Radar Guard

Note: The virtual machine needs to be in the same LAN as the car, and the ROS_DOMAIN_ID needs to be the same. You can check [Must Read Before Use] to set the IP and ROS_DOMAIN_ID on the board.

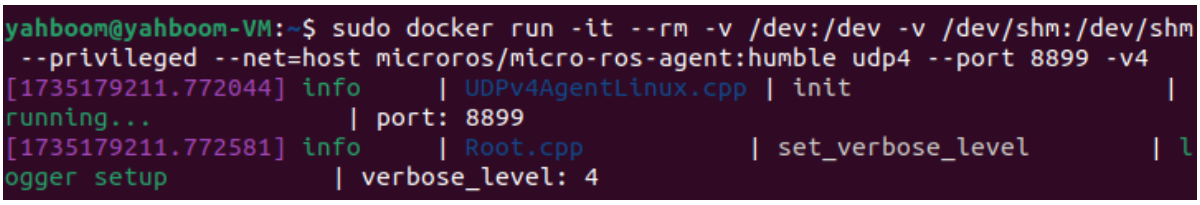
1. Program Function Description

The car connects to the proxy and runs the program. The radar on the car scans the nearest object within the set range and tracks the object by rotating. 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 obstacle avoidance detection distance.

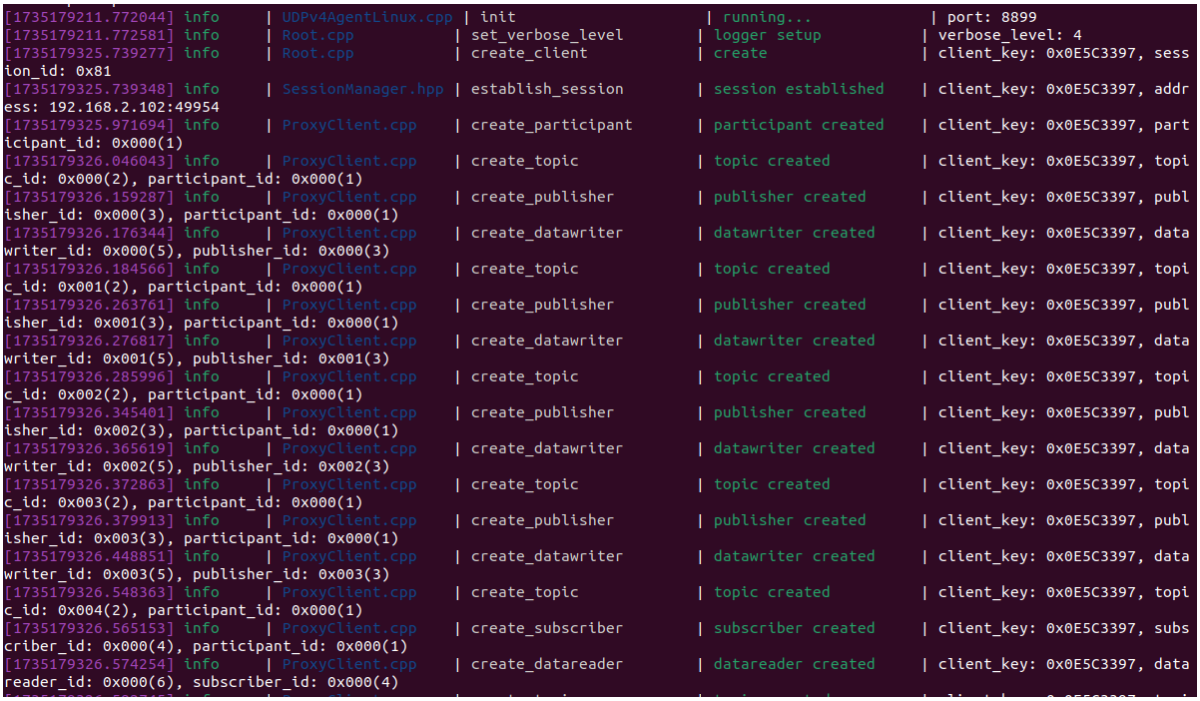
2. Start and connect the agent

Take the matching virtual machine as an example, enter the following command to start the agent,

```
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --
net=host microros/micro-ros-agent:humble udp4 --port 8899 -v4
```



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,

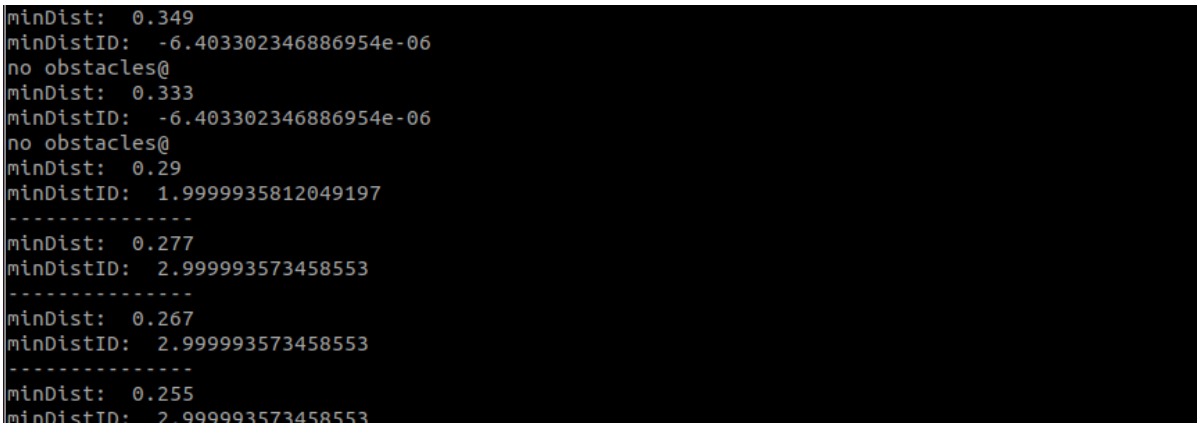


3. Start the program

3.1 Run command

Take the matching virtual machine as an example, input in the terminal,

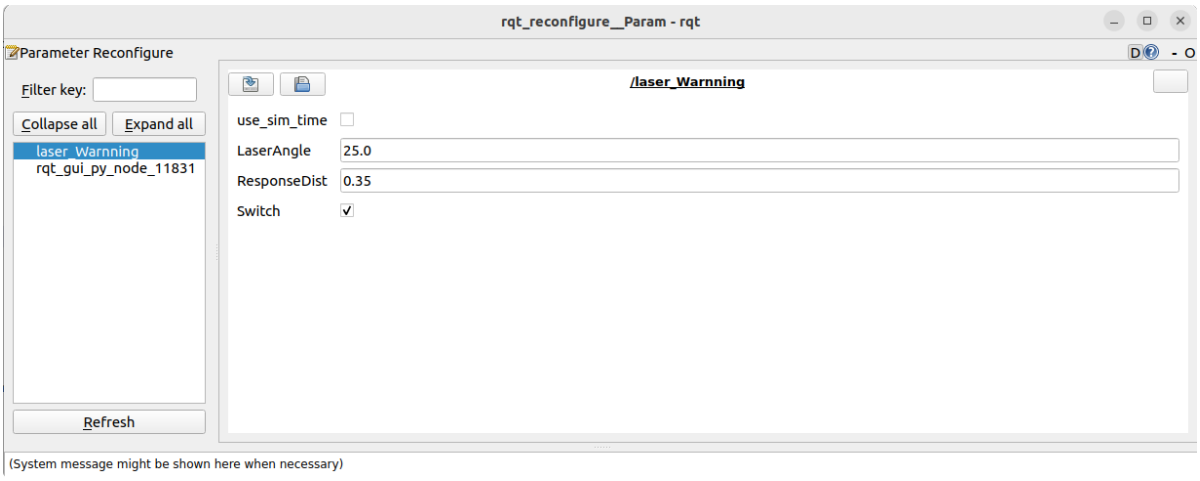
```
ros2 run yahboomcar_laser laser_Warning
```



After the program is started, it will look for the nearest object within the radar scanning range, slowly move the object, and the car will track the object by rotating.

As shown in the figure above, if it does not exceed the set range, it will print [no obstacles@], and if there is an obstacle, it will print [----------] and the buzzer will sound. You can set some parameters through the dynamic parameter regulator, terminal input,

```
ros2 run rqt_reconfigure rqt_reconfigure
```



Note: The above nodes may not be available when you first open it. Click Refresh to see all the nodes. The displayed laser_Warnning is the node of the radar guard.

Explanation of the above parameters:

- LaserAngle: Radar detection angle
- ResponseDist: Tracking distance
- Switch: Gameplay switch

After modifying the above parameters, you need to click on the blank space to pass the parameters into the program.

4. 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,

```
#Create a radar subscriber to subscribe to radar data and remote control data, a
speed publisher to publish speed data, and a buzzer publisher to publish buzzer
control data
#create a sub
self.sub_laser = self.create_subscription(LaserScan,"/scan",self.registerScan,1)
self.sub_JoyState = self.create_subscription(Bool,'/JoyState',
self.JoyStateCallback,1)

#create a pub
self.pub_vel = self.create_publisher(Twist,'/cmd_vel_bl',1)
self.pub_Buzzer = self.create_publisher(UInt16,'/beep',1)

#Radar callback function: process subscribed radar data
def registerScan(self, scan_data):
    if not isinstance(scan_data, LaserScan): return

    ranges = np.array(scan_data.ranges)
    minDistList = []
    minDistIDList = []

    for i in range(len(ranges)):
        angle = (scan_data.angle_min + scan_data.angle_increment * i) * RAD2DEG

#According to the set radar detection angle, find the nearest object within the
radar detection range
if angle > 180: angle = angle - 360
if  abs(angle) < self.LaserAngle and ranges[i] > 0:
    minDistList.append(ranges[i])
    minDistIDList.append(angle)
if len(minDistList) != 0:
    minDist = min(minDistList)
    minDistID = minDistIDList[minDistList.index(minDist)]
else:
    return

#Calculate the angular velocity based on the position deviation from the tracked
object so that the car's front end is aligned with the object
if self.Joy_active  or self.Switch == True:
    angle_pid_compute = self.ang_pid.pid_compute(minDistID/48, 0)
    if abs(angle_pid_compute) < 0.1:
        velocity.angular.z = 0.0
    else:
        velocity.angular.z = angle_pid_compute * 200
    if minDist< self.ResponseDist * 2:
        self.pub_vel.publish(velocity)
```

```
#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.ResponseDist:
    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())
```