
1. 实验名称及目的

Python 控制差动无人车位置软硬件在环仿真：软硬件在环仿真模式下，以 Python 的方式通过平台位置控制接口实现单辆/多辆无人车位置控制。

2. 实验原理

2.1. 软/硬件在环仿真（SIL/HIL）的实现[1][2]

从实现机制的角度分析，可将 RflySim 平台分为运动仿真模型、底层控制器、三维引擎、外部控制四部分。

- **运动仿真模型：**这是模拟飞行器运动的核心部分。在 RflySim 平台中，运动仿真模型是通过 MATLAB/Simulink 开发的，然后通过自动生成的 C++ 代码转化成 DLL（动态链接库）文件。在使用 RflySim 平台进行软硬件在环仿真时，会将 DLL 模型导入到 CopterSim，形成运动仿真模型。这个模型在仿真中负责生成飞行器的运动响应，它拥有多个输入输出接口与底层控制器、三维引擎、地面控制站和外部控制进行数据交互，具体数据链路、通信协议及通信端口号见 [API.pdf 中的通信接口部分](#)。
- **底层控制器：**在软/硬件在环仿真（SIL/HIL）中，真实的飞行控制硬件（如 PX4 飞行控制器）被集成到一个虚拟的飞行环境中。在软件在环仿真（SIL）中，底层控制器（通过 wsl 上的 PX4 仿真环境运行）通过网络通信与运动仿真模型交互数据。在硬件在环仿真（HIL）中，它（将 PX4 固件在真实的飞行控制器（即飞控）硬件上运行）则通过串口通信与运动仿真模型进行数据交互。飞控与 CopterSim 通过串口（硬件在环 HITL）或网络 TCP/UDP（软件在环 SITL）进行连接，使用 MAVLink 进行数据传输，实现控制闭环。
- **三维引擎：**这部分负责生成和处理仿真的视觉效果，提供仿真环境和模型的三维视图，使用户能够视觉上跟踪和分析飞行器的运动。CopterSim 发送飞机位姿、电机数据到三维引擎，实现可视化展示。
- **外部控制（offboard）：**从仿真系统外部对飞行器进行的控制，包括自动飞行路径规划、远程控制指令等。在平台例程中主要通过地面控制站（QGC）、MATLAB 和 Python 调用对应接口实现。

2.2. 通过外部控制接口（python）进行单辆无人车位置控制

单机控制脚本 CarR1Diff_OffboardPos1.py 中依次调用了 RflySim 平台飞机控制接口协议文件 PX4MavCtrlV4.py 中定义的以下接口函数

创建通信示例

```
mav1 = PX4MavCtrl.PX4MavCtrl(1)
```

创建一架飞机的通信示例

启用 Mavlink 消息监听循环

```
mav1.InitMavLoop()
```

配置 CopterSim 通信模式，该函数的参数定义如下：

```
def InitMavLoop(self,UDPMode=2):
    """ Initialize MAVLink listen loop from CopterSim
        0 and 1 for UDP_Full and UDP_Simple Modes, 2 and 3 for MAVLink_Full and
        MAVLink_Simple modes, 4 for MAVLink_NoSend
        The default mode is MAVLink_Full
    """
```

默认通信模式为 **Mavlink_Full**：Python 直接发送 MAVLink 消息给 CopterSim，再转发给 PX4，数据量较大适合单机控制；适合单机或少量载具仿真，载具数量不大于 4；

启动外部控制（offboard）

```
mav1.initOffboard()
```

使 px4 控制器进入外部控制模式，且以 30HZ 的频率发送 offboard 指令。

注，虽然在此处已经启用了外部控制模式，对于运行阶段中 **flag=0** 的部分（解锁和移动到初始位置），不需要外部控制模式，实际指令还是由底层控制器完成的。

设定航路点

```
n = 30
r = 400
missionPoints=[]
for i in range(n):
    angle = 2*math.pi*i/n
    x=r*math.sin(angle)
    y=r*math.cos(angle)
    missionPoints.append([x,y,0])
```

用一组离散的点模拟圆形运动轨迹，并在循环中通过 **append** 方法逐个将相应的轨迹点存入目标点列表（**missionPoints**）。**missionPoints.append([x,y,0])**表示在 **missionPoints** 列表的末尾添加一个新的列表**[x,y,0]**。

根据欧拉公式：

$$e^{ix} = \cos x + i\sin x$$

这些点将在 x-y 平面上形成一个圆形轨迹。

任务阶段

完成上述设置后，程序会通过检查一个 **flag** 变量的值来决定无人车应该执行哪些动作。

当 flag == 0 时，解锁无人车

解锁车辆

```
mav1.SendMavArm(True)
```

设定启动目标点

```
targetPos=[0, 50, 0]
mav1.sendMavTakeOff(targetPos[0],targetPos[1],targetPos[2])
```

发送绝对的 GPS 坐标作为起飞目标点，使用 **sendMavTakeOffGPS** 命令，最后三位分别是经度、维度和高度，会先从 **uavPosGPSHome** 向量中提取解锁 GPS 坐标，在此基础上

用绝对坐标

当 **flag == 1** 时，无人车进入航路寻迹模式

位置检测

```
curPos=mav1.uavPosNED
dis = math.sqrt((curPos[0]-targetPos[0])**2+(curPos[1]-targetPos[1])**2)
```

计算飞机当前位置和起飞目标位置的水平距离，用于判断是否到达目标位置，以开始下一阶段任务。

航路寻迹模式

```
targetPos=missionPoints[flagI]
mav1.SendPosNED(targetPos[0], targetPos[1], targetPos[2])
```

会通过航路点索引 **flagI** 的值从 **missionPoints** 列表中读取相应的航点，并通过 **SendPosNED** 函数更新为下一个目标点。

当 **flag == 2** 时，无人车在航路点之间的运行

```
targetPos=missionPoints[flagI]
```

更新目标位置为 **missionPoints** 列表中的下一个点。

```
curPos=mav1.uavPosNED
dis = math.sqrt((curPos[0]-targetPos[0])**2+(curPos[1]-targetPos[1])**2)
```

再次计算无人机与目标位置之间的距离。

```
if dis < 50:
    flagI=flagI+1
    spd = 10+flagI/3.0
    mav1.SendCruiseSpeed(spd)
```

如果距离小于 5 米，更新 **flagI** 以切换到下一个航路点，并调整无人车的巡航速度。

```
else:
    mav1.SendPosNED(targetPos[0], targetPos[1], targetPos[2])
```

如果距离不满足条件，则继续向当前目标位置运行。30 个轨迹点取完后，无人车会脱离圆形轨迹

2.3. 通过外部控制接口（python）进行多辆无人车位置控制

多机控制脚本 **CarR1Diff_OffboardPos4.py** 脚本的实现逻辑与单机控制相同，只是需要创建 4 辆无人车，再将相同的控制指令复制 4 份

创建通信示例

```
VehilceNum = 3
mav=[]
for i in range(VehilceNum):
    mav=mav+[PX4MavCtrl.PX4MavCtrler(1+i)]
```

创建 3 架飞机的通信示例

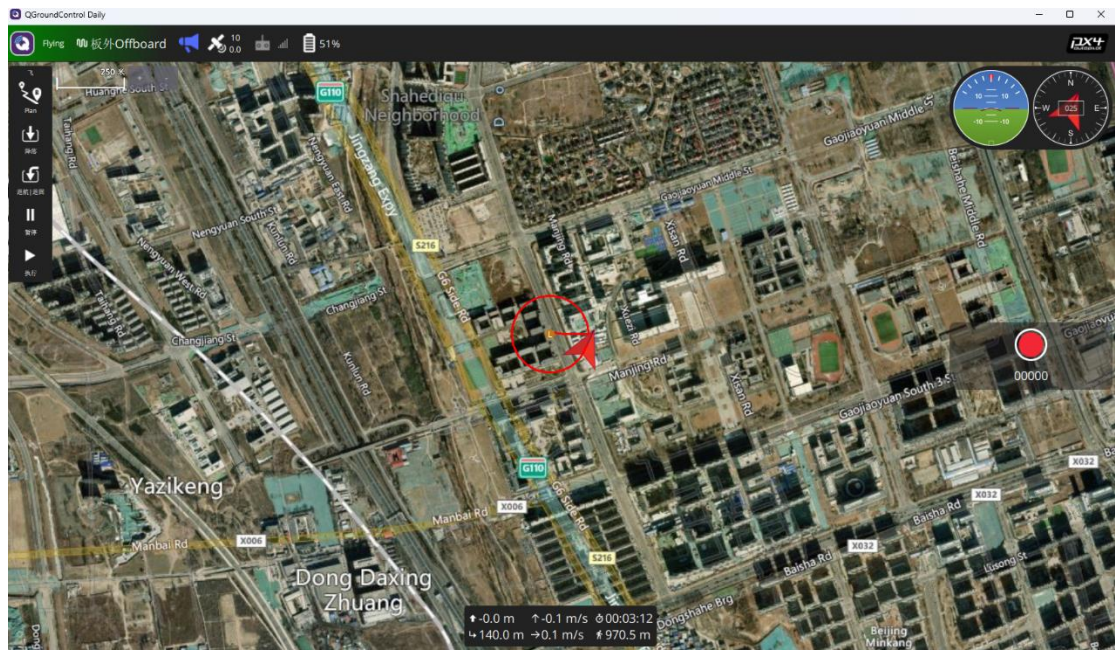
启用 **Mavlink** 消息监听循环

```
for i in range(VehilceNum):
    mav[i].InitMavLoop()
```

配置 3 架飞机的 **CopterSim** 通信模式

3. 实验效果

通过平台位置控制接口以 Python 控制单量/多辆无人车的位置实现画圆。



4. 文件目录

文件夹/文件名称	说明
CarR1Diff_OffboardPos1.bat	单辆无人车位置控制软件在环仿真批处理文件。
CarR1Diff_OffboardPos1.py	单辆无人车位置控制脚本。
CarR1Diff_OffboardPos4.bat	多辆无人车位置控制软件在环仿真批处理文件。
CarR1Diff_OffboardPos4.py	多辆无人车位置控制脚本。
CarR1Diff_HITLRun.bat	硬件在环批处理文件
CarR1Diff.dll	阿克曼底盘小车 DLL 模型文件

5. 运行环境

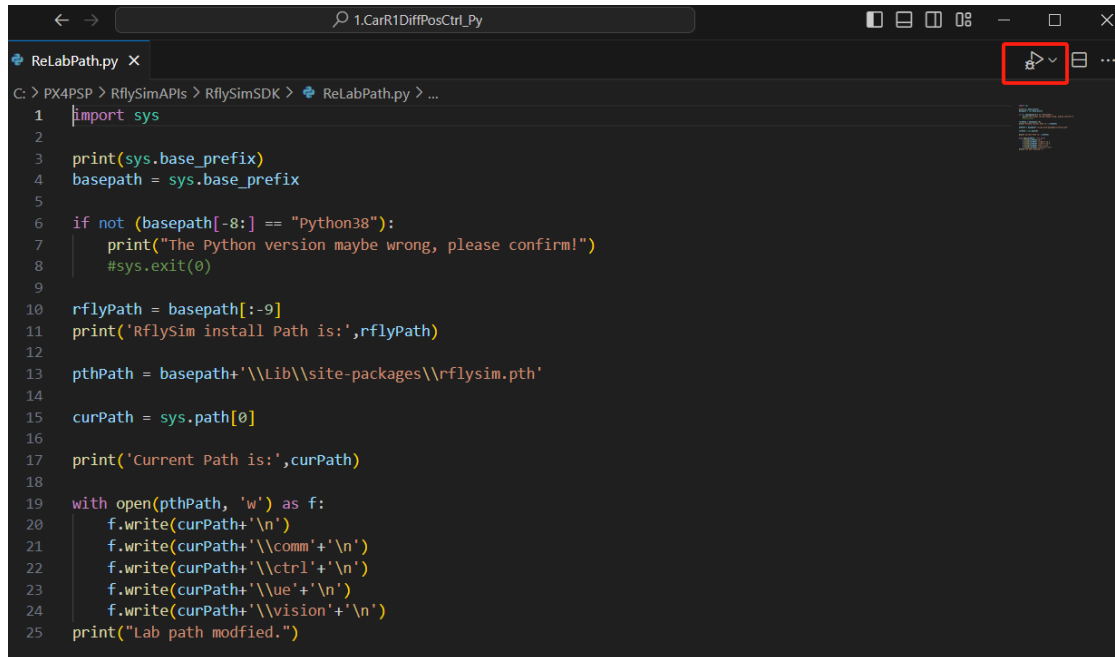
序号	软件要求	硬件要求	
		名称	数量
1	Windows 10 及以上版本	笔记本/台式电脑 ^①	1
2	RflySim 平台免费版	Pixhawk 6C ^②	1
3	Python	数据线	1

① 推荐配置请见：<https://doc.rflysim.com/1.1InstallMethod.html>
② 须保证平台安装时的编译命令为：px4_fmuv6c_default，固件版本为：1.13.3。其他配套飞控请见：<http://doc.rflysim.com/hardware.html>

6. 实验步骤

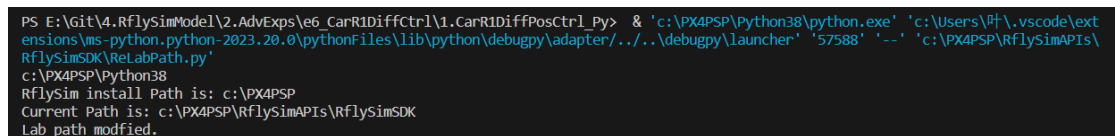
6.1 Python 库文件部署

以 VsCode 打开 “C:\PX4PSP\RflySimAPIs\RflySimSDK\ ReLabPath.py”，并运行。



```
1 import sys
2
3 print(sys.base_prefix)
4 basepath = sys.base_prefix
5
6 if not (basepath[-8:] == "Python38"):
7     print("The Python version maybe wrong, please confirm!")
8     #sys.exit(0)
9
10 rflyPath = basepath[:-9]
11 print('RflySim install Path is:', rflyPath)
12
13 pthPath = basepath+'\\Lib\\site-packages\\rflysim.pth'
14
15 curPath = sys.path[0]
16
17 print('Current Path is:', curPath)
18
19 with open(pthPath, 'w') as f:
20     f.write(curPath+'\n')
21     f.write(curPath+'\\comm'+'\n')
22     f.write(curPath+'\\ctrl'+'\n')
23     f.write(curPath+'\\ue'+'\n')
24     f.write(curPath+'\\vision'+'\n')
25 print("Lab path modified.")
```

完成 Python 公共库环境部署。



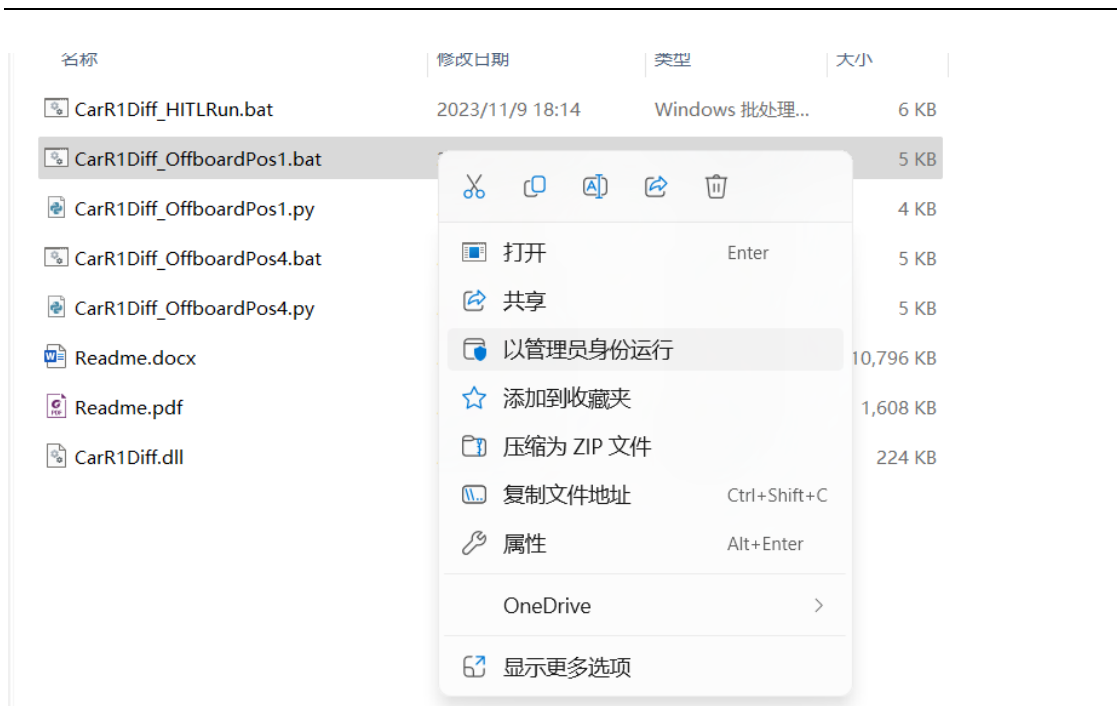
```
PS E:\Git\4.RflySimModel\2.AdvExps\6.CarR1DiffCtrl\1.CarR1DiffPosCtrl_Py> & 'c:\PX4PSP\Python38\python.exe' 'c:\Users\Hf\..vscode\extensions\ms-python.python-2023.20.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '57588' '--' 'c:\PX4PSP\RflySimAPIs\RflySimSDK\ReLabPath.py'
c:\PX4PSP\Python38
RflySim install Path is: c:\PX4PSP
Current Path is: c:\PX4PSP\RflySimAPIs\RflySimSDK
Lab path modified.
```

6.2 软件在环仿真

6.2.1 单辆无人车仿真

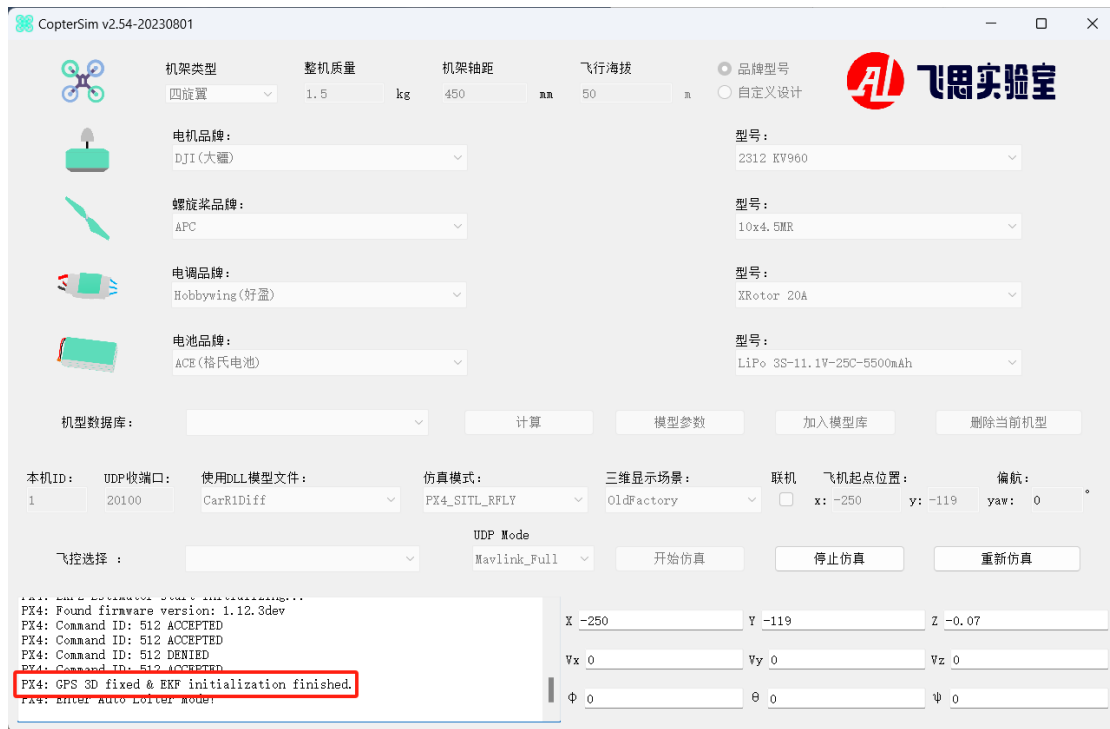
Step 1:

右键以管理员身份运行 CarR1Diff_OffboardPos1.bat 批处理文件。



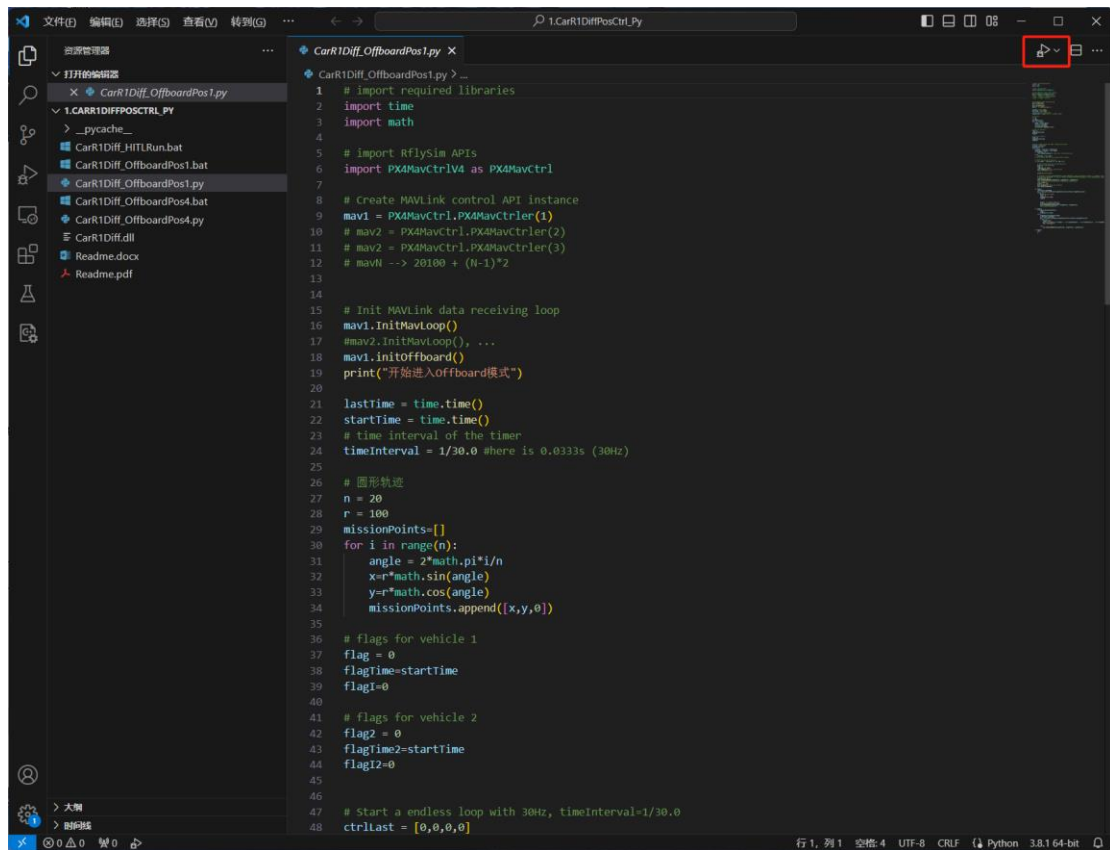
Step 2:

等待 CopterSim 中显示 “GPS 3D fixed &EKF initialization finished.”，表明已完成仿真初始化。



Step 3:

在 VsCode 中打开 CarR1Diff_OffboardPos1.py 脚本，并点击运行该脚本。



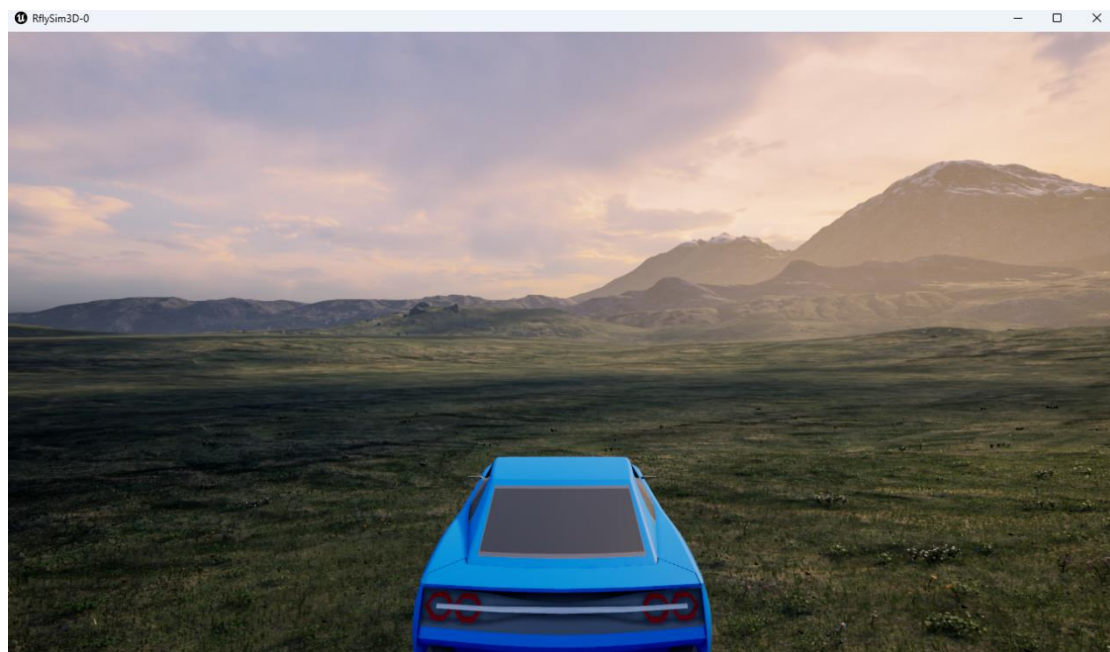
The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named '1.CarR1DFFPosCtrl_Py' with files like 'CarR1DFFPosCtrl_Py', 'CarR1DFFPosCtrl.bat', 'CarR1DFFPosCtrl.py', 'CarR1DFFPosCtrl4.bat', 'CarR1DFFPosCtrl4.py', 'CarR1DFF.dll', 'Readme.docx', and 'Readme.pdf'. The code editor shows a Python script named 'CarR1DFFPosCtrl_Py' with the following code:

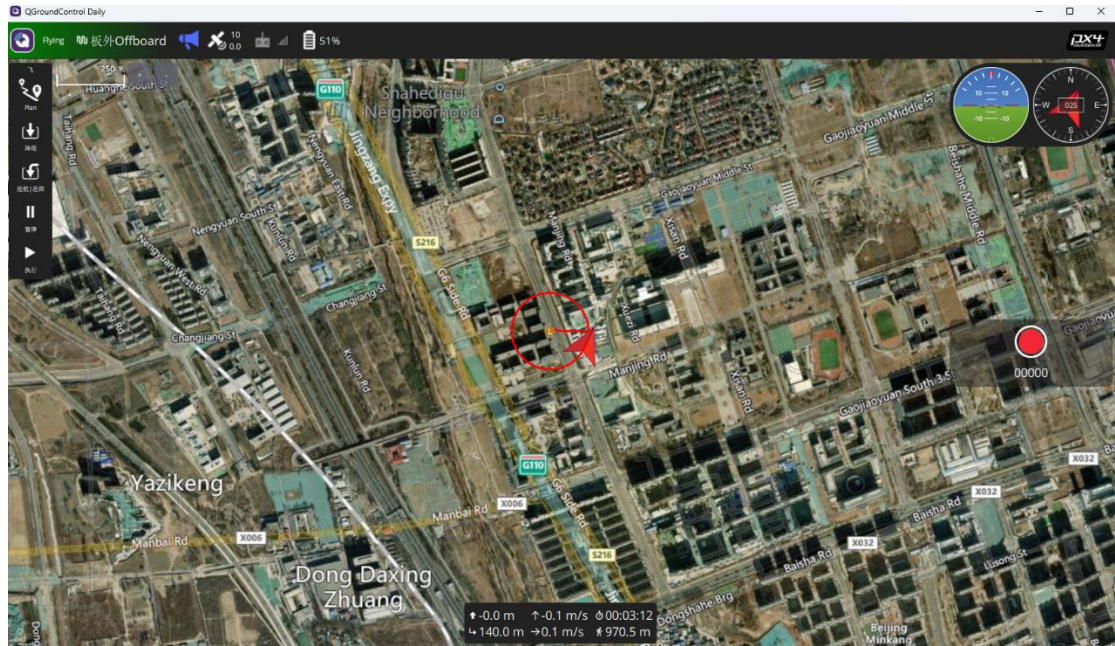
```
1 # import required libraries
2 import time
3 import math
4
5 # import RflySim APIs
6 import PX4MavCtrlV4 as PX4MavCtrl
7
8 # Create MAVLink control API instance
9 mav1 = PX4MavCtrl.PX4MavCtrl(1)
10 # mav2 = PX4MavCtrl.PX4MavCtrl(2)
11 # mav2 = PX4MavCtrl.PX4MavCtrl(3)
12 # mavN --> 20100 + (N-1)*2
13
14
15 # Init MAVLink data receiving loop
16 mav1.InitMavLoop()
17 #mav2.InitMavLoop(), ...
18 mav1.InitOffboard()
19 print("开始进入offboard模式")
20
21 lastTime = time.time()
22 startTime = time.time()
23 # time interval of the timer
24 timeInterval = 1/30.0 #here is 0.0333s (30Hz)
25
26 # 圆形轨迹
27 n = 20
28 r = 100
29 missionPoints=[]
30 for i in range(n):
31     angle = 2*math.pi*i/n
32     x=r*math.sin(angle)
33     y=r*math.cos(angle)
34     missionPoints.append([x,y,0])
35
36 # flags for vehicle 1
37 flag = 0
38 flagTime=startTime
39 flagI=0
40
41 # flags for vehicle 2
42 flag2 = 0
43 flagTime2=startTime
44 flagI2=0
45
46
47 # start a endless loop with 30Hz, timeInterval=1/30.0
48 ctrlLast = [0,0,0,0]
```

The status bar at the bottom indicates '行 1, 列 1 空格 4 UTF-8 CRLF Python 3.8.1 64-bit'.

Step 4:

在 RflySim3D 中观察无人车运行状态，观察 QGC 中无人车的运动轨迹 是否为圆形。

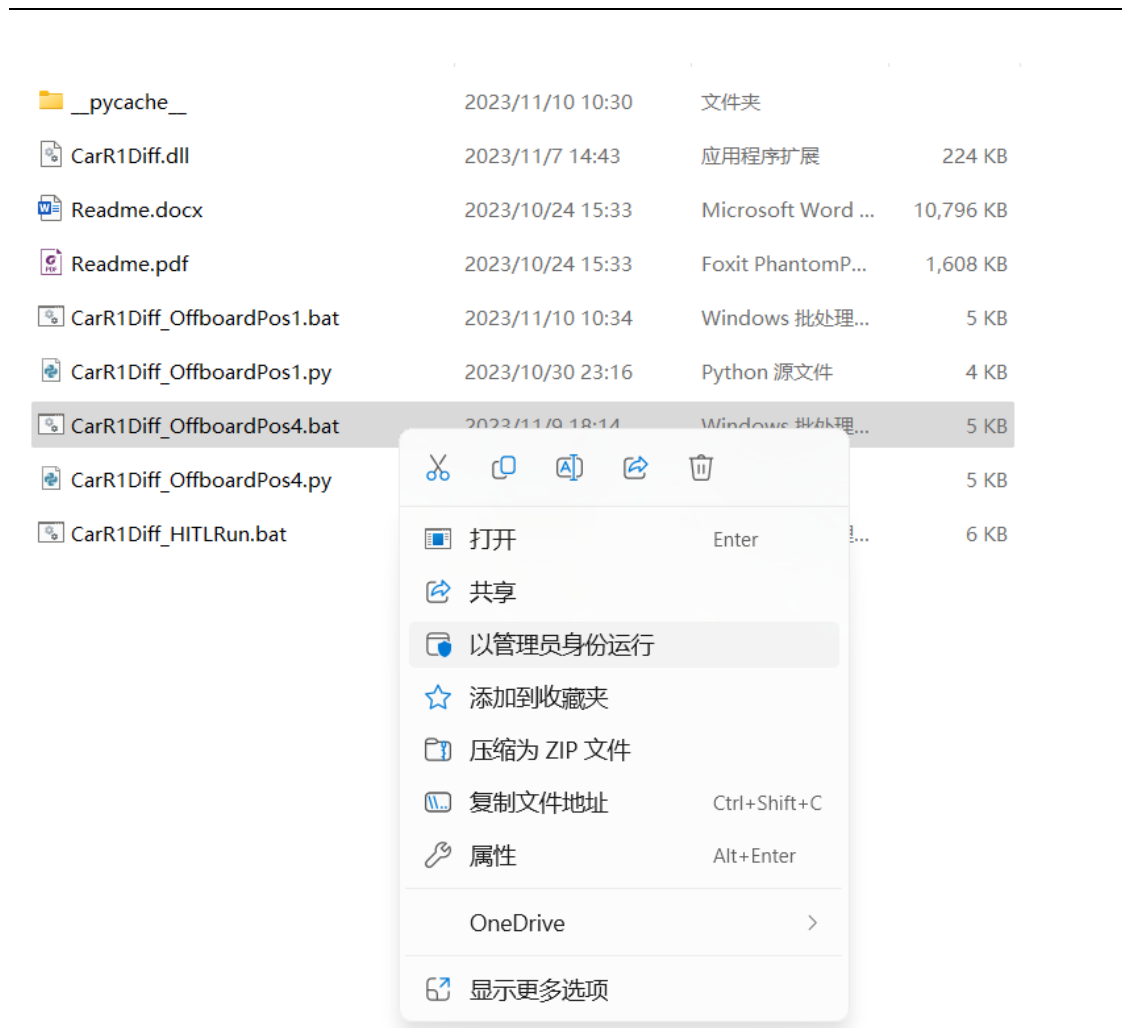




5.2.2、多辆无人车仿真

Step 1:

右键以管理员身份运行 CarR1Diff_OffboardPos4.bat 批处理文件。



Step 2:

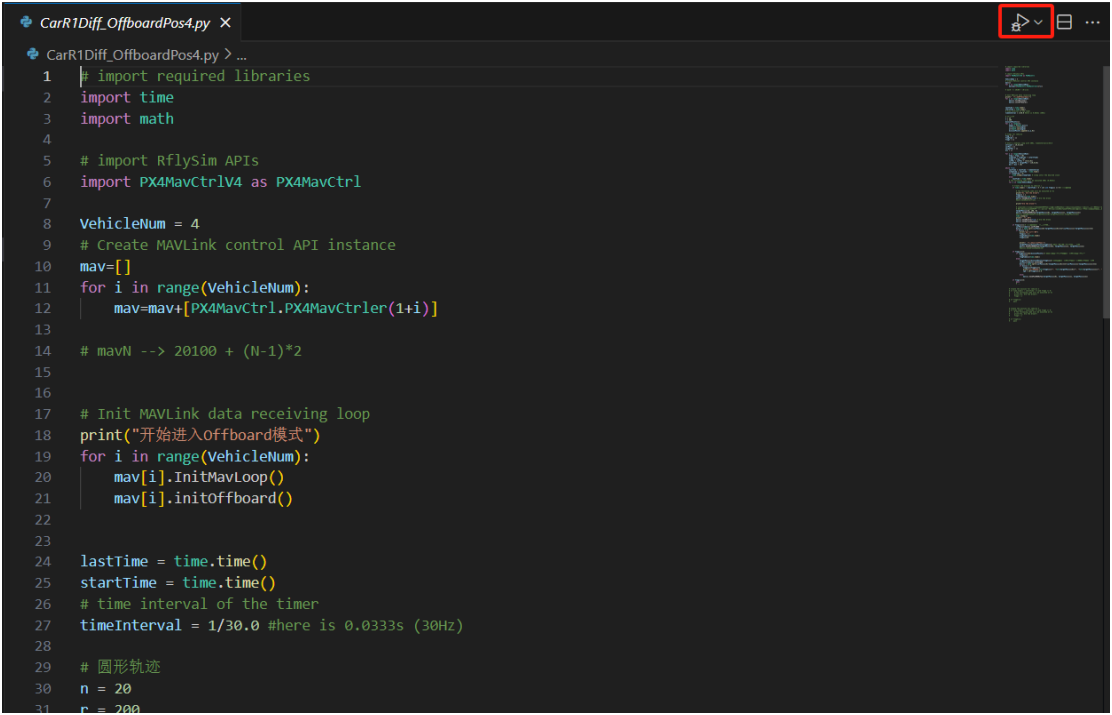
等待四辆无人车初始化完成。



Step 3:

右键以 VsCode 打开并运行 CarR1Diff_OffboardPos4.py 文件，

名称	修改日期	类型	大小
__pycache__	2023/11/10 10:30	文件夹	
CarR1Diff.dll	2023/11/7 14:43	应用程序扩展	224 KB
Readme.docx	2023/10/24 15:33	Microsoft Word ...	10,796 KB
Readme.pdf	2023/10/24 15:33	Foxit PhantomP...	1,608 KB
CarR1Diff_OffboardPos1.bat	2023/11/10 10:34	Windows 批处理...	5 KB
CarR1Diff_OffboardPos1.py	2023/10/30 23:16	Python 源文件	4 KB
CarR1Diff_OffboardPos4.bat	2023/11/9 18:14	Windows 批处理...	5 KB
CarR1Diff_OffboardPos4.py	2023/10/30 23:16	Python 源文件	5 KB
CarR1Diff_HITLRun.bat	2023/11/9 18:14	Windows 批处理...	6 KB



```
1 # import required libraries
2 import time
3 import math
4
5 # import RflySim APIs
6 import PX4MavCtrlV4 as PX4MavCtrl
7
8 VehicleNum = 4
9 # Create MAVLink control API instance
10 mav=[]
11 for i in range(VehicleNum):
12     mav=mav+[PX4MavCtrl.PX4MavCtrl(1+i)]
13
14 # mavN --> 20100 + (N-1)*2
15
16
17 # Init MAVLink data receiving loop
18 print("开始进入Offboard模式")
19 for i in range(VehicleNum):
20     mav[i].InitMavLoop()
21     mav[i].initOffboard()
22
23
24 lastTime = time.time()
25 startTime = time.time()
26 # time interval of the timer
27 timeInterval = 1/30.0 #here is 0.0333s (30Hz)
28
29 # 圆形轨迹
30 n = 20
31 r = 200
```

Step 4:

观察 QGC 和 RflySim3D 中无人车的运动轨迹是否为圆形，如下图所示。



6.3 硬件在环仿真

6.3.1 飞控硬件设置

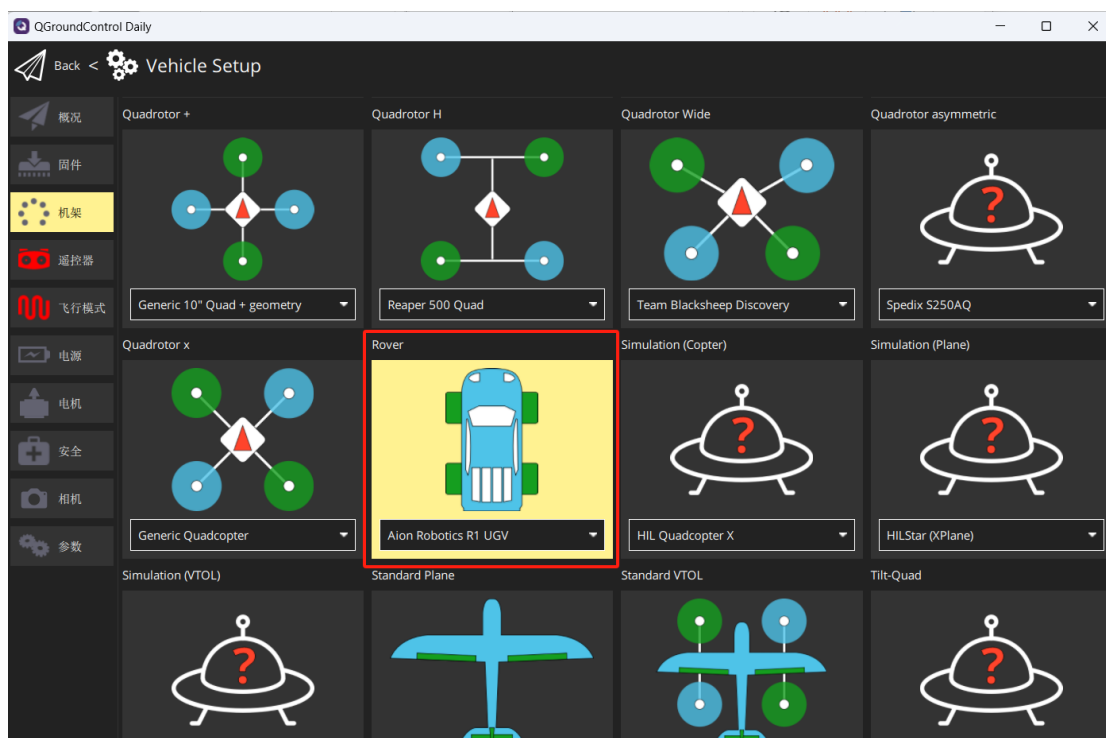
Step 1:

按下图所示将飞控与计算机链接，飞控上的接口名称为 USB。



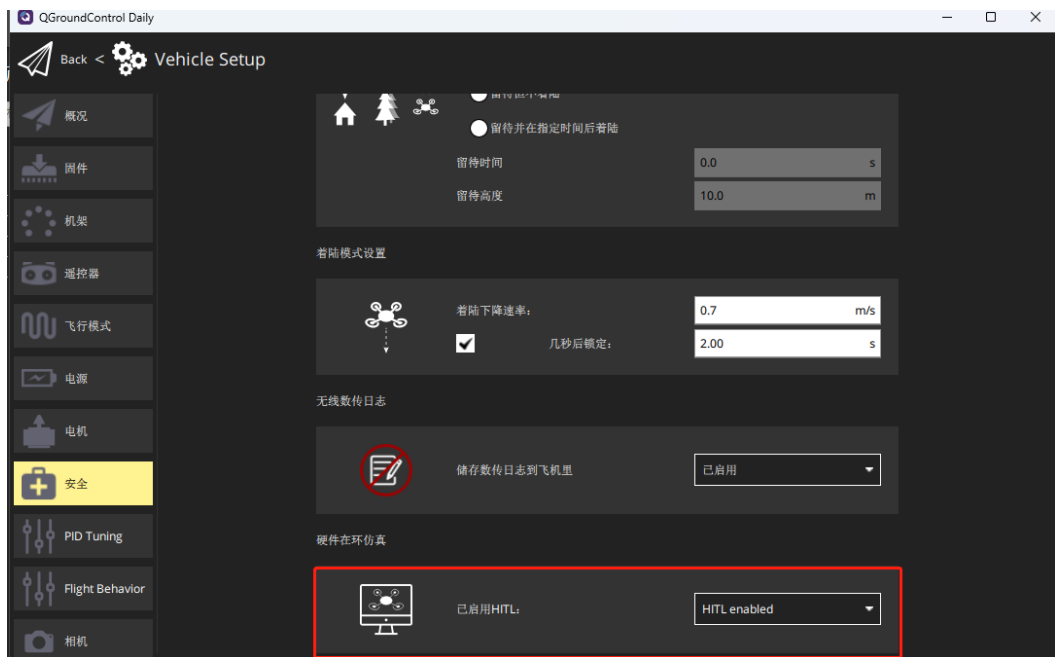
Step 2:

推荐使用 Pixhawk 6C 飞控进行硬件在环仿真，将飞控烧录至 1.13.3 固件版本，机架设置为“Aion Robotics R1 UGV”，点击 QGC 右上角的“应用并重启”。



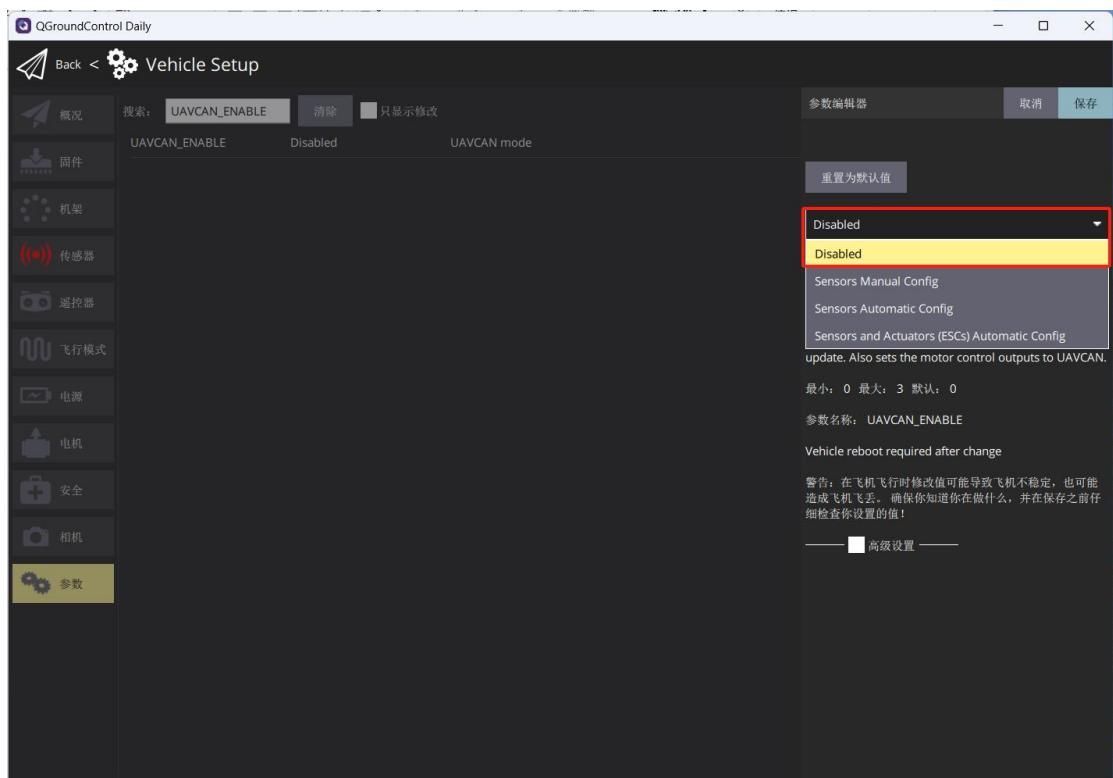
Step 3:

点击“安全”，设置硬件在环仿真为“HITL enabled”，重新插拔飞控。

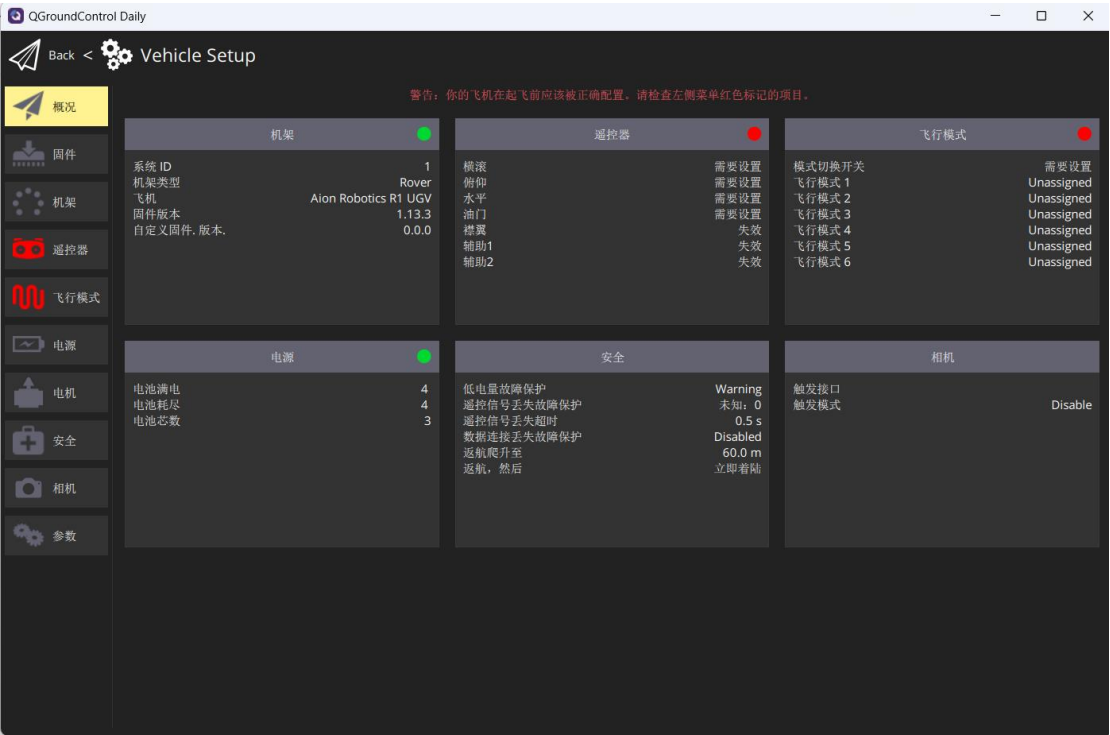


Step 4:

点击“参数”，在搜索栏中输入“UAVCAN_ENABLE”，在弹出框中设置为“Disabled”，保存后重新插拔飞控即可。



下图为完成硬件在环仿真相关配置后的示意图。



6.3.2 单辆无人车仿真

Step 1:

右键以管理员身份运行 CarR1Diff_HITLRun.bat 批处理文件。

文件夹	2023/11/10 10:30	文件夹	
CarR1Diff.dll	2023/11/7 14:43	应用程序扩展	224 KB
Readme.docx	2023/10/24 15:33	Microsoft Word ...	10,796 KB
Readme.pdf	2023/10/24 15:33	Foxit PhantomP...	1,608 KB
CarR1Diff_OffboardPos1.bat	2023/11/10 10:34	Windows 批处理...	5 KB
CarR1Diff_OffboardPos1.py	2023/10/30 23:16	Python 源文件	4 KB
CarR1Diff_OffboardPos4.bat	2023/11/9 18:14	Windows 批处理...	5 KB
CarR1Diff_OffboardPos4.py	2023/10/30 23:16	Python 源文件	5 KB
CarR1Diff_HITLRun.bat	2023/11/9 18:14	Windows 批处理...	6 KB

Step 2:

在终端中根据提示输入串口号，启动一辆无人车的仿真。

```
CAWindows\System32\cmd.exe
已复制 1 个文件。
-----
Please input the Pixhawk COM port list for HITL
Use ',' as the separator if more than one Pixhawk
E.g., input 3 for COM3 of Pixhawk on the computer
Input 3,6,7 for COM3, COM6 and COM7 of Pixhawks

Available COM ports on this computer are:
COM7: MindPX

Recommended COM list input is: 7

My COM list for HITL simulation is:5
Kill all CopterSims
Start QGroundControl
请按任意键继续. . .
```

Step 3:

之后的与单辆无人车软件在环仿真中的 Step3 到 Step4 相同，可进行位置控制仿真，运行后可在 QGC 中观察运行轨迹。

6.3.3 多辆无人车仿真

Step 1:

4 辆无人车以管理员身份运行 CarR1Diff_HITLRun.bat 脚本，然后输入 4 个飞控的串口号敲击回车就可以连接 4 架无人车。

名称	修改日期	类型	大小
 CarNoCtrl.dll	2022/8/16 23:22	应用程序扩展	226 KB
 CarNoCtrlHITLRun	2022/9/20 17:07	Windows 批处理...	6 KB
 CarNoCtrlOffboardPos1	2022/9/20 17:07	Windows 批处理...	5 KB
 CarNoCtrlOffboardPos1	2022/8/15 13:13	PY 文件	4 KB
 CarNoCtrlOffboardPos4	2022/9/20 17:07	Windows 批处理...	5 KB
 CarNoCtrlOffboardPos4	2022/8/15 13:25	PY 文件	5 KB
 PX4MavCtrlV4	2023/6/7 10:43	PY 文件	137 KB

Step 2:

之后步骤与多辆无人车软件在环仿真中的 Step3 到 Step4 相同，可进行位置控制仿真，运行后可在 RflySim3D 中观察运行轨迹。

7. 参考资料

- [1]. DLL/SO 模型与通信接口 [..\..\API.pdf](#)
- [2]. 外部控制接口 [..\..\API.pdf](#)
- [3].

8. 常见问题

Q1.

A1.