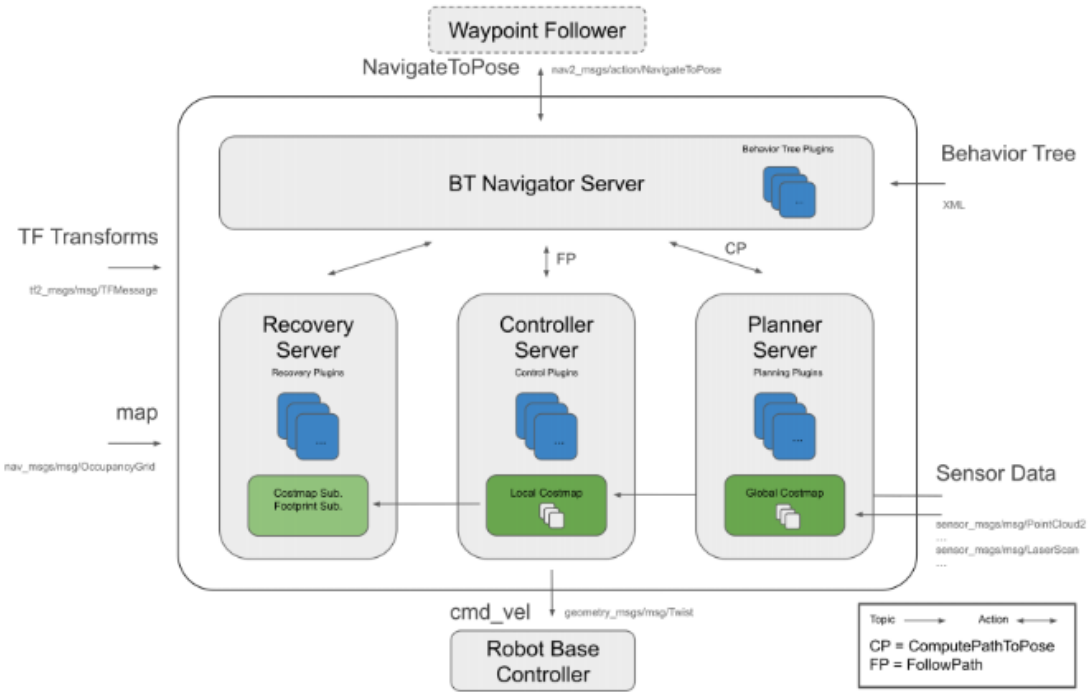
Navigation2 navigation avoid

Note: The ROS_DOMAIN_ID of the Raspberry Pi and the microROS control board need to be consistent. You can check **[MicroROS Control Board Parameter Configuration]** to set the microROS control board ROS_DOMAIN_ID. Check the tutorial **[Connect MicroROS Agent]** to determine whether the IDs are consistent.

1、Introduction to Navigation2

Navigation2 overall architecture diagram



Navigation2 has the following tools:

● Tools for loading, serving, and storing maps (Map Server)

● Tool to locate the robot on the map (AMCL)

● Navigate from point A to point B with Nav2 Planner

● Tool to control the robot during the following path (Nav2 Controller)

● Tools for converting sensor data into cost map representations in the world of robotics (Nav2 Costmap 2D)

● Tools for building complex robot behaviors using behavior trees (Nav2 Behavior Trees and BT Navigator)

● Tool to calculate recovery behavior in the event of a failure (Nav2 Recoveries)

● Nav2 Waypoint Follower

● Tools and watchdog for managing server lifecycles (Nav2 Lifecycle Manager)

● Plugins that enable user-defined algorithms and behaviors (Nav2 Core)

Navigation 2 (Nav 2) is the navigation framework that comes with ROS 2 and aims to move mobile robots from point A to point B in a safe way. As a result, Nav 2 can perform dynamic path planning, calculate motor speed, avoid obstacles, and restore structures.

Nav 2 uses Behavior Trees (BT) to call modular servers to complete an action. Actions can be calculated paths, control efforts, recovery, or other navigation-related actions. These actions are independent nodes that communicate with the Behavior Tree (BT) through the Action Server.

Information reference website:

Navigation2 Documentation：https://navigation.ros.org/index.html

Navigation2 github：https://github.com/ros-planning/navigation2

Papers corresponding to Navigation2：https://arxiv.org/pdf/2003.00368.pdf

Plug-ins provided by Navigation2：https://navigation.ros.org/plugins/index.html#plugins

2、Program function description

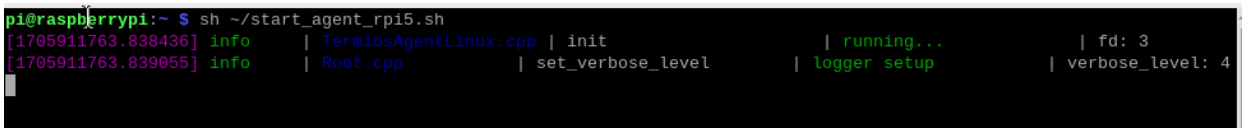
The car connects to the agent, runs the program, and the map will be loaded in rviz. In the rviz interface, use the [2D Pose Estimate] tool to give the initial pose of the car, and then use the [2D Goal Pose] tool to give the car a target point. The car will plan a path based on its own environment and move to the destination according to the planned path. If it encounters an obstacle during the process, it will avoid the obstacle by itself and stop when it reaches the destination.

3、Query car information

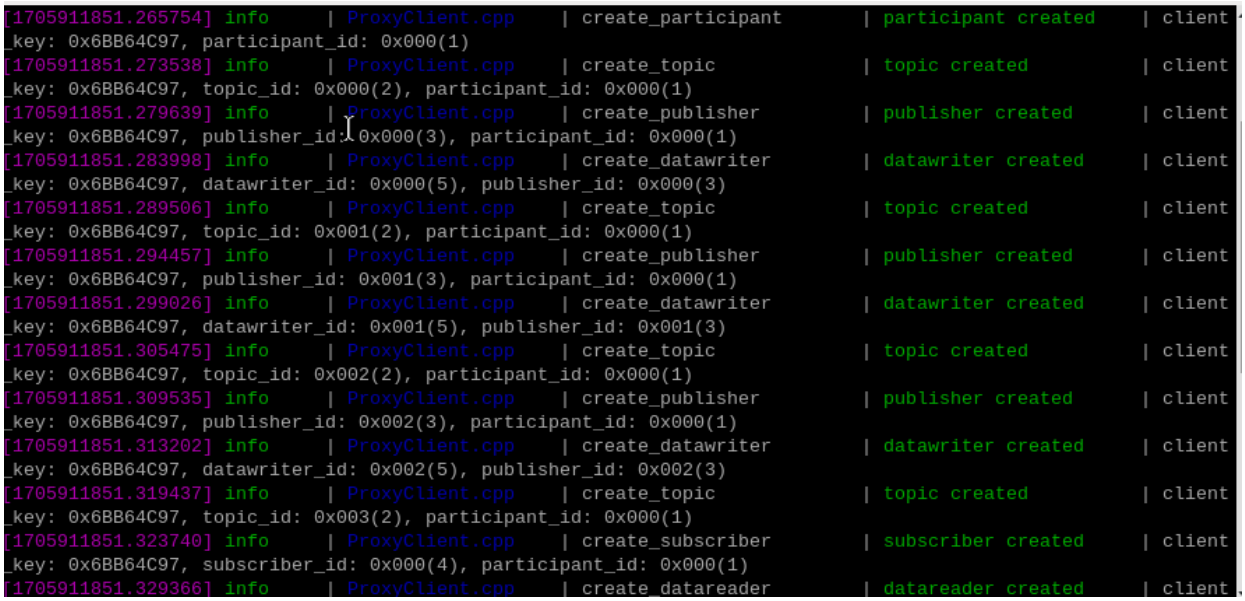
3.1、Start and connect to the agent

After the Raspberry Pi is successfully powered on, open the terminal and enter the following command to open the agent.

```
sh ~/start_agent_rpi5.sh
```



Press the reset button on the microROS control board and wait for the car to connect to the agent. The connection is successful as shown in the figure below.

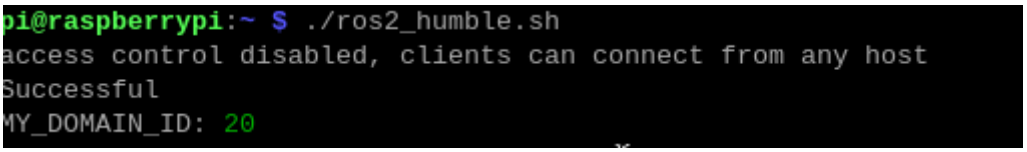


3.2、Enter the car docker

Open another terminal and enter the following command to enter docker.

```
sh ros2_humble.sh
```

When the following interface appears, you have successfully entered docker. Now you can control the car through commands.



4、starting program

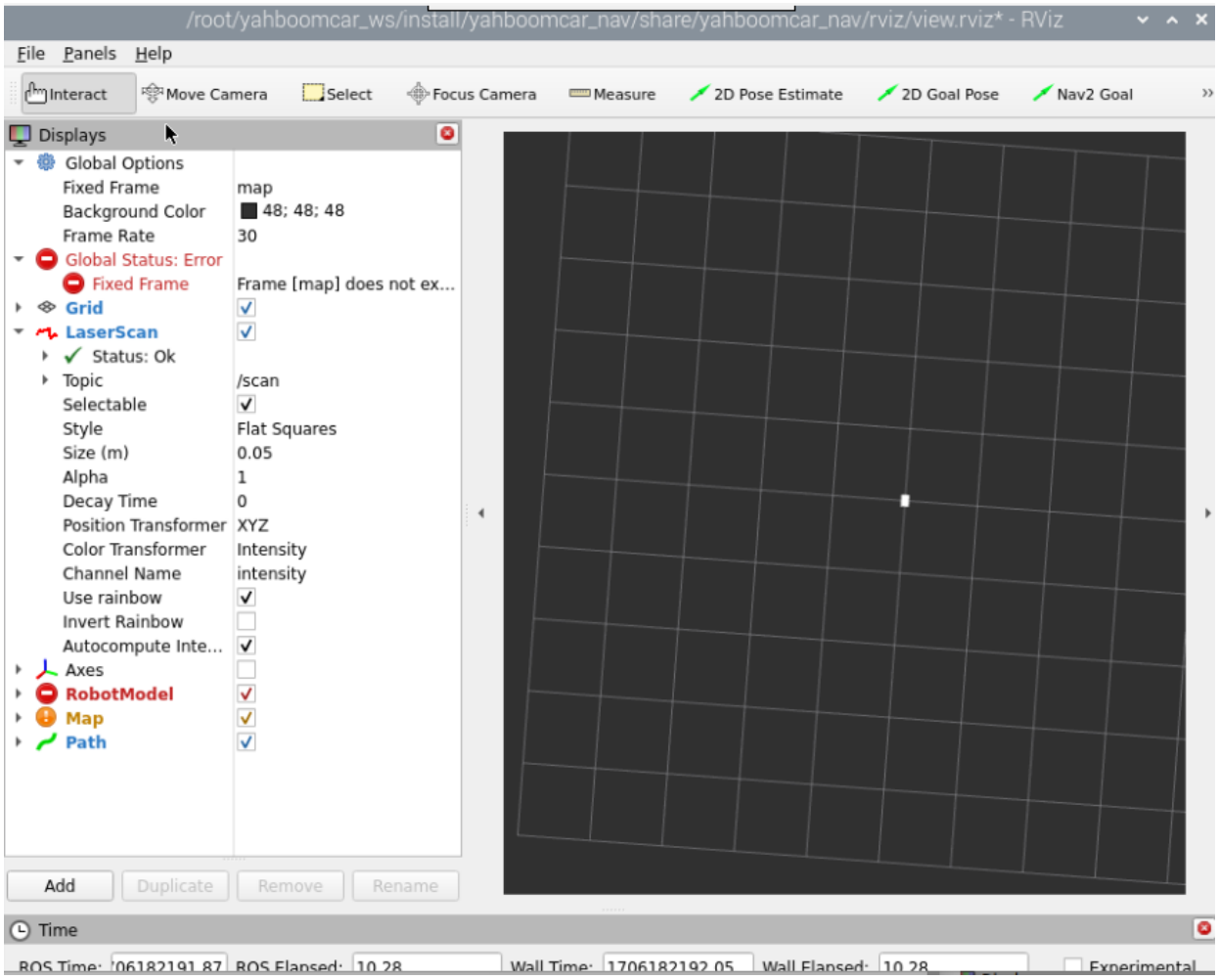
First, start the car to process the underlying data program, and enter the following command in the terminal

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

```
[INFO] [imu_filter_madgwick_node-1]: process started with pid [6263]
[INFO] [ekf_node-2]: process started with pid [6265]
[INFO] [static_transform_publisher-3]: process started with pid [6267]
[INFO] [joint_state_publisher-4]: process started with pid [6269]
[INFO] [robot_state_publisher-5]: process started with pid [6271]
[INFO] [static_transform_publisher-6]: process started with pid [6286]
[static_transform_publisher-3] [WARN] [1706181650.342105372] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-3] [INFO] [1706181650.459314055] [base_link_to_base_imu]: Spinning until stopped - publishing transform
[static_transform_publisher-3] translation: ('-0.002999', '-0.003000', '0.031701')
[static_transform_publisher-3] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-3] from 'base_link' to 'imu_frame'
[imu_filter_madgwick_node-1] [INFO] [1706181650.478942143] [imu_filter]: Starting ImuFilter
[imu_filter_madgwick_node-1] [INFO] [1706181650.480114862] [imu_filter]: Using dt computed from message headers
[imu_filter_madgwick_node-1] [INFO] [1706181650.480197121] [imu_filter]: The gravity vector is kept in the IMU message.
[imu_filter_madgwick_node-1] [INFO] [1706181650.480631749] [imu_filter]: Imu filter gain set to 0.100000
[imu_filter_madgwick_node-1] [INFO] [1706181650.480707508] [imu_filter]: Gyro drift bias set to 0.000000
[imu_filter_madgwick_node-1] [INFO] [1706181650.480720249] [imu_filter]: Magnetometer bias values: 0.000000 0.000000 0.000000
[static_transform_publisher-6] [WARN] [1706181650.493533858] []: Old-style arguments are deprecated; see --help for new-style arguments
[robot_state_publisher-5] [WARN] [1706181650.639387954] [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] [1706181650.640061915] [robot_state_publisher]: got segment base_link
[robot_state_publisher-5] [INFO] [1706181650.640174267] [robot_state_publisher]: got segment imu_Link
[robot_state_publisher-5] [INFO] [1706181650.640191229] [robot_state_publisher]: got segment jq1_Link
[robot_state_publisher-5] [INFO] [1706181650.640201822] [robot_state_publisher]: got segment jq2_Link
[robot_state_publisher-5] [INFO] [1706181650.640211952] [robot_state_publisher]: got segment radar_Link
[robot_state_publisher-5] [INFO] [1706181650.640221211] [robot_state_publisher]: got segment yh_Link
[robot_state_publisher-5] [INFO] [1706181650.640229452] [robot_state_publisher]: got segment yq_Link
[robot_state_publisher-5] [INFO] [1706181650.640238470] [robot_state_publisher]: got segment zh_Link
[robot_state_publisher-5] [INFO] [1706181650.640246655] [robot_state_publisher]: got segment zq_Link
[imu_filter_madgwick_node-1] [INFO] [1706181650.655098332] [imu_filter]: First IMU message received.
[static_transform_publisher-6] [INFO] [1706181650.681177050] [static_transform_publisher_JarNTEa1OrW2k0Zb]: Spinning until stopped - publishing transform
[static_transform_publisher-6] translation: ('0.000000', '0.000000', '0.050000')
[static_transform_publisher-6] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-6] from 'base_footprint' to 'base_link'
[joint_state_publisher-4] [INFO] [1706181650.989117137] [joint_state_publisher]: Waiting for robot_description to be published on the robot_description topic...
```

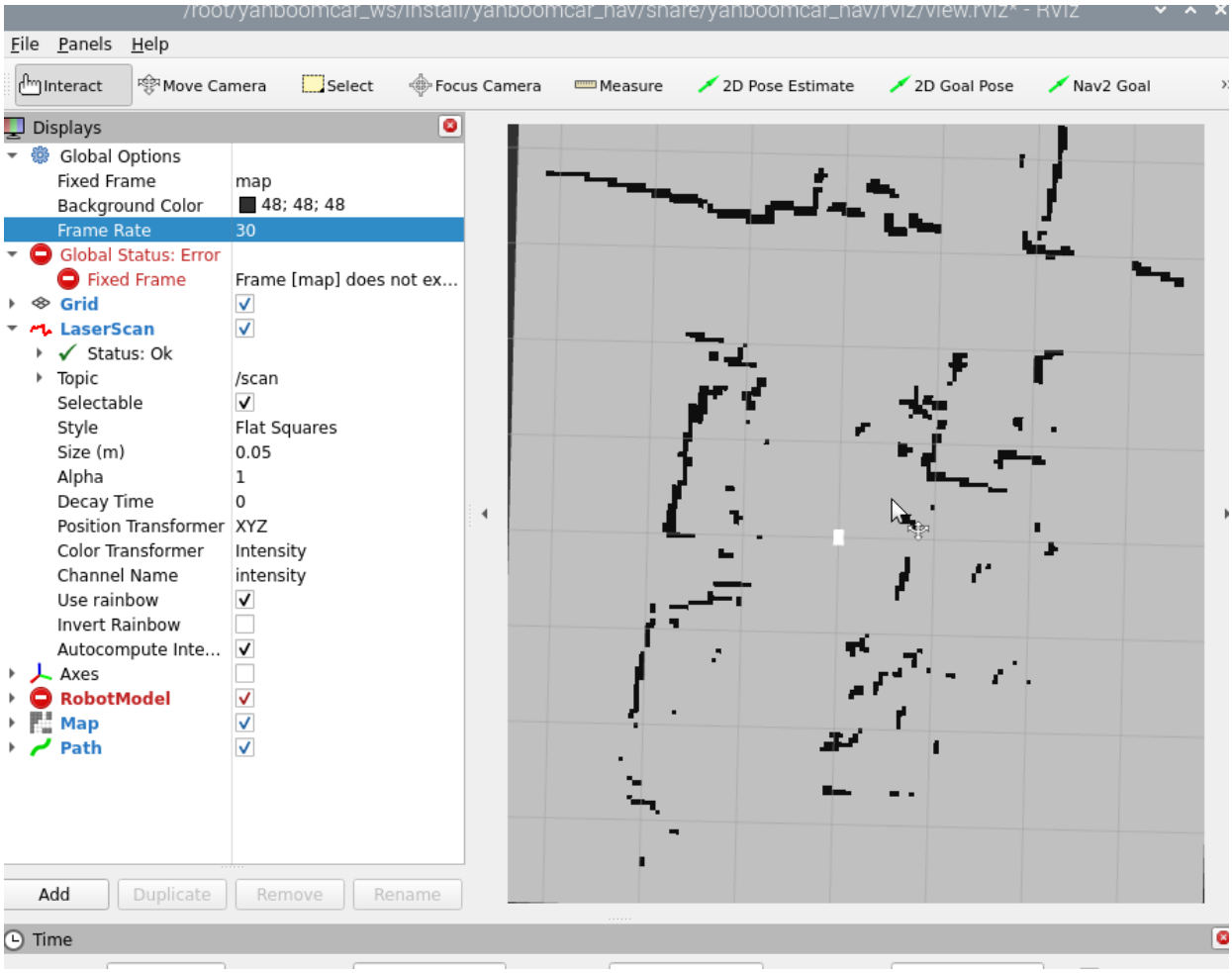
Then enter the command in the terminal to start rviz for visual navigation.

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

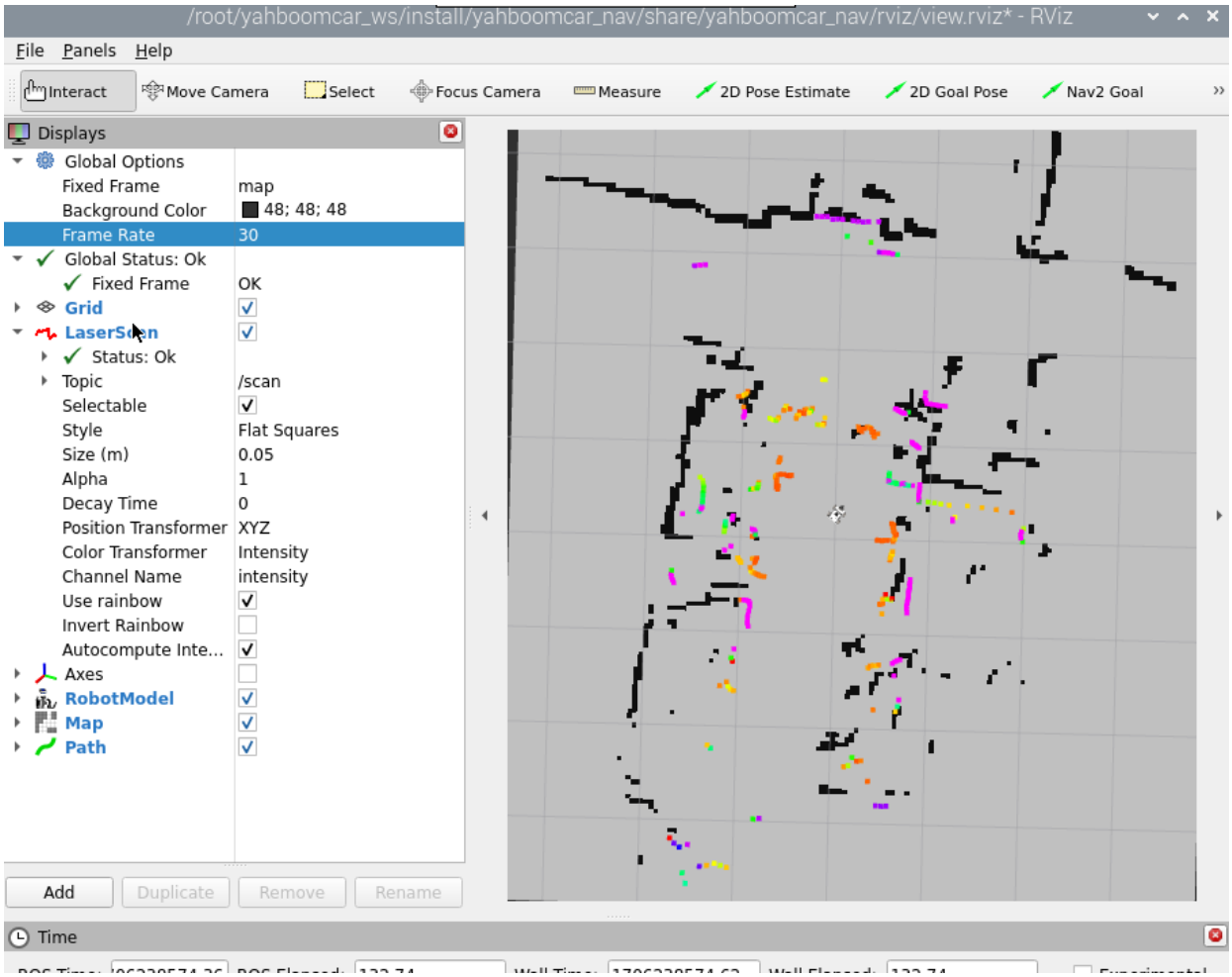


At this time, the map loading is not displayed because the navigation program has not been started yet, so there is no map loading. Next, run the navigation node and enter the following command in the terminal.

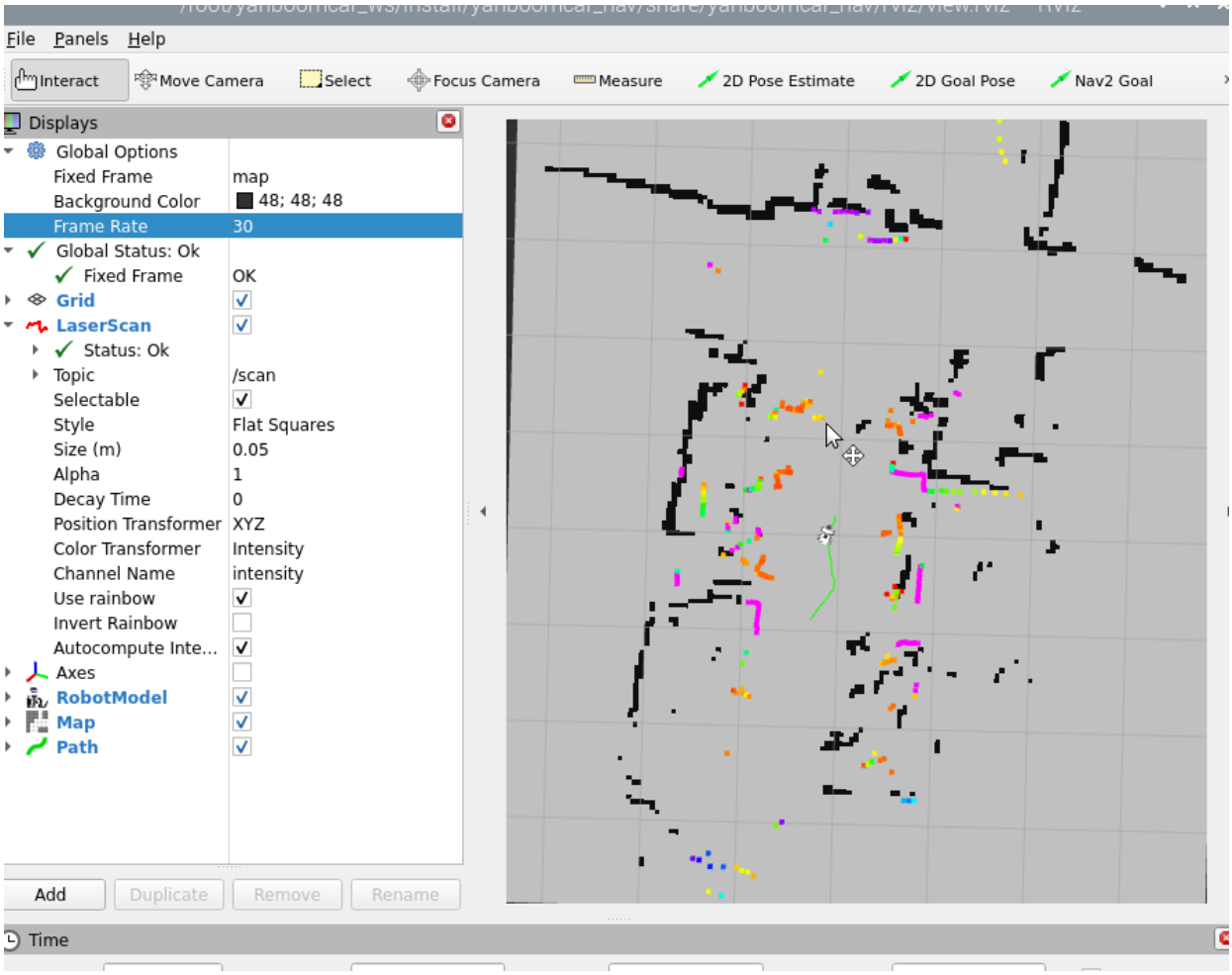
```
ros2 launch  yahboomcar_nav navigation_dwb_launch.py
```



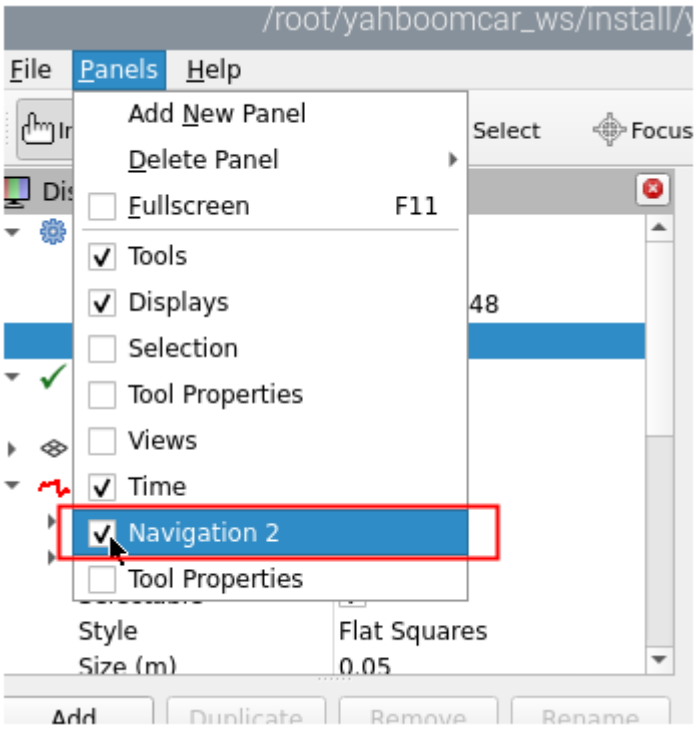
At this point you can see that the map is loaded, and then we click [2D Pose Estimate] to set the initial pose for the car. According to the position of the car in the actual environment, click and drag with the mouse in rviz, and the car model moves according to the position we set Location. As shown in the figure below, if the area scanned by the radar roughly coincides with the actual obstacle, it means the pose is accurate.



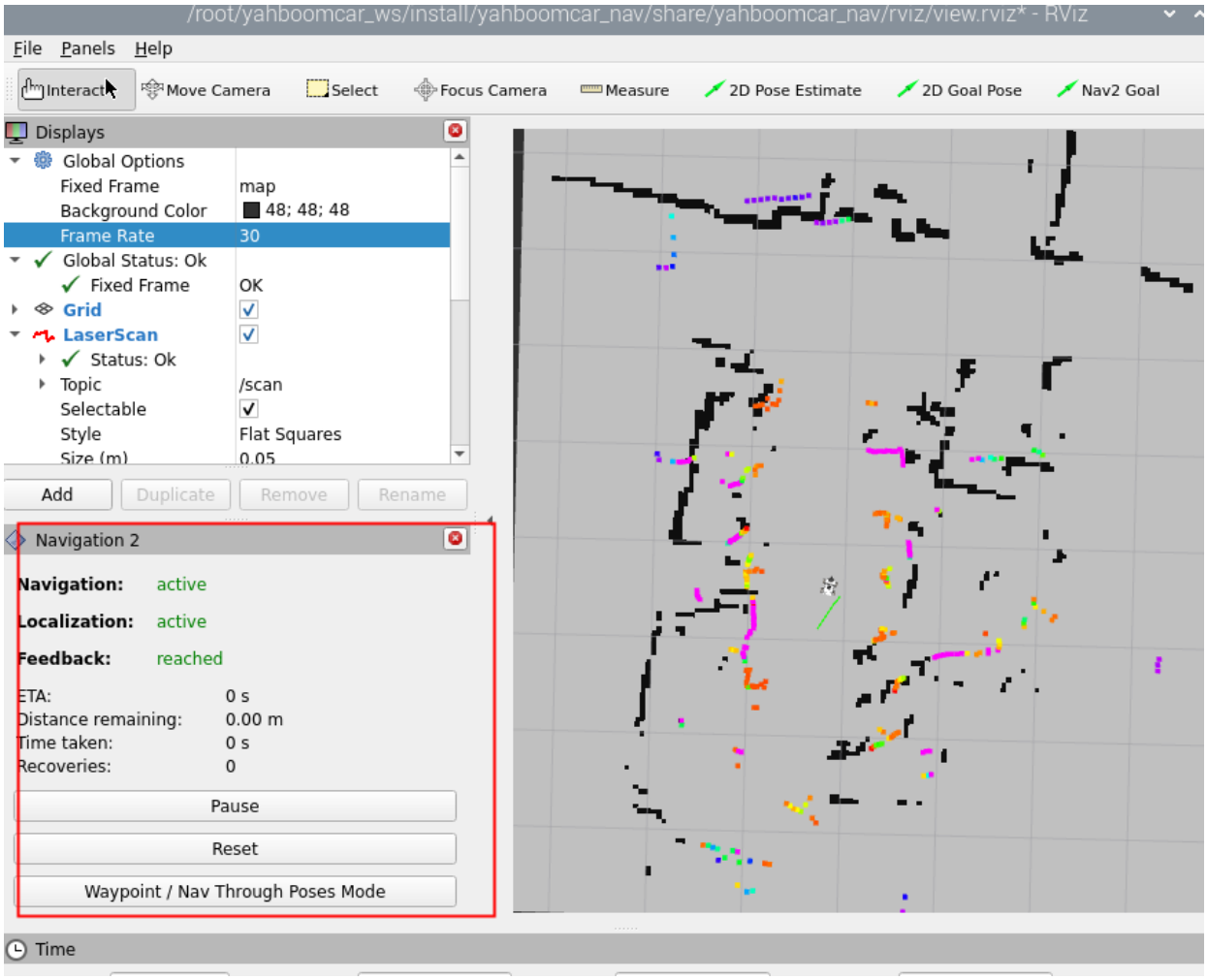
For single-point navigation, click the [2D Goal Pose] tool, and then select a target point in rviz. The car will plan a path based on the surrounding situation and move along the path to the target point.



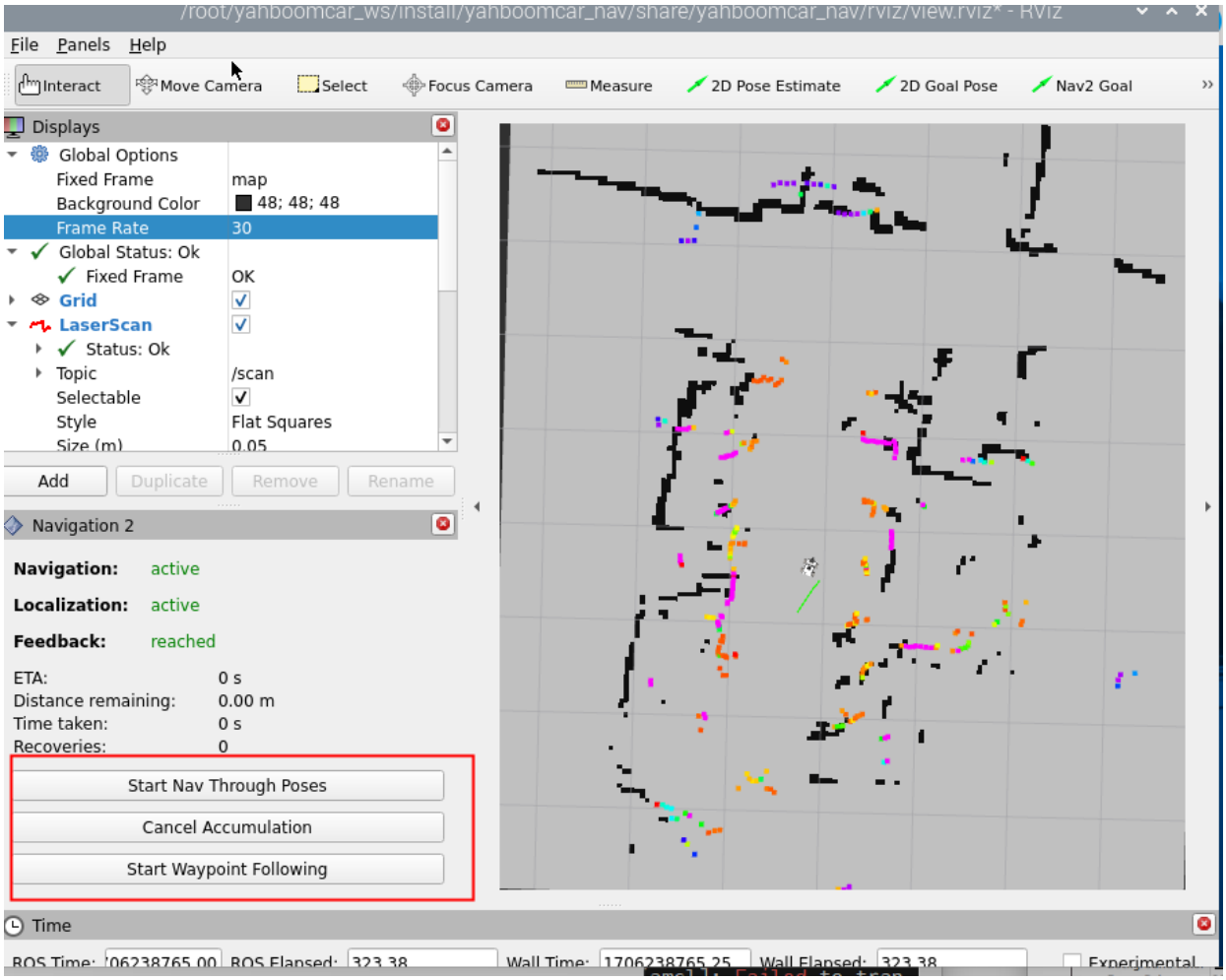
For multi-point navigation, you need to add the nav2 plug-in.



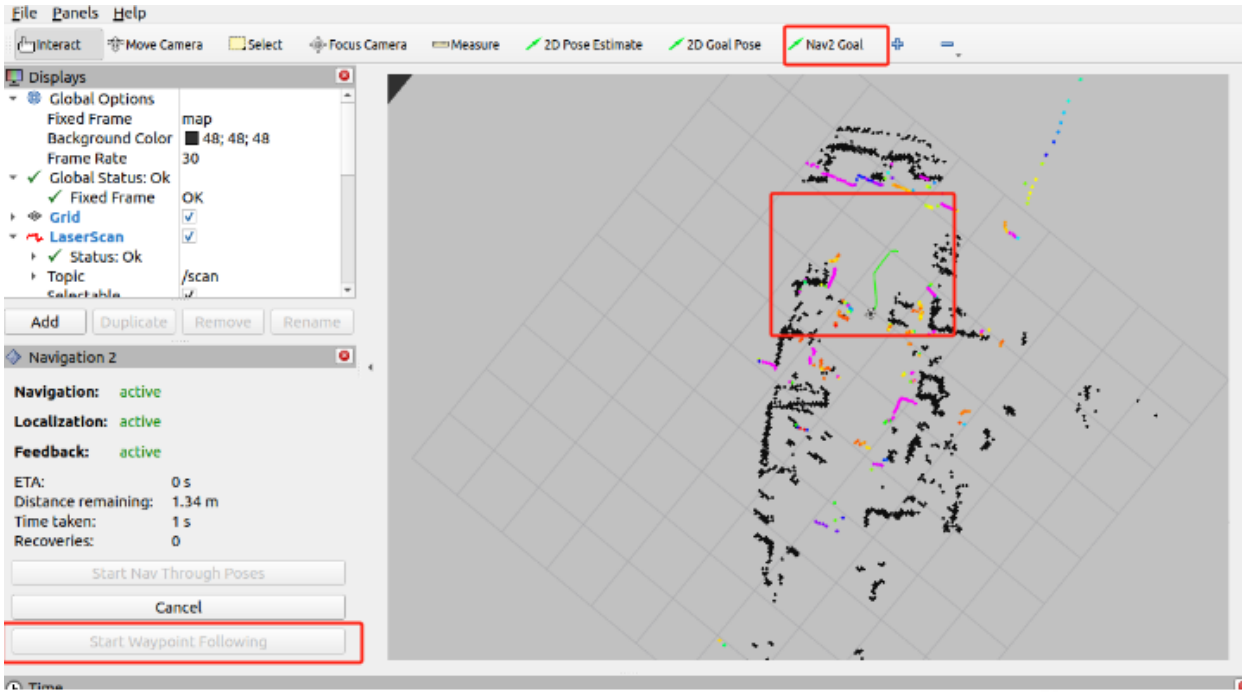
After adding, rviz displays as follows.



Then click [Waypoint/Nav Through Poses Mode].



Use [Nav2 Goal] in the rivz toolbar to specify any target point, and then click [Start Waypoint Following] to start planning path navigation. The car will automatically go to the next point according to the order of the selected points, and no operation is required after reaching the target point. After reaching the last point, the car stops and waits for the next instruction.



5、View node communication diagram

Terminal input command

```
ros2 run rqt_graph rqt_graph
```

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

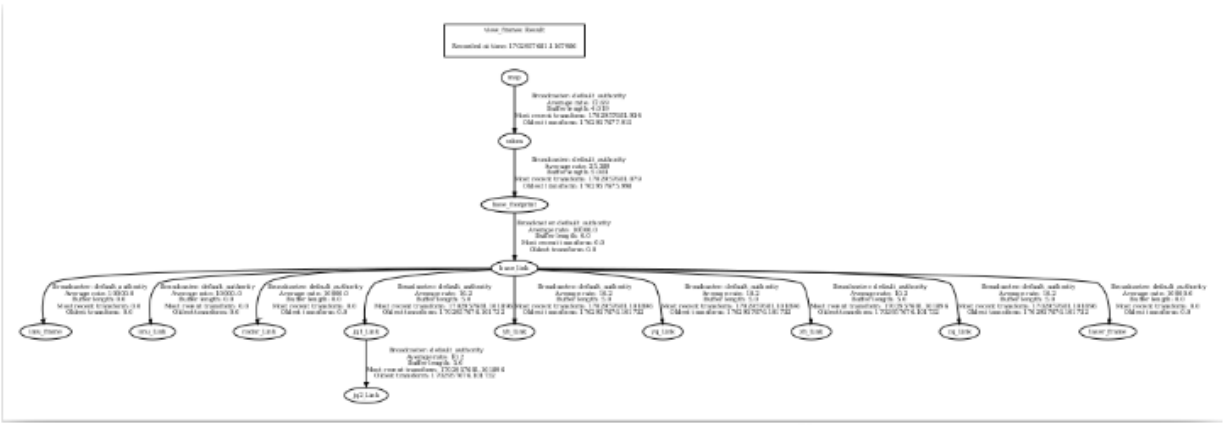
6、View TF tree

Terminal input command

```
ros2 run tf2_tools view_frames
```

```
root@raspberrypi:~# ros2 run tf2_tools view_frames
[INFO] [1706240274.802480879] [view_frames]: Listening to tf data for 5.0 seconds...
[INFO] [1706240279.805539455] [view_frames]: Generating graph in frames.pdf file...
[INFO] [1706240279.811022965] [view_frames]: Result:tf2_msgs.srv.FrameGraph_Response(frame_yaml=
"odom: \n  parent: 'map'\n  broadcaster: 'default_authority'\n  rate: 193.930\n  most_recent_tra
nsform: 1706240279.804413\n  oldest_transform: 1706240274.771664\n  buffer_length: 5.033\nimu_fr
ame: \n  parent: 'base_link'\n  broadcaster: 'default_authority'\n  rate: 10000.000\n  most_rece
nt_transform: 0.000000\n  oldest_transform: 0.000000\n  buffer_length: 0.000\nbase_link: \n  par
ent: 'base_footprint'\n  broadcaster: 'default_authority'\n  rate: 10000.000\n  most_recent_tran
sform: 0.000000\n  oldest_transform: 0.000000\n  buffer_length: 0.000\nbase_footprint: \n  paren
t: 'odom'\n  broadcaster: 'default_authority'\n  rate: 19.545\n  most_recent_transform: 17062402
79.773000\n  oldest_transform: 1706240274.810000\n  buffer_length: 4.963\nlaser_frame: \n  paren
t: '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\njq1_Link: \n  parent: 'base_link
'\n  broadcaster: 'default_authority'\n  rate: 10.204\n  most_recent_transform: 1706240279.71725
1\n  oldest_transform: 1706240274.817256\n  buffer_length: 4.900\njq2_Link: \n  parent: 'jq1_Lin
k'\n  broadcaster: 'default_authority'\n  rate: 10.204\n  most_recent_transform: 1706240279.7172
51\n  oldest_transform: 1706240274.817256\n  buffer_length: 4.900\nyh_Link: \n  parent: 'base_li
nk'\n  broadcaster: 'default_authority'\n  rate: 10.204\n  most_recent_transform: 1706240279.717
251\n  oldest_transform: 1706240274.817256\n  buffer_length: 4.900\nyq_Link: \n  parent: 'base_l
ink'\n  broadcaster: 'default_authority'\n  rate: 10.204\n  most_recent_transform: 1706240279.71
7251\n  oldest_transform: 1706240274.817256\n  buffer_length: 4.900\nzh_Link: \n  parent: 'base_
link'\n  broadcaster: 'default_authority'\n  rate: 10.204\n  most_recent_transform: 1706240279.7
17251\n  oldest_transform: 1706240274.817256\n  buffer_length: 4.900\nzq_Link: \n  parent: 'base
_link'\n  broadcaster: 'default_authority'\n  rate: 10.204\n  most_recent_transform: 1706240279.
717251\n  oldest_transform: 1706240274.817256\n  buffer_length: 4.900\nimu_Link: \n  parent: 'ba
se_link'\n  broadcaster: 'default_authority'\n  rate: 10000.000\n  most_recent_transform: 0.0000
00\n  oldest_transform: 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  oldes
t_transform: 0.000000\n  buffer_length: 0.000\n")
root@raspberrypi:~#
```

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



7、Code analysis

Here we only describe the navigation_dwb_launch.py of navigation. The path of this file is as follows.

```
/root/yahboomcar_ws/src/yahboomcar_nav/launch
```

navigation_dwb_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.actions import IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource
from launch.substitutions import LaunchConfiguration
from launch_ros.actions import Node

def generate_launch_description():
    package_path = get_package_share_directory('yahboomcar_nav')
    nav2_bringup_dir = get_package_share_directory('nav2_bringup')

    use_sim_time = LaunchConfiguration('use_sim_time', default='false')
    map_yaml_path = LaunchConfiguration(
        'maps', default=os.path.join(package_path, 'maps', 'yahboom_map.yaml'))
    nav2_param_path = LaunchConfiguration('params_file', default=os.path.join(
        package_path, 'params', 'dwb_nav_params.yaml'))

    return LaunchDescription([
        DeclareLaunchArgument('use_sim_time', default_value=use_sim_time,
                              description='Use simulation (Gazebo) clock if true'),
        DeclareLaunchArgument('maps', default_value=map_yaml_path,
                              description='Full path to map file to load'),
        DeclareLaunchArgument('params_file', default_value=nav2_param_path,
                              description='Full path to param file to load'),

        IncludeLaunchDescription(
            PythonLaunchDescriptionSource(
                [nav2_bringup_dir, '/launch', '/bringup_launch.py']),
            launch_arguments={
                'map': map_yaml_path,
                'use_sim_time': use_sim_time,
                'params_file': nav2_param_path}.items(),
        ),
        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']
        ),
        Node(
```

```
            package='yahboomcar_nav',
            executable='stop_car'
        )
    ])
```

The following nodes are started here

- base_link_to_base_laser：Publish static TF transformation;
- stop_car：For the parking node, after ctrl c exits the program, the parking speed will be announced to the car;
- bringup_launch.py：Launch navigation launch file, the file is located at, `/opt/ros/humble/share/nav2_bringup/launch`

In addition, a navigation parameter configuration file dwb_nav_params.yaml and a map file yahboom_map.yaml are also loaded. The path to the navigation parameter table is as follows.

```
/root/yahboomcar_ws/src/yahboomcar_nav/params
```

The map file is as follows.

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

dwb_nav_params.yaml，

```yaml
amcl:
  ros__parameters:
    use_sim_time: False
    alpha1: 0.2
    alpha2: 0.2
    alpha3: 0.2
    alpha4: 0.2
    alpha5: 0.2
    base_frame_id: "base_footprint"
    beam_skip_distance: 0.5
    beam_skip_error_threshold: 0.9
    beam_skip_threshold: 0.3
    do_beamskip: false
    global_frame_id: "map"
    lambda_short: 0.1
    laser_likelihood_max_dist: 2.0
    laser_max_range: 100.0
    laser_min_range: -1.0
    laser_model_type: "likelihood_field"
    max_beams: 60
    max_particles: 2000
    min_particles: 500
    odom_frame_id: "odom"
    pf_err: 0.05
    pf_z: 0.99
    recovery_alpha_fast: 0.0
```

```
    recovery_alpha_slow: 0.0
    resample_interval: 1
    robot_model_type: "nav2_amcl::DifferentialMotionModel"
    save_pose_rate: 0.5
    sigma_hit: 0.2
    tf_broadcast: true
    transform_tolerance: 1.0
    update_min_a: 0.2
    update_min_d: 0.25
    z_hit: 0.5
    z_max: 0.05
    z_rand: 0.5
    z_short: 0.05
    scan_topic: scan

bt_navigator:
  ros__parameters:
    use_sim_time: False
    global_frame: map
    robot_base_frame: base_link
    odom_topic: /odom
    bt_loop_duration: 10
    default_server_timeout: 20
    default_bt_xml_filename: "navigate_to_pose_w_replanning_and_recovery.xml"
    # 'default_nav_through_poses_bt_xml' and 'default_nav_to_pose_bt_xml' are use
defaults:
    # nav2_bt_navigator/navigate_to_pose_w_replanning_and_recovery.xml
    # nav2_bt_navigator/navigate_through_poses_w_replanning_and_recovery.xml
    # They can be set here or via a RewrittenYaml remap from a parent launch file to
Nav2.
    plugin_lib_names:
      - nav2_compute_path_to_pose_action_bt_node
      - nav2_compute_path_through_poses_action_bt_node
      - nav2_smooth_path_action_bt_node
      - nav2_follow_path_action_bt_node
      - nav2_spin_action_bt_node
      - nav2_wait_action_bt_node
      - nav2_assisted_teleop_action_bt_node
      - nav2_back_up_action_bt_node
      - nav2_drive_on_heading_bt_node
      - nav2_clear_costmap_service_bt_node
      - nav2_is_stuck_condition_bt_node
      - nav2_goal_reached_condition_bt_node
      - nav2_goal_updated_condition_bt_node
      - nav2_globally_updated_goal_condition_bt_node
      - nav2_is_path_valid_condition_bt_node
      - nav2_initial_pose_received_condition_bt_node
      - nav2_reinitialize_global_localization_service_bt_node
      - nav2_rate_controller_bt_node
      - nav2_distance_controller_bt_node
      - nav2_speed_controller_bt_node
      - nav2_truncate_path_action_bt_node
```

```yaml
      - nav2_truncate_path_local_action_bt_node
      - nav2_goal_updater_node_bt_node
      - nav2_recovery_node_bt_node
      - nav2_pipeline_sequence_bt_node
      - nav2_round_robin_node_bt_node
      - nav2_transform_available_condition_bt_node
      - nav2_time_expired_condition_bt_node
      - nav2_path_expiring_timer_condition
      - nav2_distance_traveled_condition_bt_node
      - nav2_single_trigger_bt_node
      - nav2_goal_updated_controller_bt_node
      - nav2_is_battery_low_condition_bt_node
      - nav2_navigate_through_poses_action_bt_node
      - nav2_navigate_to_pose_action_bt_node
      - nav2_remove_passed_goals_action_bt_node
      - nav2_planner_selector_bt_node
      - nav2_controller_selector_bt_node
      - nav2_goal_checker_selector_bt_node
      - nav2_controller_cancel_bt_node
      - nav2_path_longer_on_approach_bt_node
      - nav2_wait_cancel_bt_node
      - nav2_spin_cancel_bt_node
      - nav2_back_up_cancel_bt_node
      - nav2_assisted_teleop_cancel_bt_node
      - nav2_drive_on_heading_cancel_bt_node
      - nav2_is_battery_charging_condition_bt_node

bt_navigator_navigate_through_poses_rclcpp_node:
  ros__parameters:
    use_sim_time: False

bt_navigator_navigate_to_pose_rclcpp_node:
  ros__parameters:
    use_sim_time: False

controller_server:
  ros__parameters:
    use_sim_time: False
    controller_frequency: 20.0
    min_x_velocity_threshold: 0.001
    min_y_velocity_threshold: 0.5
    min_theta_velocity_threshold: 0.001
    failure_tolerance: 0.3
    progress_checker_plugin: "progress_checker"
    goal_checker_plugins: ["general_goal_checker"] # "precise_goal_checker"
    controller_plugins: ["FollowPath"]

    # Progress checker parameters
    progress_checker:
      plugin: "nav2_controller::SimpleProgressChecker"
      required_movement_radius: 0.5
      movement_time_allowance: 10.0
```

```
    # Goal checker parameters
    #precise_goal_checker:
    #  plugin: "nav2_controller::SimpleGoalChecker"
    #  xy_goal_tolerance: 0.25
    #  yaw_goal_tolerance: 0.25
    #  stateful: True
    general_goal_checker:
      stateful: True
      plugin: "nav2_controller::SimpleGoalChecker"
      xy_goal_tolerance: 0.25
      yaw_goal_tolerance: 0.25
    # DWB parameters
    FollowPath:
      plugin: "dwb_core::DWBLocalPlanner"
      debug_trajectory_details: True
      min_vel_x: -0.20
      min_vel_y: 0.0
      max_vel_x: 0.30
      max_vel_y: 0.0
      max_vel_theta: 1.0
      min_speed_xy: -0.20
      max_speed_xy: 0.30
      min_speed_theta: -0.5
      # Add high threshold velocity for turtlebot 3 issue.
      # https://github.com/ROBOTIS-GIT/turtlebot3_simulations/issues/75
      acc_lim_x: 2.5
      acc_lim_y: 0.0
      acc_lim_theta: 3.2
      decel_lim_x: -2.5
      decel_lim_y: 0.0
      decel_lim_theta: -3.2
      vx_samples: 20
      vy_samples: 5
      vtheta_samples: 20
      sim_time: 1.7
      linear_granularity: 0.05
      angular_granularity: 0.025
      transform_tolerance: 0.2
      xy_goal_tolerance: 0.25
      trans_stopped_velocity: 0.25
      short_circuit_trajectory_evaluation: True
      stateful: True
      critics: ["RotateToGoal", "Oscillation", "BaseObstacle", "GoalAlign",
"PathAlign", "PathDist", "GoalDist"]
      BaseObstacle.scale: 0.02
      PathAlign.scale: 32.0
      PathAlign.forward_point_distance: 0.1
      GoalAlign.scale: 24.0
      GoalAlign.forward_point_distance: 0.1
      PathDist.scale: 32.0
      GoalDist.scale: 24.0
      RotateToGoal.scale: 32.0
```

```yaml
      RotateToGoal.slowing_factor: 5.0
      RotateToGoal.lookahead_time: -1.0

local_costmap:
  local_costmap:
    ros__parameters:
      update_frequency: 5.0
      publish_frequency: 2.0
      global_frame: odom
      robot_base_frame: base_link
      use_sim_time: False
      rolling_window: true
      width: 3
      height: 3
      resolution: 0.05
      robot_radius: 0.22
      plugins: ["voxel_layer", "inflation_layer"]
      inflation_layer:
        plugin: "nav2_costmap_2d::InflationLayer"
        cost_scaling_factor: 3.0
        inflation_radius: 0.55
      voxel_layer:
        plugin: "nav2_costmap_2d::VoxelLayer"
        enabled: True
        publish_voxel_map: True
        origin_z: 0.0
        z_resolution: 0.05
        z_voxels: 16
        max_obstacle_height: 2.0
        mark_threshold: 0
        observation_sources: scan
        scan:
          topic: /scan
          max_obstacle_height: 2.0
          clearing: True
          marking: True
          data_type: "LaserScan"
          raytrace_max_range: 3.0
          raytrace_min_range: 0.0
          obstacle_max_range: 2.5
          obstacle_min_range: 0.0
      static_layer:
        plugin: "nav2_costmap_2d::StaticLayer"
        map_subscribe_transient_local: True
      always_send_full_costmap: True

global_costmap:
  global_costmap:
    ros__parameters:
      update_frequency: 1.0
      publish_frequency: 1.0
      global_frame: map
```

```yaml
      robot_base_frame: base_link
      use_sim_time: False
      robot_radius: 0.22
      resolution: 0.05
      track_unknown_space: true
      plugins: ["static_layer", "obstacle_layer", "inflation_layer"]
      obstacle_layer:
        plugin: "nav2_costmap_2d::ObstacleLayer"
        enabled: True
        observation_sources: scan
        scan:
          topic: /scan
          max_obstacle_height: 2.0
          clearing: True
          marking: True
          data_type: "LaserScan"
          raytrace_max_range: 3.0
          raytrace_min_range: 0.0
          obstacle_max_range: 2.5
          obstacle_min_range: 0.0
      static_layer:
        plugin: "nav2_costmap_2d::StaticLayer"
        map_subscribe_transient_local: True
      inflation_layer:
        plugin: "nav2_costmap_2d::InflationLayer"
        cost_scaling_factor: 3.0
        inflation_radius: 0.55
      always_send_full_costmap: True

map_server:
  ros__parameters:
    use_sim_time: False
    # Overridden in launch by the "map" launch configuration or provided default
value.
    # To use in yaml, remove the default "map" value in the tb3_simulation_launch.py
file & provide full path to map below.
    yaml_filename: ""

map_saver:
  ros__parameters:
    use_sim_time: False
    save_map_timeout: 5.0
    free_thresh_default: 0.25
    occupied_thresh_default: 0.65
    map_subscribe_transient_local: True

planner_server:
  ros__parameters:
    expected_planner_frequency: 20.0
    use_sim_time: False
    planner_plugins: ["GridBased"]
    GridBased:
```

```
      plugin: "nav2_navfn_planner/NavfnPlanner"
      tolerance: 0.5
      use_astar: false
      allow_unknown: true

smoother_server:
  ros__parameters:
    use_sim_time: False
    smoother_plugins: ["simple_smoother"]
    simple_smoother:
      plugin: "nav2_smoother::SimpleSmoother"
      tolerance: 1.0e-10
      max_its: 1000
      do_refinement: False

behavior_server:
  ros__parameters:
    costmap_topic: local_costmap/costmap_raw
    footprint_topic: local_costmap/published_footprint
    cycle_frequency: 10.0
    behavior_plugins: ["spin", "backup", "drive_on_heading", "assisted_teleop",
"wait"]
    spin:
      plugin: "nav2_behaviors/Spin"
    backup:
      plugin: "nav2_behaviors/BackUp"
    drive_on_heading:
      plugin: "nav2_behaviors/DriveOnHeading"
    wait:
      plugin: "nav2_behaviors/Wait"
    assisted_teleop:
      plugin: "nav2_behaviors/AssistedTeleop"
    global_frame: odom
    robot_base_frame: base_link
    transform_tolerance: 0.1
    use_sim_time: False
    simulate_ahead_time: 2.0
    max_rotational_vel: 1.0
    min_rotational_vel: 0.4
    rotational_acc_lim: 3.2

robot_state_publisher:
  ros__parameters:
    use_sim_time: False

waypoint_follower:
  ros__parameters:
    use_sim_time: False
    loop_rate: 20
    stop_on_failure: false
    waypoint_task_executor_plugin: "wait_at_waypoint"
    wait_at_waypoint:
```

```
      plugin: "nav2_waypoint_follower::WaitAtWaypoint"
      enabled: True
      waypoint_pause_duration: 200

velocity_smoother:
  ros__parameters:
    use_sim_time: False
    smoothing_frequency: 20.0
    scale_velocities: False
    feedback: "OPEN_LOOP"
    max_velocity: [0.26, 0.0, 1.0]
    min_velocity: [-0.26, 0.0, -1.0]
    max_accel: [2.5, 0.0, 3.2]
    max_decel: [-2.5, 0.0, -3.2]
    odom_topic: "odom"
    odom_duration: 0.1
    deadband_velocity: [0.0, 0.0, 0.0]
    velocity_timeout: 1.0
```

This parameter table configures the parameters required for each node launched in the navigation launch file.