

# 1. 实验名称及目的

## 1.1 实验名称

基于系统模板的固定翼模型验证（python航点控制）

## 1.2 实验目的

该例程通过平台固定翼位置控制接口，在软硬件在环仿真过程中让固定翼往期望航点飞行。

## 1.3 关键知识点

### 软/硬件在环仿真的实现原理

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[PX4PSP\RflySimAPIs\4.RflySimModel\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。

**注意**，针对仿真时的机架设置，在进行软件在环仿真时，需要在对应的bat脚本中set PX4SITLFrame= Standard Plane，硬件在环时需要在QGC机架设置页面进行相同设置。

## 外部控制接口 (Python)

部分代码如下：

### 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**，可在此处启用外部控制模式，也可在到达初始位置后启用，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。

## ■ 运行阶段

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机应该执行哪些动作。

### 当 flag == 0 时，解锁固定翼

解锁车辆

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

设定启动目标点

```
targetPos=[200, 0, -100]
```

```
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，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，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，会脱离圆形轨迹。

## 2. 实验效果

在平台进行软硬件在环仿真，通过平台固定翼位置控制接口实现固定翼飞行轨迹画圆。

## 3. 文件目录

例程目录：

[安装目录]\RflySimAPIs\4.RflySimModel\2.AdvExps\e3\_FWingModelCtrl\2.FWPosCtrlAPI

文件夹/文件名称	说明
<a href="#">AircraftMathworksMavlinkSITLRun.bat</a>	软件在环仿真批处理文件。
<a href="#">AircraftMathworksMavlinkHITLRun.bat</a>	硬件在环仿真批处理文件。
<a href="#">AircraftMathworksController.py</a>	单个固定翼固定轨迹控制脚本。
<a href="#">AircraftMathworksController3.py</a>	多个固定翼固定轨迹控制脚本。
AircraftMathworks.dll	固定翼无人机DLL模型文件

## 4. 运行环境

### 4.1 软件要求

Windows 10及以上版本；RflySim工具链；Python。

①：若使用Pixhawk 6X飞控，平台安装时的编译命令为：px4\_fmu-v6x\_default，推荐PX4固件版本为：1.12.3。其他配套飞控及编译命令请见：

<https://rflysim.com/doc/zh/1/Hardware.html>

## 4.2 硬件要求

笔记本/台式电脑① 1台；Pixhawk 6X或其它飞控 1台；数据线 1台。

①：推荐配置请见：<https://rflysim.com/>

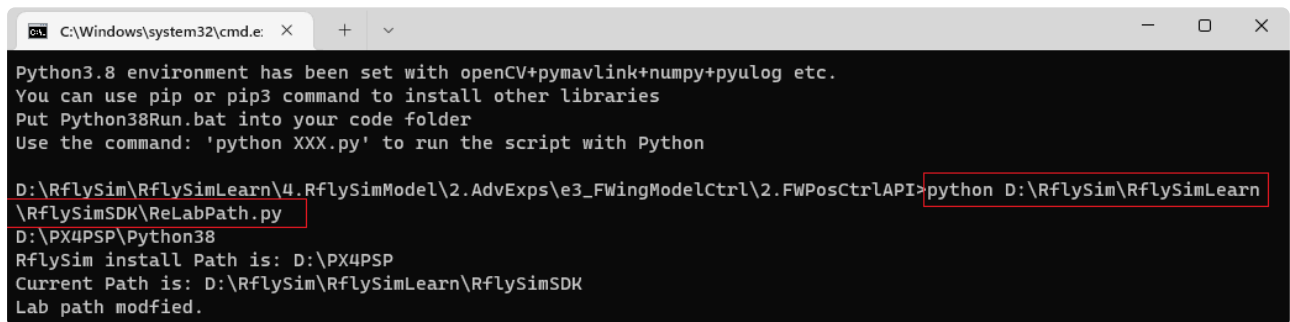
## 5. 实验步骤

### 5.1. Python库文件部署

在文件夹下，双击 `Python38Run.bat`，打开集成好的python环境，在该环境下运行 `ReLabPath.py`文件，输入

```
python C:\PX4PSP\RflySimAPIs\RflySimSDK\
```

```
ReLabPath.py，注意输入要根据实际路径。回车，完成Python公共库环境部署。
```



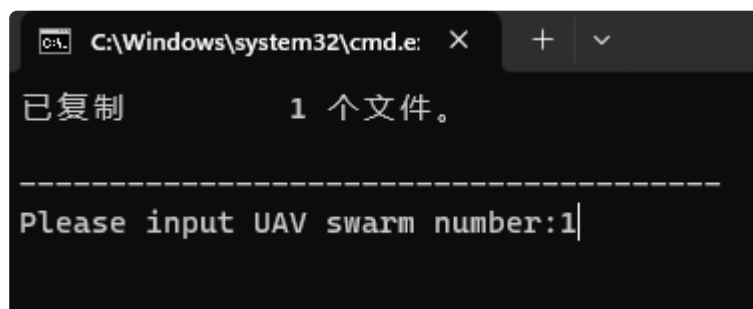
```
C:\Windows\system32\cmd.e: X + v
Python3.8 environment has been set with openCV+pymavlink+numpy+pyulog etc.
You can use pip or pip3 command to install other libraries
Put Python38Run.bat into your code folder
Use the command: 'python XXX.py' to run the script with Python
D:\RflySim\RflySimLearn\4.RflySimModel\2.AdvExps\e3_FWingModelCtrl\2.FWPosCtrlAPI>python D:\RflySim\RflySimLearn
\RflySimSDK\ReLabPath.py
D:\PX4PSP\Python38
RflySim install Path is: D:\PX4PSP
Current Path is: D:\RflySim\RflySimLearn\RflySimSDK
Lab path modified.
```

### 5.2. 必做实验：软件在环仿真

#### 5.2.1 单架固定翼仿真

##### Step 1: 启动仿真

右键以管理员身份运行 `AircraftMathworksMavlinkSITLRun.bat` 批处理文件，在弹出的终端窗口中输入1，启动1架飞机的软件在环仿真。



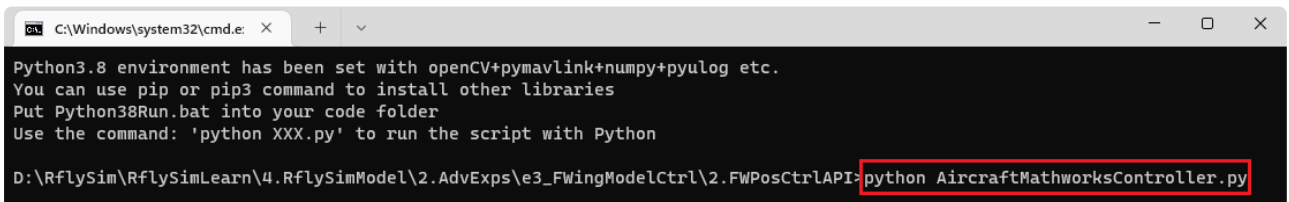
```
C:\Windows\system32\cmd.e: X + v
已复制 1 个文件。
-----
Please input UAV swarm number:1
```

## Step 2: 等待初始化完成



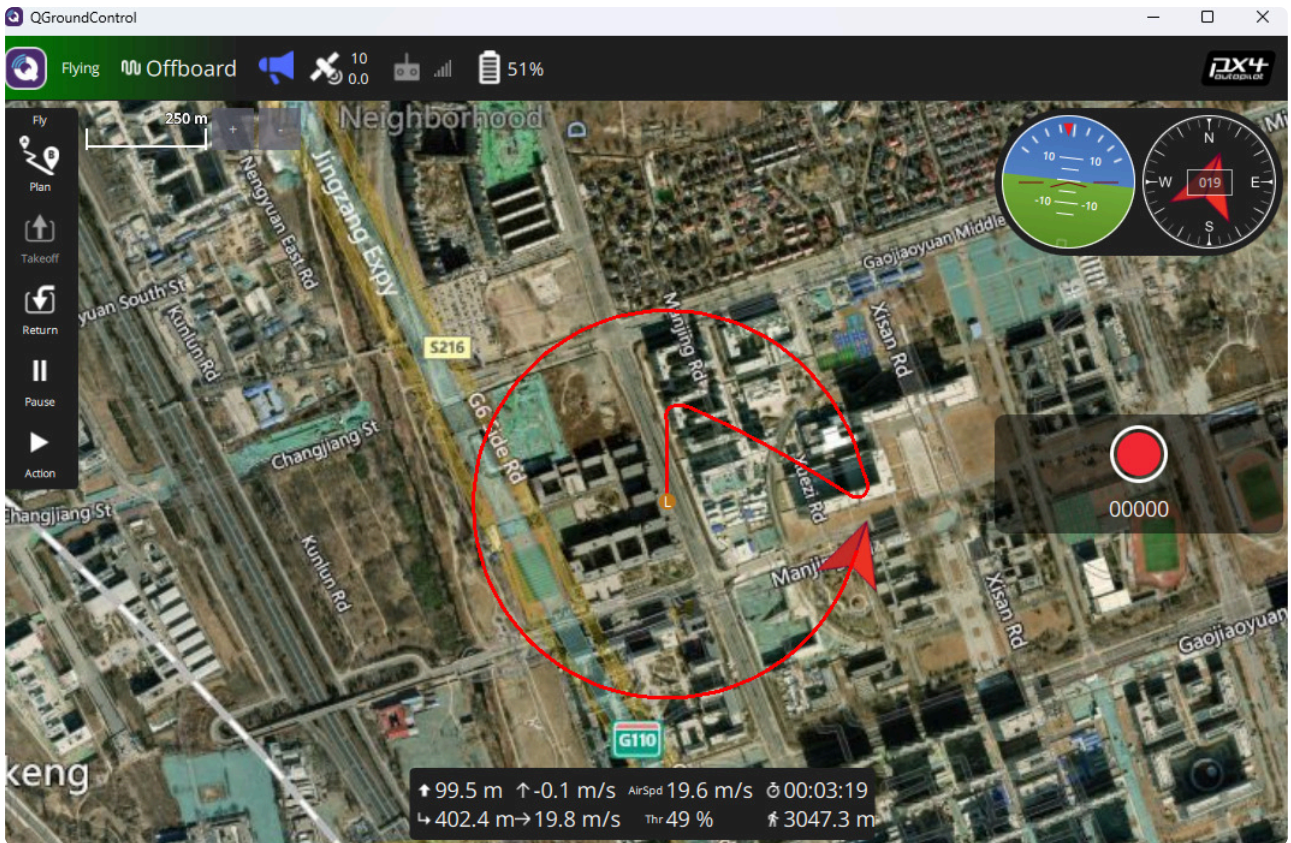
## Step 3: 运行控制程序

在文件夹下，双击 [Python38Run.bat](#)，打开集成好的python环境，在该环境下运行 [AircraftMathworksController.py](#) 文件，输入 `python AircraftMathworksController.py`



## Step 4: 观察结果

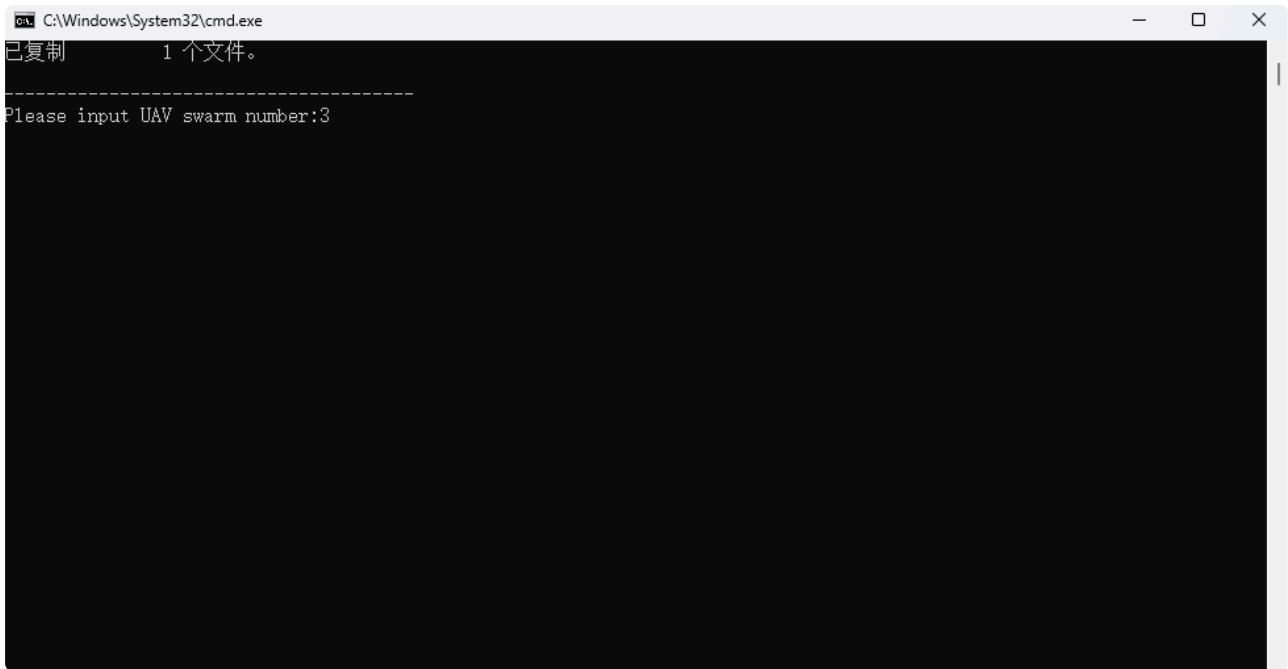
在RflySim3D中观察飞机运行状况，并在QGC地面站中观察运动轨迹。



## 5.2.2 多架固定翼仿真

### Step 1: 启动仿真

管理员身份运行 `AircraftMathworksMavlinkSITLRun.bat` 文件后，在弹出的终端窗口中输入3.



## Step 2: 等待初始化完成

等待三架固定翼飞机均完成初始化。



## Step 3: 运行控制程序

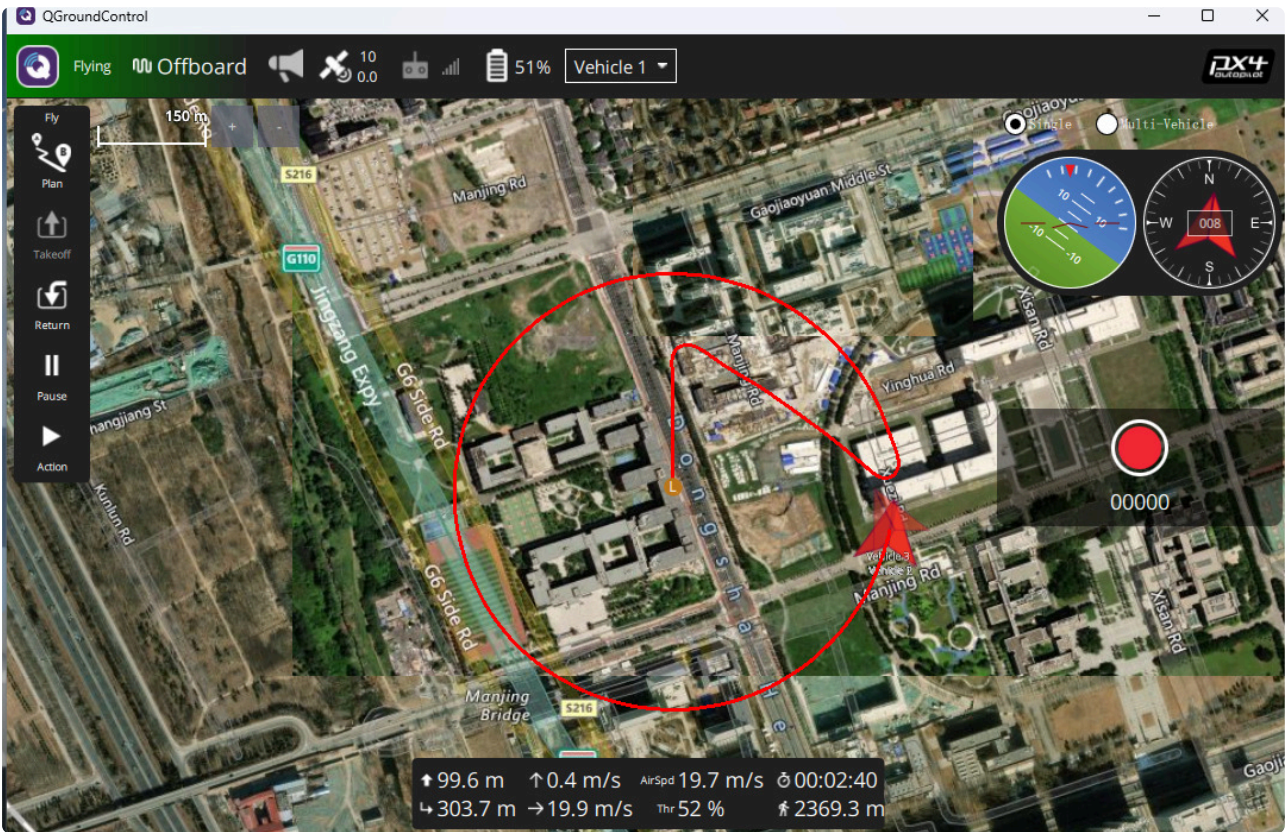
在文件夹下，双击 [Python38Run.bat](#)，打开集成好的python环境，在该环境下运行 [AircraftMathworksController3.py](#) 文件，输入 `python AircraftMathworksController3.py`

```
C:\Windows\system32\cmd.e. x + v
Python3.8 environment has been set with openCV+pymavlink+numpy+pyulog etc.
You can use pip or pip3 command to install other libraries
Put Python38Run.bat into your code folder
Use the command: 'python XXX.py' to run the script with Python

D:\RflySim\RflySimLearn\4.RflySimModel\2.AdvExps\e3_FWingModelCtrl\2.FWPosCtrlAPI-python AircraftMathworksController3.py
```

## Step 4: 观察结果

可从QGC处看到固定翼按期望轨迹画圆。



## 5.3. 选做实验：硬件在环仿真












### Step 1: 连接飞控

如下图所示，将飞控通过USB线连接电脑，并确保完成硬件在环仿真配置。注意，本图使用Pixhawk6x飞控，其他飞控配置方法类似（推荐使用Pixhawk飞控）。

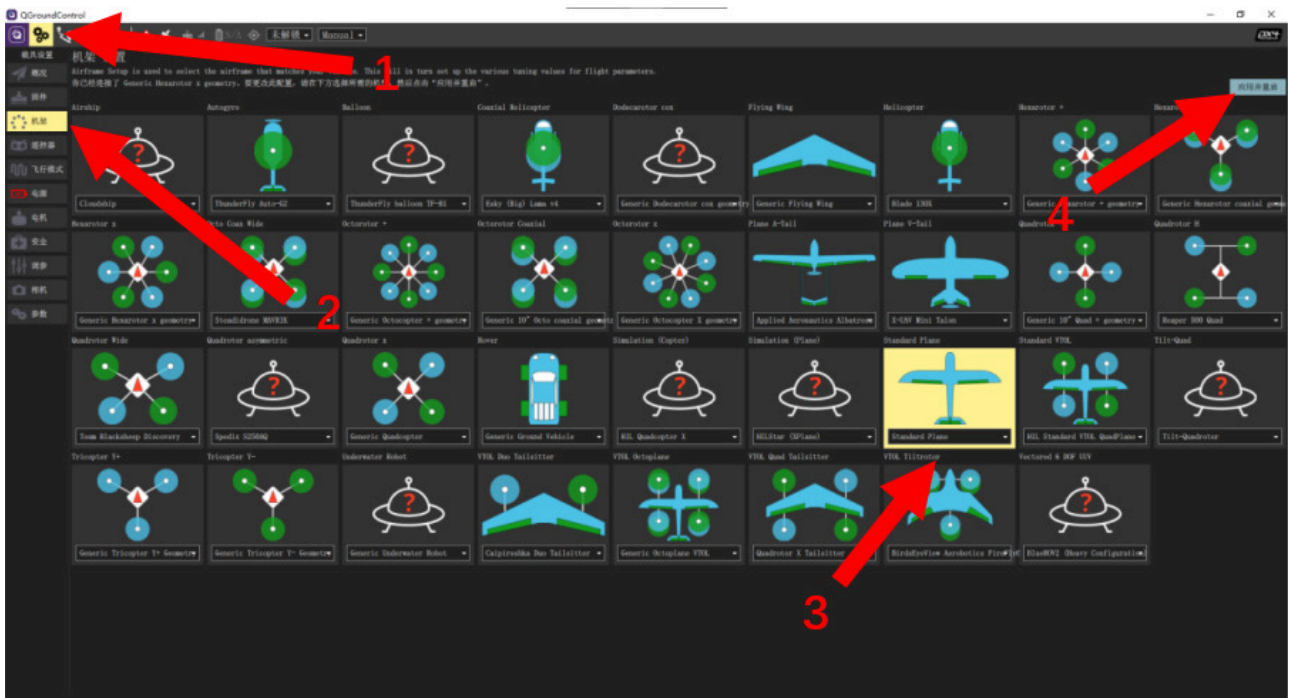


## Step 2: 设置硬件在环机架

在 Rflytools 文件夹中打开 QGC 地面站。

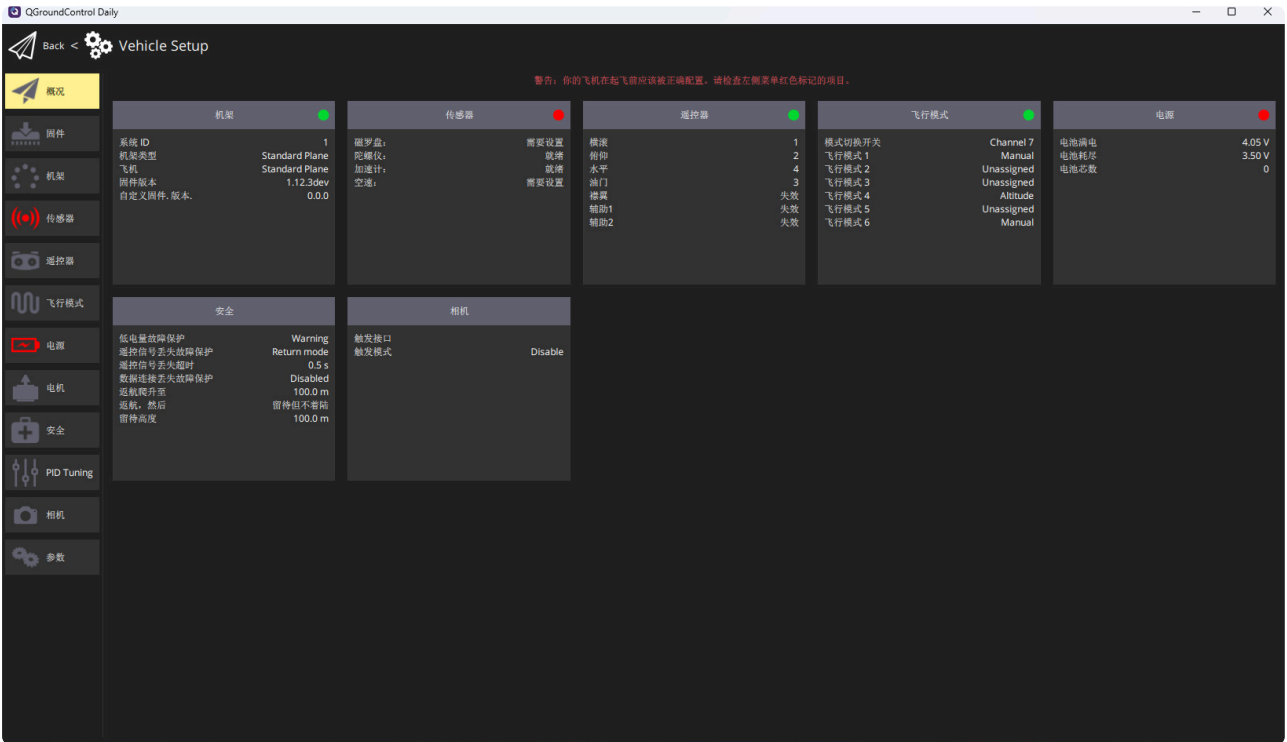
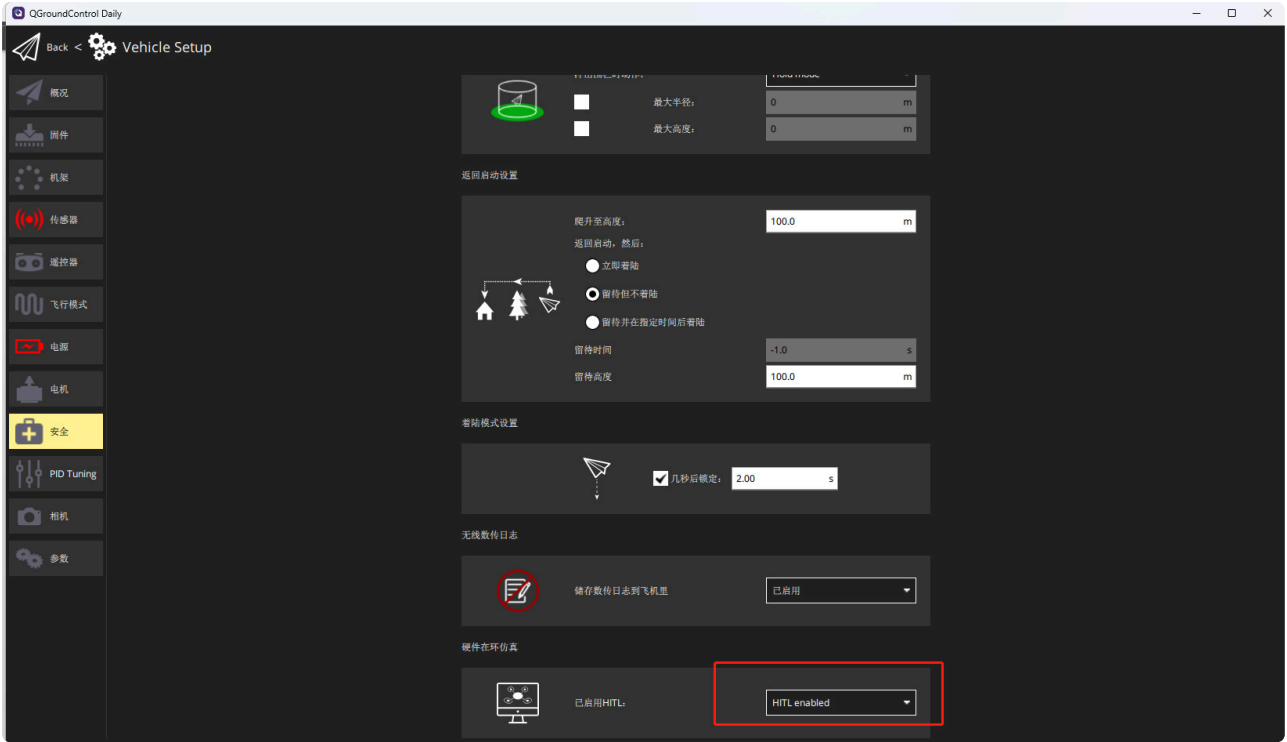
 3DDisplay	2023/7/27 15:02	快捷方式	1 KB
 CopterSim	2023/7/27 15:02	快捷方式	1 KB
 FlightGear-F450	2023/7/27 15:02	快捷方式	2 KB
 HITLRun	2023/7/27 15:02	快捷方式	2 KB
 Python38Env	2023/7/27 15:02	快捷方式	2 KB
 QGroundControl	2023/7/27 15:02	快捷方式	1 KB
 RflySim3D	2023/7/27 15:02	快捷方式	1 KB
 RflySimAPIs	2023/7/27 15:02	快捷方式	1 KB
 RflySimUE5	2023/7/27 15:02	快捷方式	1 KB
 SITLRun	2023/7/27 15:02	快捷方式	2 KB
 Win10WSL	2023/7/27 15:02	快捷方式	2 KB

在机架界面设置机架型号为“Standard Plane”，设置完毕后点击右侧“应用并重启”。



## Step 3: 配置硬件在环参数

在“安全”界面，选择“HITL enabled”启动硬件在环仿真，之后在概况界面中确认配置完成后，重新插拔飞控完成设置。



## Step 4: 启动仿真

双击运行“AircraftMathworksHITLRun.bat”批处理文件，在弹出的终端窗口中根据提示输入串口号，启动硬件在环仿真。

名称	修改日期	类型	大小
AircraftMathworks.dll	2024/7/25 13:25	应用程序扩展	255 KB
AircraftMathworksController.py	2024/7/25 13:25	Python File	5 KB
AircraftMathworksController3.py	2024/7/25 13:25	Python File	4 KB
AircraftMathworksMavlinkHITLRun.bat	2024/8/16 0:14	Windows 批处理...	6 KB
AircraftMathworksMavlinkSITLRun.bat	2024/8/16 0:14	Windows 批处理...	6 KB
dir.xlsx	2024/7/25 13:25	XLSX 工作表	11 KB
Python38Run.bat	2024/8/16 22:48	Windows 批处理...	1 KB
Readme.docx	2024/8/22 10:50	Microsoft Word ...	8,585 KB
Readme.pdf	2024/8/21 9:15	WPS PDF 文档	1,793 KB

```
C:\Windows\system32\cmd.e: X + v
已复制 1 个文件。
-----
Please input the Pixhawk COM port list for HIL
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:
COM3: ??????????
COM4: ??????????
COM5: USB ???
-----
Recommended COM list input is: 3,4,5
-----
My COM list for HITL simulation is:5|
```

## Step 5: 仿真过程

之后测试步骤与5.2.1相同，运行Python文件后可在QGC与RflySim3D中观测运行状态与角度。

注意事项：在固定翼的offboard控制中，用到如下控制接口：

1. SendMavTakeOff：起飞指令。
2. SendPosNED：在北东地坐标系下发送目标位置。

3. SendCruiseSpeed: 发送盘旋速度。

4. SendCruiseRadius: 发送盘旋半径。

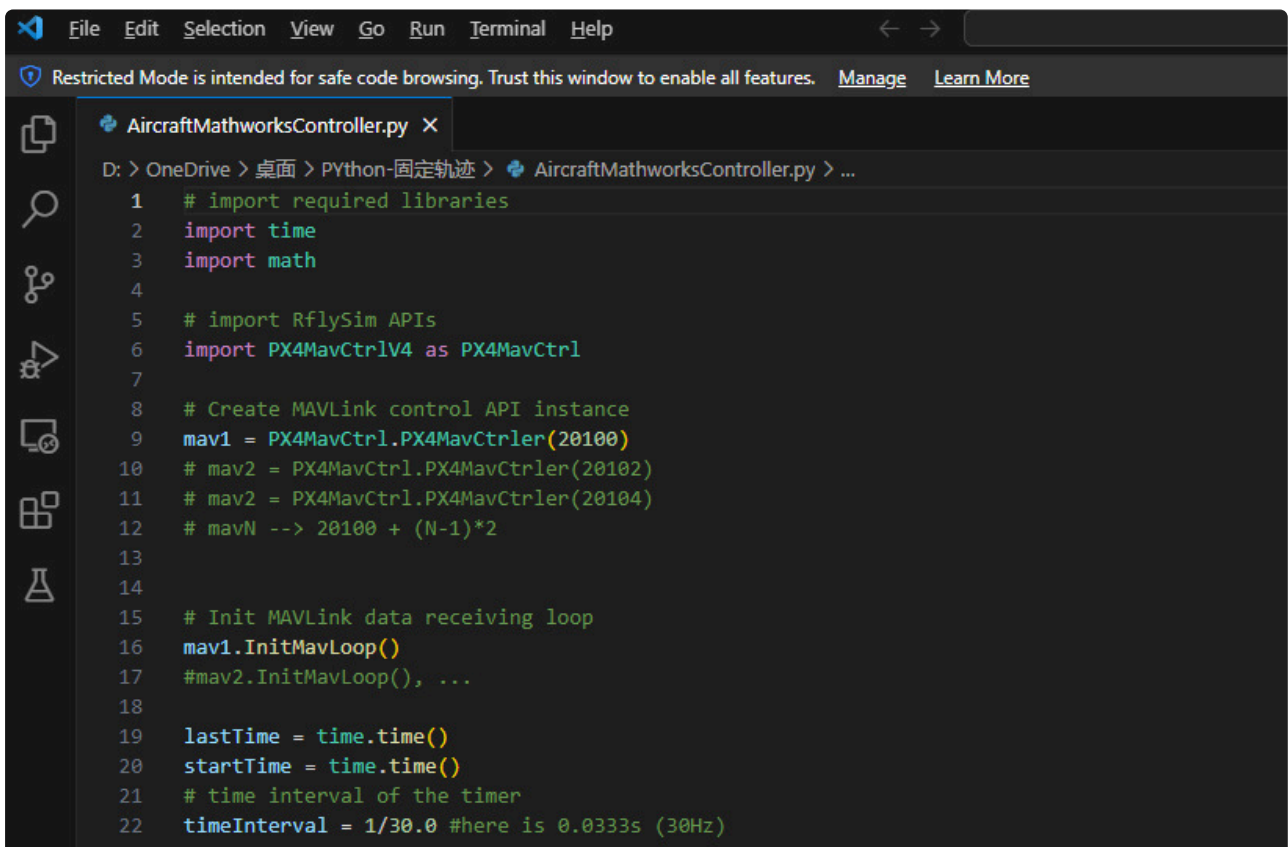
## 5.4. 选做实验 (VS Code调试运行)

### 准备工作

- 先确保已经按 [RflySimAPIs\1.RflySimIntro\2.AdvExps\3.PythonConfig\Readme.pdf](#) 步骤，正确配置VS Code环境。或者配置了自己的Pycharm等自定义Python环境。
- 其他步骤与上文相同，运行 [AircraftMathworksController.py](#) 时，可使用VS Code (或Pycharm等工具) 来打开 [AircraftMathworksController.py](#) 文件，并阅读代码，修改代码，调试执行等。

### 扩展实验

- 请自行使用VS Code阅读 [AircraftMathworksController.py](#) 源码，通过程序跳转，了解每条代码的执行原理；再通过调试工具，验证每条指令的执行效果。



```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
AircraftMathworksController.py X
D: > OneDrive > 桌面 > Python-固定轨迹 > AircraftMathworksController.py > ...
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(20100)
10 # mav2 = PX4MavCtrl.PX4MavCtrl(20102)
11 # mav2 = PX4MavCtrl.PX4MavCtrl(20104)
12 # mavN --> 20100 + (N-1)*2
13
14
15 # Init MAVLink data receiving loop
16 mav1.InitMavLoop()
17 #mav2.InitMavLoop(), ...
18
19 lastTime = time.time()
20 startTime = time.time()
21 # time interval of the timer
22 timeInterval = 1/30.0 #here is 0.0333s (30Hz)
23
```

## 6.参考资料

1. DLL/SO模型与通信接口..\..\..\PX4PSP\RflySimAPIs\4.RflySimModel\API.pdf
2. 外部控制接口..\..\..\PX4PSP\RflySimAPIs\4.RflySimModel\API.pdf
- 3.

## 7.常见问题

Q1:

A1:

Q2: 编译报错，无法加载库文件



A2: 这可能是由于安装平台时PX4PSP工具箱未更新到最新版，更新RflySim安装包后按照如下配置重新安装平台即可

Toolbox one-key installation script: RflySimA... — □ ×

(1) Software package installation directory  
C:\PX4PSP

(2) PX4 firmware compiling command: firmware versions <= PX4-1.8 use format px4fmu-v3\_default; >= PX4-1.9 use format px4\_fmu-v3\_default  
px4\_fmu-v6c\_default

(3) PX4 firmware version (1: PX4-1.7.3, ... , 6: PX4-1.12.3, 7: PX4-1.13.2, 8: PX4-1.14.4, 9: PX4-1.15.0)  
9

(4) PX4 firmware compiling toolchain (1: WinWSL[suitable for all versions], 2: Msys2[suitable for <= PX4-1.8], 3: Cygwin[for >=PX4-1.8])  
1

(5) Whether to reinstall PSP toolbox (yes to reinstall and no to remain current installation)  
yes

(6) Whether to reinstall the dependent software packages (CopterSim, QGroundControl, CopterSim, etc. About 5 minites)  
no

(7) Whether to reinstall the selected compiling toolchain (yes to reinstall and no to remain unchanged, about 5 minites)  
no

(8) Whether to reinstall the selected PX4 firmware source code (yes to reinstall and no to remain unchanged, about 5 minites)  
no

(9) Whether to pre-compile the selected firmware with the selected command (yes to compile and no to remain unchanged, about 5 minites)  
no

(10) Whether to block the actuator outputs in the PX4 firmware code ("yes" to use Simulink controller, "no" to use PX4 official controller)  
no

OK Cancel