4. Robot state estimation

1. Program Function Description

After the program runs, it combines the imu data and speedvel data read from the ROS expansion board, and outputs an odom data which combines the imu and odom data, and this data is applied when doing the positioning function.

2. Program Code Reference Path

After entering the docker container, the source code of this function is located at.

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_bringup/launch/yahboomcar_bringup_R2_launch.py
```

ekf fusion program code reference path.

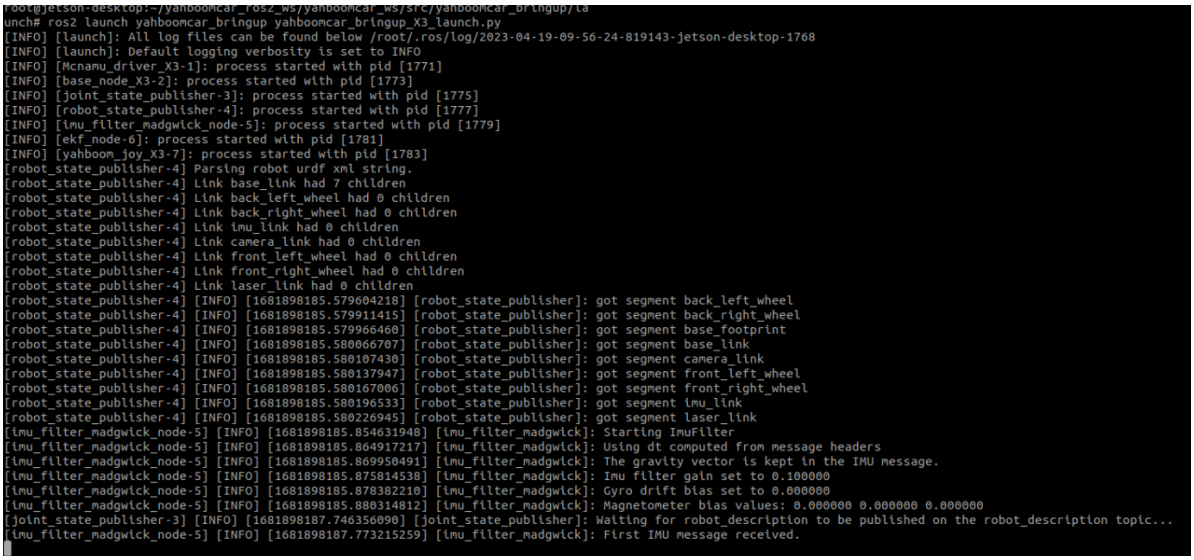
```
/root/yahboomcar_ros2_ws/software/library_ws/src/robot_localization/launch/ekf.launch.py
```

3. Program startup

3.1. Startup commands

After entering the docker container, depending on the actual model, terminal input, the

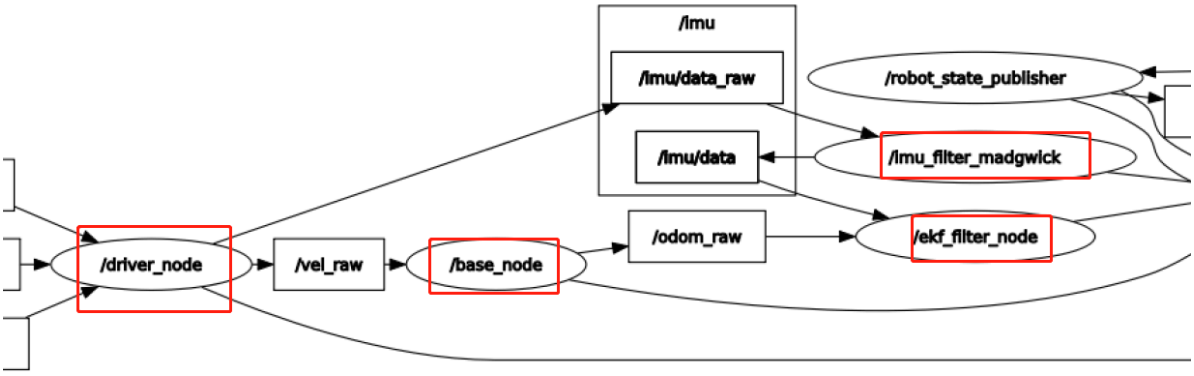
```
ros2 launch yahboomcar_bringup yahboomcar_bringup_R2_launch.py
```



3.2.Viewing the node communication graph

Open the terminal and enter the container and type,

```
ros2 run rqt_graph rqt_graph
```



The main thing is to look at the node inputs and outputs in the red box of the above figure, as you can see, /ekf_filter_node receives the odom_raw data and the imu_data data for fusion, and finally outputs the release of an odom data, which we can look at through the ros2 node tool with the terminal inputs,

```
ros2 node info /ekf_filter_node
```

```
root@jetson-desktop:~# ros2 node info /ekf_filter_node
/ekf_filter_node
  Subscribers:
    /example/odom2: nav_msgs/msg/Odometry
    /example/pose: geometry_msgs/msg/PoseWithCovarianceStamped
    /example/twist: geometry_msgs/msg/TwistWithCovarianceStamped
    /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:
```

4. Parsing the launch file

Let's look at the main relevant nodes of the launch file.

- /driver_node: start the chassis of the cart, get the velocity vel data of the wheel, publish it to /base_node node, get the imu data, publish it to /Imu_filter_madgwick node; /base_node: receive the vel data, through calculation, convert it to odom_raw data, publish it to /Imu_filter_madgwick node.
- /base_node: receive vel data, through calculation, convert to odom_raw data, publish to /ekf_filter_node node;

- /Imu_filter_madgwick: receive imu data posted by chassis, filter it through its own algorithm, and post the filtered imu/data data to /ekf_filter_node node; /ekf_filter_node.
- /ekf_filter_node: receive odom data released by /base_node node and imu/data data released by /Imu_filter_madgwick, through its own algorithm, after fusion, release odom data.