Radar obstacle avoidance

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 check [Must-read before use] to set the IP and ROS_DOMAIN_ID on the board.

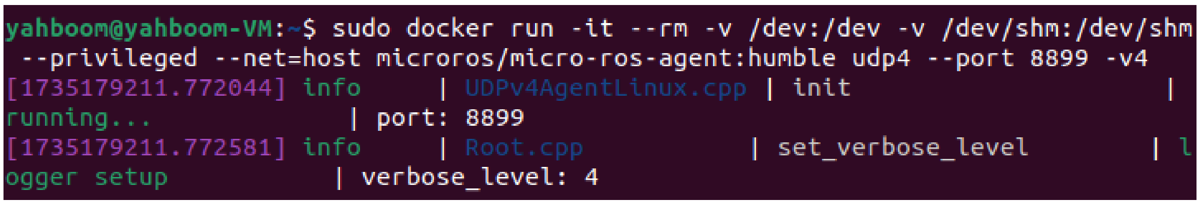
1. Program function description

The car connects to the proxy and runs the program. The radar on the car scans whether there are obstacles within the set range. If there are obstacles, it will automatically adjust the speed according to the location of the obstacle to avoid the obstacle. The dynamic parameter regulator can adjust the parameters such as the radar detection range and obstacle avoidance detection distance.

2. Start and connect the agent

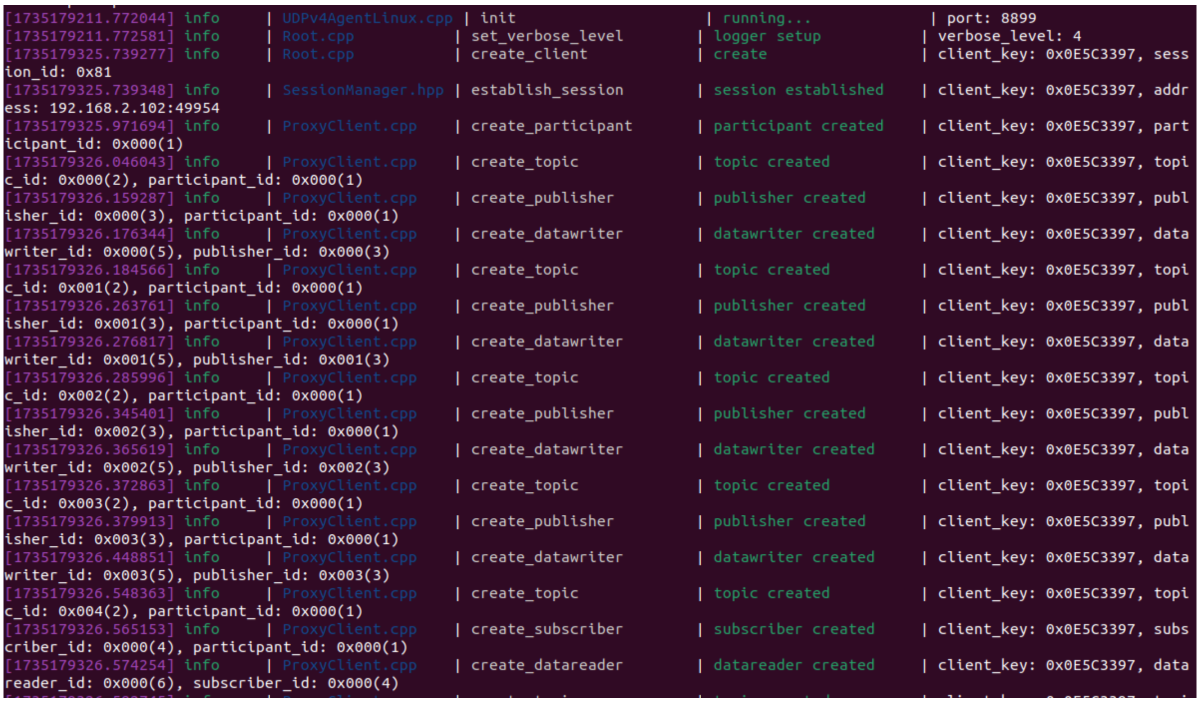
Take the matching virtual machine as an example, enter the following command to start the agent,

```
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
```



```
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 8899 -v4
[1735179211.772044] info     | UDPv4AgentLinux.cpp | init                     |
running...            | port: 8899
[1735179211.772581] 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,



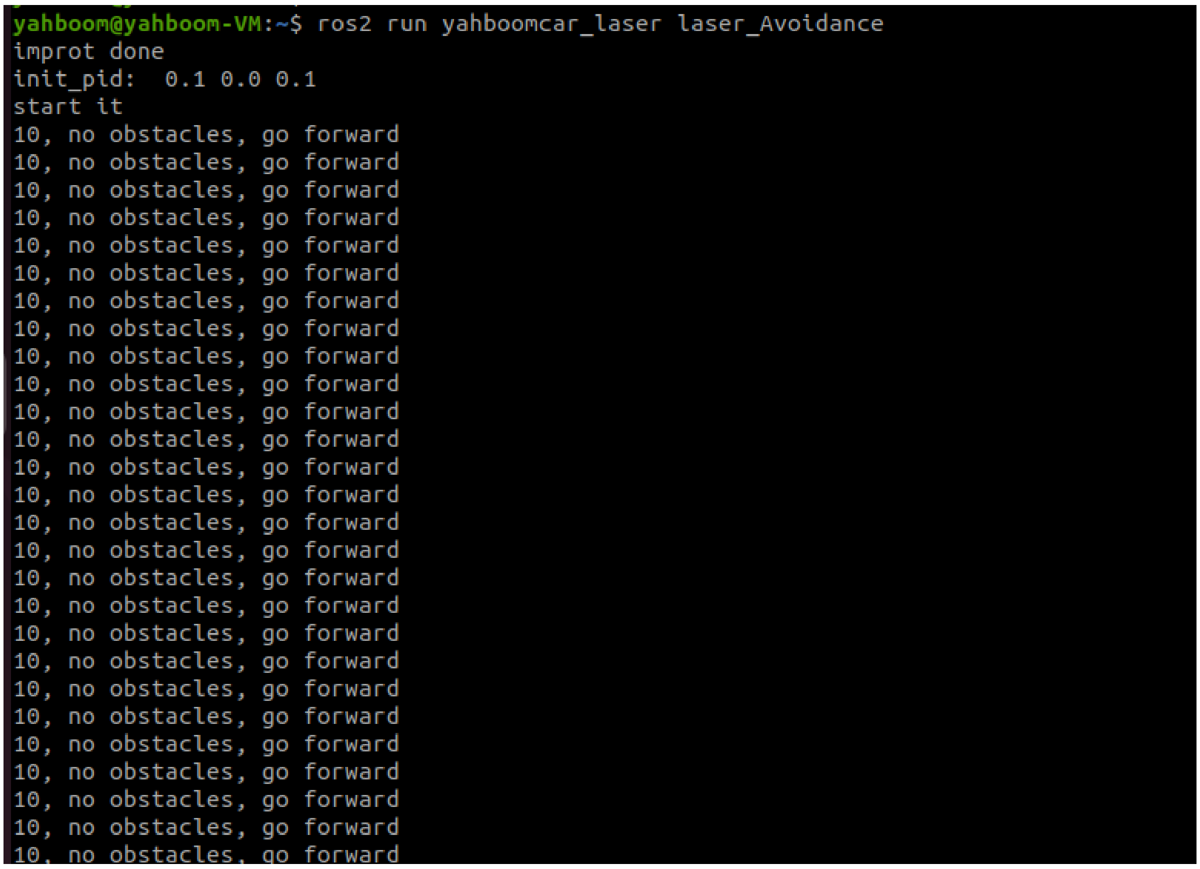
```
[1735179211.772044] info     | UDPv4AgentLinux.cpp | init                     | running...            | port: 8899
[1735179211.772581] info     | Root.cpp            | set_verbose_level        | logger setup          | verbose_level: 4
[1735179325.739277] info     | Root.cpp            | create_client            | create                | client_key: 0x0E5C3397, sess
ion_id: 0x81
[1735179325.739348] info     | SessionManager.hpp | establish_session         | session established   | client_key: 0x0E5C3397, addr
ess: 192.168.2.102:49954
[1735179325.971694] info     | ProxyClient.cpp     | create_participant       | participant created   | client_key: 0x0E5C3397, part
icipant_id: 0x000(1)
[1735179326.046043] info     | ProxyClient.cpp     | create_topic             | topic created         | client_key: 0x0E5C3397, topi
c_id: 0x000(2), participant_id: 0x000(1)
[1735179326.159287] info     | ProxyClient.cpp     | create_publisher         | publisher created     | client_key: 0x0E5C3397, publ
isher_id: 0x000(3), participant_id: 0x000(1)
[1735179326.176344] info     | ProxyClient.cpp     | create_datawriter        | datawriter created    | client_key: 0x0E5C3397, data
writer_id: 0x000(5), publisher_id: 0x000(3)
[1735179326.184566] info     | ProxyClient.cpp     | create_topic             | topic created         | client_key: 0x0E5C3397, topi
c_id: 0x001(2), participant_id: 0x000(1)
[1735179326.263761] info     | ProxyClient.cpp     | create_publisher         | publisher created     | client_key: 0x0E5C3397, publ
isher_id: 0x001(3), participant_id: 0x000(1)
[1735179326.276817] info     | ProxyClient.cpp     | create_datawriter        | datawriter created    | client_key: 0x0E5C3397, data
writer_id: 0x001(5), publisher_id: 0x001(3)
[1735179326.285996] info     | ProxyClient.cpp     | create_topic             | topic created         | client_key: 0x0E5C3397, topi
c_id: 0x002(2), participant_id: 0x000(1)
[1735179326.345401] info     | ProxyClient.cpp     | create_publisher         | publisher created     | client_key: 0x0E5C3397, publ
isher_id: 0x002(3), participant_id: 0x000(1)
[1735179326.365619] info     | ProxyClient.cpp     | create_datawriter        | datawriter created    | client_key: 0x0E5C3397, data
writer_id: 0x002(5), publisher_id: 0x002(3)
[1735179326.372863] info     | ProxyClient.cpp     | create_topic             | topic created         | client_key: 0x0E5C3397, topi
c_id: 0x003(2), participant_id: 0x000(1)
[1735179326.379913] info     | ProxyClient.cpp     | create_publisher         | publisher created     | client_key: 0x0E5C3397, publ
isher_id: 0x003(3), participant_id: 0x000(1)
[1735179326.448851] info     | ProxyClient.cpp     | create_datawriter        | datawriter created    | client_key: 0x0E5C3397, data
writer_id: 0x003(5), publisher_id: 0x003(3)
[1735179326.548363] info     | ProxyClient.cpp     | create_topic             | topic created         | client_key: 0x0E5C3397, topi
c_id: 0x004(2), participant_id: 0x000(1)
[1735179326.565153] info     | ProxyClient.cpp     | create_subscriber        | subscriber created    | client_key: 0x0E5C3397, subs
criber_id: 0x000(4), participant_id: 0x000(1)
[1735179326.574254] info     | ProxyClient.cpp     | create_datareader        | datareader created    | client_key: 0x0E5C3397, data
reader_id: 0x000(6), subscriber_id: 0x000(4)
```

3. Start the program

3.1 Run command

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

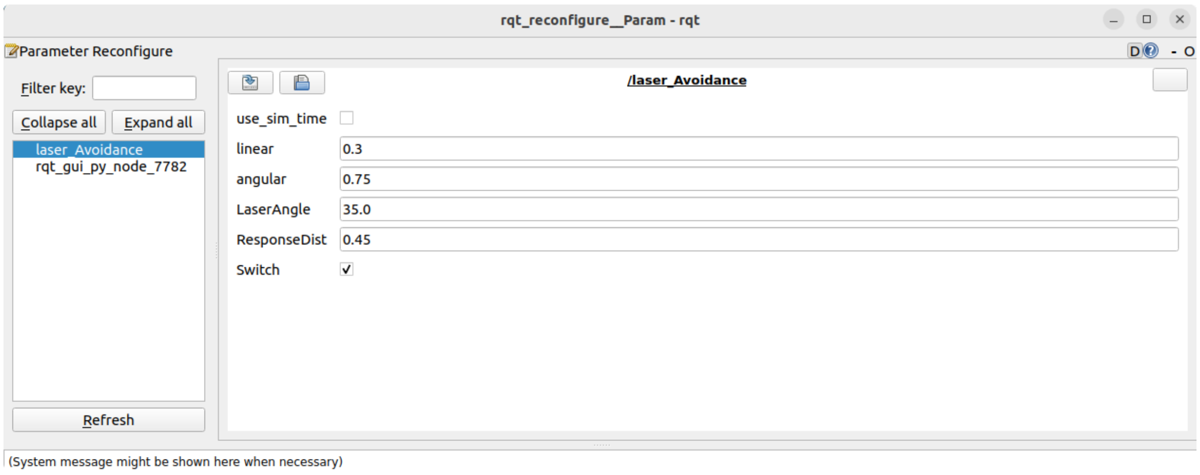
```
ros2 run yahboomcar_laser laser_Avoidance
```



```
yahboom@yahboom-VM:~$ ros2 run yahboomcar_laser laser_Avoidance
improt done
init_pid:  0.1 0.0 0.1
start it
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
10, no obstacles, go forward
```

As shown in the figure above, if the radar on the car does not detect an obstacle, it will move forward. You can set some parameters through the dynamic parameter regulator, input in the terminal,

```
ros2 run rqt_reconfigure rqt_reconfigure
```



laser_Avoidance
rqt_gui_py_node_7782

use_sim_time ☐
linear 0.3
angular 0.75
LaserAngle 35.0
ResponseDist 0.45
Switch ✓

(System message might be shown here when necessary)

Note: When you first open it, there may be no above nodes. Click Refresh to refresh and you can see all the nodes. The displayed laser_Avoidance is the node of radar obstacle avoidance.

The above parameters are explained as follows:

- linera: Linear velocity
- angular: Angular velocity
- LaserAngle: Radar detection angle
- ResponseDist: Obstacle detection distance. When the detected object is within this range, it is considered an obstacle
- 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):

laser_Avoidance, the core code is as follows,

```
#Create a radar subscriber to subscribe to radar data and remote control data and
a speed publisher to publish speed 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)

#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)
    self.Right_warning = 0
    self.Left_warning = 0
    self.front_warning = 0

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

#Determine whether there are obstacles in front, left, or right according to the
set radar detection angle and obstacle detection distance
if angle > 180: angle = angle - 360
#60 means that the range of radar detection is set to 120 degrees
if 20 < angle < self.LaserAngle:
    if ranges[i] < self.ResponseDist:
        self.Left_warning += 1
if -self.LaserAngle < angle < -20:
    if ranges[i] < self.ResponseDist:
        self.Right_warning += 1
if abs(angle) <= 20:
     if ranges[i] <= self.ResponseDist:
         self.front_warning += 1

#According to the detected obstacles, publish the speed of the car to let the car
avoid the obstacles
```

```python
if self.front_warning > 10 and self.Left_warning > 10 and self.Right_warning > 10:
    print ('1, there are obstacles in the left and right, turn right')
    twist.linear.x = self.linear
    twist.angular.z = -self.angular
    self.pub_vel.publish(twist)
    sleep(0.2)

elif self.front_warning > 10 and self.Left_warning <= 10 and self.Right_warning > 10:
    print ('2, there is an obstacle in the middle right, turn left')
    twist.linear.x = self.linear
    twist.angular.z = self.angular
    self.pub_vel.publish(twist)
    sleep(0.2)
    if self.Left_warning > 10 and self.Right_warning <= 10:
        twist.linear.x = self.linear
        twist.angular.z = -self.angular
        self.pub_vel.publish(twist)
        sleep(0.5)
.....
```