● 机械臂三维模型的状态显示

wpr_warehouseing_monitor 里的两个机械臂分别编号为 arm_4 和 arm_5，可以通过两个主题去控制其在 Rviz 里的显示：

编号	主题	消息类型
arm_4	arm_4/joint_position	sensor_msgs::JointState
arm_5	arm_5/joint_position	sensor_msgs::JointState

所以，我们只需要如下步骤就可以实现对 Rviz 里机械臂模型的状态显示：

(一) 定义一个 ROS 主题的发布对象；

(二) 发布到相应机械臂的主题上（比如 arm_4/joint_position）；

(三) 定义一个 sensor_msgs::JointState 类型的消息包；

(四) 在消息包里填写要显示的角度数值；

(五) 将消息包发布到对应的主题上。

示例代码如下：

```
#include <ros/ros.h>
#include <sensor_msgs/JointState.h>

int main(int argc, char** argv)
{
    ros::init(argc, argv, "joints_demo");

    // 发布主题
    ros::NodeHandle n;
    ros::Publisher arm_4_pub =
n.advertise<sensor_msgs::JointState>("arm_4/joint_position", 30);
    ros::Publisher arm_5_pub =
n.advertise<sensor_msgs::JointState>("arm_5/joint_position", 30);

    // 初始化消息包
    sensor_msgs::JointState joint_msg;
    joint_msg.name.resize(7);
    joint_msg.position.resize(7);
    joint_msg.velocity.resize(7);
    joint_msg.name[0] = "joint1";
    joint_msg.name[1] = "joint2";
    joint_msg.name[2] = "joint3";
    joint_msg.name[3] = "joint4";
    joint_msg.name[4] = "joint5";
    joint_msg.name[5] = "joint_griper";
    joint_msg.name[6] = "joint_finger";
    joint_msg.position[0] = 0.0f;
    joint_msg.position[1] = 0.0f;
```

```
    joint_msg.position[2] = 0.0f;
    joint_msg.position[3] = 0.0f;
    joint_msg.position[4] = 0.0f;
    joint_msg.position[5] = 0.0f;
    joint_msg.position[6] = 0.0f;

    // 消息包发送频率
    ros::Rate r(10);

    while (ros::ok())
    {
        // 发给arm4机械臂（joint1转动）
        joint_msg.position[0] += 0.1;
        arm_4_pub.publish(joint_msg);
        // 发给arm5机械臂（joint2转动）
        joint_msg.position[1] += 0.1;
        arm_5_pub.publish(joint_msg);

        ros::spinOnce();
        r.sleep();
    }

    return 0;
}
```

在这个程序里，进行了如下操作：

(一) 定义了两个发布对象，分别在 arm_4 和 arm_5 的主题上发布机械臂角度消息；

(二) 定义了一个 sensor_msgs::JointState 类型的消息包 joint_msg，初始化所有角度为 0；

(三) 构建一个 while 循环，在循环内部对 joint_msg 消息包里的角度值进行累加，然后分别通过两个发布对象发布到相应的显示主题上。

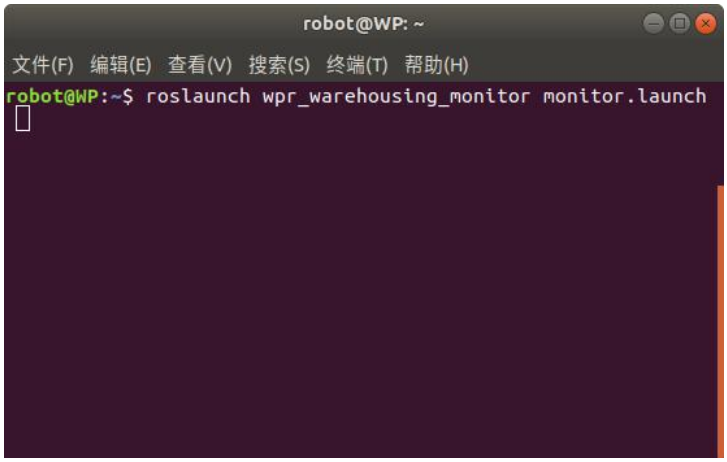
这个例子程序在 Github 上有相应的包，网址是：

https://github.com/zju-g/s6h4d_demo

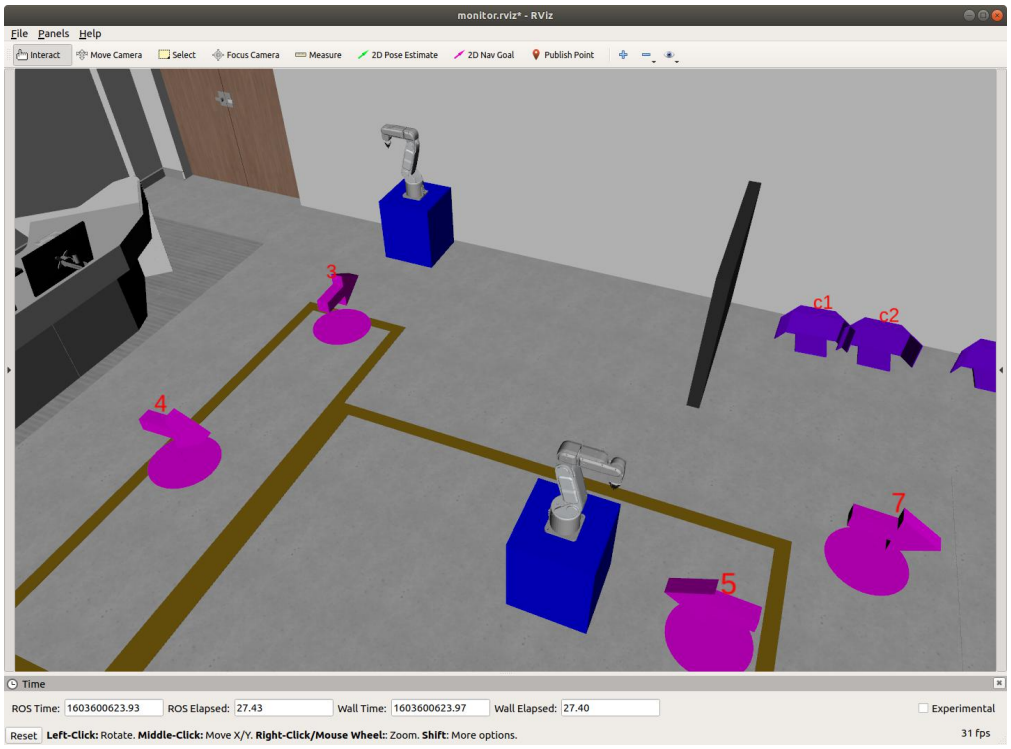
将这个 s6h4d_demo 包 clone 到 catkin 工作目录里，编译之后即可运行查看效果：

(一) 运行 wpr_warehousing_monitor 的 Rviz 界面：

```
roslaunch wpr_warehousing_monitor monitor.launch
```

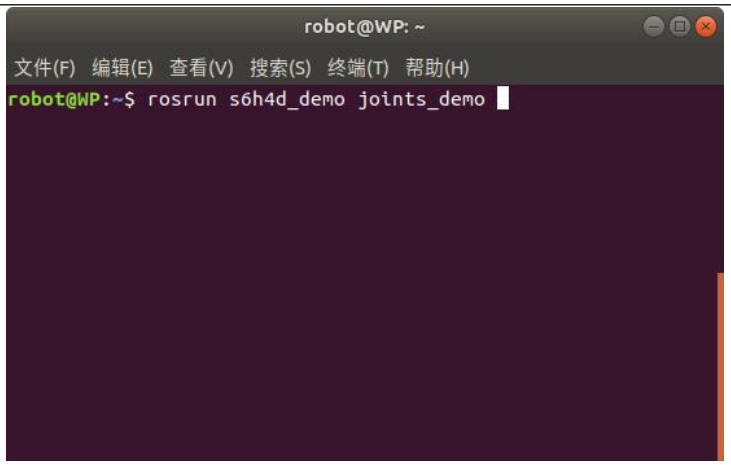


(二) 在弹出的 Rviz 里可以看到两个机械臂模型，其中下侧的是 arm_4，上侧的是 arm_5



(三) 运行编译好的例子程序，对机械臂的状态进行改变：

```
rosrun s6h4d_demo joints_demo
```



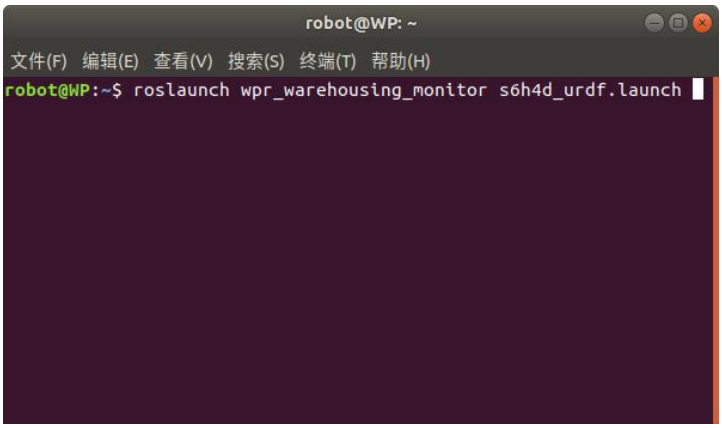
(四) 运行后再回到 Rviz 界面，可以看到两个机械臂开始进行对应的运动。

● 机械臂预设动作显示

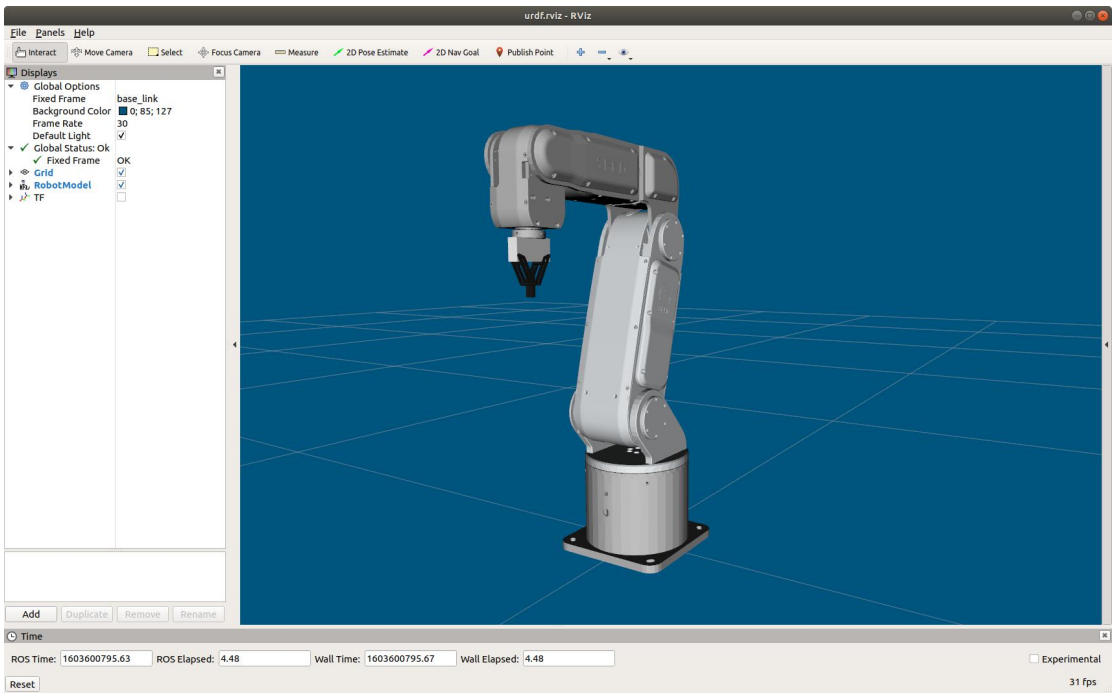
在没有连接机械臂实体的时候，可以通过给机械臂设置一些指定的动作进行显示。我们只需要先确定几个机械臂姿态的关键帧，然后记录下机械臂当时的所有关节角度，然后将这些角度写到代码里，wpr_warehousing_monitor 里的 ArmAction 会自动对关键帧之间的离散动作进行插补，最后形成一个连贯的动画。添加预设动作的步骤如下：

(一) 记录关键帧的角度数值。首先运行机械臂单独的模型显示：

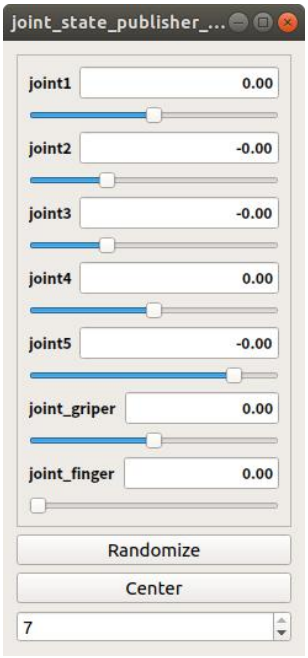
```
roslaunch wpr_warehousing_monitor s6h4d_urdf.launch
```



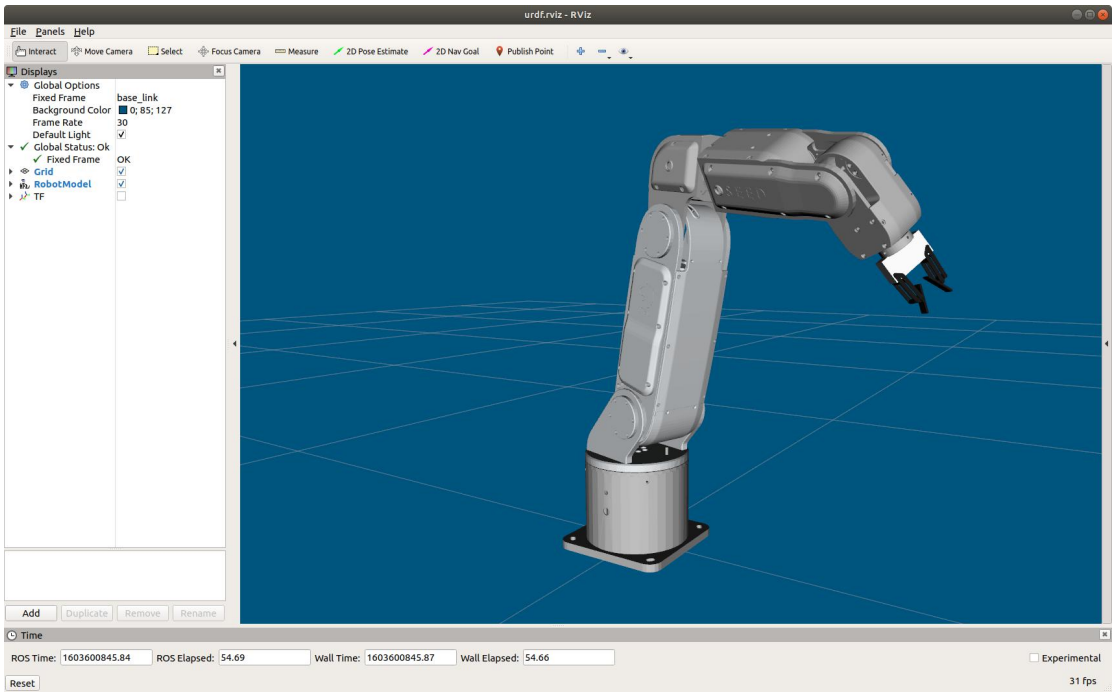
运行后会弹出一个 Rviz，显示单个机械臂的模型：



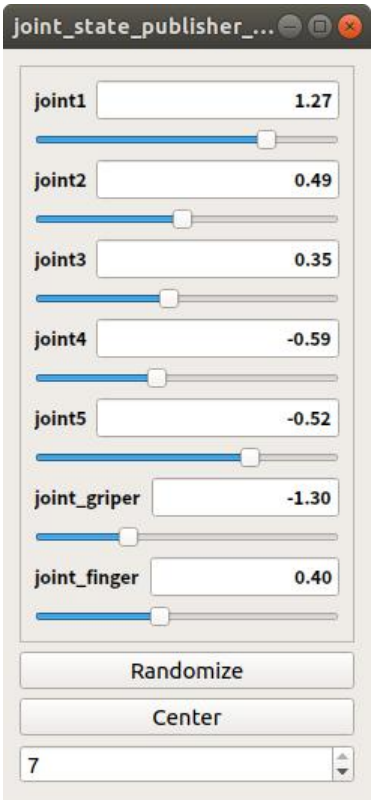
(二) 在任务栏可以看到一个调节机械臂角度的面板，在面板上拖动滑杆，可以改变机械臂三维模型的显示角度。



拖动窗体中的滑杆，可以改变 Rviz 里机械臂的姿态：

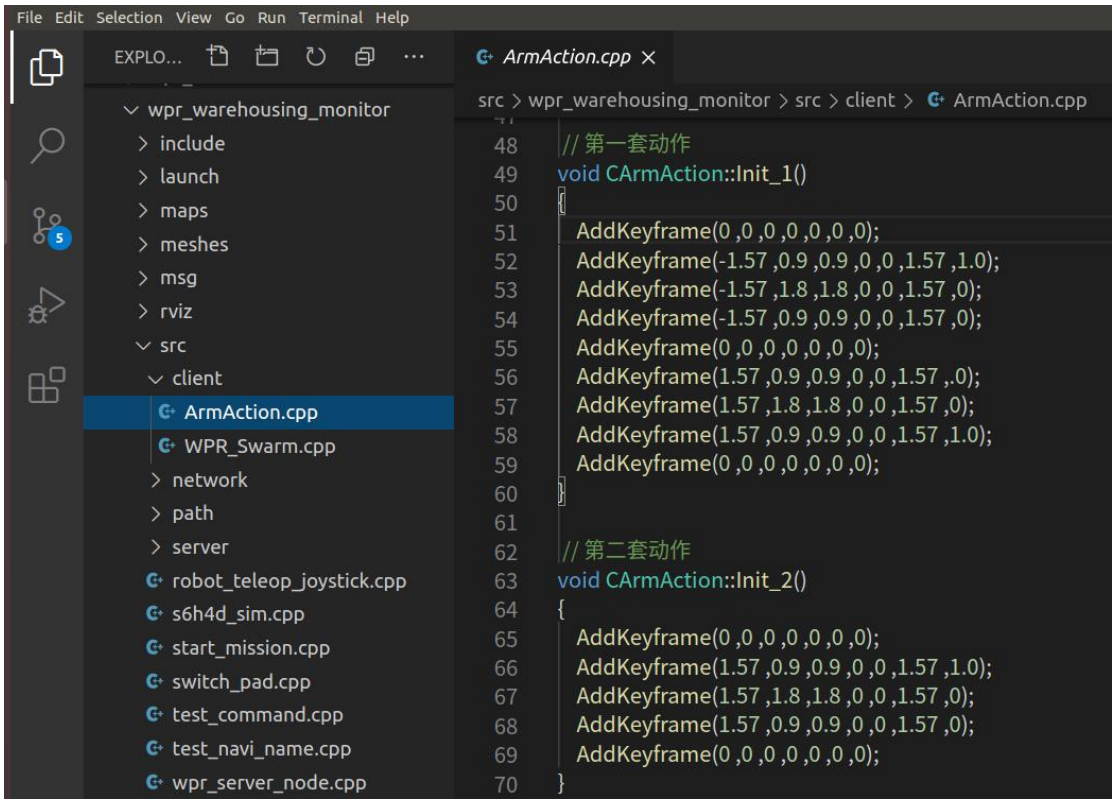


将机械臂运动到关键帧姿态时，记录下所有关节的角度数值：



将所有关键帧的角度数值全部记录完毕后，可以开始往代码里填写这些数值。

(三) 打开 wpr_warehousing_monitor/src/client/ArmAction.cpp 文件，在 49 行和 63 行可以看到两个预设动作，这两个动作和 wpr_command_center 界面上的两个按钮分别对应。



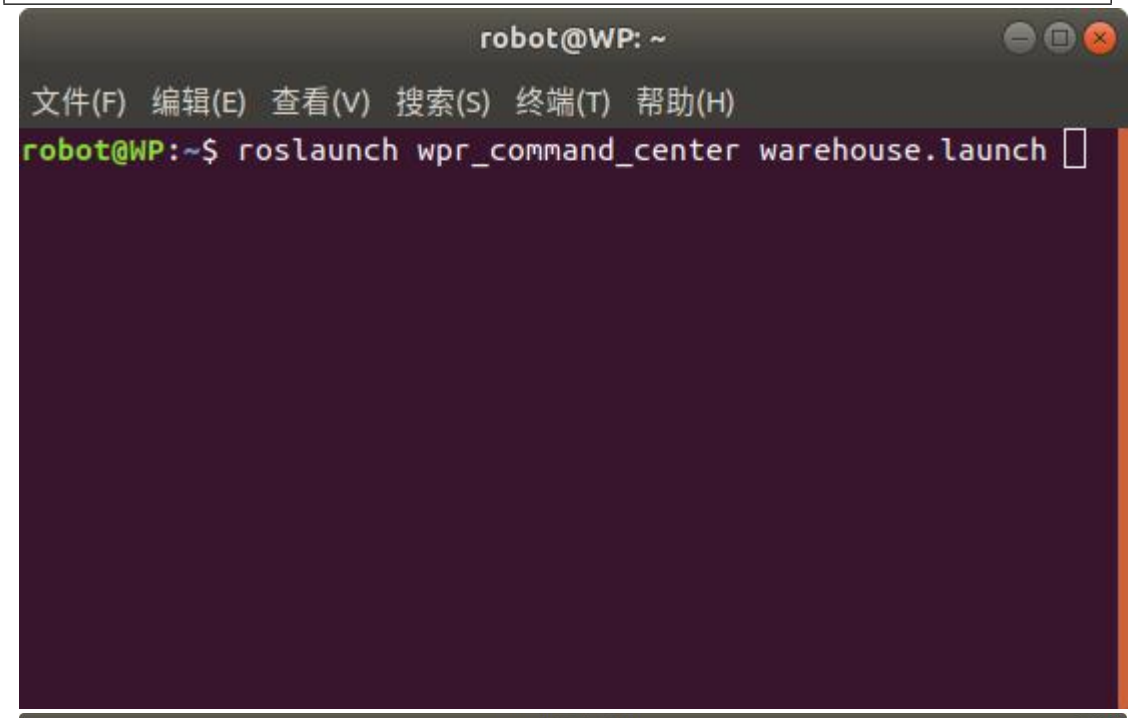
可以看到 AddKeyFrame()函数一共有 7 个参数，就是刚才记录的关键帧里的 7 个角度数值。

每一个关键帧对应一句 AddKeyFrame()函数，相邻关键帧的间隔时间是 3 秒钟。

填写完所有关键帧后，运行 catkin_make 编译生效。

(四) 修改完的动作，可以通过 wpr_command_center 界面上的两个按钮进行激活。在终端程序里运行如下指令：

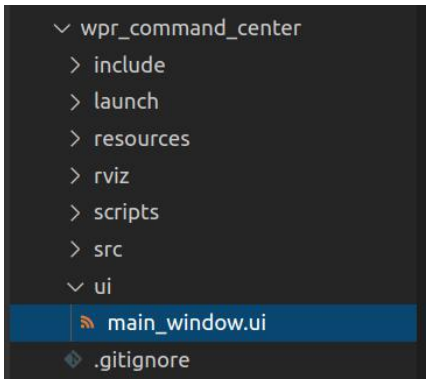
```
roslaunch wpr_command_center warehouse.launch
```



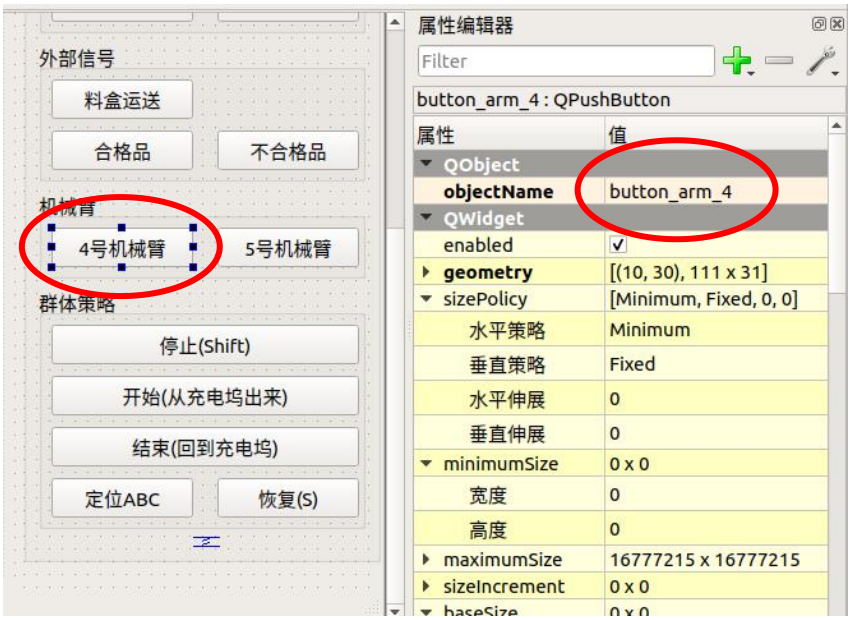
在弹出的界面里，按钮“4 号机械臂”是让 arm_4 机械臂运行第一套动作（Init_1()函数里的动作）；按钮“4 号机械臂”是让 arm_5 机械臂运行第二套动作（Init_2()函数里的动作）。

在这个界面添加新按钮的方法：

(一) 用 QtDesigner 打开 wpr_command_center/ui/main_window.ui；



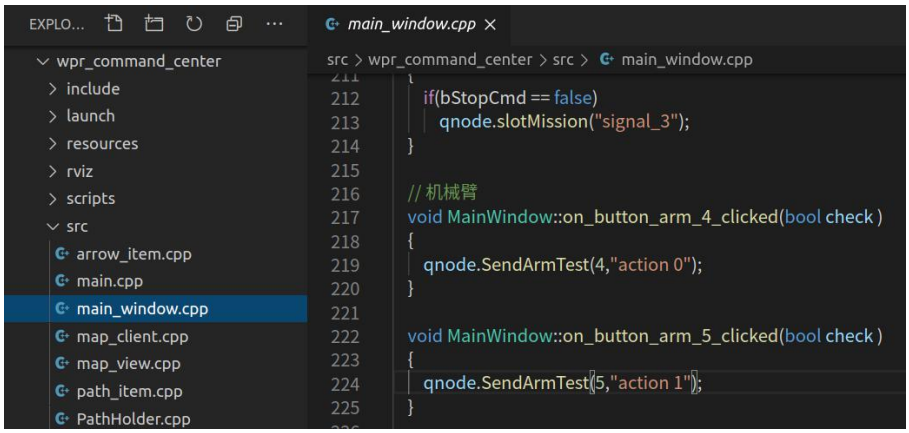
(二) 在打开的界面视图上添加按钮，并修改它的 objectName 为特定意义的名称，后面定义这个按钮的响应函数时，会和它的 objectName 有关联；



(三) 在 wpr_command_center/include/main_window.h 以及 wpr_command_center/src/ 下的 main_window.cpp 添加按钮的响应函数。函数名称格式为:

on_（上面按钮的 objectName）_clicked(bool check)

这样这个函数就自动和前面的按钮建立起联系。

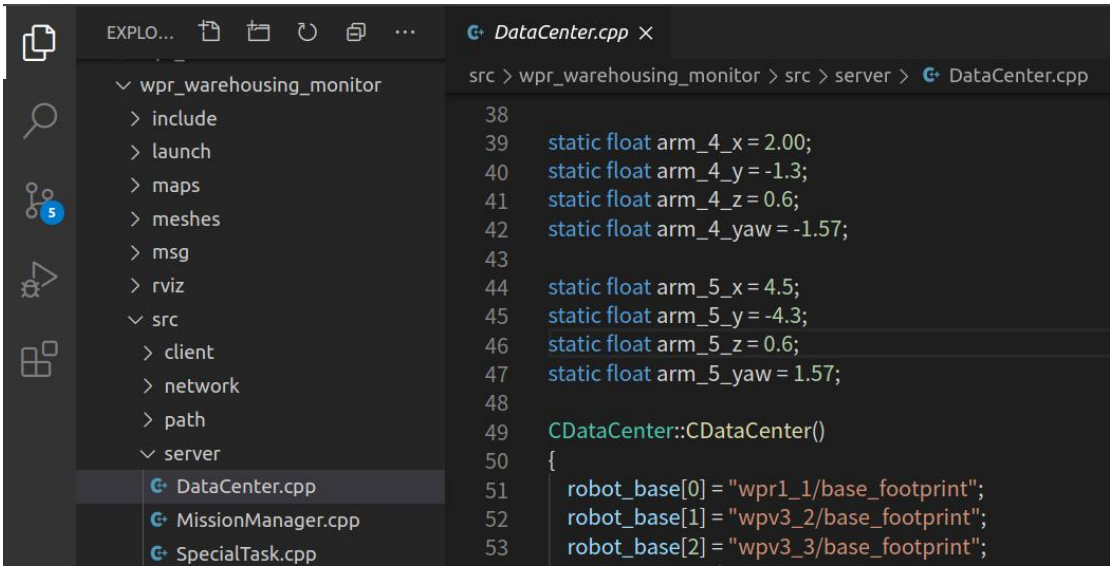


● 机械臂模型的显示位置

机械臂模型在 Rviz 里的位置坐标是可以在代码里修改的，对应的代码文件为：

wpr_warehousing_monitor/src/server/DataCenter.cpp

文件的 39 行开始 arm_4_x、arm_4_y、arm_4_z、arm_4_yaw 就是 arm_4 机械臂模型的基座中心坐标，单位是米。44 行开始 arm_5_x、arm_5_y、arm_5_z、arm_5_yaw 就是 arm_5 机械臂模型的基座中心坐标，单位是米。



坐标系的原点在开始建图时机器人所处的位置，以机器人的正前方为 X 轴的正方向：