4、Radar Guard fun gameplay

1、Program function description

After the program starts, the trolley will track the target of the nearest point, when the target point moves laterally, it will move with the target point, when the target point is close to the trolley, less than the set distance, the buzzer will beep the alarm until the target point is far away from the distance set by the trolley.

2、Program code reference path

After entering the docker container, the location of the source code of this function is located at,

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_laser/yahboomcar_laser/laser_Warning_a1_X3.py
```

The A1 radar has the same architecture as the S2 radar and can be shared.

3、The program starts

3.1、start the command

After entering the docker container, according to the actual model and radar model, the terminal input,

```
#Start the trolley chassis
ros2 run yahboomcar_bringup Mcnamu_driver_X3
#Start the A1 radar
ros2 launch sllidar_ros2 sllidar_launch.py
#Start the S2 radar
ros2 launch sllidar_ros2 sllidar_s2_launch.py
#Start the radar obstacle avoidance program X3 model, A1/S2 radar
ros2 run yahboomcar_laser laser_Warning_a1_X3
#Start the handle, if needed
ros2 run yahboomcar_ctrl yahboom_joy_X3
ros2 run joy joy_node
```

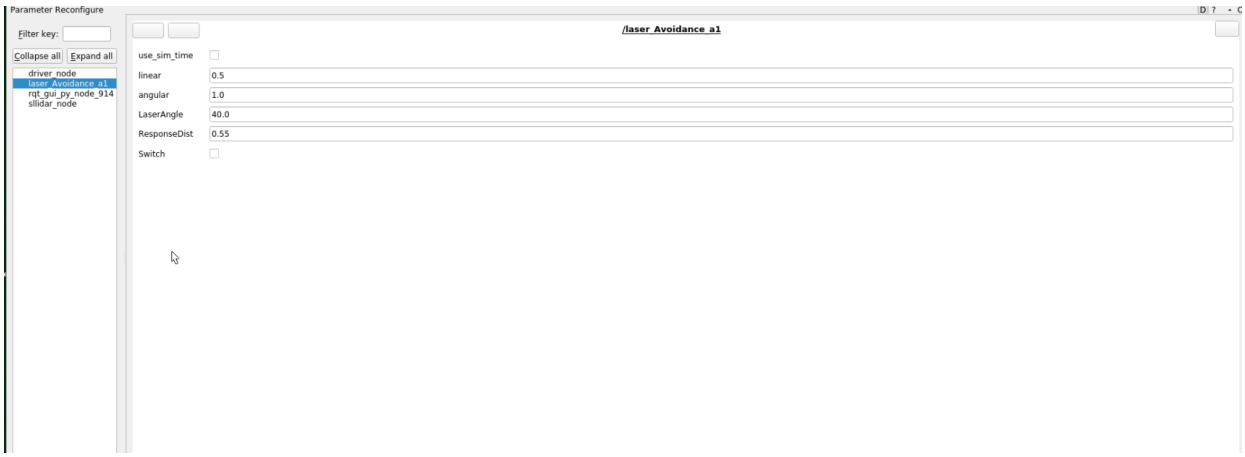
3.2、View the topic communication node diagram

docker terminal input,

```
ros2 run rqt_graph rqt_graph
```

It is also possible to set the size of parameters, terminal input, and terminal input through the dynamic parameter adjuster,

```
ros2 run rqt_reconfigure rqt_reconfigure
```



The meaning of each parameter is as follows,

Parameter name	Parameter meaning
linear	Line speed
angular	Angular velocity
LaserAngle	Radar detection angle
ResponseDist	Obstacle detection distance
Switch	Gameplay switch

4、Core source code analysis

Taking the X3 model, the source code of the A1 radar as an example, mainly looking at the callback function of the radar, here explains how to obtain the obstacle distance information of each angle, then obtain the ID of the minimum distance, calculate the size of the angular velocity, and then compare the minimum distance with the set distance, if it is less than the set distance, let the buzzer sound, and finally release the speed topic data.

```
for i in range(len(ranges)):
    angle = (scan_data.angle_min + scan_data.angle_increment * i) * RAD2DEG
    if abs(angle) > (180 - self.LaserAngle):
        minDistList.append(ranges[i])
        minDistIDList.append(angle)
        if len(minDistList) == 0: return
    minDist = min(minDistList)
    minDistID = minDistIDList[minDistList.index(minDist)]
    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
    if minDist <= self.ResponseDist:
        if self.Buzzer_state == False:
            b = Bool()
            b.data = True
            self.pub_Buzzer.publish(b)
            self.Buzzer_state = True
        else:
            if self.Buzzer_state == True:
                self.pub_Buzzer.publish(Bool())
                self.Buzzer_state = False
    velocity = Twist()
    ang_pid_compute = self.ang_pid.pid_compute((180 - abs(minDistID)) / 36, 0)
    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)
```