Gmapping Mapping

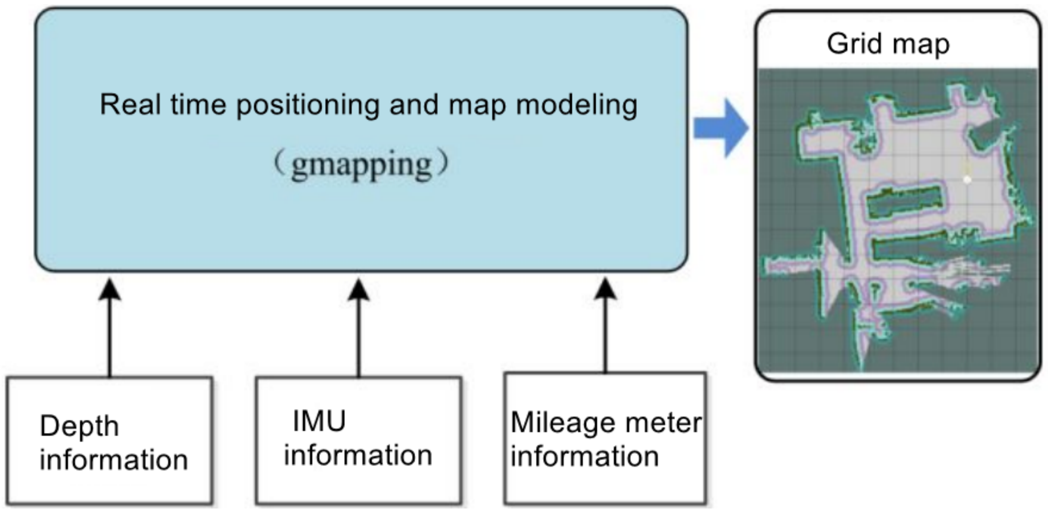
Note: The virtual machine needs to be in the same LAN as the car, and the ROS_DOMAIN_ID needs to be consistent. You can check [Must Read Before Use] to set the IP and ROS_DOMAIN_ID on the board.

1. Introduction to Gmapping

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

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

Disadvantages: As the scene increases, the number of particles required increases. Because each particle carries a map, the memory and calculation required to build a large map will increase. 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 make the map closed, it comes at the cost of increasing the amount of calculation and memory.


Real time positioning and map modeling
(gmapping)
Grid map
Depth information
IMU information
Mileage meter information

2. Program function description

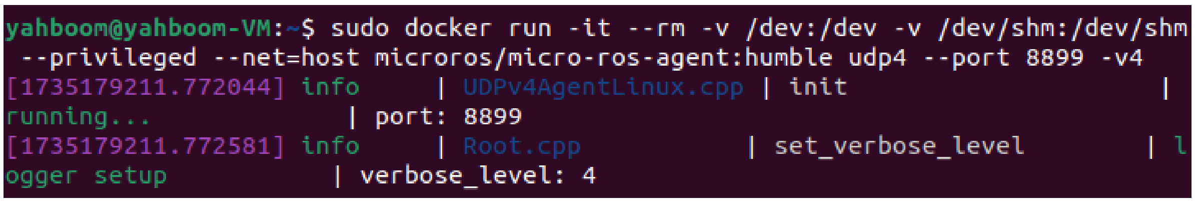
The car connects to the proxy and runs the program. The map building interface will be displayed in rviz. Use the keyboard or handle to control the movement of the car until the map is built. Then run the command to save the map.

```
Note: Before running the program, the car needs to be restarted in a stable
standing position to ensure that all sensors are reset
```

3. Start and connect the agent

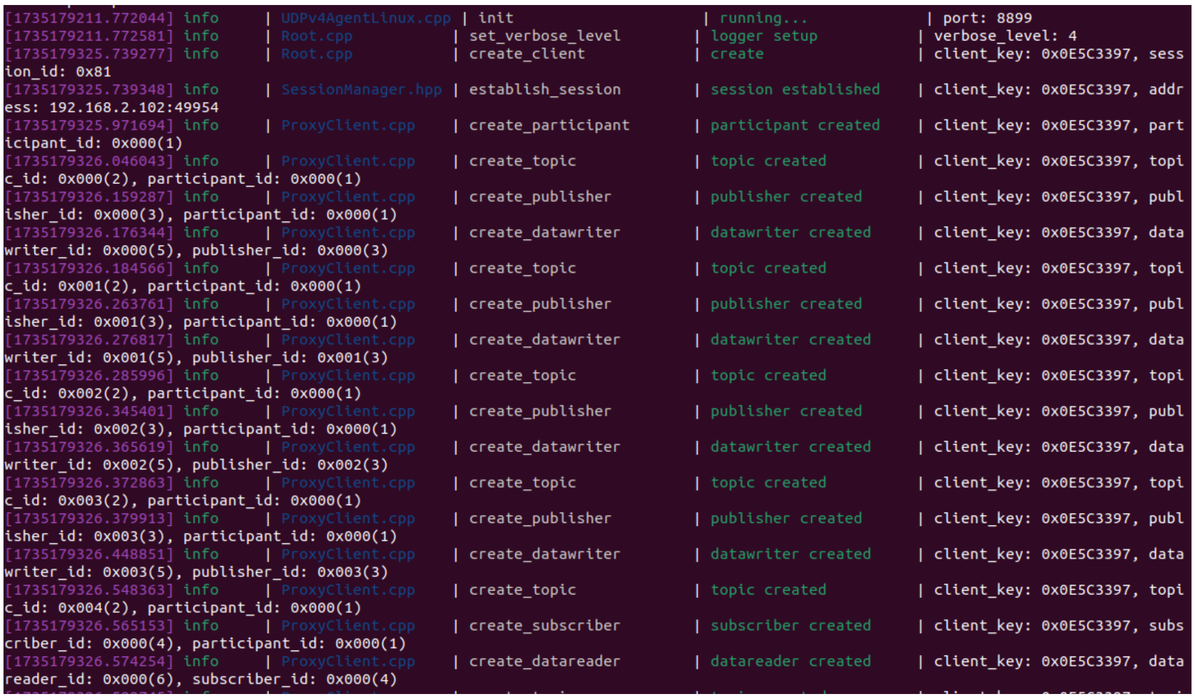
Take the supporting virtual machine as an example, enter the following command to start the agent,

```
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --
net=host microros/micro-ros-agent:humble udp4 --port 8899 -v4
```



```
yahboom@yahboom-VM:~$ sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm
 --privileged --net=host microros/micro-ros-agent:humble udp4 --port 8899 -v4
[1735179211.772044] info     | UDPv4AgentLinux.cpp | init                  |
running...             | port: 8899
[1735179211.772581] info     | Root.cpp            | set_verbose_level     | l
ogger setup            | verbose_level: 4
```

Then, turn on the car switch and wait for the car to connect to the proxy. The connection is successful as shown in the figure below.



```
[1735179211.772044] info     | UDPv4AgentLinux.cpp | init                   | running...             | port: 8899
[1735179211.772581] info     | Root.cpp            | set_verbose_level      | logger setup           | verbose_level: 4
[1735179325.739277] info     | Root.cpp            | create_client          | create                 | client_key: 0x0E5C3397, sess
ion_id: 0x81
[1735179325.739348] info     | SessionManager.hpp | establish_session      | session established    | client_key: 0x0E5C3397, addr
ess: 192.168.2.102:49954
[1735179325.971694] info     | ProxyClient.cpp     | create_participant     | participant created    | client_key: 0x0E5C3397, part
icipant_id: 0x000(1)
[1735179326.046043] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x000(2), participant_id: 0x000(1)
[1735179326.159287] info     | ProxyClient.cpp     | create_publisher       | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x000(3), participant_id: 0x000(1)
[1735179326.176344] info     | ProxyClient.cpp     | create_datawriter      | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x000(5), publisher_id: 0x000(3)
[1735179326.184566] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x001(2), participant_id: 0x000(1)
[1735179326.263761] info     | ProxyClient.cpp     | create_publisher       | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x001(3), participant_id: 0x000(1)
[1735179326.276817] info     | ProxyClient.cpp     | create_datawriter      | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x001(5), publisher_id: 0x001(3)
[1735179326.285996] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x002(2), participant_id: 0x000(1)
[1735179326.345401] info     | ProxyClient.cpp     | create_publisher       | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x002(3), participant_id: 0x000(1)
[1735179326.365619] info     | ProxyClient.cpp     | create_datawriter      | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x002(5), publisher_id: 0x002(3)
[1735179326.372863] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x003(2), participant_id: 0x000(1)
[1735179326.379913] info     | ProxyClient.cpp     | create_publisher       | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x003(3), participant_id: 0x000(1)
[1735179326.448851] info     | ProxyClient.cpp     | create_datawriter      | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x003(5), publisher_id: 0x003(3)
[1735179326.548363] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x004(2), participant_id: 0x000(1)
[1735179326.565153] info     | ProxyClient.cpp     | create_subscriber      | subscriber created     | client_key: 0x0E5C3397, subs
criber_id: 0x000(4), participant_id: 0x000(1)
[1735179326.574254] info     | ProxyClient.cpp     | create_datareader      | datareader created     | client_key: 0x0E5C3397, data
reader_id: 0x000(6), subscriber_id: 0x000(4)
```

4. Start the program

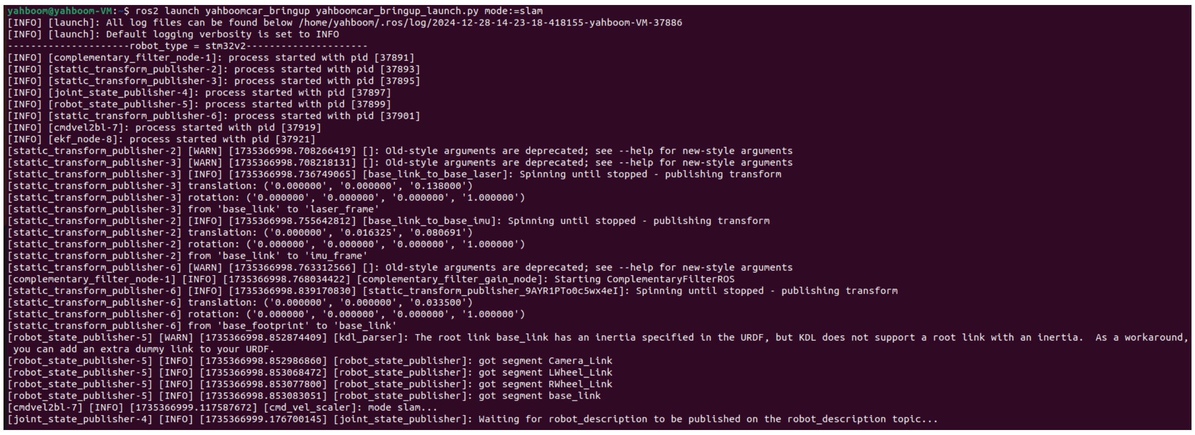
4.1 Run the command

Take the matching virtual machine as an example, enter in the terminal,

```
ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py mode:=slam
```

```
#Parameter description, the speed of the car will be adjusted. This is the map
building mode
mode:=slam
```

First, start the car to process the underlying data program.

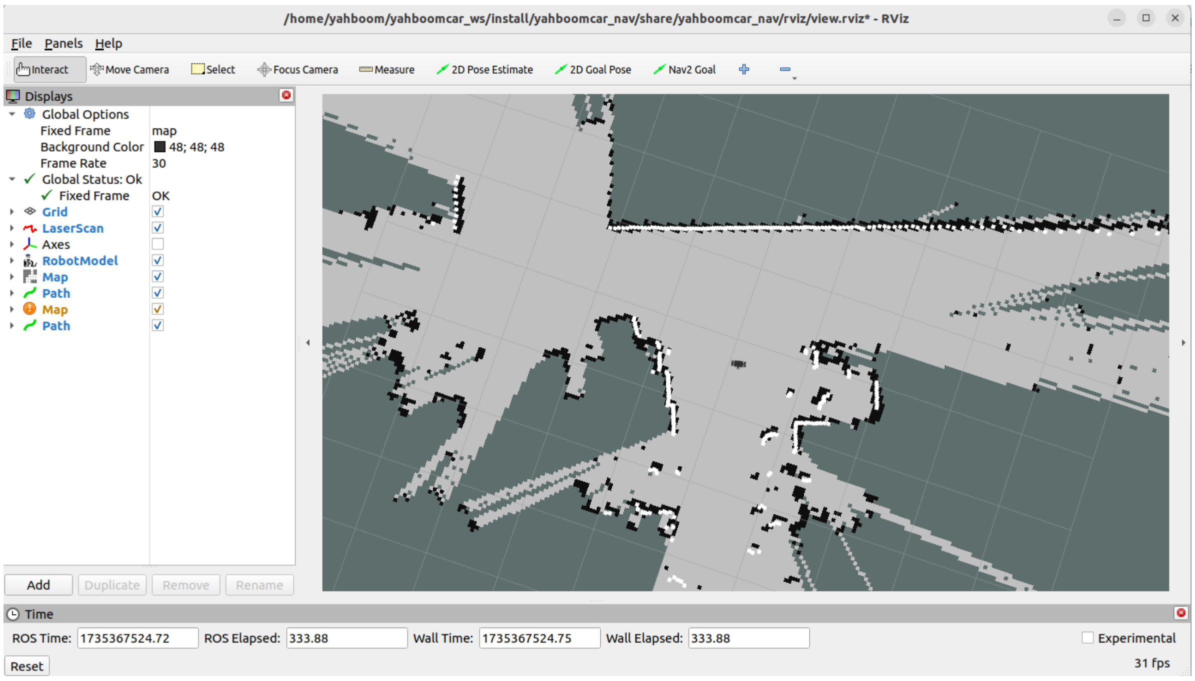


```
yahboom@yahboom-VM:~$ ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py mode:=slam
[INFO] [launch]: All log files can be found below /home/yahboom/.ros/log/2024-12-28-14-23-18-418155-yahboom-VM-37886
[INFO] [launch]: Default logging verbosity is set to INFO
---------------------robot_type = stm32v2--------------------
[INFO] [complementary_filter_node-1]: process started with pid [37891]
[INFO] [static_transform_publisher-2]: process started with pid [37893]
[INFO] [static_transform_publisher-3]: process started with pid [37895]
[INFO] [joint_state_publisher-4]: process started with pid [37897]
[INFO] [robot_state_publisher-5]: process started with pid [37899]
[INFO] [static_transform_publisher-6]: process started with pid [37901]
[INFO] [cmdvel2bl-7]: process started with pid [37919]
[INFO] [ekf_node-8]: process started with pid [37921]
[static_transform_publisher-2] [WARN] [1735366998.708266419] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-3] [WARN] [1735366998.708218131] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-3] [INFO] [1735366998.736749065] [base_link_to_base_laser]: Spinning until stopped - publishing transform
[static_transform_publisher-3] translation: ('0.000000', '0.000000', '0.138000')
[static_transform_publisher-3] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-3] from 'base_link' to 'laser_frame'
[static_transform_publisher-2] [INFO] [1735366998.755642812] [base_link_to_base_imu]: Spinning until stopped - publishing transform
[static_transform_publisher-2] translation: ('0.000000', '0.016325', '0.080691')
[static_transform_publisher-2] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-2] from 'base_link' to 'imu_frame'
[static_transform_publisher-6] [WARN] [1735366998.763312566] []: Old-style arguments are deprecated; see --help for new-style arguments
[complementary_filter_node-1] [INFO] [1735366998.768034422] [complementary_filter_gain_node]: Starting ComplementaryFilterROS
[static_transform_publisher-6] [INFO] [1735366998.839170830] [static_transform_publisher_9AYR1PTo0c5wx4eI]: Spinning until stopped - publishing transform
[static_transform_publisher-6] translation: ('0.000000', '0.000000', '0.033500')
[static_transform_publisher-6] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-6] from 'base_footprint' to 'base_link'
[robot_state_publisher-5] [WARN] [1735366998.852874409] [kdl_parser]: The root link base_link has an inertia specified in the URDF, but KDL does not support a root link with an inertia.  As a workaround,
 you can add an extra dummy link to your URDF.
[robot_state_publisher-5] [INFO] [1735366998.852986860] [robot_state_publisher]: got segment Camera_Link
[robot_state_publisher-5] [INFO] [1735366998.853068472] [robot_state_publisher]: got segment LWheel_Link
[robot_state_publisher-5] [INFO] [1735366998.853077800] [robot_state_publisher]: got segment RWheel_Link
[robot_state_publisher-5] [INFO] [1735366998.853083051] [robot_state_publisher]: got segment base_link
[cmdvel2bl-7] [INFO] [1735366999.117587672] [cmd_vel_scaler]: mode slam...
[joint_state_publisher-4] [INFO] [1735366999.176700145] [joint_state_publisher]: Waiting for robot_description to be published on the robot_description topic...
```

Next, run the graph building node, input in the terminal, and wait for rviz to open automatically.

```
ros2 launch yahboomcar_nav map_gmapping_launch.py use_rviz:=true
```

```
#Parameter description, select whether to open rviz, true means open, false means
not open
use_rviz:=true
```



/home/yahboom/yahboomcar_ws/install/yahboomcar_nav/share/yahboomcar_nav/rviz/view.rviz* - RViz

Fixed Frame: map
Background Color: 48; 48; 48
Frame Rate: 30
Global Status: Ok
Fixed Frame: OK

ROS Time: 1735367524.72 ROS Elapsed: 333.88 Wall Time: 1735367524.75 Wall Elapsed: 333.88

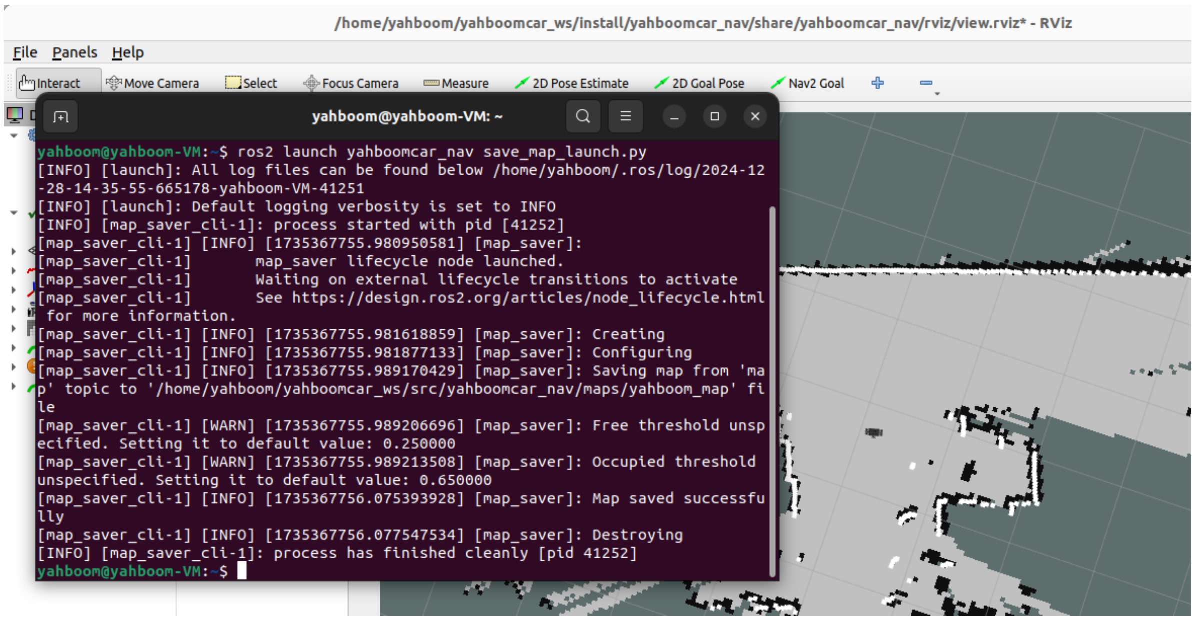
31 fps

Then run handle control or keyboard control, choose one of the two, terminal input,

```
#Keyboard Control
ros2 run yahboomcar_ctrl yahboom_keyboard
#handle control
ros2 launch yahboomcar_ctrl yahboomcar_joy_launch.py
```

Then control the car to slowly walk through the area that needs to be mapped. After the map is built, enter the following command to save the map. Enter in the terminal:

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



```
yahboom@yahboom-VM:~$ ros2 launch yahboomcar_nav save_map_launch.py
[INFO] [launch]: All log files can be found below /home/yahboom/.ros/log/2024-12
-28-14-35-55-665178-yahboom-VM-41251
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [map_saver_cli-1]: process started with pid [41252]
[map_saver_cli-1] [INFO] [1735367755.980950581] [map_saver]:
[map_saver_cli-1]        map_saver lifecycle node launched.
[map_saver_cli-1]        Waiting on external lifecycle transitions to activate
[map_saver_cli-1]        See https://design.ros2.org/articles/node_lifecycle.html
 for more information.
[map_saver_cli-1] [INFO] [1735367755.981618859] [map_saver]: Creating
[map_saver_cli-1] [INFO] [1735367755.981877133] [map_saver]: Configuring
[map_saver_cli-1] [INFO] [1735367755.989170429] [map_saver]: Saving map from 'ma
p' topic to '/home/yahboom/yahboomcar_ws/src/yahboomcar_nav/maps/yahboom_map' fi
le
[map_saver_cli-1] [WARN] [1735367755.989206696] [map_saver]: Free threshold unsp
ecified. Setting it to default value: 0.250000
[map_saver_cli-1] [WARN] [1735367755.989213508] [map_saver]: Occupied threshold
unspecified. Setting it to default value: 0.650000
[map_saver_cli-1] [INFO] [1735367756.075393928] [map_saver]: Map saved successfu
lly
[map_saver_cli-1] [INFO] [1735367756.077547534] [map_saver]: Destroying
[INFO] [map_saver_cli-1]: process has finished cleanly [pid 41252]
yahboom@yahboom-VM:~$ 
```

A map named yahboom_map will be saved. This map is saved in,

Take the matching virtual machine as an example code path:

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_nav/maps
```

There will be two files generated, one is yahboom_map.pgm, the other is yahboom_map.yaml, look at the contents of yaml,

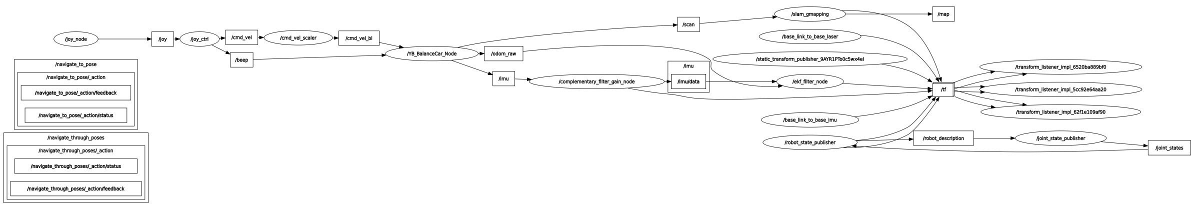
```yaml
image: yahboom_map.pgm
mode: trinary
resolution: 0.05
origin: [-10, -10, 0]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.25
```

- image: the image representing the map, i.e. yahboom_map.pgm
- mode: this property can be one of trinary, scale or raw, depending on the selected mode, trinary mode is the default mode
- resolution: the resolution of the map, meters/pixels
- origin: the 2D pose (x,y,yaw) of the lower left corner of the map, where yaw is rotated counterclockwise (yaw=0 means no rotation). Currently many parts of the system ignore the yaw value.
- negate: whether to invert the meaning of white/black and free/occupied (the interpretation of the threshold is not affected)
- occupied_thresh: pixels with an occupied probability greater than this threshold are considered fully occupied.
- free_thresh: pixels with an occupied probability less than this threshold are considered fully free.

5. View the node communication diagram

Terminal input,

```
ros2 run rqt_graph rqt_graph
```

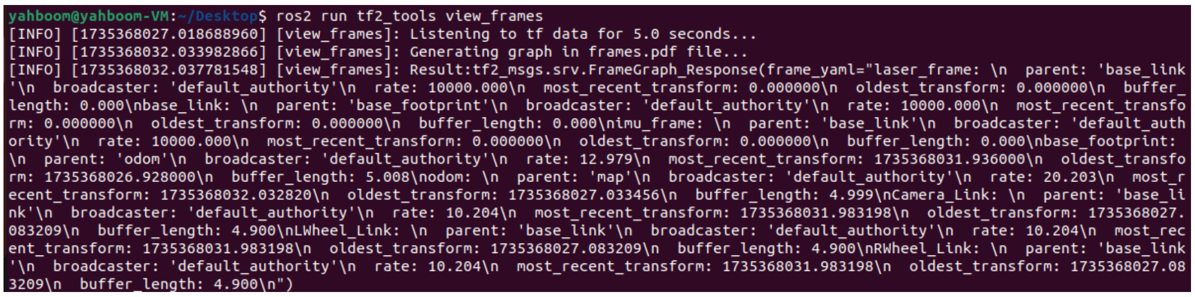

/joy_node
/joy
/joy_ctrl
/cmd_vel
/cmd_vel_scaler
/cmd_vel_bl
/beep
/YB_BalanceCar_Node
/odom_raw
/imu
/complementary_filter_gain_node
/imu
/imu/data
/scan
/slam_gmapping
/map
/base_link_to_base_laser
/static_transform_publisher_9AYR1PTo0c5wx4el
/ekf_filter_node
/base_link_to_base_imu
/robot_state_publisher
/robot_description
/joint_state_publisher
/joint_states
/tf
/transform_listener_impl_6520ba889bf0
/transform_listener_impl_5cc92e64aa20
/transform_listener_impl_62f1e109af90
/navigate_to_pose
/navigate_to_pose/_action
/navigate_to_pose/_action/feedback
/navigate_to_pose/_action/status
/navigate_through_poses
/navigate_through_poses/_action
/navigate_through_poses/_action/status
/navigate_through_poses/_action/feedback

If it is not displayed at first, select [Nodes/Topics (all)] and click the refresh button in the upper left corner.

6. View TF tree

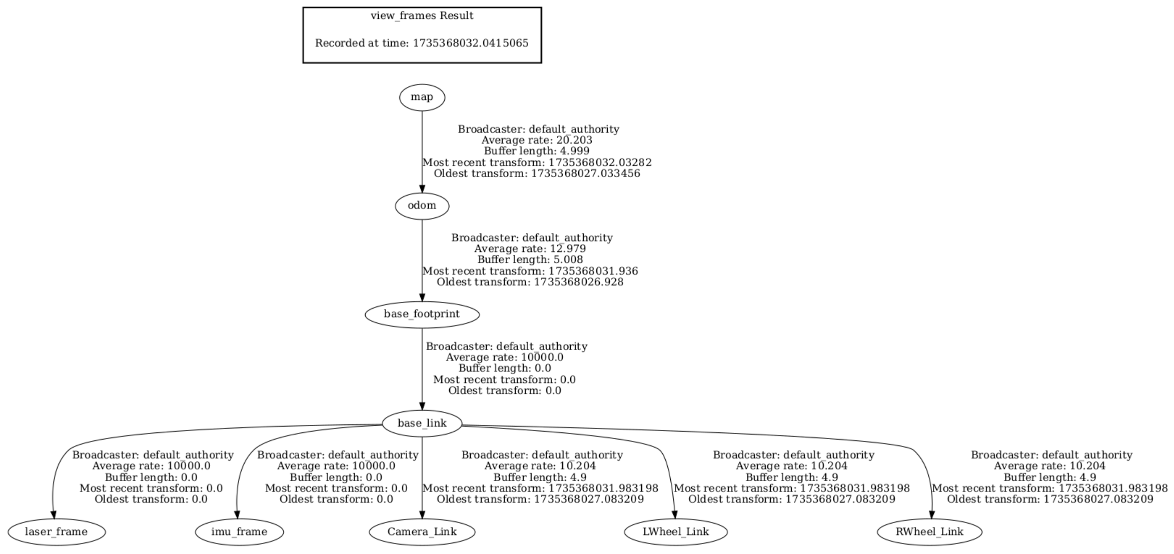
Terminal input,

```
ros2 run tf2_tools view_frames
```



```
yahboom@yahboom-VM:~/Desktop$ ros2 run tf2_tools view_frames
[INFO] [1735368027.018688960] [view_frames]: Listening to tf data for 5.0 seconds...
[INFO] [1735368032.033982866] [view_frames]: Generating graph in frames.pdf file...
[INFO] [1735368032.037781548] [view_frames]: Result:tf2_msgs.srv.FrameGraph_Response(frame_yaml="laser_frame: \n  parent: 'base_link
'\n  broadcaster: 'default_authority'\n  rate: 10000.000\n  most_recent_transform: 0.000000\n  oldest_transform: 0.000000\n  buffer_
length: 0.000\nbase_link: \n  parent: 'base_footprint'\n  broadcaster: 'default_authority'\n  rate: 10000.000\n  most_recent_transfo
rm: 0.000000\n  oldest_transform: 0.000000\n  buffer_length: 0.000\nimu_frame: \n  parent: 'base_link'\n  broadcaster: 'default_auth
ority'\n  rate: 10000.000\n  most_recent_transform: 0.000000\n  oldest_transform: 0.000000\n  buffer_length: 0.000\nbase_footprint:
\n  parent: 'odom'\n  broadcaster: 'default_authority'\n  rate: 12.979\n  most_recent_transform: 1735368031.936000\n  oldest_transfo
rm: 1735368026.928000\n  buffer_length: 5.008\nodom: \n  parent: 'map'\n  broadcaster: 'default_authority'\n  rate: 20.203\n  most_r
ecent_transform: 1735368032.032820\n  oldest_transform: 1735368027.033456\n  buffer_length: 4.999\nCamera_Link: \n  parent: 'base_li
nk'\n  broadcaster: 'default_authority'\n  rate: 10.204\n  most_recent_transform: 1735368031.983198\n  oldest_transform: 1735368027.
083209\n  buffer_length: 4.900\nLWheel_Link: \n  parent: 'base_link'\n  broadcaster: 'default_authority'\n  rate: 10.204\n  most_rec
ent_transform: 1735368031.983198\n  oldest_transform: 1735368027.083209\n  buffer_length: 4.900\nRWheel_Link: \n  parent: 'base_link
'\n  broadcaster: 'default_authority'\n  rate: 10.204\n  most_recent_transform: 1735368031.983198\n  oldest_transform: 1735368027.08
3209\n  buffer_length: 4.900\n")
```

After the execution is complete, two files, .gv and .pdf, will be generated in the terminal directory. The pdf file is the TF tree.


view_frames Result
Recorded at time: 1735368032.0415065
map
Broadcaster: default_authority
Average rate: 20.203
Buffer length: 4.999
Most recent transform: 1735368032.03282
Oldest transform: 1735368027.033456
odom
Broadcaster: default_authority
Average rate: 12.979
Buffer length: 5.008
Most recent transform: 1735368031.936
Oldest transform: 1735368026.928
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: 10.204
Buffer length: 4.9
Most recent transform: 1735368031.983198
Oldest transform: 1735368027.083209
Broadcaster: default_authority
Average rate: 10.204
Buffer length: 4.9
Most recent transform: 1735368031.983198
Oldest transform: 1735368027.083209
Broadcaster: default_authority
Average rate: 10.204
Buffer length: 4.9
Most recent transform: 1735368031.983198
Oldest transform: 1735368027.083209
laser_frame
imu_frame
Camera_Link
LWheel_Link
RWheel_Link

7. Code analysis

Take the virtual machine as an example, here we only explain the map_gmapping_launch.py file for building the map. The file path is,

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_nav/launch
```

map_gmapping_launch.py

```
from launch.substitutions import LaunchConfiguration
from launch import LaunchDescription
from launch_ros.actions import Node
from launch.actions import IncludeLaunchDescription,DeclareLaunchArgument
from launch.launch_description_sources import PythonLaunchDescriptionSource
from ament_index_python.packages import
get_package_share_directory,get_package_share_path
import os

def generate_launch_description():

    package_path = get_package_share_path('yahboomcar_nav')
    default_rviz_config_path = package_path / 'rviz/view.rviz'
    rviz_arg = DeclareLaunchArgument(name='rvizconfig',
default_value=str(default_rviz_config_path),
                                     description='Absolute path to rviz config
file')

    slam_gmapping_launch = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([os.path.join(
        get_package_share_directory('slam_gmapping'), 'launch'),
         '/slam_gmapping.launch.py'])
    )

    base_link_to_laser_tf_node = Node(
     package='tf2_ros',
     executable='static_transform_publisher',
     name='base_link_to_base_laser',
     arguments=['0.0', '0.0', '0.138', '0', '0', '0', 'base_link',
'laser_frame']
    )

    rviz_node = Node(
        package='rviz2',
        executable='rviz2',
        name='rviz2',
        output='screen',
        arguments=['-d', LaunchConfiguration('rvizconfig')],
    )

    return
LaunchDescription([rviz_arg,rviz_node,slam_gmapping_launch,base_link_to_laser_tf
_node])
```

Here, a launch file - slam_gmapping_launch and a node that publishes static transformations - base_link_to_laser_tf_node are started.

Take the virtual machine as an example, and take a closer look at slam_gmapping_launch. The file is located at,

```
/home/yahboom/gmapping_ws/src/slam_gmapping/launch
```

slam_gmapping.launch.py,

```
from launch import LaunchDescription
from launch.substitutions import EnvironmentVariable
import launch.actions
import launch_ros.actions
import os
from ament_index_python.packages import get_package_share_directory

def generate_launch_description():
    return LaunchDescription([
        launch_ros.actions.Node(
            package='slam_gmapping',
            executable='slam_gmapping',
            output='screen',
            parameters=
[os.path.join(get_package_share_directory("slam_gmapping"), "params",
"slam_gmapping1.yaml")]),
    ])
```

Here, the slam_gmapping node is started and the slam_gmapping1.yaml parameter file is loaded.

The parameter file is located in (taking the matching virtual machine as an example),

```
/home/yahboom/gmapping_ws/src/slam_gmapping/params
```

slam_gmapping1.yaml

```
/slam_gmapping:
  ros__parameters:
    angularUpdate: 1.0
    astep: 0.10
    base_frame: base_footprint
    map_frame: map
    odom_frame: odom
    delta: 0.05
    iterations: 5
    kernelSize: 1
    lasamplerange: 0.005
    lasamplestep: 0.005
    linearUpdate: 0.2
    llsamplerange: 0.01
    llsamplestep: 0.01
    lsigma: 0.075
    lskip: 2
    lstep: 0.10
    map_update_interval: 0.3
    maxRange: 6.0
    maxUrange: 4.0
    minimum_score: 0.0
    occ_thresh: 0.25
    ogain: 3.0
    particles: 15
    qos_overrides:
      /parameter_events:
        publisher:
          depth: 1000
          durability: volatile
          history: keep_all
          reliability: reliable
      /tf:
        publisher:
          depth: 1000
          durability: volatile
          history: keep_last
          reliability: reliable
    resampleThreshold: 0.5
    sigma: 0.05
    srr: 0.2
    srt: 0.0
    str: 0.0
    stt: 0.0
    temporalUpdate: 1.0
    transform_publish_period: 0.05
    use_sim_time: false
    xmax: 20.0
    xmin: -20.0
    ymax: 20.0
    ymin: -20.0
```