

# 1. 实验名称及目的

## 1.1 实验名称

基于系统模板的直升机模型验证（python位置控制）

## 1.2 实验目的

通过该实验熟悉直升机Offboard位置控制接口的使用。

## 1.3 关键知识点

### 软/硬件在环仿真（SIL/HIL）的实现

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。底层控制器是实际控制飞行器硬件（如电机和传感器）的部分。
- 三维引擎：这部分负责生成和处理仿真的视觉效果，提供仿真环境的三维视图，使用户能够视觉上跟踪和分析飞行器的运动。
- 外部控制：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。支持通过UDP\_Full、UDP\_Simple、MAVLINK\_Full、MAVLINK\_Simple等链接模式，获取无人机的位置、速度、姿态信息，并对无人机的位置、速度、航向进行控制。

## ■ 载具坐标系

机体坐标系：原点

$O_b$

位于飞行器的重心；

$x$

轴（由

$x_b$

表示）指向飞机正前方，位于飞行器的对称平面内；

$y$

轴（由

$y_b$

表示）指向飞机右侧；

$z$

轴（由

$z_b$

表示）指向飞机下方。在机体坐标系中表示的坐标向量附加了一个下标

$b$

。

NED坐标系：原点

$O_n$

任意固定在地球表面的一点上；

$x$

轴（由

$x_n$



## ■ PX4机架对应的混控器

[添加一个新的机型 | PX4 自动驾驶用户指南](#)

- 详细的PX4机架文件配置参考 [\(v1.12\)](#)

直升机模型的机架配置在

\PX4PSP\Firmware\ROMFS\px4fmu\_common\init.d\airframes\16001\_helicopter中定义如下：

```
.$R/etc/init.d/rc.mc_defaults
```

执行rc.mc\_defaults脚本，它包含了旋翼飞行器的默认参数设置，可以用来设置一些基本的系统参数

```
param set-default MAV_TYPE 4
```

设置了参数MAV\_TYPE的默认值为4，在MAVLink协议中，MAV\_TYPE参数用于指定飞行器的类型，数值4代表直升机。

```
set PWM_OUT none
```

PWM（脉冲宽度调制）是飞控系统用来控制电机和舵机的一种信号。设置PWM输出为none表示这个配置不使用标准的PWM输出，这是因为直升机的控制方式与一般的多旋翼不同。

```
set MIXER blade130
```

设置了混控器配置为blade130（一个特定类型的直升机混控器配置）。在PX4中，混控器定义了飞控如何控制飞行器的不同电机或舵机。

## ■ 混控通道对应的执行器

[混控器和执行器 | PX4 自动驾驶用户指南](#)

- 详细的PX4混控文件逻辑见：[\(v1.12\)](#)
- 详细的映射过程可参考：[PX4混控器相关知识梳理-CSDN博客](#)

本例程中直升机的具体混控文件可参考\PX4PSP\Firmware\ROMFS\px4fmu\_common\

mixers\blade130.main.mix，其混控逻辑如下：

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.1 & 0 & -0.8 \\ 0.13054 & 0.1 & 0 & -0.8 \\ 0.13054 & 0.1 & 0 & -0.8 \\ 0 & 0 & 0.1 & -1 \end{bmatrix} \begin{bmatrix} \delta_c \\ \delta_l \\ \delta_p \\ \delta_t \end{bmatrix}$$

其中， $u_1, u_2, u_3$  是三个斜盘舵机的输出， $u_4$ 是尾舵机的输出， $\delta_c, \delta_l, \delta_p, \delta_t$  是集体俯仰，横滚，偏航，油门的控制输入。例如，当增加俯仰输入时，所有的斜盘舵机都会增加输出，从而提升直升机的升力；当增加横滚输入时，左右两个斜盘舵机的输出会相反变化，从而产生横滚力矩；当增加偏航输入时，尾舵机的输出会增加，从而产生偏航力矩。

## I 位置控制接口

在直升机的offboard位置控制中，主要用到如下控制接口：

1. SendMavTakeOff：起飞指令。
2. SendPosNEDNoYaw：发送目标位置指令(在北东地坐标系下)。
3. SendCruiseSpeed：发送巡航速度指令。

具体使用方法见 [HelicopterPos.py](#) 文件。

## I 2.实验效果

直升机软硬件在环仿真时，通过Offboard位置控制接口实现画圆的效果。

## I 3.文件目录

例程目录：

[安装目录]\RflySimAPIs\4.RflySimModel\2.AdvExps\e8\_Helicopter\2.HeliCopterPosCtrl

文件夹/文件名称	说明
HeliCopter.dll	直升机模型动态链接库
<a href="#">Helicopter_HITLRun.bat</a>	硬件在环仿真启动脚本
<a href="#">Helicopter_SITLRun.bat</a>	软件在环仿真启动脚本
HIL.params	参数增量文件
<a href="#">HelicopterPos.py</a>	Offboard位置控制接口例程
<a href="#">Python38Run.bat</a>	Python程序运行脚本

## 4. 运行环境

### 4.1 软件要求

Windows 10及以上版本；RflySim工具链；MATLAB 2017B及以上③。

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② 1台；数据线 1台。

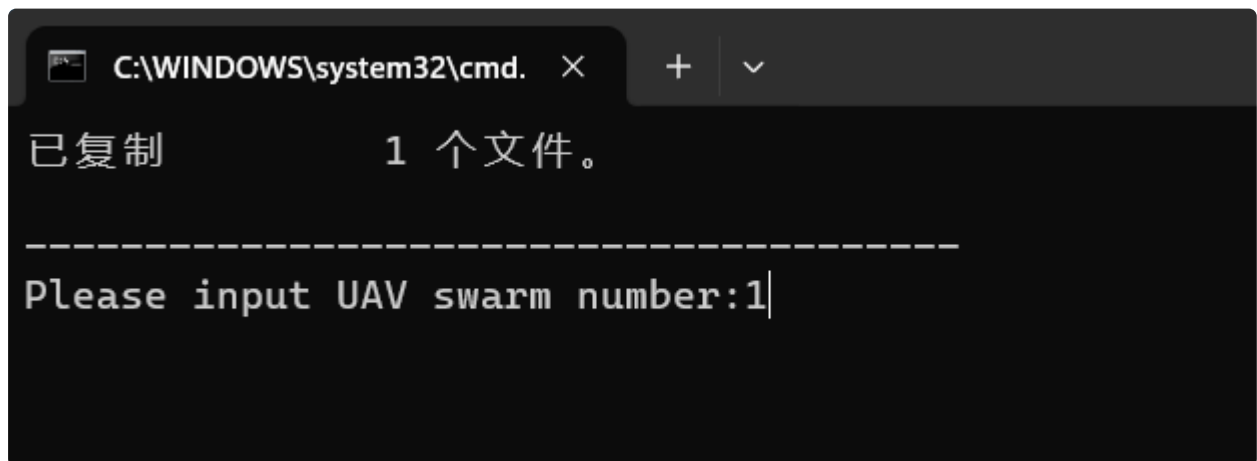
①：推荐配置请见：<https://rflysim.com/>

## 5. 实验步骤

### 5.1. 必做实验：软件在环仿真

#### Step 1：启动仿真

以管理员身份运行 `Helicopter_SITLRun.bat`，输入1后回车启动1架直升机的软件在环仿真。



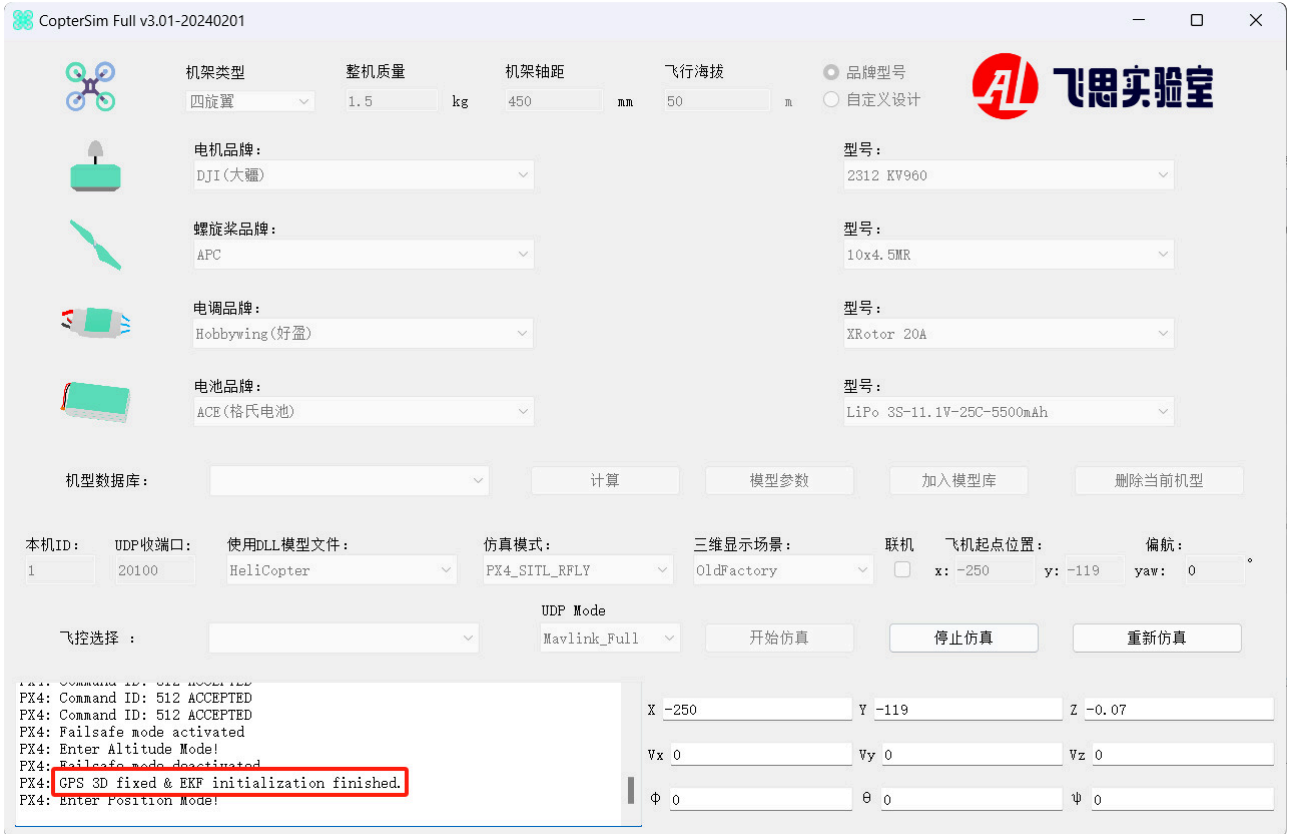
```
C:\WINDOWS\system32\cmd.  X  +  v
已复制      1 个文件。
-----
Please input UAV swarm number:1|
```

注意，在脚本文件中确定机架为直升机对应的机架类型：

```
REM Set the vehicle-model (airframe) of PX4 SITL simulation, the default airframe is a quadcopter: iris
REM Check folder Firmware\ROMFS\px4fmu_common\init.d-posix (or init.d/airframes) for supported airframes
REM E.g., fixed-wing aircraft: PX4SITLFrame=plane; small cars: PX4SITLFrame=rover
set PX4SITLFrame=helicopter
```

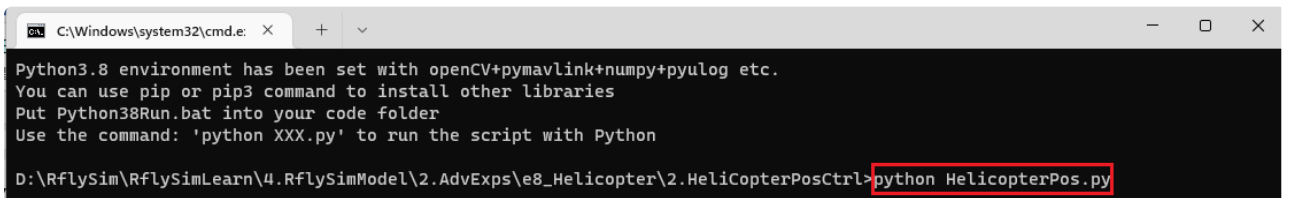
## Step 2: 等待初始化完成

CopterSim右下角显示以下信息时，表明仿真初始化完成。



## Step 3: 运行控制程序

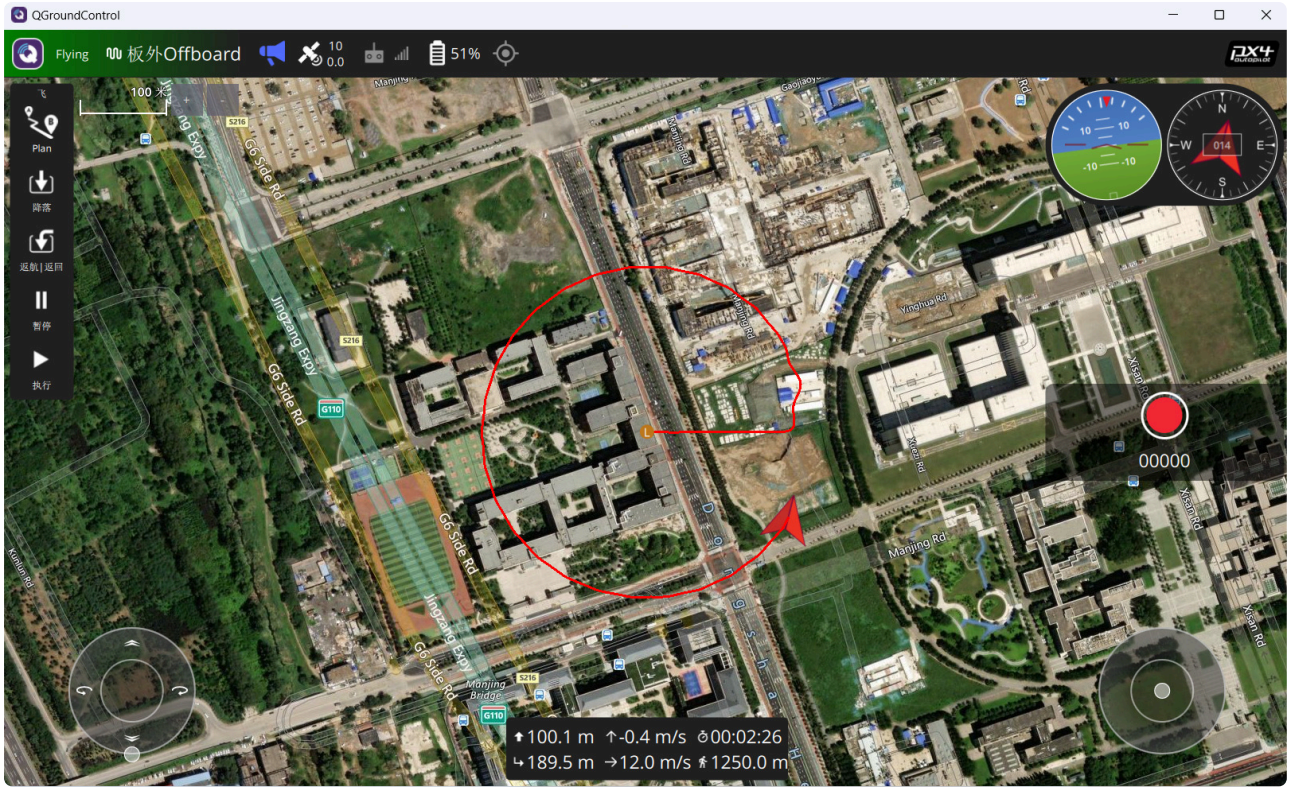
在文件夹下，双击 [Python38Run.bat](#)，打开集成好的python环境，在该环境下运行 [HelicopterPos.py](#) 文件，输入 `python HelicopterPos.py`



## Step 4: 观测结果

在RflySim3D和QGC中观察直升机运动情况。

在QGC中可看到实现了画圆的效果。



## 5.2. 选做实验：硬件在环仿真

### Step 1: 连接飞控

硬件在环仿真需要准备一个飞控，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）。

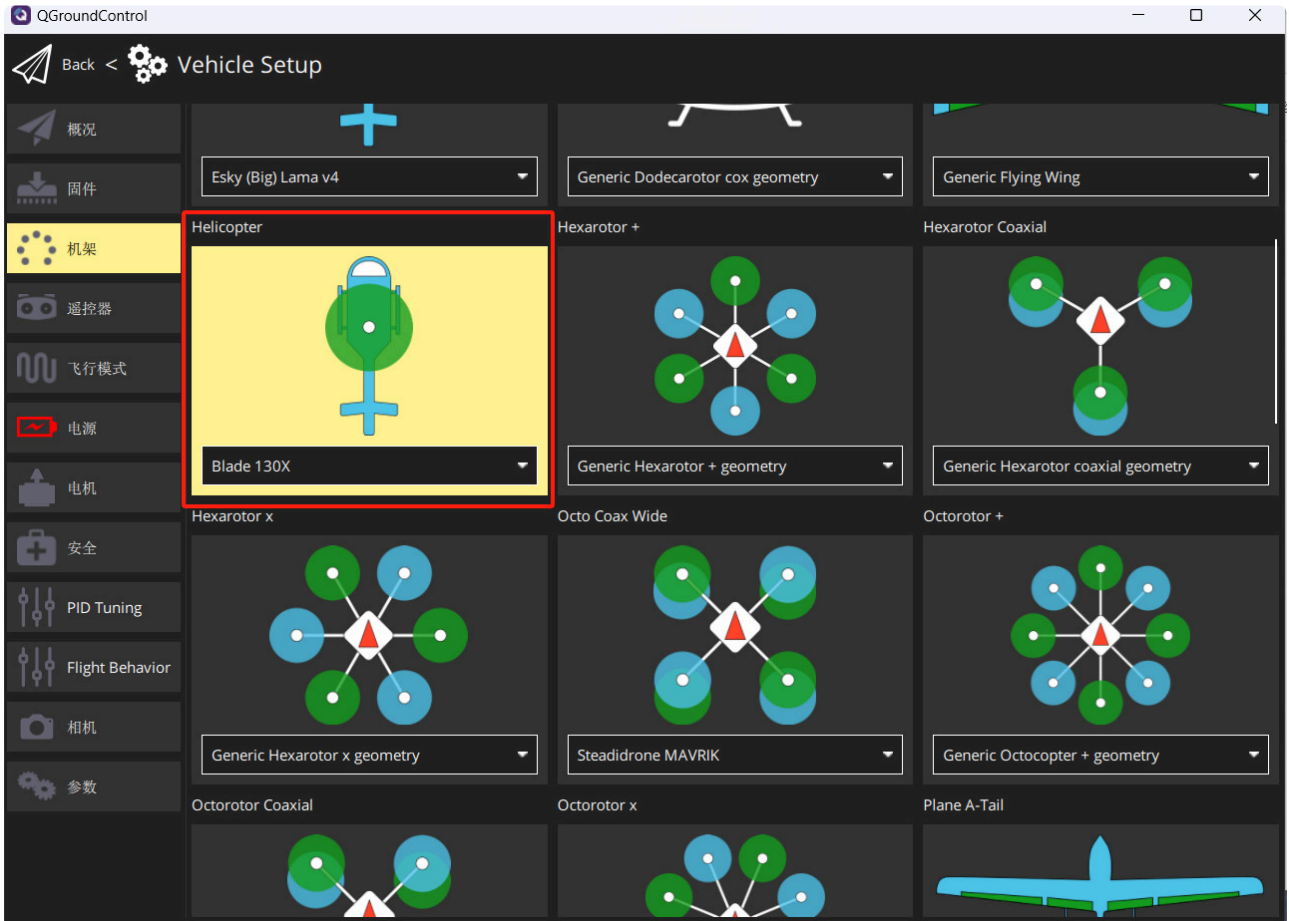


## Step 2: 设置硬件在环机架

在桌面的RflyTools文件夹中打开QGroundControl。

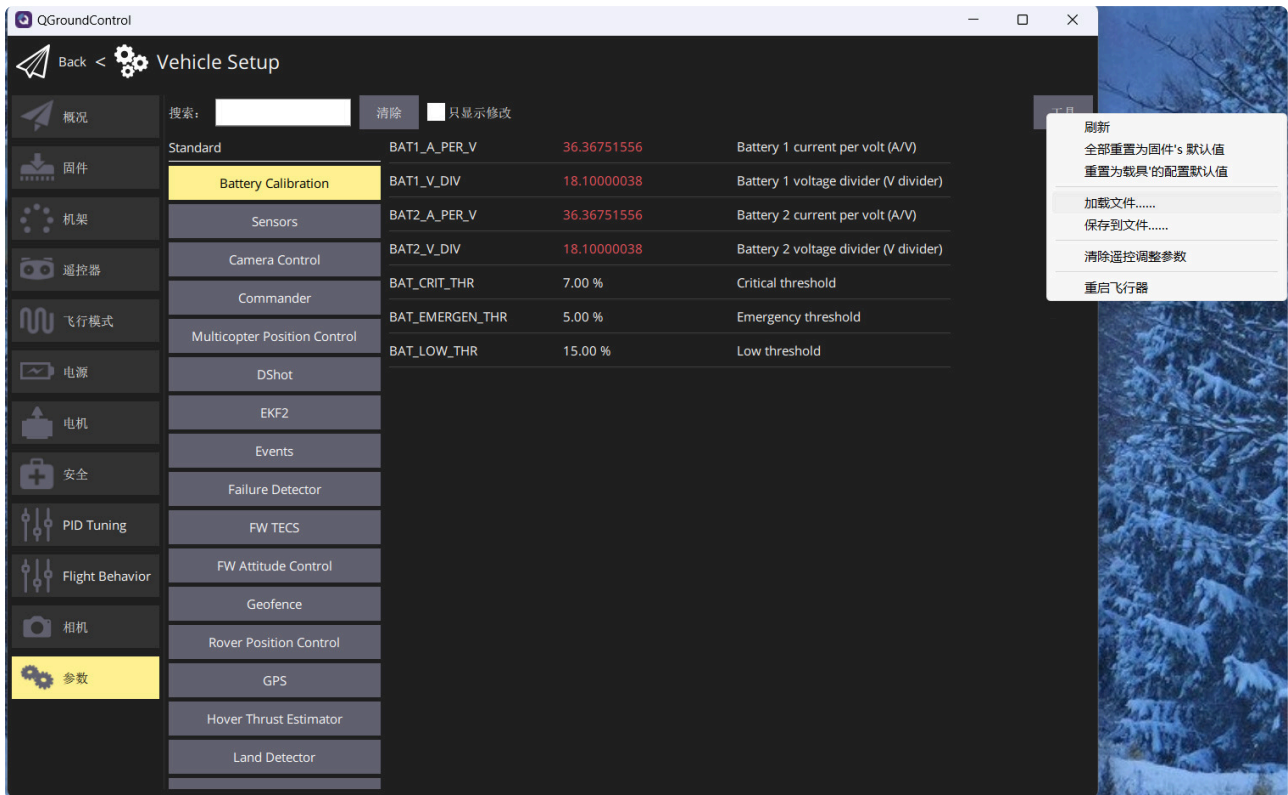
	3DDisplay	2023/11/8 12:56	快捷方式	1 KB
	CopterSim	2023/11/9 16:13	快捷方式	1 KB
	FlightGear-F450	2023/11/8 12:56	快捷方式	2 KB
	HITLRun	2023/11/13 9:43	快捷方式	2 KB
	HowToUse	2023/11/9 16:13	快捷方式	1 KB
	PPTs	2023/9/12 14:05	快捷方式	1 KB
	Python38Env	2023/11/13 9:43	快捷方式	2 KB
	QGroundControl	2023/11/13 9:43	快捷方式	1 KB
	rflysim.com	2022/11/30 13:12	Internet 快捷方式	1 KB
	RflySim3D	2023/11/9 16:13	快捷方式	1 KB
	RflySimAPIs	2023/11/13 9:43	快捷方式	1 KB
	RflySimUE5	2023/11/9 16:13	快捷方式	1 KB
	SITLRun	2023/11/13 9:43	快捷方式	2 KB
	Win10WSL	2023/11/9 18:14	快捷方式	2 KB

点击“机架”，选择“Blade 130X”，点击右上角“应用并重启”。

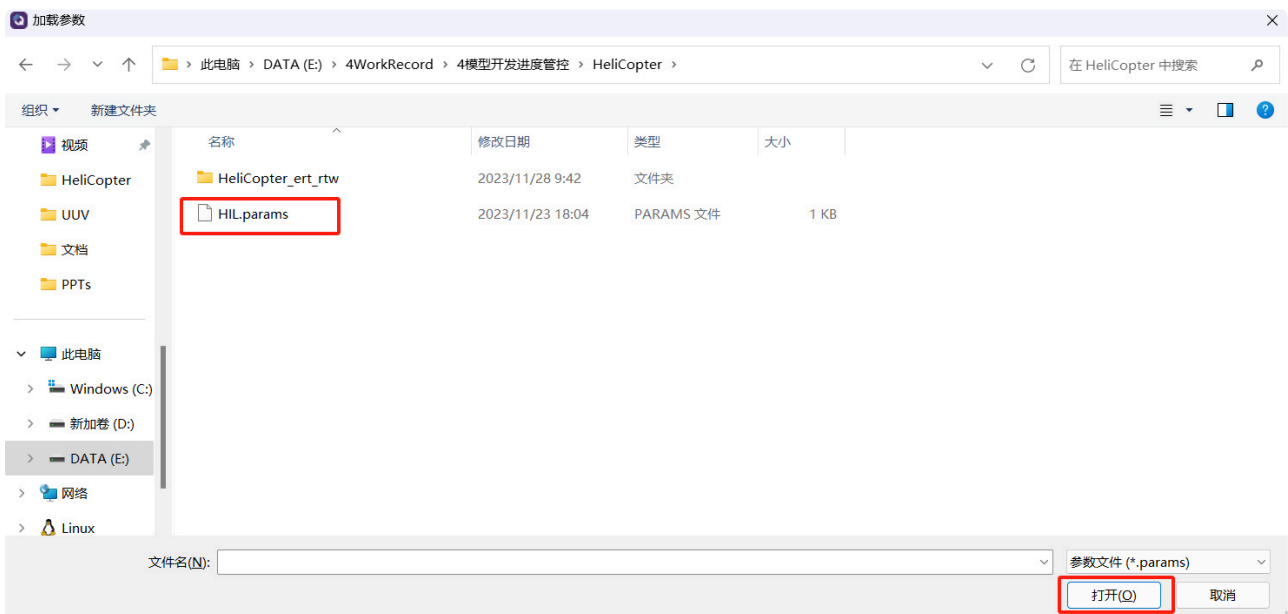


## Step 3: 配置硬件在环参数

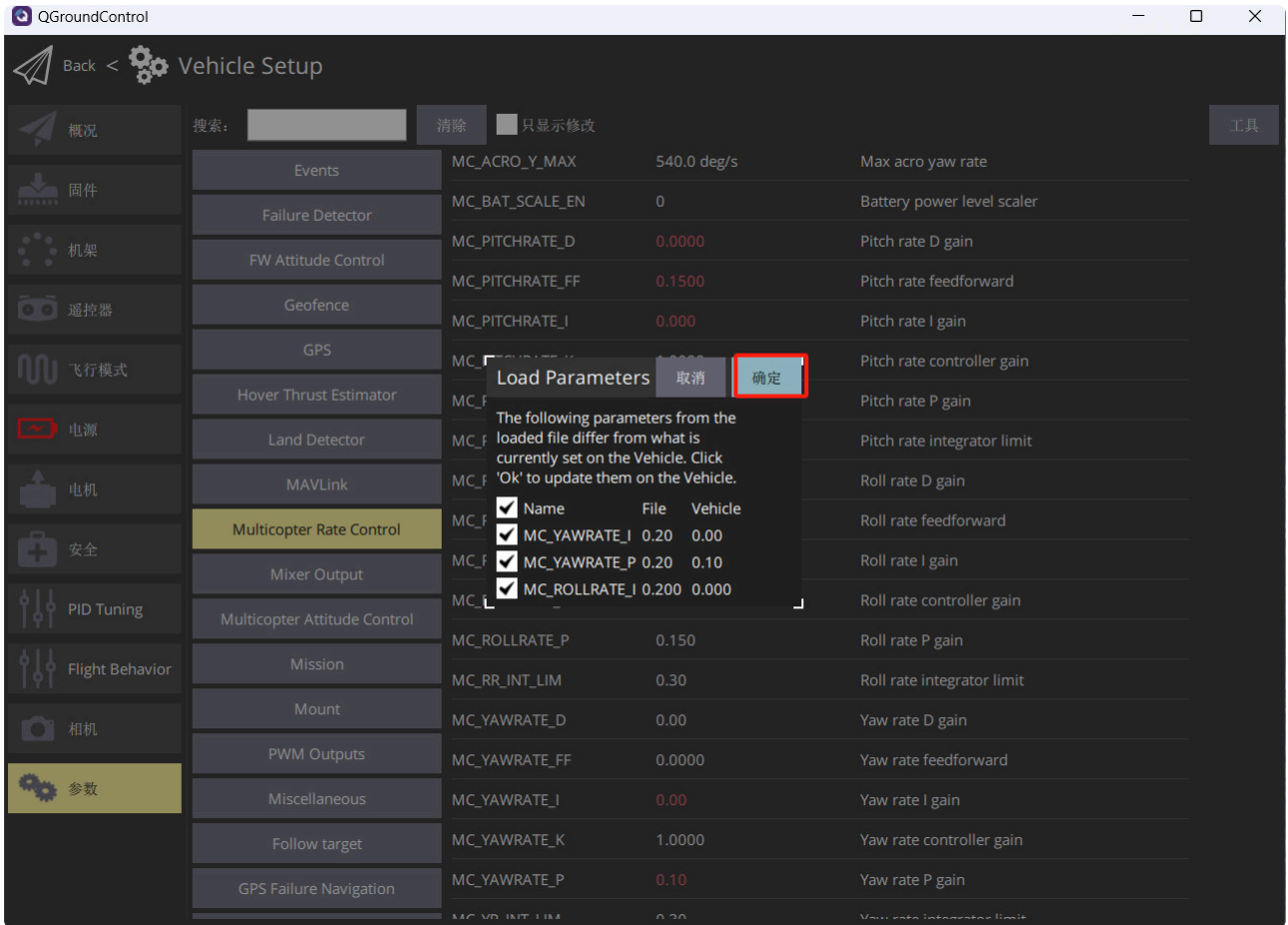
选择“参数”，点击“工具”，选择“加载文件”。



选择“HIL.params”，点击打开。

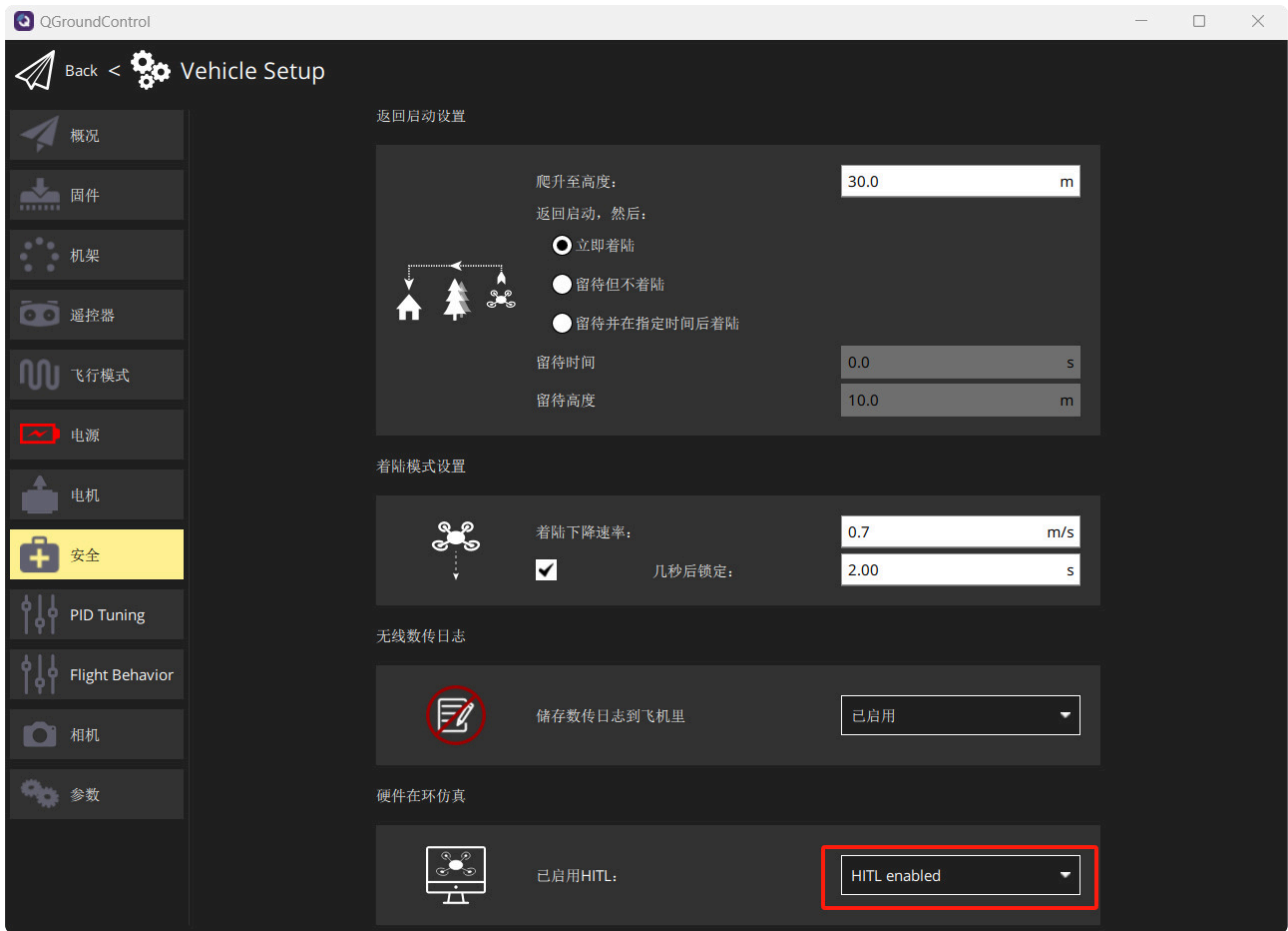


点击“确定”。



## Step 4: 安全界面设置

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



## Step 5: 启动仿真

运行 [Helicopter\\_HITLRun.bat](#)，在弹出窗口中输入飞控端口号，启动飞控硬件在环仿真。

名称	修改日期	类型	大小
dir.xlsx	2024/7/25 13:25	XLSX 工作表	11 KB
HeliCopter.dll	2024/7/25 13:25	应用程序扩展	226 KB
<b>Helicopter_HITLRun.bat</b>	2024/8/10 22:36	Windows 批处理文件	6 KB
Helicopter_SITLRun.bat	2024/7/25 13:25	Windows 批处理文件	6 KB
HelicopterPos.py	2024/7/25 13:25	Python File	4 KB
HIL.params	2024/7/25 13:25	PARAMS 文件	1 KB
Python38Run.bat	2024/8/14 10:32	Windows 批处理文件	1 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 ???
COM6: ??????????
COM7: ??????????

Recommended COM list input is: 3,4,5

-----
My COM list for HITL simulation is:5|
```

## Step 6: 仿真过程

之后测试步骤与软件在环仿真的步骤相同，等待初始化完成后，运行Python程序并观察结果。

## 5.3. 选做实验（VS Code调试运行）

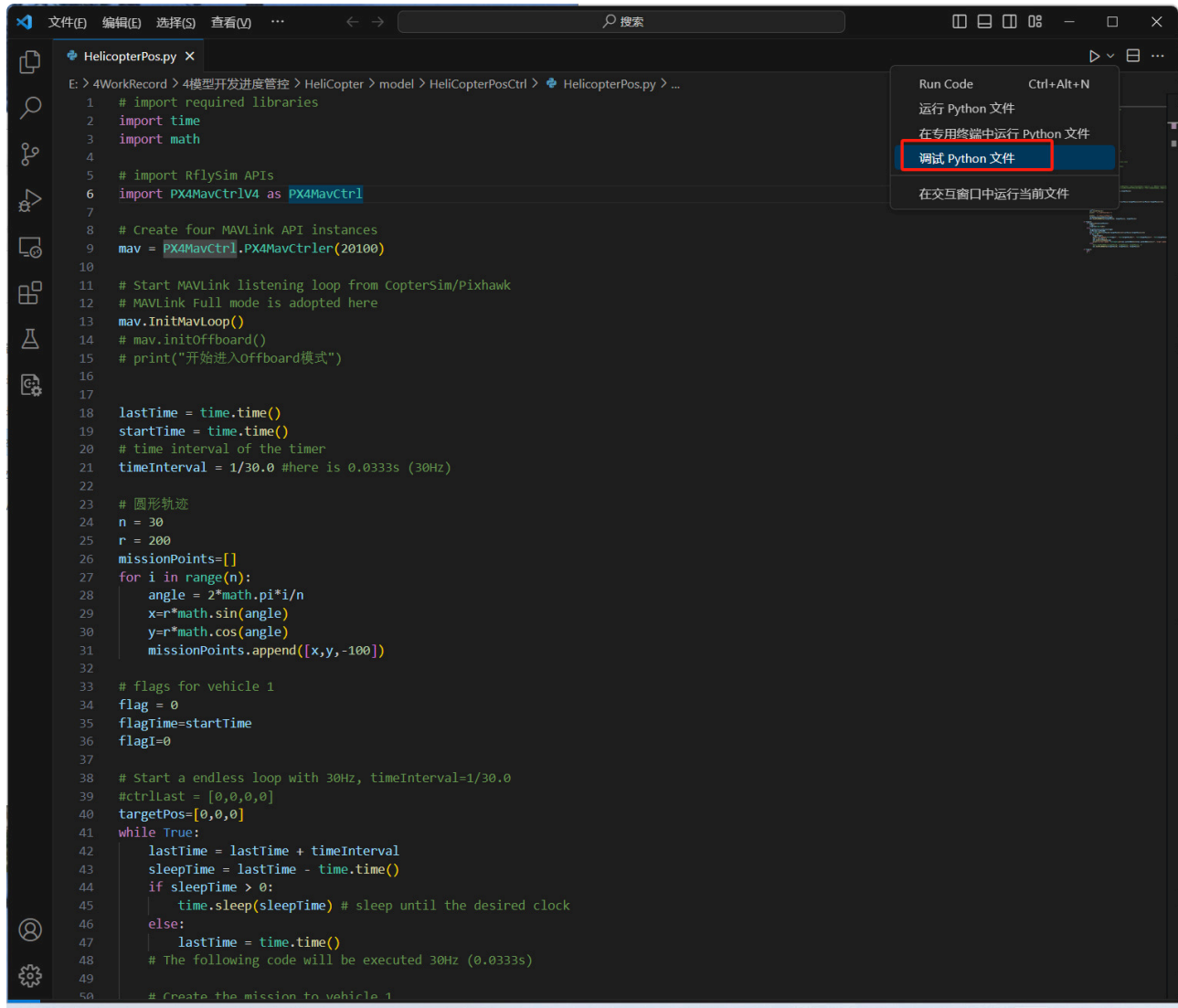
### 准备工作

- 先确保已经按 [RflySimAPIs\1.RflySimIntro\2.AdvExps\e3.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 [HelicopterPos.py](#) 时，可使用VS Code（或Pycharm等工具）来打开 [HelicopterPos.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 [HelicopterPos.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。



```
1 # import required libraries
2 import time
3 import math
4
5 # import RflySim APIs
6 import PX4MavCtrlV4 as PX4MavCtrl
7
8 # Create four MAVLink API instances
9 mav = PX4MavCtrl.PX4MavCtrl(20100)
10
11 # Start MAVLink listening loop from CopterSim/Pixhawk
12 # MAVLink Full mode is adopted here
13 mav.InitMavLoop()
14 # mav.initOffboard()
15 # print("开始进入Offboard模式")
16
17
18 lastTime = time.time()
19 startTime = time.time()
20 # time interval of the timer
21 timeInterval = 1/30.0 #here is 0.0333s (30Hz)
22
23 # 圆形轨迹
24 n = 30
25 r = 200
26 missionPoints=[]
27 for i in range(n):
28     angle = 2*math.pi*i/n
29     x=r*math.sin(angle)
30     y=r*math.cos(angle)
31     missionPoints.append([x,y,-100])
32
33 # flags for vehicle 1
34 flag = 0
35 flagTime=startTime
36 flagI=0
37
38 # Start a endless loop with 30Hz, timeInterval=1/30.0
39 ctrlLast = [0,0,0,0]
40 targetPos=[0,0,0]
41 while True:
42     lastTime = lastTime + timeInterval
43     sleepTime = lastTime - time.time()
44     if sleepTime > 0:
45         time.sleep(sleepTime) # sleep until the desired clock
46     else:
47         lastTime = time.time()
48     # The following code will be executed 30Hz (0.0333s)
49
50 # create the mission to vehicle 1
```

## 6.参考资料

1. PX4PSP\RflySimAPIs\4.RflySimModel\API.pdf中DLL/SO模型与通信接口的重要参数部分。
2. [\[RflySim安装目录\]/RflySimAPIs/4.RflySimModel/API.pdf](#)
3. [\[RflySim安装目录\]/RflySimAPIs/4.RflySimModel/API.pdf](#)
- 4.

# 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