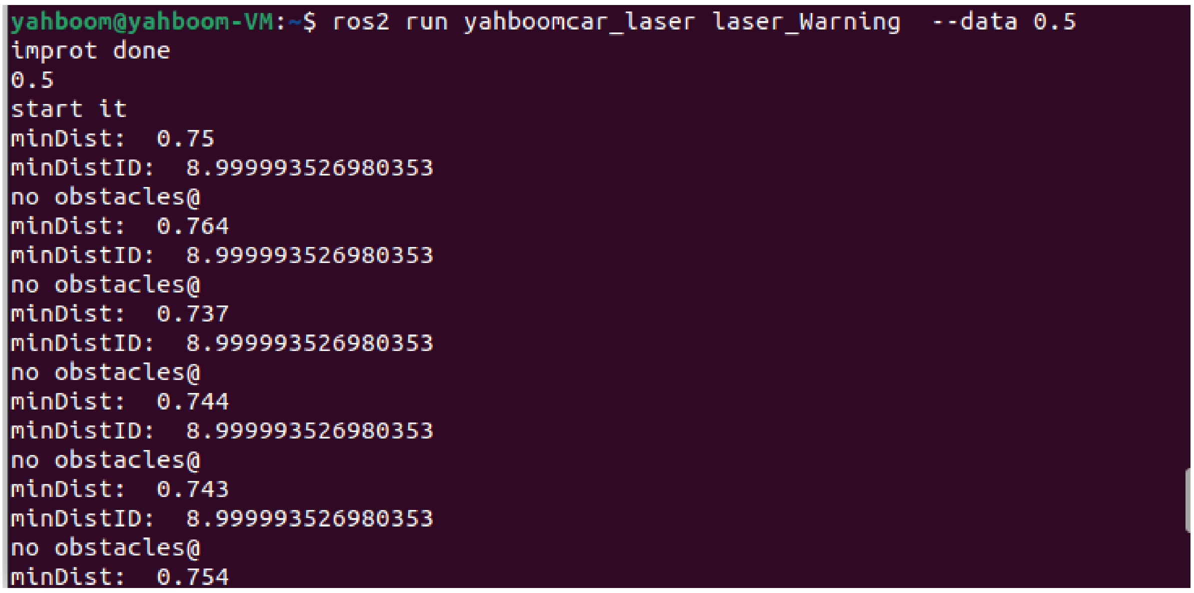
```
pi@raspberrypi:~ $ ./ros2_humble.sh
access control disabled, clients can connect from any host
MY_DOMAIN_ID: 20
root@raspberrypi:/#
```

Then enter in the Docker terminal,

```
ros2 run yahboomcar_laser laser_Warning --data 0.5
```

Take the matching virtual machine as an example, enter the following command in the terminal to start, 0.5 means that the buzzer will keep ringing when the obstacle in front is 0.5 meters away.

```
ros2 run yahboomcar_laser laser_Warning --data 0.5
```


```
yahboom@yahboom-VM:~$ ros2 run yahboomcar_laser laser_Warning  --data 0.5
improt done
0.5
start it
minDist:  0.75
minDistID:  8.999993526980353
no obstacles@
minDist:  0.764
minDistID:  8.999993526980353
no obstacles@
minDist:  0.737
minDistID:  8.999993526980353
no obstacles@
minDist:  0.744
minDistID:  8.999993526980353
no obstacles@
minDist:  0.743
minDistID:  8.999993526980353
no obstacles@
minDist:  0.754
```

After the program is started, it will search for the nearest object in the front range of the radar scan. If the distance is less than the set distance, the buzzer will sound an alarm

5. Code analysis

Source code reference path (taking the supporting virtual machine as an example):

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_laser/yahboomcar_laser
```

laser_Warnning.py, the core code is as follows,

```
#ros lib
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import LaserScan
from std_msgs.msg import Bool,UInt16
#commom lib
import os
import sys
import math
import numpy as np
import time
from time import sleep
from yahboomcar_laser.common import *
print ("improt done")
RAD2DEG = 180 / math.pi
import argparse

class laserWarning(Node):
    def __init__(self,name):
        super().__init__(name)
        #create a sub
        #雷达回调函数：处理订阅到的雷达数据
        #Radar callback function: process subscribed radar data
        self.sub_laser =
self.create_subscription(LaserScan,"/scan",self.registerScan,1)
        #create a pub
        self.pub_Buzzer = self.create_publisher(UInt16,'/beep',1)
        self.Buzzer_state = False
        self.parser = argparse.ArgumentParser()
        self.parser.add_argument('--data', type= float, help='Data to pass to the
node')
        self.parsed_args = self.parser.parse_args()

        self.args = self.parsed_args.data
        print(self.args)


    def registerScan(self, scan_data):
        if not isinstance(scan_data, LaserScan): return

        ranges = np.array(scan_data.ranges)
        minDistList = []
        minDistIDList = []
        #根据设定的雷达检测角度，找到雷达检测范围内最近的物体
        #According to the set radar detection angle, find the nearest object
within the radar detection range
        for i in range(len(ranges)):
            angle = (scan_data.angle_min + scan_data.angle_increment * i) *
RAD2DEG
            if angle > 180: angle = angle - 360
            if  abs(angle)< 10 and ranges[i] > 0:
                minDistList.append(ranges[i])
                minDistIDList.append(angle)

        #print(minDistList)
        if len(minDistList) != 0:
            minDist = min(minDistList)
            minDistID = minDistIDList[minDistList.index(minDist)]
        else:
            return

        print("minDist: ",minDist)
        print("minDistID: ",minDistID)
        #判断最小物体的雷达检测的距离是否小于设定的范围，随后做出判断是否需要让蜂鸣器响
        #Judge whether the radar detection distance of the smallest object is
less than the set range, and then determine whether the buzzer needs to sound
        if minDist <= self.args:
            print("---------------")
            b = UInt16()
            b.data = 1
            self.pub_Buzzer.publish(b)

        else:
            print("no obstacles@")
            b = UInt16()
            b.data = 0
            self.pub_Buzzer.publish(UInt16())



def main():
    rclpy.init()
    laser_warn = laserWarning("laser_Warnning")
    print ("start it")
    try:
        rclpy.spin(laser_warn)
    except KeyboardInterrupt:
        pass
    finally:
        rclpy.shutdown()
```