Lidar patrol

This chapter needs to be used with a car chassis and other sensors to operate. This is just an explanation of the implementation method. In fact, it cannot be run. It needs to be used with the company's rosmaster-X3 car to achieve this function.

If you need to transplant it to your own motherboard, you need to install other dependency files.

1. Function description

After the program is started, the car moves according to the set patrol route.

During operation, the radar works simultaneously and will stop if an obstacle is detected within the detection range.

2. Code path

The source code of this function is located in the supporting virtual machine system.

```
~/rplidar_ws/src/yahboomcar_bringup/yahboomcar_bringup/patrol_a1_X3.py
```

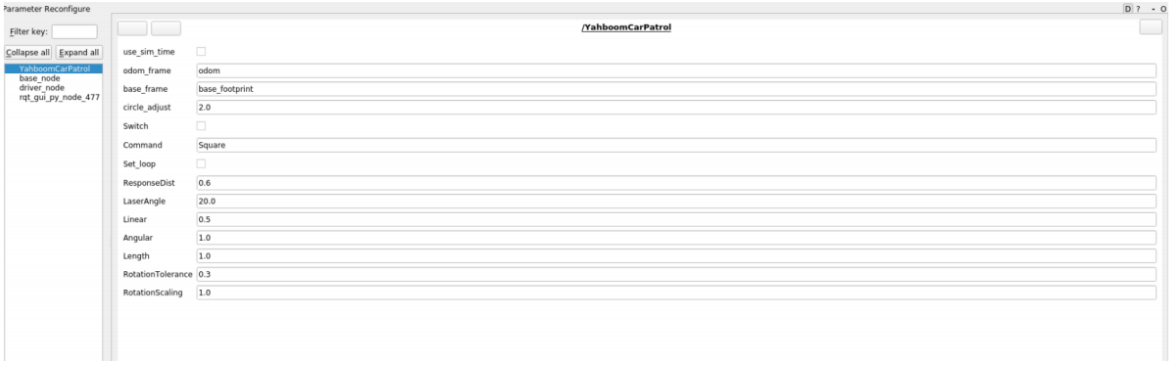
3. Start up

Input following command:

```
ros2 run yahboomcar_bringup Mcnamu_driver_X3
ros2 launch rplidar_ros lidar.launch.py
ros2 run yahboomcar_bringup patrol_a1_X3
```

Enter the following command to open the parameter adjuster.

```
ros2 run rqt_reconfigure rqt_reconfigure
```



After starting the program, enter any of the following routes in the [Command] column on the GUI interface of the dynamic parameter adjuster:

- LengthTest: straight line test
- Circle: circular route patrol
- Square: Square route patrol
- Triangle: Triangular route patrol

After selecting the route, click on the blank space to write the parameters, and then click the [Switch] button to start patrolling.
After an exercise is completed, if [Set_loop] is checked, the last route will be cycled for patrol, otherwise it will stop after completing a patrol.

4. Code analysis

patrol_a1_X3.py

The role of the lidar is to detect obstacles ahead and stop the car after scanning the obstacles.

```
def LaserScanCallback(self,scan_data):
    if self.ResponseDist == 0: return
    ranges = np.array(scan_data.ranges)
    sortedIndices = np.argsort(ranges)
    self.warning = 1
    for i in range(len(ranges)):
        angle = (scan_data.angle_min + scan_data.angle_increment * i) * RAD2DEG
        if abs(angle) > (180 - self.LaserAngle):
            if ranges[i] < self.ResponseDist: self.warning += 1
if self.Joy_active or self.warning > 10:
    if self.moving == True:
        self.pub_cmdVel.publish(Twist())
        self.moving = False
        print("obstacles")
    else:
        #print("Go")
        self.pub_cmdVel.publish(move_cmd)
self.moving = True
return False
```