1.实验名称及目的

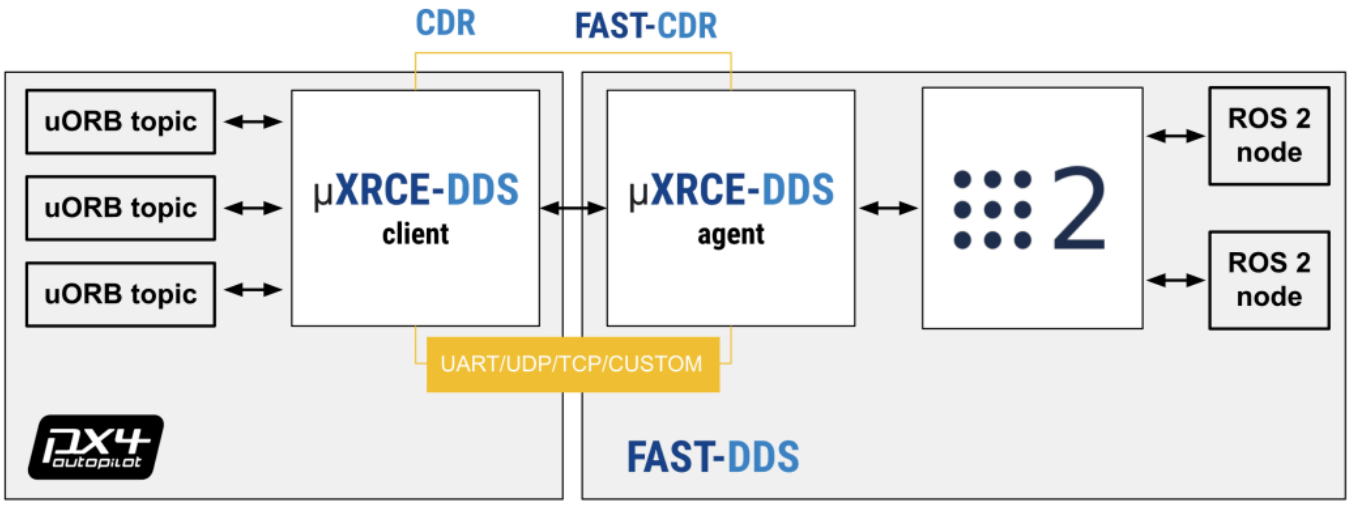
1.1实验名称

PX4+ROS2的uXRCE-DDS串口通信及控制实验

1.2实验目的

- 掌握PX4飞控与ROS2系统之间基于uXRCE-DDS的串口通信方法
- 了解PX4与ROS2之间的数据交互方式和消息类型
- 实践基于ROS2的无人机offboard控制功能
- 熟悉PX4飞控在硬件在环(HIL)仿真环境下的调试方法

1.3关键知识点



- PX4 v1.14及以上：最新版LTS引入更强任务航线引擎、统一参数树与丰富机型预设，性能与可维护性双升级。
- ROS2：基于DDS的分布式实时通信架构，让感知、规划、控制节点即插即用。
- XRCE-DDS：仅75KB的轻量协议栈，实现飞控 MCU 与 ROS 2 话题的毫秒级互联。

1.3.1 PX4与ROS2通信架构

- DDS (Data Distribution Service) 是一种发布/订阅协议，用于实时系统的分布式通信
- Micro XRCE-DDS 是DDS协议的轻量级实现，专门用于资源受限的嵌入式系统
- PX4使用uORB消息系统内部通信，通过Micro XRCE-DDS Bridge转换为ROS2消息

1.3.2 关键示例程序分析

sensor_combined_listener.cpp详细分析

该程序实现了订阅PX4 IMU传感器数据的功能：

1. 程序结构：

```
class SensorCombinedListener : public rclcpp::Node {
    rclcpp::Subscription<px4_msgs::msg::SensorCombined>::SharedPtr subscription_;

    SensorCombinedListener() : Node("sensor_combined_listener") {
        subscription_ = this->create_subscription<px4_msgs::msg::SensorCombined>(
            "fmu/out/sensor_combined",    // 话题名
            10,                           // QoS设置
            std::bind(&SensorCombinedListener::topic_callback,
                    this, std::placeholders::_1));
    }
};
```

2. 关键函数分析：

```
void topic_callback(const px4_msgs::msg::SensorCombined::SharedPtr msg)
{
    // 获取时间戳(微秒)
    const auto timestamp = msg->timestamp;
    // 获取加速度数据(m/s^2)
    const auto acc_x = msg->accelerometer_m_s2[0];
    const auto acc_y = msg->accelerometer_m_s2[1];
    const auto acc_z = msg->accelerometer_m_s2[2];
    // 获取陀螺仪数据(rad/s)
    const auto gyro_x = msg->gyro_rad[0];
    const auto gyro_y = msg->gyro_rad[1];
    const auto gyro_z = msg->gyro_rad[2];
}
```

3. 消息类型说明：

- SensorCombined消息包含IMU原始数据
- accelerometer_m_s2：三轴加速度，单位m/s²
- gyro_rad：三轴角速度，单位rad/s
- timestamp：数据时间戳，微秒级

offboard_control.cpp详细分析

该程序实现了无人机的离线控制功能：

1. 关键变量定义：

```
class OffboardControl : public rclcpp::Node {
    rclcpp::Publisher<OffboardControlMode>::SharedPtr offboard_control_mode_publisher_;
    rclcpp::Publisher<TrajectorySetpoint>::SharedPtr trajectory_setpoint_publisher_;
    rclcpp::Publisher<VehicleCommand>::SharedPtr vehicle_command_publisher_;
    rclcpp::TimerBase::SharedPtr timer_;
};
```

2. 控制流程详解：

```
void timer_callback()
{
    // 1. 发布offboard控制模式
    publish_offboard_control_mode();  // 设置位置控制模式

    // 2. 发布轨迹设定点
    publish_trajectory_setpoint();     // 设置目标位置(0, 0, -5)

    // 3. 发布解锁命令
    publish_vehicle_command(VehicleCommand::VEHICLE_CMD_COMPONENT_ARM_DISARM, 1.0);

    // 4. 切换到offboard模式
    publish_vehicle_command(VehicleCommand::VEHICLE_CMD_DO_SET_MODE, 1.0);
}
```

3. 具体控制指令说明：

- OffboardControlMode消息：设置控制模式（位置/速度/姿态等）

```
auto msg = OffboardControlMode();
msg.position = true;         // 启用位置控制
msg.velocity = false;        // 禁用速度控制
msg.acceleration = false;    // 禁用加速度控制
```

- TrajectorySetpoint消息：设置目标轨迹点

```
auto msg = TrajectorySetpoint();
msg.position = {0.0, 0.0, -5.0};  // 目标位置(x,y,z)
msg.yaw = 0.0;                    // 目标偏航角
```

- VehicleCommand消息：发送飞控指令

```
auto msg = VehicleCommand();
msg.command = command;             // 指令类型
msg.param1 = param1;               // 指令参数
msg.target_system = 1;             // 目标系统ID
msg.target_component = 1;          // 目标组件ID
msg.timestamp = this->get_clock()->now().nanoseconds() / 1000;  // 时间戳
```

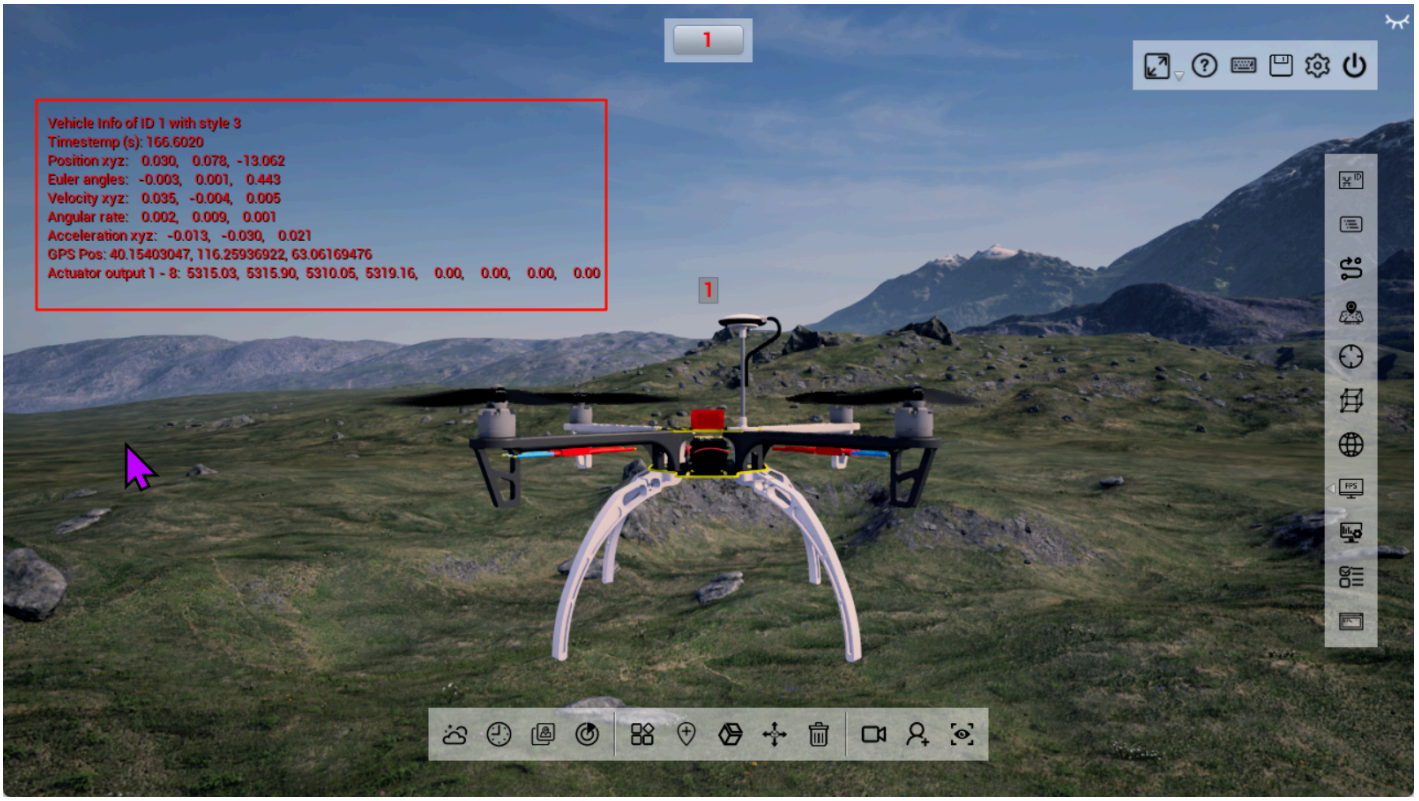
4. 通信频率：

- 控制指令以10Hz频率发送
- 位置控制精度约±0.1m
- 保持稳定offboard控制需持续发送心跳包

2.实验效果

本实验通过以下步骤成功实现了PX4与ROS2的通信控制:

1. 完成了PX4飞控与机载计算机之间基于uXRCE-DDS的串口通信配置；
2. 实现了ROS2节点订阅飞控IMU传感器数据,验证了数据传输的正确性；
3. 通过ROS2的offboard_control节点成功控制无人机完成:
    - 解锁并切换到offboard模式
    - 垂直起飞上升约5米
    - 保持稳定悬停



3.文件目录

此处编写例程目录。
例程目录：[安装目录]\RflySimAPIs\6.RflySimExtCtrl\0.ApiExps\e19_uXRCE-DDS_ROS2CtrlExps\2.SerPortComm

4.运行环境

此处编写本实验的，运行环境要求，可参考如下设置：

4.1 软件要求

- Windows 10及以上版本；
- RflySim工具链;

4.2 硬件要求

笔记本/台式电脑1台

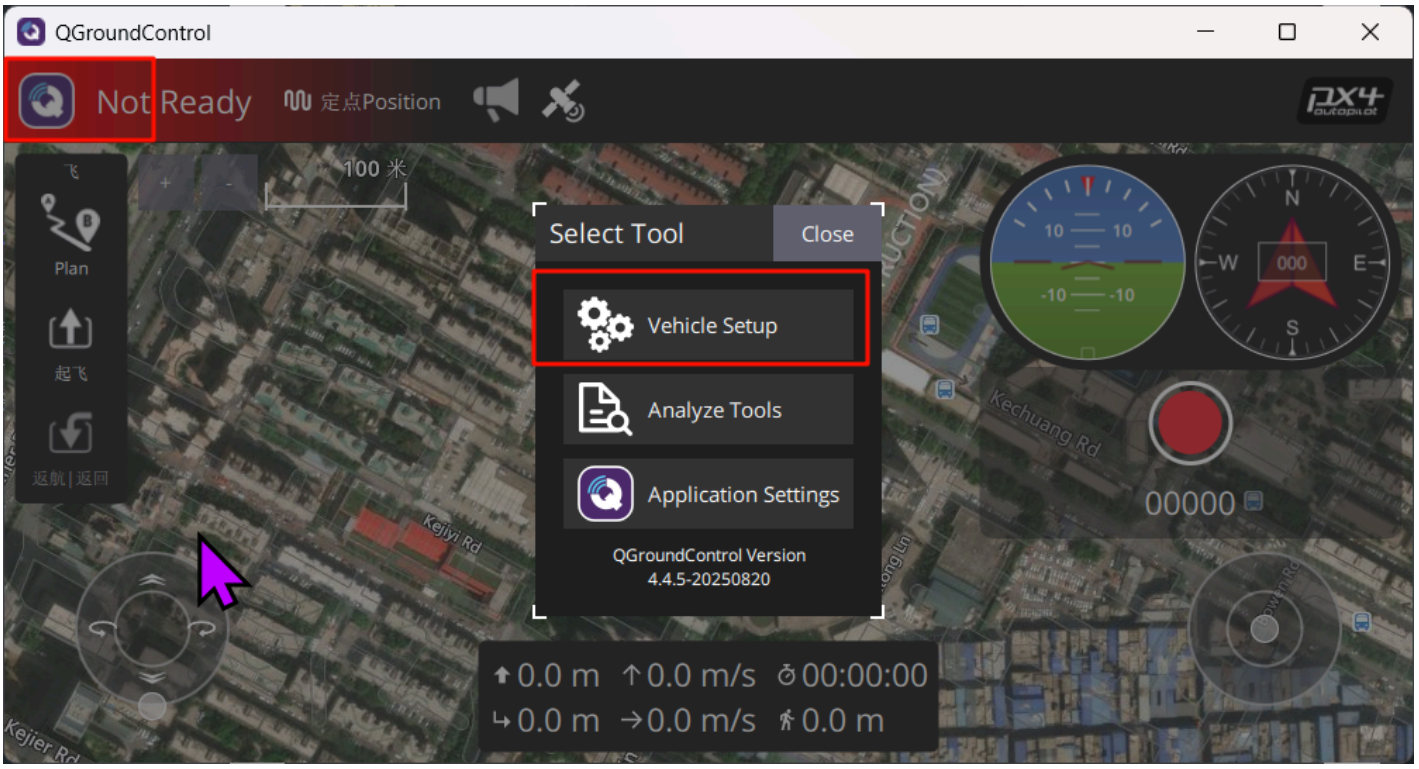
> ①：推荐配置请见：https://rflysim.com/

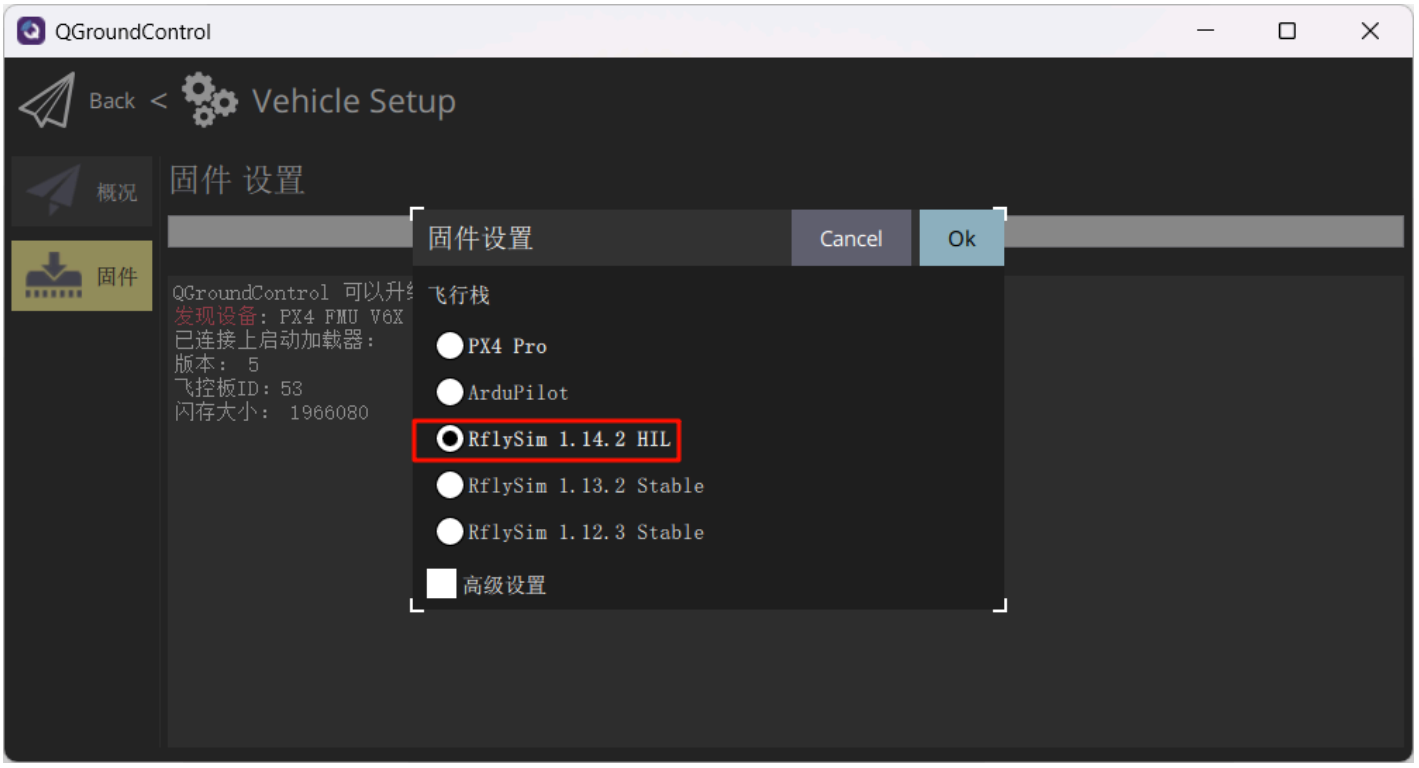
5.实验步骤

此处编写实验步骤，并添加相关图片。以及每一步设置结束后，具体的变化等等

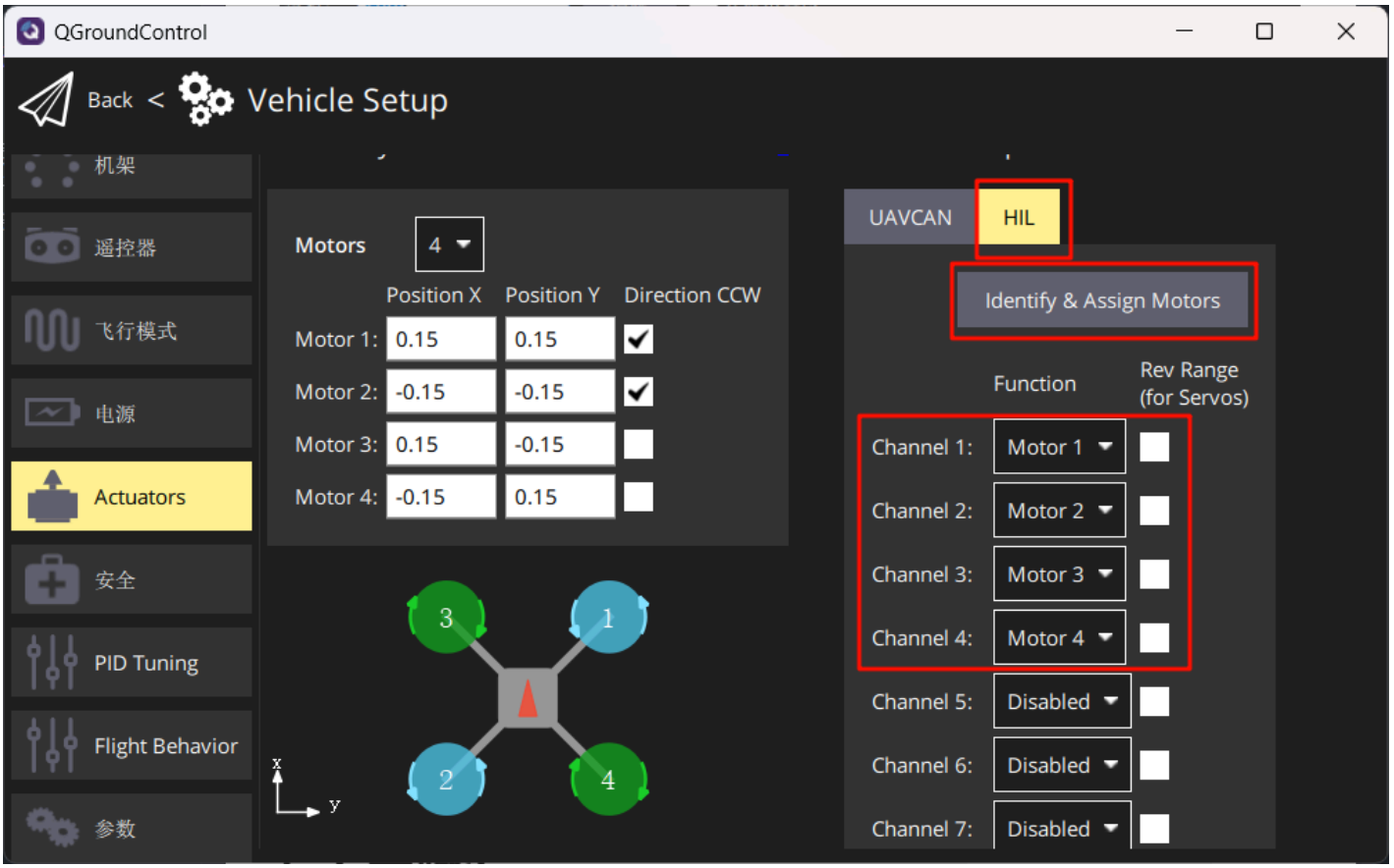
5.1 飞控设置

本实验要求飞控的固件是PX4 1.14.0及以上的版本，可直接通过地面站烧录1.14及以上的固件，打开QGC地面站进入固件界面，然后插入飞控将自动弹出对话框，如下图所示。选择1.14.0以上版本，点击“OK”，即可开始烧录固件，等待烧录完成即可。

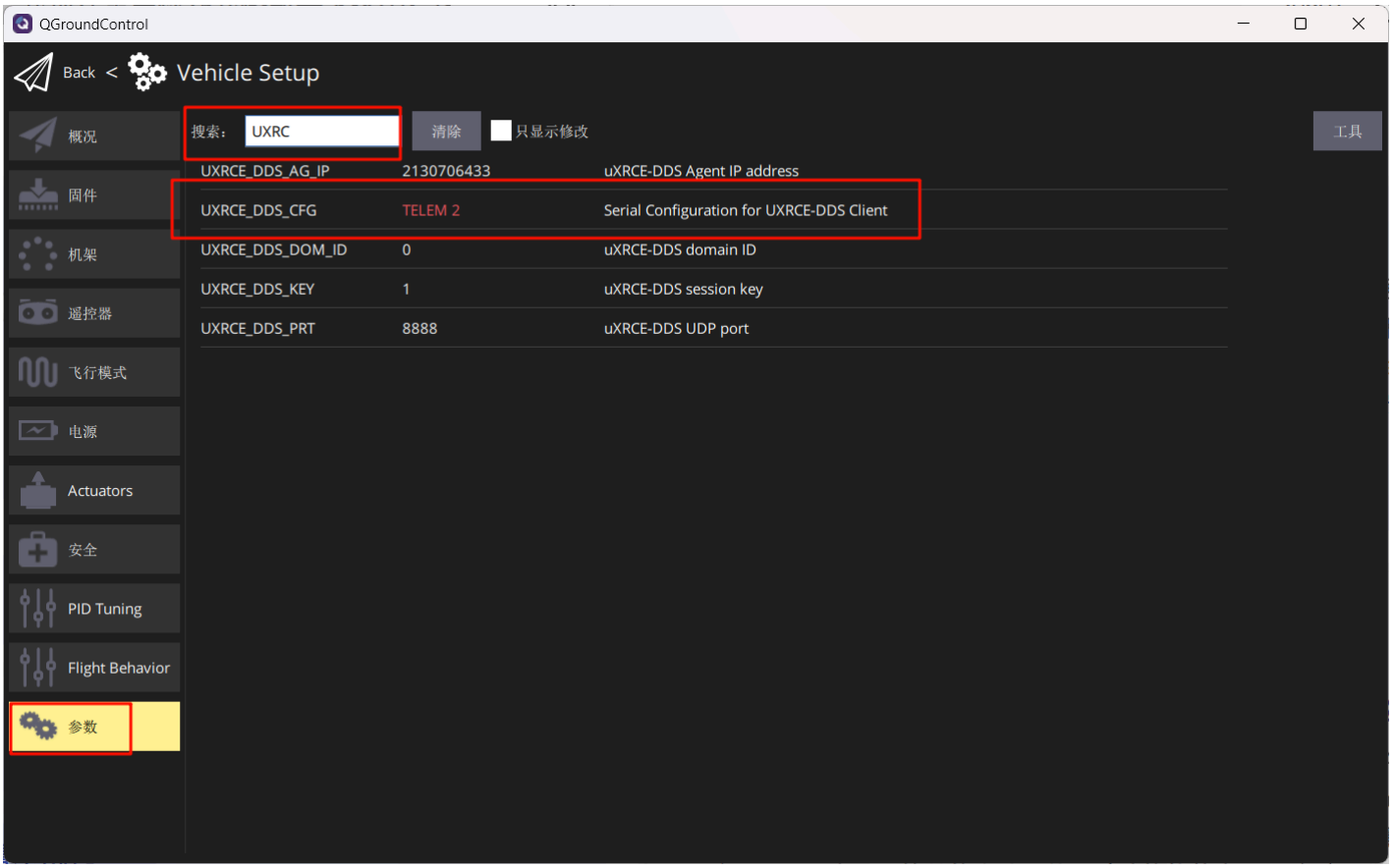




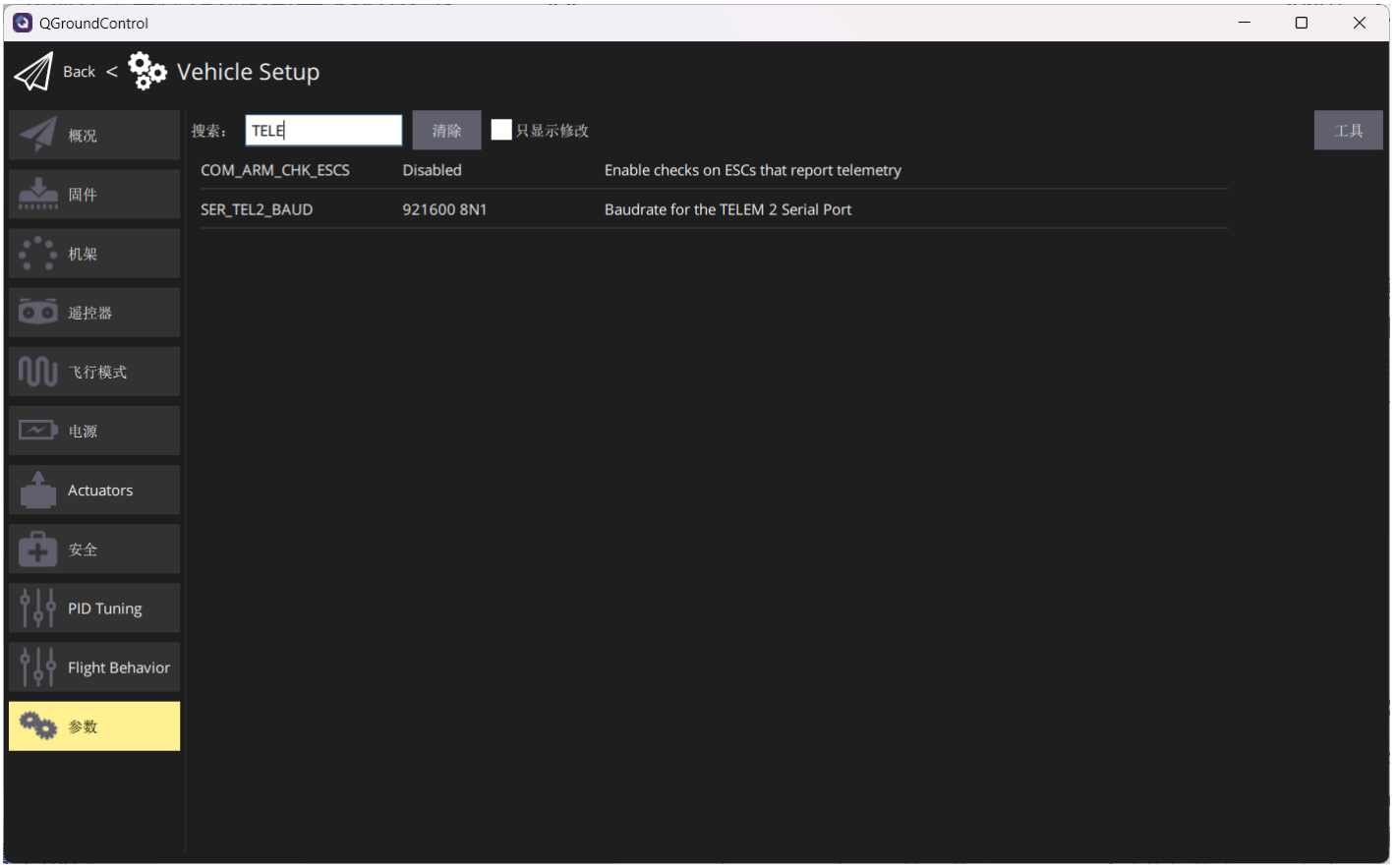
烧录完成后，等待飞控再次自动链接到QGC地面站中。进入“Actuators”界面，确认“HIL”界面中的Channel1~Channel4如下图设置，然后点击“Identify Assign Motors”即可。



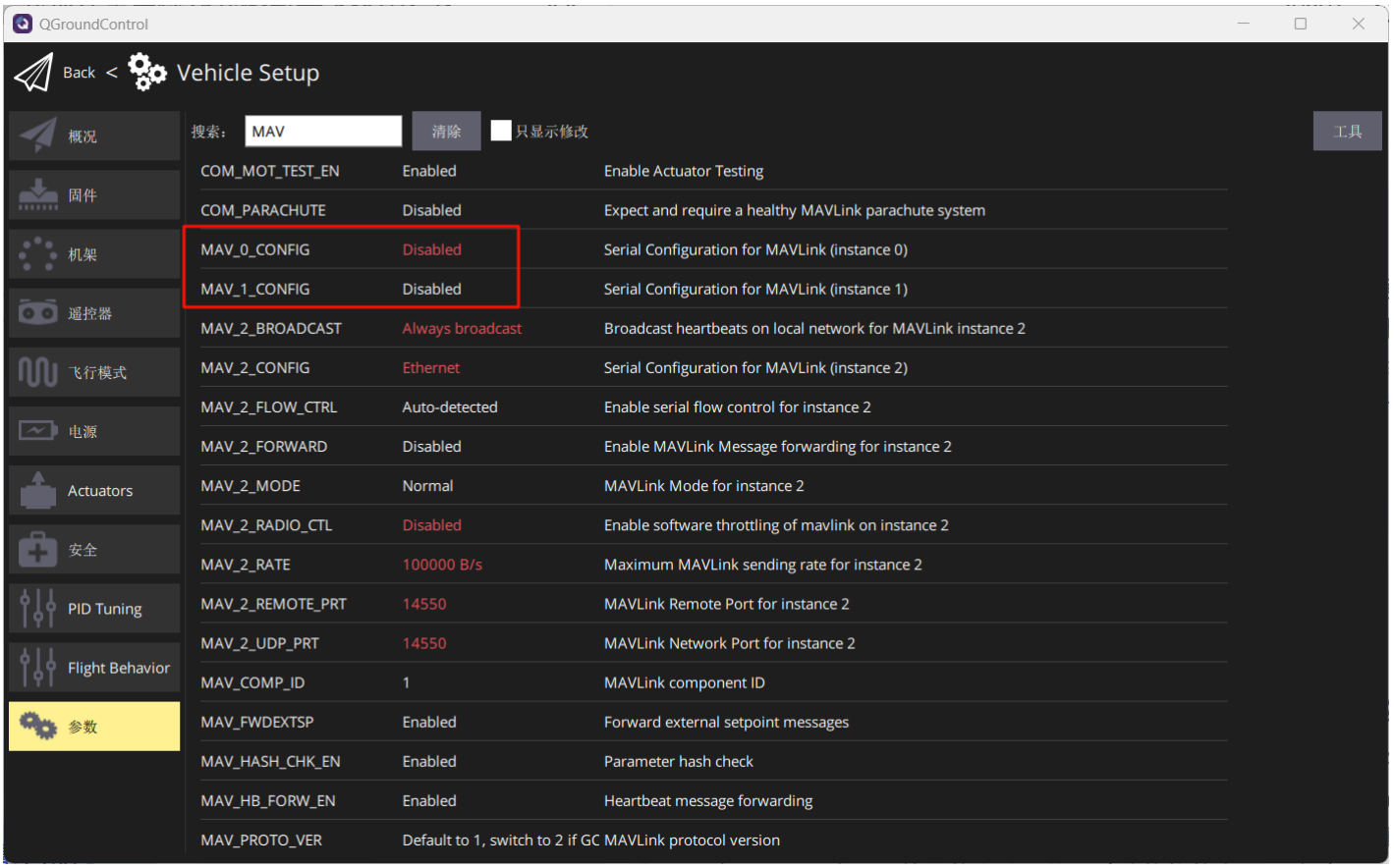
在“参数”界面搜索“UXRCE_DDS”，在筛选出的参数中，修改“UXRCE_DDS_CFG”为“TELEM2”。



然后搜索“TELE”在筛选出的参数中，确认“SER_TEL2_BAUD”参数为921600 8N1。



搜索“MAV”在筛选出的参数中，修改“MAV_0_CONFIG”和“MAV_1_CONFIG”参数为“Disable”。

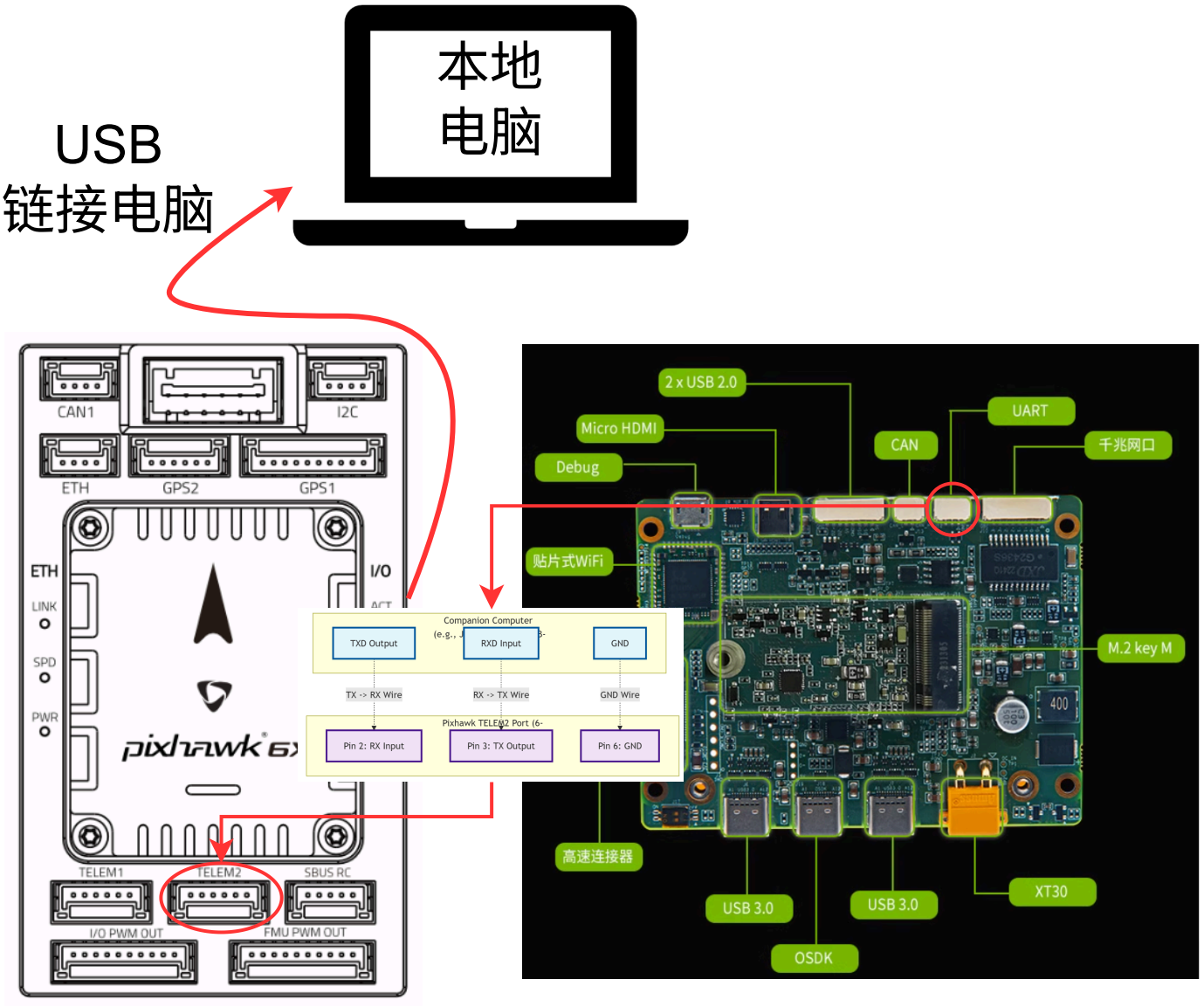


MAV_0_CONFIG 和 MAV_1_CONFIG 是用于配置 MAVLink 通信实例 的参数，主要决定 MAVLink 数据流通过哪个串口或其他通信接口（例如 UART、USB 或 TCP/UDP）进行传输。本次实验为HIL仿真实验，飞控上只有TELEM2与机载计算机（英伟达 Jetson Orin NX）进行通信，因此，上述两个参数均可设置“Disable”。

> 实飞中，需根据插入的设备来进行自定义设置。

5.2 硬件链接

查阅相关资料，了解Pixhawk 6X mini飞控和机载计算机（如：英伟达 Jetson Orin NX）的TELEM2和串口引脚接口定义，按照如下图所示的形式制作相关线材并链接。



> 上图中所用的Pixhawk 6X mini飞控的相关接口定义可见：https://docs.holybro.com/autopilot/pixhawk-baseboards/pixhawk-mini-baseboard-ports。所有到的英伟达 Jetson Orin NX为第三方厂商产品具体可见：https://item.taobao.com/item.htm?id=723672163022&spm=a1z10.5-c-s.w4002-24969555478.20.18bf7114UI0beh&skuId=5073502901185

5.3 机载计算机配置

首先，安装ROS2任意版本即可，本实验使用的为ROS2的Foxy版本（Ubuntu 20.04）。使用科学上网等工具配置机载计算机能够使用外网，复制本实验中文件Micro-XRCE-DDS-Agent.zip到机载计算机的任意路径下，打开一个终端，

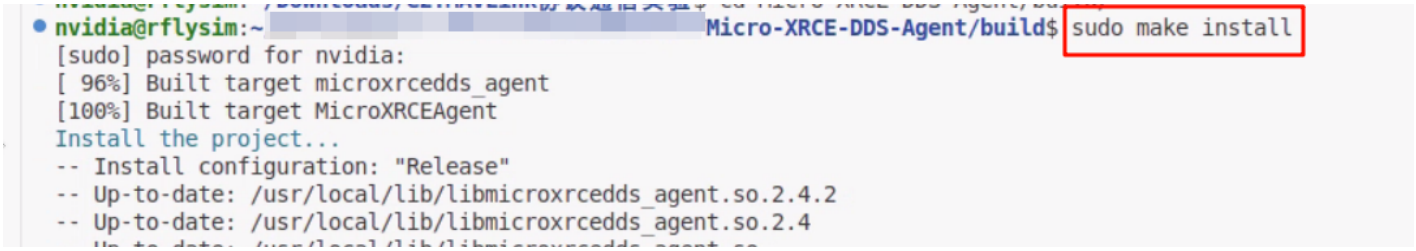
> Micro-XRCE-DDS-Agent.zip中的源码来自于 https://github.com/eProsima/Micro-XRCE-DDS-Agent.git 的v2.4.2分支中。

依次输入：

```
## 进入该文件夹
cd Micro-XRCE-DDS-Agent
## 创建build文件夹
mkdir build
cd build
## 编译
cmake ..
make
```

`cmake ..`时会出现部分警告，此处可忽略；`make`之后应该会有出现部分报错，目前测试过程中遇到的问题可见7 常见问题。该过程编译时间较长，耐心等待编译完成。完成后，依次输入：

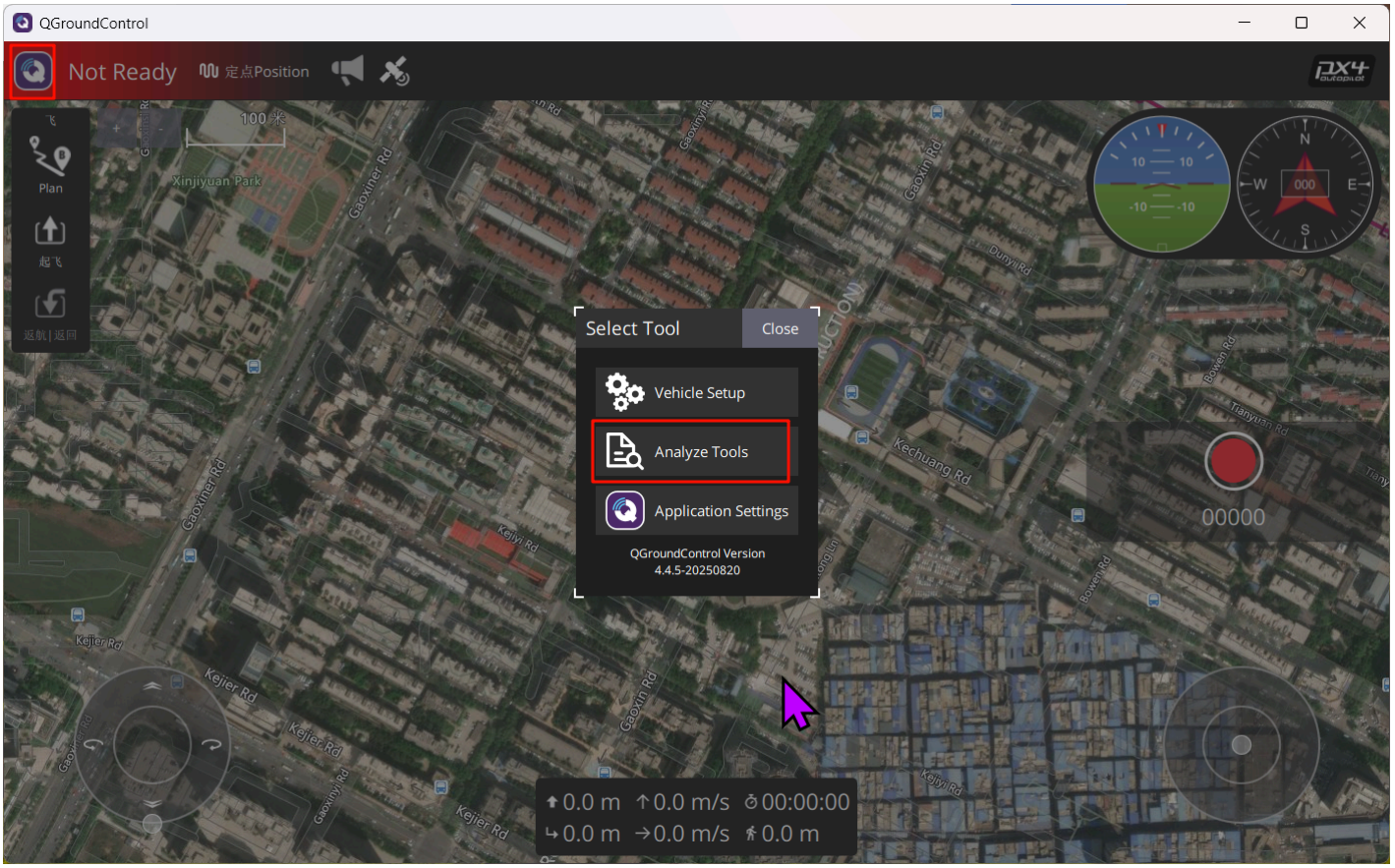
```
sudo make install
sudo ldconfig /usr/local/lib/
```

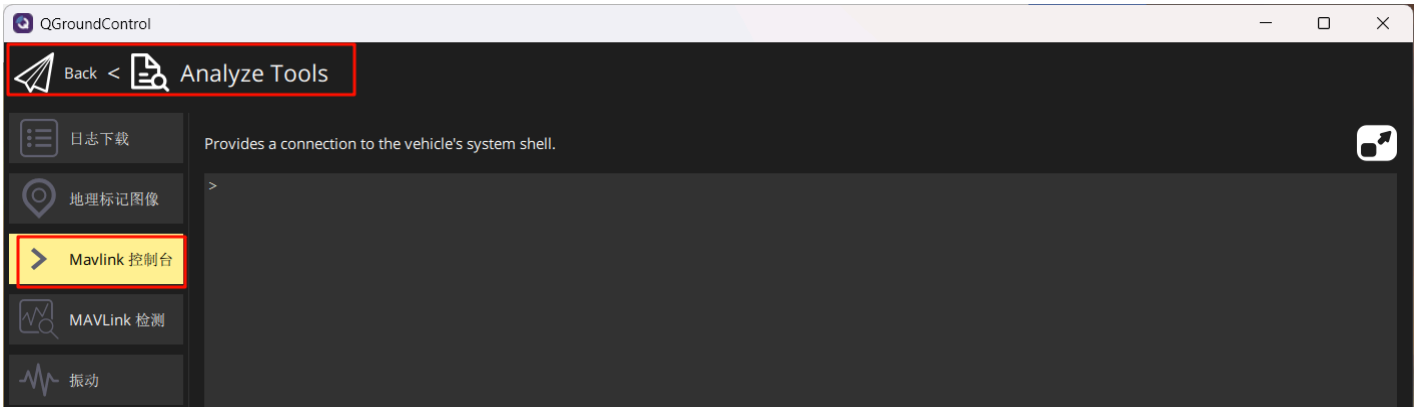


上述都安装完成后，即可进入下一步。

5.3 启动Micro-XRCE-DDS-client

Micro-XRCE-DDS的client端是在飞控中，打开QGC地面站（"*\Desktop\RflyTools\QGroundControl.lnk"），进入Mavlink控制台中





输入：

```
uxrce_dds_client status  # 查看uxrce_dds_client的状态

# 若在运行中可以通过以下命令停止
uxrce_dds_client stop
```

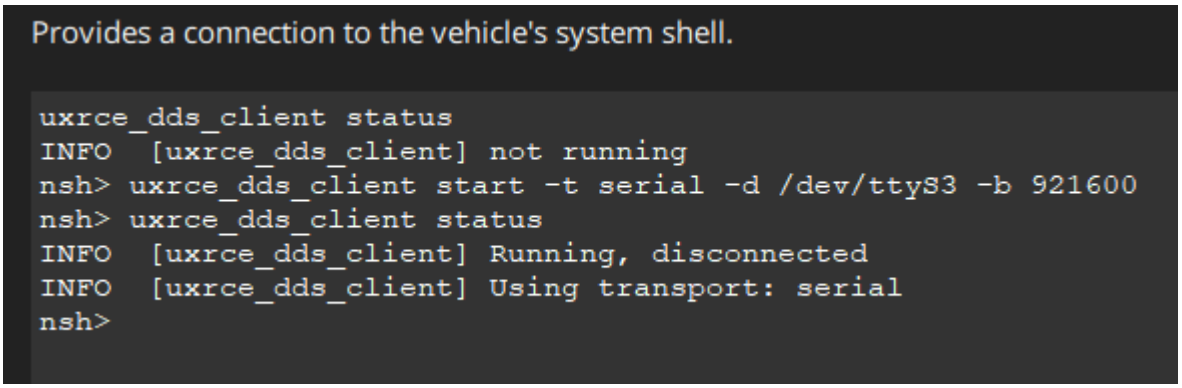
通常会显示如下结果，表示并没有启动Micro-XRCE-DDS的client端。

```
uxrce_dds_client status
INFO  [uxrce_dds_client] not running
nsh>
```

然后，输入：

```
# Pixhawk 6X mini飞控的TELEM2口在Nuttx中对应的端口名称为 /dev/ttyS3
uxrce_dds_client start -t serial -d /dev/ttyS3 -b 921600
uxrce_dds_client status
```

如下图所示，则表示Micro-XRCE-DDS的client端启动成功



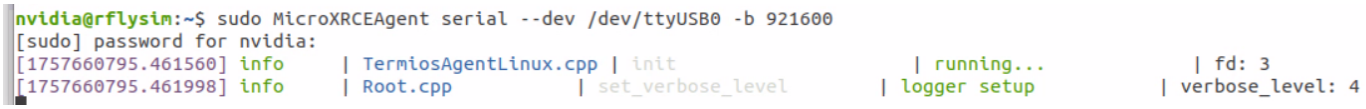
此时，关闭QGC地面站即可。

5.4 启动Micro-XRCE-DDS-Agent

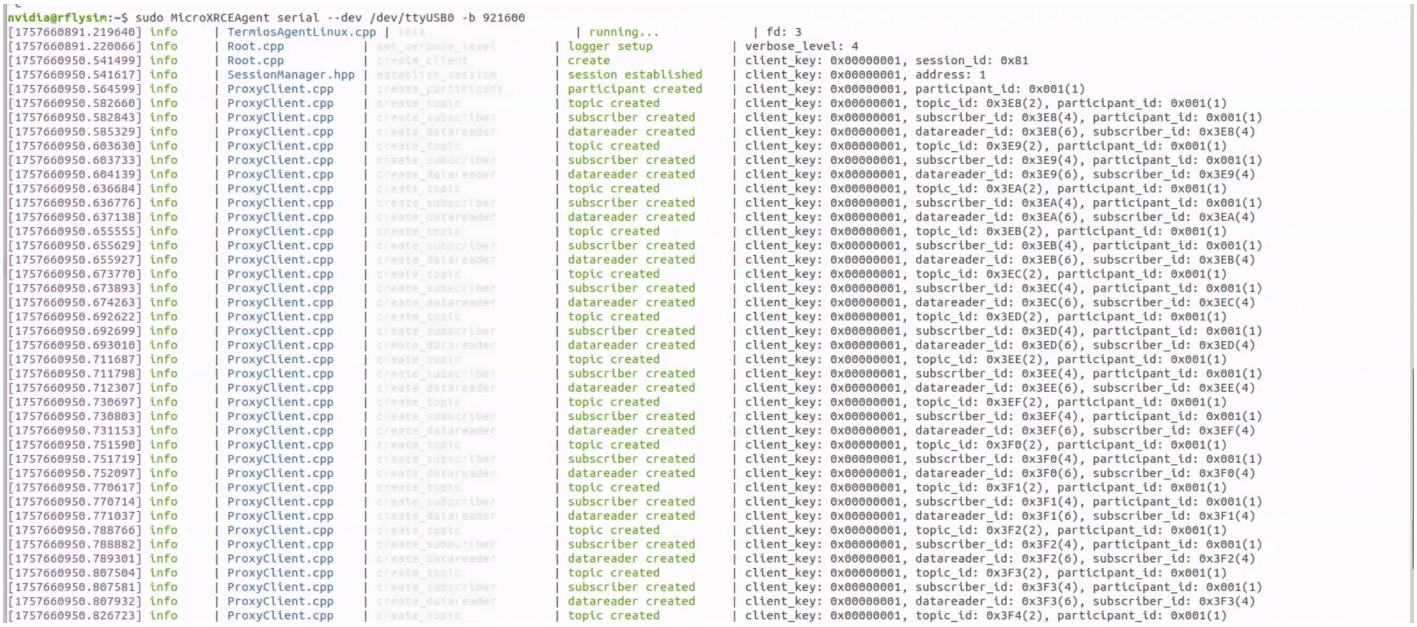
Micro-XRCE-DDS的Agent端是运行在机载计算机中的，确保机载计算机和飞控按照 # 5.2 步骤中的硬件链接，通常通过串口链接的端口名称为 `/dev/ttyTHS0` 或 `/dev/ttyACM0`，本实验中是通过USB与 `/dev/ttyUSB0`，如果使用其他端口，请自行修改。打开任意终端运行：

```
sudo chmod 777 /dev/ttyUSB0
sudo MicroXRCEAgent serial --dev /dev/ttyUSB0 -b 921600
```

若如下图所示，则表示并没有链接成功，可以尝试重启飞控即可解决。



最终，链接成功的状态如下：



> 注意：上述窗口，请勿关闭。

5.5 构建ROS2功能包

打开任意终端，依次输入：

```
mkdir -p ~/ws_sensor_combined/src/
cd ~/ws_sensor_combined/src/
```

然后，下载控制程序文件（也可直接复制本例程中的px4_ros_com.zip和px4_msgs.zip到`~/ws_sensor_combined/src/`文件夹中）。

```
git clone https://github.com/PX4/px4_msgs.git -b release/1.14
git clone https://github.com/PX4/px4_ros_com.git -b release/v1.14
```

> 注意：本实验中飞控的固件为1.14，机载计算机的Ubuntu版本为20.04，ROS2版本为Foxy。根据https://github.com/PX4/px4_msgs.git 中的说明文档，选择下载release/1.14分支程序文件。

px4_msgs

ROS 2 message definitions for the PX4 Autopilot project.

Building this package generates all the required interfaces to interface ROS 2

Supported versions and compatibility

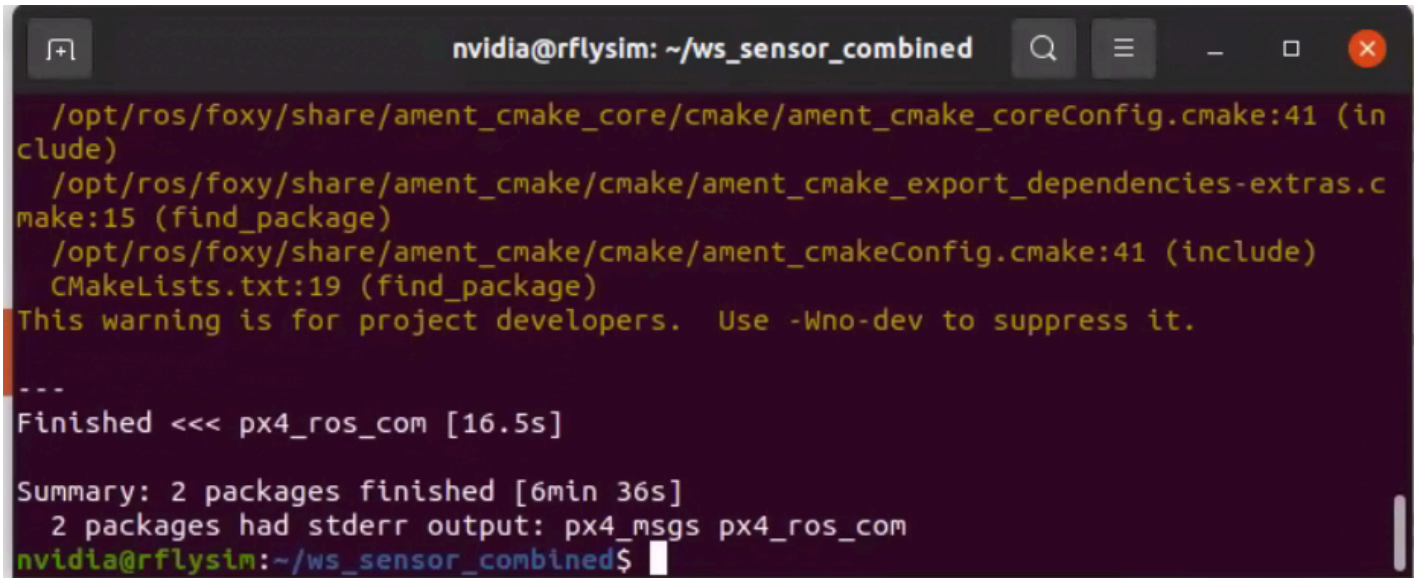
Depending on the PX4 and ROS versions you want to use, you need to check package:

PX4	ROS 2	Ubuntu	branch
v1.13	Foxy	Ubuntu 20.04	release/1.13
v1.14	Foxy	Ubuntu 20.04	release/1.14
v1.14	Humble	Ubuntu 22.04	release/1.14
v1.14	Rolling	Ubuntu 22.04	release/1.14
v1.15	Foxy	Ubuntu 20.04	release/1.15
v1.15	Humble	Ubuntu 22.04	release/1.15
v1.15	Rolling	Ubuntu 22.04	release/1.15
main	Foxy	Ubuntu 22.04	main
main	Humble	Ubuntu 22.04	main
main	Rolling	Ubuntu 22.04	main

https://github.com/PX4/px4_ros_com.git 仓库也推荐下载release/v1.14分支程序

```
cd ~/ws_sensor_combined/
source /opt/ros/foxy/setup.bash
colcon build
```

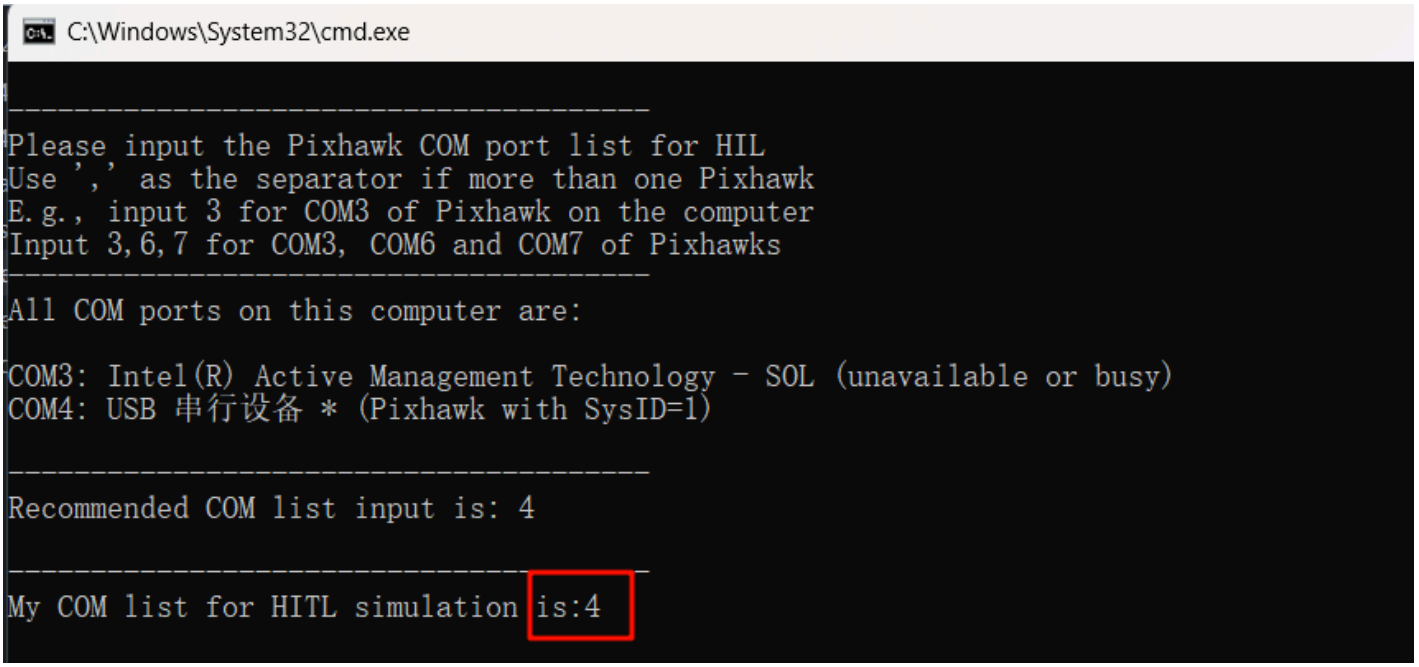
等待变成成功。



> 警告可忽略。

5.6 启动仿真

双击运行"*\Desktop\RflyTools\HITLRun.lnk"硬件在环仿真脚本，输入端口号（如：4），RflySim将自动启动一个CopterSim、QGC以及RflySim3D软件。



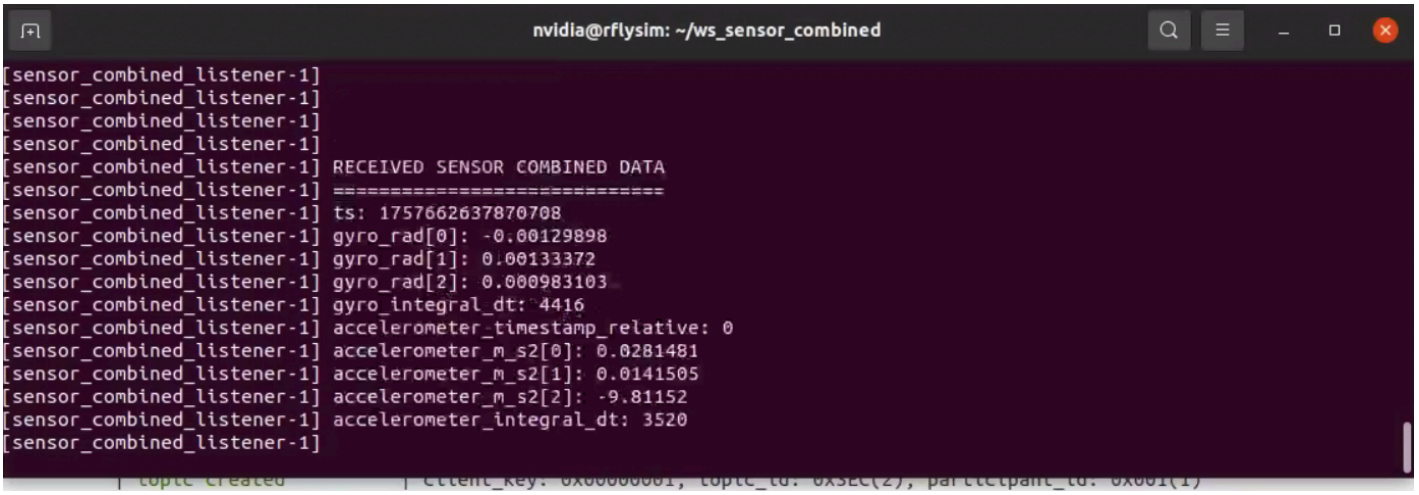
等待CopterSim软件的左下角出现`PX4: GPS 3D fixed & EKF initialization finished.`，表示初始化完成。

在机载电脑中，打开任意终端，依次输入

```
cd ~/ws_sensor_combined/
source install/setup.bash

## 监听IMU的数据
ros2 launch px4_ros_com sensor_combined_listener.launch.py
```

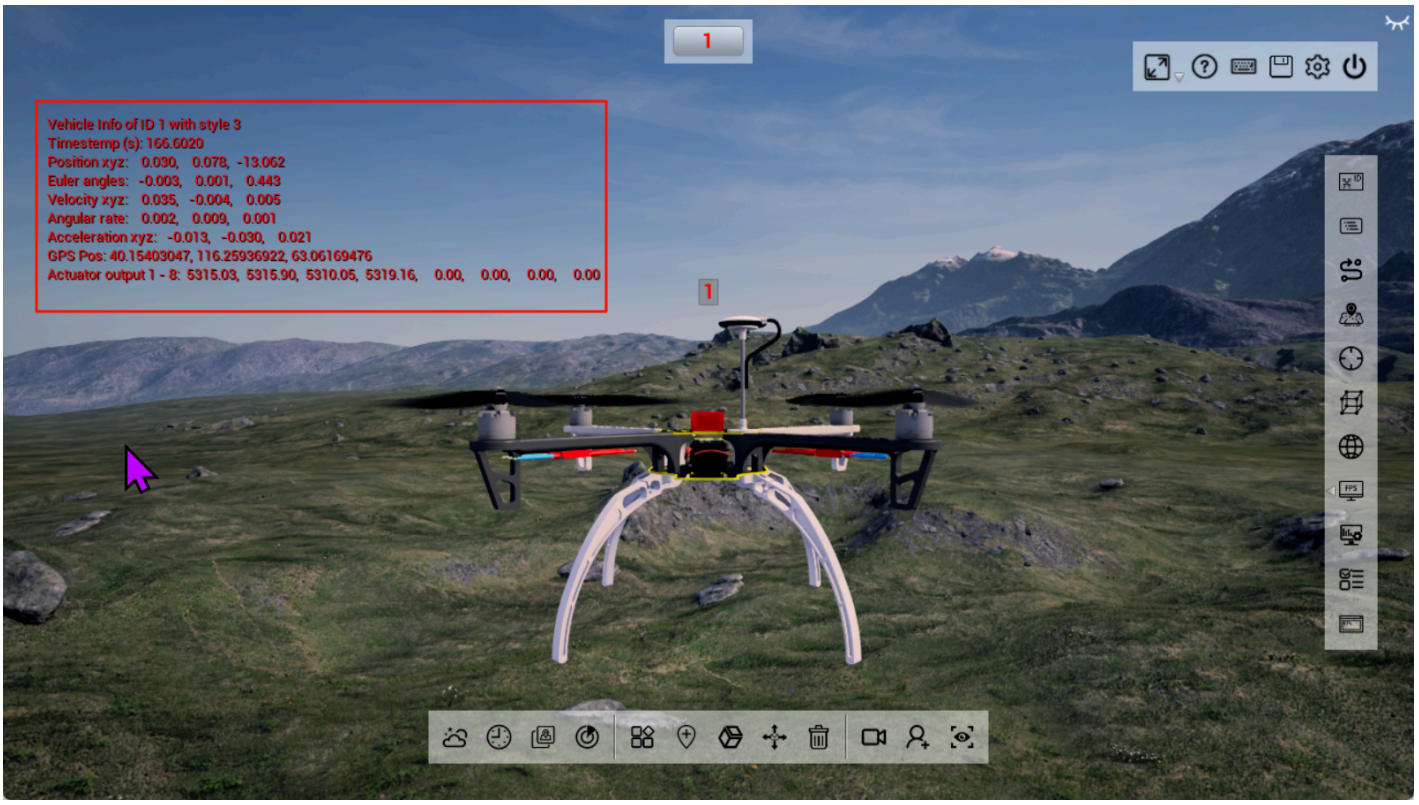
正常情况下，终端将打印出如下图所示的信息。



在当前终端下按下`Ctrl+C`终止程序运行。然后输入：

```
ros2 run px4_ros_com offboard_control
```

可在，RflySim3D中看到无人机正常起飞后，垂直上升5m左右，然后，进入悬停状态。RflySim3D中查看无人机的坐标信息如下：



无人机初始位置的Z轴坐标为-8左右，因此，最终显示悬停在-13米左右。

6.参考资料

1. https://github.com/PX4/px4_msgs
2. https://github.com/PX4/PX4-Autopilot
3. https://docs.px4.io/main/en/
4. https://github.com/eProsima/Fast-DDS
5. https://micro-xrce-dds.docs.eprosima.com/en/latest/
6. https://github.com/PX4/PX4-Micro-XRCE-DDS-Agent

7.常见问题

Q1：在进行make编译时，显示如下报错

```
CMake Error at CMakeLists.txt:18 (cmake_minimum_required):
  CMake 3.22 or higher is required.  You are running version 3.16.3


-- Configuring incomplete, errors occurred!
make[2]: *** [CMakeFiles/fastdds.dir/build.make:110: fastdds/src/fastdds-stamp/fastdds-configure] Error
1
make[1]: *** [CMakeFiles/Makefile2:141: CMakeFiles/fastdds.dir/all] Error 2
make: *** [Makefile:84: all] Error 2
```

A1：该问题主要是因为cmake的版本问题，该项目需要cmake的版本大于等于3.22及以上，可按照如下步骤进行解决：

```
cmake --version  # 查阅当前系统中的cmake版本
sudo apt purge cmake cmake-data  # 卸载系统版（如果有依赖问题，可跳过）
hash -r  # 刷新 shell 缓存

cd ~
wget https://github.com/Kitware/CMake/releases/download/v3.30.5/cmake-3.30.5-linux-aarch64.tar.gz

tar -xzf cmake-3.30.5-linux-aarch64.tar.gz
sudo mv cmake-3.30.5-linux-aarch64 /opt/cmake  # 移动到系统目录
sudo ln -s /opt/cmake/bin/* /usr/local/bin/  # 创建符号链接到 PATH

cmake --version  # 应显示 3.30.5 或更高

# 如果 PATH 未更新，重启终端或运行
source ~/.bashrc
```