4.Lidar guard

1. Program function description

After the program is started, Muto will track the target at the nearest point. When the target point moves laterally, it will move with the target point.

When the target point is close to Muto and is less than the set distance, the buzzer will beep until the target point is far away from the set distance of Muto.

2. Program code reference path

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

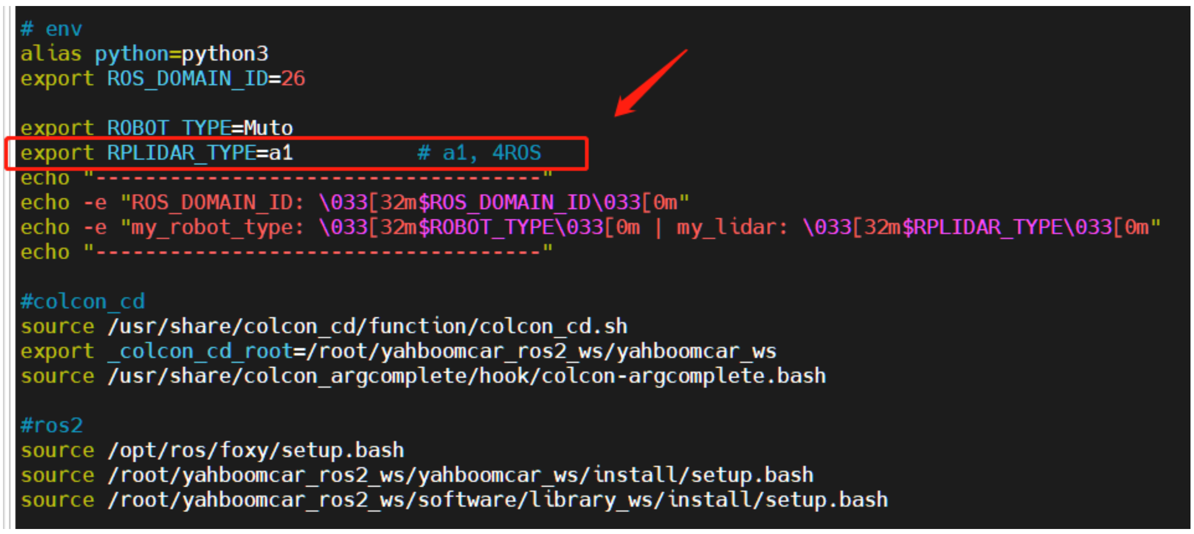
```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_laser/yahboomcar_laser/laser_Warning_a1.py
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_laser/yahboomcar_laser/laser_Warning_4ROS.py
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_laser/launch/laser_warning_launch.py
```

3. Configuration before use

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

After entering the container: Make the following modifications according to the lidar 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

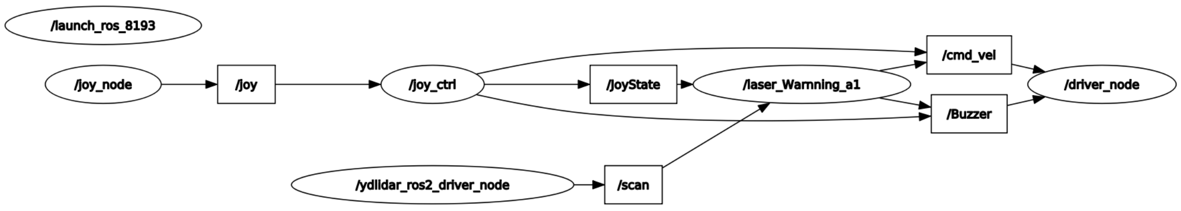
After entering the docker container, enter in the terminal

```
ros2 launch yahboomcar_laser laser_warning_launch.py
```

4.2. View topic communication node graph

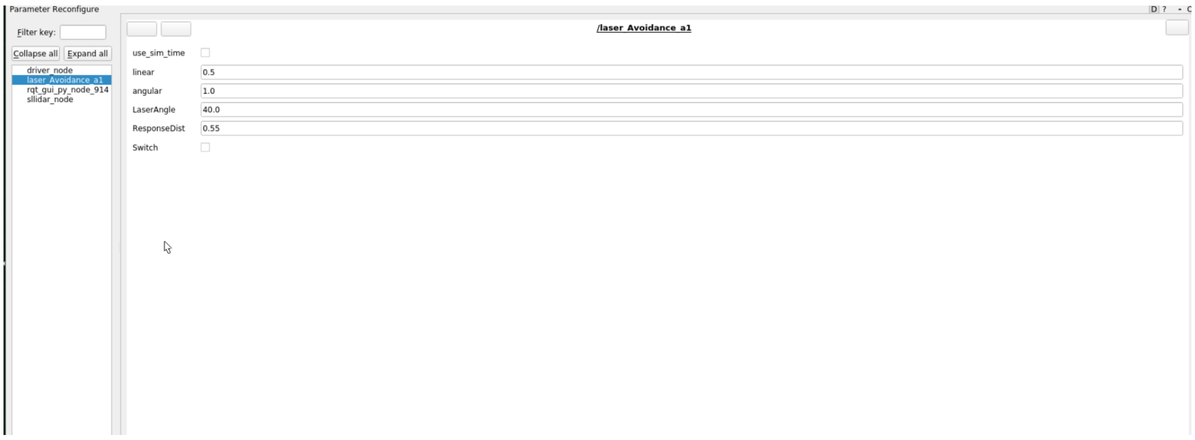
docker terminal input,

```
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
```



/laser_Avoidance_a1

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

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, and then obtain the ID of the minimum distance.Calculate the magnitude of the angular velocity, and then compare the minimum distance with the set distance. If it is less than the set distance, let the buzzer sound, and finally publish the speed topic data.

```
for i in range(len(ranges)):
    angle = (scan_data.angle_min + scan_data.angle_increment * i) * RAD2DEG
    if abs(angle) > (180 - self.LaserAngle):
        minDistList.append(ranges[i])
        minDistIDList.append(angle)
        if len(minDistList) == 0: return
    minDist = min(minDistList)
    minDistID = minDistIDList[minDistList.index(minDist)]
    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
    if minDist <= self.ResponseDist:
        if self.Buzzer_state == False:
            b = Bool()
            b.data = True
            self.pub_Buzzer.publish(b)
            self.Buzzer_state = True
        else:
            if self.Buzzer_state == True:
                self.pub_Buzzer.publish(Bool())
                self.Buzzer_state = False
    velocity = Twist()
    ang_pid_compute = self.ang_pid.pid_compute((180 - abs(minDistID)) / 36, 0)
    if minDistID > 0: velocity.angular.z = -ang_pid_compute
    else: velocity.angular.z = ang_pid_compute
    if ang_pid_compute < 0.02: velocity.angular.z = 0.0
    self.pub_vel.publish(velocity)
```