3. Radar guard

3. Radar guard
    3.1. How to use
    3.2. Source code analysis

Radar Guard gameplay introduction:

- Set lidar detection angle and response distance.
- After turning on the car, the car faces the target closest to the car.
- When the distance between the target and the car is less than the response distance, the buzzer will sound until there is no target within the response distance.
- The angular speed PID of the car can be adjusted to achieve the best rotation effect of the car.

3.1. How to use

Note: The [R2] of the remote control handle has the [pause/start] function of this gameplay. Due to the problem of movement method, the case in this section does not support the Ackerman model. Different models will have different parameter ranges, but the principle is the same; take the X3 McLunner as an example.

```
#Raspberry Pi 5 master needs to enter docker first, please perform this step
#If running the script into docker fails, please refer to ROS/07, Docker tutorial
~/run_docker.sh
```

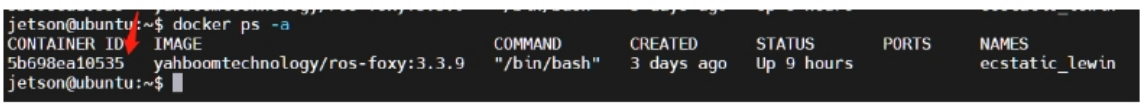
Start with one click and the car will start moving after executing the command.

```
roslaunch yahboomcar_laser laser_Warning.launch
```

<PI5 needs to open another terminal to enter the same docker container

1. In the above steps, a docker container has been opened. You can open another terminal on the host (car) to view:

```
docker ps -a
```



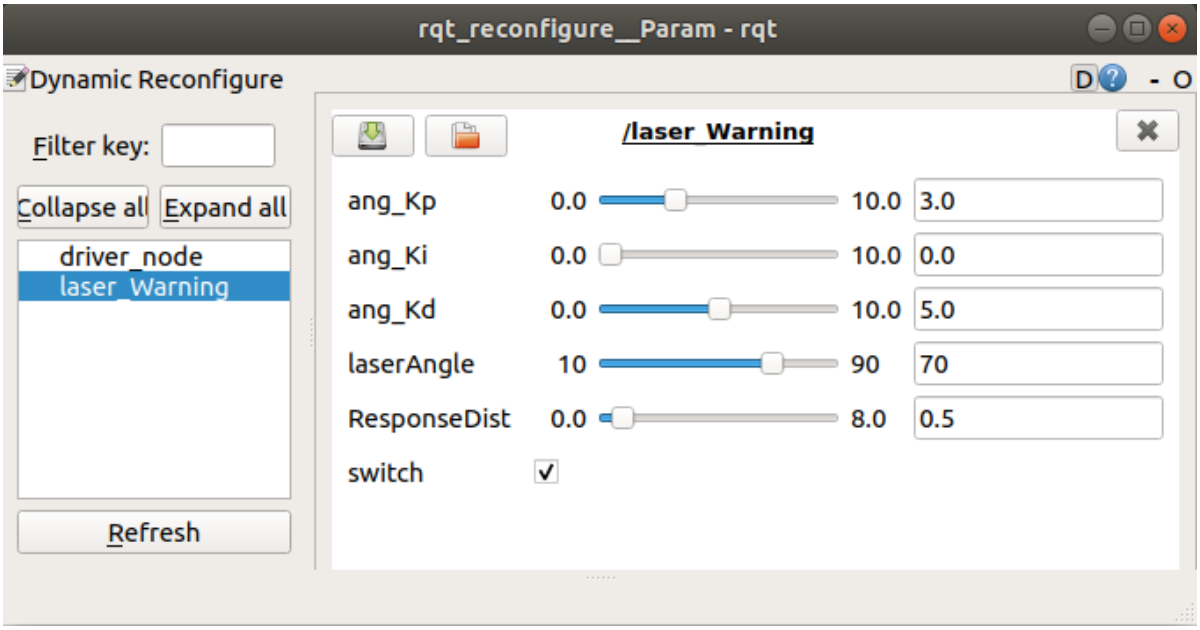
2. Now enter the docker container in the newly opened terminal:

```
docker exec -it 5b698ea10535 /bin/bash
```

After successfully entering the container, you can open countless terminals to enter the container.

Dynamic debugging parameters

```
rosrun rqt_reconfigure rqt_reconfigure
```



Parameter analysis:

Parameters	Range	Parse
[LaserAngle]	[10, 90]	Lidar detection angle (left and right angles)
[ResponseDist]	[0.0, 8.0]	Car response distance
[switch]	[False, True]	Car movement [Start/Pause]

[ang_Kp], [ang_Ki], [ang_Kd]: PID debugging of car angular speed.

In the box in front of [switch], click the value of [switch] to True, and the car will stop. [switch] Default is False, the car moves.

- Parameter modification

When the parameters are adjusted to the optimal state, the corresponding parameters are modified into the file, and no adjustment is required when using again.

According to the optimal parameters of the [rqt_reconfigure] debugging tool, enter the [scripts] folder of the [yahboomcar_laser] function package and modify the parameters corresponding to the [laser_Warning.py] file, as shown below

```
class laserWarning:
    def __init__(self):
        rospy.on_shutdown(self.cancel)
        ... ...
        self.ang_pid = SinglePID(3.0, 0.0, 5.0)
        Server(laserWarningPIDConfig, self.dynamic_reconfigure_callback)
        self.laserAngle = 70
        self.ResponseDist = 0.5
```

[rqt_reconfigure] Modification of the initial value of the debugging tool

```
gen.add("ang_Kp", double_t, 0, "Kp in PID", 3.0, 0, 10)
gen.add("ang_Ki", double_t, 0, "Ki in PID", 0.0, 0, 10)
gen.add("ang_Kd", double_t, 0, "Kd in PID", 5.0, 0, 10)
gen.add("laserAngle", int_t, 0, "laserAngle", 70, 10, 90)
gen.add("ResponseDist", double_t, 0, "ResponseDist", 0.5, 0, 8)
gen.add("switch", bool_t, 0, "switch in rosbot", False)
```

Enter the [cfg] folder of the [yahboomcar_laser] function package and modify the initial values of the parameters corresponding to the [laserWarningPID.cfg] file.

```
gen.add("linear", double_t, 0, "linear in robot", 0.5, 0, 1.0)
```

Take the above article as an example to analyze

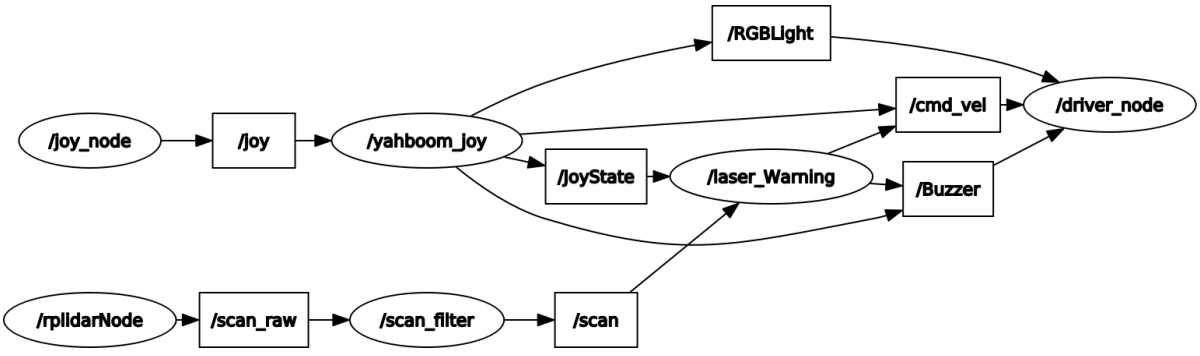
Parameters	Analysis	Corresponding parameters
name	The name of the parameter	"linear"
type	parameter data type	double_t
level	a bitmask passed to the callback	0
description	A description parameter	"linear in robot"
default	Initial value for node startup	0.5
min	parameter minimum value	0
max	parameter maximum value	1.0

Note: After modification, you must recompile and update the environment to be effective.

```
cd ~/yahboomcar_ws
catkin_make
source devel/setup.bash
```

Node view

```
rqt_graph
```



3.2. Source code analysis

launch file

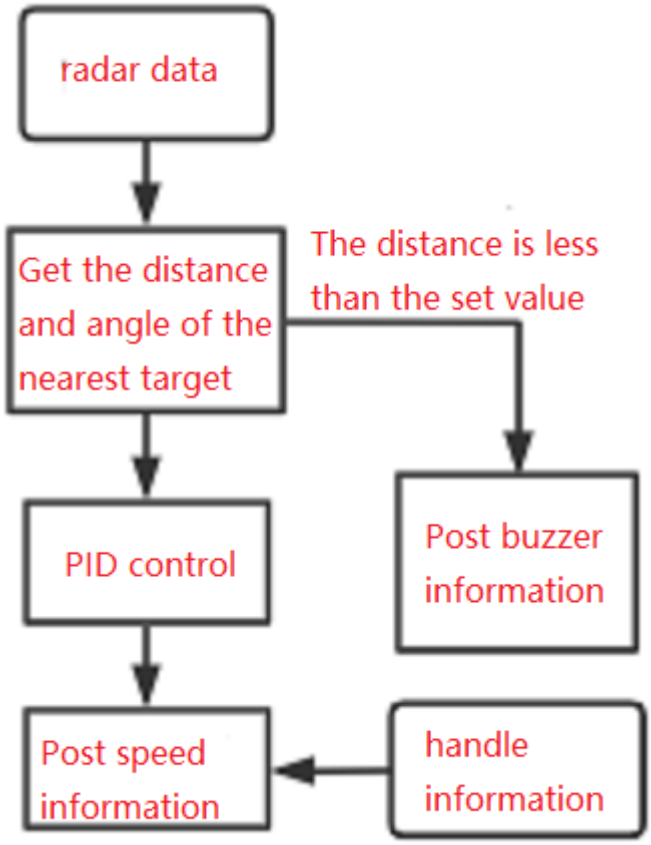
-base.launch

```
<launch>
     <!-- Start lidar node -->
     <!-- Activate the lidar node -->
     <include file="$(find rplidar_ros)/launch/rplidar.launch"/>
     <!-- Start the car chassis drive node-->
     <!-- Start the car chassis drive node -->
     <include file="$(find yahboomcar_bringup)/launch/yahboomcar.launch"/>
     <!-- Handle control node -->
     <!-- Handle control node -->
     <include file="$(find yahboomcar_ctrl)/launch/yahboom_joy.launch"/>
</launch>
```

- laser_Warning.launch

```
<launch>
     <!-- Launch base.launch file -->
     <!-- Launch the base.launch file -->
     <include file="$(find yahboomcar_laser)/launch/base.launch"/>
     <!-- Start the lidar guard node -->
     <!-- Activate the Lidar guard node -->
     <node name='laser_Warning' pkg="yahboomcar_laser" type="laser_Warning.py"
required="true" output="screen"/>
</launch>
```

laser_Warning.py source code flow chart:



According to the position of the target, the car automatically rotates to face the target; when the target approaches a certain distance, the buzzer alarms.