RealMan Robotic Arm ROS2 User Manual V1.0

RealMan Intelligent Technology (Beijing) Co., Ltd.

Revision History:

No.	Date	Comment
V1.0	11/27/2023	Draft

Content

1. Package Overview

This document introduces the overall robotic arm based on the ROS2 package, and the main role is to help you achieve two goals.

1. Understand the ROS2 package function.
2. Master the current ROS package use.

Source code address: https://github.com/RealManRobot/ros2_rm_robot.git。

The following is the overall introduction of the package.

1. Installation and environment configuration (rm_install)
2. Hardware driver (rm_driver)
3. Launch (rm_bringup)
4. Model description (rm_description)
5. ROS message interface (rm_ros_interfaces)
6. Moveit2 configuration (rm_moveit_config)
7. Moveit2 and hardware driver communication connection (rm_config)
8. Gazebo simulation robotic control (rm_gazebo)
9. Use examples (rm_examples)
10. Technical documentation (rm_docs)

2. rm_driver Package Introduction

2.1 rm_driver package function

(1) Establish a connection with the robot through the API function. Robot default IP: 192.168.1.18. Please ensure that the IP of the host computer is in the same local network. Use ROS2 to control the robot. Make sure that the robot is in Ethernet port communication mode;

(2) The underlying driver of the robotic arm, subscribe and publish the topic data, and update the joint angle of the robot arm in RVIZ.

2.2 rm_driver package use

When using this node, enter the following commands into the end.

```
ros2 launch rm_driver rm_<arm_type>_driver.launch.py
```

In practice, the above <arm_type> needs to be replaced by the actual model of the robotic arm. The available models of the robotic arm are 65, 63, eco65, and 75.

For example, if the used robot arm is 65 series, then start the node with the following commands.

```
ros2 launch rm_driver rm_65_driver.launch.py
```

For a more detailed introduction to the package, please refer to the detailed description document of the package.

3. rm_description Package Introduction

3.1 rm_description package introduction

(1) RM65-B, RM63-B, RMeco65-B, and RM75-B robot functionality description, which provides robot models and configuration files.

(2) Provide the planned TF transformation through rm_65.urdf, rml_63.urdf, rm_eco65.urdf, rm_75.urdf robot model files.

3.2 rm_description package use

When using this node, enter the following commands into the end.

```
ros2 launch rm_description rm_<arm_type>_display.launch.py
```

In practice, the above <arm_type> needs to be replaced by the actual model of the robotic arm. The available models of the robotic arm are 65, 63, eco65, and 75.

For example, if the used robot arm is 65 series, then start the node with the following commands.

```
ros2 launch rm_description rm_65_display.launch.py
```

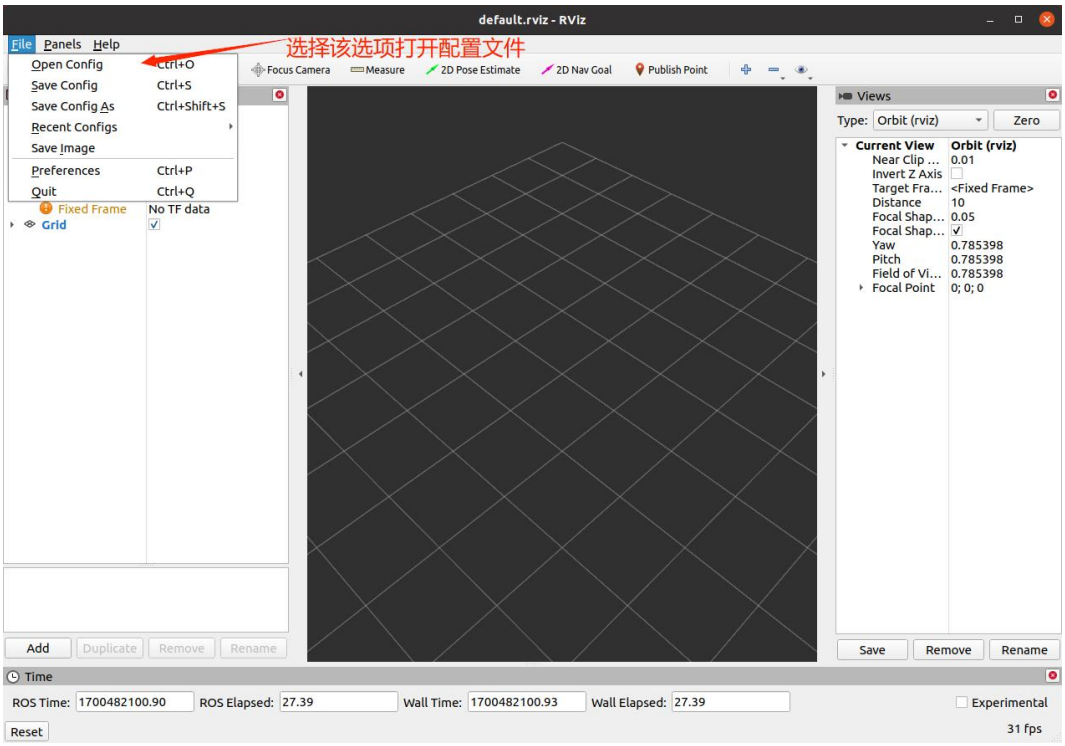
After a successful launch, continue to launch the rm_driver node.

```
ros2 launch rm_driver rm_<arm_type>_driver.launch.py
```

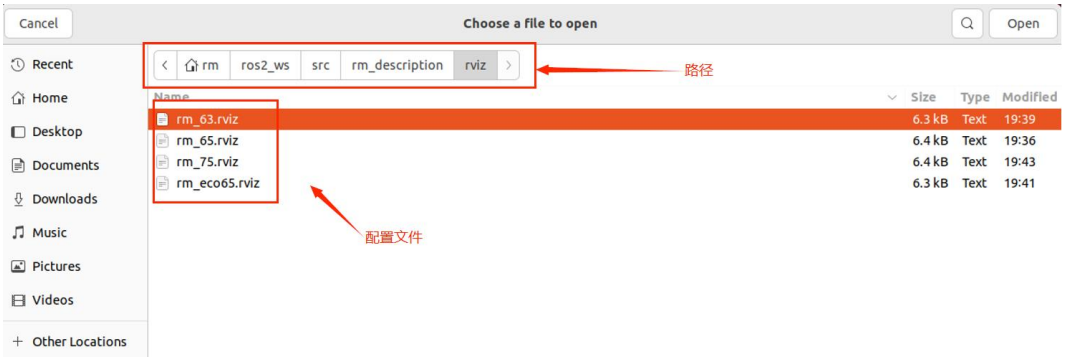
Then launch the rviz2 node.

```
rviz2
```

You can call the rviz configuration file designed before to load the model directly.



Find the corresponding configuration file under the rviz folder of the package.



We can view the robotic arm model through the rviz2 interface, and its position and posture are identical to those of the actual robotic arm.

4. rm_ros_interfaces Package Introduction

4.1. rm_ros_interfaces package function

(1) All the control and state messages used by RM65, RM63, RMECO65, and RM75.

4.2. rm_ros_interfaces package use

This package mainly provides the message files needed for the launch and running of other packages and does not execute any commands.

5. rm_control Package Introduction

5.1. rm_control package function

(1) Provide a docking action with moveit2, as well as a communication bridge for moveit2 to control the real robotic arm.

(2) The robot controller subdivides the robotic arm trajectory planned by Moveit through cubic spline interpolation and sends it to the rm_driver node according to the 20 ms control cycle.

5.2. rm_control package use

When using this node, enter the following commands into the end.

```
ros2 launch rm_control rm_<arm_type>_control.launch.py
```

The node itself does not implement specific visible functions and needs to be used with the relevant functions of the rm_moveit2_config package.

6. rm_moveit2_config File Introduction

6.1 rm_moveit2_config file function

(1) Use the Setup Assistant tool to create a MoveIt2 configuration package according to the robot URDF model rm_65.urdf. The package contains the most required configuration files and launch files required for the launch of MoveIt2, as well as a simple demo.

(2) Rewrite the demo according to the characteristics of the robotic arm, and realize the control function of the real robotic arm with rm_control and rm_driver packages.

6.2 rm_moveit2_config file use

This package can realize the planning and control demonstration of the virtual robotic arm, which can be realized through the following commands.

```
ros2 launch rm_<arm_type>_config demo.launch.py
```

In practice, the above <arm_type> needs to be replaced by the actual model of the robotic arm. The available models of the robotic arm are 65, 63, eco65, and 75.

For example, if the used robot arm is 65 series, then start the node with the following commands.

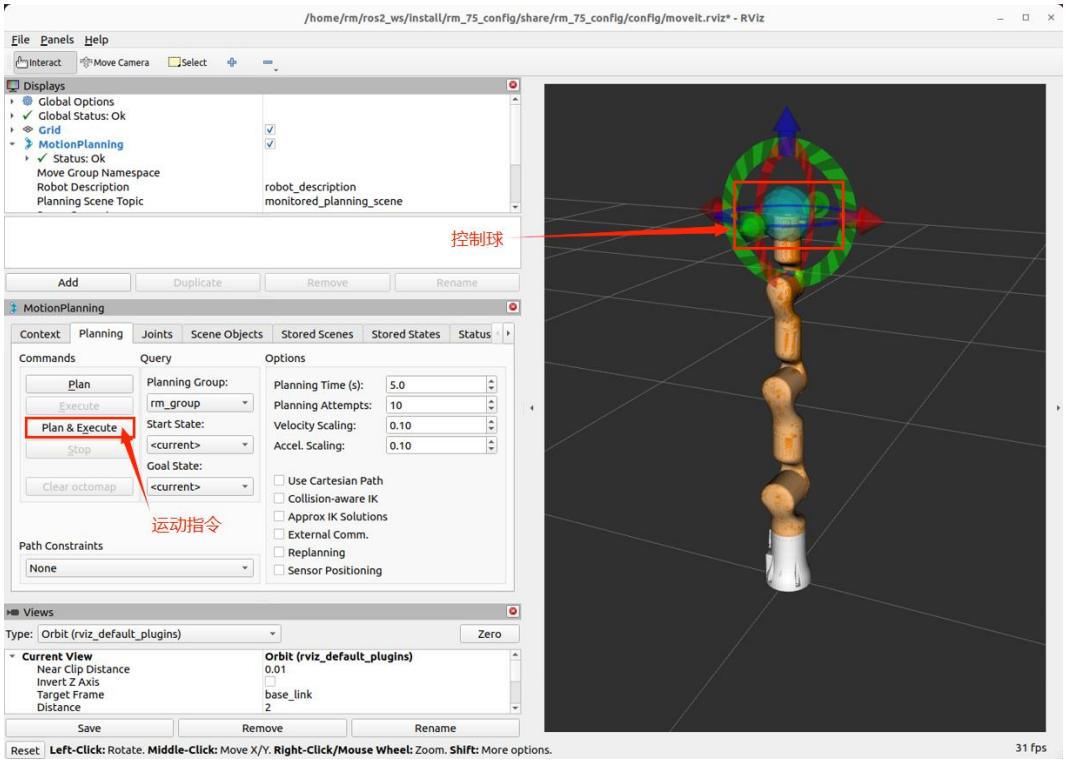
```
ros2 launch rm_65_config demo.launch.py
```

After a successful launch, the robotic arm can be controlled by dragging.

This package can realize the planning and control demonstration of the real robotic arm, which can be realized through the following commands.

```
ros2 launch rm_driver rm_<arm_type>_driver.launch.py
ros2 launch rm_description rm_<arm_type>_display.launch.py
ros2 launch rm_control rm_<arm_type>_control.launch.py
ros2 launch rm_<arm_type>_config real_moveit_demo.launch.py
```

The message related to the moveit2 planning of the robotic arm appears in rviz2, and we can control the real robotic arm by dragging the control ball.



7. rm_bringup Package Introduction

7.1 rm_bringup package function

(1) Realize the integration of complex launch commands of the robotic arm, which can be realized through a launch file to achieve the complex function of multi-node coordination.

7.2 rm_bringup package use

When using this node, enter the following commands, in the end, to directly launch moveit2 to realize the control of moveit2 on the real robotic arm.

```
ros2 launch rm_bringup rm_<arm_type>_bringup.launch.py
```

In practice, the above <arm_type> needs to be replaced by the actual model of the robotic arm. The available models of the robotic arm are 65, 63, eco65, and 75.

For example, if the used robot arm is 65 series, then start the node with the following commands.

```
ros2 launch rm_bringup rm_65_bringup.launch.py
```

When using this node, enter the following commands in the end to directly launch the gazebo to realize the control of moveit2 on the simulation robotic arm.

```
ros2 launch rm_bringup rm_<arm_type>_gazebo.launch.py
```

In practice, the above <arm_type> needs to be replaced by the actual model of the robotic arm. The available models of the robotic arm are 65, 63, eco65, and 75.

For example, if the used robot arm is 65 series, then start the node with the following commands.

```
ros2 launch    rm_bringup rm_65_gazebo.launch.py
```

For a more detailed introduction to this package, please refer to ROS2 Robotic Arm rm_bringup Package Detailed Description.

8. rm_gazebo Package Introduction

8.1. rm_gazebo package function

(1) Realize the control of the simulation robotic arm. It can be used when there is no robot arm entity and you want to use Moveit2 to plan the robotic arm and view the robotic arm's motion trajectory.

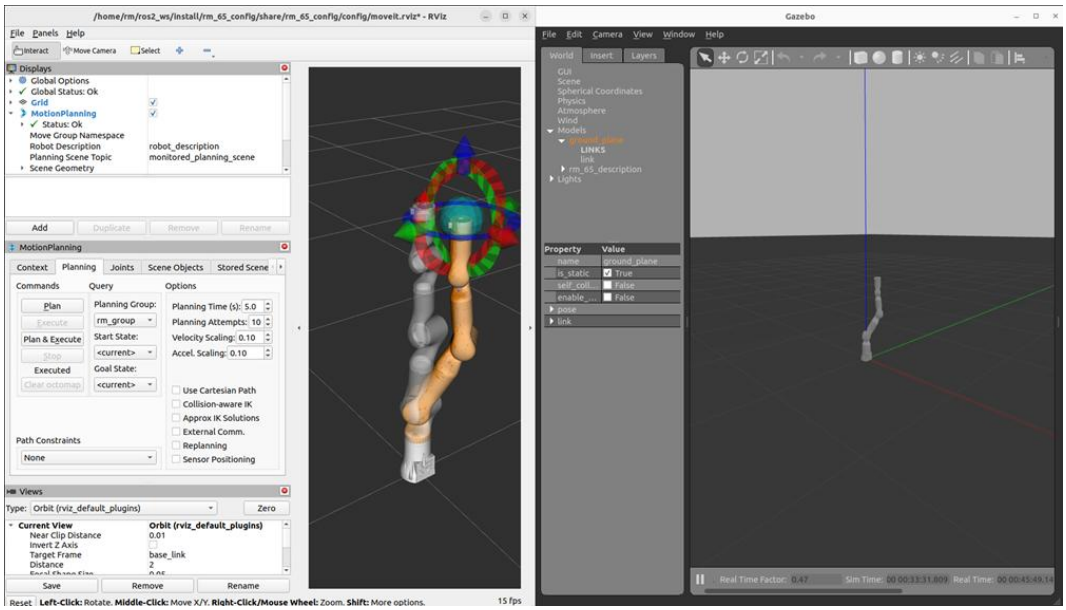
8.2. rm_gazebo package use

When using this node, enter the following commands to start the gazebo simulation and moveit2 to realize the control of moveit2 on the simulation robotic arm. The commands are as follows.

```
ros2 launch rm_gazebo gazebo_<arm_type>_demo.launch.py
ros2 launch rm_<arm_type>_config gazebo_moveit_demo.launch.py
```

In practice, the above <arm_type> needs to be replaced by the actual model of the robotic arm. The available models of the robotic arm are 65, 63, eco65, and 75.

The following screen appears in the interface after the successful startup.



For a more detailed introduction to the package, please refer to the detailed description document of the package.

9. rm_example Package Introduction

9.1. rm_example package function

(1) Realize the use and code development of some common functions of the robotic arm, mainly including changing the work coordinate system, obtaining the current state of the robotic arm, moveJ angle planning motion, moveJP pose planning movement and moveL linear motion.

9.2. rm_example package use

1. Change the work coordinate system.

To change the current work coordinate system of the robotic arm, execute the following two commands to realize this function.

```
ros2 launch rm_driver rm_<arm_type>_driver.launch.py
ros2 run rm_example rm_change_work_frame
```

In practice, the above <arm_type> needs to be replaced by the actual model of the robotic arm. The available models of the robotic arm are 65, 63, eco65, and 75.

After a successful operation, the following statement appears in the end.

```
[INFO] [1701416861.664450164] [changeframe]: *******Switching the tool coordinate system succeeded
```

2. Get the current state information of the robotic arm.

To get the current state of the robotic arm, execute the following two commands to realize this function.

```
ros2 launch rm_driver rm_<arm_type>_driver.launch.py
```

```
ros2 run rm_example rm_get_state
```

In practice, the above <arm_type> needs to be replaced by the actual model of the robotic arm. The available models of the robotic arm are 65, 63, eco65, and 75.

After a successful operation, the following statement appears in the end.

```
[INFO] [1701417669.964643809] [get_state]: joint state is: [-0.001000, 0.002000, 0.000000, 0.000000, 0.000000, -0.001000, -0.001000]
[INFO] [1701417669.964801453] [get_state]: pose state is: [0.000025, 0.000000, 0.850500, 0.000000, 0.000000, 0.000000]
[INFO] [1701417669.964838239] [get_state]: arm_err is: 0
[INFO] [1701417669.964852780] [get_state]: sys_err is: 0
```

3. MoveJ the motion of the robotic arm.

To control the joint motion of the robotic arm, execute the following two commands to realize this function.

```
ros2 launch rm_driver rm_<arm_type>_driver.launch.py
ros2 launch rm_example rm_<dof>_movej.launch.py
```

In practice, the above <arm_type> needs to be replaced by the actual model of the robotic arm. The available models of the robotic arm are 65, 63, eco65, and 75.

dof represents the current degree of freedom message of the arm, and the parameters can be selected as 6dof and 7dof.

For example, when starting the 7-axis robotic arm, the following commands are needed.

```
ros2 launch rm_example rm_7dof_movej.launch.py
```

After a successful operation, the following statement appears in the end.

```
[INFO] [launch]: All log files can be found below /home/rm/.ros/log/2023-12-01-16-31-59-669068-rm-desktop-9816
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [movej_demo-1]: process started with pid [9817]
[movej_demo-1] [INFO] [1701419519.880260228] [Movej_demo]: arm_dof is 7
[movej_demo-1]
[movej_demo-1] [INFO] [1701419520.691271905] [Movej_demo]: *******Movej succeeded
[movej_demo-1]
```

4. MoveJ_P motion of the robotic arm.

To control the joint MoveJ_P motion of the robotic arm, execute the following two commands to realize this function.

```
ros2 launch rm_driver rm_<arm_type>_driver.launch.py
ros2 run rm_example movejp_demo
```

In practice, the above <arm_type> needs to be replaced by the actual model of the robotic arm. The available models of the robotic arm are 65, 63, eco65, and 75.

After a successful operation, the following statement appears in the end.

```
[INFO] [1701421702.758055237] [Movejp_demo_node]: *******MoveJP succeeded
```

5. MoveL the motion of the robotic arm.

To control the joint MoveL motion of the robotic arm, execute the following two commands to realize this function.

```
ros2 launch rm_driver rm_<arm_type>_driver.launch.py
ros2 run rm_example movel_demo
```

In practice, the above <arm_type> needs to be replaced by the actual model of the robotic arm. The available models of the robotic arm are 65, 63, eco65, and 75.

After a successful operation, the following statement appears in the end.

```
[INFO] [1701422229.234605201] [Movel_demo_node]: *******MoveJP succeeded
[INFO] [1701422230.095942637] [Movel_demo_node]: *******MoveL succeeded
```

10. rm_doc Package Introduction

10.1 rm_doc package function

(1) rm_doc is a document introduction of the package, mainly including the introduction document and use document of each package

11. rm_install Package Introduction

11.1. rm_install package function

(1) rm_install is an installation introduction package, mainly including three scripts: 1. ROS2 installation script ros2_install.sh; 2. moveit2 installation script moveit2_install.sh; 3. package installation script lib_install.sh, and an installation instruction document "RealMan Robotic Arm ROS2 rm_install Package Detailed Description V1.0", which must be referred to for script use in actual use.