4ROS to SLAM S2 Lidar tutorial

1、Install the 4ROS lidar driver and function package, please refer to the 4ROS lidar tutorial for details;

2、Modified command for radar activation. All parts involving radar activation are to be modified.

Input following command to start SLAM S2 Lidar

```
roslaunch rplidar_ros rplidar.launch
```

Input following command to start 4ROS Lidar

```
roslaunch ydlidar_ros_driver TG.launch
```

3、Modify the code of [1.3 Lidar avoiding] [1.4 Lidar guard] [1.5 Lidar follow].

These functions all need to read the angle value of the lidar. Due to the structural differences between the s2 lidar and the 4ROS lidar, their angles will be different, so we need to modify this part to fit the structure of the 4ROS lidar.

laser_Avoidance.py file,

```
if -10 > angle > -10-self.LaserAngle:
    if ranges[i] < self.ResponseDist:
        self.Right_warning += 1
    #print(angle)
    if 10+self.LaserAngle > angle > 10:
        if ranges[i] < self.ResponseDist:
            self.Left_warning += 1
        #print(angle)
    if abs(angle) < 10:
        if ranges[i] <= self.ResponseDist:
            self.front_warning += 1
#print(angle)
```

There are two places in laser_Tracker.py that need to be modified,

```
#First:
if abs(angle) < self.priorityAngle:
    if ranges[i] < (self.ResponseDist + offset) and ranges[i] != 0.0:
        frontDistList.append(ranges[i])
        frontDistIDList.append(angle)
    elif abs(angle) > self.priorityAngle and abs(angle) < self.laserAngle and
ranges[i] != 0.0:
        minDistList.append(ranges[i])
        minDistIDList.append(angle)
#Second:
ang_pid_compute = self.ang_pid.pid_compute((180 - abs(minDistID)) / 72, 0)
```

There are two places in laser_Warning.py that need to be modified,

```
#First:
if abs(angle) < self.laserAngle and ranges[i] != 0.0:
    minDistList.append(ranges[i])
    minDistIDList.append(angle)
#Second:
ang_pid_compute = self.ang_pid.pid_compute((180 - abs(minDistID)) / 128, 0)
```

The above three parts are to modify the source code. In addition, we also need to change the launch that starts the s2 lidar to the launch that starts the 4ROS lidar.

4、Modify the code of [1.2 Hand-held lidar mapping] [1.6 Lidar mapping] [1.7 Navigation and avoiding] [1.8 APP mapping and navigation] [1.9 Patrol game] [1.10 RTAB-Map 3D mapping navigation]

Because the direction of the lidar coordinate system of s2 lidar and 4ROS is different, some values are required during the TF transformation process.

Here we take an example, select laser+bringup as the underlying driver launch file for mapping navigation startup.

```
<launch>
    <arg name="robot_type" value="$(env ROBOT_TYPE)" doc="robot_type
[X1,X3,X3plus,R2,X7]"/>
    <!-- drive module || driver module -->
    <include file="$(find ydlidar_ros_driver)/launch/TG.launch"/>
    <include file="$(find yahboomcar_bringup)/launch/bringup.launch"/>
    <!-- Coordinate system of lidar|| Lidar coordinate system -->
    <node pkg="tf" type="static_transform_publisher" name="base_link_to_laser"
          args="0.0435 5.258E-05 0.11 3.1416 0 0  /base_link /laser 30"
if="$(eval arg('robot_type') == 'X3')"/>
    <node pkg="tf" type="static_transform_publisher" name="laser_link_to_laser"
          args="0 0 0 6.28 0 0  /laser_link /laser 30" if="$(eval
arg('robot_type') == 'X3plus')"/>
</launch>
```

There are two places in this launch file that need to be modified

- **The command to start the lidar**
- **TF coordinate transformation**

The first three values are the values of xyz, and the last three values are yaw pitch roll, mainly to modify the value of yaw, here it is radian system, 6.28 means 360°.