

# 雷达警卫

注：虚拟机需要与小车处在同一个局域网下，且ROS\_DOMAIN\_ID，需要一致，可以查看【使用前必看】来设置板子上的IP和ROS\_DOMAIN\_ID。

## 1、程序功能说明

小车连接上代理，运行程序，小车上的雷达扫描设定范围内的最近的一个物体，并且会通过自转跟踪该物体，如果该物体靠近雷达小于设定的距离，则小车上的蜂鸣器会响以示警告。通过动态参数调节器可以调整雷达检测的范围和避障检测的距离等参数。

## 2、启动并连接代理

以配套虚拟机为例，输入以下指令启动代理，

```
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --net=host microros/micro-ros-agent:humble udp4 --port 8090 -v4
```

```
yahboom@yahboom-VM:~$ sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm
--privileged --net=host microros/micro-ros-agent:humble udp4 --port 8090 -v4
[1704167422.995513] info      | UDPv4AgentLinux.cpp | init      |
running...                | port: 8090
[1704167422.995832] info      | Root.cpp            | set_verbose_level | 1
ogger setup                | verbose_level: 4
```

然后，打开小车开关，等待小车连接上代理，连接成功如下图所示，

```
[1702630014.015846] info      | ProxyClient.cpp     | create_participant | participant created | client_key: 0x0862A009, part
icipant_id: 0x000(1)
[1702630014.135363] info      | ProxyClient.cpp     | create_topic        | topic created       | client_key: 0x0862A009, topl
c_id: 0x000(2), participant_id: 0x000(1)
[1702630014.223689] info      | ProxyClient.cpp     | create_publisher    | publisher created    | client_key: 0x0862A009, publ
isher_id: 0x000(3), participant_id: 0x000(1)
[1702630014.415510] info      | ProxyClient.cpp     | create_datawriter   | datawriter created   | client_key: 0x0862A009, data
writer_id: 0x000(5), publisher_id: 0x000(3)
[1702630014.428530] info      | ProxyClient.cpp     | create_topic        | topic created       | client_key: 0x0862A009, topl
c_id: 0x001(2), participant_id: 0x000(1)
[1702630014.527190] info      | ProxyClient.cpp     | create_publisher    | publisher created    | client_key: 0x0862A009, publ
isher_id: 0x001(3), participant_id: 0x000(1)
[1702630014.543889] info      | ProxyClient.cpp     | create_datawriter   | datawriter created   | client_key: 0x0862A009, data
writer_id: 0x001(5), publisher_id: 0x001(3)
[1702630014.554490] info      | ProxyClient.cpp     | create_topic        | topic created       | client_key: 0x0862A009, topl
c_id: 0x002(2), participant_id: 0x000(1)
[1702630014.737059] info      | ProxyClient.cpp     | create_publisher    | publisher created    | client_key: 0x0862A009, publ
isher_id: 0x002(3), participant_id: 0x000(1)
[1702630014.755072] info      | ProxyClient.cpp     | create_datawriter   | datawriter created   | client_key: 0x0862A009, data
writer_id: 0x002(5), publisher_id: 0x002(3)
[1702630014.818905] info      | ProxyClient.cpp     | create_topic        | topic created       | client_key: 0x0862A009, topl
c_id: 0x003(2), participant_id: 0x000(1)
[1702630014.840001] info      | ProxyClient.cpp     | create_subscriber   | subscriber created   | client_key: 0x0862A009, subs
criber_id: 0x000(4), participant_id: 0x000(1)
[1702630014.864010] info      | ProxyClient.cpp     | create_datareader   | datareader created   | client_key: 0x0862A009, data
reader_id: 0x000(6), subscriber_id: 0x000(4)
[1702630014.959908] info      | ProxyClient.cpp     | create_topic        | topic created       | client_key: 0x0862A009, topl
c_id: 0x004(2), participant_id: 0x000(1)
[1702630015.033537] info      | ProxyClient.cpp     | create_subscriber   | subscriber created   | client_key: 0x0862A009, subs
criber_id: 0x001(4), participant_id: 0x000(1)
[1702630015.140350] info      | ProxyClient.cpp     | create_datareader   | datareader created   | client_key: 0x0862A009, data
reader_id: 0x001(6), subscriber_id: 0x001(4)
[1702630015.158510] info      | ProxyClient.cpp     | create_topic        | topic created       | client_key: 0x0862A009, topl
c_id: 0x005(2), participant_id: 0x000(1)
[1702630015.241039] info      | ProxyClient.cpp     | create_subscriber   | subscriber created   | client_key: 0x0862A009, subs
criber_id: 0x002(4), participant_id: 0x000(1)
[1702630015.347393] info      | ProxyClient.cpp     | create_datareader   | datareader created   | client_key: 0x0862A009, data
reader_id: 0x002(6), subscriber_id: 0x002(4)
```

## 3、启动程序

### 3.1运行指令

如果是树莓派桌面版本和jetson nano桌面版本，需要提前进入docker，终端输入，

```
sh ros2_humble.sh
```

出现以下界面就是进入docker成功，

```
pi@raspberrypi:~$ ./ros2_humble.sh
access control disabled, clients can connect from any host
MY_DOMAIN_ID: 20
root@raspberrypi:/#
```

之后在docker里输入，（查看【docker环境】章节，如何进入同一个docker终端）

```
ros2 run yahboomcar_laser laser_warning      #雷达警卫
ros2 run yahboomcar_laser laser_warning      #动态参数调节
```

以配套的虚拟机为例，终端输入，

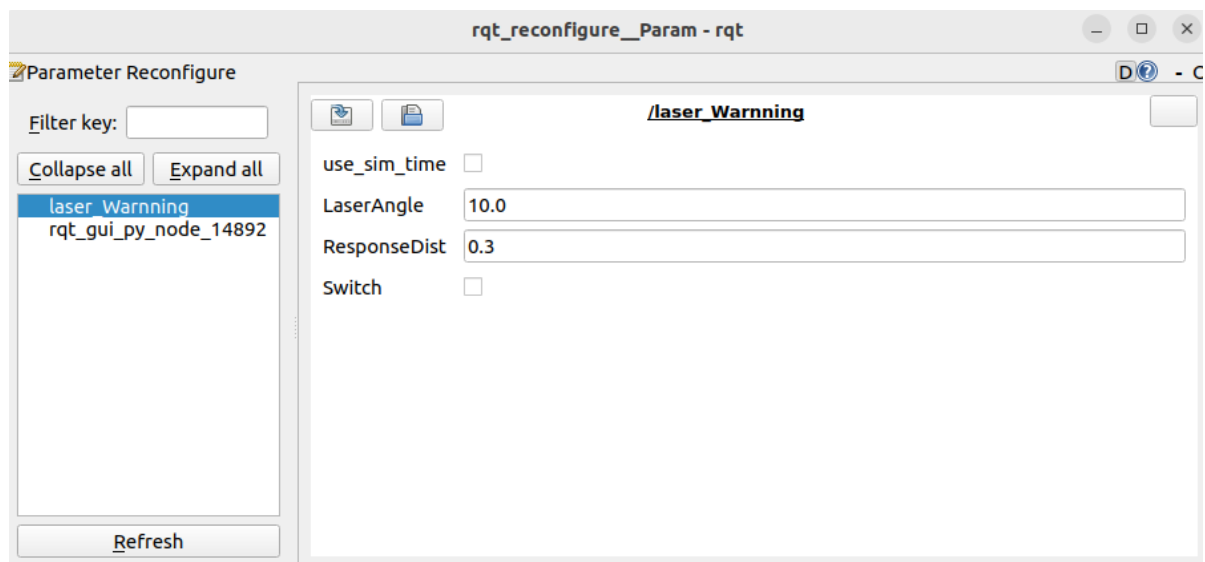
```
ros2 run yahboomcar_laser laser_warning
```

```
minDist: 0.349
minDistID: -6.403302346886954e-06
no obstacles@
minDist: 0.333
minDistID: -6.403302346886954e-06
no obstacles@
minDist: 0.29
minDistID: 1.9999935812049197
-----
minDist: 0.277
minDistID: 2.999993573458553
-----
minDist: 0.267
minDistID: 2.999993573458553
-----
minDist: 0.255
minDistID: 2.999993573458553
```

程序启动后，会寻找雷达扫描范围内最近的物体，缓慢移动该物体，小车会通过自转去跟踪该物体。

如上图所示，没有超过设定范围则打印【no obstacles@】，出现了障碍物则会打印【-----】并且蜂鸣器会响。可以通过动态参数调节器去设置一些参数，终端输入，

```
ros2 run rqt_reconfigure rqt_reconfigure
```



注：刚开始打开的时候可能没有以上节点，点击Refresh刷新后可以看到全部节点。显示的laser\_Warning就是雷达警卫的节点。

以上参数说明：

- LaserAngle：雷达检测角度
- ResponseDist：跟踪的距离
- Switch：玩法开关

修改完以上的参数，需要点击空白处，才能把参数传入程序中。

## 4、代码解析

源码参考路径（以配套虚拟机为例）：

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_laser/yahboomcar_laser
```

jetson nano代码路径：

```
/root/yahboomcar_ws/src/yahboomcar_laser/yahboomcar_laser
```

树莓派代码路径：

```
/root/yahboomcar_ws/src/yahboomcar_laser/yahboomcar_laser
```

laser\_Warning.py，核心代码如下，

```
#创建雷达订阅者订阅雷达数据和遥控控制数据以及速度发布者发布速度数据和蜂鸣器发布者发布蜂鸣器控制数据
self.sub_laser = self.create_subscription(LaserScan, "/scan", self.registerScan, 1)
self.sub_JoyState = self.create_subscription(Bool, '/JoyState',
self.JoyStateCallback, 1)
self.pub_vel = self.create_publisher(Twist, '/cmd_vel', 1)
self.pub_Buzzer = self.create_publisher(UInt16, '/beep', 1)
#雷达回调函数：处理订阅到的雷达数据
ranges = np.array(scan_data.ranges)
minDistList = []
minDistIDList = []
```

```

for i in range(len(ranges)):
    angle = (scan_data.angle_min + scan_data.angle_increment * i) * RAD2DEG
    #根据设定的雷达检测角度，找到雷达检测范围内最近的物体
    if angle > 180: angle = angle - 360
    if abs(angle) < self.LaserAngle and ranges[i] > 0:
        minDistList.append(ranges[i])
        minDistIDList.append(angle)
    if len(minDistList) != 0:
        minDist = min(minDistList)
        minDistID = minDistIDList[minDistList.index(minDist)]
    else:
        return
    #根据与跟踪物体的位置偏差，计算角速度，使得小车车头对准该物体
    angle_pid_compute = self.ang_pid.pid_compute(minDistID/48, 0)
    if abs(angle_pid_compute) < 0.1:
        velocity.angular.z = 0.0
    else:
        velocity.angular.z = angle_pid_compute
    self.pub_vel.publish(velocity)
    #判断最小物体的雷达检测的距离是否小于设定的范围，随后做出判断是否需要让蜂鸣器响
    if minDist <= self.ResponseDist:
        print("-----")
        b = UInt16()
        b.data = 1
        self.pub_Buzzer.publish(b)

    else:
        print("no obstacles@")
        b = UInt16()
        b.data = 0
        self.pub_Buzzer.publish(UInt16())

```