Robot state estimation

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.

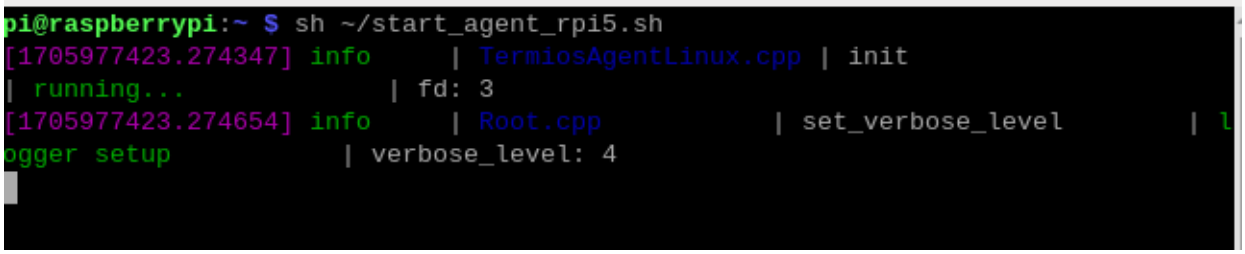
1. Program function description

After the program is started, it will subscribe to imu and odom data, filter out part of the imu data, and then fuse it with the odom data. Finally, a fused odom data will be output to estimate the status of the robot. This data is mostly used in mapping and navigation. .

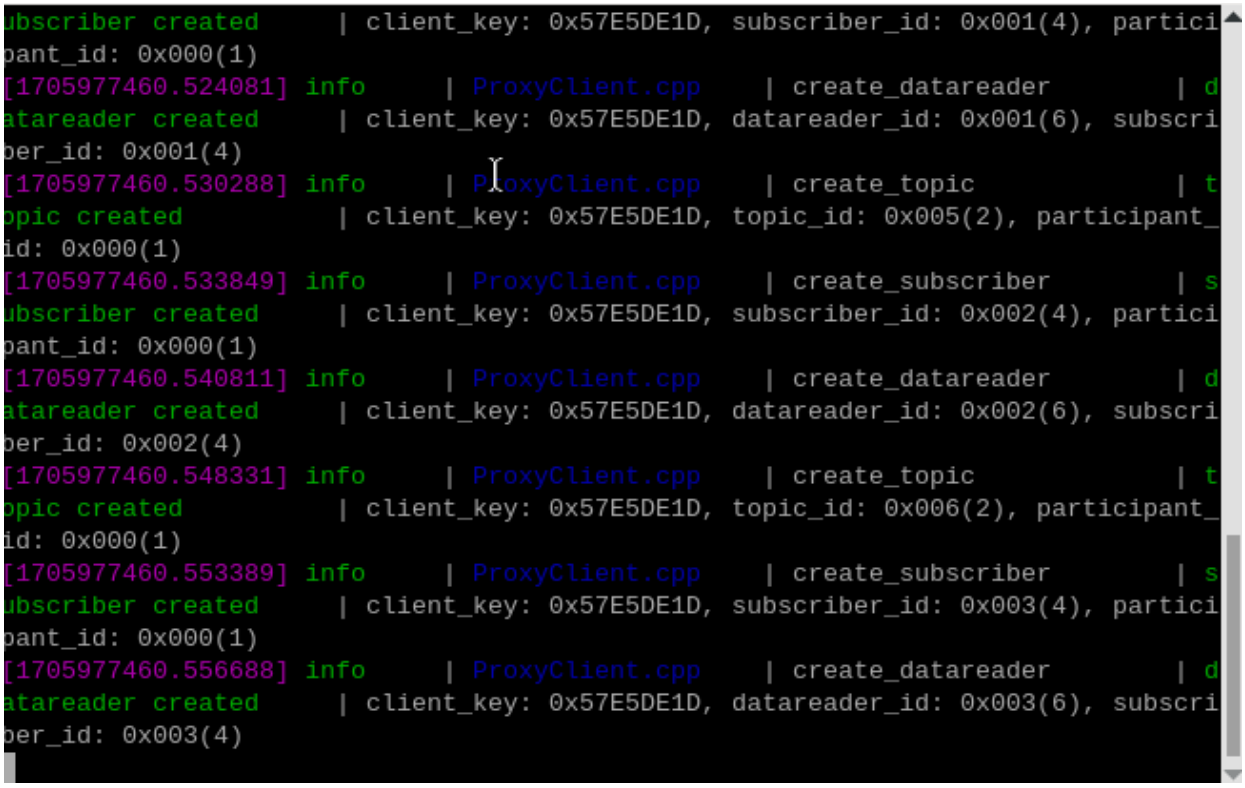
2.Start and connect to the agent

After successfully starting the Raspberry Pi, enter the following command to start the agent.

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



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

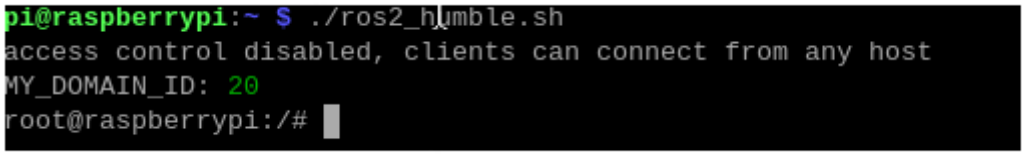


3. Enter the car system docker

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. Start the program

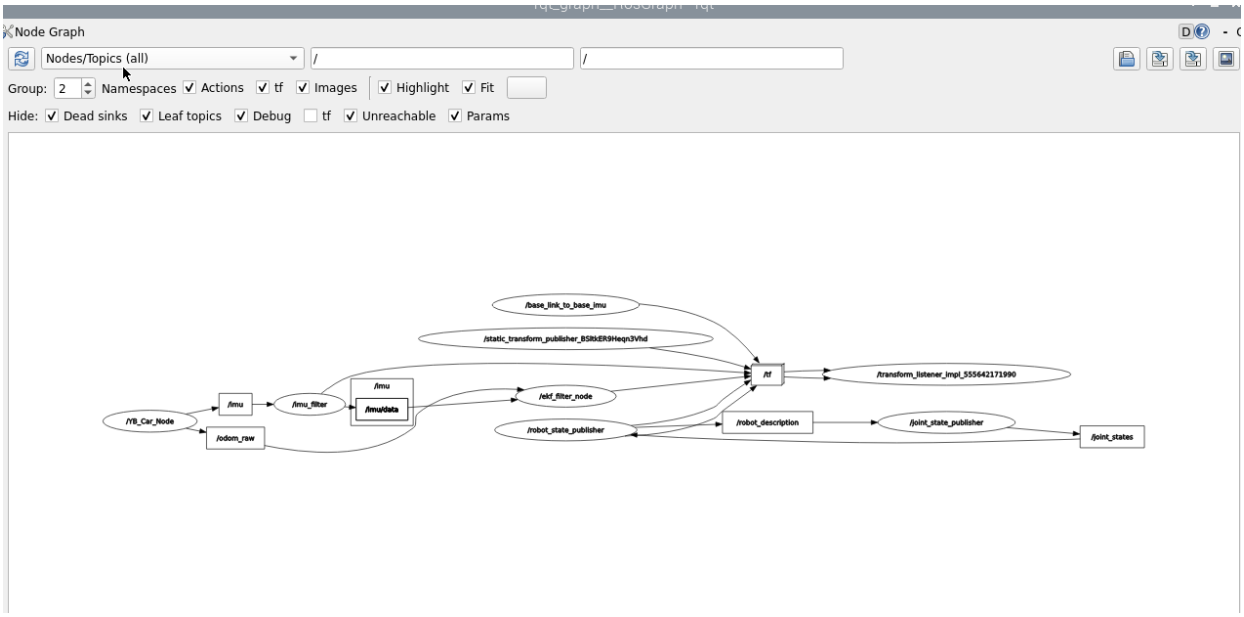
Enter the following command in the terminal.

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

```
----------------------robot_type = x3----------------------
[INFO] [imu_filter_madgwick_node-1]: process started with pid [467]
[INFO] [ekf_node-2]: process started with pid [469]
[INFO] [static_transform_publisher-3]: process started with pid [471]
[INFO] [joint_state_publisher-4]: process started with pid [473]
[INFO] [robot_state_publisher-5]: process started with pid [475]
[INFO] [static_transform_publisher-6]: process started with pid [477]
[static_transform_publisher-6] [WARN] [1705983153.436730358] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-3] [WARN] [1705983153.440907069] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-6] [INFO] [1705983153.493758940] [static_transform_publisher_BSltkER9Heqn3Vhd]: 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'
[static_transform_publisher-3] [INFO] [1705983153.493758791] [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'
[robot_state_publisher-5] [WARN] [1705983153.549719441] [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] [1705983153.550449464] [robot_state_publisher]: got segment base_link
[robot_state_publisher-5] [INFO] [1705983153.550870115] [robot_state_publisher]: got segment imu_Link
[robot_state_publisher-5] [INFO] [1705983153.551059023] [robot_state_publisher]: got segment jq1_Link
[robot_state_publisher-5] [INFO] [1705983153.551073171] [robot_state_publisher]: got segment jq2_Link
[robot_state_publisher-5] [INFO] [1705983153.551084338] [robot_state_publisher]: got segment radar_Link
[robot_state_publisher-5] [INFO] [1705983153.551111375] [robot_state_publisher]: got segment yh_Link
[robot_state_publisher-5] [INFO] [1705983153.551122042] [robot_state_publisher]: got segment yq_Link
[robot_state_publisher-5] [INFO] [1705983153.551132209] [robot_state_publisher]: got segment zh_Link
[robot_state_publisher-5] [INFO] [1705983153.551142320] [robot_state_publisher]: got segment zq_Link
[imu_filter_madgwick_node-1] [INFO] [1705983153.923360093] [imu_filter]: Starting ImuFilter
[imu_filter_madgwick_node-1] [INFO] [1705983153.925493903] [imu_filter]: Using dt computed from message headers
[imu_filter_madgwick_node-1] [INFO] [1705983153.925539088] [imu_filter]: The gravity vector is kept in the IMU message.
[imu_filter_madgwick_node-1] [INFO] [1705983153.926906838] [imu_filter]: Imu filter gain set to 0.100000
[imu_filter_madgwick_node-1] [INFO] [1705983153.926966838] [imu_filter]: Gyro drift bias set to 0.000000
[imu_filter_madgwick_node-1] [INFO] [1705983153.926988672] [imu_filter]: Magnetometer bias values: 0.000000 0.000000 0.000000
[joint_state_publisher-4] [INFO] [1705983154.028776814] [joint_state_publisher]: Waiting for robot_description to be published on the robot_description topic...
[imu_filter_madgwick_node-1] [INFO] [1705983156.027033468] [imu_filter]: First IMU message received.
```

Enter the following command to view the communication diagram between nodes.

```
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.

The fused node is /ekf_filter_node. You can query the relevant information of this node and input it through the terminal.

```
ros2 node info /ekf_filter_node
```

```
root@raspberrypi:~# ros2 node info /ekf_filter_node
/ekf_filter_node
  Subscribers:
    /imu/data: sensor_msgs/msg/Imu
    /odom_raw: nav_msgs/msg/Odometry
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /set_pose: geometry_msgs/msg/PoseWithCovarianceStamped
  Publishers:
    /diagnostics: diagnostic_msgs/msg/DiagnosticArray
    /odom: nav_msgs/msg/Odometry
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /tf: tf2_msgs/msg/TFMessage
  Service Servers:
    /ekf_filter_node/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /ekf_filter_node/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /ekf_filter_node/get_parameters: rcl_interfaces/srv/GetParameters
    /ekf_filter_node/list_parameters: rcl_interfaces/srv/ListParameters
    /ekf_filter_node/set_parameters: rcl_interfaces/srv/SetParameters
    /ekf_filter_node/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
    /enable: std_srvs/srv/Empty
    /reset: std_srvs/srv/Empty
    /set_pose: robot_localization/srv/SetPose
    /toggle: robot_localization/srv/ToggleFilterProcessing
  Service Clients:

  Action Servers:

  Action Clients:
```

Combined with the node communication diagram above, it can be seen that the node subscribes to /imu/data and /odom_raw data, and then publishes a /odom data.

5. Parse launch file

launch file location（Take the supporting virtual machine as an example）：

```
/root/yahboomcar_ws/src/yahboomcar_bringup/launch
```

yahboomcar_bringup_launch.py

```
from ament_index_python.packages import get_package_share_path

from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument
from launch.conditions import IfCondition, UnlessCondition
from launch.substitutions import Command, LaunchConfiguration

from launch_ros.actions import Node
from launch_ros.parameter_descriptions import ParameterValue

import os
from ament_index_python.packages import get_package_share_directory

from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource

print("----------------------robot_type = x3----------------------")
def generate_launch_description():
    imu_filter_config = os.path.join(
        get_package_share_directory('yahboomcar_bringup'),
        'param',
        'imu_filter_param.yaml'
    )

    imu_filter_node = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([os.path.join(
            get_package_share_directory('imu_filter_madgwick'), 'launch'),
            '/imu_filter.launch.py'])
    )

    ekf_node = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([os.path.join(
            get_package_share_directory('robot_localization'), 'launch'),
            '/ekf.launch.py'])
    )

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

    base_link_to_imu_tf_node = Node(
```

```
     package='tf2_ros',
     executable='static_transform_publisher',
     name='base_link_to_base_imu',
     arguments=['-0.002999',
'-0.0030001','0.031701','0','0','0','base_link','imu_frame']
    )

    return LaunchDescription([
        imu_filter_node,
        ekf_node,
        base_link_to_imu_tf_node,
        description_launch
    ])
```

The launch file starts the following nodes.

- imu_filter_node：Filter imu data nodes, mainly filter some imu data.
- ekf_node：Fusion node, mainly fuses odom data and filtered imu data.
- base_link_to_imu_tf_node：Publish a static change, mainly publishing the pose transformation of the imu module and the car
- description_launch：Load the URDF model.