雷达巡逻

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。

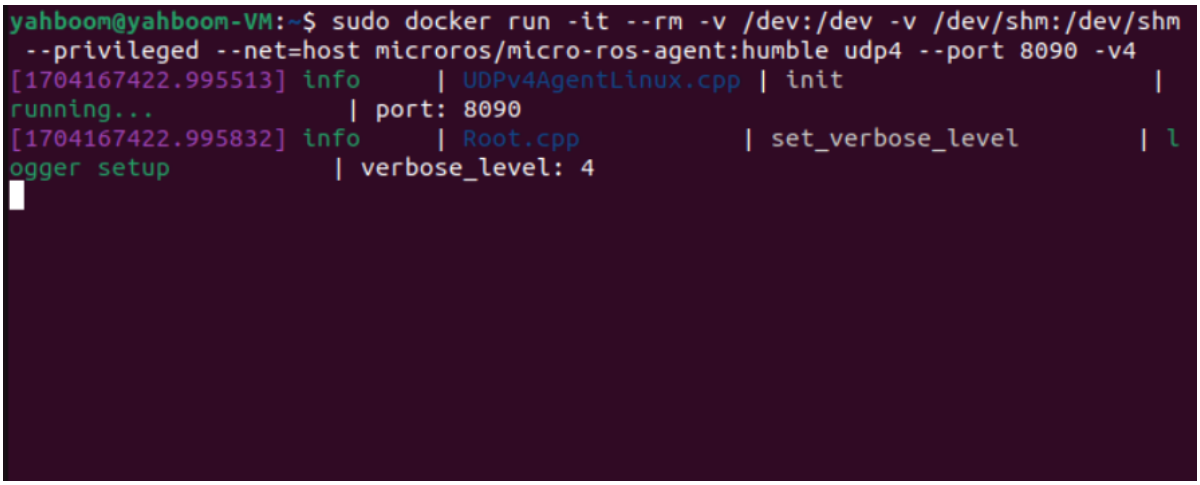
1、程序功能说明

小车连接上代理，运行程序，在动态参数调节器设定需要巡逻的路线，然后点击开始。小车会根据设定的巡逻路线进行运动，同时小车上的雷达会扫描设定的雷达角度和设定障碍检测距离范围内是否有障碍物，有障碍则会停下且蜂鸣器会响，没有障碍物则继续巡逻。

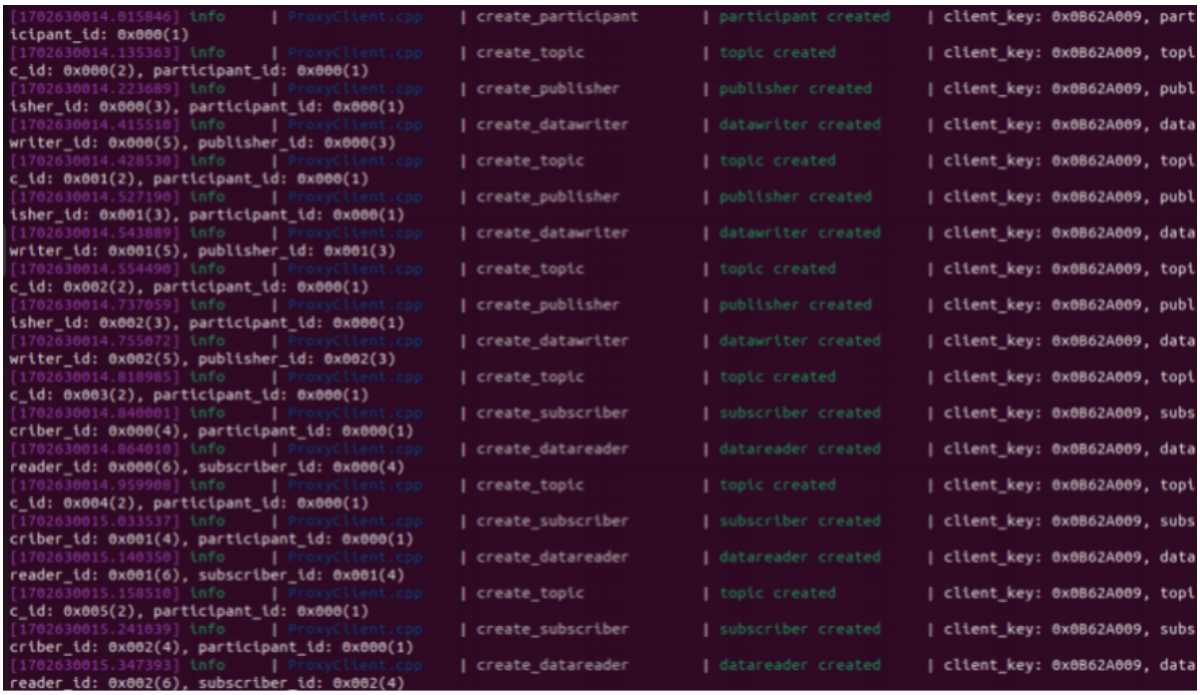
2、启动并连接代理

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
```



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，



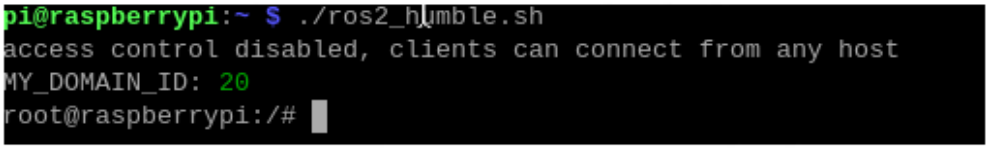
3、启动程序

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，



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_base_node base_node_X3                  #底层处理程序
ros2 run yahboomcar_bringup patrol                          #巡逻程序
ros2 run rqt_reconfigure rqt_reconfigure                    #动态参数调节
```

以配套的虚拟机为例，终端输入，

```
ros2 run yahboomcar_base_node base_node_X3
```

首先启动小车处理底层数据程序，该程序会发布odom->base_footprint的TF变换，有了这个TF变化接可以计算出“小车走了多远”，

```
[INFO] [imu_filter_madgwick_node-1]: process started with pid [6638]
[INFO] [ekf_node-2]: process started with pid [6640]
[INFO] [static_transform_publisher-3]: process started with pid [6642]
[INFO] [joint_state_publisher-4]: process started with pid [6644]
[INFO] [robot_state_publisher-5]: process started with pid [6646]
[INFO] [static_transform_publisher-6]: process started with pid [6658]
[static_transform_publisher-3] [WARN] [1702865272.944043208] []: Old-style arguments are deprecated; see --help for new-style argume
nts
[static_transform_publisher-6] [WARN] [1702865272.984740987] []: Old-style arguments are deprecated; see --help for new-style argume
nts
[static_transform_publisher-3] [INFO] [1702865272.991057276] [base_link_to_base_imu]: Spinning until stopped - publishing transform
[static_transform_publisher-3] translation: ('-0.002999', '-0.003000', '0.031701')
[static_transform_publisher-3] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-3] from 'base_link' to 'imu_frame'
[static_transform_publisher-6] [INFO] [1702865273.005707993] [static_transform_publisher_JH06Gexf4GRodmgs]: Spinning until stopped -
 publishing transform
[static_transform_publisher-6] translation: ('0.000000', '0.000000', '0.050000')
[static_transform_publisher-6] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-6] from 'base_footprint' to 'base_link'
[robot_state_publisher-5] [WARN] [1702865273.013202438] [kdl_parser]: The root link base_link has an inertia specified in the URDF,
but KDL does not support a root link with an inertia.  As a workaround, you can add an extra dummy link to your URDF.
[robot_state_publisher-5] [INFO] [1702865273.013312806] [robot_state_publisher]: got segment base_link
[robot_state_publisher-5] [INFO] [1702865273.013516195] [robot_state_publisher]: got segment imu_Link
[robot_state_publisher-5] [INFO] [1702865273.013524175] [robot_state_publisher]: got segment jq1_Link
[robot_state_publisher-5] [INFO] [1702865273.013528144] [robot_state_publisher]: got segment jq2_Link
[robot_state_publisher-5] [INFO] [1702865273.013531665] [robot_state_publisher]: got segment radar_Link
[robot_state_publisher-5] [INFO] [1702865273.013535185] [robot_state_publisher]: got segment yh_Link
[robot_state_publisher-5] [INFO] [1702865273.013538763] [robot_state_publisher]: got segment yq_Link
[robot_state_publisher-5] [INFO] [1702865273.013542135] [robot_state_publisher]: got segment zh_Link
[robot_state_publisher-5] [INFO] [1702865273.013545612] [robot_state_publisher]: got segment zq_Link
[imu_filter_madgwick_node-1] [INFO] [1702865273.030399479] [imu_filter]: Starting ImuFilter
[imu_filter_madgwick_node-1] [INFO] [1702865273.031826501] [imu_filter]: Using dt computed from message headers
[imu_filter_madgwick_node-1] [INFO] [1702865273.031858361] [imu_filter]: The gravity vector is kept in the IMU message.
[imu_filter_madgwick_node-1] [INFO] [1702865273.032488302] [imu_filter]: Imu filter gain set to 0.100000
[imu_filter_madgwick_node-1] [INFO] [1702865273.032525566] [imu_filter]: Gyro drift bias set to 0.000000
[imu_filter_madgwick_node-1] [INFO] [1702865273.032531441] [imu_filter]: Magnetometer bias values: 0.000000 0.000000 0.000000
[imu_filter_madgwick_node-1] [INFO] [1702865273.053298796] [imu_filter]: First IMU message received.
[joint_state_publisher-4] [INFO] [1702865273.282975810] [joint_state_publisher]: Waiting for robot_description to be published on th
e robot_description topic...
```

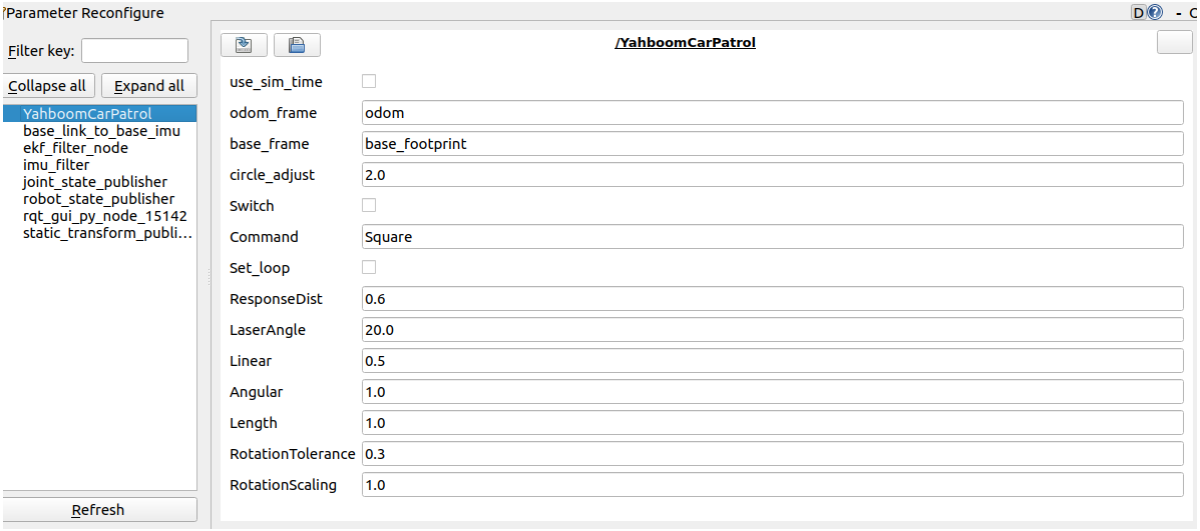
然后，启动雷达巡逻的程序，终端输入，

```
ros2 run yahboomcar_bringup patrol
```

```
yahboom@yahboom-VM:~$ ros2 run yahboomcar_bringup patrol
import finish
create done
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
Switch False
```

打开参数调节器，给定巡逻路线，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YahboomCarPatrol就是巡逻的节点。

4、开始巡逻

在rqt界面中，找到YahboomCarPatrol节点，里边的【Command】就是设定的巡逻路线这里以正方形巡逻路线为例，下边会说明有哪几种巡逻路线。在【Command】设定了路线后，点击Switch即可开始巡逻。

```
distance:  0.6335016035139822
Switch True
Length
distance:  0.6335016035139822
Switch True
Length
distance:  0.7284602368836334
Switch True
Length
distance:  0.7284602368836334
Switch True
Length
distance:  0.7284602368836334
Switch True
Length
distance:  0.8070901854409693
Switch True
Length
distance:  0.8070901854409693
Switch True
Length
distance:  0.8070901854409693
Switch True
Length
```

以正方形为例，先走直线后自转90度，再走直线，再自转90度，以此类推，直到走完的路线为一个正方形。如果在走的过程中，遇到障碍物，则会停下蜂鸣器响，

```
Length
distance:  5.138314275626346
obstacles
Switch True
Length
distance:  5.121662891248637
obstacles
Switch True
Length
distance:  5.121662891248637
obstacles
Switch True
Length
distance:  5.09916912722615
obstacles
Switch True
Length
```

如上图所示，遇到障碍物会打印obstacles。

rqt界面其它参数说明如下：

- odom_frame：里程计的坐标系名称
- base_frame：基坐标系的名称
- circle_adjust：巡逻路线为圆形时，该值可以作为调整圆大小的系数，详细见代码
- Switch：玩法开关
- Command：巡逻路线，有以下几种路线：【LengthTest】-直线巡逻、【Circle】-圆圈巡逻、【Square】-正方形巡逻、【Triangle】-三角形巡逻。
- Set_loop：重新巡逻的开发，设置后将会更具设定的路线继续循环巡逻下去
- ResponseDist：障碍物检测的距离
- LaserAngle：雷达检测的角度

- Linear：线速度大小
- Angular：角速度大小
- Length：直线运动的距离
- RotationTolerance：旋转误差允许的容忍值
- RotationScaling：旋转的比例系数

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。

5、代码解析

源码参考路径（以配套虚拟机为例）：

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_bringup/yahboomcar_bringup
```

jetson nano代码路径：

```
/root/yahboomcar_ws/src/yahboomcar_bringup/yahboomcar_bringup
```

树莓派代码路径：

```
/root/yahboomcar_ws/src/yahboomcar_bringup/yahboomcar_bringup
```

patrol.py，核心代码如下，

创建雷达、遥控数据订阅者

```
self.sub_scan =
self.create_subscription(LaserScan,"/scan",self.LaserScanCallback,1)
self.sub_joy = self.create_subscription(Bool,"/JoyState",self.JoyStateCallback,1)
```

创建速度、蜂鸣器数据发布者

```
self.pub_cmdVel = self.create_publisher(Twist,"cmd_vel",5)
self.pub_Buzzer = self.create_publisher(UInt16,'/beep',1)
```

监听odom和base_footprint的TF变换，计算当前的XY坐标以及旋转的角度，

```
def get_position(self):
    try:
        now = rclpy.time.Time()
        trans =
self.tf_buffer.lookup_transform(self.odom_frame,self.base_frame,now)
        return trans
    except (LookupException, ConnectivityException, ExtrapolationException):
        self.get_logger().info('transform not ready')
        raise
        return
self.position.x = self.get_position().transform.translation.x
self.position.y = self.get_position().transform.translation.y

def get_odom_angle(self):
```

```
    try:
        now = rclpy.time.Time()
        rot =
self.tf_buffer.lookup_transform(self.odom_frame,self.base_frame,now)
        #print("oring_rot: ",rot.transform.rotation)
        cacl_rot = PyKDL.Rotation.Quaternion(rot.transform.rotation.x,
rot.transform.rotation.y, rot.transform.rotation.z, rot.transform.rotation.w)
        #print("cacl_rot: ",cacl_rot)
        angle_rot = cacl_rot.GetRPY()[2]
        return angle_rot
    except (LookupException, ConnectivityException, ExtrapolationException):
        self.get_logger().info('transform not ready')
        raise
        return
self.odom_angle = self.get_odom_angle()
```

两个重要的函数，直线advancing和旋转Spin，所有的巡逻路线不过与是直线与旋转的组合动作，

advancing

```
def advancing(self,target_distance):
    self.position.x = self.get_position().transform.translation.x
    self.position.y = self.get_position().transform.translation.y
    move_cmd = Twist()
    self.distance = sqrt(pow((self.position.x - self.x_start), 2) +
                         pow((self.position.y - self.y_start), 2))
    self.distance *= self.LineScaling
    print("distance: ",self.distance)
    self.error = self.distance - target_distance
    move_cmd.linear.x = self.Linear
    if abs(self.error) < self.LineTolerance :
        print("stop")
        self.distance = 0.0
        self.pub_cmdVel.publish(Twist())
        self.x_start = self.position.x;
        self.y_start = self.position.y;
        self.Switch  =
rclpy.parameter.Parameter('Switch',rclpy.Parameter.Type.BOOL,False)
        all_new_parameters = [self.Switch]
        self.set_parameters(all_new_parameters)
        return True
    else:
        if self.Joy_active or self.warning > 10:
            if self.moving == True:
                self.pub_cmdVel.publish(Twist())
                self.moving = False
                b = UInt16()
                b.data = 1
                self.pub_Buzzer.publish(b)
                print("obstacles")
            else:
                #print("Go")
                b = UInt16()
                b.data = 0
                self.pub_Buzzer.publish(UInt16())
```

```
                self.pub_cmdVel.publish(move_cmd)
                self.moving = True
                return False
```

Spin

```
def Spin(self,angle):
    self.target_angle = radians(angle)
    self.odom_angle = self.get_odom_angle()
    self.delta_angle = self.RotationScaling *
self.normalize_angle(self.odom_angle -        self.last_angle)
    self.turn_angle += self.delta_angle
    print("turn_angle: ",self.turn_angle)
    self.error = self.target_angle - self.turn_angle
    print("error: ",self.error)
    self.last_angle = self.odom_angle
    move_cmd = Twist()
    if abs(self.error) < self.RotationTolerance or self.Switch==False :
        self.pub_cmdVel.publish(Twist())
        self.turn_angle = 0.0
        return True
    if self.Joy_active or self.warning > 10:
        if self.moving == True:
            self.pub_cmdVel.publish(Twist())
            self.moving = False
            b = UInt16()
            b.data = 1
            self.pub_Buzzer.publish(b)
            print("obstacles")
    else:
        b = UInt16()
        b.data = 0
        self.pub_Buzzer.publish(UInt16())
        if self.Command == "Square" or self.Command == "Triangle":
            move_cmd.angular.z = copysign(self.Angular, self.error)
        elif self.Command == "Circle":
            length = self.Linear * self.circle_adjust / self.Length
            move_cmd.linear.x = self.Linear
            move_cmd.angular.z = copysign(length, self.error)
        self.pub_cmdVel.publish(move_cmd)
    self.moving = True
```

有了走直线和旋转的函数了，那么则可以根据巡逻路线来排列组合，以正方形为例，

```
def Square(self):
    if self.index == 0:
    print("Length")
    #首先直线行走1米
    step1 = self.advancing(self.Length)
    #sleep(0.5)
    if step1 == True:
        #self.distance = 0.0
        self.index = self.index + 1;
```

```
        self.Switch  =
rclpy.parameter.Parameter('Switch',rclpy.Parameter.Type.BOOL,True)
        all_new_parameters = [self.Switch]
        self.set_parameters(all_new_parameters)
    elif self.index == 1:
        print("Spin")
        #旋转90度
        step2 = self.Spin(90)
        #sleep(0.5)
        if step2 == True:
            self.index = self.index + 1;
            self.Switch  =
rclpy.parameter.Parameter('Switch',rclpy.Parameter.Type.BOOL,True)
            all_new_parameters = [self.Switch]
            self.set_parameters(all_new_parameters)
.....
```