

## 2. Lidar obstacle avoidance

### 1. Program function description

After the program is started, Muto will move forward. When an obstacle appears within the detection range, it will adjust its posture to avoid the obstacle, and then continue to move forward. If the controller node is activated, the R2 key of the controller can pause/enable this function.

### 2. Program code reference path

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

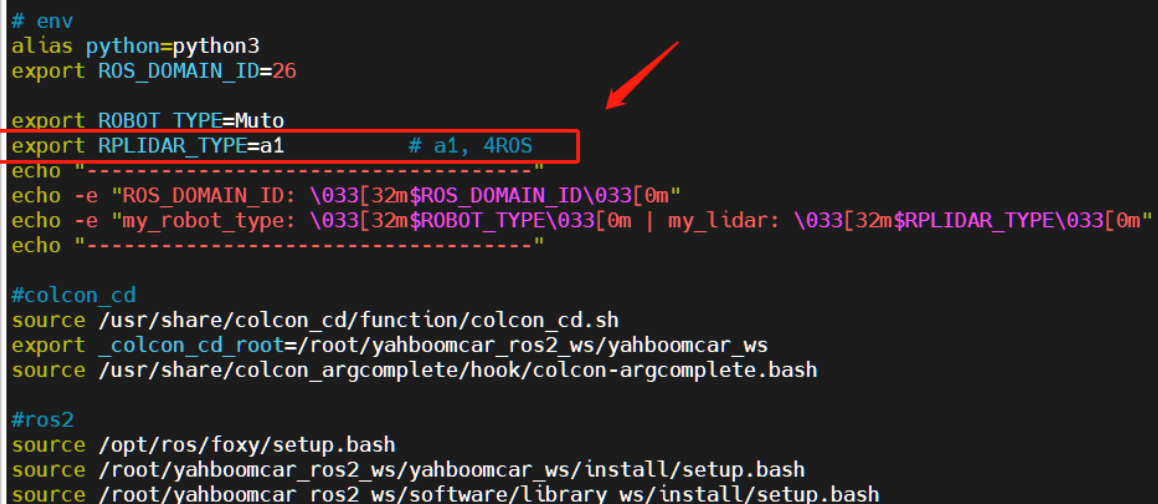
```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_laser/yahboomcar_laser/laser_Avoidance_a1.py      # a1雷达  
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_laser/yahboomcar_laser/laser_Avoidance_4ROS.py    # 4ros雷达  
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_laser/launch/laser_avoidance_launch.py
```

### 3. Configuration before use

Note: Since the Muto series robots are equipped with multiple radar devices, the factory system has been configured with routines for multiple devices. However, since the product cannot be automatically recognized, the radar model needs to be manually set.

After entering the container: Make the following modifications according to the radar type:

```
root@ubuntu:/# cd  
root@ubuntu:~# vim .bashrc
```



```
# env  
alias python=python3  
export ROS_DOMAIN_ID=26  
  
export ROBOT_TYPE=Muto  
export RPLIDAR_TYPE=a1      # a1, 4ROS  
echo "-----"  
echo -e "ROS_DOMAIN_ID: \033[32m$ROS_DOMAIN_ID\033[0m"  
echo -e "my_robot_type: \033[32m$ROBOT_TYPE\033[0m | my_lidar: \033[32m$RPLIDAR_TYPE\033[0m"  
echo "-----"  
  
#colcon_cd  
source /usr/share/colcon_cd/function/colcon_cd.sh  
export _colcon_cd_root=/root/yahboomcar_ros2_ws/yahboomcar_ws  
source /usr/share/colcon_argcomplete/hook/colcon-argcomplete.bash  
  
#ros2  
source /opt/ros/foxy/setup.bash  
source /root/yahboomcar_ros2_ws/yahboomcar_ws/install/setup.bash  
source /root/yahboomcar_ros2_ws/software/library_ws/install/setup.bash
```

After the modification is completed, save and exit vim, and then execute:

```

root@jetson-desktop:~# source .bashrc
-----
ROS_DOMAIN_ID: 26
my_robot_type: Muto | my_lidar: a1
-----
root@jetson-desktop:~#

```

You can see the current modified lidar type.

## 4. Program startup

### 4.1. Start command

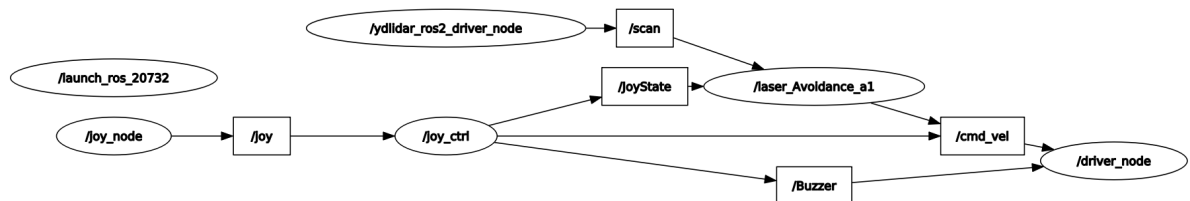
After entering the docker container, enter in the terminal

```
ros2 launch yahboomcar_laser laser_avoidance_launch.py
```

### 4.2. View the topic communication node diagram

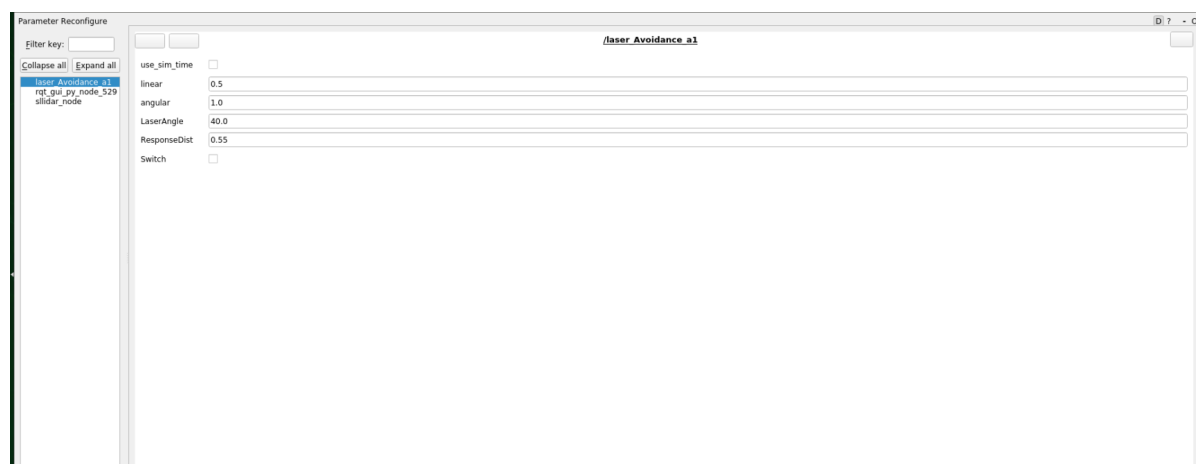
docker terminal input

```
ros2 run rqt_graph rqt_graph
```



You can also set the parameter size through the dynamic parameter adjuster, terminal input,

```
ros2 run rqt_reconfigure rqt_reconfigure
```



The meaning of each parameter is as follows:

Parameter name	Parameter meaning
linear	Line speed
angular	Angular speed
LaserAngle	Lidar detection angle
ResponseDist	Obstacle detection distance
Switch	Game switch

The above parameters are all adjustable. Except for Switch, the other four parameters need to be set to decimals. After modification, click on the blank space to write.

## 5. Core source code analysis

Taking the source code of a1 lidar as an example, we mainly look at the callback function of lidar. Here is an explanation of how to obtain the obstacle distance information at each angle.

```
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#The angle of the lidar information is in radians, and it needs to be
converted into an angle for calculation.
        if 160 > angle > 180 - self.LaserAngle:#Angle sets the judgment range
based on the structure of the lidar.
            if ranges[i] < self.ResponseDist*1.5: #range[i]It is the result of
lidar scanning, which refers to 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
```

