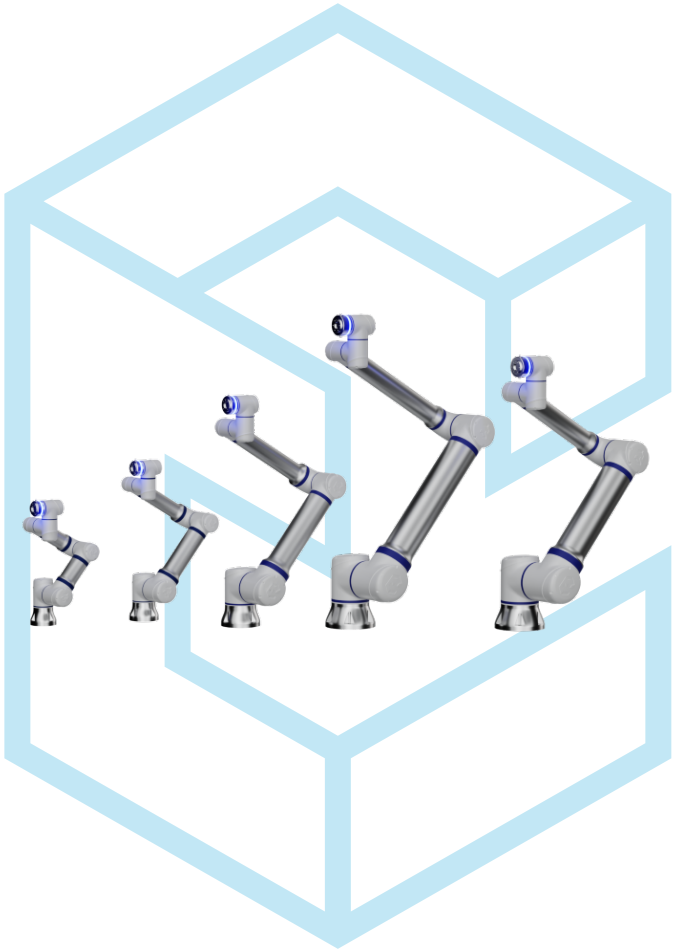
ELITE ROBOTS
艾利特机器人

# ELITE ROBOTS
# 二次开发手册



## ROS2 软件包快速使用手册

艾利特智能机器人股份有限公司

2026-01-19

版本：Ver2.15.0

# 使用前请仔细阅读本手册

此版本手册同控制软件版本保持一致，产品版本信息及所需软件版本请参阅第 1 章，使用前请仔细核对实际产品版本信息，确保一致和符合版本限制。

本手册会定期进行检查和修正，更新后的内容将出现在新版本中。本手册中的内容或信息如有变更, 恕不另行通知。

艾利特智能机器人股份有限公司对本手册中可能出现的任何错误概不负责。

艾利特智能机器人股份有限公司对因使用本手册及其中所述产品而引起的意外或间接伤害概不负责。

安装、使用产品前，请阅读本手册。

请保管好本手册，以便可以随时阅读和参考。

本手册图片仅供参考，请以收到的实物为准。

# 目录

# 第 1 章 概述

ROS（Robot Operating System，机器人操作系统）是一组开源的软件库和工具，为机器人应用的开发与运行提供了灵活而强大的框架。ROS 2 是 ROS 1 的升级版本，在性能、通信机制、安全性以及实时性方面均进行了大幅优化，更适合科研与工业级机器人系统。

在 ROS 2 中，程序通常以软件包（packages）的形式组织，并在一个工作空间（workspace）中使用 colcon（ROS 2 多包构建工具）进行构建。每个软件包可以包含一个或多个节点（nodes），节点之间可通过话题（topics）、服务（services）、动作（actions）等通信机制进行交互。

ELITE ROS 2 软件包主要使用以下 ROS 2 通信方式：

- **话题（Topics）**发送机器人状态信息。
- **动作（Actions）**处理长时间运行的任务，如运动执行等。
- **服务（Services）**处理快速请求，如启用机器人、切换机器人状态等。

## 1.1 主要功能

Elite ROS 2 软件包提供以下核心功能：

- **自定义控制器**
- **MoveIt 2 集成：**
  提供运动规划与执行、碰撞检测、运动学求解与轨迹生成的完整支持，并可根据需求扩展其他功能。

- **RViz** 可视化与类仿真环境
  - 实时可视化机器人模型、关节状态及传感器数据等。
  - 提供基础 **类仿真环境**，用于预览规划执行的运动、实现与机器人交互控制等功能。

- **Gazebo 仿真环境：**
  提供完整的物理仿真环境，用于测试控制算法及机器人在逼真场景下的行为响应。
- **控制驱动接口：**
  ROS 2 驱动节点用于向机器人控制器发送控制指令并接收状态反馈。

# 1.2 ELITE ROS2 软件包结构

ELITE ROS 2 软件包整体结构如下：

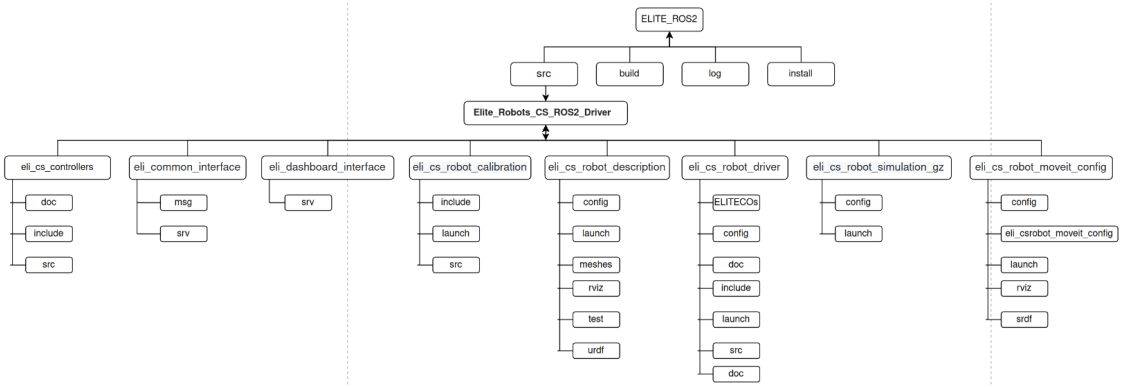



**图 1-1:** ELITE ROS2 软件包结构

**软件包目录说明**

- eli_common_interface：定义一些共用的服务或消息接口。
- eli_dashboard_interface：定义 Dashboard 节点使用的消息接口。
- eli_cs_controllers：Elite CS 机器人控制器的具体实现。
- eli_cs_robot_calibration：用于从真实机器人读取标定数据的工具。
- eli_cs_robot_description ：Elite CS 机器人的描述文件和模型。
- eli_cs_robot_driver：与机器人通信的硬件接口、驱动与示例。
- eli_cs_robot_simulation_gz：Gazebo 仿真配置与示例。
- eli_cs_robot_moveit_config：MoveIt 配置与示例。

# 1.3 支持的系统与平台要求

ELITE ROS2 软件包旨在为 ELITE 协作机器人提供完整的 ROS 2 集成支持，兼容 CS、ES 系列全部机型。目前，该软件包已在 **Ubuntu 22.04（Jammy Jellyfish）+ ROS 2 Humble** 上正式测试和支持。

**表 1-1.** ROS2 软件包硬件要求

| 组件 | 推荐最低要求 | 备注 |
| --- | --- | --- |
| 处理器（CPU） | 64 位 Intel i5 / i7（或同等 AMD 处理器） | 推荐使用多核 CPU，可显著提升编译与运行效率。 |
| 内存（RAM） | 至少 8 GB（推荐 16 GB） | MoveIt 2 运动规划和 RViz 可视化需要较大的运行内存。 |
| 存储（Storage） | 至少 30 GB 可用空间（建议使用 SSD） | 更快的存储可提升启动速度和数据处理能力。 |
| GPU（optional） | 支持 CUDA 的 NVIDIA GPU | 非必需，但对高级感知应用（如基于 AI 的视觉处理、实时 SLAM 等）有显著加速效果。 |

## 1.4 ROS2 基本命令

本节介绍一些常用的 ROS2 基础命令。这些命令可用于构建、运行和调试 ROS 2 节点，管理查询主题、服务、参数以及启动文件等。

**表 1-2.** ROS2 基础命令

| 命令 | 说明 |
| --- | --- |
| cd /<ros2_ws> | 进入 ROS 2 工作空间目录 |
| colcon build | 编译 ROS 2 工作空间内所有软件包 |
| source ./install/setup.bash | 添加软件包到环境变量（编译完成后必须执行） |
| ros2 pkg list | 查看所有可用的软件包 |
| ros2 pkg prefix <package-name> | 查找软件包的安装路径 |
| ros2 run <package-name> <executable-name> | 运行 ROS 2 节点 |
| ros2 node list | 查看当前正在运行的节点 |

| 命令 | 说明 |
| --- | --- |
| ros2 node info <node-name> | 查看指定节点的信息 |
| ros2 node kill <node-name> | 停止运行中的节点 |
| ros2 topic –help | 查看所有可用的话题操作 |
| ros2 topic list | 查看所有活动中的话题 |
| ros2 topic echo /<topic_name> | 实时显示指定话题的消息内容 |
| ros2 topic info /<topic_name> | 查看话题的信息（消息类型、发布者、订阅者等） |
| ros2 topic type /<topic_name> | 查看话题的消息格式 |
| ros2 interface show <msg_type> | 显示特定消息类型的详细信息（类似于 ROS 1 中的 rosmsg show） |
| ros2 service –help | 查看所有可用的服务操作 |
| ros2 service list | 查看活动中的服务列表 |
| ros2 service call <service-name> <service-type> argument | 手动调用服务 |
| ros2 launch <package_name> <file.launch.py> | 运行启动文件 |
| ros2 param list | 查看某个节点的可用参数 |
| ros2 param get <node-name> <param-name> | 获取某个参数的值 |
| ros2 param set <node-name> <param-name value> | 设置参数的值 |
| ros2 bag record /<topic_name> | 记录某个话题的消息到 ROS 2 bag 文件中 |
| ros2 bag play <bag_file> | 重新播放已记录的 bag 文件消息 |
| ros2 control list_controllers | 查看当前加载的所有控制器及其状态（active/inactive 等） |
| ros2 control list_controller_types | 查看可用控制器类型（例如 joint_trajectory_controller 等） |

| 命令 | 说明 |
| --- | --- |
| ros2 control list_hardware_components | 查看当前系统中注册的硬件组件（机器人本体、传感器等） |
| ros2 control list_hardware_interfaces | 查看可用的硬件接口（position, velocity, effort, sensor interfaces 等） |
| roslaunch <package> <file.launch> | 启动某个包内的 launch 文件（常用于统一管理多个节点） |

# 第 2 章 机器人端配置

使用 Elite_ROS2_Drivers 控制机器人有两种方式：实体机器人或虚拟示教器（ELIBOT-Sim）。

**提示**

两种方式任选其一。如无实体机器人，可选择使用虚拟示教器（ELIBOTSim）。

## 2.1 机器人配置

### 2.1.1 网络配置

#### （1） CS 系列标准控制柜

如您使用 CS 系列标准控制柜，需将 FB1、FB2 同时接入局域网，并设置好 IP 地址。

**为什么需要连接两个网口?**

因为 CS 系列标准控制柜采用双处理器的分布式架构：

- FB1 网口对应一个处理器，负责运行机器人的 dashboard 功能；
- FB2 网口对应另一个处理器，负责运行与 Elite ROS2 Drivers 进行通讯的控制软件。

因此需要同时连接两个网口。

从机器人软件 2.14.3 版本起，若使用 Elite ROS2 Drivers 的 headless 模式，可以只连接 FB2 网口。

#### （2） CS 系列 B1 控制柜

如您使用 CS 系列 B1 控制柜，仅需连接一个网口。

## 2.1.2 设置远程控制模式

1. 点击示教器右上角的 Elite 图标 > 「设置」。

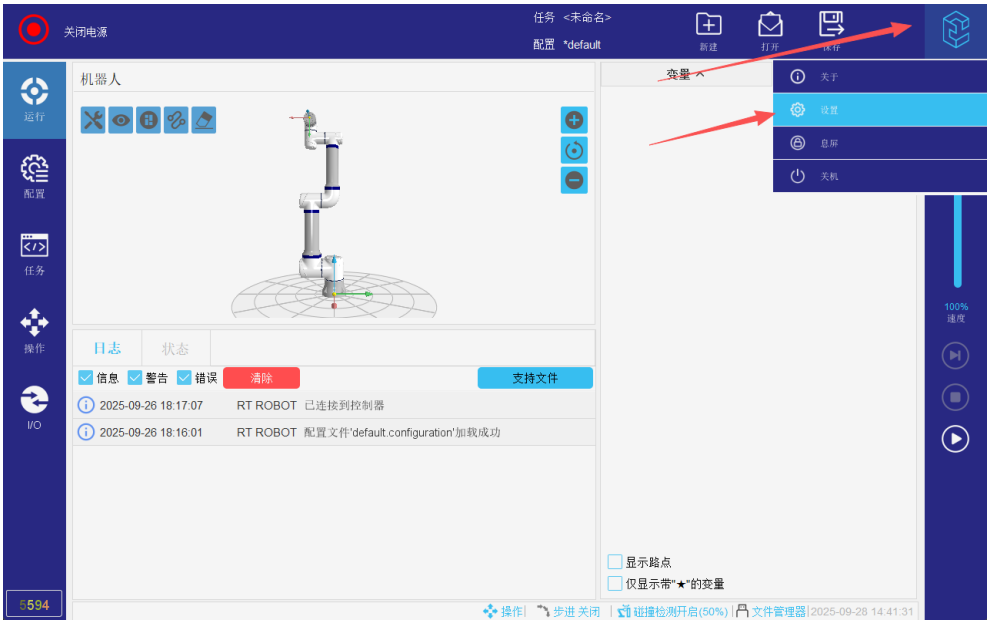



图 2-1: 示教器设置

2. 启用/禁用「远程控制模式」，点击左下角退出：


图 2-2: 远程控制模式设置

3. 若选择启用远程控制模式，界面右侧栏会出现「本地控制」选项，请切换到「远程控制」模式。


**图 2-3:** 远程控制选项


**图 2-4:** 远程控制界面

**提示**

如果 CS 和 ES 系列机器人的远程控制模式是禁用状态，则会处于“混合模式”，即外部控制和示教器本地控制均可生效。因此，在上述步骤中，也可将远程控制模式设为禁用。

### 2.1.3 测试网络连接

可通过以下两种方式测试 SDK 是否能与机器人通讯：

1. 使用机器人脚本
2. 使用 bash 指令测试

推荐使用方法 1（机器人脚本）。若脚本测试成功，说明机器人配置基本没有问题，通常不会出现后续运行运动指令时的连接问题。

方法 2（Bash 指令）可用于初步排查网络是否畅通，即使成功也可能因路由等原因导致运动控制失败。因此，若使用方法 2 无法正常控制机器人运动，请使用方法 1。

#### 1. 使用机器人脚本（推荐）

1. **获取 IP 地址**
   在要运行 SDK 的 Linux 操作系统中执行以下指令：

   ```
   1 ifconfig
   ```

   记录与机器人连接的网口 IP（记为 Local IP）。

   > **提示**
   >
   > **可能遇到的问题**：
   > Linux 中执行 **ifconfig** 时可能会出现找不到命令的提示，此时需要安装一下该指令，例如在 Ubuntu 中执行 **sudo apt install net-tools** 指令安装。

2. **在示教器上创建测试脚本**
   - 点击示教器左侧栏的「任务」。


图 2-5: 选择任务

- 选择「占位节点」>「高级」>「脚本」。

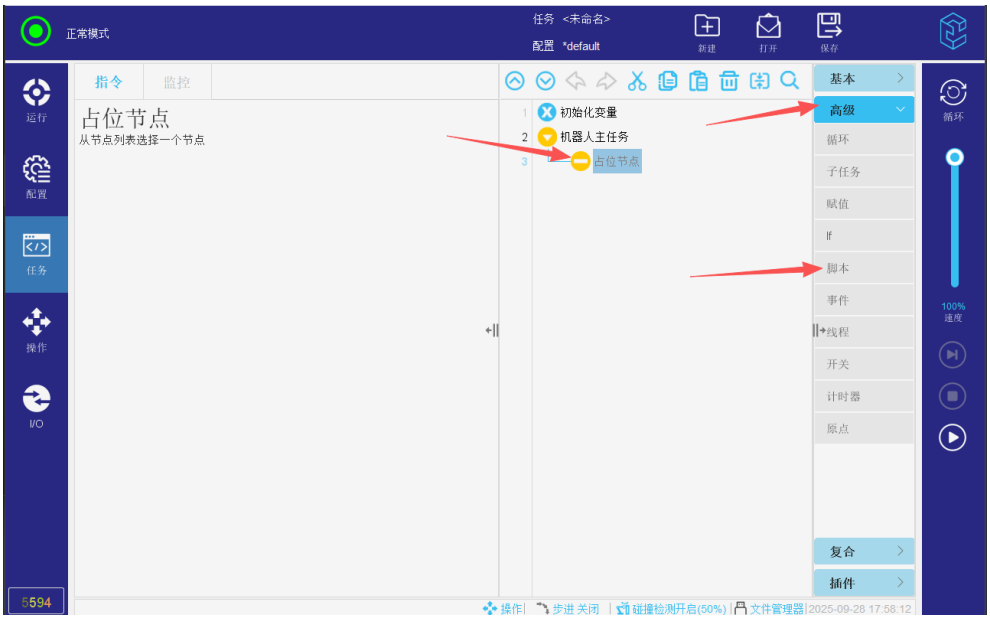



图 2-6: 选择脚本

- 点击左侧下拉框，选择「文件」，点击「编辑」。

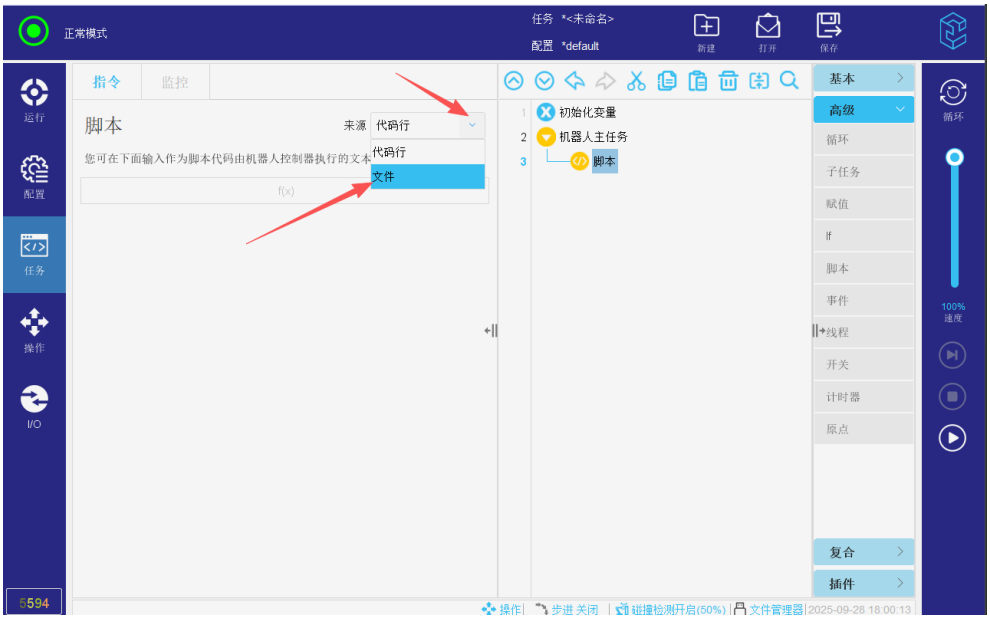



图 2-7: 编辑文件

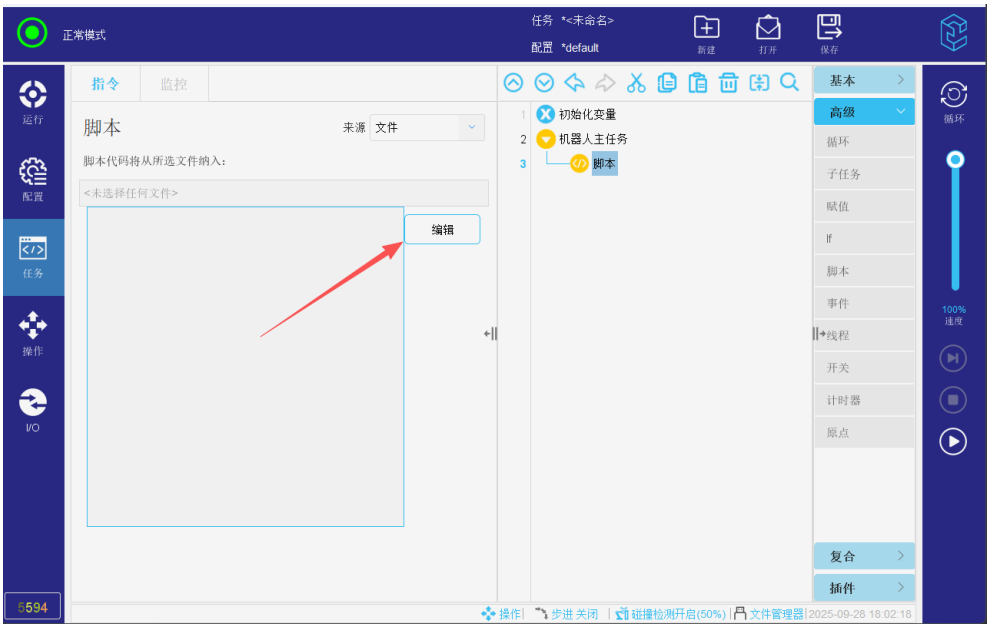



图 2-8: 编辑脚本

- 将以下内容写入文本编辑器中（注意将 Your.Local.IP.Addr 替换为刚才记录的 Local IP）：

```
1 socket_open("Your.Robot.IP.Addr", 50001)
2 socket_send_string("Hello SDK\n")
3 sleep(1)
```

- 点击「保存」，并设置文件名。

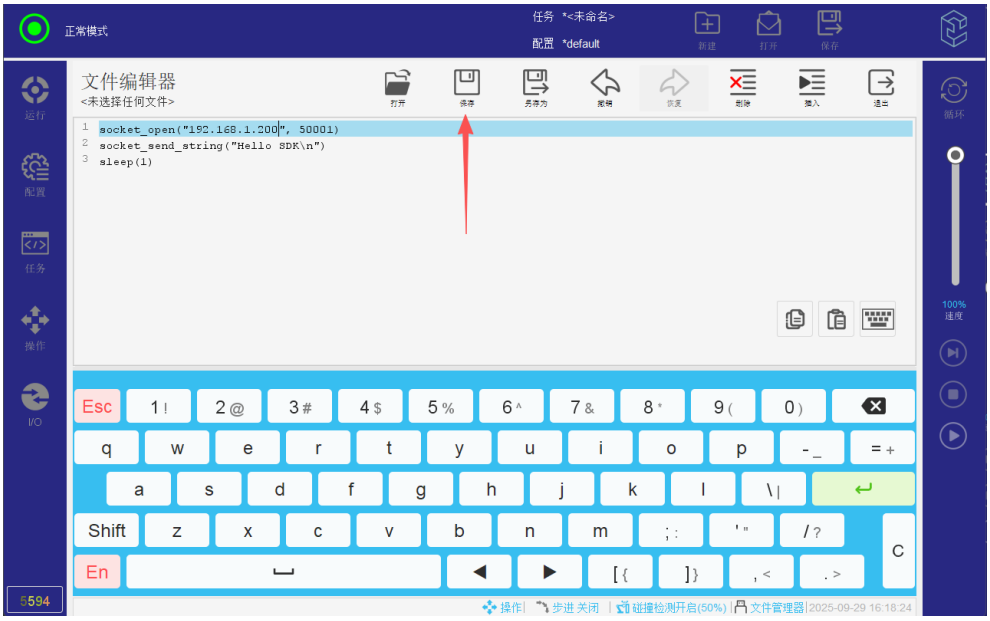



图 2-9: 保存脚本

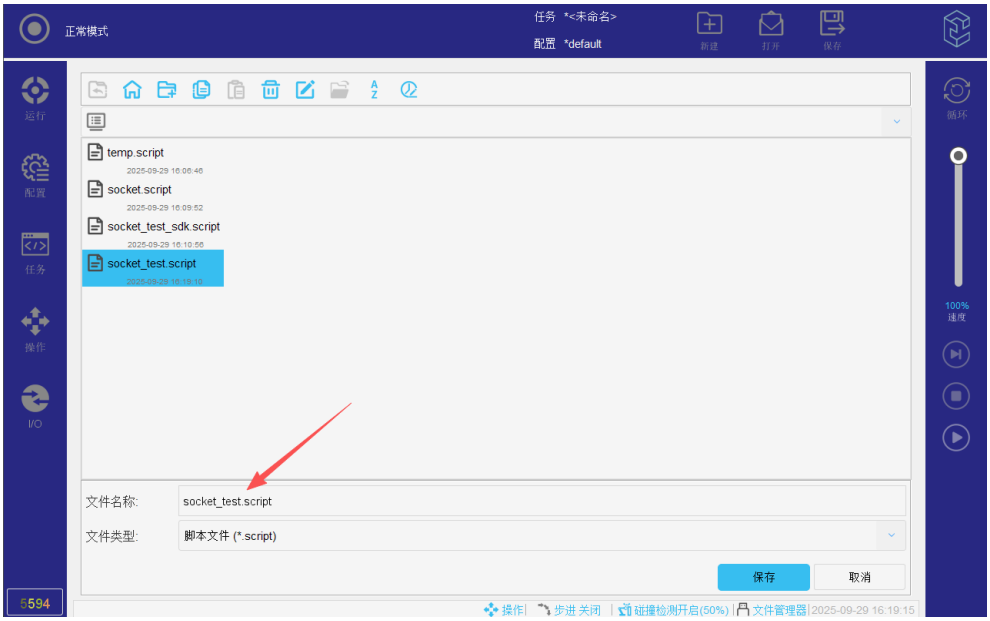



图 2-10: 设置脚本名称

3. **创建 TCP 测试服务器**

   在要运行 Elite_ROS2_Drivers 的操作系统中创建一个端口为 50001 的 TCP 服务。

   将以下内容保存为 sdk_connection_test.py，然后执行 **python3 sdk_connection_test.py** 运行。

```python
import socket


def start_tcp_server(host='localhost', port=50001):
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

    try:
        # 设置accept超时（1秒）
        server_socket.settimeout(1.0)

        server_socket.bind((host, port))
        server_socket.listen(5)
        print(f"TCP服务器启动在 {host}:{port}，等待客户端连接...")

        while True:
            try:
                # 接受客户端连接（现在会超时，让出控制权）
                client_socket, client_address = server_socket.accept()
                print(f"客户端 {client_address} 已连接")

                try:
                    while True:
                        data = client_socket.recv(1024)
                        if not data:
                            print(f"客户端 {client_address} 已断开连接")
                            break

                        received_string = data.decode('utf-8')
                        print(f"来自 {client_address} 的数据: {received_string}")

                except ConnectionResetError:
                    print(f"客户端 {client_address} 异常断开连接")
                except Exception as e:
                    print(f"处理客户端 {client_address} 时发生错误: {e}")
                finally:
                    client_socket.close()

            except socket.timeout:
```

```
                # 超时是正常的，继续循环
                continue

    except KeyboardInterrupt:
        print("\n服务器被用户中断")
    except Exception as e:
        print(f"服务器发生错误: {e}")
    finally:
        server_socket.close()
        print("服务器已关闭")


if __name__ == "__main__":
    start_tcp_server()
```

4. **运行测试**

   点击示教器上的任务运行按钮，如果一切正常，会看到 TCP 服务器接收到”Hello SDK”的一行字符串。

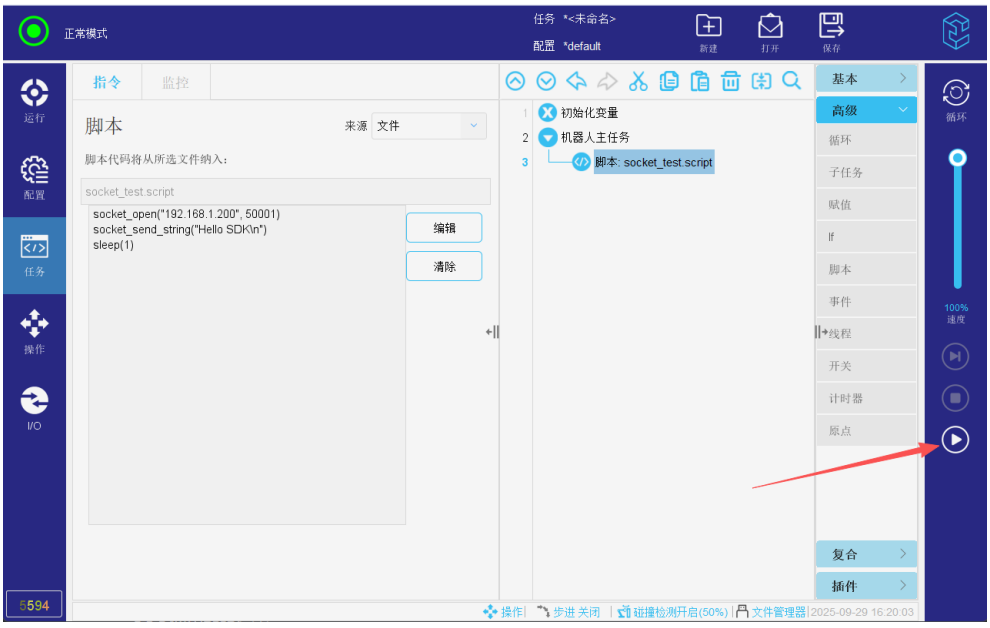



**图 2-11:** 运行测试任务

**可能遇到的问题**:

1. TCP 服务器无法启动

   如果在创建 TCP 服务时失败，可能是系统中其他程序占用了 50001 端口。如需更换端口，机器人脚本中的端口号需要同步修改。

2. 示教器报错：“socket_send_string”：未连接服务端“socket_0”。

这说明机器人无法连接 SDK 的 TCP 服务端口。可能的原因包括：

- 如果使用了虚拟机，需要将虚拟机的网卡设置为桥接模式，并桥接到与机器人相连的网卡上，然后设置好 IP 地址。
- 如您使用的是 CS 系列标准控制柜，可能没有连接 FB2 网口。
- 局域网设置可能关闭了连接，建议将机器人与运行 SDK 的电脑直连。如果您使用的是 CS 系列标准控制柜，可以使用交换机将运行 SDK 的电脑、控制柜 FB1、控制柜 FB2 连接在一起。
- IP 设置不正确，运行 SDK 的电脑 IP 与控制柜网口 IP 不在同一网段。
- 电脑开启了 VPN，建议关闭。
- 服务器没有启动成功，系统中某些应用占用了 50001 端口，请尝试更换端口。（注意服务器端口和机器人脚本中 **socket_open** 指令端口需要同时修改）。
- 服务器被关闭，可能在接收到“Hello SDK”字符串后您关闭了服务器，但此时机器人仍在循环执行脚本。
- 如上述情况均确认无误但仍无法连接，请重启控制柜再重试。

## 2. 使用 Bash 指令

### 1. 获取 IP 地址

在要运行 SDK 的 Linux 操作系统中输入以下指令：

```
1 ifconfig
```

记录下与机器人相连网口的 IP，记为 Local IP。

**提示**

**可能遇到的问题**：
Linux 中执行 **ifconfig** 时可能会出现找不到命令的提示，此时需要安装一下该指令，例如在 Ubuntu 中使用 **sudo apt install net-tools** 指令安装。

### 2. 测试机器人到 SDK 的连接

在要运行 SDK 的 Linux 操作系统中运行以下指令，使用 ssh 登录到机器人的操作系统里（注意替换 **aaa.bbb.ccc.ddd** 为机器人 IP，如您使用的是 CS 系列标准控制柜，则为 FB2 的 IP），密码为 **elibot**：

```
ssh root@aaa.bbb.ccc.ddd
```

登录成功后输入以下指令，检查机器人是否能连接到 SDK（注意替换 **aaa.bbb.ccc.ddd** 为刚才记录的 Local IP）。

```
ping aaa.bbb.ccc.ddd
```

如果连接成功，会有类似以下输出：

```
PING 192.168.1.110 (192.168.1.110): 56 data bytes
64 bytes from 192.168.1.110: seq=0 ttl=64 time=0.373 ms
64 bytes from 192.168.1.110: seq=1 ttl=64 time=0.409 ms
64 bytes from 192.168.1.110: seq=2 ttl=64 time=0.397 ms
64 bytes from 192.168.1.110: seq=3 ttl=64 time=0.432 ms
```

如无法连接，请检查网络设置，例如：IP 地址是否冲突，SDK 所在系统与机器人是否在同一网段，所在的局域网环境等。

**3. 测试 SDK 到机器人的连接**

输入 **exit** 退出登录，返回至要运行 SDK 的操作系统中，执行以下指令，简单检查 SDK 是否能连接上机器人（注意替换 **aaa.bbb.ccc.ddd** 为机器人 IP，如果是 CS 系列标准控制柜，需要确保能够接通 FB1 和 FB2）。

```
ping aaa.bbb.ccc.ddd
```

# 2.2 虚拟示教器仿真配置

请按照以下方法配置虚拟示教器（ELIBOTSim）和 Docker：

**(1) 安装 Docker**

在系统终端中输入以下内容：

```
curl -fsSL https://get.docker.com | sudo sh
```

**(2) 下载仿真脚本**

打开 链接，下载 start_elibotsim.sh 仿真脚本。

下载完成后，进入 start_elibotsim.sh 文件所在目录，右键点击「在终端打开/Open the terminal」。

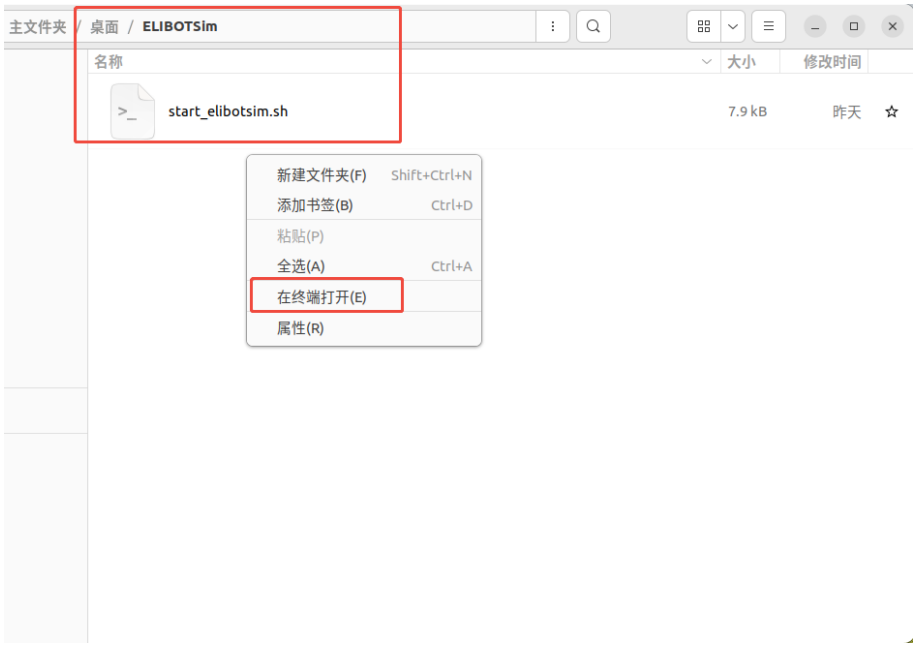



图 2-12: 找到仿真脚本

在终端输入以下内容：

```
chmod +x start_elibotsim.sh

sudo usermod -aG docker $USER
```

**(3) 启动仿真**

输入以下内容启动仿真脚本，首次启动默认访问 Docker 官网拉取仿真镜像。

```
./start_elibotsim.sh
```

如启动失败，出现类似“Failed to create network”的提示，可在指令前加 **sudo**，即输入以下内容：

```
sudo ./start_elibotsim.sh
```

如终端返回类似以下内容，则说明拉取仿真镜像成功：

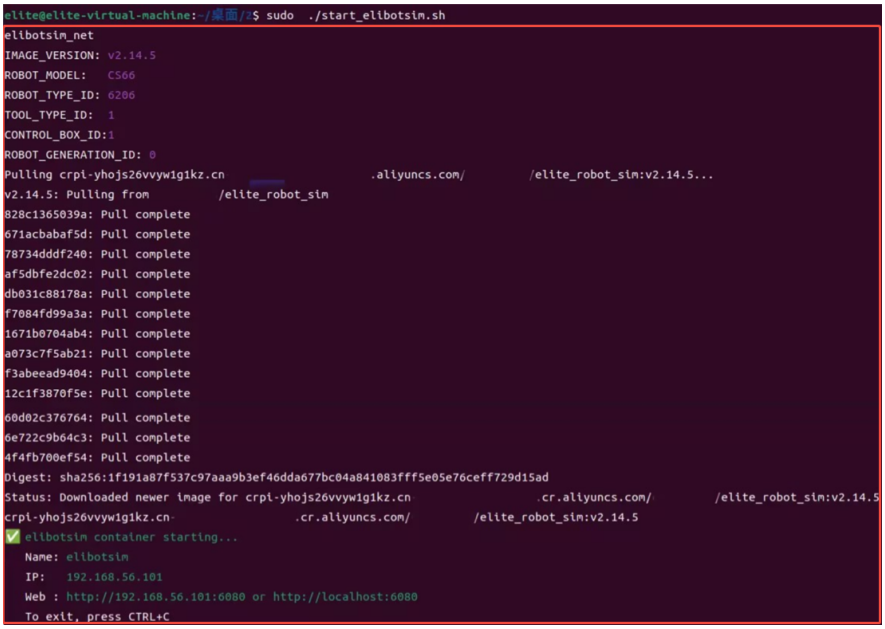



图 2-13: 拉取仿真镜像成功

打开 Web 端输入 http://localhost:6080 ，出现以下仿真画面说明仿真器启动成功。


图 2-14: 仿真器启动成功

**提示**

除第 4.3 节外，其他章节如无实体机器人都需提前启动该仿真环境。

**仿真脚本参数使用说明：**

```
Starts ELIBOTSim inside a docker container

Syntax: start_elibot_sim.sh [options]

options:
  -m <model>   Robot model (CS63, CS66, ES66, etc). Defaults to CS66.
  -c           CONTROL_BOX_ID. Defaults to 3
  -t           TOOL_TYPE_ID. Defaults to 1
  -v <version> Image tag to use. Defaults to '2.14.5'
  -p <folder>  Program dir on host
  -u <folder>  ELITECO dir on host
  -n           Container name. Default: '$CONTAINER_NAME'
  -i           Container IP. Default: $IP_ADDRESS
  -d           Detached mode (background)
  -f           Port forwarding string (e.g. "-p 6080:6080 -p 5900:5900")
  -r <0|1>     Robot series flag. 0 = CS_SERIES, 1 = ES_SERIES
  -h           Show this help
```

例如，如需指定机械臂型号，可输入以下内容：

```
./start_elibotsim.sh   -m cs616
```

配置完成后，启动虚拟仿真环境，随后启动 Elite_Robots_CS_ROS2_Driver 功能包节点，即可实现 ROS 控制。

# 第 3 章 本地环境安装

在编译和使用 Elite_Robots_CS_ROS2_Driver 软件包之前，请确保系统中已安装 ROS 2、MoveIt 2、Elite SDK 以及 Gazebo。

## 3.1 ROS 2 Humble 安装

以下两种安装方式任选其一即可。

### 3.1.1 官方安装

以下步骤参考 ROS 2 Humble 官方安装指南。

**（1）配置 ROS 2 软件包源：**

在系统终端输入以下内容：

```
sudo apt install software-properties-common

sudo add-apt-repository universe

sudo apt update && sudo apt install curl -y

sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
```

**（2）推荐安装桌面版本（包含 Rviz 等可视化工具和常用教程）：**

```
sudo apt update

sudo apt upgrade

sudo apt install ros-humble-desktop
```

```
6
7 sudo apt install ros-dev-tools        #可额外安装此编译工具包
```

**（3）配置 ROS 2 启动环境：**

```
1 echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc
2
3 source ~/.bashrc
```

### 3.1.2 其他安装方式

如上述方式执行失败，可使用下列替代安装方式：

```
1 wget http://fishros.com/install -O fishros && . fishros
```

按提示操作即可完成安装。

输入以下内容，检测 ROS 2 是否安装成功：

```
1 ros2 --help
```

如终端输出类似以下内容，则说明安装成功：

```
usage: ros2 [-h] [--use-python-default-buffering]
            Call `ros2 <command> -h` for more detailed usage. ...

ros2 is an extensible command-line tool for ROS 2.

options:
  -h, --help            show this help message and exit
  --use-python-default-buffering
                        Do not force line buffering in stdout and instead use
                        the python default buffering, which might be affected
                        by PYTHONUNBUFFERED/-u and depends on whatever stdout
                        is interactive or not
```

**图 3-1:** 安装成功

# 3.2 Moveit 2 环境

## 3.2.1 Moveit 2 概述

MoveIt 2 是 ROS 2 中用于机器人**运动规划、控制与操作**的完整解决方案，为机械臂等机器人提供从路径规划到动作执行的全流程能力，可通过 URDF/SRDF 自动配置。

其主要特性包括：

- **运动规划**（如避障路径规划）
- **逆/正运动学（IK/FK）**
- **碰撞检测**
- **轨迹生成与优化**
- **机器人控制接口（ros2_control）**
- **抓取与操作（grasping）**
- **可视化调试（MoveIt RViz 插件）**

## 3.2.2 Moveit 安装

MoveIt 2 可通过官方 Debian 软件包（via apt）直接安装，安装方法可参考官方二进制安装指南。

**安装命令：**

```
sudo apt update

sudo apt install ros-humble-moveit

sudo apt install ros-humble-moveit-servo

source /opt/ros/humble/setup.bash
```

# 3.3 Gazebo Fortress 环境

## 3.3.1 Gazebo 概述

Gazebo Fortress 是新一代 Gazebo（原 Ignition Gazebo）系列的 LTS（长期支持）版本，

是专为机器人仿真设计的高性能模拟器。相比经典版 Gazebo（Gazebo 11），Fortress 在性能、架构和可扩展性上全面升级，主要提供：

- 更先进的**渲染引擎**（OGRE2 / Vulkan）
- 强大的**物理引擎支持**（ODE、Bullet、DART、Ignition Physics 等）
- **模块化设计**（基于 Ignition Libraries：Physics、Rendering、Transport 等）
- 更流畅的**图形界面与场景编辑工具**
- 更高性能的**传输通信系统**（Ignition Transport）
- **完善的 ROS 2 Humble 集成**（ros_gz_bridge）

### 3.3.2 Gazebo 安装

Gazebo Fortress 的安装方法可参考 官方二进制安装指南。

**（1）安装基础工具**

```
sudo apt-get update

sudo apt-get install lsb-release gnupg
```

**（2）安装 Ignition Fortress**

```
sudo curl https://packages.osrfoundation.org/gazebo.gpg --output /usr/share/keyrings/pkgs-osrf-archive-keyring.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/pkgs-osrf-archive-keyring.gpg] http://packages.osrfoundation.org/gazebo/ubuntu-stable $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/gazebo-stable.list > /dev/null

sudo apt-get update

sudo apt-get install ignition-fortress
```

**（3）安装 ROS 2 集成（ros-gz Bridge）及设置环境变量**

```
sudo apt install -y ros-humble-ros-gz

echo 'source /usr/share/gz/setup.bash' >> ~/.bashrc

source ~/.bashrc
```

**Ubuntu 虚拟机环境配置**

如果您在 **虚拟机**中运行 Gazebo Fortress，可能会遇到 **网格闪烁**或 **空白渲染窗口**的问题，这是由于硬件加速有限导致的。为解决这个问题，可添加以下内容强制启用软件渲染：

```
echo 'export LIBGL_ALWAYS_SOFTWARE=true' >> ~/.bashrc

source ~/.bashrc
```

## 3.4 ELITE SDK 安装

Elite SDK 是艾利特为 CS 和 ES 系列机械臂开发的 C++ 库，借助该库，开发者可以基于 C++ 实现驱动程序，从而充分利用艾利特机械臂的多功能性来开发外部应用程序。在安装 Elite_Robots_CS_ROS2_Driver 之前需要在系统中安装该 SDK 包，推荐使用编译安装方式。如需了解 Elite SDK 包的安装和使用操作，请查阅《C++SDK 快速使用手册》。

## 3.5 ROS2_Driver 安装

**（1）新建工作空间**

在终端输入以下内容：

```
cd

mkdir -p elite_ros_ws/src

cd elite_ros_ws/src
```

**（2）获取相应代码**

```
git clone https://github.com/Elite-Robots/Elite_Robots_CS_ROS2_Driver.
    git
```

**（3）自动安装 Elite ROS2 Drivers 所需依赖**

```
cd ..

sudo rosdep init

rosdep update

rosdep install --ignore-src --rosdistro $ROS_DISTRO --from-paths src -y
```

### （4）编译 Elite ROS2 Drivers

```
colcon build
```

### （5）配置终端环境

构建完成后，需对环境进行设置以使用该软件包：

```
source ~/elite_ros_ws/install/setup.bash
```

如需在每次打开终端时自动加载，可添加：

```
echo "source ~/elite_ros_ws/install/setup.bash" >> ~/.bashrc

source ~/.bashrc
```

至此，ELITE ROS 2 驱动已完成编译与安装，可用于机器人控制与运动规划。

### （6）验证软件包是否安装成功

```
ros2 launch eli_cs_robot_driver elite_control.launch.py robot_ip:=xxx.
   xxx.xxx.xxx local_ip:=yyy.yyy.yyy.yyy cs_type:=zzzz
```

其中：

- robot_ip 填写机器人端 IP 地址（实体机器人填实际机器人端 IP 地址，仿真环境填本地电脑的 IP 地址）；
- local_ip 填写本地电脑的 IP 地址；
- cs_type 填写对应的机器人型号，如 cs66/cs63/cc612 等。

# 第 4 章 入门指南

## 4.1 基础教程：使用 eli_cs_robot_driver 控制 ELITE 机器人

eli_cs_robot_driver 是用于控制 ELITE 机器人的 ROS 2 驱动程序包，通过 ROS 2 的服务（Service）和主题（Topic）与 ELITE 机器人通信。本节将提供示例服务调用及控制器命令发布方式，展示如何向机器人发送基本控制指令并查询其状态。

按照以下步骤启动并连接机器人驱动程序节点：

**（1）配置机器人网络或启动仿真**

请先配置机器人端网络（参见 第 2.1 节），或启动虚拟示教器仿真环境（参见 第 2.2 节）。

**（2）在示教器上打开机器人电源和抱闸，使机器人处于正常模式**

点击「打开电源」> 「释放抱闸」。


图 4-1: 打开电源


图 4-2: 释放抱闸

完成后，示教器应显示为**正常模式**。


图 4-3: 正常模式

### （3）启动 ELITE ROS 2 驱动程序并与机器人建立连接：

```
ros2 launch eli_cs_robot_driver elite_control.launch.py robot_ip:=<
    robot_ip> cs_type:=<cs_type> headless_mode:=true
```

说明：

- <robot_ip> 需替换为机器人实际 IP 地址（如您使用的是虚拟仿真器，则为本机 IP）。
- <cs_type> 需替换为实际机器人型号（如 cs66）。
- headless_mode 表示是否启用外部程序控制模式。

运行成功后，Rviz 中显示的机器人姿态应与示教器一致，如下所示：

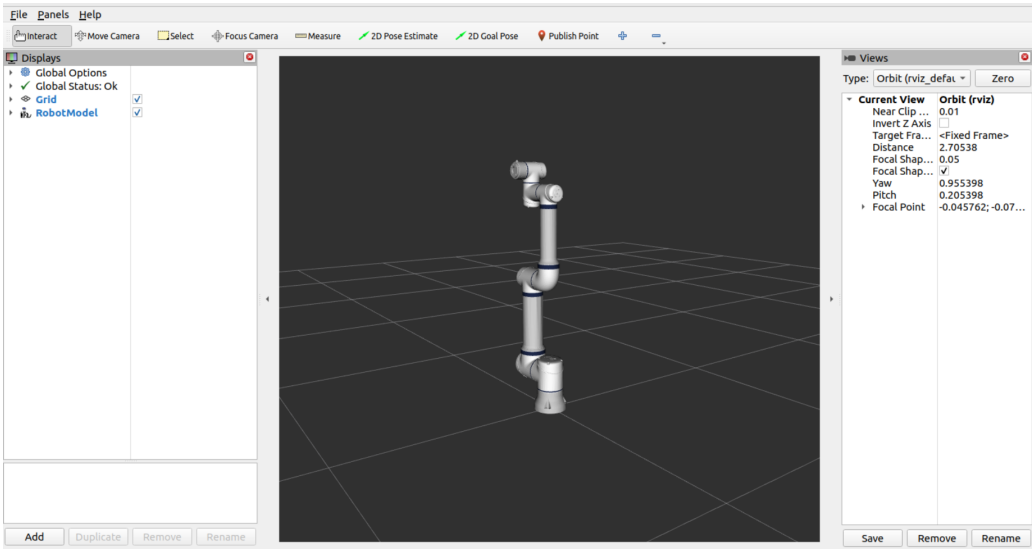



**图 4-4:** Rviz 显示形态

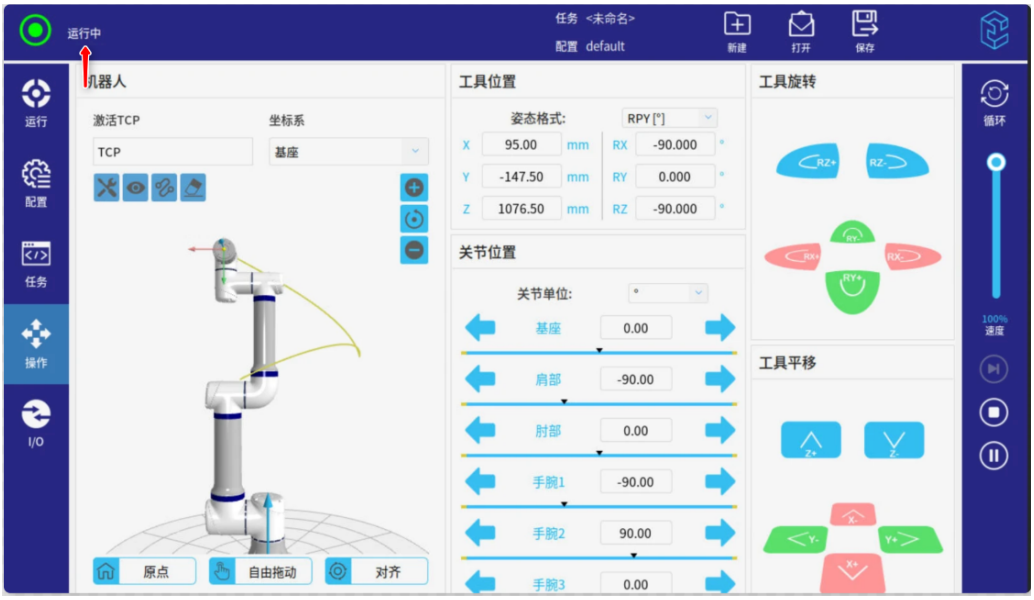



**图 4-5:** 示教器中机器形态

此时示教器左上角状态显示为**运行模式**。

### 4.1.1 示例服务命令

本节展示如何通过 ROS 2 服务接口向 ELITE 机器人发送控制命令。完整的服务列表请参

阅ROS2 接口文档。

**提示**

1. 执行以下命令前，请确保 **elite_ros2_driver 节点正在运行**。
2. 执行这些命令**不需要**提前给机器人上电或释放抱闸。

### （1）打开电源

```
ros2 service call /dashboard_client/power_on std_srvs/srv/Trigger "{}"
```

执行后示教器左上角机器人状态将从关机切换为待机。

```
dukang@dukang-YTF:~/elite_ros_ws$ ros2 service call /dashboard_client/power_on s
td_srvs/srv/Trigger "{}"
requester: making request: std_srvs.srv.Trigger_Request()

response:
std_srvs.srv.Trigger_Response(success=True, message='')
```

**图 4-6:** 转为待机

### （2）释放抱闸

```
ros2 service call /dashboard_client/brake_release std_srvs/srv/Trigger
   "{}"
```

执行后示教器左上角状态变为正常模式。

### （3）获取当前任务的状态

```
ros2 service call /dashboard_client/get_task_status eli_common_interface
   /srv/GetTaskStatus "{}"
```

返回结果如下：

```
dukang@dukang-YTF:~/elite_ros_ws$ ros2 service call /dashboard_client/get_task_s
tatus eli_common_interface/srv/GetTaskStatus "{}"
waiting for service to become available...
requester: making request: eli_common_interface.srv.GetTaskStatus_Request()

response:
eli_common_interface.srv.GetTaskStatus_Response(success=True, message='', status
=eli_common_interface.msg.TaskStatus(status=3))
```

**图 4-7:** 返回结果

### 4.1.2 示例控制器命令

ROS 2 control 是 ROS 中用于向机器人发送控制命令（关节、速度、轨迹）并管理硬件的核心组件。eli_cs_controllers 包提供了适用于 Elite CS 和 ES 系列机器人的 ros2_control 专用控制器与硬件接口，包括：

- **speed_scaling_interface**：读取当前速度缩放值并传递给控制器。
- **scaled_joint_command_interface**：结合速度缩放值提供对关节值和命令的访问。
- **speed_scaling_state_controller**：将机器人报告的当前执行速度（0-1 浮动值）发布到主题接口。
- **scaled_joint_trajectory_controller**：与 joint_trajectory_controller 类似，但它使用机器人报告的速度缩放值来减少轨迹的执行进度。

**提示**

执行下述机器人命令前，须先确保 **elite_ros2_driver 节点正在运行，且机器人处于运行模式**。

**（1）列出当前可用控制器**

```
ros2 control list_controllers
```

返回结果如下：

```
dukang@dukang-YTF:~/elite_ros_ws$ ros2 control list_controllers
force_torque_sensor_broadcaster    force_torque_sensor_broadcaster/ForceTorqueSe
nsorBroadcaster  active
joint_state_broadcaster            joint_state_broadcaster/JointStateBroadcaster
                 active
forward_position_controller        position_controllers/JointGroupPositionContro
ller             inactive
scaled_joint_trajectory_controller eli_cs_controllers/ScaledJointTrajectoryContr
oller            inactive
speed_scaling_state_broadcaster    eli_cs_controllers/SpeedScalingStateBroadcast
er               active
tcp_pose_broadcaster               pose_broadcaster/PoseBroadcaster
                 active
freedrive_controller               eli_cs_controllers/FreedriveController
                 inactive
io_and_status_controller           eli_cs_controllers/GPIOController
                 active
```

**图 4-8:** 返回结果

### （2）关节轨迹控制

示例如下：

```
ros2 topic pub /scaled_joint_trajectory_controller/joint_trajectory
   trajectory_msgs/msg/JointTrajectory "
joint_names: [
  'shoulder_pan_joint',
  'shoulder_lift_joint',
  'elbow_joint',
  'wrist_1_joint',
  'wrist_2_joint',
  'wrist_3_joint'
]
points:
- positions: [0.0, -1.0, 1.0, -1.5, 1.5, 0.0]
  time_from_start: {sec: 2}
  "
```

或使用：

```
- ros2 action send_goal /scaled_joint_trajectory_controller/
   follow_joint_trajectory \
    control_msgs/action/FollowJointTrajectory "{
      trajectory: {
        joint_names: [shoulder_pan_joint, shoulder_lift_joint,
   elbow_joint, wrist_1_joint, wrist_2_joint, wrist_3_joint],
        points: [
          {positions: [0,-1.57,1,-1.5,1.5,0.0], time_from_start: {sec:
   2}}
        ]
      }
    }"
```

成功执行后机器人将到达指定关节位置：


图 4-9: 返回结果

### 4.1.3 程序控制示例

**提示**

1. 执行下述命令前，请先确保 **elite_ros2_driver 节点正在运行且机器人处于运行模式**。
2. 运行下列命令前，须确保机器人周围环境安全。

Python 示例：

```
ros2 run eli_cs_robot_driver example_safe_joint_move.launch.py --target 0 -1.2 0 -1.57 0 0 --max-vel 0.5 --min-time 2.0 --controller scaled_joint_trajectory_controller
```

具体代码和参数使用方法参见 Python 源代码。

C++ 示例：

```
ros2 run eli_cs_robot_driver example_safe_joint_move.cpp --ros-args -p target:="[0.0,-1.57,0.0,-1.0,-1.57,0.0]" -p max_vel:=0.5 -p min_time:=2.0 -p controller:="scaled_joint_trajectory_controller"
```

具体代码和参数使用方法参见 C++ 源代码。

**参数说明**

- target：目标关节角度。若角度为整数，请使用浮点数表示，例如用 1.0 表示整数 1。
- max_vel：最大关节速度比例。该值越大，关节运动速度越快。
- min_time：最小运动时间。
- **运行逻辑**：在执行运动时，系统会优先采用较大的时间值。例如，当最大关节速度计算出的运动时间与最小运动时间不同，系统将选择两者中的较大值作为实际运动时间。

## 4.2 Moveit 2 教程：规划与执行

本节示例展示如何基于 Moveit 2 对 ELITE 机器人（以 CS66 为例）进行运动规划与路径执行。

**提示**

执行下述命令前，请先确保示教器中机器人处于**正常模式**。

**（1）启动机器人驱动和控制器程序：**

```
ros2 launch eli_cs_robot_driver elite_control.launch.py robot_ip:=<
   robot_ip> headless_mode:=true launch_rviz:=false cs_type:=cs66
```

<robot_ip> 需替换为实际的机器人 IP 地址（如您使用的是虚拟仿真器，则为本地电脑的 IP 地址）。

**（2）启动 Moveit2 控制程序：**

```
ros2 launch eli_cs_robot_moveit_config cs_moveit.launch.py cs_type:=cs66
```

启动完成后，Rviz 中的机器人姿态应与示教器中的状态一致。如不一致，可重启机器人驱动节点与 Moveit 2 控制节点。

**（3）生成规划轨迹并执行**

在 Rviz 中使用交互标记（机器人末端执行器球体）拖动至目标位置：

- 点击「Plan」生成机器人的规划轨迹；

- 或点击「Plan & Execute」直接规划并执行。

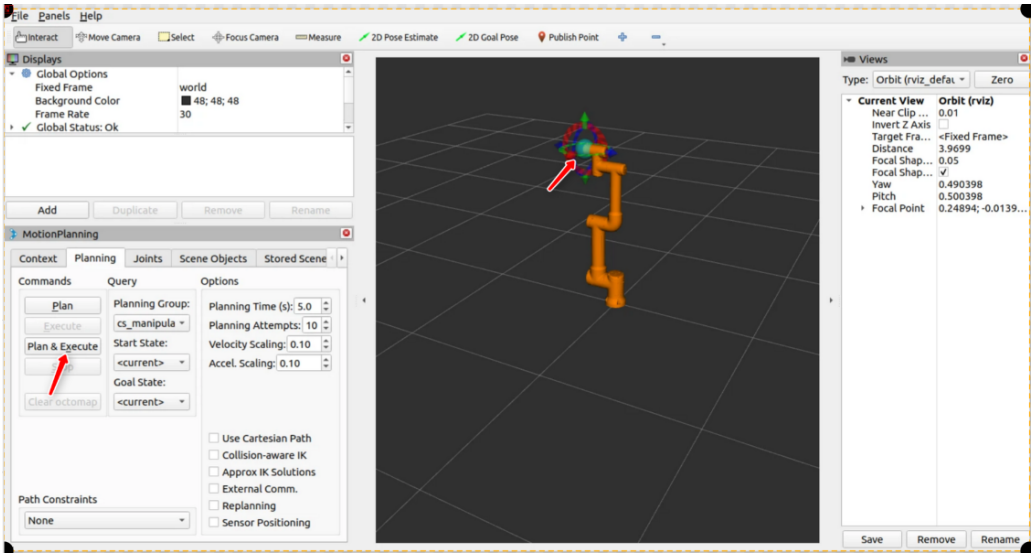



**图 4-10:** 生成轨迹

执行规划后，机器人示教器端将实时同步显示运动过程。

如需了解 Moveit 2 的完整配置方法与编程控制示例，可参阅ELITE Robots ROS2 教程。

## 4.3 Gazebo 教程：物理仿真世界

Gazebo 可模拟机器人的物理特性，并允许添加障碍物测试运动规划、碰撞检测与避障算法。本节展示如何在 Gazebo 中启动 ELITE 机器人并实现程序控制。

**提示**

启动 Gazebo 仿真前，请确保 **已关闭其他机器人端连接**。

### 1. 通过程序控制 Gazebo 机器人

**（1）启动 Gazebo 环境：**

```
ros2 launch eli_cs_robot_simulation_gz cs_sim_control.launch.py
```

启动后 Gazebo 将加载机器人模型，显示效果如下：

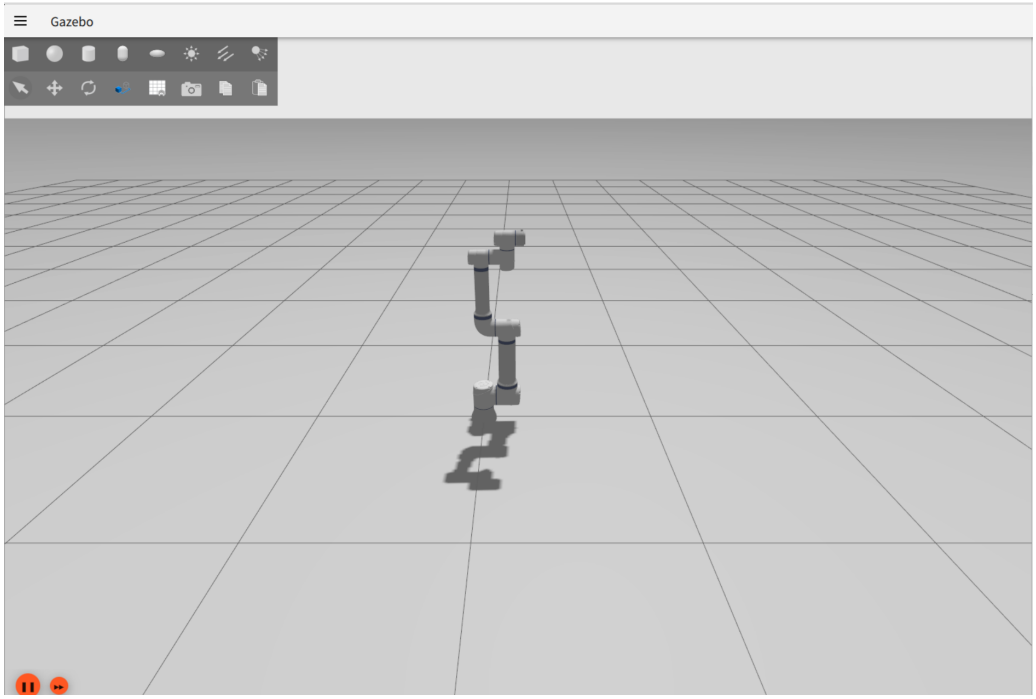



**图 4-11:** 启动结果

**（2）控制 Gazebo 中的机器人运动：**

```
ros2 run eli_cs_robot_simulation_gz gz_control.py --example
```

更多程序控制方案可参考 该脚本源代码。

## 2. Moveit 2 控制 Gazebo 机器人

**（1）同时启动 Moveit 2 和 Gazebo 环境：**

```
ros2 launch eli_cs_robot_simulation_gz cs_sim_moveit.launch.py
```

**（2）执行运动规划**

可参考第 4.2 节的步骤 3（同样适用 Gazebo ）实现控制。

ELITE ROBOTS
艾利特机器人

# 明天比今天更简单一点

## 联系我们

商务合作：market@elibot.cn
技术咨询：technical@elibot.cn

- **苏州公司（生产基地）**
  苏州市工业园区长阳街 259 号中新钟园工业坊 4 栋
  +86-400-189-9358
- **北京公司**
  北京市经济技术开发区荣华南路 10 号院 5 号楼 611 室
- **上海公司（研创中心）**
  上海市浦东新区张江人工智能岛川和路 55 弄 20 号楼 3 层
- **深圳公司**
  深圳市宝安区航空路泰华梧桐岛科技创新园 1A 栋 202 室
- **美国公司**
  10521 Research Dr., Ste. 104, 37932, Knoxville, TN (USA)
- **德国公司**
  Münchener Str. 53, 85290, Geisenfeld, Bavaria (Germany)
- **日本公司**
  TOSHIN Hirokoji Honmachi Bldg., 1F, 2-4-3 Sakae, Naka-ku, 460-0008, Nagoya (Japan)
- **墨西哥公司**
  Calzada del pedregal 523, fraccionamiento el pedregal

关注公众号了解更多