Cartographer

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 [Expansion Board Parameter Configuration] to set the IP and ROS_DOMAIN_ID on the board.

1. Introduction to Cartographer

Cartographer is a 2D and 3D SLAM (simultaneous localization and mapping) library supported by Google's open source ROS system. The mapping algorithm is based on graph optimization (multi-threaded backend optimization, problem optimization built by cere). It can combine data from multiple sensors (such as LIDAR, IMU and camera) to synchronously calculate the position of the sensor and draw the environment around the sensor.

The source code of cartographer mainly includes three parts: cartographer, cartographer_ros and ceres-solver (backend optimization).

Cartographer uses the mainstream SLAM framework, which is a three-stage feature extraction, closed-loop detection, and backend optimization. A certain number of LaserScans form a submap, and a series of submaps form a global map. The short-term cumulative error of using LaserScan to build submaps is not large, but the long-term process of using submaps to build a global map will have a large cumulative error, so closed-loop detection is needed to correct the position of these submaps. The basic unit of closed-loop detection is submap, and closed-loop detection uses scan_match strategy.

The focus of cartographer is the creation of submaps that integrate multi-sensor data (odometry, IMU, LaserScan, etc.) and the implementation of scan_match strategy for closed-loop detection.

cartographer_ros runs under ROS and can receive various sensor data in the form of ROS messages. After processing, it is published in the form of messages for easy debugging and visualization.

2. Program Function Description

The car connects to the agent and runs the program. The map building interface will be displayed in rviz. The handheld radar moves in parallel. This algorithm needs to move slowly until the map is built. Then run the command to save the map.

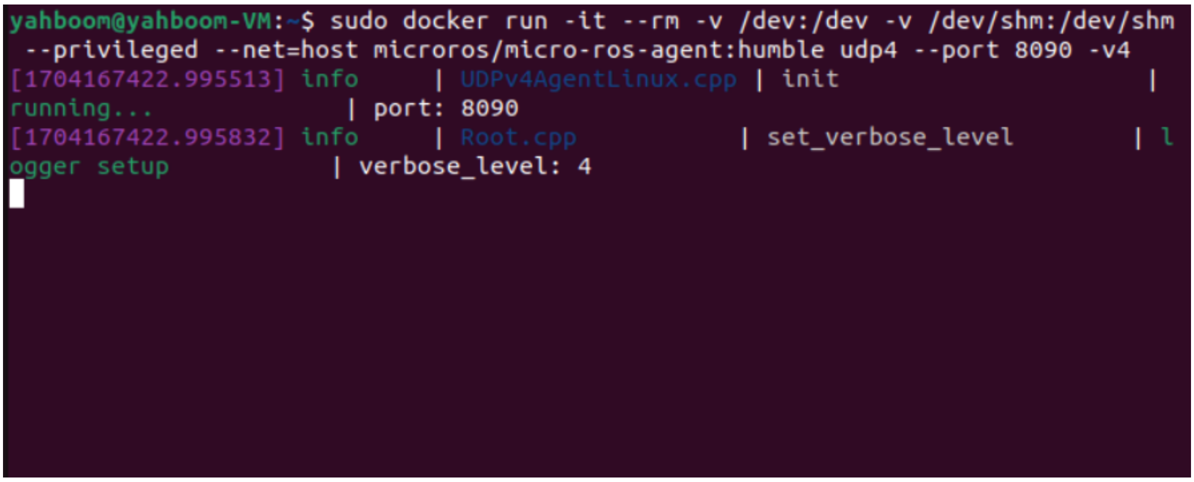
3. Start and connect the agent

If you use the factory virtual machine system, you can enter in the terminal:

```
sh ~/start_agent_computer.sh
```

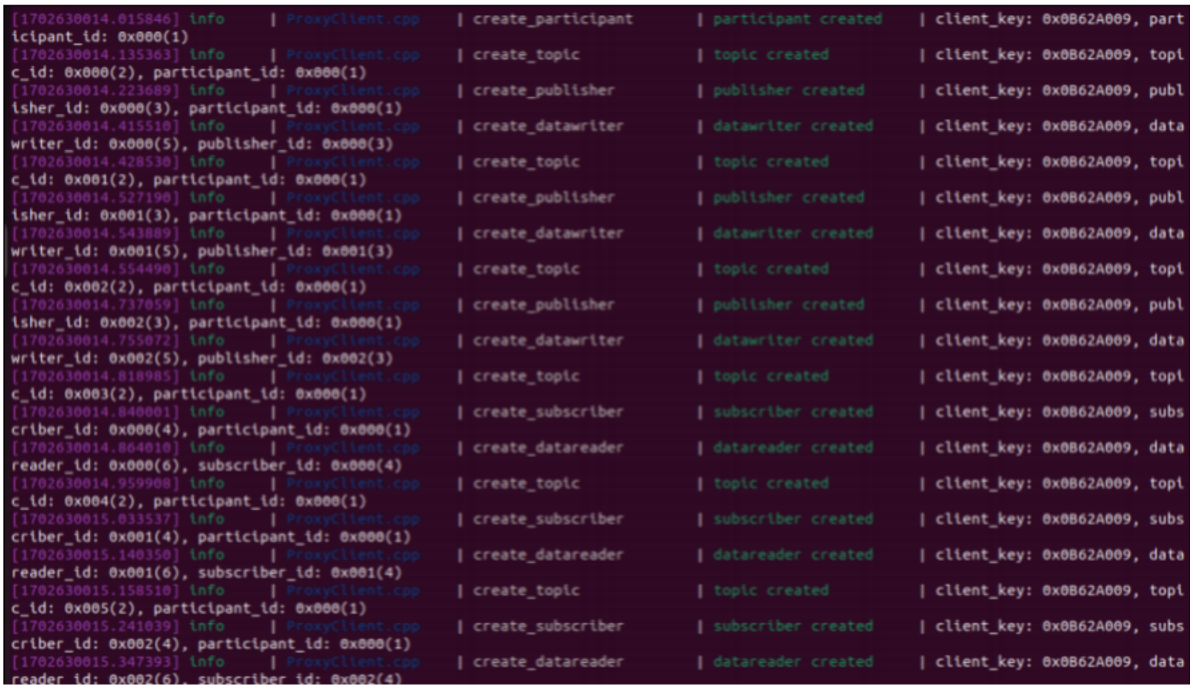
If you use a third-party virtual machine system, you need to install the docker development environment first, and open the terminal and enter:

```
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --
net=host microros/micro-ros-agent:humble udp4 --port 8090 -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 8090 -v4
[1704167422.995513] info     | UDPv4AgentLinux.cpp | init                     |
running...             | port: 8090
[1704167422.995832] 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.



```
[1702630014.015846] info     | ProxyClient.cpp     | create_participant     | participant created    | client_key: 0x0B62A009, part
icipant_id: 0x000(1)
[1702630014.135363] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0B62A009, topi
c_id: 0x000(2), participant_id: 0x000(1)
[1702630014.223689] info     | ProxyClient.cpp     | create_publisher       | publisher created      | client_key: 0x0B62A009, publ
isher_id: 0x000(3), participant_id: 0x000(1)
[1702630014.415510] info     | ProxyClient.cpp     | create_datawriter      | datawriter created     | client_key: 0x0B62A009, data
writer_id: 0x000(5), publisher_id: 0x000(3)
[1702630014.428530] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0B62A009, topi
c_id: 0x001(2), participant_id: 0x000(1)
[1702630014.527190] info     | ProxyClient.cpp     | create_publisher       | publisher created      | client_key: 0x0B62A009, publ
isher_id: 0x001(3), participant_id: 0x000(1)
[1702630014.543889] info     | ProxyClient.cpp     | create_datawriter      | datawriter created     | client_key: 0x0B62A009, data
writer_id: 0x001(5), publisher_id: 0x001(3)
[1702630014.554490] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0B62A009, topi
c_id: 0x002(2), participant_id: 0x000(1)
[1702630014.737059] info     | ProxyClient.cpp     | create_publisher       | publisher created      | client_key: 0x0B62A009, publ
isher_id: 0x002(3), participant_id: 0x000(1)
[1702630014.755072] info     | ProxyClient.cpp     | create_datawriter      | datawriter created     | client_key: 0x0B62A009, data
writer_id: 0x002(5), publisher_id: 0x002(3)
[1702630014.818985] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0B62A009, topi
c_id: 0x003(2), participant_id: 0x000(1)
[1702630014.840001] info     | ProxyClient.cpp     | create_subscriber      | subscriber created     | client_key: 0x0B62A009, subs
criber_id: 0x000(4), participant_id: 0x000(1)
[1702630014.864010] info     | ProxyClient.cpp     | create_datareader      | datareader created     | client_key: 0x0B62A009, data
reader_id: 0x000(6), subscriber_id: 0x000(4)
[1702630014.959908] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0B62A009, topi
c_id: 0x004(2), participant_id: 0x000(1)
[1702630015.033537] info     | ProxyClient.cpp     | create_subscriber      | subscriber created     | client_key: 0x0B62A009, subs
criber_id: 0x001(4), participant_id: 0x000(1)
[1702630015.140350] info     | ProxyClient.cpp     | create_datareader      | datareader created     | client_key: 0x0B62A009, data
reader_id: 0x001(6), subscriber_id: 0x001(4)
[1702630015.158510] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0B62A009, topi
c_id: 0x005(2), participant_id: 0x000(1)
[1702630015.241039] info     | ProxyClient.cpp     | create_subscriber      | subscriber created     | client_key: 0x0B62A009, subs
criber_id: 0x002(4), participant_id: 0x000(1)
[1702630015.347393] info     | ProxyClient.cpp     | create_datareader      | datareader created     | client_key: 0x0B62A009, data
reader_id: 0x002(6), subscriber_id: 0x002(4)
```

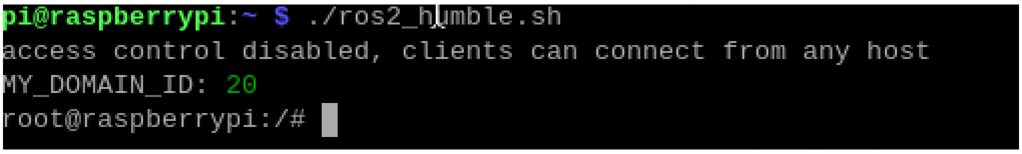
4. Start the program

4.1 Run the command

If it is the Raspberry Pi desktop version and the Jetson Nano desktop version, you need to enter docker in advance and enter the terminal.

```
sh ros2_humble.sh
```

When the following interface appears, it means that you have successfully entered docker.



```
pi@raspberrypi:~ $ ./ros2_humble.sh
access control disabled, clients can connect from any host
MY_DOMAIN_ID: 20
root@raspberrypi:/#
```

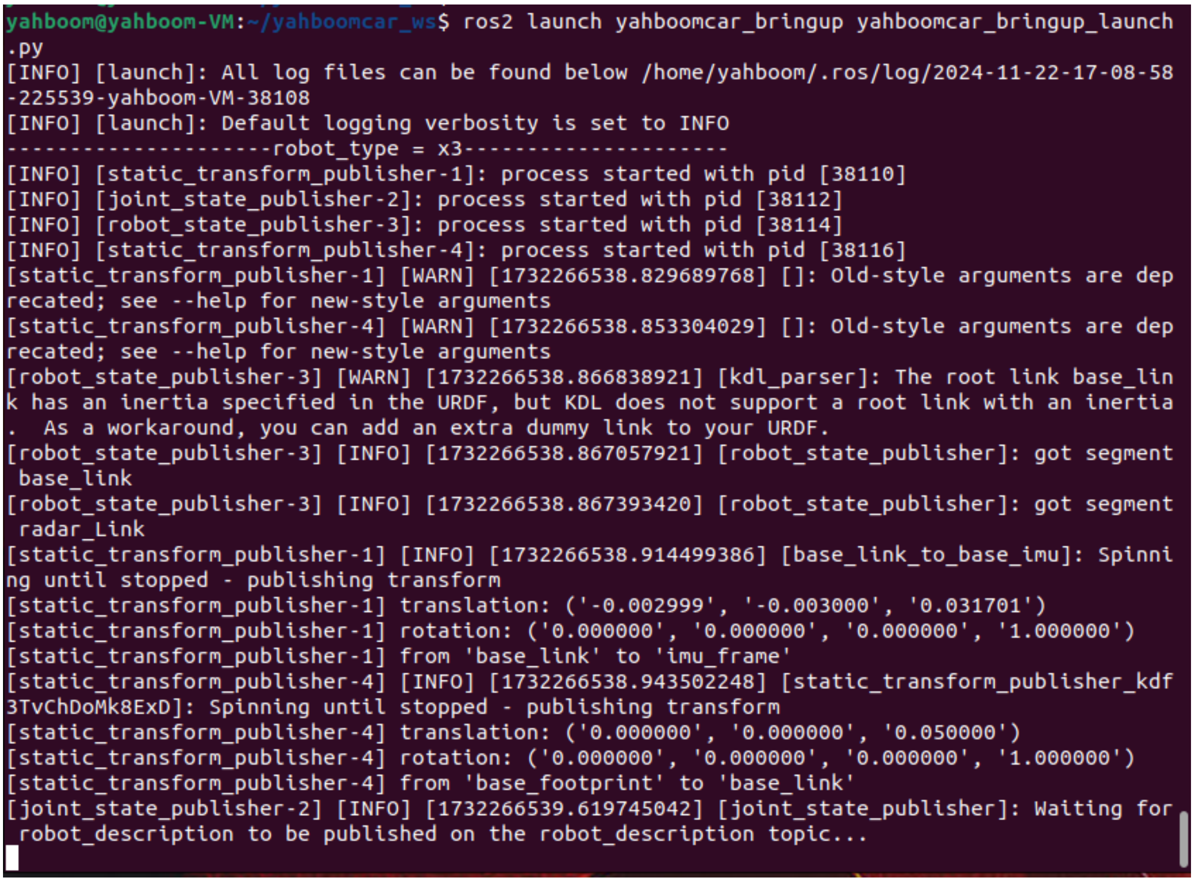
Then enter in docker separately, **(see the [Enter the same docker terminal] section)**

```
ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py ••#underlying data
program
ros2 launch yahboomcar_nav display_launch.py ••#map visualization
ros2 launch yahboomcar_nav map_cartographer_launch.py ••#map node
#save map
ros2 launch yahboomcar_nav save_map_launch.py
```

Take the matching virtual machine as an example, terminal input,

```
ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py
```

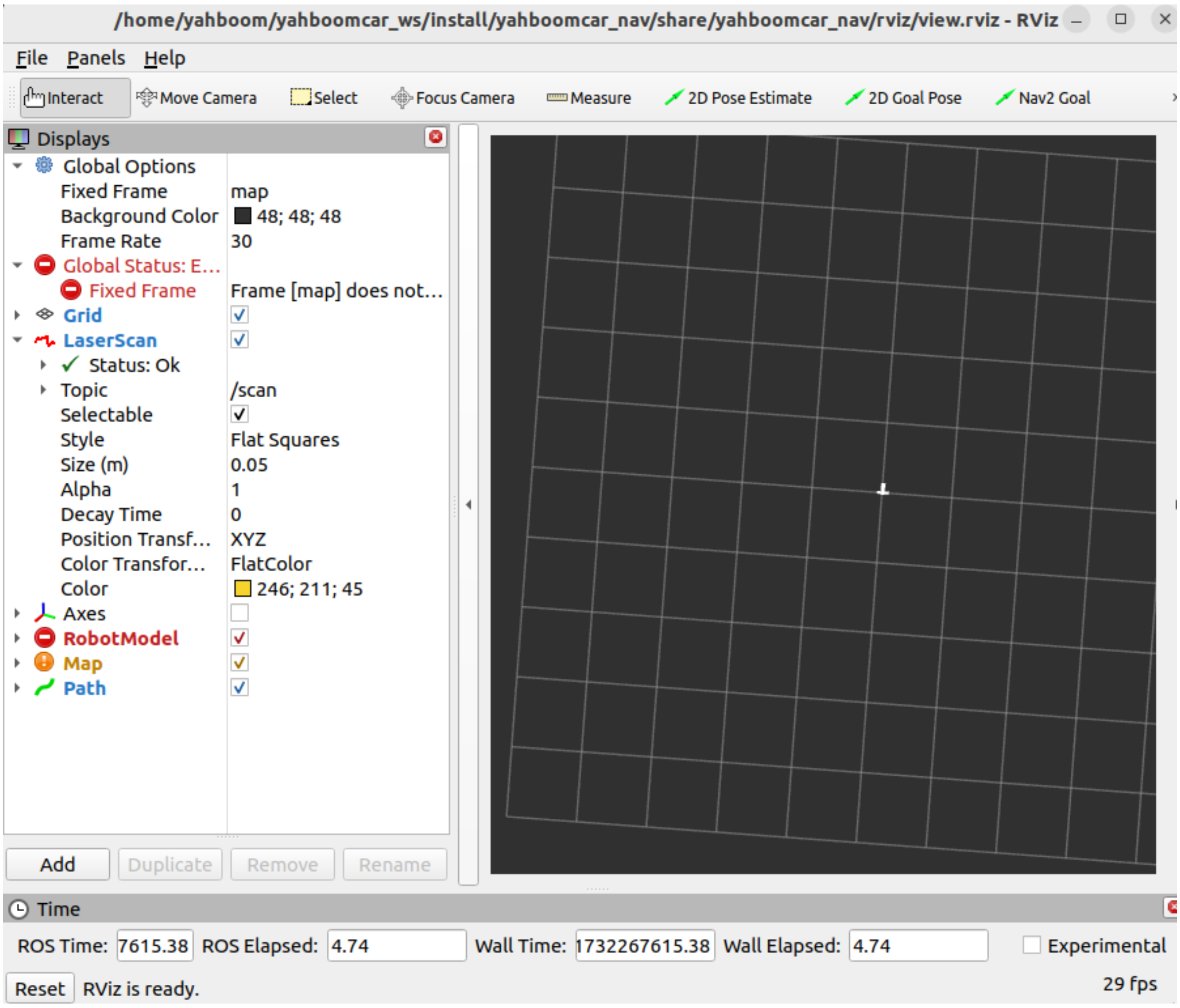
First, start the car to process the underlying data program



```
yahboom@yahboom-VM:~/yahboomcar_ws$ ros2 launch yahboomcar_bringup yahboomcar_bringup_launch
.py
[INFO] [launch]: All log files can be found below /home/yahboom/.ros/log/2024-11-22-17-08-58
-225539-yahboom-VM-38108
[INFO] [launch]: Default logging verbosity is set to INFO
---------------------robot_type = x3--------------------
[INFO] [static_transform_publisher-1]: process started with pid [38110]
[INFO] [joint_state_publisher-2]: process started with pid [38112]
[INFO] [robot_state_publisher-3]: process started with pid [38114]
[INFO] [static_transform_publisher-4]: process started with pid [38116]
[static_transform_publisher-1] [WARN] [1732266538.829689768] []: Old-style arguments are dep
recated; see --help for new-style arguments
[static_transform_publisher-4] [WARN] [1732266538.853304029] []: Old-style arguments are dep
recated; see --help for new-style arguments
[robot_state_publisher-3] [WARN] [1732266538.866838921] [kdl_parser]: The root link base_lin
k 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-3] [INFO] [1732266538.867057921] [robot_state_publisher]: got segment
 base_link
[robot_state_publisher-3] [INFO] [1732266538.867393420] [robot_state_publisher]: got segment
 radar_Link
[static_transform_publisher-1] [INFO] [1732266538.914499386] [base_link_to_base_imu]: Spinni
ng until stopped - publishing transform
[static_transform_publisher-1] translation: ('-0.002999', '-0.003000', '0.031701')
[static_transform_publisher-1] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-1] from 'base_link' to 'imu_frame'
[static_transform_publisher-4] [INFO] [1732266538.943502248] [static_transform_publisher_kdf
3TvChDoMk8ExD]: Spinning until stopped - publishing transform
[static_transform_publisher-4] translation: ('0.000000', '0.000000', '0.050000')
[static_transform_publisher-4] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-4] from 'base_footprint' to 'base_link'
[joint_state_publisher-2] [INFO] [1732266539.619745042] [joint_state_publisher]: Waiting for
 robot_description to be published on the robot_description topic...
```

Then, start rviz, visualize the map, and input in the terminal.

```
ros2 launch yahboomcar_nav display_launch.py
```

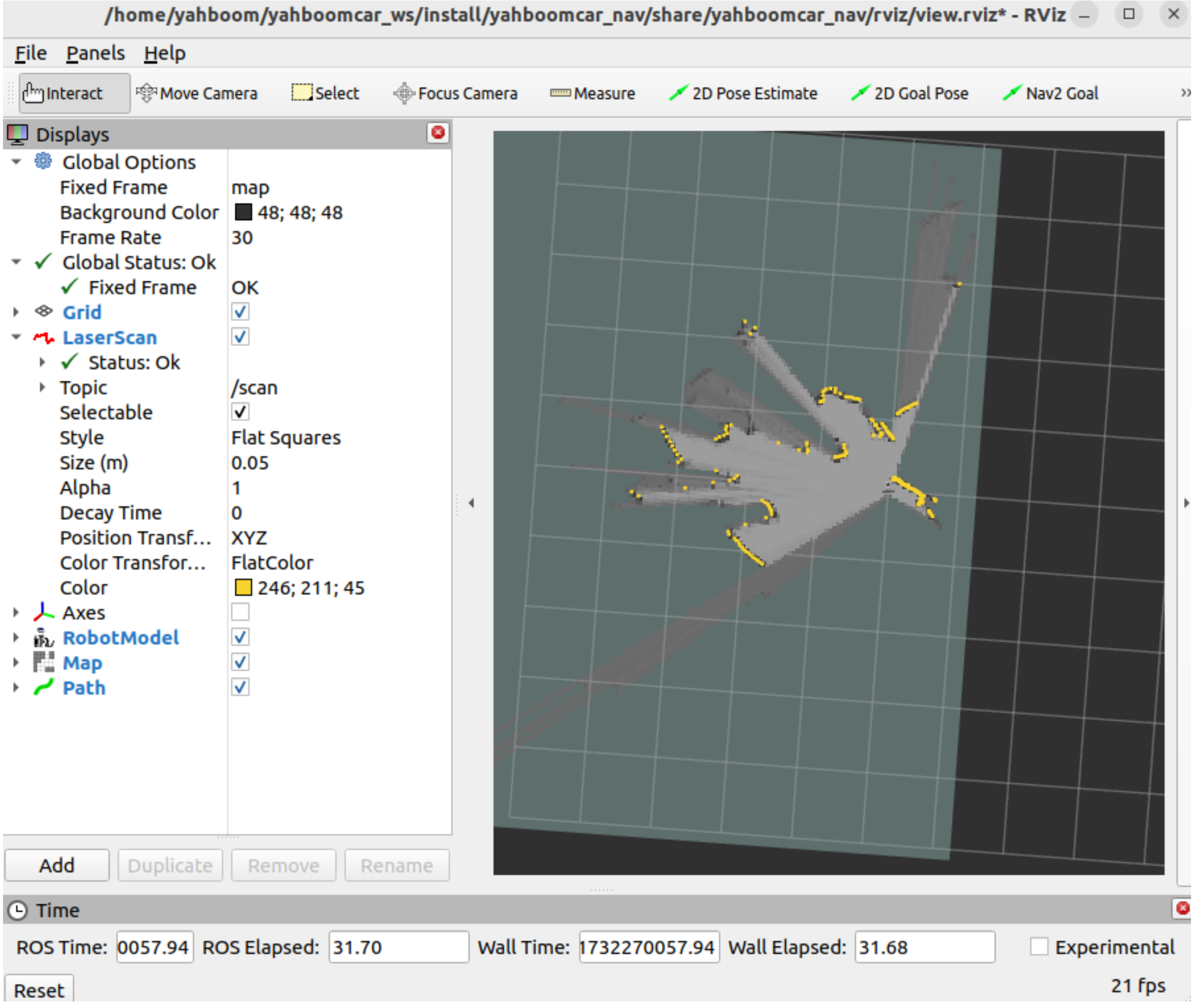


```
Global Options
  Fixed Frame            map
  Background Color       ■ 48; 48; 48
  Frame Rate             30
Global Status: E…
  Fixed Frame            Frame [map] does not…
Grid                     ☑
LaserScan                ☑
  ✓ Status: Ok
  Topic                  /scan
  Selectable             ☑
  Style                  Flat Squares
  Size (m)               0.05
  Alpha                  1
  Decay Time             0
  Position Transf…       XYZ
  Color Transfor…        FlatColor
  Color                  🟨 246; 211; 45
Axes                     □
RobotModel               ☑
Map                      ☑
Path                     ☑

ROS Time: 7615.38  ROS Elapsed: 4.74  Wall Time: 1732267615.38  Wall Elapsed: 4.74  □ Experimental
RViz is ready.
```

At this time, the map node has not been run, so there is no data. Next, run the map node and input in the terminal,

```
ros2 launch yahboomcar_nav map_cartographer_launch.py
```



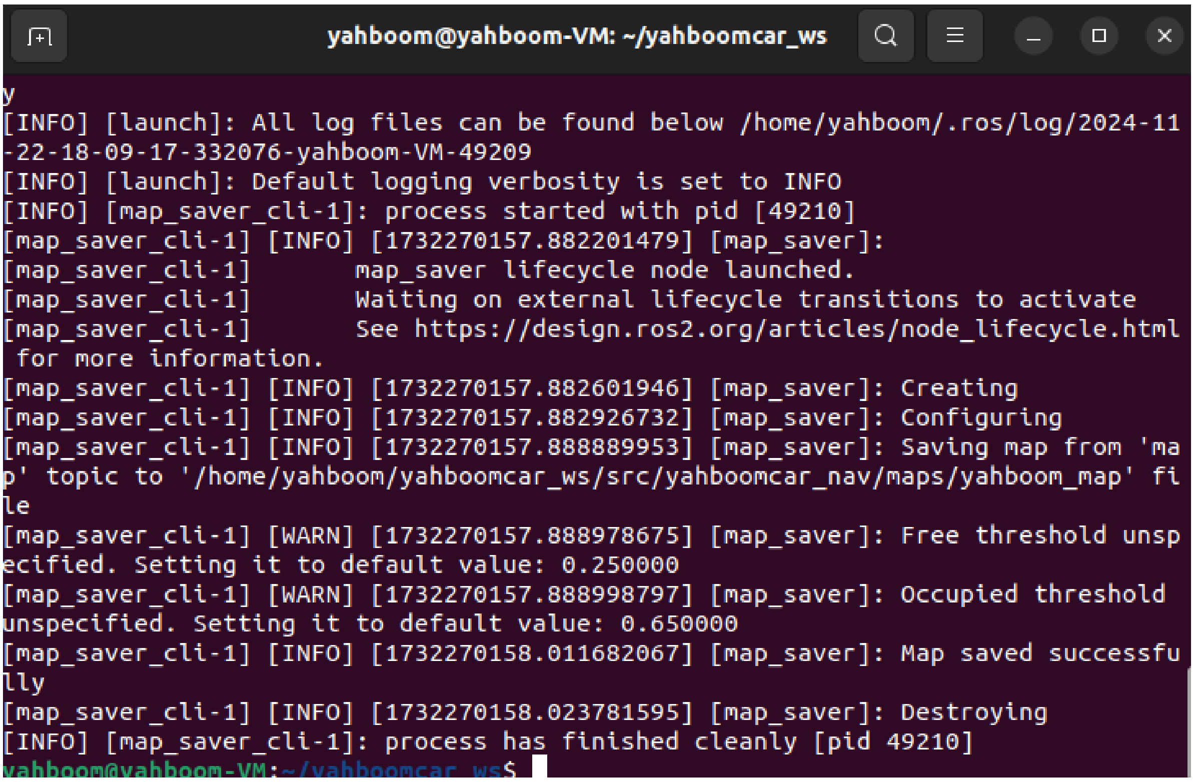
```
Global Options
    Fixed Frame            map
    Background Color       48; 48; 48
    Frame Rate             30
✓ Global Status: Ok
    ✓ Fixed Frame          OK
Grid                       ☑
LaserScan                  ☑
    ✓ Status: Ok
    Topic                  /scan
    Selectable             ☑
    Style                  Flat Squares
    Size (m)               0.05
    Alpha                  1
    Decay Time             0
    Position Transf...     XYZ
    Color Transfor...      FlatColor
    Color                  246; 211; 45
Axes                       □
RobotModel                 ☑
Map                        ☑
Path                       ☑
```

ROS Time: 0057.94 ROS Elapsed: 31.70 Wall Time: 1732270057.94 Wall Elapsed: 31.68 □ Experimental

21 fps

Then hold the handheld radar and walk slowly and parallelly through the area that needs to be mapped. After the map is built, enter the following command to save the map and input in the terminal,

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



```
y
[INFO] [launch]: All log files can be found below /home/yahboom/.ros/log/2024-11
-22-18-09-17-332076-yahboom-VM-49209
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [map_saver_cli-1]: process started with pid [49210]
[map_saver_cli-1] [INFO] [1732270157.882201479] [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] [1732270157.882601946] [map_saver]: Creating
[map_saver_cli-1] [INFO] [1732270157.882926732] [map_saver]: Configuring
[map_saver_cli-1] [INFO] [1732270157.888889953] [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] [1732270157.888978675] [map_saver]: Free threshold unsp
ecified. Setting it to default value: 0.250000
[map_saver_cli-1] [WARN] [1732270157.888998797] [map_saver]: Occupied threshold
unspecified. Setting it to default value: 0.650000
[map_saver_cli-1] [INFO] [1732270158.011682067] [map_saver]: Map saved successfu
lly
[map_saver_cli-1] [INFO] [1732270158.023781595] [map_saver]: Destroying
[INFO] [map_saver_cli-1]: process has finished cleanly [pid 49210]
yahboom@yahboom-VM:~/yahboomcar_ws$
```

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

Take the matching virtual machine as an example code path:

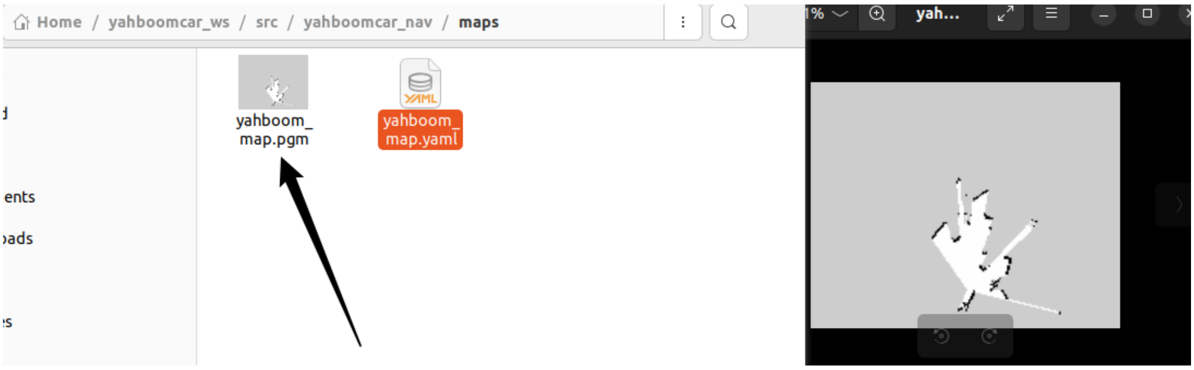
```
/home/yahboom/yahboomcar_ws/src/yahboomcar_nav/maps
```

Two files will be generated, one is yahboom_map.pgm and the other is yahboom_map.yaml. Check the contents of 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
- 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 yaw values.
- 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.

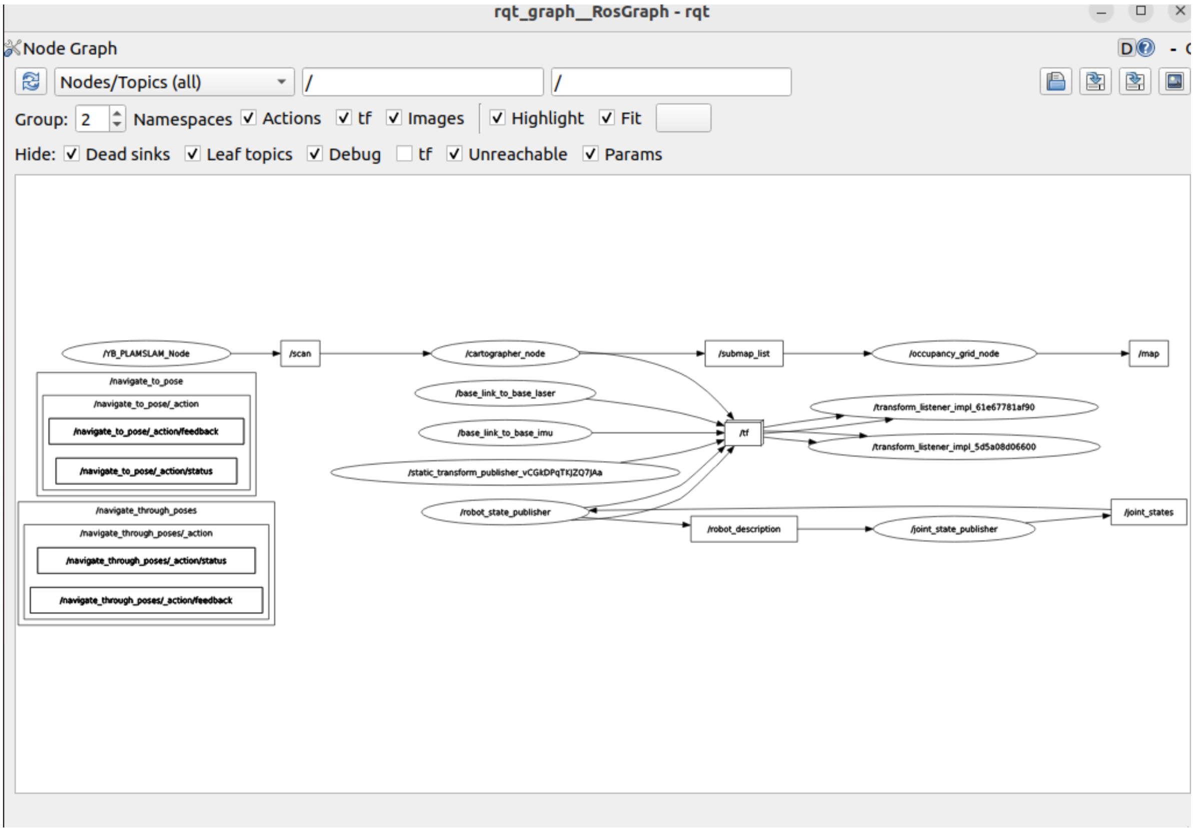
The other file is a pdf file,



5. View the node communication graph

Terminal input,

```
ros2 run rqt_graph rqt_graph
```



rqt_graph__RosGraph - rqt

Node Graph

Nodes/Topics (all) / /

Group: 2 Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit

Hide: ☑ Dead sinks ☑ Leaf topics ☑ Debug □ tf ☑ Unreachable ☑ Params

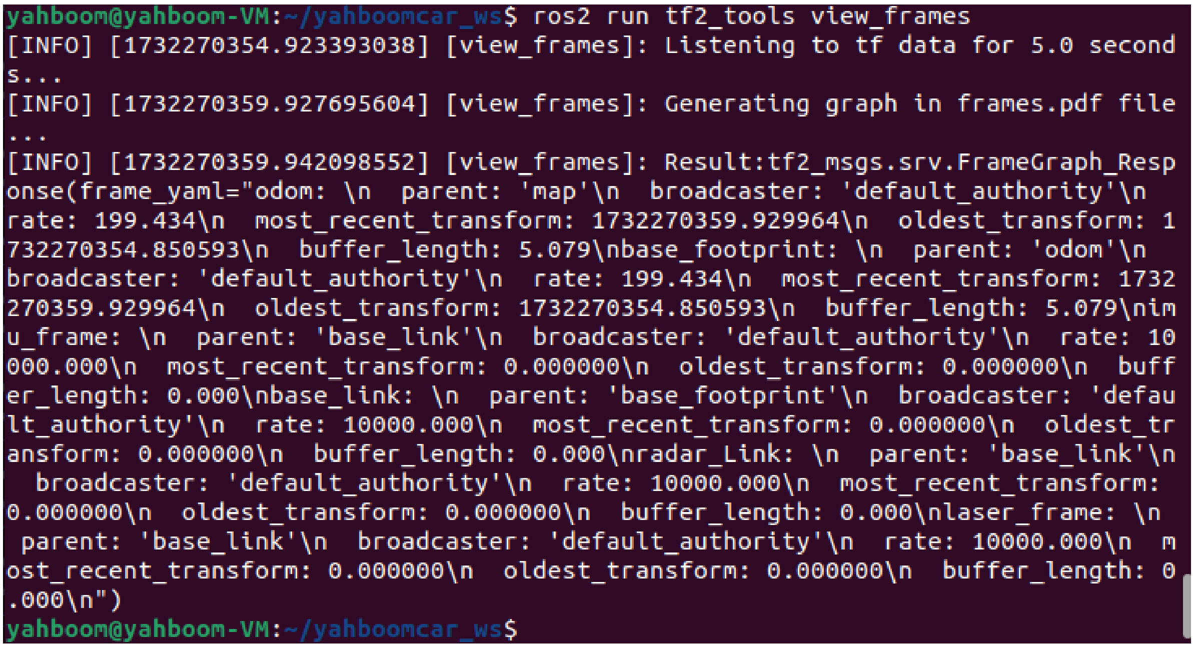
/YB_PLAMSLAM_Node
/scan
/cartographer_node
/submap_list
/occupancy_grid_node
/map
/navigate_to_pose
/navigate_to_pose/_action
/navigate_to_pose/_action/feedback
/navigate_to_pose/_action/status
/base_link_to_base_laser
/base_link_to_base_imu
/static_transform_publisher_vCGkDPqTKJZQ7JAa
/tf
/transform_listener_impl_61e67781af90
/transform_listener_impl_5d5a08d06600
/navigate_through_poses
/navigate_through_poses/_action
/navigate_through_poses/_action/status
/navigate_through_poses/_action/feedback
/robot_state_publisher
/robot_description
/joint_state_publisher
/joint_states

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

6. View TF tree

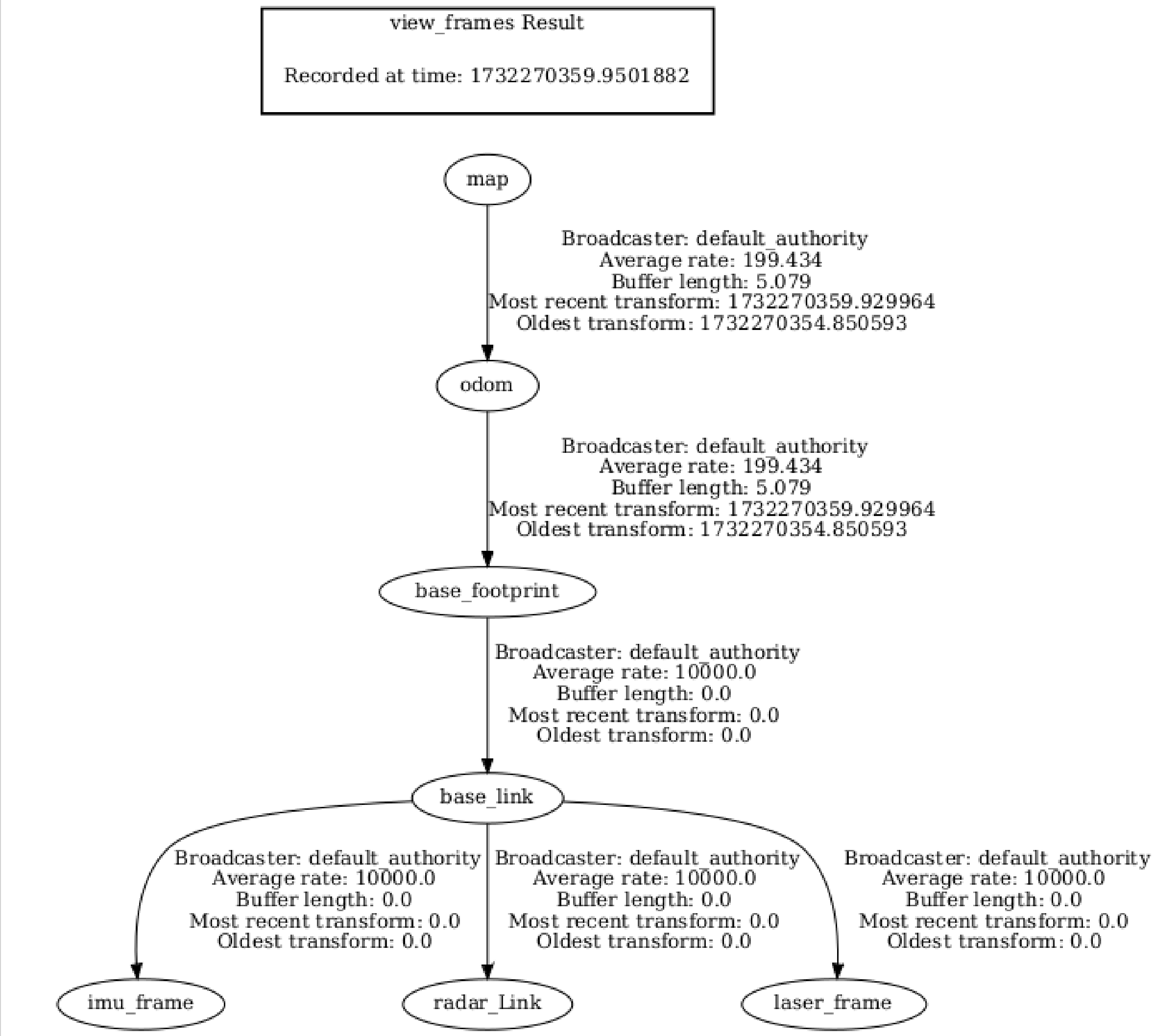
Terminal input,

```
ros2 run tf2_tools view_frames
```



```
yahboom@yahboom-VM:~/yahboomcar_ws$ ros2 run tf2_tools view_frames
[INFO] [1732270354.923393038] [view_frames]: Listening to tf data for 5.0 second
s...
[INFO] [1732270359.927695604] [view_frames]: Generating graph in frames.pdf file
...
[INFO] [1732270359.942098552] [view_frames]: Result:tf2_msgs.srv.FrameGraph_Resp
onse(frame_yaml="odom: \n  parent: 'map'\n  broadcaster: 'default_authority'\n  
rate: 199.434\n  most_recent_transform: 1732270359.929964\n  oldest_transform: 1
732270354.850593\n  buffer_length: 5.079\nbase_footprint: \n  parent: 'odom'\n  
broadcaster: 'default_authority'\n  rate: 199.434\n  most_recent_transform: 1732
270359.929964\n  oldest_transform: 1732270354.850593\n  buffer_length: 5.079\nim
u_frame: \n  parent: 'base_link'\n  broadcaster: 'default_authority'\n  rate: 10
000.000\n  most_recent_transform: 0.000000\n  oldest_transform: 0.000000\n  buff
er_length: 0.000\nbase_link: \n  parent: 'base_footprint'\n  broadcaster: 'defau
lt_authority'\n  rate: 10000.000\n  most_recent_transform: 0.000000\n  oldest_tr
ansform: 0.000000\n  buffer_length: 0.000\nradar_Link: \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\nlaser_frame: \n 
 parent: 'base_link'\n  broadcaster: 'default_authority'\n  rate: 10000.000\n  m
ost_recent_transform: 0.000000\n  oldest_transform: 0.000000\n  buffer_length: 0
.000\n")
yahboom@yahboom-VM:~/yahboomcar_ws$
```

After running, two files will be generated in the terminal directory, namely .gv and .pdf files, and the pdf file is the TF tree.


view_frames Result
Recorded at time: 1732270359.9501882
map
Broadcaster: default_authority
Average rate: 199.434
Buffer length: 5.079
Most recent transform: 1732270359.929964
Oldest transform: 1732270354.850593
odom
Broadcaster: default_authority
Average rate: 199.434
Buffer length: 5.079
Most recent transform: 1732270359.929964
Oldest transform: 1732270354.850593
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
imu_frame
radar_Link
laser_frame

7. Code analysis

Take the virtual machine as an example, here only the map_cartographer_launch.py for building the map is explained, this file path is,

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

map_cartographer_launch.py

```
import os
from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource
from launch_ros.actions import Node


def generate_launch_description():
    package_launch_path
=os.path.join(get_package_share_directory('yahboomcar_nav'), 'launch')

    cartographer_launch =
IncludeLaunchDescription(PythonLaunchDescriptionSource(
        [package_launch_path, '/cartographer_launch.py'])
    )
    base_link_to_laser_tf_node = Node(
     package='tf2_ros',
     executable='static_transform_publisher',
     name='base_link_to_base_laser',
     arguments=['-0.0046412', '0' ,
'0.094079','0','0','0','base_link','laser_frame']
    )
    return LaunchDescription([cartographer_launch,base_link_to_laser_tf_node])
```

Here, a launch file - cartographer_launch and a node that publishes static transformations - base_link_to_laser_tf_node are run.

** Taking the virtual machine as an example, take a closer look at cartographer_launch.py, which is located at,**

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

cartographer_launch.py

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


def generate_launch_description():
    use_sim_time = LaunchConfiguration('use_sim_time', default='true')
    package_path = get_package_share_directory('yahboomcar_nav')
    configuration_directory = LaunchConfiguration('configuration_directory',
default=os.path.join(
                                                  package_path, 'params'))
    configuration_basename = LaunchConfiguration('configuration_basename',
default='lds_2d.lua')

    resolution = LaunchConfiguration('resolution', default='0.05')
    publish_period_sec = LaunchConfiguration(
        'publish_period_sec', default='1.0')

    return LaunchDescription([
        DeclareLaunchArgument(
            'configuration_directory',
            default_value=configuration_directory,
            description='Full path to config file to load'),
        DeclareLaunchArgument(
            'configuration_basename',
            default_value=configuration_basename,
            description='Name of lua file for cartographer'),
        DeclareLaunchArgument(
            'use_sim_time',
            default_value='false',
            description='Use simulation (Gazebo) clock if true'),

        Node(
            package='cartographer_ros',
            executable='cartographer_node',
            name='cartographer_node',
            output='screen',
            parameters=[{'use_sim_time': use_sim_time}],
            arguments=['-configuration_directory', configuration_directory,
                       '-configuration_basename', configuration_basename],
                       remappings=[('/scan', '/scan')]),

        DeclareLaunchArgument(
            'resolution',
            default_value=resolution,
            description='Resolution of a grid cell in the published occupancy
grid'),

        DeclareLaunchArgument(
            'publish_period_sec',
            default_value=publish_period_sec,
            description='OccupancyGrid publishing period'),

        IncludeLaunchDescription(
            PythonLaunchDescriptionSource(
                [ThisLaunchFileDir(), '/occupancy_grid_launch.py']),
            launch_arguments={'use_sim_time': use_sim_time, 'resolution':
resolution,
                              'publish_period_sec': publish_period_sec}.items(),
        ),
    ])
```

Here we mainly run the cartographer_node map building node and occupancy_grid_launch.py, and load the parameter configuration file.

```
The file is located in (taking the supporting virtual machine as an example),
/home/yahboom/yahboomcar_ws/src/yahboomcar_nav/params
```

lds_2d.lua，

```
include "map_builder.lua"
include "trajectory_builder.lua"

options = {
  map_builder = MAP_BUILDER,
  trajectory_builder = TRAJECTORY_BUILDER,
  map_frame = "map",
  tracking_frame = "base_footprint",
  published_frame = "base_footprint",
  odom_frame = "odom",
  provide_odom_frame = true,
  publish_frame_projected_to_2d = false,

  use_odometry = false,
  use_nav_sat = false,
  use_landmarks = false,
  num_laser_scans = 1,
  num_multi_echo_laser_scans = 0,
  num_subdivisions_per_laser_scan = 1,
  num_point_clouds = 0,
  lookup_transform_timeout_sec = 0.2,
  submap_publish_period_sec = 0.3,
  pose_publish_period_sec = 5e-3,
  trajectory_publish_period_sec = 30e-3,
  rangefinder_sampling_ratio = 1.,
  odometry_sampling_ratio = 1.,
  fixed_frame_pose_sampling_ratio = 1.,
  imu_sampling_ratio = 1.,
  landmarks_sampling_ratio = 1.,
}


MAP_BUILDER.use_trajectory_builder_2d = true

TRAJECTORY_BUILDER_2D.use_imu_data = false
TRAJECTORY_BUILDER_2D.min_range = 0.10
TRAJECTORY_BUILDER_2D.max_range = 30.0

TRAJECTORY_BUILDER_2D.missing_data_ray_length = 3.
TRAJECTORY_BUILDER_2D.use_online_correlative_scan_matching = true
TRAJECTORY_BUILDER_2D.motion_filter.max_angle_radians = math.rad(0.3)

POSE_GRAPH.constraint_builder.min_score = 0.65
POSE_GRAPH.constraint_builder.global_localization_min_score = 0.7

-- 设置0可关闭全局SLAM
-- Set to 0 to disable global SLAM
POSE_GRAPH.optimize_every_n_nodes = 0


return options
```