3. Lidar following

3.1 Description of program function

After the program starts, the lidar scans for the nearest object, then locks on, the object moves, and the car follows.

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.

3.2 Application 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_tracker.py
#launch文件# launch file
/userdata/yahboomcar_ws/src/yahboomcar_laser/launch/laser_tracker_launch.py
```

3.3. Program Startup

3.3.1. Car start command

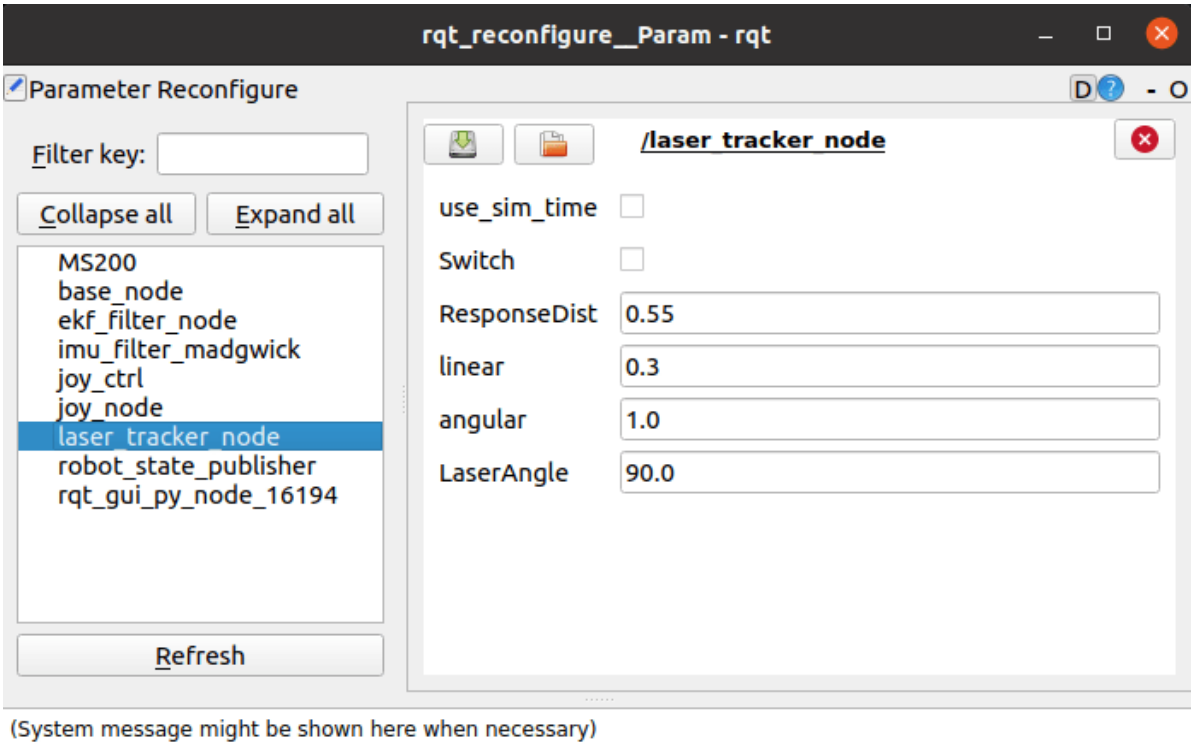
After SSH connects to the car, terminal input,

```
ros2 launch yahboomcar_laser laser_tracker_launch.py
```

3.3.2. 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

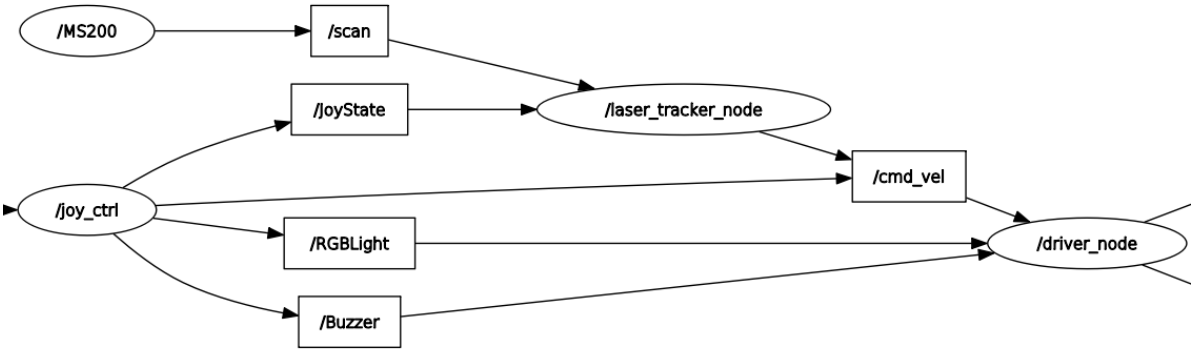
The above parameters can be adjusted to enable or disable the lidar tracking function by checking or unchecking [Switch].

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

3.3.3. View the topic communication node graph

Virtual machine terminal input,

```
ros2 run rqt_graph rqt_graph
```



3.4. Core Source Code Parsing

Mainly look at the lidar callback function, which explains how to obtain the distance information of obstacles at each Angle, then find out the nearest point, then judge the distance, then calculate the speed data, and finally publish.

```
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.priorityAngle: #priorityAngle是小车优先考虑跟随范围
        #priorityAngle is the cart preference to follow the range
        if 0 < ranges[i] < self.ResponseDist + self.offset:
            frontDistList.append(ranges[i])
            frontDistIDList.append(angle)
    elif abs(angle) < self.LaserAngle and ranges[i] > 0:
        minDistList.append(ranges[i])
        minDistIDList.append(angle)

#计算出最小距离点及ID# Calculate the minimum distance point and ID
if len(frontDistIDList) != 0:
    minDist = min(frontDistList)
    minDistID = frontDistIDList[frontDistList.index(minDist)]
else:
    minDist = min(minDistList)
    minDistID = minDistIDList[minDistList.index(minDist)]

velocity = Twist()
#计算线速度# Calculate the linear velocity
if abs(minDist - self.ResponseDist) < 0.1:
    velocity.linear.x = 0.0
else:
    velocity.linear.x = -self.lin_pid.pid_compute(self.ResponseDist, minDist)
#计算角速度# Calculate the angular velocity
angle_pid_compute = self.ang_pid.pid_compute(minDistID / 72, 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)
```