Multi-vehicle Chassis Control

1. Program Description

This function allows you to control multiple cars simultaneously using the keyboard.

1.1. Functional Requirements

Using two cars as an example, these two cars must meet the following three requirements:

- Both cars must be on the same local area network and connected to the same Wi-Fi network.
- Both cars must have the same ros_domain_id. This can be achieved by modifying the ROS_DOMAIN_ID value in the terminal running the ROS environment. The default value is 61. The modification is based on the board:
  - For Raspberry Pi and Jetson-Nano boards: Enter Docker, modify the ROS_DOMAIN_ID value in the /root/.bashrc file, save and exit, and then enter `source ~/.bashrc` in the terminal to refresh the environment variables. The modified [MY_DOMAIN_ID] should appear in the terminal.
  - For RDKX5 and Orin: Open a terminal and modify the value of ROS_DOMAIN_ID in the ~/.bashrc file. Save and exit. Enter the command `source ~/.bashrc` in the terminal to refresh the environment variables. The modified [MY_DOMAIN_ID] should appear in the terminal.

2. Program Reference Path

For Raspberry Pi and Jetson-Nano boards, you need to enter the Docker container first. For RDKX5 and Orin controllers, this is not necessary.

The source code for this function is located at

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_multi/launch/A1_bringup_multi_ctrl_launch.xml
```

3. Program Startup

3.1. Startup Command

For Raspberry Pi and Jetson-Nano boards, you need to enter the Docker container first. For RDKX5 and Orin controllers, this is not necessary.

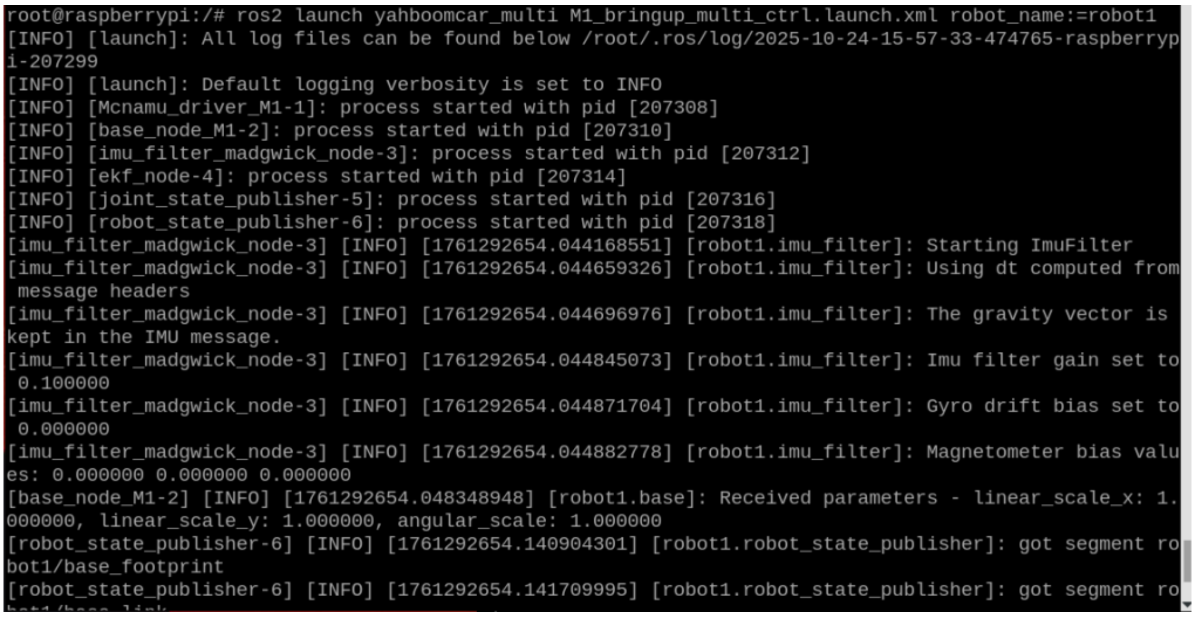
Enter the Docker container (see [Docker course] --- [4. Docker Startup Script] for steps).

All the following commands must be executed within the same Docker container **(see [Docker course] --- [3. Docker Submission and Multi-Terminal Access] for steps).**

Depending on the vehicle model, enter the command in each terminal separately.

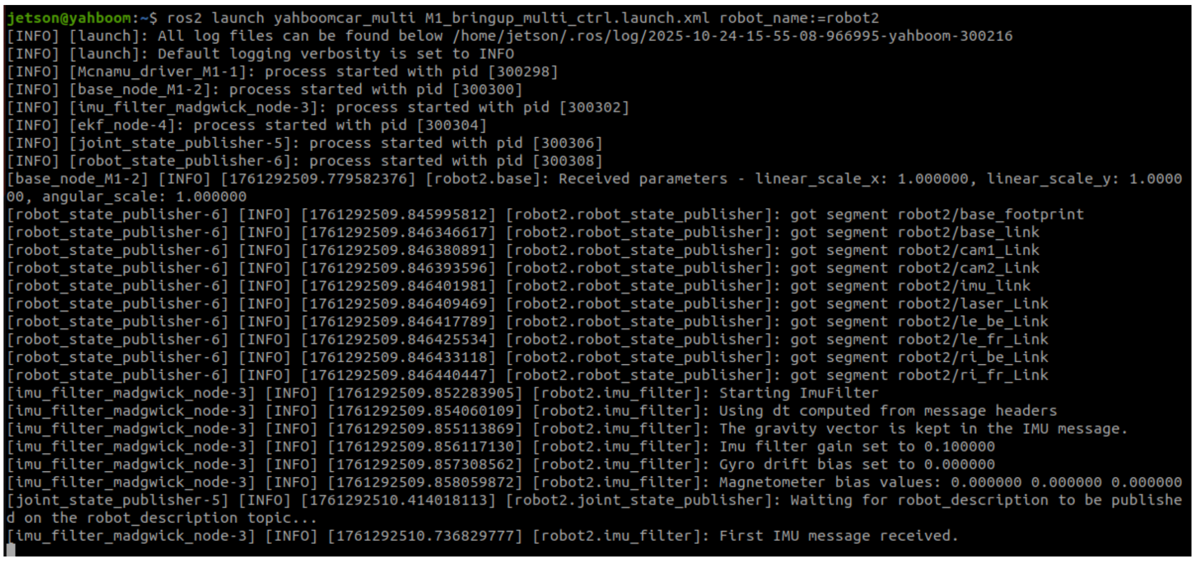
```
#car1
ros2 launch yahboomcar_multi M1_bringup_multi_ctrl.launch.xml robot_name:=robot1

#Car 2
ros2 launch yahboomcar_multi M1_bringup_multi_ctrl.launch.xml robot_name:=robot2
```

Robot 1:



```
root@raspberrypi:/# ros2 launch yahboomcar_multi M1_bringup_multi_ctrl.launch.xml robot_name:=robot1
[INFO] [launch]: All log files can be found below /root/.ros/log/2025-10-24-15-57-33-474765-raspberryp
i-207299
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [Mcnamu_driver_M1-1]: process started with pid [207308]
[INFO] [base_node_M1-2]: process started with pid [207310]
[INFO] [imu_filter_madgwick_node-3]: process started with pid [207312]
[INFO] [ekf_node-4]: process started with pid [207314]
[INFO] [joint_state_publisher-5]: process started with pid [207316]
[INFO] [robot_state_publisher-6]: process started with pid [207318]
[imu_filter_madgwick_node-3] [INFO] [1761292654.044168551] [robot1.imu_filter]: Starting ImuFilter
[imu_filter_madgwick_node-3] [INFO] [1761292654.044659326] [robot1.imu_filter]: Using dt computed from
 message headers
[imu_filter_madgwick_node-3] [INFO] [1761292654.044696976] [robot1.imu_filter]: The gravity vector is
kept in the IMU message.
[imu_filter_madgwick_node-3] [INFO] [1761292654.044845073] [robot1.imu_filter]: Imu filter gain set to
 0.100000
[imu_filter_madgwick_node-3] [INFO] [1761292654.044871704] [robot1.imu_filter]: Gyro drift bias set to
 0.000000
[imu_filter_madgwick_node-3] [INFO] [1761292654.044882778] [robot1.imu_filter]: Magnetometer bias valu
es: 0.000000 0.000000 0.000000
[base_node_M1-2] [INFO] [1761292654.048348948] [robot1.base]: Received parameters - linear_scale_x: 1.
000000, linear_scale_y: 1.000000, angular_scale: 1.000000
[robot_state_publisher-6] [INFO] [1761292654.140904301] [robot1.robot_state_publisher]: got segment ro
bot1/base_footprint
[robot_state_publisher-6] [INFO] [1761292654.141709995] [robot1.robot_state_publisher]: got segment ro
```

Robot 2:

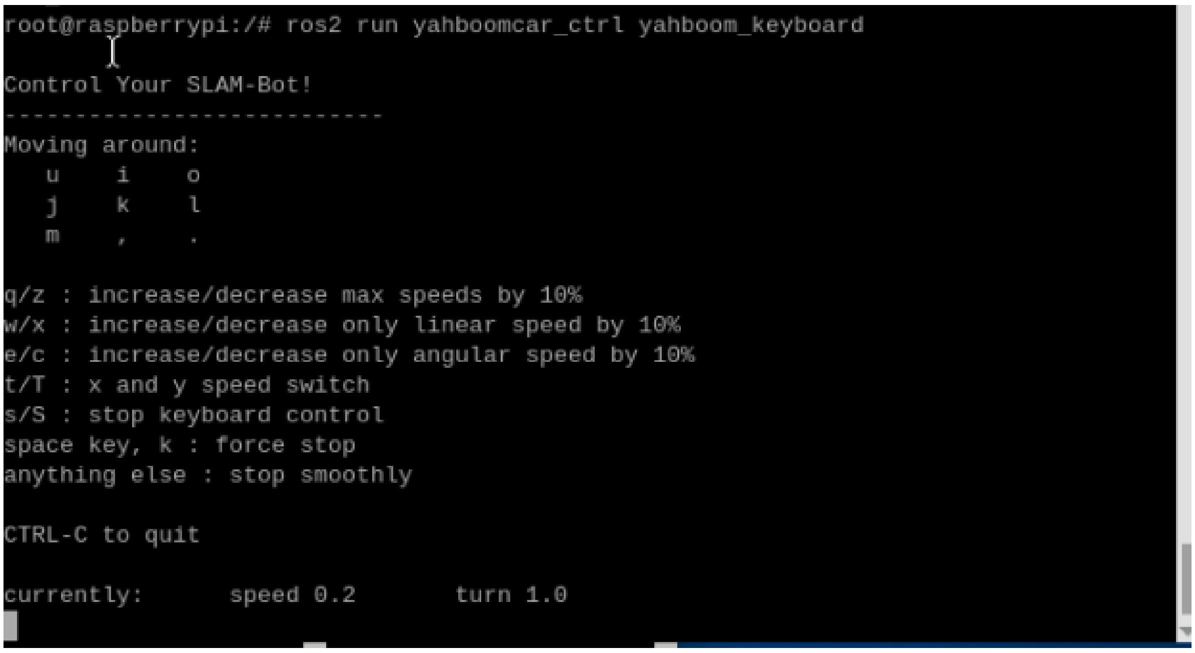


```
jetson@yahboom:~$ ros2 launch yahboomcar_multi M1_bringup_multi_ctrl.launch.xml robot_name:=robot2
[INFO] [launch]: All log files can be found below /home/jetson/.ros/log/2025-10-24-15-55-08-966995-yahboom-300216
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [Mcnamu_driver_M1-1]: process started with pid [300298]
[INFO] [base_node_M1-2]: process started with pid [300300]
[INFO] [imu_filter_madgwick_node-3]: process started with pid [300302]
[INFO] [ekf_node-4]: process started with pid [300304]
[INFO] [joint_state_publisher-5]: process started with pid [300306]
[INFO] [robot_state_publisher-6]: process started with pid [300308]
[base_node_M1-2] [INFO] [1761292509.779582376] [robot2.base]: Received parameters - linear_scale_x: 1.000000, linear_scale_y: 1.0000
00, angular_scale: 1.000000
[robot_state_publisher-6] [INFO] [1761292509.845995812] [robot2.robot_state_publisher]: got segment robot2/base_footprint
[robot_state_publisher-6] [INFO] [1761292509.846346617] [robot2.robot_state_publisher]: got segment robot2/base_link
[robot_state_publisher-6] [INFO] [1761292509.846380891] [robot2.robot_state_publisher]: got segment robot2/cam1_Link
[robot_state_publisher-6] [INFO] [1761292509.846393596] [robot2.robot_state_publisher]: got segment robot2/cam2_Link
[robot_state_publisher-6] [INFO] [1761292509.846401981] [robot2.robot_state_publisher]: got segment robot2/imu_link
[robot_state_publisher-6] [INFO] [1761292509.846409469] [robot2.robot_state_publisher]: got segment robot2/laser_Link
[robot_state_publisher-6] [INFO] [1761292509.846417789] [robot2.robot_state_publisher]: got segment robot2/le_be_Link
[robot_state_publisher-6] [INFO] [1761292509.846425534] [robot2.robot_state_publisher]: got segment robot2/le_fr_Link
[robot_state_publisher-6] [INFO] [1761292509.846433118] [robot2.robot_state_publisher]: got segment robot2/ri_be_Link
[robot_state_publisher-6] [INFO] [1761292509.846440447] [robot2.robot_state_publisher]: got segment robot2/ri_fr_Link
[imu_filter_madgwick_node-3] [INFO] [1761292509.852283905] [robot2.imu_filter]: Starting ImuFilter
[imu_filter_madgwick_node-3] [INFO] [1761292509.854060109] [robot2.imu_filter]: Using dt computed from message headers
[imu_filter_madgwick_node-3] [INFO] [1761292509.855113869] [robot2.imu_filter]: The gravity vector is kept in the IMU message.
[imu_filter_madgwick_node-3] [INFO] [1761292509.856117130] [robot2.imu_filter]: Imu filter gain set to 0.100000
[imu_filter_madgwick_node-3] [INFO] [1761292509.857308562] [robot2.imu_filter]: Gyro drift bias set to 0.000000
[imu_filter_madgwick_node-3] [INFO] [1761292509.858059872] [robot2.imu_filter]: Magnetometer bias values: 0.000000 0.000000 0.000000
[joint_state_publisher-5] [INFO] [1761292510.414018113] [robot2.joint_state_publisher]: Waiting for robot_description to be publishe
d on the robot_description topic...
[imu_filter_madgwick_node-3] [INFO] [1761292510.736829777] [robot2.imu_filter]: First IMU message received.
```

[robot_name] represents the serial number of the vehicle to be launched. Currently, the program can launch either robot 1 or robot 2.

Enter the following command in a terminal on any car to enable keyboard and joystick control.

```
#Keyboard
ros2 run yahboomcar_ctrl yahboom_keyboard
# Joystick
ros2 launch yahboomcar_multi M1_joy_ctrl.launch.py
```

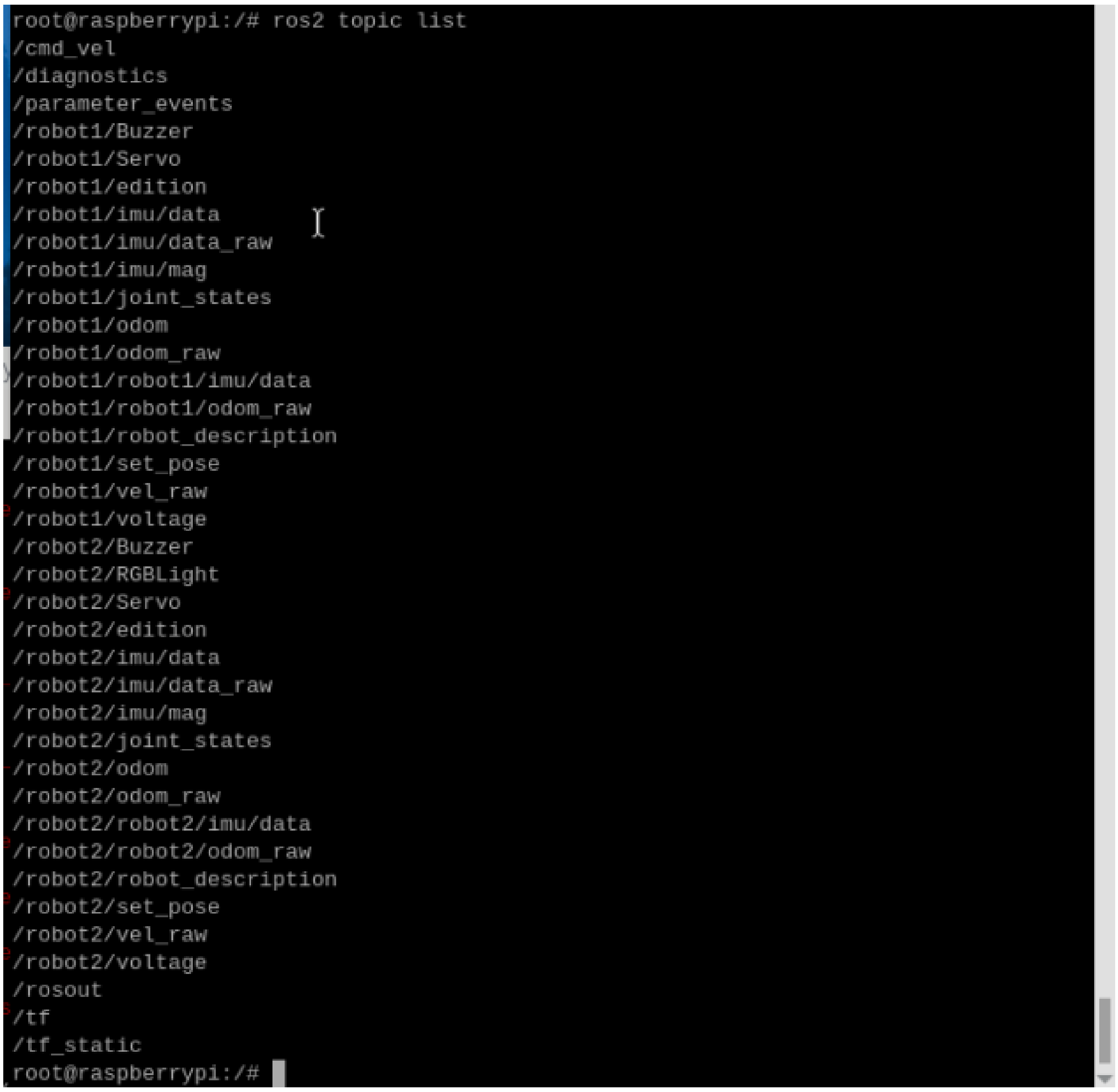


```
root@raspberrypi:/# ros2 run yahboomcar_ctrl yahboom_keyboard

Control Your SLAM-Bot!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
t/T : x and y speed switch
s/S : stop keyboard control
space key, k : force stop
anything else : stop smoothly

CTRL-C to quit

currently:      speed 0.2       turn 1.0
```

3.2. View Topic Nodes

Enter the following command in the terminal:

```
ros2 topic list
```

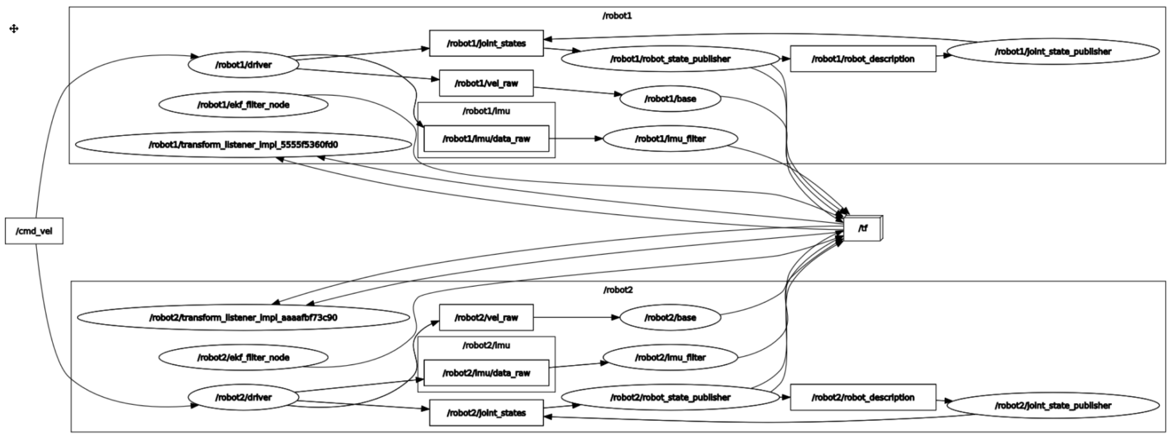


```
root@raspberrypi:/# ros2 topic list
/cmd_vel
/diagnostics
/parameter_events
/robot1/Buzzer
/robot1/Servo
/robot1/edition
/robot1/imu/data
/robot1/imu/data_raw
/robot1/imu/mag
/robot1/joint_states
/robot1/odom
/robot1/odom_raw
/robot1/robot1/imu/data
/robot1/robot1/odom_raw
/robot1/robot_description
/robot1/set_pose
/robot1/vel_raw
/robot1/voltage
/robot2/Buzzer
/robot2/RGBLight
/robot2/Servo
/robot2/edition
/robot2/imu/data
/robot2/imu/data_raw
/robot2/imu/mag
/robot2/joint_states
/robot2/odom
/robot2/odom_raw
/robot2/robot2/imu/data
/robot2/robot2/odom_raw
/robot2/robot_description
/robot2/set_pose
/robot2/vel_raw
/robot2/voltage
/rosout
/tf
/tf_static
root@raspberrypi:/#
```

3.3. View the Node Communication Graph

Enter the following command in the terminal:

```
ros2 run rqt_graph rqt_graph
```



As you can see, the keyboard/controller control node publishes the speed topic /cmd_vel. Both robot1 and robot2 chassis subscribe to this topic. Therefore, when the topic data is published, both robots receive it and control their respective chassis movements.

4. Core Code Analysis

The code on the virtual machine side is the same as the controller control code on the robot side, so I won't go into detail here. Let's focus on the content on the robot side and see how it starts.

The launch file is M1_bringup_multi_ctrl.launch.xml, and the path is as follows: ,

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_multi/launch/A1_bringup_multi_
ctrl.launch.xml
```

```
<launch>
    <arg name="robot_name" default="robot1"/>
    <group>
        <push-ros-namespace namespace="$(var robot_name)"/>
        <!--driver_node-->
        <node name="driver" pkg="yahboomcar_bringup" exec="Mcnamu_driver_M1"
output="screen">
            <param name="imu_link" value="$(var robot_name)/imu_link"/>
            <remap from="cmd_vel" to="/cmd_vel"/>
            <remap from="Servo" to="/Servo"/>
        </node>
        <!--base_node-->
        <node name="base" pkg="yahboomcar_base_node" exec="base_node_M1"
output="screen">
            <param name="odom_frame" value="$(var robot_name)/odom"/>
            <param name="base_footprint_frame" value="$(var
robot_name)/base_footprint"/>
            <param name="pub_odom_tf" value="false"/>
        </node>
        <!--imu_filter_node-->
        <node name="imu_filter" pkg="imu_filter_madgwick"
exec="imu_filter_madgwick_node" output="screen">
            <param name="fixed_frame" value="$(var robot_name)/base_link"/>
            <param name="use_mag" value="false"/>
            <param name="publish_tf" value="false"/>
            <param name="world_frame" value="enu"/>
            <param name="orientation_stddev" value="0.05"/>
        </node>
        <!--ekf_node-->
        <node name="ekf_filter_node" pkg="robot_localization" exec="ekf_node">
            <param name="odom_frame" value="$(var robot_name)/odom"/>
            <param name="base_link_frame" value="$(var
robot_name)/base_footprint"/>
            <param name="world_frame" value="$(var robot_name)/odom"/>
            <param from="$(find-pkg-share yahboomcar_multi)/param/ekf_$(var
robot_name).yaml"/>
            <remap from="odometry/filtered" to="odom"/>
            <remap from="/odom_raw" to="odom_raw"/>
        </node>
    </group>
    <include file="$(find-pkg-share
yahboomcar_description)/launch/description_M1_multi_$(var
robot_name).launch.py"/>
</launch>
```

The XML format is used here to write the launch file, which makes it convenient for us to add namespaces in front of multiple nodes. Adding a namespace is to resolve conflicts caused by the same node name. We have two cars here. Take the chassis program as an example. If the two chassis nodes are both named driver, then after establishing multi-machine communication, this is not allowed, so we add the namespace **namespace** in front of the node name. In this way, the chassis node name of car 1 is /robot1/driver, and the chassis node program of car 2 is /robot2/driver. In the launch file in XML format, a group group is used to stipulate that within this group, both the node name and the topic name need to be added with **namespace**.

```
<group>
    <push-ros-namespace namespace="$(var robot_name)"/>
</group>
```

The parameter defined in the XML launch file is **$(var robot_name)**. The parameter passed in here is robot_name. When entering the command, you can enter it in the command line.

```
ros2 launch yahboomcar_multi M1_bringup_multi_ctrl.launch.xml robot_name:=robot1
```

One thing to note here is that after adding the namespace, we remapped the topic name /robot1/cmd_vel to /cmd_vel. This is to enable receiving topic messages sent by the virtual machine.

```
<node name="driver" pkg="yahboomcar_bringup" exec="Mcnamu_driver_M1"
output="screen">
    <param name="imu_link" value="$(var robot_name)/imu_link"/>
    <remap from="cmd_vel" to="/cmd_vel"/>
    <remap from="Servo" to="/Servo"/>
</node>
```

Regarding the input of parameters, adding namespaces needs to be described in the launch file, such as the following parameters,

```
<node name="base" pkg="yahboomcar_base_node" exec="base_node_M1"
output="screen">
    <param name="odom_frame" value="$(var robot_name)/odom"/>
    <param name="base_footprint_frame" value="$(var
robot_name)/base_footprint"/>
    <param name="pub_odom_tf" value="false"/>
</node>
```

It can be seen that the **odom_frame** parameter is assigned to **$(var robot_name)/odom**, and the **base_footprint_frame** parameter is assigned to **$(var robot_name)/base_footprint**.

If some parameters do not require a namespace, they can be passed in as a parameter list. For example,

```
<param from="$(find-pkg-share yahboomcar_multi)/param/M1_ekf_$(var
robot_name).yaml"/>
```

What needs to be noted about this parameter table is that the namespace needs to be added to be able to find it accurately. If we start robot1, then the node started becomes robot1/ekf_filter_node, so the beginning of the parameter file should be,

```
### ekf config file ###
robot1/ekf_filter_node:
    ros__parameters:
```