

Lidar guard

1. Program function description

After the program is started, the car will track the target at the nearest point. When the target point moves laterally, it will move with the target point. When the target point is close to the car and is less than the set distance, there will be a voice reminder until the target point is far away. The distance set by the car.

2. Program code reference path

The source code of this function is located at,

```
#pi4version
/home/pi/cartographer_ws2/src/yahboom_laser/laser_warning_xgo_RS200.py

#pi5version
/root/yahboomcar_ws/src/yahboom_laser/yahboom_laser/laser_warning_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
```

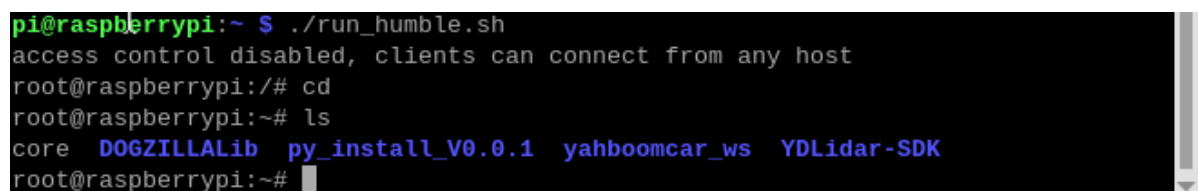
If the radar and chassis are started, you need to enter in the terminal:

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



```
pi@raspberrypi:~ $ ./run_humble.sh
access control disabled, clients can connect from any host
root@raspberrypi:~# cd
root@raspberrypi:~# ls
core DOGZILLALib py_install_V0.0.1 yahboomcar_ws YDLidar-SDK
root@raspberrypi:~#
```

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

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

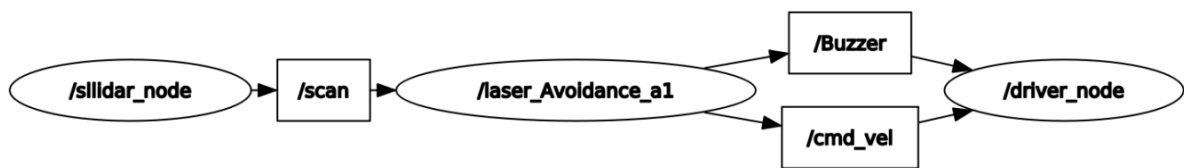
Then enter the command to run the obstacle avoidance program, ** (PI5 version needs to enter the same docker to run, please refer to "Introduction and Use of Radar" for steps)**

```
#Start radar guard program radar MS200
ros2 run yahboom_laser laser_warning_RS200
```

3.2. View the topic communication node diagram

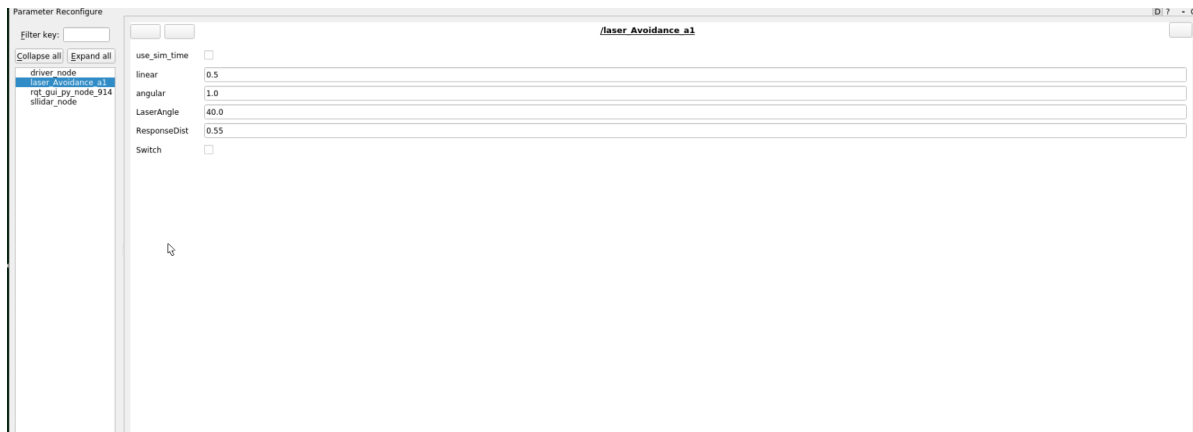
Terminal input, ** (PI5 version needs to run in the same docker, please refer to "Introduction and Use of Radar" for steps)**

```
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:

| Parameter Name | Parameter Means |
|----------------|-----------------------|
| linear | Linear velocity |
| angular | Linear velocity |
| LaserAngle | Angular velocity |
| ResponseDist | Lidar detection angle |
| Switch | Function switch |

4. Core source code analysis

Mainly look at the radar callback function. Here is an explanation of how to obtain the obstacle distance information at each angle, then obtain the ID of the minimum distance, calculate the angular velocity, and then compare the minimum distance with the set distance. If it is less than the set distance distance, the buzzer will sound, and finally the speed topic data will be released.

```
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)
```