
Dora小车室内定位导航+机械臂抓取 使用手册

版本记录和版本号变动与修订记录

版本	更改描述	创建/更改日期	创建/更改人
1.0	车辆部分初始版本	2025-4-23	张旭东
2.0	机械臂及大模型部分初始版本	2025-4-24	袁典

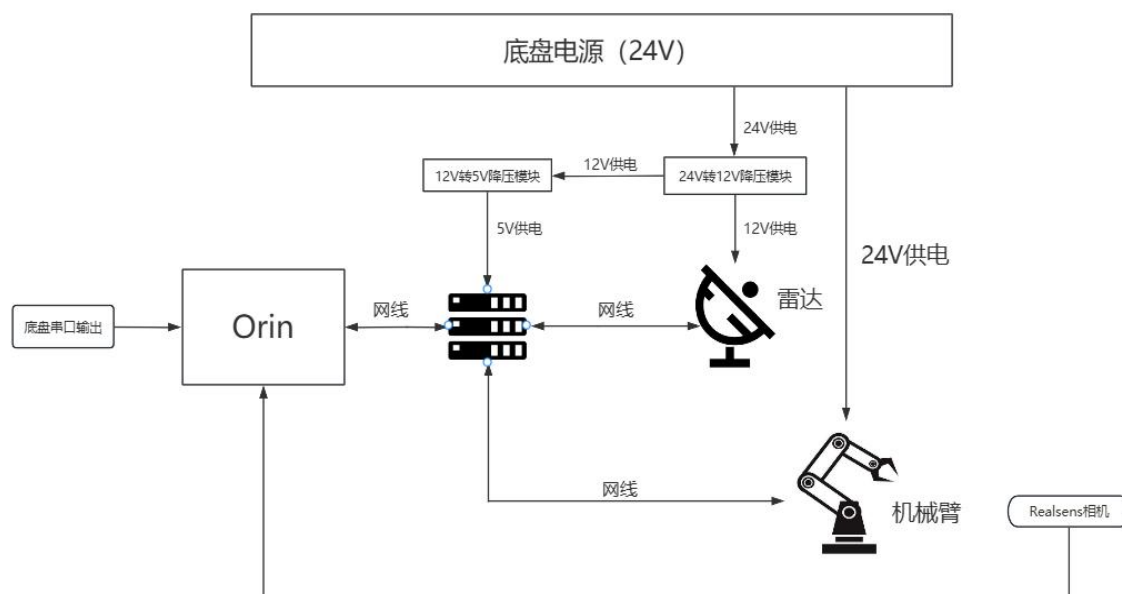
目录

版本记录和版本号变动与修订记录.....	2
目录.....	3
1.硬件连接说明.....	5
1.1总体接线图.....	5
1.2小车遥控器.....	5
1.3车辆接口说明.....	6
1.4orin连接线	7
2.车辆部分.....	8
2.1系统介绍.....	8
2.2运行环境及其部署.....	8
2.2.1 运行环境.....	8
2.2.2 环境安装.....	8
2.3使用本系统流程.....	10
2.3.1 录制 rosbag 包	10
2.3.2slam 建图	12
2.3.3 录制线路	13
2.3.4 绘制线路	13
2.3.5 运行系统.....	13
3.机械臂部分.....	15
3.1系统介绍.....	15
3.2运行环境及其部署.....	15
3.2.1 运行环境.....	15
3.2.2 环境安装.....	16
3.3边缘服务器部署.....	16
3.3.1whisper 部署	16
3.3.1qwen 部署	17
3.4具体流程.....	18
3.4.1 机械臂启动	18
3.4.2 语音节点启动	19

3.5测试节点说明.....	19
3.5.1 小车-机械臂通信.....	19
3.5.2 边缘服务器-机械臂通信	19

1.硬件连接说明

1.1总体接线图



1.2小车遥控器

- 1.遥控器左上角的拨动开关为功能选择按键：置于L档，即最上方位置，表示不使能遥控器，为自动驾驶模式。置于H档表示由遥控器控制小车，忽略上位机命令。
- 2.遥控器右上角拨动开关上中下位置分别对应小车1m/s、2m/s、3.5m/s速度。
- 3.左边摇杆竖直方向通道控制小车前后运动（如下图所示），右边的摇杆水平通道控制小车左右旋转。



图1.1 遥控器使用说明

1.3 车辆接口说明

状态灯常亮绿色为正常，若是黄色闪亮，表示未连接到遥控器。若是红色闪亮，表示未打开急停按钮。下图为车辆接口说明

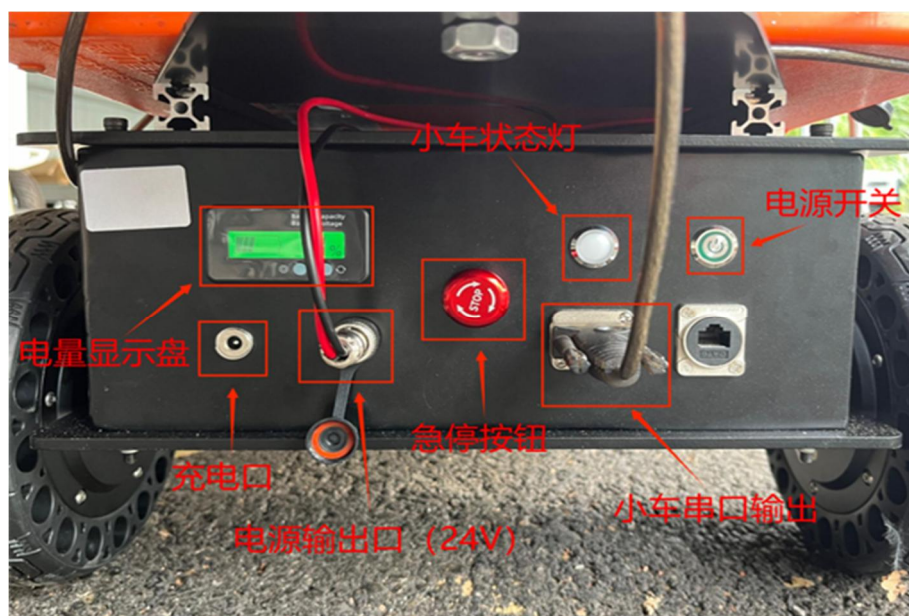


图1.2 车辆接口说明

1.4 orin连接线

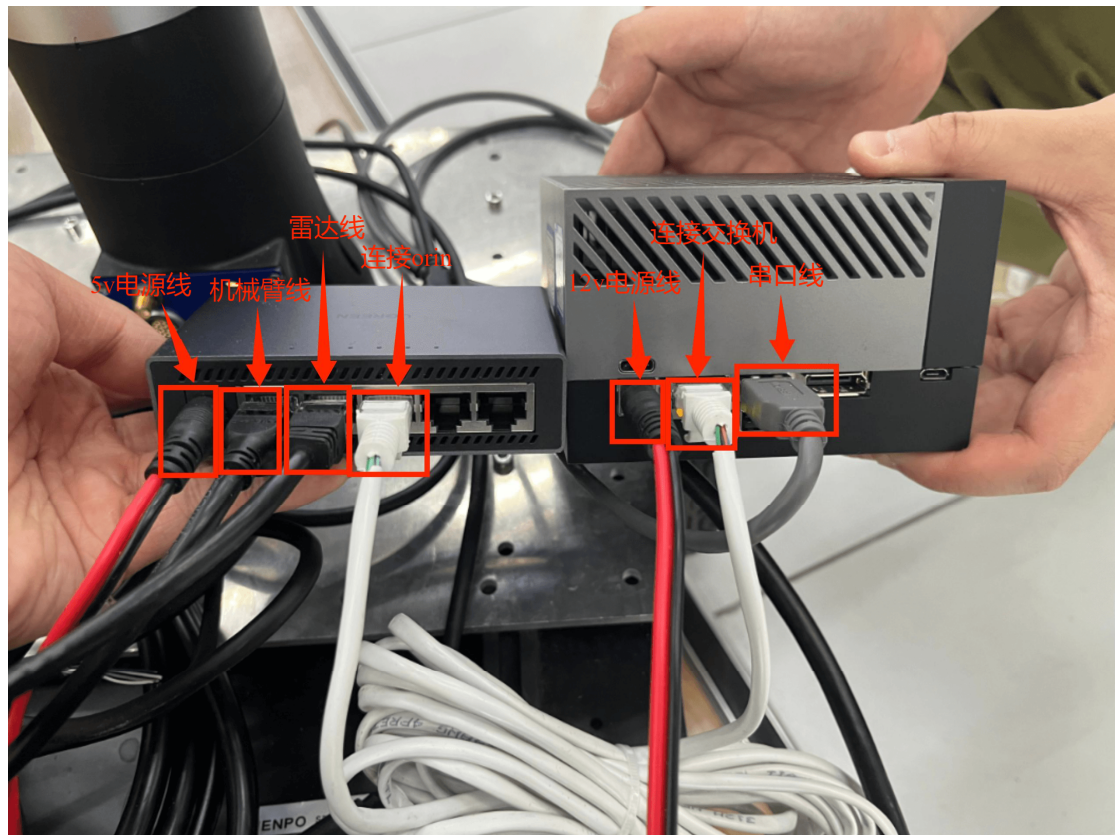


图1.3 orin接线图

2.车辆部分

2.1系统介绍

本系统是基于dora的室内定位导航，想要运行本系统需要用ROS提前建好的PCD点云地图和全局路线。图1为本系统的代码文件图。

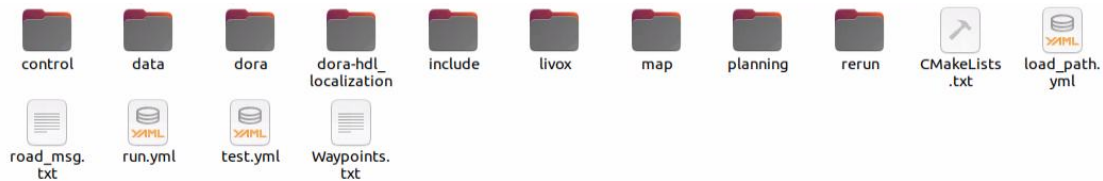


图1 代码文件

运行本系统的大致流程为：通过ROS建好点云地图，将建好的点云地图改名为map.pcd并且放入data目录中。启动load_path.yml，遥控车辆走预定好的路线。走过的路线（x、y坐标）会记录在/data/trajectory.txt中，将路线导入arcgis中规整成间隔为1m的点。将规整好的点复制到Waypoints.txt中。运行run.yml就完整运行本系统。

2.2运行环境及其部署

2.2.1运行环境

所需环境：Ubuntu20.04、dora-rs（0.3.8）、ROS1-Noetic、Rerun_cpp_sdk、serial、livox_sdk

2.2.2环境安装

（1）dora-rs安装

```
git clone https://github.com/dora-rs/dora.git
git checkout tags/v0.3.8
cd dora
cargo build -p dora-cli --release
```

编译完成后将/target/release下的dora放入/bin目录，测试dora
-V如果输出dora-cli 0.3.8则安装成功，下面编译dora的C节点

```
cd ../examples/c++-dataflow
cargo run --example cxx-dataflow # compile C++ node
cargo build -p dora-node-api-c --release # compile dora-node-api-c
```

(2) ROS1安装

使用小鱼一键安装：打开终端输入`wget http://fishros.com/install -O fishros`
&& `. fishros`，按照提示安装ROS1-Noetic。

(3) slam环境安装

这里选择hdl_grapg_slam算法进行建图。

```
sudo apt-get install ros-noetic-geodesy ros-noetic-pcl-ros ros-noetic-nmea-msgs
ros-noetic-libg2o
mkdir -p catkin_ws/src
cd catkin_ws/src
git clone https://github.com/koide3/ndt_omp.git
git clone https://github.com/SMRT-AIST/fast_gicp.git --recursive
git clone https://github.com/koide3/hdl_graph_slam
cd .. && catkin_make -DCMAKE_BUILD_TYPE=Release
```

(4) 雷达驱动安装

```
git clone https://github.com/Livox-SDK/Livox-SDK2.git
cd ./Livox-SDK2/
mkdir build
cd build
cmake .. && make -j6
sudo make install
```

(5) serial库安装(使用ROS环境)

```
git clone https://github.com/wjwwood/serial.git
mkdir build && cd build
cmake ..
make -j12
sudo make install
```

(6) rerun_sdk安装

详情可见: <https://blog.csdn.net/candygua/article/details/146050143?spm=1001.2014.3001.5502>

编译Rerun仓库

```
git clone https://github.com/rerun-io/rerun.git
cd rerun
cargo install rerun-cli --locked
```

测试rerun

--help如果有输出, 则代表rerun安装完成。下面安装rerun_cpp_sdk。

<https://github.com/rerun-io/rerun/releases/tag/0.22.1>在这个目录下下载rerun_cpp_sdk.zip。

```
cd rerun_cpp_sdk
mkdir build && cd build
cmake ..
make -j6
sudo make install
```

2.3使用本系统流程

2.3.1录制rosbag包

修改ip地址如下图

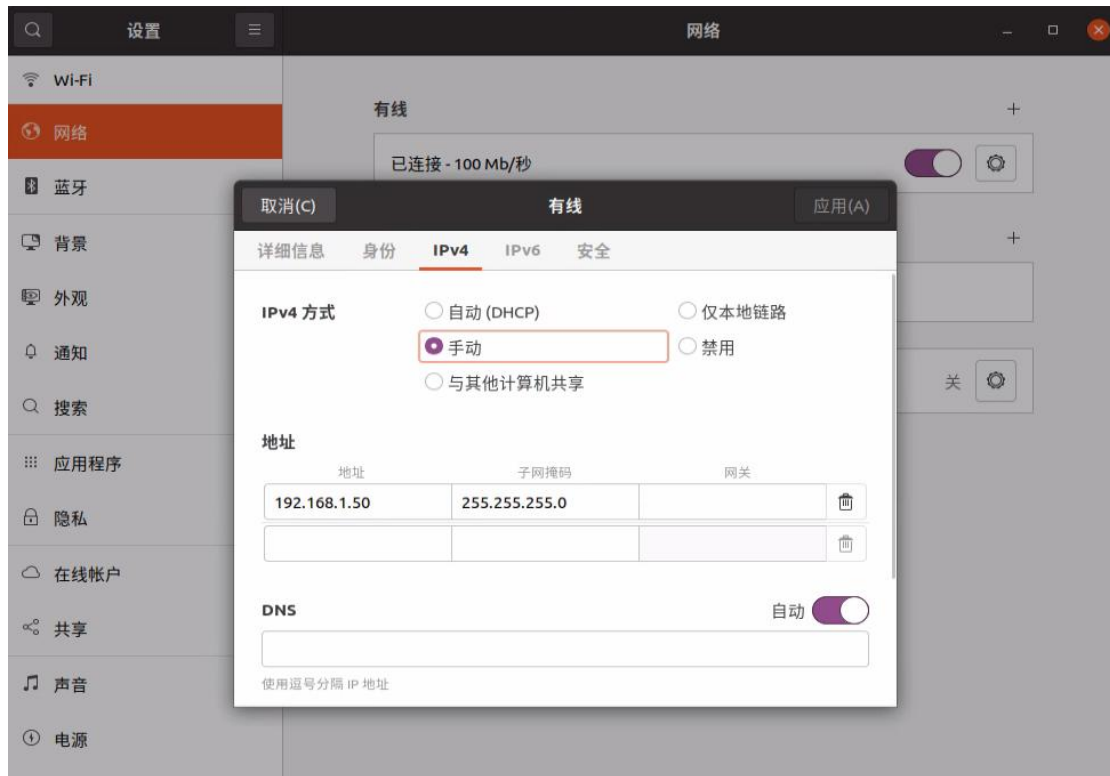


图2 修改本机ip

测试，ping 192.168.1.113

```
mkdir -p livox_driver/src
cd livox_driver/src
git clone https://github.com/Livox-SDK/livox_ros_driver2.git
cd livox_ros_driver2
source /opt/ros/noetic/setup.sh
./build.sh ROS1
source ../../devel/setup.sh
roslaunch livox_ros_driver2 rviz_MID360.launch
```

alt+t另开一个终端

```
source ./devel/setup.bash
rosbag record -o livoxbag.bag /livox/lidar
```

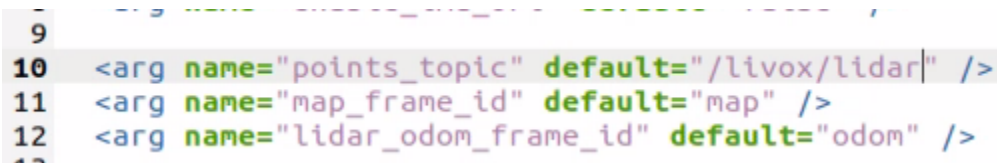
将车辆绕场景一圈，录制完之后ctrl+c杀掉终端。

2.3.2slam建图

进入hdl_graph_slam文件夹

```
cd launch  
gedit hdl_graph_slam_501.launch
```

修改话题/points_topic为/livox/lidar。如图3。



```
9  
10 <arg name="points_topic" default="/livox/lidar" />  
11 <arg name="map_frame_id" default="map" />  
12 <arg name="lidar_odom_frame_id" default="odom" />  
13
```

图3. 修改雷达点云话题

回到hdl_graph_slam主目录，有build和devel那个目录

```
source ./devel/setup.bash  
rosparam set use_sim_time true  
roslaunch hdl_graph_slam hdl_graph_slam_501.launch
```

找到刚刚录制的雷达点云包

```
rosbag play --clock livoxbag.bag
```

当点云包播放完之后，回到hdl_graph_slam主目录，再起一个终端

```
source ./devel/setup.bash  
rosservice call /hdl_graph_slam/save_map "utm: false  
resolution: 0.01  
destination: '/home/nvidia/map.pcd'" #路径可自定义
```

2.3.3录制线路

```
https://github.com/ZcanMelo/slam-navigation.git  
cd build && cmake ..  
make -j12
```

将刚刚建好的图放入/data目录。

启动线路录制yml

```
dora up  
dora start load_path.yml
```

按着预定路线走一圈之后，杀掉终端。路线点记录在/data/path/trajectory.txt中。

2.3.4绘制线路

本部分需要在windows系统中完成，需要安装Arcgis软件和postgresql数据库。

将点导入Arcgis中将路径点规整成间隔为1m的有序点。

将绘制好的点复制到Waypoints.txt中，注意将文件中的tab更换为空格。

2.3.5运行系统

将dora的头文件和库放入dora文件夹。

其中include放入node_api.h, operator_api.h, operator_types.h; （这三个头文件位于dora仓库中的apis/c/node和apis/c/operator）

lib中放入libdora_node_api_c.a（这个库位于dora仓库的target/release）

完成上述步骤之后就可以开始运行本系统

```
sudo chmod +777 /dev/ttyUSB0
```

```
dora up
```

```
dora start run.yml
```

Rerun会自动弹出，白色的线为全局路线图，蓝色的为规划线。

3.机械臂部分

3.1系统介绍

本系统是基于ROS1，Qwen的机械臂识别+抓取，运行前需提前在边缘服务器上部署相关模型，用作语音识别以及推理，其中语音识别部分为whisper，推理部分为Qwen2.5，图一为机械臂控制部分代码文件；图二为服务器运行代码文件。

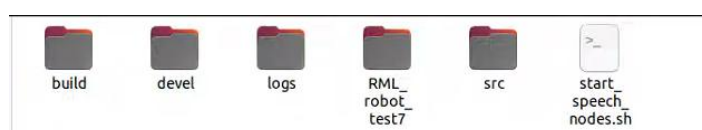


图1 机械臂控制代码文件



图2 服务器运行代码文件

运行大致流程：将机械臂安装好固定位置，同orin连接网线，按第一部分介绍更改好IP，运行命令行启动语音节点：

```
./start_speech_nodes.sh
```

和运行命令行启动机械臂控制节点：

```
roslaunch robot_ctrl robot.launch
```

即可对麦克风说话，例如“帮我拿个水瓶”、“帮我拿个鼠标”等指令，等待小车运行到指定地点，完成抓取。

3.2运行环境及其部署

3.2.1运行环境

所需环境：Ubuntu20.04、ROS1-Noetic、RM_ARM V1.7.0、API1

V4.3.3（代码文件中已替换）、Qwen 2.5VL、Whisper Model(base)。

3.2.2环境安装

(1) 机械臂控制器版本

登陆睿尔曼示教器界面192.168.1.18查看，点进机械臂配置/版本信息，如下图所示。



图3 控制器版本

若版本不同，详见文件夹所带工具包，在示教器页面选择文件，先选择工具包中RMFirmwareUpdate文件上传后选择升级，完成后重启机械臂，再进入示教器页面点击tool.bin文件选择升级，完成后重启机械臂。

3.3边缘服务器部署

3.3.1whisper部署

将whisper文件夹迁移至边缘服务器后，进入代码文件中的whisper目录下，将whisper项目下载至本地：

```
git clone https://github.com/ggerganov/whisper.cpp
```

whisper.cpp

项目里提供了几个现成的模型。建议下载small以上的模型，如若边缘服务器性能不够可使用base模型。

进入whisper.cpp目录下打开终端，使用下述命令下载指定模型，下载好之后，模型存储在models文件夹下

```
./models/download-ggml-model.sh small
```

项目根目录执行 make 即可编译，得到main可执行文件

```
make -j12
```

3.3.1qwen部署

创建虚拟环境，python版本要求：3.11，可执行如下示例命令行：

```
conda create --name qwen python=3.11  
conda activate qwen
```

完成上述步骤后，代码文件whisper中自带的qwen72.py和requirements.txt两个文件移到whisper.cpp文件夹下，在whisper.cpp目录下打开终端，执行命令行：

```
pip install -r requirements.txt
```

完成依赖安装后执行命令行：

```
python qwen72.py
```

第一次执行会将代码所指定大模型自动部署在`~/.cache/modelscope/models/`目录下，模型下载完毕后，执行成功界面如下，完成后可启动小车-机械臂控制部分等待连接，详情见下节。

```
(dora) cheku@cheku:~/chatglm/whisper_cpp/whisper.cpp$ python qwen72.py
Loading checkpoint shards: 100% |██████████| 5/5 [00:00<00:00, 36.02it/s]
等待客户端连接...
```

图4 终端界面

3.4具体流程

3.4.1机械臂启动

在robot_gen72目录下启动终端，执行命令行catkin make，在终端中执行

⋮

```
source ./devel/setup.bash
```

启动机械臂节点:

```
roslaunch robot_ctrl robot.launch
```

机械臂初始位置设定：对机械臂按住物理绿色按键将机械臂移动到合适位置，在示教器主页面下读取当前机械臂当前关节角度信息，如图4所示。

示教	位姿编辑	拖动示教
关节1(°)	⊖	33.53 ⊕
关节2(°)	⊖	28.828 ⊕
关节3(°)	⊖	56.155 ⊕
关节4(°)	⊖	-87.305 ⊕
关节5(°)	⊖	-58.564 ⊕
关节6(°)	⊖	4.443 ⊕
关节7(°)	⊖	48.273 ⊕

图5 控制器版本

在robot_gen72/src/robot_ctrl/src/Rml_robot.cpp中，更改初始化角度float joint_base[7]。

3.4.2语音节点启动

在robot_gen72目录下启动终端，输入命令行./start_speech_nodes.sh启动三个语音节点，即offline_interactive_ros.launch离线语法构建节点，vad_record.py音频文件记录节点，test.py服务器通信节点，具体流程即为通过麦克风输入语音，记录为音频文件，通过通信节点将音频文件传输给服务器，服务器接收后将其转换为文本并转发给大模型，最后大模型输出返回给机械臂控制处理。(语音节点日志信息可根据本机时间戳在robot_gen72/logs目录下查看三个节点具体日志信息)

3.5测试节点说明

3.5.1小车-机械臂通信

若想单独测试小车自驾逻辑，可进入/robot_gen72/RML_robot_test下，打开终端执行如下命令行：

```
./udp_test
```

出现如下界面后，输入start小车自动前进到停止地点后，输入finish，小车继续前进并返回至终点。



```
nvidia@nvidia-desktop:~/SSD/IndoorSLAM/robot_gen72/RML_robot_test7$ ./udp_test
Enter command (start, finish, exit):
```

图6 终端界面

3.5.2边缘服务器-机械臂通信

若想代替边缘服务器同机械臂通信完成测试抓取-自驾整体逻辑可进入robot_gen72/src/handsfree_speech/script/robot目录下，打开终端执行：

```
python send.py
```

其逻辑为代替服务器发送固定文本信息，机械臂接收到完成初始化并开启

摄像头并发送信息给小车，小车前进至停车点，机械臂识别物体后开始抓取，抓取完毕后小车继续前进并返回至终点。