

# 7.ROS2\_Gmapping mapping algorithm

---

## 7.ROS2\_Gmapping mapping algorithm

- 1 Introduction
2. Use
  - 2.1. Configuration before use
  - 2.2. Specific use
3. Node analysis
  - 3.1. Display calculation graph
  - 3.2. gmapping node details
  - 3.3. TF transformation

wiki: <http://wiki.ros.org/gmapping/>

ros2 gmapping: [https://github.com/Project-MANAS/slam\\_gmapping](https://github.com/Project-MANAS/slam_gmapping)

## 1 Introduction

---

- gmapping is only applicable to points where the number of two-dimensional laser points in a single frame is less than 1440. If the number of laser points in a single frame is greater than 1440, then problems such as [[mapping-4] process has died] will occur.
- Gmapping is a commonly used open source SLAM algorithm based on the filtered SLAM framework.
- Gmapping is based on the RBpf particle filter algorithm, which separates the real-time positioning and mapping processes. Positioning is performed first and then mapping is performed.
- Gmapping has made two major improvements on the RBpf algorithm: improved proposal distribution and selective resampling.

**Advantages:**Gmapping can construct indoor maps in real time. The amount of calculation required to construct small scene maps is small and the accuracy is high.

**Disadvantages:** As the scene grows, the number of particles required increases because each particle carries a map, so the amount of memory and computation required when building a large map increases. Therefore it is not suitable for building large scene maps. And there is no loop detection, so the map may be misaligned when the loop is closed. Although increasing the number of particles can close the map, it comes at the expense of increased calculations and memory.

## 2. Use

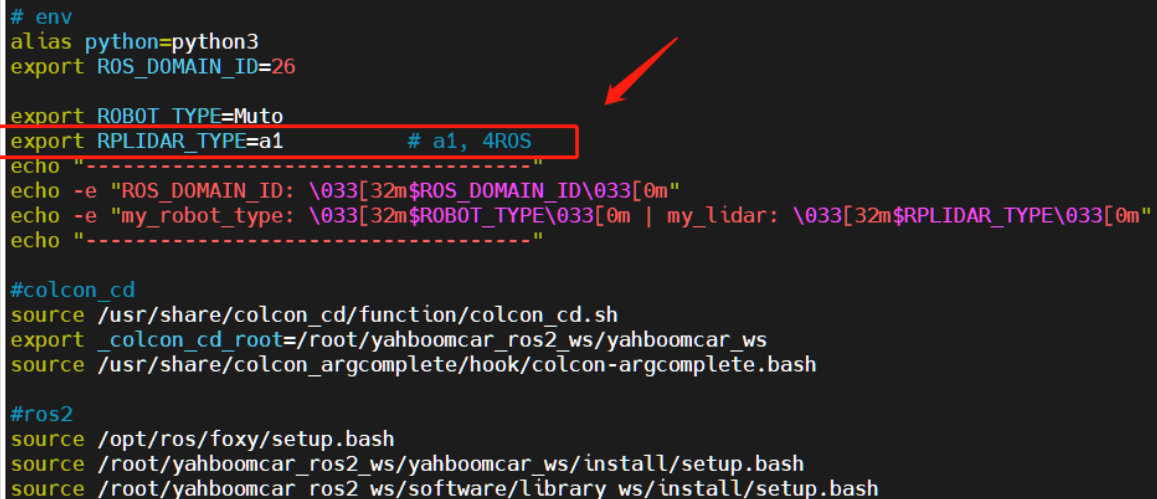
---

## 2.1. 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 radar 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:~#
```

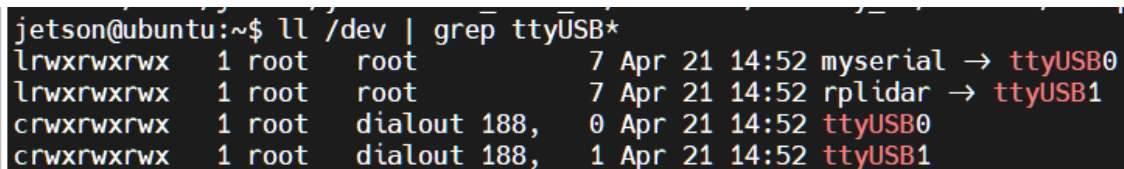
You can see the current modified lidar type.

## 2.2. Specific use

**Note: When building a map, the slower the speed, the better the effect (note that the rotation speed should be slower). If the speed is too fast, the effect will be poor.**

First, the port binding operation needs to be done in the host [Muto]. The two devices of lidar and serial port are mainly used here;

Then check whether the lidar and serial device are in port binding state: Run the following command on the host machine [Muto] to check. Successful binding is as follows:



```
jetson@ubuntu:~$ ll /dev | grep ttyUSB*
lrwxrwxrwx 1 root root 7 Apr 21 14:52 myserial -> ttyUSB0
lrwxrwxrwx 1 root root 7 Apr 21 14:52 rplidar -> ttyUSB1
crwxrwxrwx 1 root dialout 188, 0 Apr 21 14:52 ttyUSB0
crwxrwxrwx 1 root dialout 188, 1 Apr 21 14:52 ttyUSB1
```

If it shows that the lidar or serial device is not bound, you can plug and unplug the USB cable to check again.

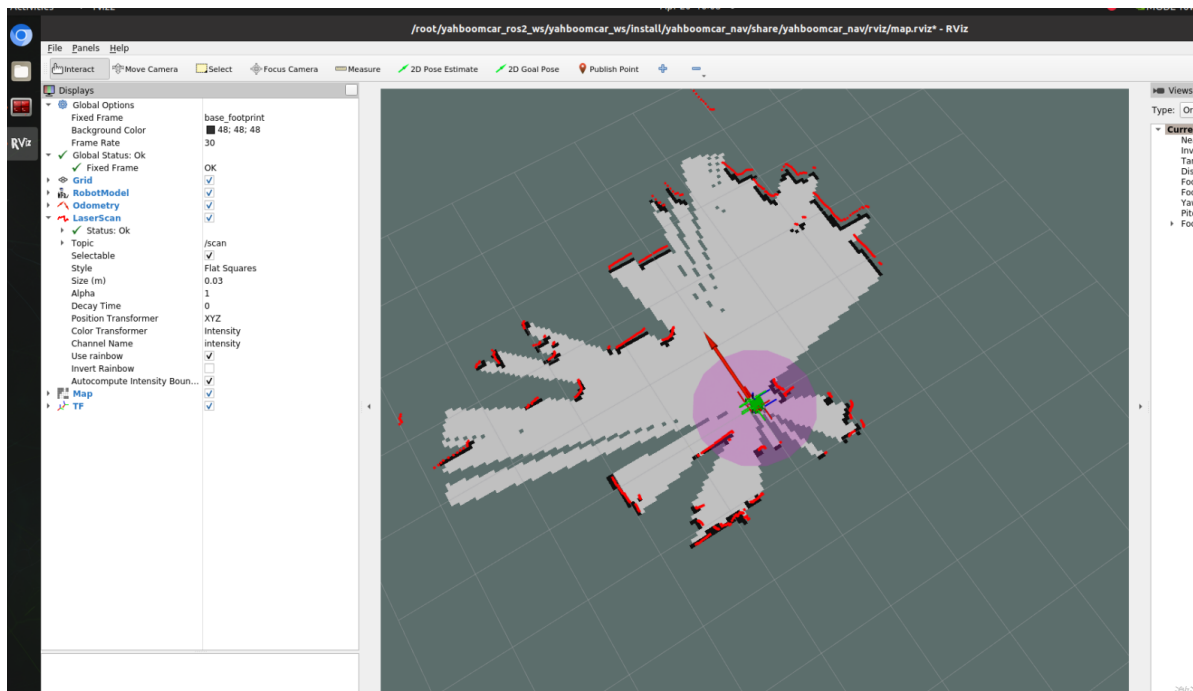
Enter the docker container and execute the following launch file in a terminal:

1. Start mapping

```
ros2 launch yahboomcar_nav map_gmapping_launch.py
```

2. Start rviz to display the map. It is recommended to perform this step in [Virtual Machine].  
Multi-machine communication needs to be configured in the virtual machine.

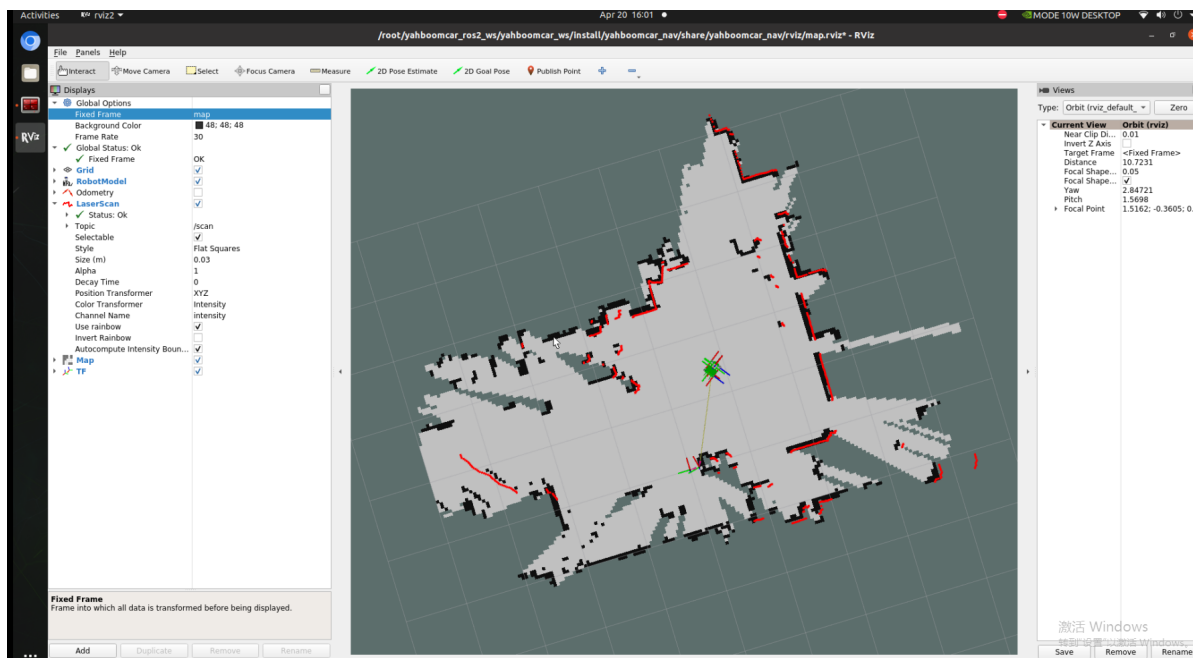
```
ros2 launch yahboomcar_nav display_map_launch.py
```



3. Start the keyboard control node. It is recommended to perform this step in a virtual machine. Multi-machine communication needs to be configured in the virtual machine.

Or use the remote control [slowly moving car] to start mapping until a complete map is created

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

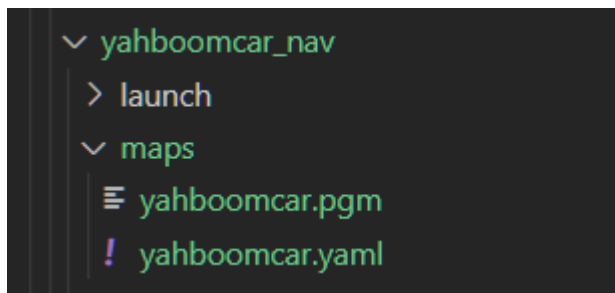


4. Save the map with the following path:

```
ros2 launch yahboomcar_nav save_map_launch.py
```

Save path is as follows:

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_nav/maps/
```



A pgm picture, a yaml file yahboomcar.yaml

```
image:
  /root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_nav/maps/yahboomcar.pgm
mode: trinary
resolution: 0.05
origin: [-10, -10, 0]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.25
```

Parameter analysis:

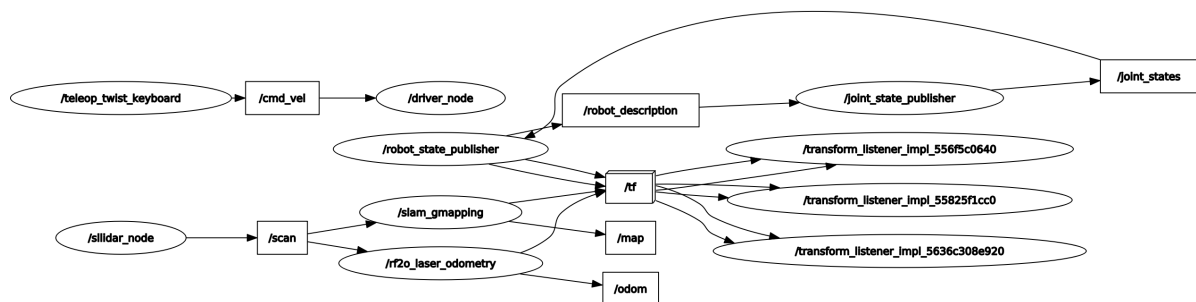
- image: The path of the map file, which can be an absolute path or a relative path.
- mode: This attribute can be one of trinary, scale or raw, depending on the selected mode. trinary mode is the default mode

- resolution: resolution of the map, meters/pixel
- Origin: 2D pose (x, y, yaw) in the lower left corner of the map. The yaw here is rotated counterclockwise (yaw=0 means no rotation). Many parts of the current system ignore the yaw value.
- negate: whether to reverse the meaning of white/black and free/occupied (the interpretation of the threshold is not affected)
- occupied\_thresh: Pixels with an occupation probability greater than this threshold will be considered fully occupied.
- free\_thresh: Pixels with occupancy probability less than this threshold will be considered completely free.

## 3. Node analysis

### 3.1. Display calculation graph

rqt\_graph



### 3.2. gmapping node details

```

yahboom@VM:~$ ros2 node info /slam_gmapping
/slam_gmapping
Subscribers:
  /parameter_events: rcl_interfaces/msg/ParameterEvent
  /scan: sensor_msgs/msg/LaserScan
Publishers:
  /entropy: std_msgs/msg/Float64
  /map: nav_msgs/msg/OccupancyGrid
  /map_metadata: nav_msgs/msg/MapMetaData
  /parameter_events: rcl_interfaces/msg/ParameterEvent
  /rosout: rcl_interfaces/msg/Log
  /tf: tf2_msgs/msg/TFMessage
Service Servers:
  /slam_gmapping/describe_parameters: rcl_interfaces/srv/DescribeParameters
  /slam_gmapping/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
  /slam_gmapping/get_parameters: rcl_interfaces/srv/GetParameters
  /slam_gmapping/list_parameters: rcl_interfaces/srv/ListParameters
  /slam_gmapping/set_parameters: rcl_interfaces/srv/SetParameters
  /slam_gmapping/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Service Clients:

Action Servers:

Action Clients:

```

```

yahboom@VM:~$ ros2 node info /rf2o_laser_odometry
There are 2 nodes in the graph with the exact name "/rf2o_laser_odometry". You are seeing information about only one of them.
/rf2o_laser_odometry
Subscribers:
  /parameter_events: rcl_interfaces/msg/ParameterEvent
  /scan: sensor_msgs/msg/LaserScan
Publishers:
  /odom: nav_msgs/msg/Odometry
  /parameter_events: rcl_interfaces/msg/ParameterEvent
  /rosout: rcl_interfaces/msg/Log
  /tf: tf2_msgs/msg/TFMessage
Service Servers:
  /rf2o_laser_odometry/describe_parameters: rcl_interfaces/srv/DescribeParameters
  /rf2o_laser_odometry/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
  /rf2o_laser_odometry/get_parameters: rcl_interfaces/srv/GetParameters
  /rf2o_laser_odometry/list_parameters: rcl_interfaces/srv/ListParameters
  /rf2o_laser_odometry/set_parameters: rcl_interfaces/srv/SetParameters
  /rf2o_laser_odometry/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Service Clients:

Action Servers:

Action Clients:

```

### 3.3. TF transformation

```
ros2 run tf2_tools view_frames.py
```

