Lidar guard

4.1. Program Function Description

After the program is started, the car will track the nearest target point. When the target point moves horizontally, it will move with the target point. When the target point is close to the car and less than the set distance, the buzzer will beep until the target point is away from the car by the set distance.

Start the dynamic parameter regulator on the virtual machine and click [Switch] 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.

4.2, Program code reference path

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

```
#python file
/home/sunrise/yahboomcar_ws/src/yahboomcar_laser/yahboomcar_laser/laser_warning.
py
#launch file
/home/sunrise/yahboomcar_ws/src/yahboomcar_laser/launch/laser_warning_launch.py
```

4.3, Program startup

4.3.1, Start command

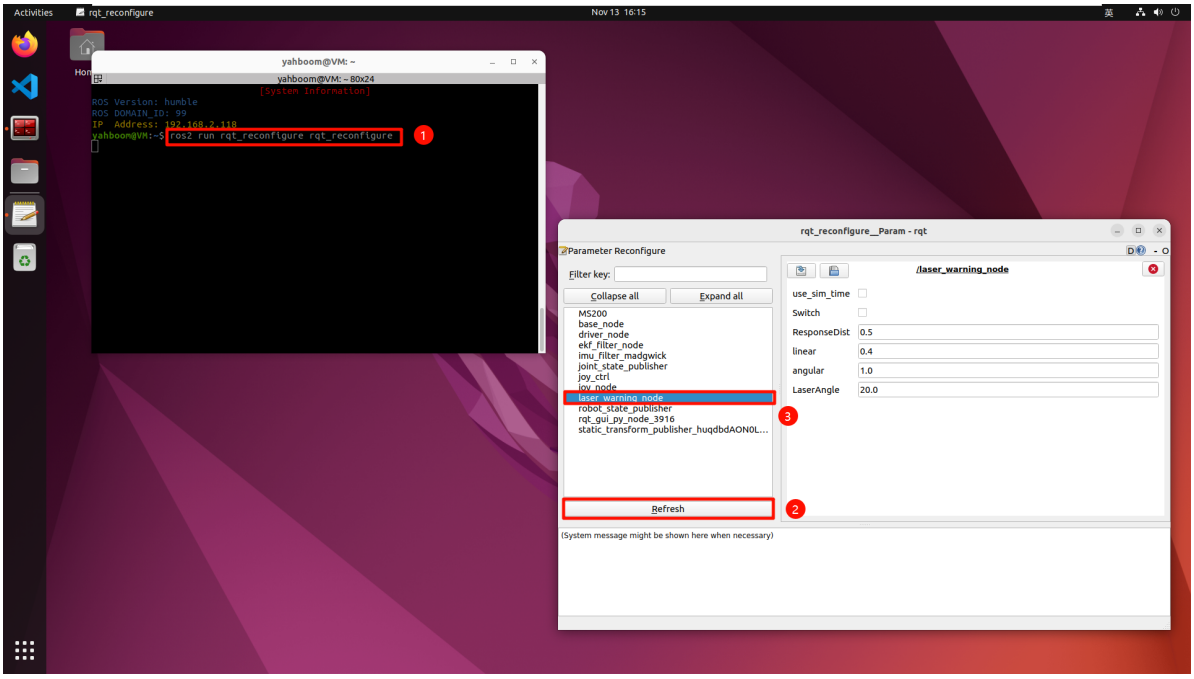
After SSH connects to the car, enter in the terminal,

```
ros2 launch yahboomcar_laser laser_warning_launch.py
```

4.3.2, Modify parameters on the virtual machine side

Open the dynamic parameter adjuster on the virtual machine side, 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

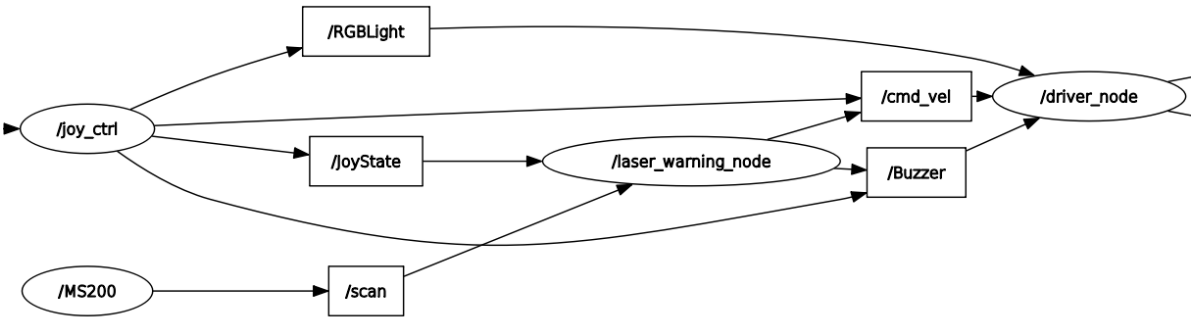
The above parameters can be adjusted. Check or uncheck [Switch] to turn on or pause the radar tracking function.

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

4.3.3, View the topic communication node graph

Virtual machine terminal input,

```
ros2 run rqt_graph rqt_graph
```



4.4, 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, 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, the buzzer will sound, and finally the speed topic data will be released.

```
ranges = np.array(scan_data.ranges)
for i in range(len(ranges)):
    angle = (scan_data.angle_min + scan_data.angle_increment * i) * 180 / pi
    if angle > 180: angle = angle - 360
    if abs(angle) < self.LaserAngle and ranges[i] > 0:
        minDistList.append(ranges[i])
        minDistIDList.append(angle)
if len(minDistIDList) != 0:
    minDist = min(minDistList)
    minDistID = minDistIDList[minDistList.index(minDist)]
else:
    return

velocity = Twist()
angle_pid_compute = self.ang_pid.pid_compute(minDistID / 36, 0)
if abs(angle_pid_compute) < 0.02:
    velocity.angular.z = 0.0
else:
    velocity.angular.z = angle_pid_compute
self.pub_vel.publish(velocity)

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
```