

Simulink/DLL 综合模型 ROS 转发 (C++版)

1. 实验目的

- 使用 C++ ROS 转发节点 (udp_ros_bridge) 将 RflySim 综合模型 UDP 数据转换为 ROS 话题，替代 mavros 的基础功能。
- 让基于 PX4 软硬件在环的视觉/控制算法快速迁移到 RflySim 综合模型。
- 演示 ROS1/ROS2 环境下对多机（默认 3 架）的状态订阅与控制。

2. 实验要求

- 软件要求：Windows 10 及以上版本；RflySim 工具链^[1]；WSL2 (RflySim-20.04)；ROS1 Noetic 或 ROS2；Python 3^[2]。
- 硬件要求：笔记本或台式电脑 1 台^[3]。

3. 实验地址

例程目录：

[\[安装目录\]\RflySimAPIs\6.RflySimExtCtrl\0.ApiExps\e21.RosTransExps\1.C++Demo](#)

[\[1_Deploy_and_Build_ROS1.bat\]](#)：ROS1一键部署与启动脚本

[\[2_Deploy_and_Build_ROS2.bat\]](#)：ROS2一键部署与启动脚本

[\[build_bridge.sh\]](#)：一键编译与启动脚本

[\[run_bridge_in_wsl.sh\]](#)：在WSL环境中运行桥接程序的脚本

[\[udp_ros_bridge_template\]](#)：C++ ROS转发节点模板（ROS1/ROS2）

[\[udp_ros_bridge_template/src/udp_ros_bridge.cpp\]](#)：UDP与ROS通信的核心转发节点源码

[\[udp_ros_bridge_template/package.xml\]](#)：ROS包配置文件

[\[udp_ros_bridge_template/CMakeLists.ros1.txt\]](#)：ROS1环境下的编译配置文件

[\[udp_ros_bridge_template/CMakeLists.ros2.txt\]](#)：ROS2环境下的编译配置文件

[udp_ros_bridge_template/msg/OutCopterStruct.msg]: 定义输出飞行器结构的消息格式

[udp_ros_bridge_template/msg/SILIntFloat.msg]: 定义软硬件在环仿真中整数浮点数消息格式

[udp_ros_bridge_template/msg/SOut2Simulator.msg]: 定义输出到模拟器的消息格式

[udp_ros_bridge_template/msg/SOutCopterStruct.msg]: 定义输出飞行器结构的消息格式

[Simulink--dll]: 综合模型dll文件及仿真与控制脚本

[Simulink--dll/MulticopterNOpx4.bat]: 启动多旋翼综合模型仿真的批处理脚本

[Simulink--dll/MulticopterNOpx4.dll]: 多旋翼综合模型的动态链接库文件

[Simulink--dll/WinWSL.bat]: 用于连接Windows与WSL环境的批处理脚本

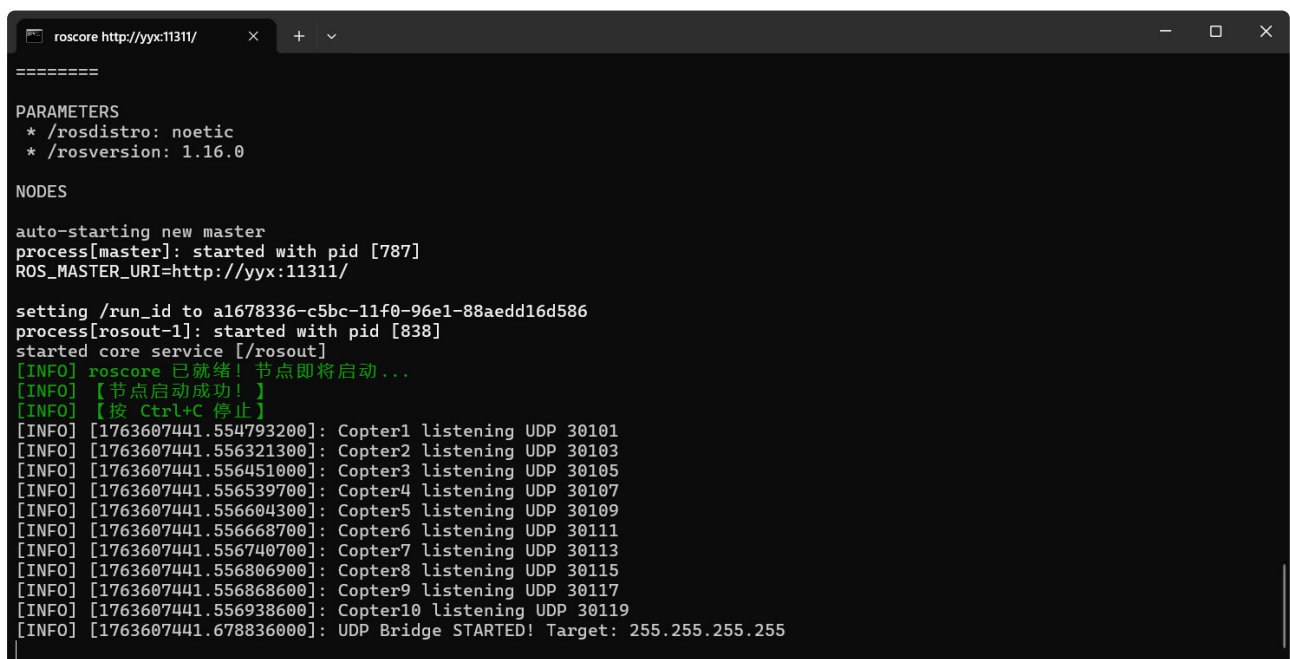
[Simulink--dll/Ros12MultiUav.py]: ROS1/ROS2环境下多无人机控制验证脚本

4. 实验内容或步骤

4.1 自动编译并启动 ROS 转发节点（推荐）

本例程提供了 Windows 批处理脚本，可自动将必要文件复制到 WSL 并根据目标 ROS 版本进行编译并启动节点。启动方式二选一即可。

- **ROS1 环境：**双击运行 `1_Deploy_and_Build_ROS1.bat`
编译完成后若看到节点启动信息，说明转发节点运行正常。

A terminal window titled 'roscore http://yyx:11311/' showing the output of a ROS1 deployment script. The output includes parameters for 'noetic' and '1.16.0', followed by node startup messages. It shows the master node starting with PID 787, and then several 'Copter' nodes (Copter1 to Copter10) starting and listening on various UDP ports (30101 to 30117). Finally, the 'UDP Bridge' starts with target address 255.255.255.255. There are also green status messages in Chinese indicating success and a prompt to press Ctrl+C to stop.

```
=====
PARAMETERS
* /roscore: noetic
* /rosversion: 1.16.0

NODES

auto-starting new master
process[master]: started with pid [787]
ROS_MASTER_URI=http://yyx:11311/

setting /run_id to a1678336-c5bc-11f0-96e1-88aedd16d586
process[rosout-1]: started with pid [838]
started core service [/rosout]
[INFO] roscore 已就绪！节点即将启动...
[INFO] 【节点启动成功！】
[INFO] 【按 Ctrl+C 停止】
[INFO] [1763607441.554793200]: Copter1 listening UDP 30101
[INFO] [1763607441.556321300]: Copter2 listening UDP 30103
[INFO] [1763607441.556451000]: Copter3 listening UDP 30105
[INFO] [1763607441.556539700]: Copter4 listening UDP 30107
[INFO] [1763607441.556604300]: Copter5 listening UDP 30109
[INFO] [1763607441.556668700]: Copter6 listening UDP 30111
[INFO] [1763607441.556740700]: Copter7 listening UDP 30113
[INFO] [1763607441.556806900]: Copter8 listening UDP 30115
[INFO] [1763607441.556868600]: Copter9 listening UDP 30117
[INFO] [1763607441.556938600]: Copter10 listening UDP 30119
[INFO] [1763607441.678836000]: UDP Bridge STARTED! Target: 255.255.255.255
```

- **ROS2 环境：**双击运行 `2_Deploy_and_Build_ROS2.bat`

编译完成后若看到节点启动信息，说明转发节点运行正常。

```
colcon build [1/1 done] [0 ong] x + v
root@yyx:~# ./build_bridge.sh copy
[INFO] 【编译 + 启动 ROS2】
Starting >>> udp_ros_bridge
--- stderr: udp_ros_bridge
CMake Warning:
  Manually-specified variables were not used by the project:

    ROS2_BUILD

---
Finished <<< udp_ros_bridge [7.44s]

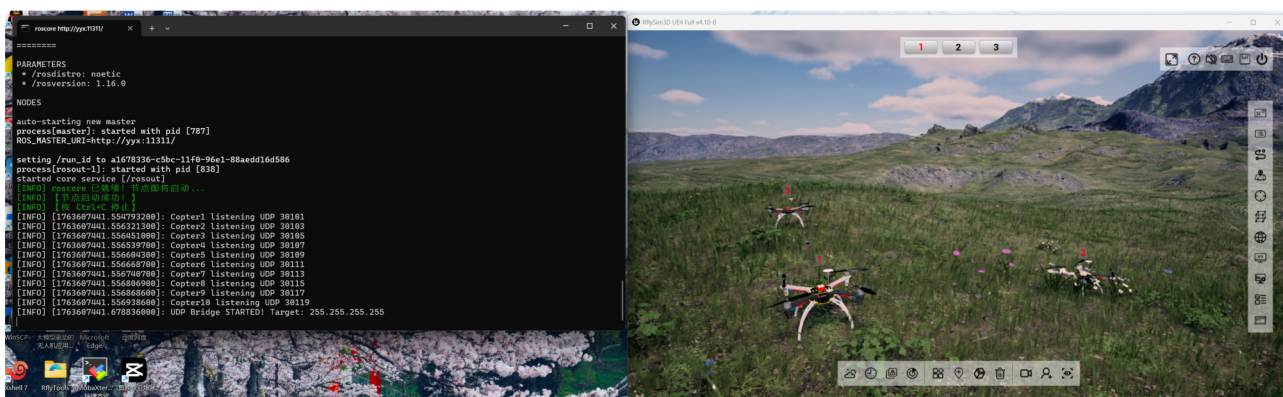
Summary: 1 package finished [7.66s]
1 package had stderr output: udp_ros_bridge
[INFO] 【ROS2 节点启动成功！】
[INFO] 【按 Ctrl+C 停止】
1763608647.837263 [0] udp_ros_br: selected interface "lo" is not multicast-capable: disabling multicast
[INFO] [1763608647.843812100] [udp_ros_bridge_node]: Copter1 listening UDP 30101
[INFO] [1763608647.845857800] [udp_ros_bridge_node]: Copter2 listening UDP 30103
[INFO] [1763608647.848323200] [udp_ros_bridge_node]: Copter3 listening UDP 30105
[INFO] [1763608647.850309200] [udp_ros_bridge_node]: Copter4 listening UDP 30107
[INFO] [1763608647.851584600] [udp_ros_bridge_node]: Copter5 listening UDP 30109
[INFO] [1763608647.851758700] [udp_ros_bridge_node]: Copter6 listening UDP 30111
[INFO] [1763608647.851889000] [udp_ros_bridge_node]: Copter7 listening UDP 30113
[INFO] [1763608647.852060500] [udp_ros_bridge_node]: Copter8 listening UDP 30115
[INFO] [1763608647.852155700] [udp_ros_bridge_node]: Copter9 listening UDP 30117
[INFO] [1763608647.852248300] [udp_ros_bridge_node]: Copter10 listening UDP 30119
[INFO] [1763608647.862942800] [udp_ros_bridge_node]: UDP Bridge STARTED! Target: 255.255.255.255
```

注意：

- 脚本运行后会自动检查 WSL 中是否已存在根据对应 ROS 版本编译好的节点程序。
- 若不存在，会自动执行复制与编译流程。
- 若已存在，则直接启动节点。
- 若需要指定 WSL 发行版（如电脑中有多个 WSL），请右键编辑脚本，修改
`set WSL_DISTRO_NAME=你的发行版名称`（如 `RflySim-20.04`）。

4.2 启动综合模型仿真

双击运行 `Simulink--dll/MulticopterNOpx4.bat`，启动 3 架综合模型仿真。

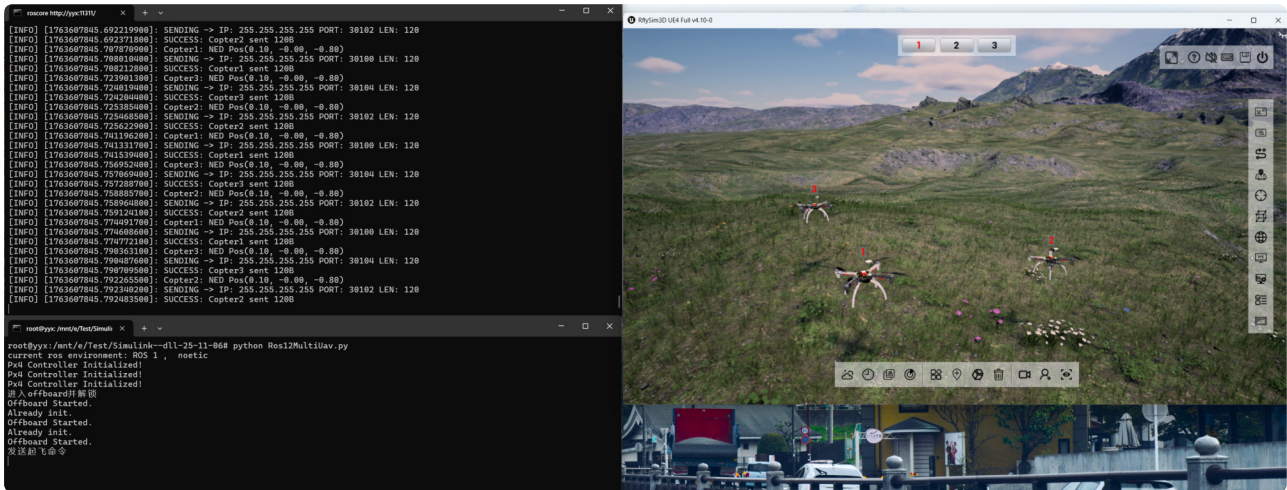


4.3 运行控制脚本验证

再次运行 `Simulink--dll/WinWSL.bat`，在 WSL 中进入示例目录（或将脚本复制到 `/root`），执行：

```
python Ros12MultiUav.py
```

可看到飞机起飞并打印实时状态，说明 ROS 转发链路生效。



5. 关键知识点

5.1 节点程序作用

能够替代 mavros 基础功能，实现基于 PX4 软硬件在环（SIL/HIL）的算法向 RflySim 综合模型（CopterSim）的快速迁移。

核心功能实现：

1. 数据上行（RflySim -> ROS）：

- 监听 UDP 端口 `30100 + id*2 - 1`（如 30101）接收仿真真值。
- 发布 ROS 话题：
 - 位置/姿态：`mavros/local_position/pose` (`geometry_msgs/PoseStamped`)
 - 速度：`mavros/local_position/velocity_local` (`geometry_msgs/TwistStamped`)
 - GPS：`mavros/global_position/global` (`sensor_msgs/NavSatFix`)
- 支持多机：`id=1` 话题前缀为 `/mavros`，`id>1` 为 `/mavros{id}`（如 `/mavros2`）。

2. 指令下行 (ROS -> RflySim):

- 订阅控制话题: `mavros/setpoint_raw/local` (`mavros_msgs/PositionTarget`)。
- 支持控制模式:
 - **NED 位置控制** (`mask: 0x0DF8`): 映射到 `CmdPosition` | `HasPos`。
 - **NED 位置+偏航控制** (`mask: 0x09F8`): 映射到 `CmdPosition` | `HasPos` | `HasYaw`。
 - **FRD 速度控制** (`mask: 0x0DC7`): 映射到 `HasVel` | `FRD`。
- 提供服务:
 - 参数设置: `mavros/param/set`
 - 模式切换: `mavros/set_mode` (支持 `OFFBOARD`, `AUTO.LAND`)
 - 解锁/上锁: `mavros/cmd/arming`

5.2 编译与启动参数详解

脚本 `build_bridge.sh` 支持 ROS1/ROS2 自动识别与编译。

命令格式:

```
./build_bridge.sh [模式] [ROS版本] [参数]
```

参数说明:

- **模式:**
 - `copy`: 复制模板代码到工作空间 -> 编译 -> 启动 (首次使用推荐)。
 - `start`: 跳过编译, 直接启动 (需确保已编译)。
- **ROS版本:** `ros1` 或 `ros2` (默认)。
- **可选参数:**
 - `--ip <IP>`: 目标 IP, 广播推荐 `255.255.255.255`, 单播如 `192.168.1.100`。
 - `--num <N>`: 监听飞机数量 (默认 10)。
 - `--id <N>`: 起始飞机 ID (默认 1)。

示例:

- ROS1 单播控制 ID=1~20 的飞机:

```
./build_bridge.sh ros1 --ip 192.168.1.100 --id 1 --num 20
```
- ROS2 仅启动不编译:

```
./build_bridge.sh start --ip 255.255.255.255
```

6. 参考资料

1. [RflySim官方文档](#)

7. 常见问题

Q1: 运行 `build_bridge.sh` 提示找不到模板或无法编译?

A1: 确认 `udp_ros_bridge_template` 与 `build_bridge.sh` 已复制到 WSL `/root` ; 检查 ROS 环境是否已安装并正确 `source` 。

Q2: ROS 话题没有数据或飞机不响应?

A2: 检查 `--ip` 是否与仿真端一致, Windows 防火墙是否拦截 UDP; 确认仿真已启动且 `--id/--num` 覆盖当前 CopterID。

Q3: 脚本能运行但控制无效?

A3: 确保 `Ros12MultiUav.py` 在 WSL 中运行, 并且 ROS1/ROS2 环境与转发节点一致; 必要时重启转发节点与仿真。

-
1. <https://rflysim.com/> ↩
 2. 建议提前完成 ROS 与 WSL2 环境配置, 再执行本实验中的桥接编译与启动步骤。 ↩
 3. 推荐配置请见: <https://rflysim.com/doc/zh/HowToInstall.pdf> ↩