2. Lidar obstacle avoidance

2. Lidar obstacle avoidance
- 2.1. Program function description
- 2.2, Program code reference path
- 2.3, Program startup
  - 2.3.1, Car startup command
  - 2.3.2, Virtual machine end parameter modification
  - 2.3.3, view the topic communication node graph
- 2.4, core source code analysis

2.1. Program function description

After the program is started, the car will move forward. When an obstacle appears within the detection range, it will adjust its posture, avoid the obstacle, and then continue to move forward.

Click [Switch] after starting the dynamic parameter regulator on the virtual machine to turn on/pause this function.

In addition, the [L1] button of the handle can lock/open the motion control of the car. When the motion control is turned on, the function will be locked; when the motion control is locked, the function can be turned on.

2.2, Program code reference path

After SSH connects to the car, the location of the function source code is,

```
#python file
/home/sunrise/yahboomcar_ws/src/yahboomcar_laser/yahboomcar_laser/laser_avoidanc
e.py
#launch file
/home/sunrise/yahboomcar_ws/src/yahboomcar_laser/launch/laser_avoidance_launch.p
y
```

2.3, Program startup

2.3.1, Car startup command

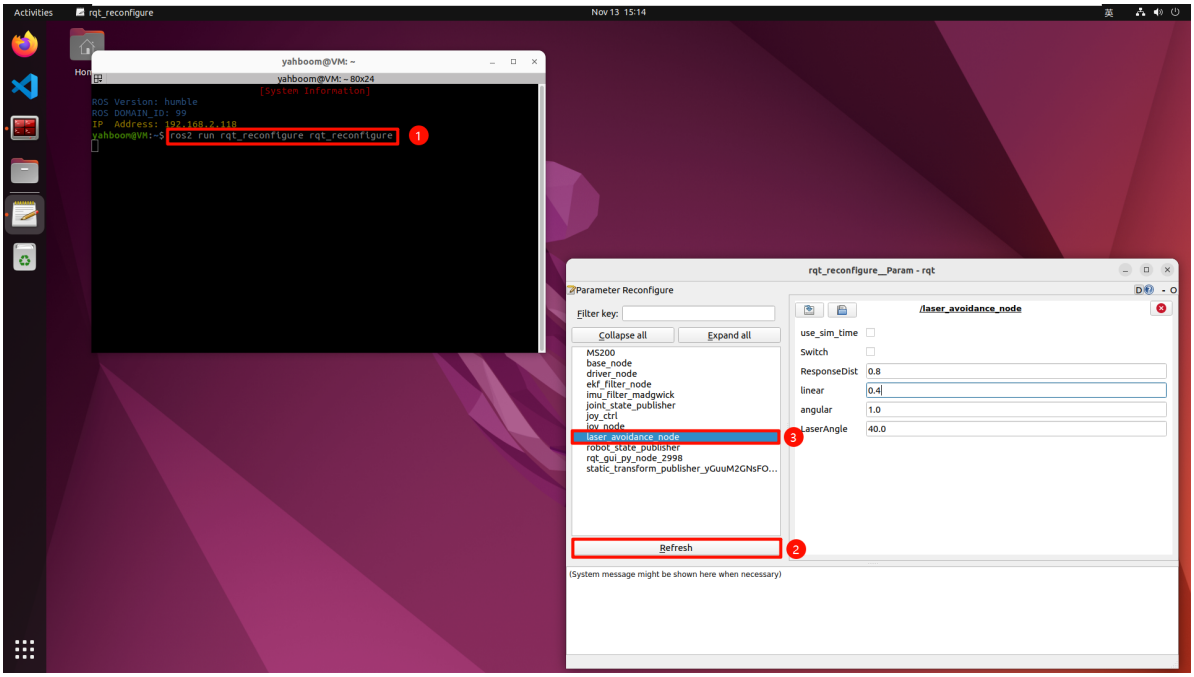
After SSH connects to the car, enter in the terminal,

```
ros2 launch yahboomcar_laser laser_avoidance_launch.py
```

2.3.2, Virtual machine end parameter modification

Open the dynamic parameter adjuster on the virtual machine end, open the terminal and enter,

```
ros2 run rqt_reconfigure rqt_reconfigure
```



The meaning of each parameter is as follows,

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

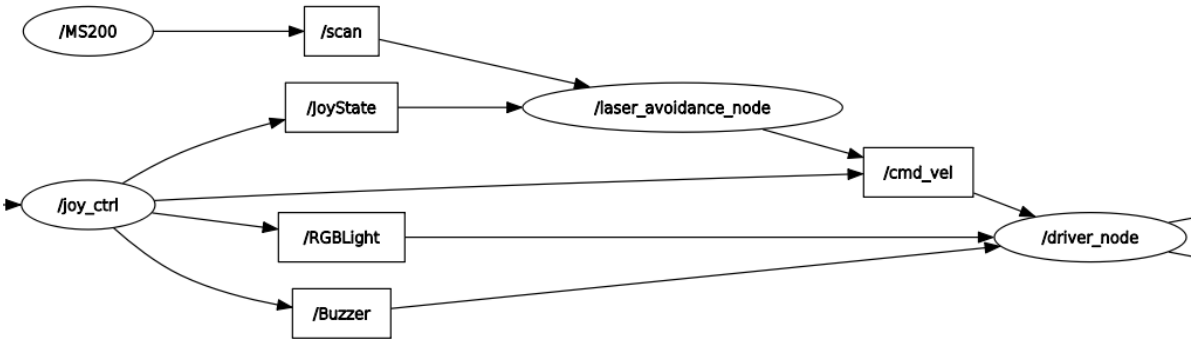
The above parameters can be adjusted. Check or uncheck [Switch] to turn on or pause the radar obstacle avoidance function.

The other four need to be set in decimals. After modification, press the Enter key or click on the blank space to write.

2.3.3, view the topic communication node graph

Virtual machine terminal input,

```
ros2 run rqt_graph rqt_graph
```



2.4, core source code analysis

Mainly look at the callback function of the radar, which explains how to obtain the obstacle distance information at each angle,

```
def registerScan(self, scan_data):
if not isinstance(scan_data, LaserScan): return
self.right_warning = 0
self.left_warning = 0
self.front_warning = 0

ranges = np.array(scan_data.ranges)
for i in range(len(ranges)):
if ranges[i] < self.ResponseDist: #range[i] is the result of the radar scan,
which refers to the distance information
angle = (scan_data.angle_min + scan_data.angle_increment * i) * 180 / pi #The
angle of the radar information is in radians, so it needs to be converted to
angles for calculation here.
if angle > 180: angle = angle - 360 #angle is used to set the judgment range
based on the structure of the radar.
if -self.LaserAngle < angle < -20:
self.right_warning += 1
elif abs(angle) <= 20:
self.front_warning += 1
elif 20 < angle < self.LaserAngle:
self.left_warning += 1
if self.Joy_active == True or self.Switch == False:
if self.Moving == True:
self.pub_vel.publish(Twist())
self.Moving = not self.Moving
return
self.Moving = True
```