4. Lidar guard

4.1 Program function Description

After the program starts, the car 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 car, less than the set distance, the buzzer will alarm and beep until the target point is far away from the distance set by the car.

Enable/pause this function by starting the Dynamic Parameter Adjuster on the virtual machine and clicking [Switch].

In addition, the [L1] button on the handle locks/turns on the car's motion controls. When motion control is enabled, the function is locked; This function can be turned on when the motion control is locked.

4.2. Program Code Reference path

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

```
#python文件#python file
/userdata/yahboomcar_ws/src/yahboomcar_laser/yahboomcar_laser/laser_warning.py
#launch文件#launch file
/userdata/yahboomcar_ws/src/yahboomcar_laser/launch/laser_warning_launch.py
```

4.3. Program Startup

4.3.1. Launch command

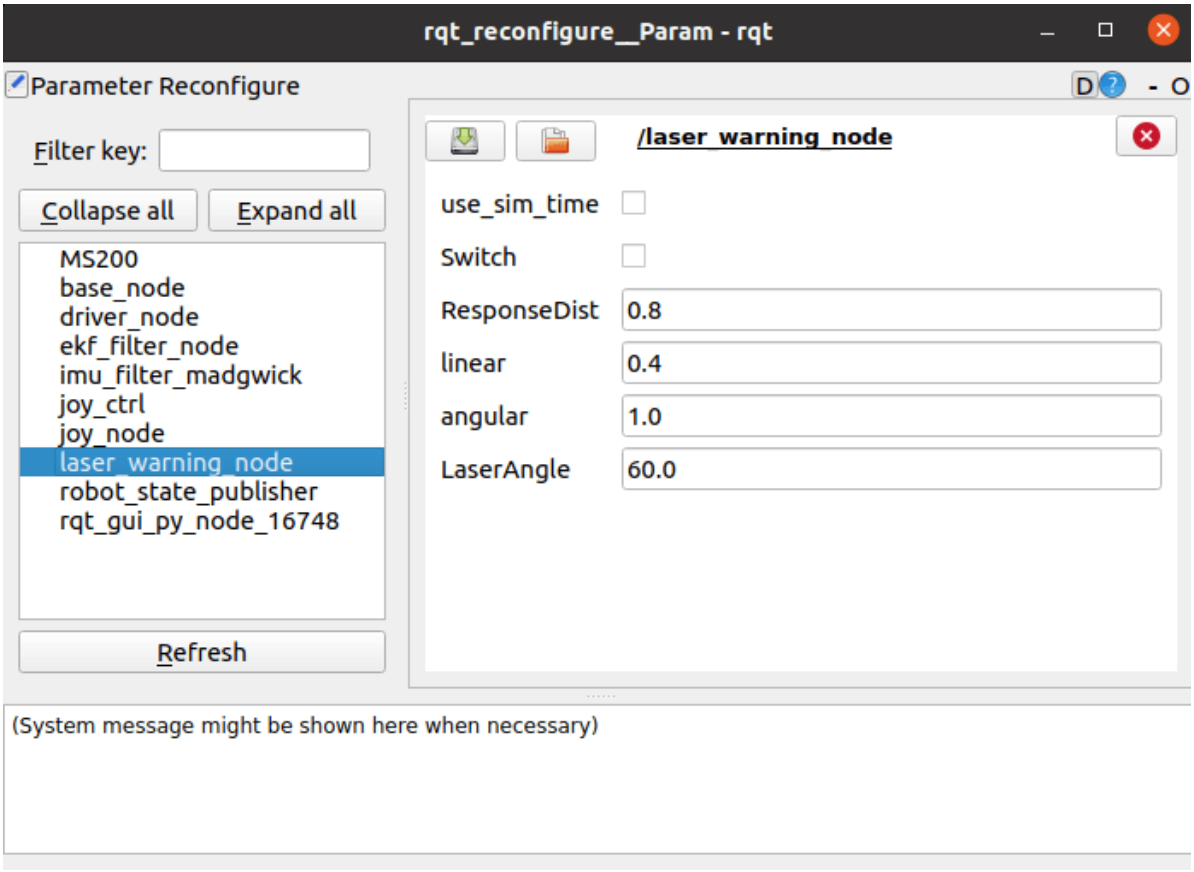
After SSH connects to the car, terminal input,

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

4.3.2. Modifying parameters on the VM side

On the virtual machine, open the dynamic parameter adjuster, open the terminal input,

```
ros2 run rqt_reconfigure rqt_reconfigure
```



The meanings of the parameters are as follows:

parameter name	parameter meaning
Switch	Play switch
ResponseDist	Obstacle detection distance
linear	The linear velocity
angular	Angular velocity
LaserAngle	Radar detection Angle

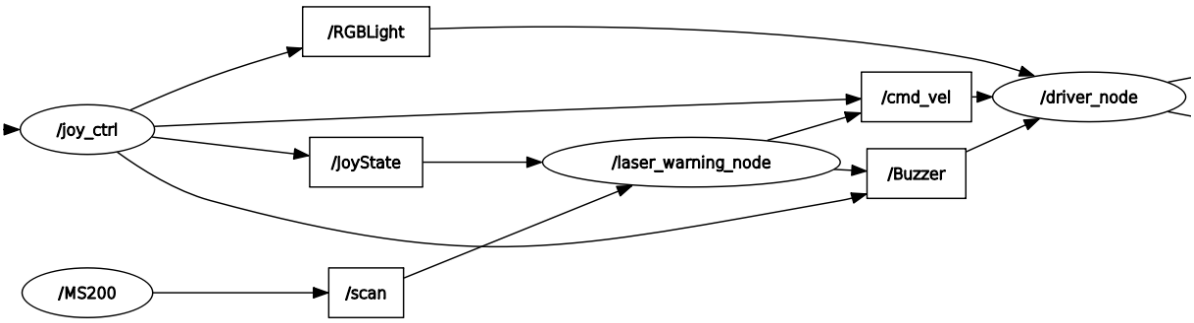
The above parameters can be adjusted to enable or disable the lidar tracking function by checking or unchecking [Switch].

The other four need to be set when the decimal, modify, press the enter key or click the blank space can be written.

4.3.3. View the topic communication node graph

Virtual machine terminal input,

```
ros2 run rqt_graph rqt_graph
```



4.4. Core Source Code Parsing

Mainly look at the lidar callback function, which 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 less than the set distance, let the buzzer sound, and finally release the speed topic data.

```
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
```