4. Patrol function play

1. Program function description

After the program starts, open the patrol route set by the dynamic parameter setter, click "switch" in the GUI interface, the car will move according to the set patrol route, during the running process, the lidar will work at the same time, and it will stop when it detects an obstacle within the detection range. After the joystick program is on, you can also pause/continue the movement of the vehicle by pressing R2 button.

2. Program code reference path

After entering the docker container, the location of the source code of this function is located at.

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_bringup/yahboomcar_bringup
/patrol_a1_R2.py
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_bringup/yahboomcar_bringup
/patrol_4ROS_R2.py
```

View the source code according to the actual lidar purchased.

3. Program startup

3.1. Start command

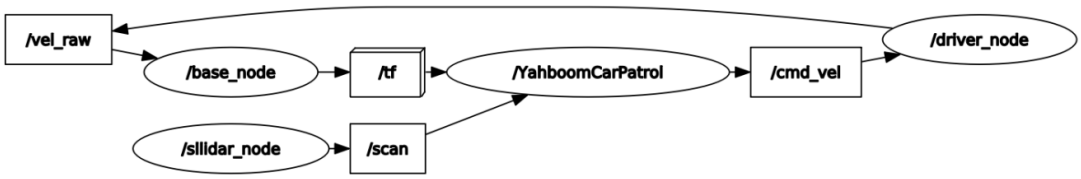
After entering the docker container, according to the actual model and lidar model, terminal input, the

```
#启动小车底盘 #Start the trolley chassis
ros2 run yahboomcar_bringup Ackman_driver_R2
ros2 run yahboomcar_base_node base_node_R2
#启动A1雷达 # Activate A1 lidar
ros2 launch sllidar_ros2 sllidar_launch.py
#启动4ROS雷达 # Activate 4ROS lidar
ros2 launch ydlidar_ros2_driver ydlidar_launch.py
#启动巡逻程序 A1雷达 # Patrol procedures initiated A1 lidar
ros2 run yahboomcar_bringup patrol_a1_R2
#启动巡逻程序 4ROS雷达 # Patrol procedures initiated 4ROS lidar
ros2 run yahboomcar_bringup patrol_4ROS_R2
#启动手柄，如果需要的话 #Start the handle, if needed
ros2 run yahboomcar_ctrl yahboom_joy_R2
ros2 run joy joy_node
```

3.2. View the topic communication node map

docker terminal by typing.

```
ros2 run rqt_graph rqt_graph
```

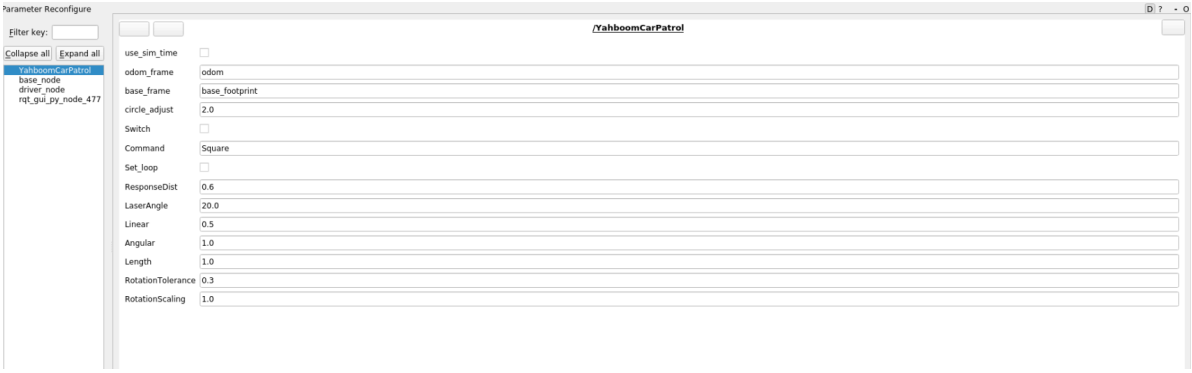


3.3. Dynamic parameter adjustment

docker terminal input.

```
ros2 run rqt_reconfigure rqt_reconfigure
```

The individual parameters of the dynamic parameter regulator are described as follows, the



Parameter name	Parameter Meaning
odom_frame	odometer coordinate system
base_frame	base elevation system
circle_adjust	Turning circle angular speed adjustment factor
Switch	Play switch
Command	Patrol routes
Set_loop	Setting up the loop
ResponseDist	lidar Obstacle Avoidance Response Distance
LaserAngle	lidar scanning angle
Linear	linear velocity
Angular	angular velocity
Length	Straight line test distance
RotationTolerance	Steering Error Tolerance
RotationScaling	Corner scaling factor

Once the program starts, enter any of the following routes in the Comand field of the GUI screen of the Dynamic Parameter Regulator's interface:

- LengthTest: straight line test
- Circle: Circle route patrol
- Square: playground route patrol

After selecting the route, click the blank space to write the parameters, and then click the Switch button to start the patrol movement. If you set up a loop, you can loop the last route to patrol, if the loop is false, then the patrol will stop after completing the patrol.

4. Core source code analysis

The source code of this code is subscribed to odom and base_footprintf TF transformation, so that we can always know "how long we have walked", and then according to the set route, issue speed instruction, take Triangle as an example, here is the analysis.

```
#设定巡逻的路线，进入self.Squaree函数
#Set the route of the patrol into the self.
self.command_src = "Square"
triangle = self.Square()
#以部分self.Square代码解析，
# Parsed as part of the self.Square code.
def Square(self):
    if self.index == 0:
            print("Length")
            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")
            step2 = self.Spin(180)
            #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)
        elif self.index == 2:
            print("Length")
            step3 = self.advancing(self.Length)
            #sleep(0.5)
            if step3 == 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)
        elif self.index == 3:
            print("Spin")
            step4 = self.Spin(180)
            #sleep(0.5)
            if step4 == 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)
        else:
            self.index = 0
            self.Switch  =
rclpy.parameter.Parameter('Switch',rclpy.Parameter.Type.BOOL,False)
            all_new_parameters = [self.Switch]
            self.set_parameters(all_new_parameters)
            #self.Command == "finish"
            print("Done!")
            return True
#完成操场巡逻，主要是看self.advancing和self.Spin函数，这两个函数执行完成后，会返回True，
#Complete the playground patrol, mainly by looking at the self.advancing and
self.Spin functions, which return True when their execution is complete.
 def advancing(self,target_distance):
    #以下是获取xy坐标，与上一时刻的坐标进行计算，计算出自己走了多远
    #获取xy坐标的方式监听odom与base_footprint的tf变换，这部分可参考self.get_position()函
数
     #The following is to get the xy coordinates, and calculate with the
coordinates of the previous moment, to calculate how far you have traveled.
    #Get the xy coordinate by listening to the tf transformation of odom and
base_footprint, this part can refer to self.get_position() function
    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
                print("obstacles")
             else:
                #print("Go")
                self.pub_cmdVel.publish(move_cmd)
            self.moving = True
            return False

def Spin(self,angle):
        self.target_angle = radians(angle)
        #以下是获取位姿，计算出自己转了多少度，获取位姿可以参考self.get_odom_angle函数，同样也是监听了odom与base_footprint的TF变换得来的。
         #The following is to get the position, calculate how many degrees you have turned, to get the position you can refer to self.get_odom_angle function, also listen to the odom and base_footprint TF transformation from.
        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
            '''self.Switch  =
rclpy.parameter.Parameter('Switch',rclpy.Parameter.Type.BOOL,False)
            all_new_parameters = [self.Switch]
            self.set_parameters(all_new_parameters)'''
            return True
        if self.Joy_active or self.warning > 10:
            if self.moving == True:
                self.pub_cmdVel.publish(Twist())
                self.moving = False
                print("obstacles")
```

```
        else:
            if self.Command == "Square" :
                move_cmd.linear.x =  self.Linear
                move_cmd.angular.z = copysign(self.Angular, self.error)
            elif self.Command == "Circle":
                length = self.Linear * self.circle_adjust / self.Length#这里的
circle_adjust是转动角度的系数，算出来length可以理解越大，转圈的半径就越大
                #The circle_adjust here is the coefficient of the rotation angle,
calculated length can be understood that the larger, the larger the radius of the
circle will be.
                #print("length: ",length)
                move_cmd.linear.x = self.Linear
                move_cmd.angular.z = copysign(length, self.error)
                #print("angular: ",move_cmd.angular.z)
                '''move_cmd.linear.x = 0.2
                move_cmd.angular.z = copysign(2, self.error)'''
            self.pub_cmdVel.publish(move_cmd)
        self.moving = True
```