Cartographer-SLAM mapping

1. Course Content

1. Learn the robot cartographer mapping algorithm for SLAM mapping function

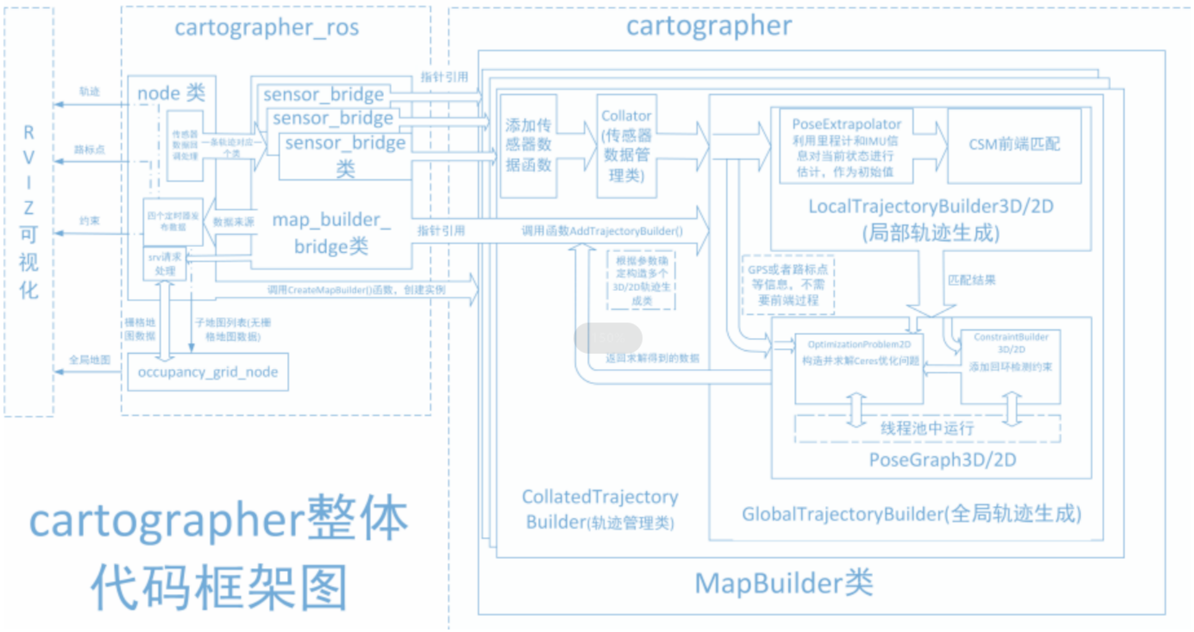
   After running the sample program, use the keyboard or handle to control the robot to move and complete the map construction and save the map.

2. Introduction to Cartographer

2.1 Introduction

Cartographer is an open-source 2D and 3D SLAM (simultaneous localization and mapping) library from Google, supported by the ROS system. This mapping algorithm utilizes graph optimizations (multi-threaded backend optimization and problem optimization built using Cere). It combines data from multiple sensors (such as LIDAR, IMU, and cameras) to simultaneously calculate sensor positions and map the surrounding environment.

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


RVIZ可视化
cartographer_ros
cartographer
轨迹
路标点
约束
全局地图
node 类
传感器数据回调处理
一条轨迹对应一个类
sensor_bridge
sensor_bridge
sensor_bridge类
四个定时器发布数据
数据来源
srv请求处理
map_builder_bridge类
指针引用
指针引用
调用CreateMapBuilder()函数，创建实例
栅格地图数据
子地图列表(无栅格地图数据)
occupancy_grid_node
添加传感器数据函数
Collator(传感器数据管理类)
PoseExtrapolator 利用里程计和IMU信息对当前状态进行估计，作为初始值
CSM前端匹配
LocalTrajectoryBuilder3D/2D (局部轨迹生成)
调用函数AddTrajectoryBuilder()
根据参数确定构造多个3D/2D轨迹生成类
GPS或者路标点等信息，不需要前端过程
匹配结果
返回求解得到的数据
OptimizationProblem2D 构造并求解Ceres优化问题
ConstraintBuilder 3D/2D 添加回环检测约束
线程池中运行
PoseGraph3D/2D
CollatedTrajectory Builder(轨迹管理类)
GlobalTrajectoryBuilder(全局轨迹生成)
MapBuilder类
cartographer整体代码框架图

Cartographer uses the mainstream SLAM framework, which consists of a three-step process of feature extraction, loop closure detection, and backend optimization. A certain number of LaserScans form a submap, and a series of submaps form the global map. While the short-term cumulative error of constructing submaps using LaserScans is small, the long-term cumulative error of constructing a global map using submaps is significant. Therefore, loop closure detection is required to correct the positions of these submaps. The basic unit of loop closure detection is the submap, which uses the scan_match strategy. Cartographer focuses on creating submaps that fuse multi-sensor data (odometry, IMU, LaserScan, etc.) and implementing the scan_match strategy for loop closure detection.

- cartographer_ros

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 a message to facilitate debugging and visualization.

2.2 Related Materials

GitHub repository

Official Documentation

3. Preparation

3.1 Content Description

This lesson uses the Jetson Orin NX as an example. For Raspberry Pi and Jetson Nano boards, you need to open a terminal and enter the command to enter the Docker container. Once inside the Docker container, enter the commands mentioned in this lesson in the terminal. For instructions on entering the Docker container, refer to the product tutorial **[Configuration and Operation Guide]--[Entering the Docker (Jetson Nano and Raspberry Pi 5 users, see here)]**. For Orin and NX boards, simply open a terminal and enter the commands mentioned in this lesson.

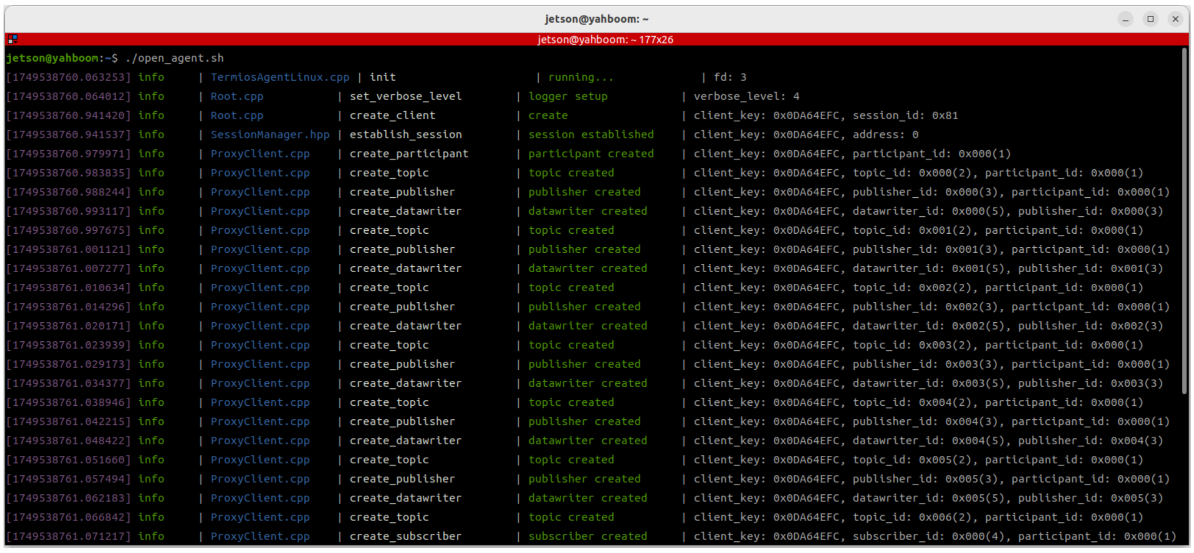
3.2 Starting the Agent

Note: To test all cases, you must start the docker agent first. If it has already been started, you do not need to start it again.

Enter the command in the vehicle terminal:

```
sh start_agent.sh
```

The terminal prints the following information, indicating that the connection is successful



```
jetson@yahboom:~$ ./open_agent.sh
[1749538760.063253] info     | TermiosAgentLinux.cpp | init                   | running...             | fd: 3
[1749538760.064012] info     | Root.cpp             | set_verbose_level      | logger setup           | verbose_level: 4
[1749538760.941420] info     | Root.cpp             | create_client          | create                 | client_key: 0x0DA64EFC, session_id: 0x81
[1749538760.941537] info     | SessionManager.hpp | establish_session      | session established    | client_key: 0x0DA64EFC, address: 0
[1749538760.979971] info     | ProxyClient.cpp      | create_participant     | participant created    | client_key: 0x0DA64EFC, participant_id: 0x000(1)
[1749538760.983835] info     | ProxyClient.cpp      | create_topic           | topic created          | client_key: 0x0DA64EFC, topic_id: 0x000(2), participant_id: 0x000(1)
[1749538760.988244] info     | ProxyClient.cpp      | create_publisher       | publisher created      | client_key: 0x0DA64EFC, publisher_id: 0x000(3), participant_id: 0x000(1)
[1749538760.993117] info     | ProxyClient.cpp      | create_datawriter      | datawriter created     | client_key: 0x0DA64EFC, datawriter_id: 0x000(5), publisher_id: 0x000(3)
[1749538760.997675] info     | ProxyClient.cpp      | create_topic           | topic created          | client_key: 0x0DA64EFC, topic_id: 0x001(2), participant_id: 0x000(1)
[1749538761.001121] info     | ProxyClient.cpp      | create_publisher       | publisher created      | client_key: 0x0DA64EFC, publisher_id: 0x001(3), participant_id: 0x000(1)
[1749538761.007277] info     | ProxyClient.cpp      | create_datawriter      | datawriter created     | client_key: 0x0DA64EFC, datawriter_id: 0x001(5), publisher_id: 0x001(3)
[1749538761.010634] info     | ProxyClient.cpp      | create_topic           | topic created          | client_key: 0x0DA64EFC, topic_id: 0x002(2), participant_id: 0x000(1)
[1749538761.014296] info     | ProxyClient.cpp      | create_publisher       | publisher created      | client_key: 0x0DA64EFC, publisher_id: 0x002(3), participant_id: 0x000(1)
[1749538761.020171] info     | ProxyClient.cpp      | create_datawriter      | datawriter created     | client_key: 0x0DA64EFC, datawriter_id: 0x002(5), publisher_id: 0x002(3)
[1749538761.023939] info     | ProxyClient.cpp      | create_topic           | topic created          | client_key: 0x0DA64EFC, topic_id: 0x003(2), participant_id: 0x000(1)
[1749538761.029173] info     | ProxyClient.cpp      | create_publisher       | publisher created      | client_key: 0x0DA64EFC, publisher_id: 0x003(3), participant_id: 0x000(1)
[1749538761.034377] info     | ProxyClient.cpp      | create_datawriter      | datawriter created     | client_key: 0x0DA64EFC, datawriter_id: 0x003(5), publisher_id: 0x003(3)
[1749538761.038946] info     | ProxyClient.cpp      | create_topic           | topic created          | client_key: 0x0DA64EFC, topic_id: 0x004(2), participant_id: 0x000(1)
[1749538761.042215] info     | ProxyClient.cpp      | create_publisher       | publisher created      | client_key: 0x0DA64EFC, publisher_id: 0x004(3), participant_id: 0x000(1)
[1749538761.048422] info     | ProxyClient.cpp      | create_datawriter      | datawriter created     | client_key: 0x0DA64EFC, datawriter_id: 0x004(5), publisher_id: 0x004(3)
[1749538761.051660] info     | ProxyClient.cpp      | create_topic           | topic created          | client_key: 0x0DA64EFC, topic_id: 0x005(2), participant_id: 0x000(1)
[1749538761.057494] info     | ProxyClient.cpp      | create_publisher       | publisher created      | client_key: 0x0DA64EFC, publisher_id: 0x005(3), participant_id: 0x000(1)
[1749538761.062183] info     | ProxyClient.cpp      | create_datawriter      | datawriter created     | client_key: 0x0DA64EFC, datawriter_id: 0x005(5), publisher_id: 0x005(3)
[1749538761.066842] info     | ProxyClient.cpp      | create_topic           | topic created          | client_key: 0x0DA64EFC, topic_id: 0x006(2), participant_id: 0x000(1)
[1749538761.071217] info     | ProxyClient.cpp      | create_subscriber      | subscriber created     | client_key: 0x0DA64EFC, subscriber_id: 0x000(4), participant_id: 0x000(1)
```

4. Run the case

4.1 Mapping Process

Notice:

- **When building a map, the slower the speed, the better the effect (mainly the slower the rotation speed). If the speed is too fast, the effect will be very poor.**
- **Jetson Nano and Raspberry Pi** series controllers need to enter the Docker container first (please refer to the [Docker course chapter - Entering the robot's Docker container] for steps).

The vehicle terminal starts the underlying sensor command:

```
ros2 launch slam_mapping bringup.launch.py
```

Restart the mapping command:

```
ros2 launch slam_mapping cartographer.launch.py
```

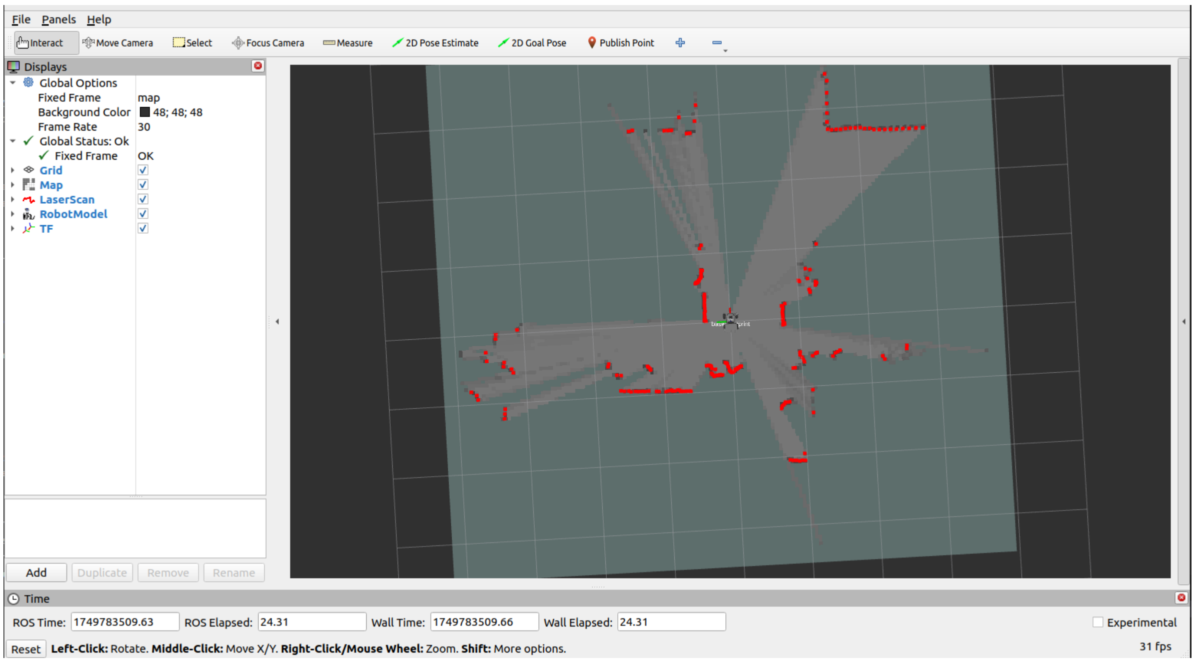
The rviz visualization function can be started on the vehicle side or the virtual machine side. **You can choose either** method to start it. It is forbidden to start it on the virtual machine side and the vehicle side repeatedly:

Taking the configuration of a virtual machine as an example, open a terminal and start the rviz visualization interface:

```
ros2 launch slam_view slam_view.launch.py
```

Start the rviz visualization interface command on the vehicle:

```
ros2 launch slam_mapping slam_view.launch.py
```

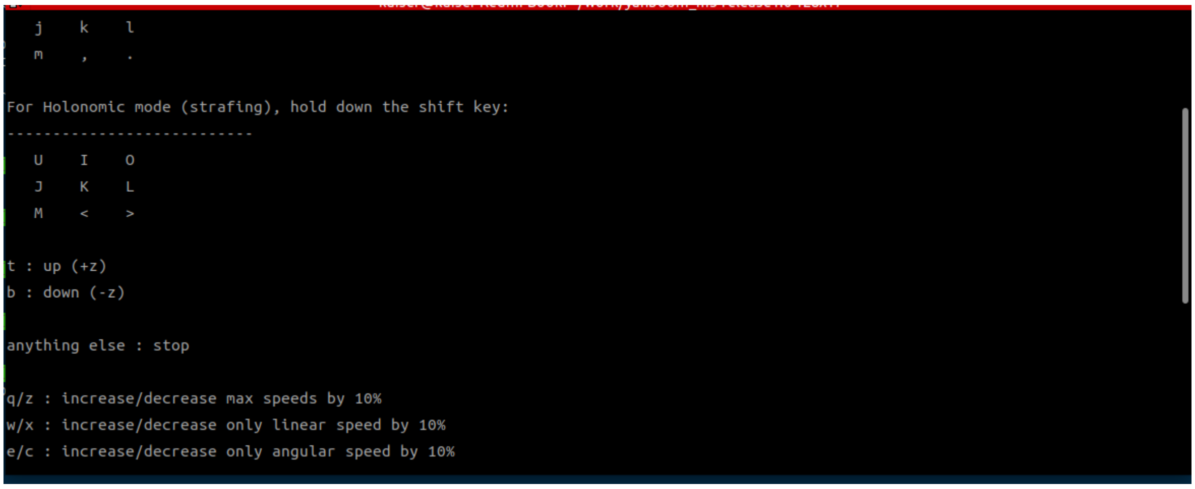


```
Global Options
  Fixed Frame        map
  Background Color   48; 48; 48
  Frame Rate         30
Global Status: Ok
  Fixed Frame        OK
Grid        ✓
Map         ✓
LaserScan   ✓
RobotModel  ✓
TF          ✓
```

```
ROS Time: 1749783509.63   ROS Elapsed: 24.31   Wall Time: 1749783509.66   Wall Elapsed: 24.31
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.
31 fps
```

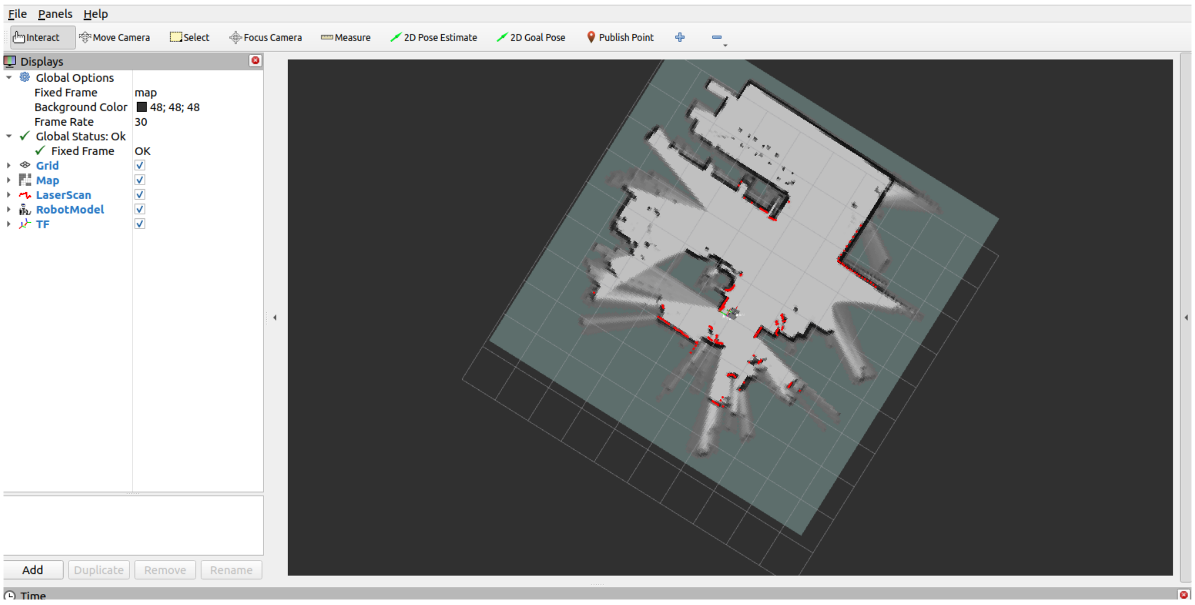
Open another terminal in the virtual machine to start the keyboard control node (you can also use the handle remote control):

```
ros2 run yahboomcar_ctrl yahboom_keyboard
```



```
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
```

Click the window in the terminal with the mouse, press z to reduce the speed appropriately, press I, <, J, L to control the car to move forward, backward, turn left, and turn right respectively, and control the car to move slowly to complete the map building

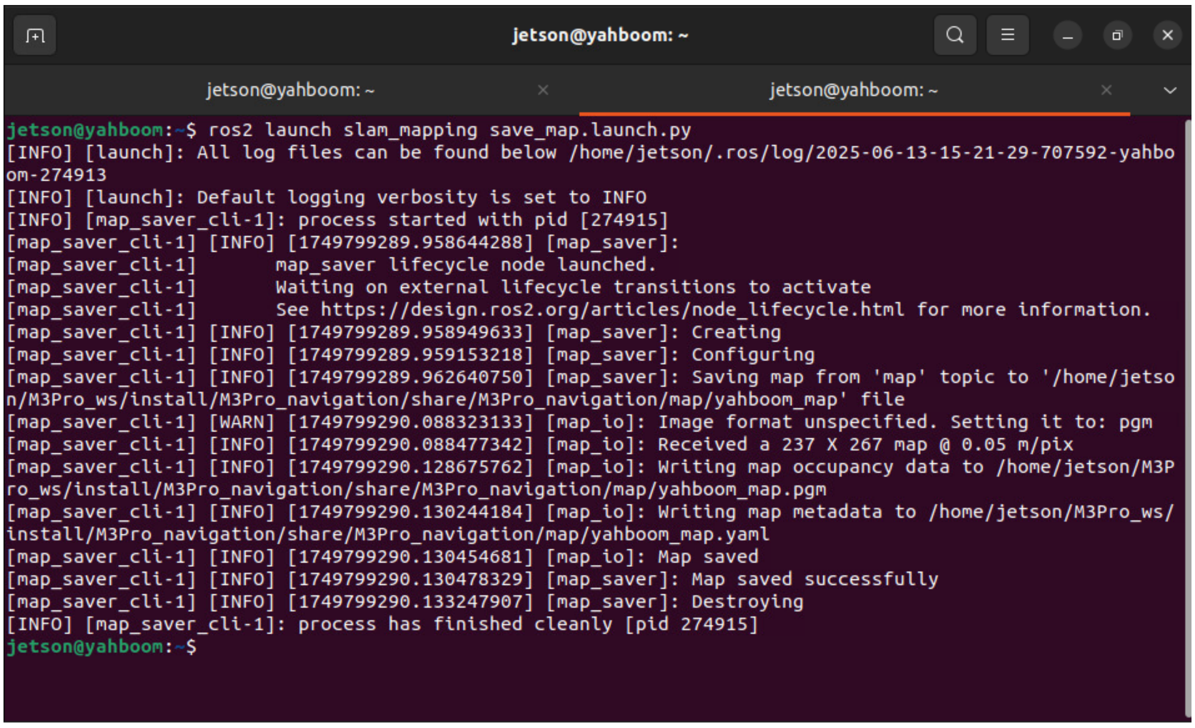


4.2 Save the map

Open a new terminal on the car and save the map

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

The terminal prompts **"Map saved successfully"** to indicate that the map has been saved successfully.



```
jetson@yahboom:~$ ros2 launch slam_mapping save_map.launch.py
[INFO] [launch]: All log files can be found below /home/jetson/.ros/log/2025-06-13-15-21-29-707592-yahbo
om-274913
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [map_saver_cli-1]: process started with pid [274915]
[map_saver_cli-1] [INFO] [1749799289.958644288] [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] [1749799289.958949633] [map_saver]: Creating
[map_saver_cli-1] [INFO] [1749799289.959153218] [map_saver]: Configuring
[map_saver_cli-1] [INFO] [1749799289.962640750] [map_saver]: Saving map from 'map' topic to '/home/jetso
n/M3Pro_ws/install/M3Pro_navigation/share/M3Pro_navigation/map/yahboom_map' file
[map_saver_cli-1] [WARN] [1749799290.088323133] [map_io]: Image format unspecified. Setting it to: pgm
[map_saver_cli-1] [INFO] [1749799290.088477342] [map_io]: Received a 237 X 267 map @ 0.05 m/pix
[map_saver_cli-1] [INFO] [1749799290.128675762] [map_io]: Writing map occupancy data to /home/jetson/M3P
ro_ws/install/M3Pro_navigation/share/M3Pro_navigation/map/yahboom_map.pgm
[map_saver_cli-1] [INFO] [1749799290.130244184] [map_io]: Writing map metadata to /home/jetson/M3Pro_ws/
install/M3Pro_navigation/share/M3Pro_navigation/map/yahboom_map.yaml
[map_saver_cli-1] [INFO] [1749799290.130454681] [map_io]: Map saved
[map_saver_cli-1] [INFO] [1749799290.130478329] [map_saver]: Map saved successfully
[map_saver_cli-1] [INFO] [1749799290.133247907] [map_saver]: Destroying
[INFO] [map_saver_cli-1]: process has finished cleanly [pid 274915]
jetson@yahboom:~$
```

The map save path is as follows:

jetson orin nano, jetson orin NX :

```
/home/jetson/M3Pro_ws/install/M3Pro_navigation/share/M3Pro_navigation/map
```

Jetson Orin Nano, Raspberry Pi :

You need to enter docker first

```
/root/M3Pro_ws/install/M3Pro_navigation/share/M3Pro_navigation/map/
```

A pgm image, a yaml file yahboom_map.yaml

```
image: yahboom_map.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: map resolution, meters/pixels
- origin: The 2D position (x, y, yaw) of the lower-left corner of the map, where yaw is a counterclockwise rotation (yaw=0 means no rotation). Currently, many parts of the system ignore the yaw value.
- negate: whether to invert the meaning of white/black, free/occupied (the interpretation of thresholds is not affected)
- occupied_thresh: Pixels with an occupancy probability greater than this threshold are considered fully occupied.
- free_thresh: Pixels with an occupancy probability less than this threshold are considered completely free.

4.3 Saving pbstream format maps

Map files in pbstream format are used for repositioning navigation, which will be explained in subsequent chapters.

After the map is built, open a new terminal and enter the command to end the map track:

```
ros2 service call /finish_trajectory cartographer_ros_msgs/srv/FinishTrajectory "
{trajectory_id: 0}"
```

Then open another terminal to save the pbstream format map file:

Save command on jetson orin nano and jetson orin NX s:

```
ros2 service call /write_state cartographer_ros_msgs/srv/WriteState "{filename:
'/home/jetson/yahboom_map.pbstream'}"
```

Save the command on Jetson Orin Nano and Raspberry Pi :

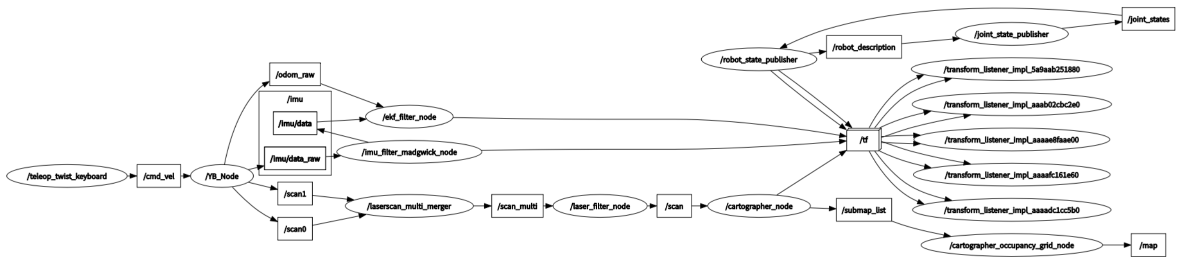
You need to enter docker first

```
ros2 service call /write_state cartographer_ros_msgs/srv/WriteState "{filename:
'/root/yahboom_map.pbstream'}"
```

5. Node parsing

5.1 Displaying the Node Computation Graph

```
ros2 run rqt_graph rqt_graph
```

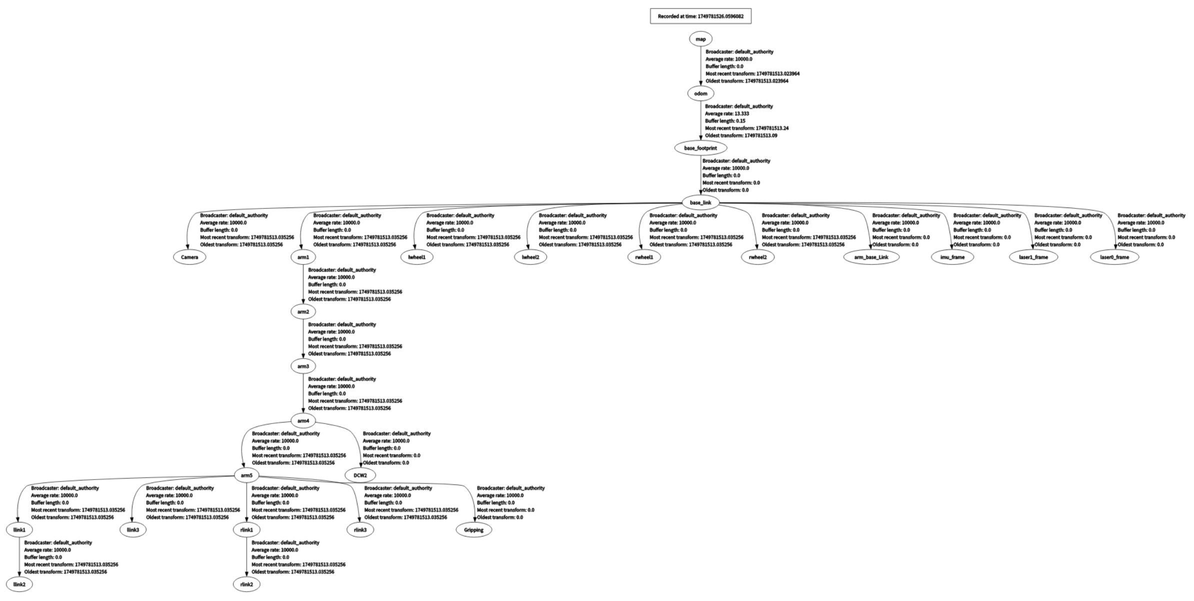


5.2 TF Transformation

The virtual machine terminal runs:

```
ros2 run rqt_tf_tree rqt_tf_tree
```

The image size is too large. The original image can be viewed in the folder of this course.



5.3 Cartographer Node Details

```
ros2 node info /slam_gmapping
```

Enter the above command in the terminal to view the subscription and publishing topics related to the gmapping node.

```
/cartographer_node
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /scan: sensor_msgs/msg/LaserScan
  Publishers:
    /constraint_list: visualization_msgs/msg/MarkerArray
    /landmark_poses_list: visualization_msgs/msg/MarkerArray
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /scan_matched_points2: sensor_msgs/msg/PointCloud2
    /submap_list: cartographer_ros_msgs/msg/SubmapList
    /tf:tf2_msgs/msg/TFMessage
    /trajectory_node_list: visualization_msgs/msg/MarkerArray
  Service Servers:
    /cartographer_node/describe_parameters:
rcl_interfaces/srv/DescribeParameters
    /cartographer_node/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /cartographer_node/get_parameters: rcl_interfaces/srv/GetParameters
    /cartographer_node/list_parameters: rcl_interfaces/srv/ListParameters
    /cartographer_node/set_parameters: rcl_interfaces/srv/SetParameters
    /cartographer_node/set_parameters_atomically:
rcl_interfaces/srv/SetParametersAtomically
    /finish_trajectory: cartographer_ros_msgs/srv/FinishTrajectory
    /get_trajectory_states: cartographer_ros_msgs/srv/GetTrajectoryStates
    /read_metrics: cartographer_ros_msgs/srv/ReadMetrics
    /start_trajectory: cartographer_ros_msgs/srv/StartTrajectory
    /submap_query: cartographer_ros_msgs/srv/SubmapQuery
    /tf2_frames: tf2_msgs/srv/FrameGraph
    /trajectory_query: cartographer_ros_msgs/srv/TrajectoryQuery
    /write_state: cartographer_ros_msgs/srv/WriteState
  Service Clients:

  Action Servers:

  Action Clients:

```