4. Robot state estimation

4.1. Program function description

After the program runs, it combines the imu data and speed vel data read from the ROS expansion board to output an odom data that integrates imu and odom data. This data can be used when performing positioning functions.

4.2, Program code reference path

After SSH connects to the car, the location of the function source code is,

```
/home/sunrise/yahboomcar_ws/src/yahboomcar_bringup/launch/yahboomcar_bringup_laun
ch.py
```

ekf fusion program code reference path,

```
/home/sunrise/yahboomcar_ws/src/robot_localization/launch/ekf.launch.py
```

4.3, Program startup

4.3.1, Start command

After SSH connects to the car, enter in the terminal,

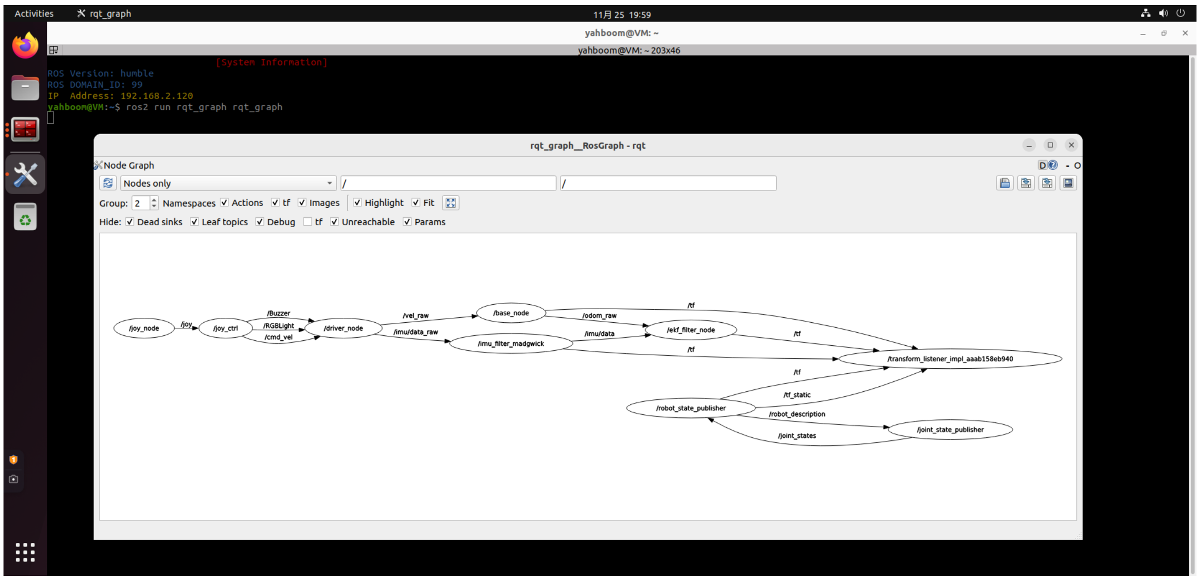
```
ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py
```

4.3.2, View the node communication diagram

Open the virtual machine terminal and enter,

```
ros2 run rqt_graph rqt_graph
```

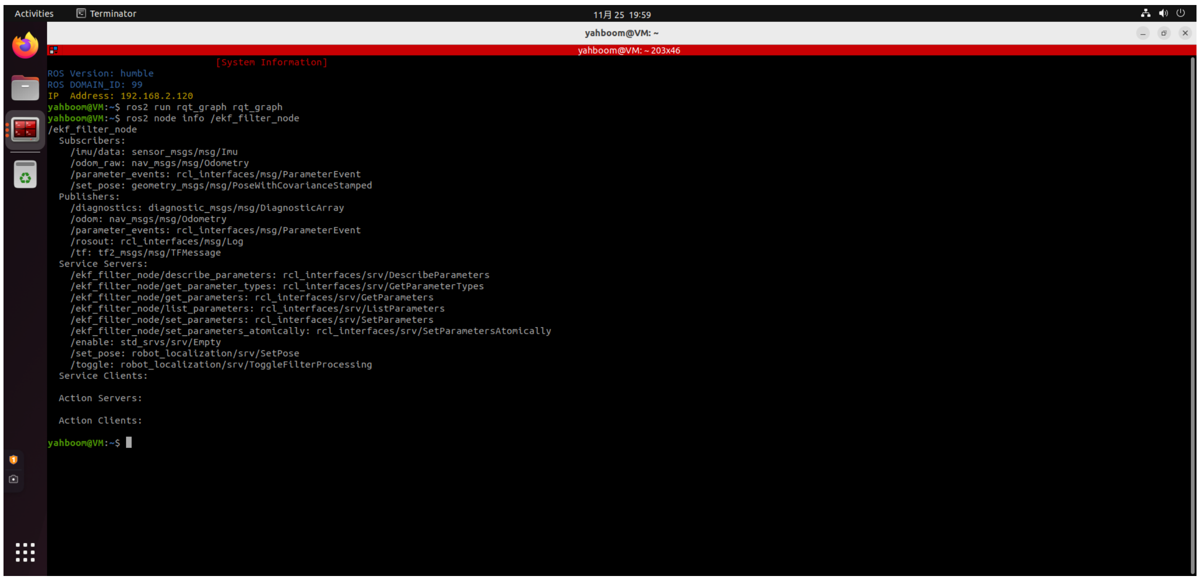
[Note] Since Xuripi does not provide a graphical desktop environment, all ROS rqt-related operations must be implemented on the PC side (virtual machine side). That is, as long as the communication between the PC side and the car side is normal, you can open the terminal without SSH connection to the car and directly run the relevant instructions.



```
                                    [System Information]
ROS Version: humble
ROS DOMAIN_ID: 99
IP  Address: 192.168.2.120
yahboom@VM:~$ ros2 run rqt_graph rqt_graph
```

Mainly look at the node input and output in the red box in the above figure. It can be seen that /ekf_filter_node receives odom_raw data and imu_data data for fusion, and finally outputs and publishes an odom data. We can view it through the ros2 node tool, terminal input,

```
ros2 node info /ekf_filter_node
```



```
                                    [System Information]
ROS Version: humble
ROS DOMAIN_ID: 99
IP  Address: 192.168.2.120
yahboom@VM:~$ ros2 run rqt_graph rqt_graph
yahboom@VM:~$ 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
    /set_pose: robot_localization/srv/SetPose
    /toggle: robot_localization/srv/ToggleFilterProcessing
  Service Clients:

  Action Servers:

  Action Clients:

yahboom@VM:~$
```

4.4, launch file analysis

Let's take a look at the main related nodes of the launch file,

- /driver_node: Start the car chassis, obtain the wheel speed vel data, publish it to the /base_node node, obtain imu data, and publish it to the /imu_filter_node node;
- /base_node: Receive vel data, convert it into odom_raw data through calculation, and publish it to the /ekf_node node;
- /imu_filter_node: Receive the imu data published by the chassis, filter it through its own algorithm, and publish the filtered imu/data data to the /ekf_node node;
- /ekf_node: Receive the odom data published by the /base_node node and the imu/data data published by the /imu_filter_node, merge them through its own algorithm, and publish the odom data.