Lidar following

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.

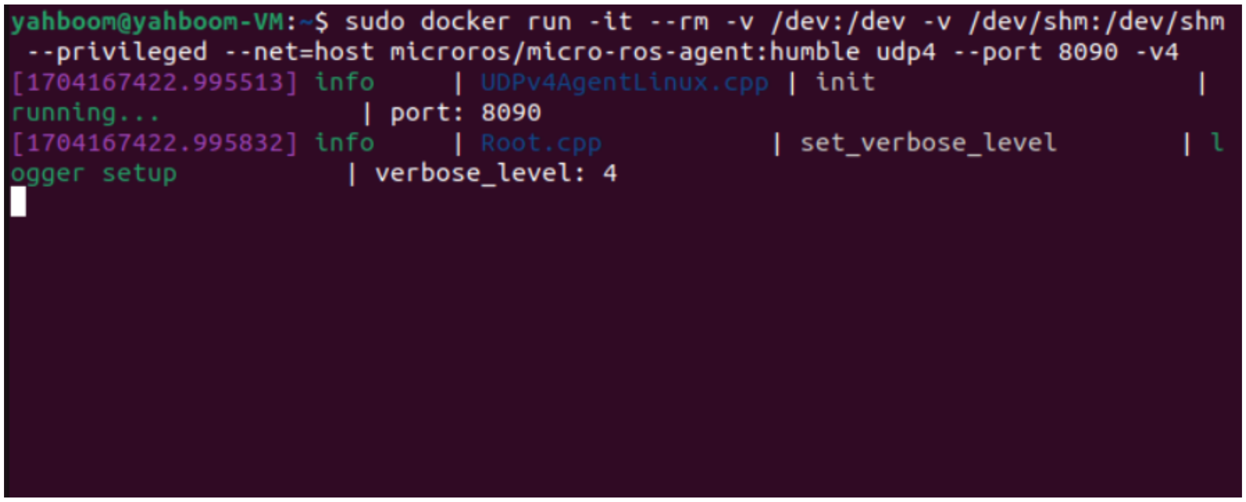
1. Program function description

The car connects to the agent and runs the program. The radar on the car scans the nearest object within the set range, and adjusts its own speed based on the set tracking distance to maintain a certain distance from the object. Parameters such as the radar detection range and obstacle avoidance detection distance can be adjusted through the dynamic parameter adjuster.

2. Start and connect to the agent

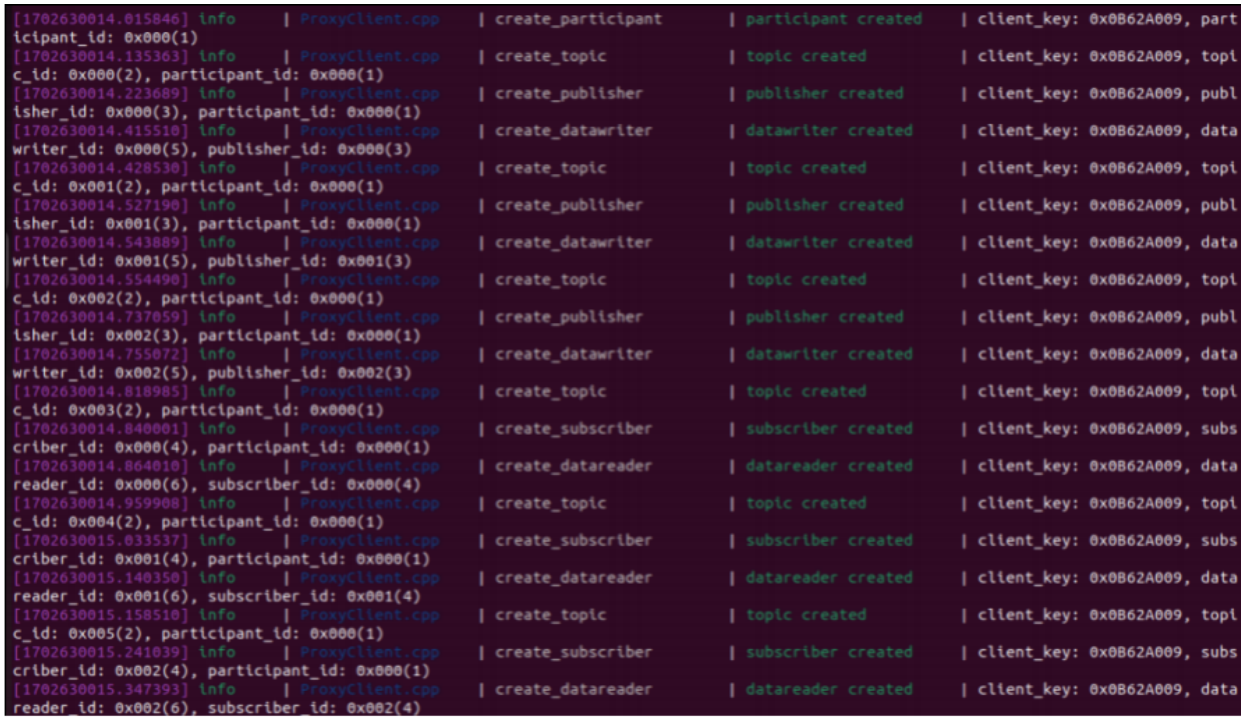
Taking the supporting virtual machine as an example, enter the following command to start the agent

```
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --net=host
microros/micro-ros-agent:humble udp4 --port 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 8090 -v4
[1704167422.995513] info     | UDPv4AgentLinux.cpp | init                     |
running...               | port: 8090
[1704167422.995832] 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.

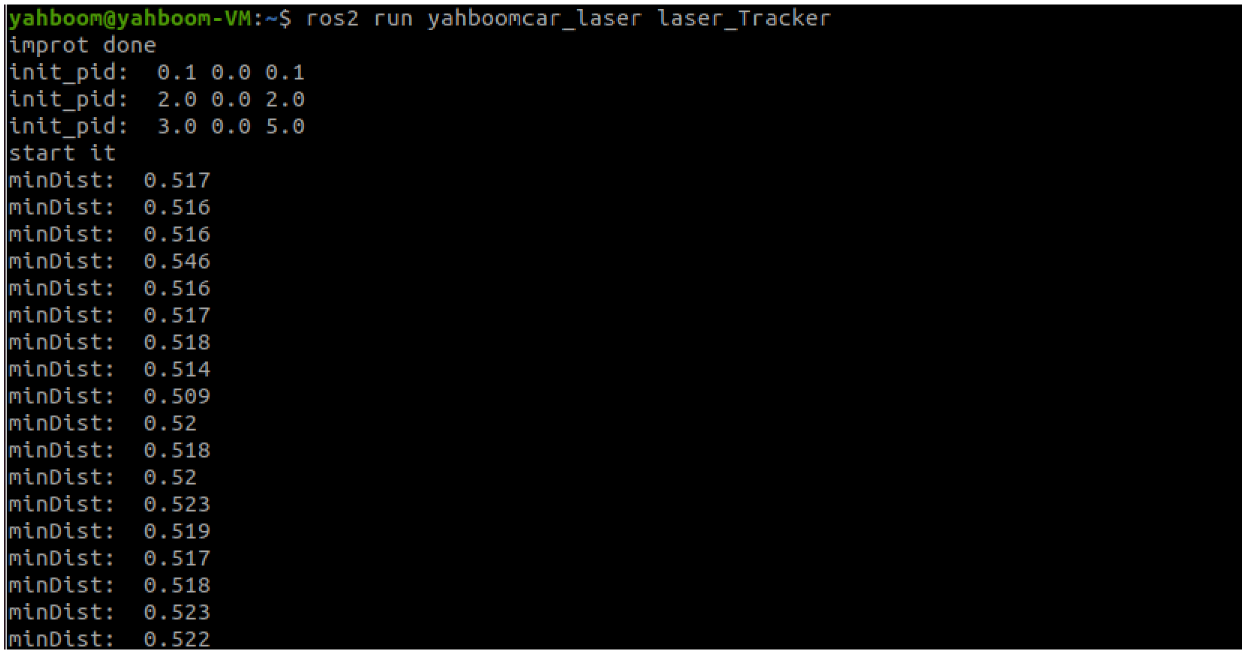


```
[1702630014.015846] info     | ProxyClient.cpp    | create_participant      | participant created      | client_key: 0x0B62A009, part
icipant_id: 0x000(1)
[1702630014.135363] info     | ProxyClient.cpp    | create_topic            | topic created            | client_key: 0x0B62A009, topi
c_id: 0x000(2), participant_id: 0x000(1)
[1702630014.223689] info     | ProxyClient.cpp    | create_publisher        | publisher created        | client_key: 0x0B62A009, publ
isher_id: 0x000(3), participant_id: 0x000(1)
[1702630014.415510] info     | ProxyClient.cpp    | create_datawriter       | datawriter created       | client_key: 0x0B62A009, data
writer_id: 0x000(5), publisher_id: 0x000(3)
[1702630014.428530] info     | ProxyClient.cpp    | create_topic            | topic created            | client_key: 0x0B62A009, topi
c_id: 0x001(2), participant_id: 0x000(1)
[1702630014.527190] info     | ProxyClient.cpp    | create_publisher        | publisher created        | client_key: 0x0B62A009, publ
isher_id: 0x001(3), participant_id: 0x000(1)
[1702630014.543889] info     | ProxyClient.cpp    | create_datawriter       | datawriter created       | client_key: 0x0B62A009, data
writer_id: 0x001(5), publisher_id: 0x001(3)
[1702630014.554490] info     | ProxyClient.cpp    | create_topic            | topic created            | client_key: 0x0B62A009, topi
c_id: 0x002(2), participant_id: 0x000(1)
[1702630014.737059] info     | ProxyClient.cpp    | create_publisher        | publisher created        | client_key: 0x0B62A009, publ
isher_id: 0x002(3), participant_id: 0x000(1)
[1702630014.755072] info     | ProxyClient.cpp    | create_datawriter       | datawriter created       | client_key: 0x0B62A009, data
writer_id: 0x002(5), publisher_id: 0x002(3)
[1702630014.818985] info     | ProxyClient.cpp    | create_topic            | topic created            | client_key: 0x0B62A009, topi
c_id: 0x003(2), participant_id: 0x000(1)
[1702630014.840001] info     | ProxyClient.cpp    | create_subscriber       | subscriber created       | client_key: 0x0B62A009, subs
criber_id: 0x000(4), participant_id: 0x000(1)
[1702630014.864010] info     | ProxyClient.cpp    | create_datareader       | datareader created       | client_key: 0x0B62A009, data
reader_id: 0x000(6), subscriber_id: 0x000(4)
[1702630014.959908] info     | ProxyClient.cpp    | create_topic            | topic created            | client_key: 0x0B62A009, topi
c_id: 0x004(2), participant_id: 0x000(1)
[1702630015.033537] info     | ProxyClient.cpp    | create_subscriber       | subscriber created       | client_key: 0x0B62A009, subs
criber_id: 0x001(4), participant_id: 0x000(1)
[1702630015.140350] info     | ProxyClient.cpp    | create_datareader       | datareader created       | client_key: 0x0B62A009, data
reader_id: 0x001(6), subscriber_id: 0x001(4)
[1702630015.158510] info     | ProxyClient.cpp    | create_topic            | topic created            | client_key: 0x0B62A009, topi
c_id: 0x005(2), participant_id: 0x000(1)
[1702630015.241039] info     | ProxyClient.cpp    | create_subscriber       | subscriber created       | client_key: 0x0B62A009, subs
criber_id: 0x002(4), participant_id: 0x000(1)
[1702630015.347393] info     | ProxyClient.cpp    | create_datareader       | datareader created       | client_key: 0x0B62A009, data
reader_id: 0x002(6), subscriber_id: 0x002(4)
```

3. Start the program

Enter the following command in the terminal to start:

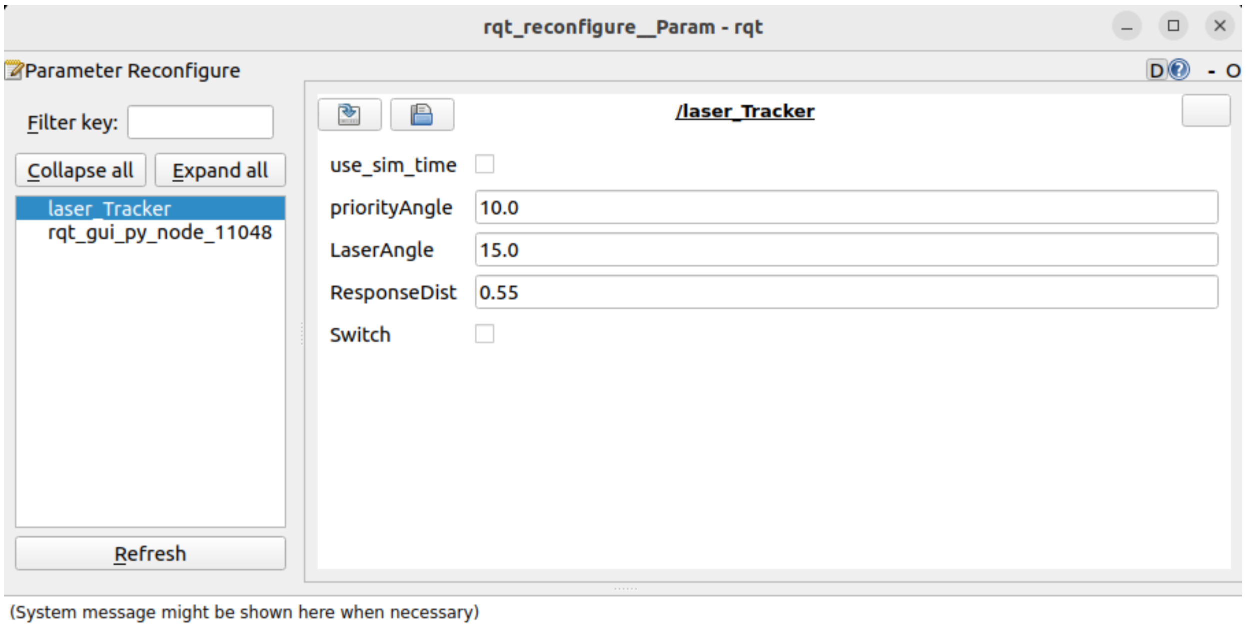
```
ros2 run yahboomcar_laser laser_Tracker
```



```
yahboom@yahboom-VM:~$ ros2 run yahboomcar_laser laser_Tracker
improt done
init_pid:  0.1 0.0 0.1
init_pid:  2.0 0.0 2.0
init_pid:  3.0 0.0 5.0
start it
minDist:  0.517
minDist:  0.516
minDist:  0.516
minDist:  0.546
minDist:  0.516
minDist:  0.517
minDist:  0.518
minDist:  0.514
minDist:  0.509
minDist:  0.52
minDist:  0.518
minDist:  0.52
minDist:  0.523
minDist:  0.519
minDist:  0.517
minDist:  0.518
minDist:  0.523
minDist:  0.522
```

After the program is started, it will search for the nearest object within the radar scanning range and maintain a set distance from it. Move the tracked object slowly, and the car will track the movement of the object. You can set some parameters through the dynamic parameter adjuster. Enter the following command at the terminal:

```
ros2 run rqt_reconfigure rqt_reconfigure
```


rqt_reconfigure__Param - rqt

Parameter Reconfigure

Filter key:

Collapse all · Expand all

laser_Tracker
rqt_gui_py_node_11048

Refresh

/laser_Tracker

use_sim_time ☐
priorityAngle 10.0
LaserAngle 15.0
ResponseDist 0.55
Switch ☐

(System message might be shown here when necessary)

Note: There may not be the above nodes when you first open it. You can see all nodes after clicking Refresh. The displayed laser_Tracker is the node tracked by the radar.

The above parameter description：

- priorityAngle： Radar priority detection angle
- LaserAngle： Radar detection angle
- ResponseDist： tracking distance
- Switch： Game switch

After modifying the above parameters, you need to click on the blank space to transfer 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_Tracker， 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
self.sub_laser = self.create_subscription(LaserScan,"/scan",self.registerScan,1)
self.sub_JoyState = self.create_subscription(Bool,'/JoyState',
self.JoyStateCallback,1)
self.pub_vel = self.create_publisher(Twist,'/cmd_vel',1)
```

```
#Radar callback function: Process the subscribed radar data. There is a
priorityAngle here, which indicates the range of priority radar detection. If there
are objects within this range, priority will be selected for tracking. The set angle
is 10 degrees.
ranges = np.array(scan_data.ranges)
for i in range(len(ranges)):
    angle = (scan_data.angle_min + scan_data.angle_increment * i) * RAD2DEG
if abs(angle) < self.priorityAngle:
    if 0 < ranges[i] < (self.ResponseDist + offset):
        frontDistList.append(ranges[i])
        frontDistIDList.append(angle)
    elif abs(angle) < self.LaserAngle and ranges[i] > 0:
        minDistList.append(ranges[i])
        minDistIDList.append(angle)
#Find the nearest object minDistID
if len(frontDistIDList) != 0:
    minDist = min(frontDistList)
    minDistID = frontDistIDList[frontDistList.index(minDist)]
else:
    minDist = min(minDistList)
    minDistID = minDistIDList[minDistList.index(minDist)]
#According to the object that needs to be tracked, calculate the angular velocity
and linear velocity, and then publish the velocity data
if abs(minDist - self.ResponseDist) < 0.1: minDist = self.ResponseDist
velocity.linear.x = -self.lin_pid.pid_compute(self.ResponseDist, minDist)
ang_pid_compute = self.ang_pid.pid_compute(minDistID/48, 0)
if minDistID > 0: velocity.angular.z = ang_pid_compute
else: velocity.angular.z = ang_pid_compute
velocity.angular.z = ang_pid_compute
if abs(ang_pid_compute) < 0.1: velocity.angular.z = 0.0
self.pub_vel.publish(velocity)
```