Obstacle avoidance

1. Description of the programme function

When the programme starts, the mechanical dog will walk forward, and when an obstacle appears within the detection range, it will adjust its posture to avoid the obstacle, and then continue to walk forward. If the handle node is activated, the R2 key of the handle can pause/enable this function.

2. Program code reference path

The source code for this function is located at.

```
/home/pi/cartographer_ws2/src/yahboom_laser/yahboom_laser/laser_Avoidance_xgo_RS
200.py
```

3. Program startup

3.1 Start command

Mechanical dog chassis and lidar has been set to boot self-start, if you find that it did not start please enter in the terminal.

```
sudo systemctl restart YahboomStart.service
```

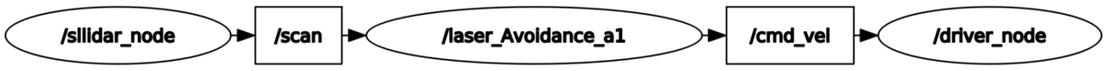
If the lidar and chassis start-up is complete then you need to enter it in the terminal:

```
cd /home/pi/cartographer_ws2
source install/setup.bash
#启动雷达避障程序 雷达MS200
# Activate lidar obstacle avoidance procedures Radar MS200
ros2 run yahboomc_laser laser_Avoidance_xgo_RS200
```

3.2 Viewing the topic communication node map

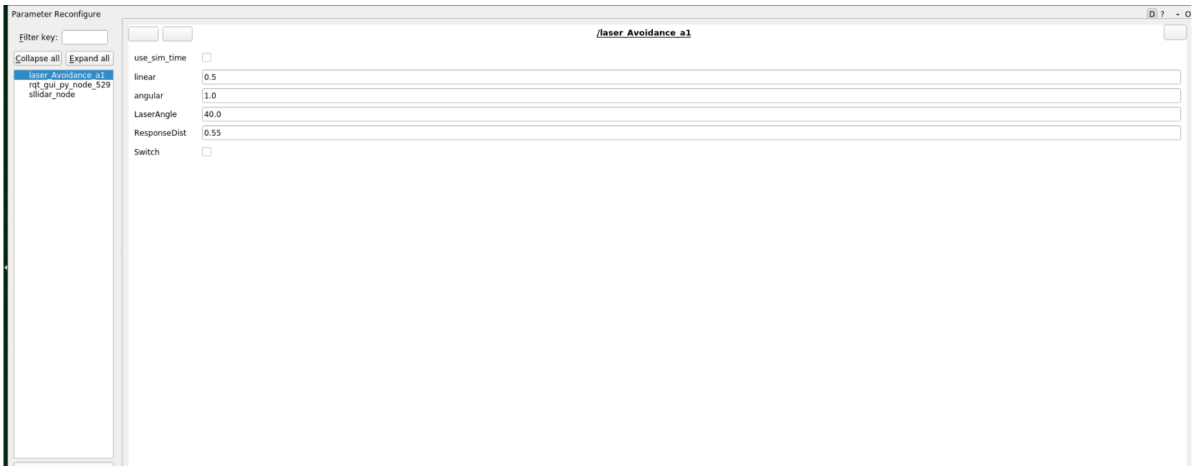
Terminal input.

```
ros2 run rqt_graph rqt_graph
```



It is also possible to set the size of the parameter, the terminal input, by means of a dynamic parameter regulator, the

```
ros2 run rqt_reconfigure rqt_reconfigure
```



use_sim_time ☐
linear 0.5
angular 1.0
LaserAngle 40.0
ResponseDist 0.55
Switch ☐

The meaning of each parameter is as follows.

Parameter name	Parameter Meaning
linear	linear velocity
angular	angular velocity
LaserAngle	lidar detection angle
ResponseDist	Obstacle detection distance
Switch	Play switch

The above parameters are adjustable, except Switc, the other four need to be set when you need to be a decimal, modified, click on the blank before you can write.

4. Core source code analysis

X3 models, A1 lidar source code as an example, mainly to see the lidar callback function, which explains how to get to each angle of the obstacle distance information.

```
def registerScan(self, scan_data):
    if not isinstance(scan_data, LaserScan): return
    ranges = np.array(scan_data.ranges)
    self.Right_warning = 0
    self.Left_warning = 0
    self.front_warning = 0

    for i in range(len(ranges)):
        angle = (scan_data.angle_min + scan_data.angle_increment * i) * RAD2DEG#
雷达的信息的angle是弧度制，这里要转换成角度进行计算
        #The angle of the lidar's information is in radians, here it is
converted to angles for calculation.
        if 160 > angle > 180 - self.LaserAngle:#angle是根据雷达的结构来设定判断范围的
            #angle sets the range of judgement based on the structure of the
lidar
            if ranges[i] < self.ResponseDist*1.5: #range[i]就是雷达扫描的结果，这里指
得是距离信息
                #range[i] is the result of the lidar scan, in this case the
distance information
                self.Right_warning += 1
        if - 160 < angle < self.LaserAngle - 180:
            if ranges[i] < self.ResponseDist*1.5:
                self.Left_warning += 1
        if abs(angle) > 160:
            if ranges[i] <= self.ResponseDist*1.5:
                self.front_warning += 1
        if self.Joy_active or self.Switch == True:
            if self.Moving == True:
                self.pub_vel.publish(Twist())
                self.Moving = not self.Moving
                    return
        self.Moving = True
```