slam_toolbox mapping

1. Course Content

Learn the SLAM mapping algorithm for robotics using the slam-toolbox.

After running the sample program, control the robot using the keyboard or controller to complete mapping and save the map.

2. Introduction to slam-toolbox

slam-toolbox is an open-source 2D SLAM (Simultaneous Localization and Mapping) algorithm package based on ROS, primarily used for mapping and localizing mobile robots in unknown environments. Based on the **Graph Optimization** framework, it combines sensor data from laser radar (such as 2D LiDAR) and inertial measurement units (IMUs) to construct a 2D grid map of the environment and update the robot's pose in real time.

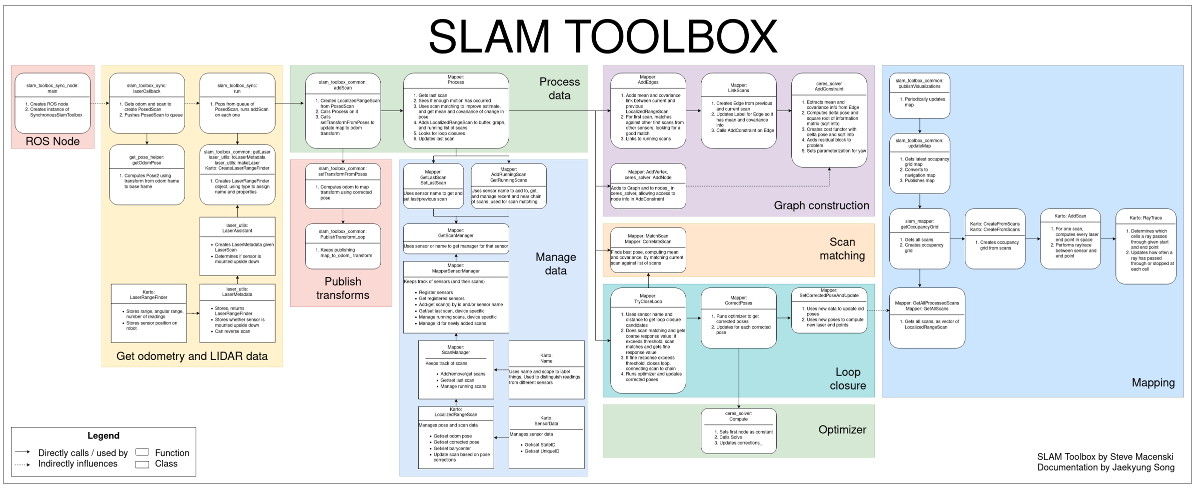
2.1 Core Technology Principles

2.1.1 Front-end Processing (Laser Scan Matching)

- **GICP Algorithm**: It estimates the robot's relative pose by iteratively calculating the optimal match between the current laser scan and the map. Compared to traditional ICP, GICP introduces a probabilistic model, making it more robust to outliers.
- **Motion Compensation**: Uses IMU data to compensate for laser scanning distortion during robot motion, improving matching accuracy in dynamic environments.

2.1.2 Backend Optimization (Graph Optimization)

- **Pose Graph Construction**: Constructs a graph structure by combining the robot's pose nodes and the relative pose constraints obtained from scan matching.
- **Global Optimization**: Uses a nonlinear optimization algorithm (such as the Ceres Solver) to optimize the pose graph, eliminating accumulated errors and achieving global consistency of the map.



GitHub Project Address: https://github.com/SteveMacenski/slam_toolbox

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.

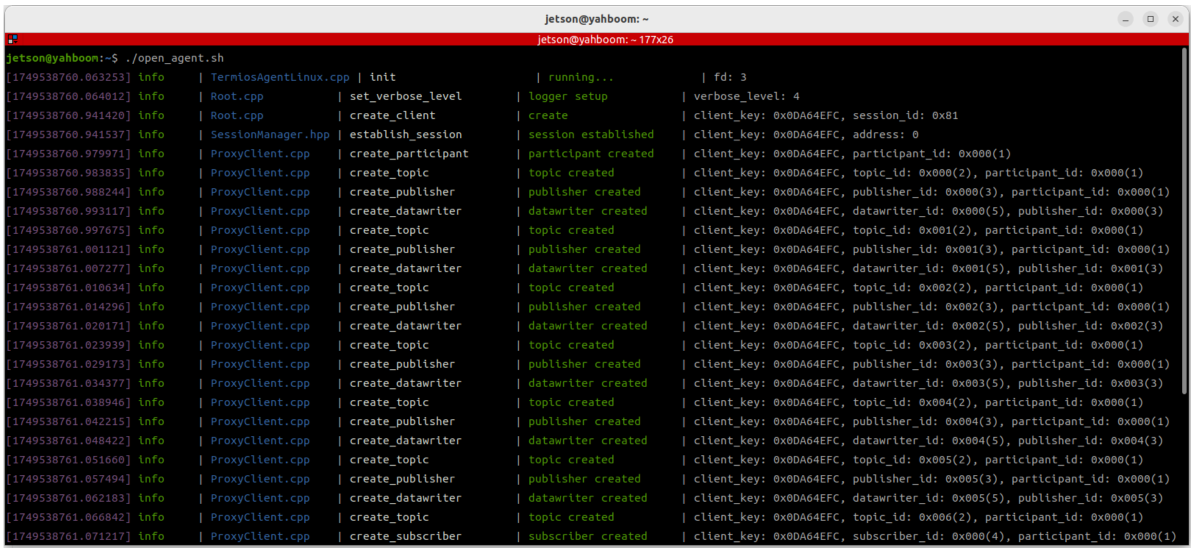
3.2 Start the agent

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

Enter the command in the vehicle terminal:

```
sh start_agent.sh
```

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



```
jetson@yahboom: ~
jetson@yahboom: ~ 177x26
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 Map creation process

Note:

- **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.**
- For the Jetson Nano and Raspberry Pi series controllers, you must first enter the Docker container (for steps, see the [Docker course chapter - Entering the Robot's Docker Container]).

Command to start mapping on the vehicle terminal:

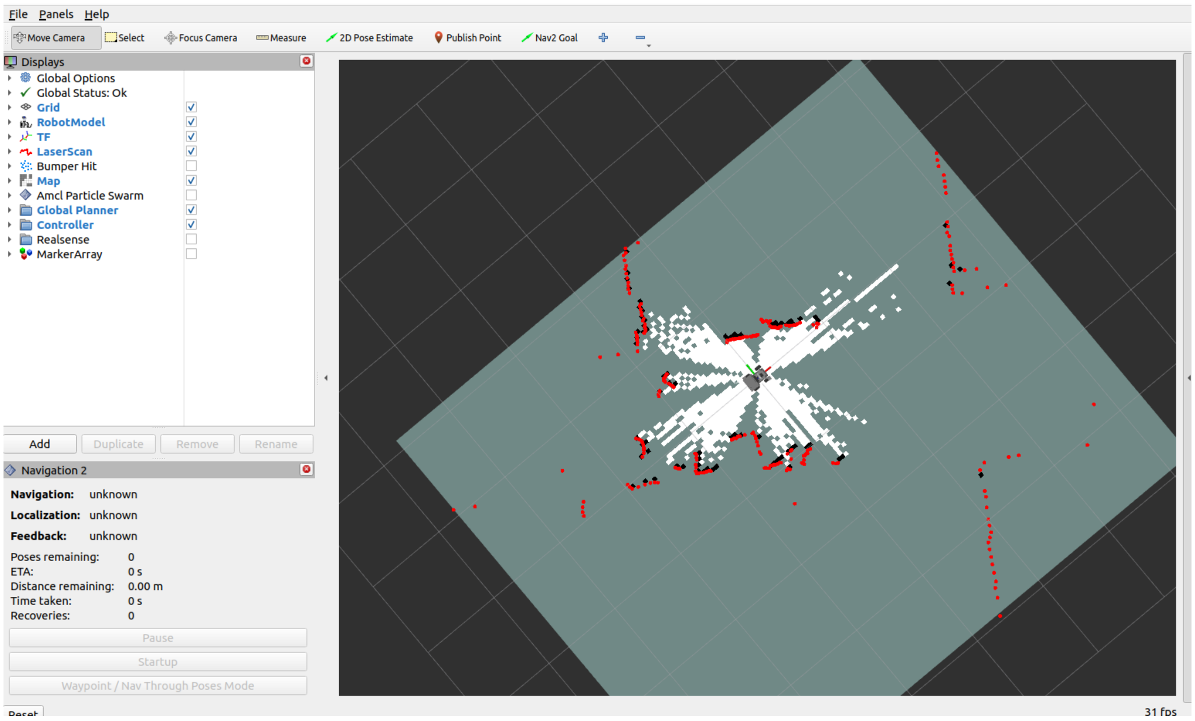
```
ros2 launch slam_mapping slam_toolbox.launch.py
```

Using the virtual machine configuration as an example, open a terminal and launch the RVIZ visualization interface:

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

Command to start the RVIZ visualization interface on the vehicle terminal:

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

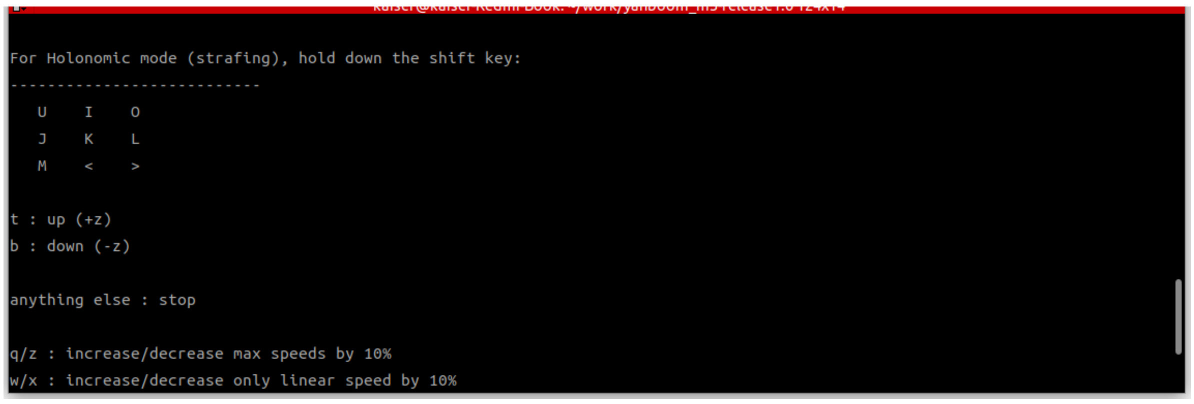


```
Navigation:      unknown
Localization:    unknown
Feedback:        unknown
Poses remaining:      0
ETA:                  0 s
Distance remaining:   0.00 m
Time taken:           0 s
Recoveries:           0
```

Open another terminal in the virtual machine and start the keyboard control node (you can also use a controller remote control. To start the controller control node in advance, refer to [5. Chassis Control - 2. Controller Control]):

```
ros2 run yahboomcar_ctrl yahboom_keyboard
```

Click the terminal window with your mouse and press z to reduce the speed.



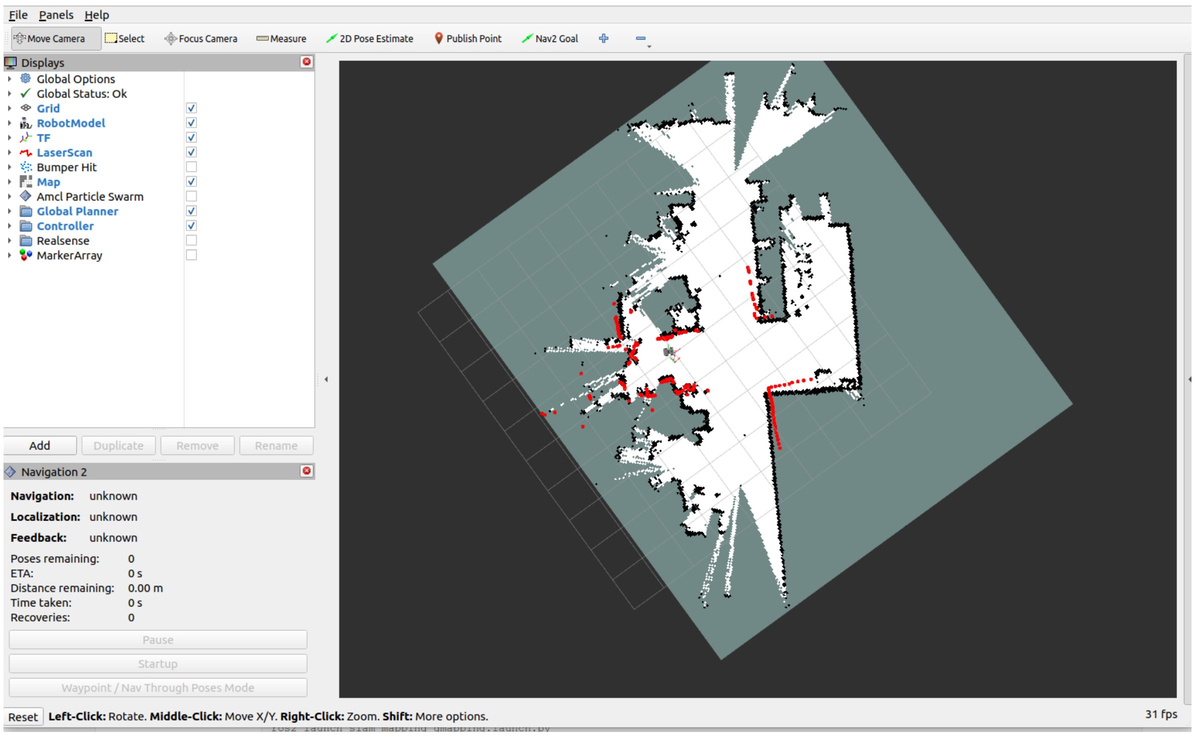
```
For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

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%
```

Click the terminal window with your mouse and press z to reduce the speed. Press I, <, J, and L to control the car's forward, backward, left, and right turns, respectively. Control the car's movement to complete the map building.



Navigation: unknown
Localization: unknown
Feedback: unknown
Poses remaining: 0
ETA: 0 s
Distance remaining: 0.00 m
Time taken: 0 s
Recoveries: 0

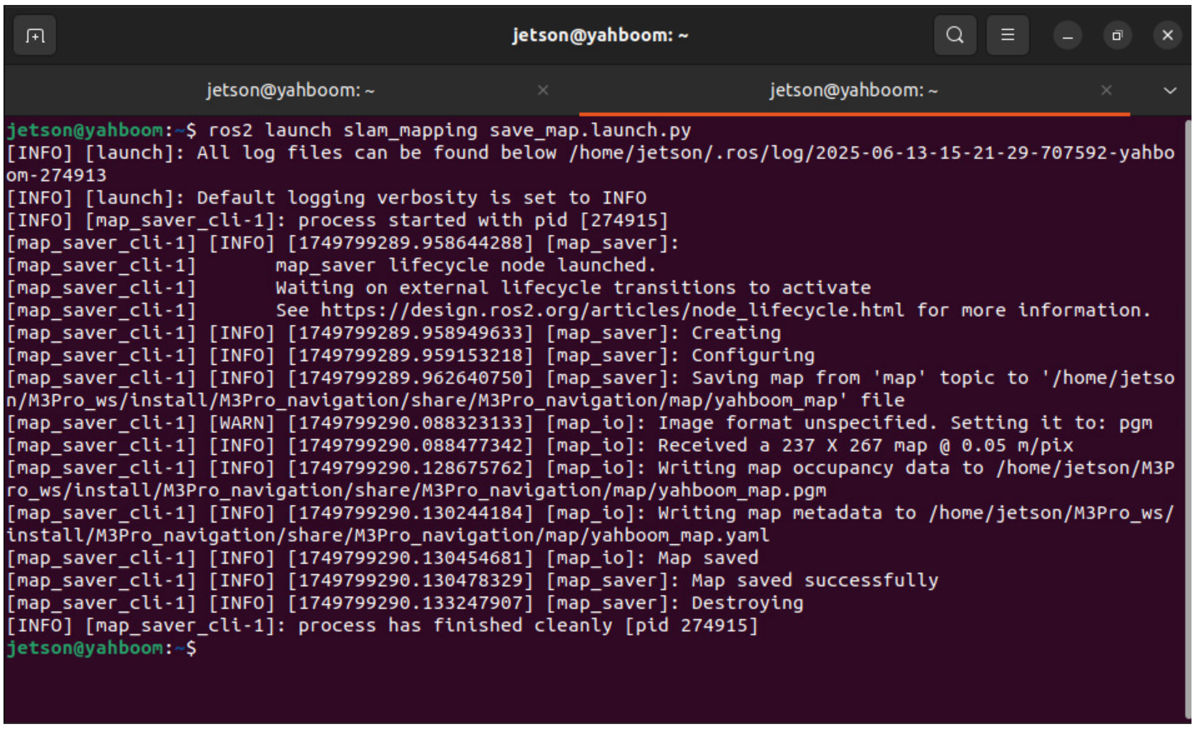
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click: Zoom. Shift: More options.

4.2 Saving the Map

Open a new terminal on the vehicle and save the map.

```
ros2 launch slam_mapping save_map.launch.py
```

The terminal prompts **Map saved "successful" indicates the map was 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/
```

One PGM image, one 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
```

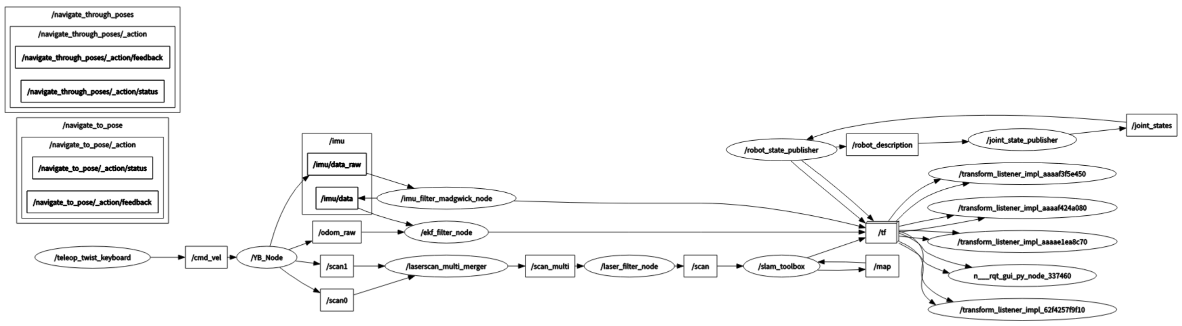
Parameter Explanation:

- image: The path to the map file, either absolute or relative.
- mode: This attribute can be one of trinary, scale, or raw, depending on the selected mode. Trinary is the default.
- resolution: The map resolution, in meters/pixels.
- origin: The 2D pose (x, y, yaw) of the lower-left corner of the map. 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 (this does not affect the interpretation of the thresholds).
- 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 completely free.

5. Node Analysis

5.1 Displaying the Node Computation Graph

```
ros2 run rqt_graph rqt_graph
```

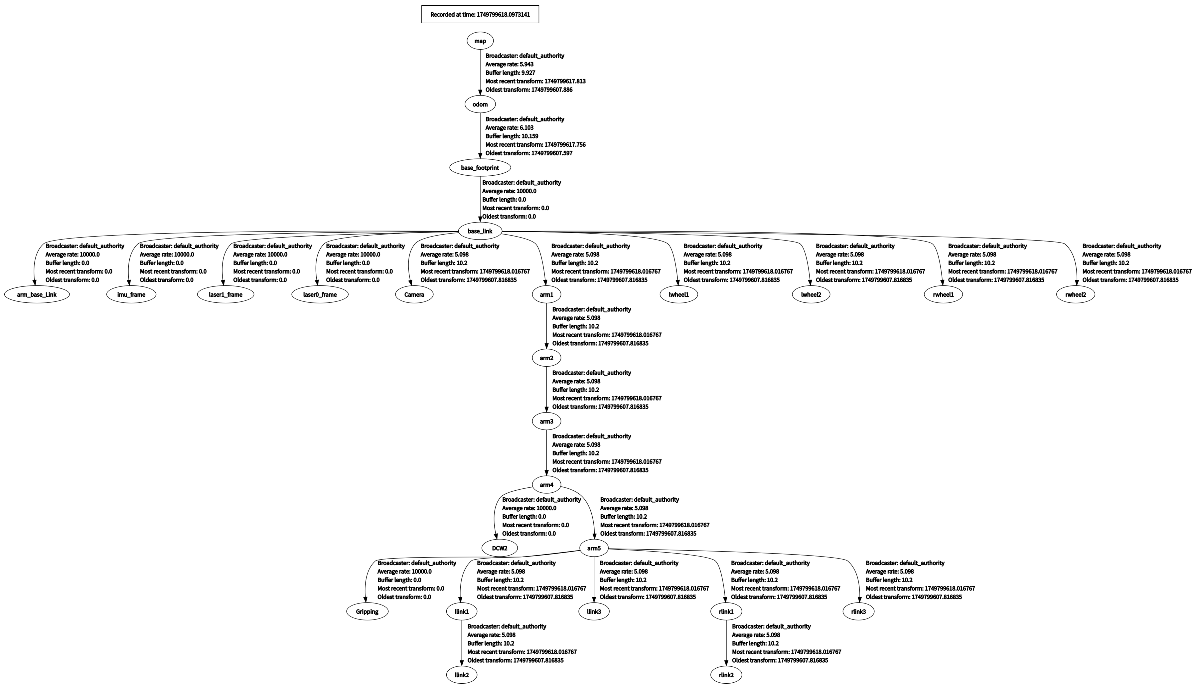

/navigate_through_poses
/navigate_through_poses/_action
/navigate_through_poses/_action/feedback
/navigate_through_poses/_action/status
/navigate_to_pose
/navigate_to_pose/_action
/navigate_to_pose/_action/status
/navigate_to_pose/_action/feedback
/imu
/imu/data_raw
/imu/data
/imu_filter_madgwick_node
/odom_raw
/ekf_filter_node
/teleop_twist_keyboard
/cmd_vel
/YB_Node
/scan1
/scan0
/laserscan_multi_merger
/scan_multi
/laser_filter_node
/scan
/slam_toolbox
/map
/tf
/robot_state_publisher
/robot_description
/joint_state_publisher
/joint_states
/transform_listener_impl_aaaaf3f5e450
/transform_listener_impl_aaaaf424a080
/transform_listener_impl_aaaae1ea8c70
n___rqt_gui_py_node_337460
/transform_listener_impl_62f4257f9f10

5.2 TF Transformation

Run in the VM terminal:

```
ros2 run rqt_tf_tree rqt_tf_tree
```

Image size is too large; the original image can be viewed in this lesson folder.



5.3 slam-toolbox Node Details

```
ros2 node info /slam_toolbox
```

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

```
/slam_toolbox
  Subscribers:
    /map: nav_msgs/msg/OccupancyGrid
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /scan: sensor_msgs/msg/LaserScan
    /slam_toolbox/feedback: visualization_msgs/msg/InteractiveMarkerFeedback
  Publishers:
    /map: nav_msgs/msg/OccupancyGrid
    /map_metadata: nav_msgs/msg/MapMetaData
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /pose: geometry_msgs/msg/PoseWithCovarianceStamped
    /rosout: rcl_interfaces/msg/Log
    /slam_toolbox/graph_visualization: visualization_msgs/msg/MarkerArray
    /slam_toolbox/scan_visualization: sensor_msgs/msg/LaserScan
    /slam_toolbox/update: visualization_msgs/msg/InteractiveMarkerUpdate
    /tf: tf2_msgs/msg/TFMessage
  Service Servers:
    /slam_toolbox/clear_changes: slam_toolbox/srv/Clear
    /slam_toolbox/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /slam_toolbox/deserialize_map: slam_toolbox/srv/DeserializePoseGraph
    /slam_toolbox/dynamic_map: nav_msgs/srv/GetMap
    /slam_toolbox/get_interactive_markers:
visualization_msgs/srv/GetInteractiveMarkers
    /slam_toolbox/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /slam_toolbox/get_parameters: rcl_interfaces/srv/GetParameters
    /slam_toolbox/list_parameters: rcl_interfaces/srv/ListParameters
    /slam_toolbox/manual_loop_closure: slam_toolbox/srv/LoopClosure
    /slam_toolbox/pause_new_measurements: slam_toolbox/srv/Pause
    /slam_toolbox/save_map: slam_toolbox/srv/SaveMap
    /slam_toolbox/serialize_map: slam_toolbox/srv/SerializePoseGraph
    /slam_toolbox/set_parameters: rcl_interfaces/srv/SetParameters
    /slam_toolbox/set_parameters_atomically:
rcl_interfaces/srv/SetParametersAtomically
    /slam_toolbox/toggle_interactive_mode: slam_toolbox/srv/ToggleInteractive
  Service Clients:

  Action Servers:

  Action Clients:
```