6、gmapping mapping algorithm

6、gmapping mapping algorithm
- 6.1、Introduction
- 6.2、use
  - 6.2.1、pre-use configuration
  - 6.2.2、specific use
- 6.3、node parsing
  - 6.3.1、Display the calculation graph
  - 6.3.2、Details of gmapping nodes
  - 6.3.3、TF transformation

wiki：http://wiki.ros.org/gmapping/

ros2 gmapping：https://github.com/Project-MANAS/slam_gmapping

The operating environment and software and hardware reference configurations are as follows:

- REFERENCE MODEL: ROSMASTER X3
- Robot hardware configuration: Arm series main control, Silan A1 lidar, AstraPro Plus depth camera
- Robot system: Ubuntu (version not required) + docker (version 20.10.21 and above)
- PC Virtual Machine: Ubuntu (20.04) + ROS2 (Foxy)
- Usage scenario: Use on a relatively clean 2D plane

6.1、Introduction

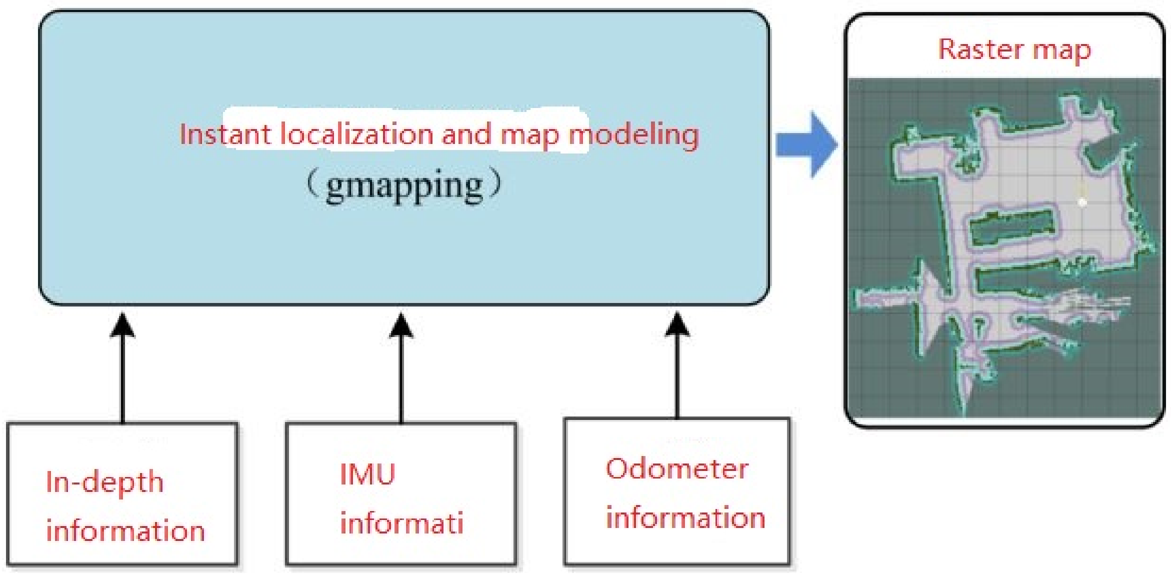
- gmapping is only suitable for points with a single frame two-dimensional laser point count less than 1440, if the number of laser points in a single frame is greater than 1440, then there will be such a problem as [mapping-4] process has died.
- Gmapping is a commonly used open source SLAM algorithm based on filtering SLAM frameworks.
- Based on the RBpf particle filter algorithm, Gmapping separates the real-time localization and mapping process, and locates before mapping.
- Gmapping makes two major improvements to the RBpf algorithm: improved proposed distribution and selective resampling.

Advantages: Gmapping can build indoor maps in real time, which requires less computation and high accuracy in building small scene maps.

Cons: The number of particles required increases as the scene grows, as each particle carries a map, hence the memory required to build a large map

The amount of computation increases. Therefore, it is not suitable for building large scene maps. And there is no loopback detection, so it may cause a map when the loopback is closed

Dislocation, although increasing the number of particles can close the map, but at the cost of increased computation and memory.



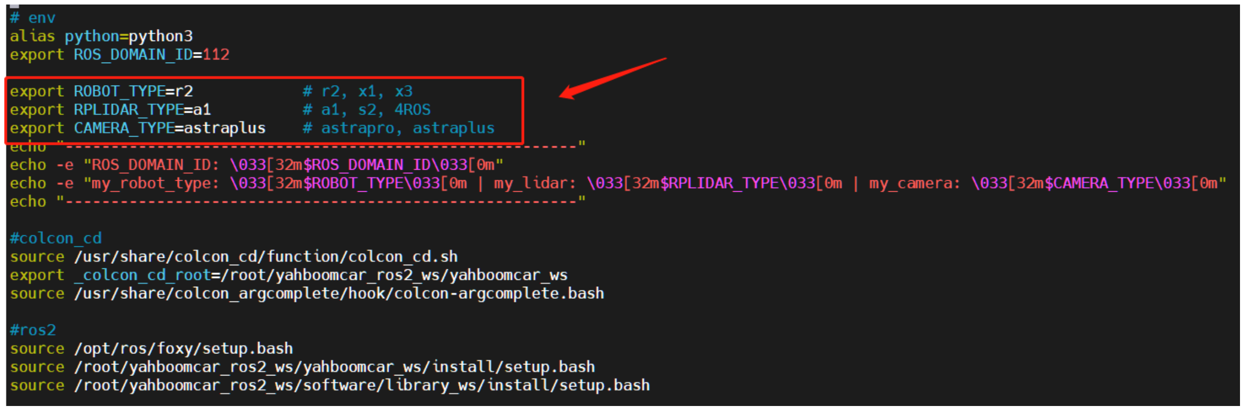
6.2、use

6.2.1、pre-use configuration

Note: Since ROSMASTER series robots are divided into multiple robots and multiple devices, the factory system has been configured with routines for multiple devices, but because the product cannot be automatically identified, it is necessary to manually set the machine type and radar model.

After entering the container: according to the model of the car, the type of radar and the type of camera, make the following modifications:

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



```
# env
alias python=python3
export ROS_DOMAIN_ID=112

export ROBOT_TYPE=r2            # r2, x1, x3
export RPLIDAR_TYPE=a1          # a1, s2, 4ROS
export CAMERA_TYPE=astraplus    # astrapro, astraplus
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$RPLIDAR_TYPE\033[0m | my_camera: \033[32m$CAMERA_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 complete, save and exit vim, and then execute:

```
root@ubuntu:~# source .bashrc
--------------------------------------------------------
ROS_DOMAIN_ID: 12
my_robot_type: x3 | my_lidar: a1 | my_camera: astraplus
--------------------------------------------------------
root@ubuntu:~#
```

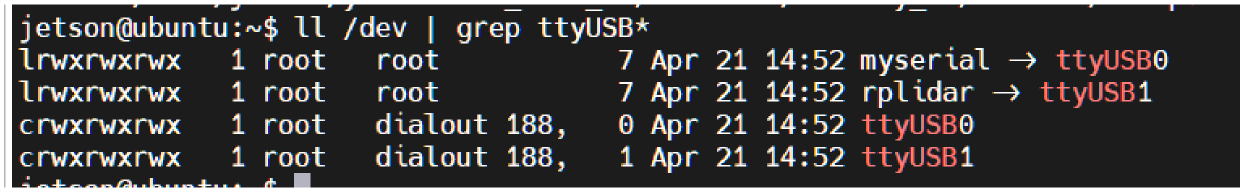
You can see the model of the currently modified car, the type of radar and the type of camera

6.2.2、specific use

Note: When building the image, the slower the effect, the better (note if the rotation speed is slower), too fast the effect, the effect will be poor.

First of all, you need to do port binding operations in the host [car] [see the port binding tutorial chapter], where two devices, radar and serial port, are mainly used;

Then check whether the radar and serial device are in the port binding state: on the host [car], refer to the following command to view, and the successful binding is in the following state:



```
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 the radar or serial device is not bound to the display, you can plug and unplug the USB cable to view it again.

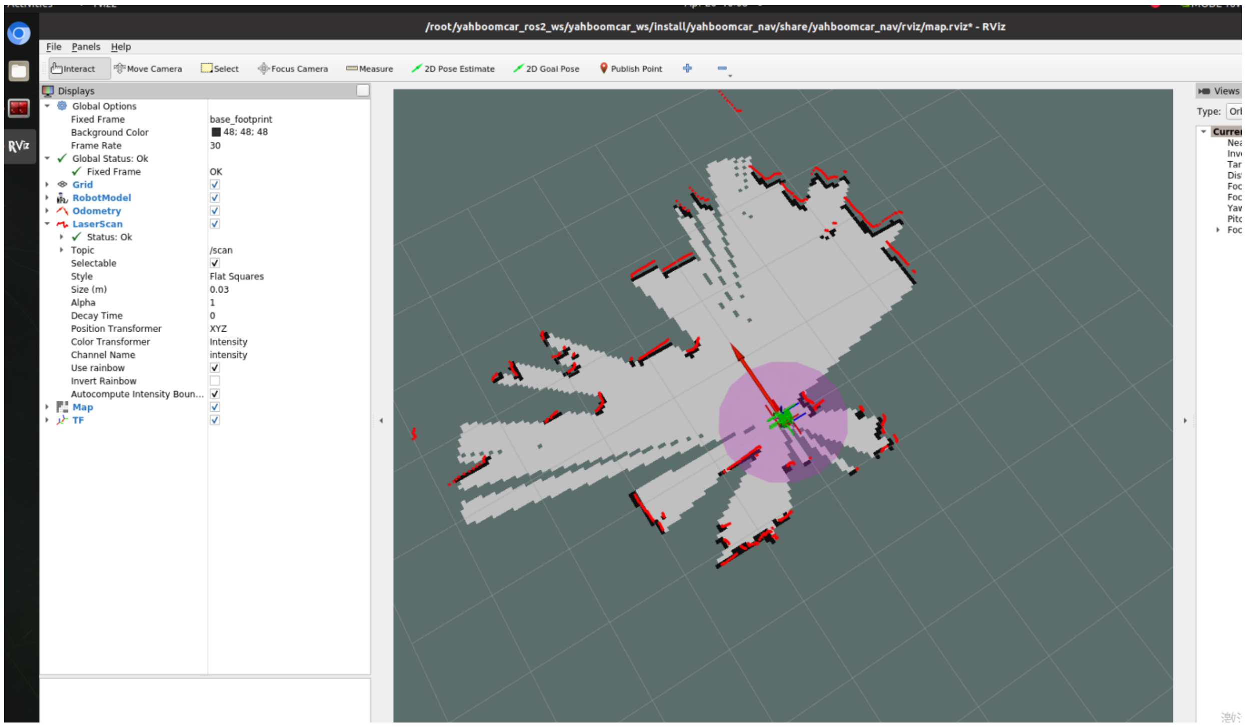
Enter the docker container (for steps, see [Docker course chapter ----- 5. Enter the docker container of the robot]), and execute the following launch file in the terminal:

1、Start mapping

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

2、Start rviz to display the map, this step is recommended to be executed in the virtual machine, and multi-machine communication needs to be configured in the virtual machine

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



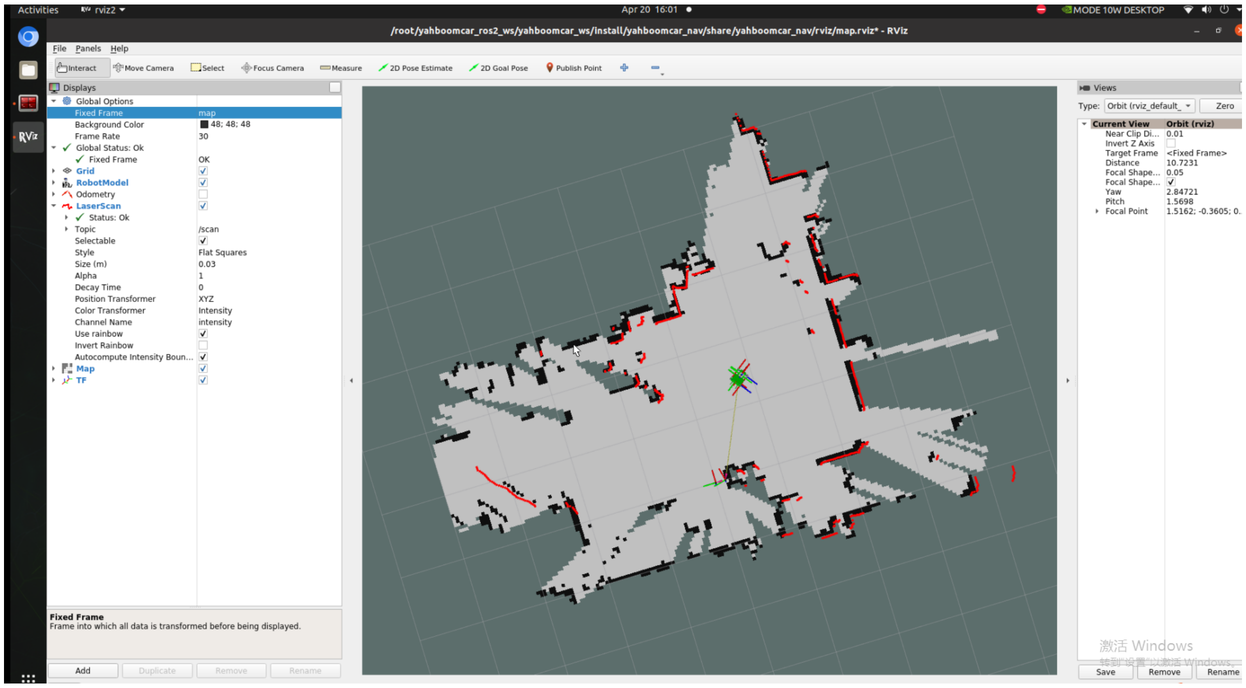
/root/yahboomcar_ros2_ws/yahboomcar_ws/install/yahboomcar_nav/share/yahboomcar_nav/rviz/map.rviz* - RViz

```
Displays
  Global Options
    Fixed Frame                       base_footprint
    Background Color                  48; 48; 48
    Frame Rate                        30
  Global Status: Ok
    Fixed Frame                       OK
  Grid                                ✓
  RobotModel                          ✓
  Odometry                            ✓
  LaserScan                           ✓
    Status: Ok
    Topic                             /scan
    Selectable                        ✓
    Style                             Flat Squares
    Size (m)                          0.03
    Alpha                             1
    Decay Time                        0
    Position Transformer              XYZ
    Color Transformer                 Intensity
    Channel Name                      intensity
    Use rainbow                       ✓
    Invert Rainbow
    Autocompute Intensity Boun...     ✓
  Map                                 ✓
  TF                                  ✓
```

3、Start the keyboard control node, this step is recommended to be executed in the virtual machine, and multi-machine communication needs to be configured in the virtual machine.

Or use the remote control [Move the trolley slowly] to start building the map until the complete map is built.

```
ros2 run yahboomcar_ctrl yahboom_keyboard
```



/root/yahboomcar_ros2_ws/yahboomcar_ws/install/yahboomcar_nav/share/yahboomcar_nav/rviz/map.rviz* - RViz

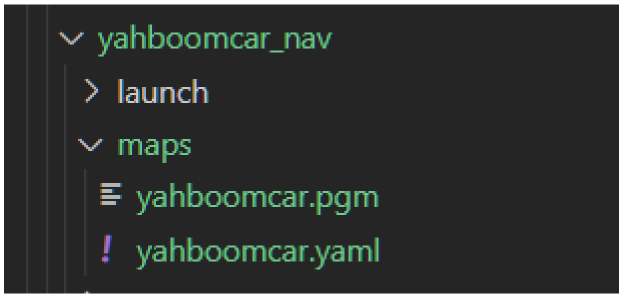
```
Displays
  Global Options
    Fixed Frame                       map
    Background Color                  48; 48; 48
    Frame Rate                        30
  Global Status: Ok
    Fixed Frame                       OK
  Grid                                ✓
  RobotModel                          ✓
  Odometry
  LaserScan                           ✓
    Status: Ok
    Topic                             /scan
    Selectable                        ✓
    Style                             Flat Squares
    Size (m)                          0.03
    Alpha                             1
    Decay Time                        0
    Position Transformer              XYZ
    Color Transformer                 Intensity
    Channel Name                      intensity
    Use rainbow                       ✓
    Invert Rainbow
    Autocompute Intensity Boun...     ✓
  Map                                 ✓
  TF                                  ✓

Fixed Frame
Frame into which all data is transformed before being displayed.

Views
Type: Orbit (rviz_default_
Current View          Orbit (rviz)
  Near Clip Di...     0.01
  Invert Z Axis
  Target Frame        <Fixed Frame>
  Distance            10.7231
  Focal Shape...      0.05
  Focal Shape...      ✓
  Yaw                 2.84721
  Pitch               1.5698
  Focal Point         1.5162; -0.3605; 0.
```

4、Save the map, the path is as follows:

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

The save path is as follows:

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


yahboomcar_nav
launch
maps
yahboomcar.pgm
yahboomcar.yaml

A PGM image, 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
```

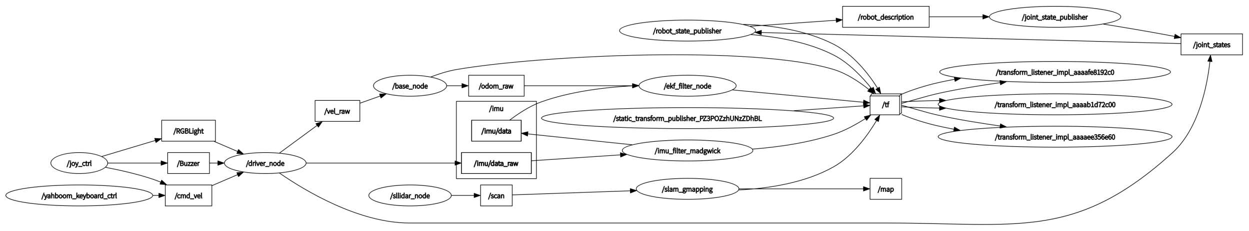
Parameter parsing:

- image: The path to the map file, which can be absolute or relative
- mode: This attribute can be one of trinary, scale, or raw, depending on the mode selected, trinary mode is the default mode
- Resolution: The resolution of the map, meters per pixel
- origin: 2D pose (x,y,yaw) in the lower left corner of the map, where yaw is rotated counterclockwise (yaw=0 means no rotation). Many parts of the system today ignore the YAW value.
- negate: whether to reverse the meaning of white/black, free/occupy (the interpretation of the threshold is not affected)
- occupied_thresh: Pixels with a probability of occupancy greater than this threshold are considered fully occupied.
- free_thresh: Pixels with a probability of occupancy less than this threshold are considered completely free.

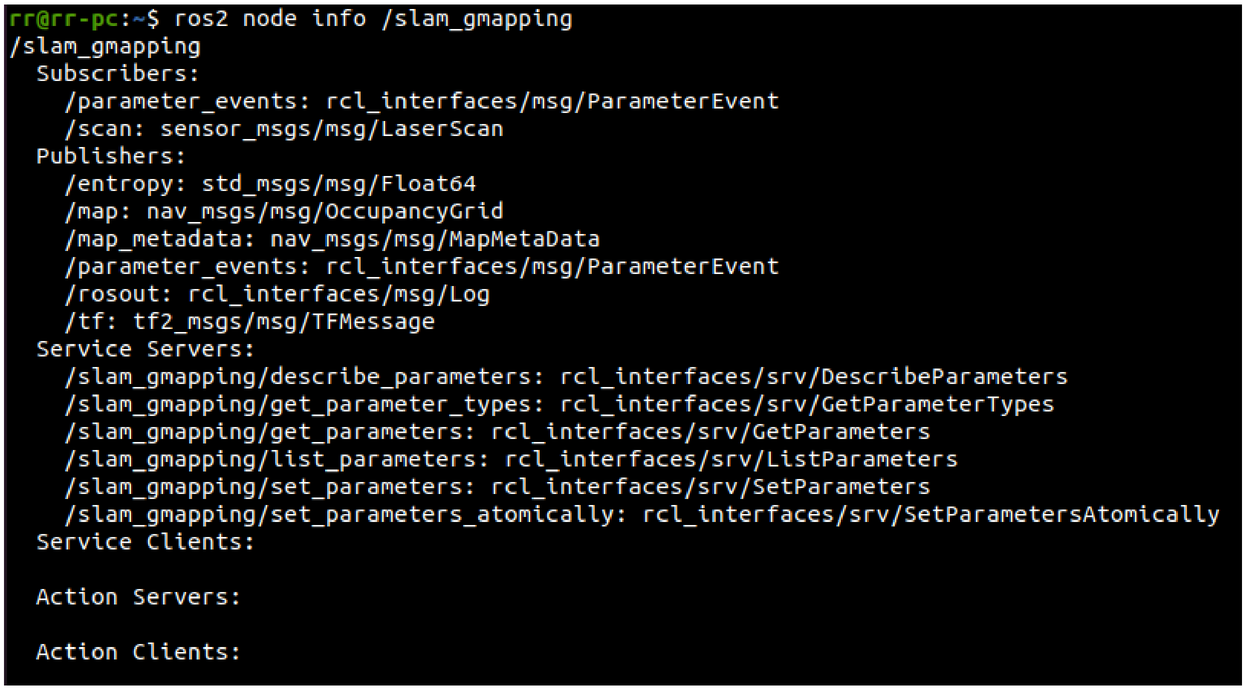
6.3、node parsing

6.3.1、Display the calculation graph

```
rqt_graph
```



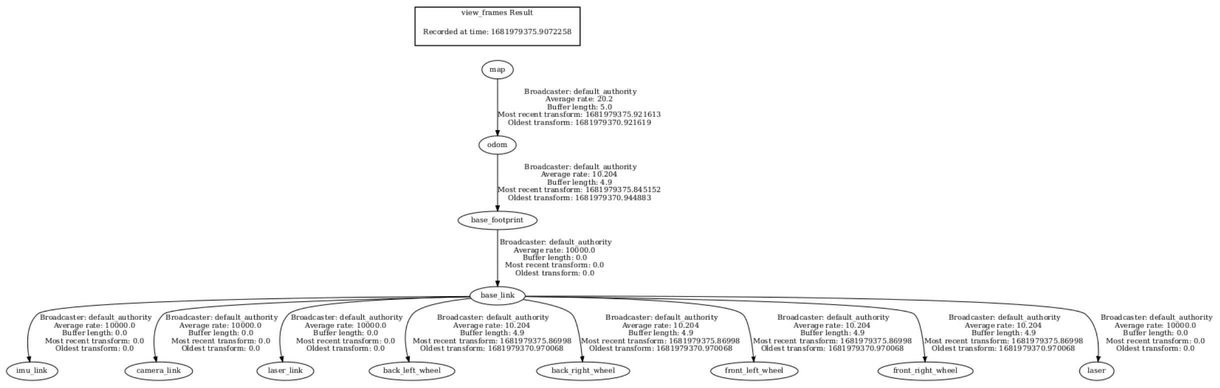
6.3.2、Details of gmapping nodes



```
rr@rr-pc:~$ 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:
```

6.3.3、TF transformation

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


view_frames Result
Recorded at time: 1681979375.9072258
map
Broadcaster: default_authority
Average rate: 20.2
Buffer length: 5.0
Most recent transform: 1681979375.921613
Oldest transform: 1681979370.921619
odom
Broadcaster: default_authority
Average rate: 10.204
Buffer length: 4.9
Most recent transform: 1681979375.845152
Oldest transform: 1681979370.944883
base_footprint
Broadcaster: default_authority
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
base_link
Broadcaster: default_authority
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
Broadcaster: default_authority
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
Broadcaster: default_authority
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
Broadcaster: default_authority
Average rate: 10.204
Buffer length: 4.9
Most recent transform: 1681979375.86998
Oldest transform: 1681979370.970068
Broadcaster: default_authority
Average rate: 10.204
Buffer length: 4.9
Most recent transform: 1681979375.86998
Oldest transform: 1681979370.970068
Broadcaster: default_authority
Average rate: 10.204
Buffer length: 4.9
Most recent transform: 1681979375.86998
Oldest transform: 1681979370.970068
Broadcaster: default_authority
Average rate: 10.204
Buffer length: 4.9
Most recent transform: 1681979375.86998
Oldest transform: 1681979370.970068
Broadcaster: default_authority
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
imu_link
camera_link
laser_link
back_left_wheel
back_right_wheel
front_left_wheel
front_right_wheel
laser