# Lidar Guard

## 1. Content Description

This section describes how the program combines chassis control with fused radar data to detect the nearest object to the vehicle in real time and control the chassis' rotation to track it. Furthermore, if the object's distance to the vehicle falls below an alarm threshold, a buzzer on the vehicle sounds an alarm.

This section requires entering commands in the terminal. The terminal you open depends on your motherboard type. This section uses the Raspberry Pi 5 as an example. For Raspberry Pi and Jetson-Nano motherboards, you'll need to open a terminal and enter commands to enter a Docker container. Once inside the Docker container, enter the commands mentioned in this section in the terminal. For instructions on entering a Docker container, refer to the product tutorial **[Robot Configuration and Operation Guide] - [Entering the Docker (Jetson-Nano and Raspberry Pi 5 users, see here)**.

Simply open the terminal on the Orin motherboard and enter the commands mentioned in this section.

## 2. Program startup

First, open the terminal and enter the following command to start the radar fusion and radar filtering programs.

```
ros2 launch yahboom_M3Pro_laser laser_driver.launch.py
```

Next, you can refer to this product tutorial [5. Chassis Control] - [2. Handle Control] to start the handle control to control the car conveniently. Press the R2 button on the handle to cancel and start the radar tracking gameplay. If you do not start the handle control, it will not affect the operation of this program. Enter the following command in the terminal to start the radar tracking program,

```
ros2 run yahboom_M3Pro_laser laser_Warning
```

After the program starts, the radar will scan for the nearest object within the detection range. Slowly moving the object left or right will cause the chassis to rotate, tracking the object and aligning the front of the vehicle with it. If the distance between the vehicle and the object is less than 0.55 meters, the onboard buzzer will sound.

## 3. Core code analysis

Program code path:

- Raspberry Pi and Jetson-Nano board

  The program code is in the running docker. The path in docker is /root/yahboomcar_ws/src/yahboom_M3Pro_laser/yahboom_M3Pro_laser/laser_Warning.py

- Orin Motherboard

  The program code path is /home/jetson/yahboomcar_ws/src/yahboom_M3Pro_laser/yahboom_M3Pro_laser/laser_Warning.py

Import the necessary library files,

```
#ros lib
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan
#commom lib
import math
import numpy as np
import time
from time import sleep
#Import chassis PID control related libraries
from yahboom_M3Pro_laser.common import *
import os
```

The program initializes and creates publishers and subscribers,

```
def __init__(self,name):
    super().__init__(name)
    #Create a subscriber to subscribe to the fused radar topic message
    self . sub_laser = self . create_subscription ( LaserScan , "/scan" , self .
registerScan , 1 )
    #Create a subscriber to subscribe to the handle remote control topic message
    self.sub_JoyState = self.create_subscription(Bool,'/JoyState',
self.JoyStateCallback,1)
    #create a pub
    #Create a publisher to publish speed topic message
    self.pub_vel = self.create_publisher(Twist,'/cmd_vel',1)
    #Create a publisher to publish buzzer topic messages
    self.pub_Buzzer = self.create_publisher(UInt16,'/beep',1)
    #declareparam
    self.declare_parameter("linear",0.5)
    self.linear =
self.get_parameter('linear').get_parameter_value().double_value
    self.declare_parameter("angular",1.0)
    self.angular =
self.get_parameter('angular').get_parameter_value().double_value
     # Radar detection angle
    self.declare_parameter("LaserAngle",45.0)
    self.LaserAngle =
self.get_parameter('LaserAngle').get_parameter_value().double_value
```

```
    #Alarm distance, the buzzer will sound if the distance is less than this
value
    self.declare_parameter("ResponseDist",0.55)
    self.ResponseDist =
self.get_parameter('ResponseDist').get_parameter_value().double_value
    #Handle control flag, the value is True means the handle controls the car,
you need to press the R2 key on the handle to take handle control or enable
handle control
    self.Joy_active = False
    #Create angular velocity PID control object
    self.ang_pid = SinglePID(3.0, 0.0, 5.0)
```

registerScan雷达话题回调函数,

```
def registerScan(self, scan_data):
    if not isinstance(scan_data, LaserScan): return
    ranges = np.array(scan_data.ranges)
    minDistList = []
    minDistIDList = []

    for i in range(len(ranges)):
        #Convert the radians in the radar topic data into degrees
        angle = (scan_data.angle_min + scan_data.angle_increment * i) * RAD2DEG
        #If the current angle is within the detection range and the distance
corresponding to the angle is not equal to 0
        if abs(angle) < self.LaserAngle and ranges[i] !=0.0 :
            #print("angle: ",angle)
            #Add the distance of this angle to the minimum distance list
            minDistList.append(ranges[i])
            #Add the angle to the minimum ID list
            minDistIDList.append(angle)
    if len(minDistList) == 0: return
    # Find the minimum value in the distance list
    minDist = min(minDistList)
    #Find the smallest angle based on the shortest distance
    minDistID = minDistIDList[minDistList.index(minDist)]
    print("minDistID: ",minDistID)
    #If self.Joy_active is True, publish the parking speed and exit this callback
function
    if self.Joy_active :
        self.pub_vel.publish(Twist())
        return

    print("minDist: ",minDist)
     #If the car distance is less than the alarm distance
    if minDist <= self.ResponseDist and minDist!=0.0:
        b = UInt16()
        b.data = 1
        self.pub_Buzzer.publish(b)
    else:
        self.pub_Buzzer.publish(UInt16())
    velocity = Twist()
    print("minDistID: ",minDistID)
    #Calculate angular velocity
    ang_pid_compute = self.ang_pid.pid_compute(abs(minDistID)  / 72, 0)
    #If the minimum angle is greater than 0, it means it is on the left side of
the car, so the car turns right
```

```
    if  0 < minDistID :
        velocity . angular . z = ang_pid_compute
    #If the minimum angle is less than 0, it means it is on the right side of the
car, so the car turns left
    elif  minDistID  < 0 :
        velocity . angular . z = - ang_pid_compute
    print ( "orin_angular.z: " , velocity . angular . z )
    if  abs ( ang_pid_compute ) <  0.5 : velocity . angular . z = 0.0
    velocity . angular . z = velocity . angular . z  * 0.5
    print ( "angular.z: " , velocity . angular . z )
    #Release speed control topic
    self . pub_vel . publish ( velocity )
```