Radar tracking fun gameplay

1. Program function description

After the program is started, the radar scans the nearest object, then locks on it, the object moves, and the mechanical dog moves with it. If the handle node is activated.

2. Program code reference path

The source code of this function is located at,

```
#pi4version
/home/pi/cartographer_ws2/src/yahboom_laser/yahboom_laser/laser_Tracker_xgo_RS200
.py

#pi5version
/root/yahboomcar_ws/src/yahboom_laser/yahboom_laser/laser_Tracker_xgo_RS200.py
```

3. Program startup

3.1. Start command

PI4 version driver:

The mechanical dog chassis and radar have been set to start automatically at boot. If you find that they have not started, please enter in the terminal.

```
sudo systemctl restart YahboomStart.service
```

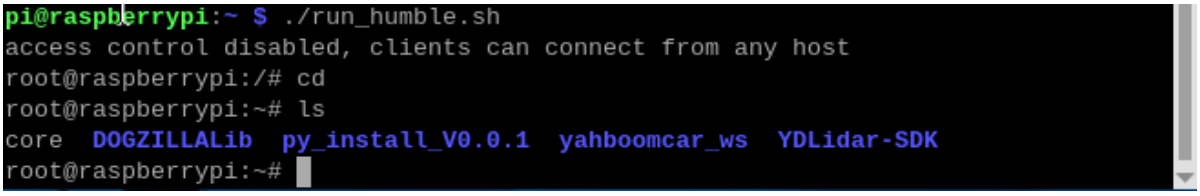
If the radar and chassis are started, you need to enter in the terminal:

```
cd /home/pi/cartographer_ws2
source install/setup.bash
#Start radar tracking program Radar MS200
ros2 run yahboom_laser laser_Tracke_RS200
```

PI5 version driver:

Open a terminal in the root directory of the Raspberry Pi and enter the following command to enter docker:

```
./run_humble.sh
```



Then enter the following command in the docker terminal to start the radar data:

```
ros2 launch bringup Navigation_bringup.launch.py
```

Then enter the command to run the obstacle avoidance program, ** (PI5 version needs to enter the same docker to run, please refer to "Introduction and Use of Radar" for steps)**

```
#Start radar obstacle avoidance program Radar MS200
ros2 run yahboom_laser laser_Tracke_RS200
```

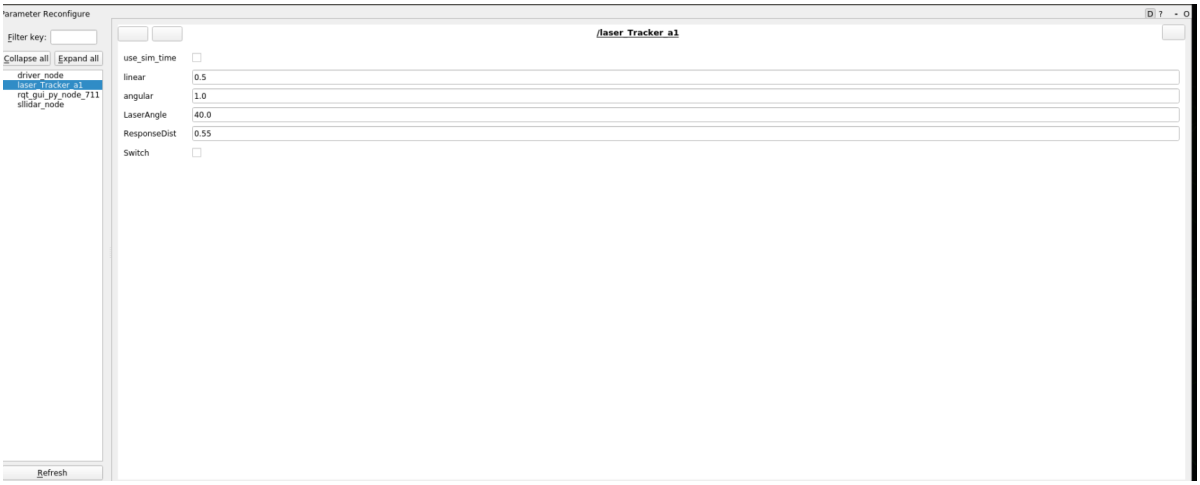
3.2. View the topic communication node diagram

Terminal input, ** (PI5 version needs to run in the same docker, please refer to "Introduction and Use of Radar" for steps)**

```
ros2 run rqt_graph rqt_graph
```

You can also set the parameter size through the dynamic parameter adjuster, terminal input, ** (PI5 version needs to run in the same docker, please refer to "Introduction and Use of Radar" for steps)**

```
ros2 run rqt_reconfigure rqt_reconfigure
```



The meaning of each parameter is as follows:

参数名称	参数含义
linear	线速度
angular	角速度
LaserAngle	雷达检测角度
ResponseDist	障碍物检测距离
Switch	玩法开关

The above parameters are all adjustable. Except for Switch, the other four need to be set to decimals. After modification, click on the blank space to write.

4. Core code

Mainly look at the radar callback function. Here is an explanation of how to obtain the obstacle distance information at each angle, then find the nearest point, then judge the distance, then calculate the speed data, and finally publish it.

```
angle = (scan_data.angle_min + scan_data.angle_increment * i) * RAD2DEG
if abs(angle) > (180 - self.priorityAngle): #priorityAngle is the car's priority
following range
     if ranges[i] < (self.ResponseDist + offset):
         frontDistList.append(ranges[i])
         frontDistIDList.append(angle)
     elif (180 - self.LaserAngle) < angle < (180 - self.priorityAngle):
         minDistList.append(ranges[i])
         minDistIDList.append(angle)
     elif (self.priorityAngle - 180) < angle < (self.LaserAngle - 180):
         minDistList.append(ranges[i])
         minDistIDList.append(angle)
         if len(frontDistIDList) != 0:
             minDist = min(frontDistList)
             minDistID = frontDistIDList[frontDistList.index(minDist)]
         else:
             minDist = min(minDistList)
             minDistID = minDistIDList[minDistList.index(minDist)]#Calculate the
ID of the minimum distance point
             if self.Joy_active or self.Switch == True:
                 if self.Moving == True:
                     self.pub_vel.publish(Twist())
                     self.Moving = not self.Moving
                     return
                 self.Moving = True
                 velocity = Twist()
                 if abs(minDist - self.ResponseDist) < 0.1: minDist =
self.ResponseDist#Judge the distance from the smallest point
                 velocity.linear.x = -self.lin_pid.pid_compute(self.ResponseDist,
minDist)#Calculate linear velocity
                 ang_pid_compute = self.ang_pid.pid_compute((180 -
abs(minDistID)) / 72, 0)#Calculate angular velocity
if minDistID > 0: velocity.angular.z = -ang_pid_compute
else: velocity.angular.z = ang_pid_compute
```

```
if ang_pid_compute < 0.02: velocity.angular.z = 0.0
self.pub_vel.publish(velocity)
```