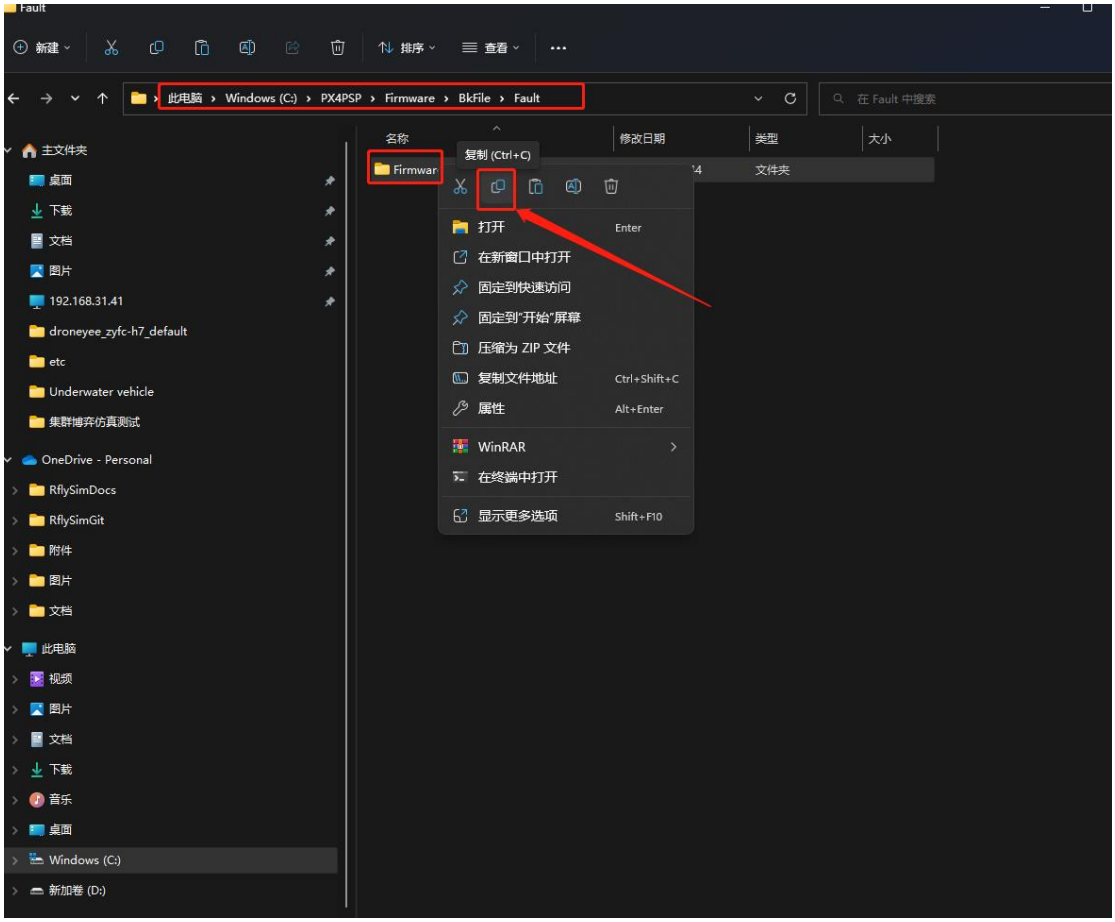
一、 故障注入固件编译及实验过程(重点)

真机自动生成代码

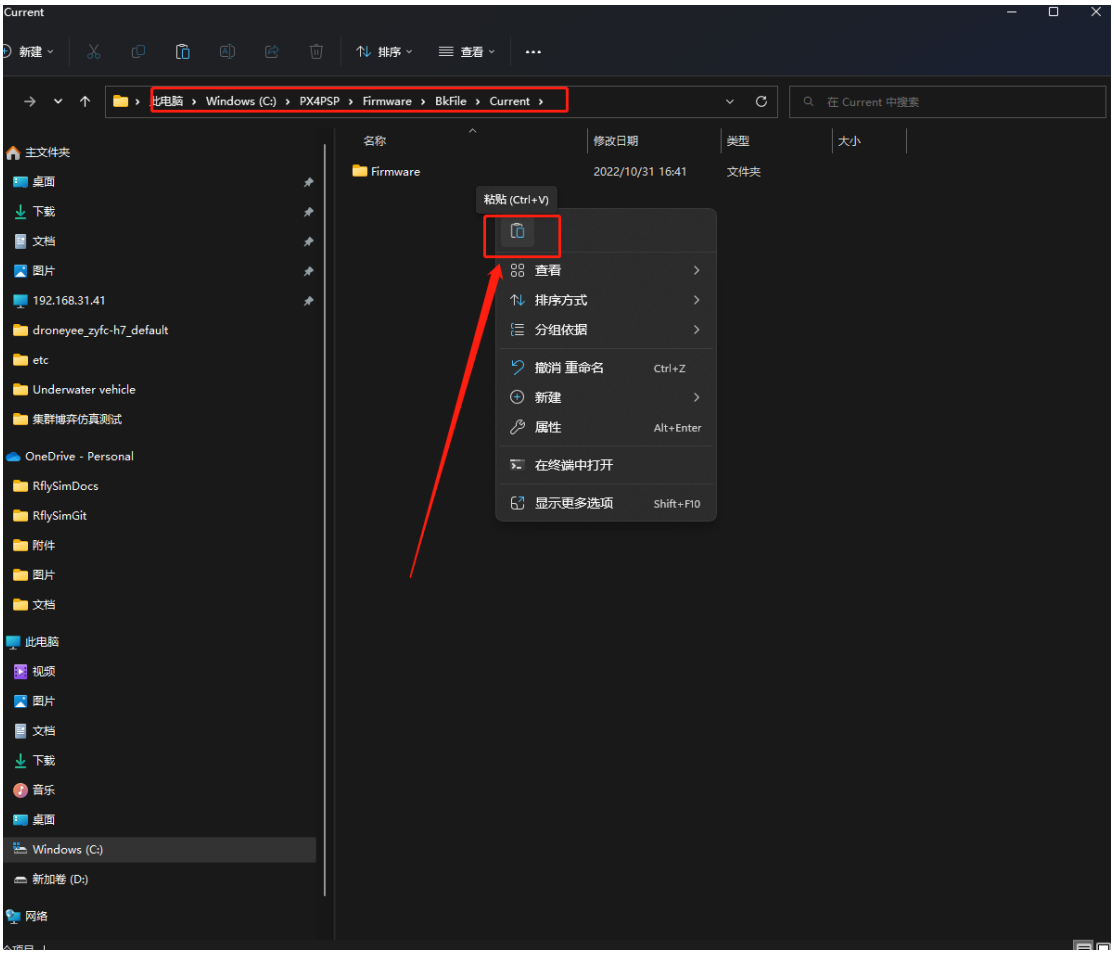
替换故障文件:

1. 将路径 C:\PX4PSP\Firmware\BkFile\Fault 下的“Firmware”文件整体复制到此路径下 C:\PX4PSP\Firmware\BkFile\Current，进行覆盖替换。

第一步

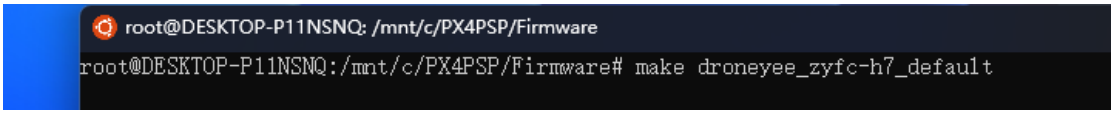


第二步

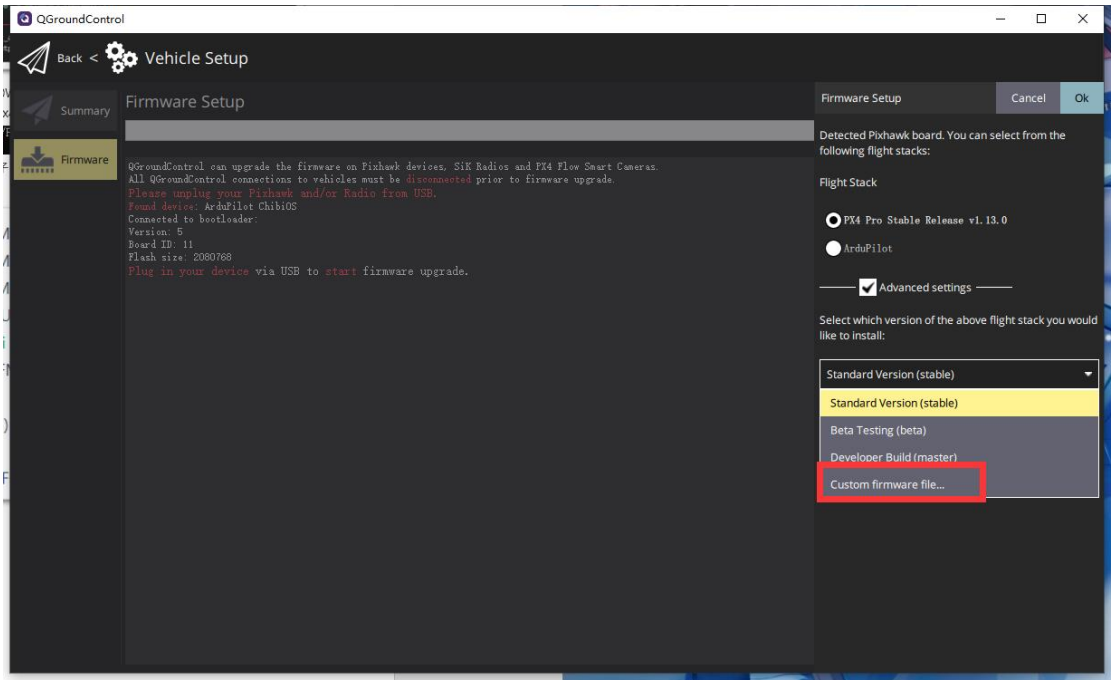


编译：（硬件在环流程）

执行完./EnvFault.sh 脚本替换成故障源码后，根据（指定飞控（如 H7））编译命令输入如下命令

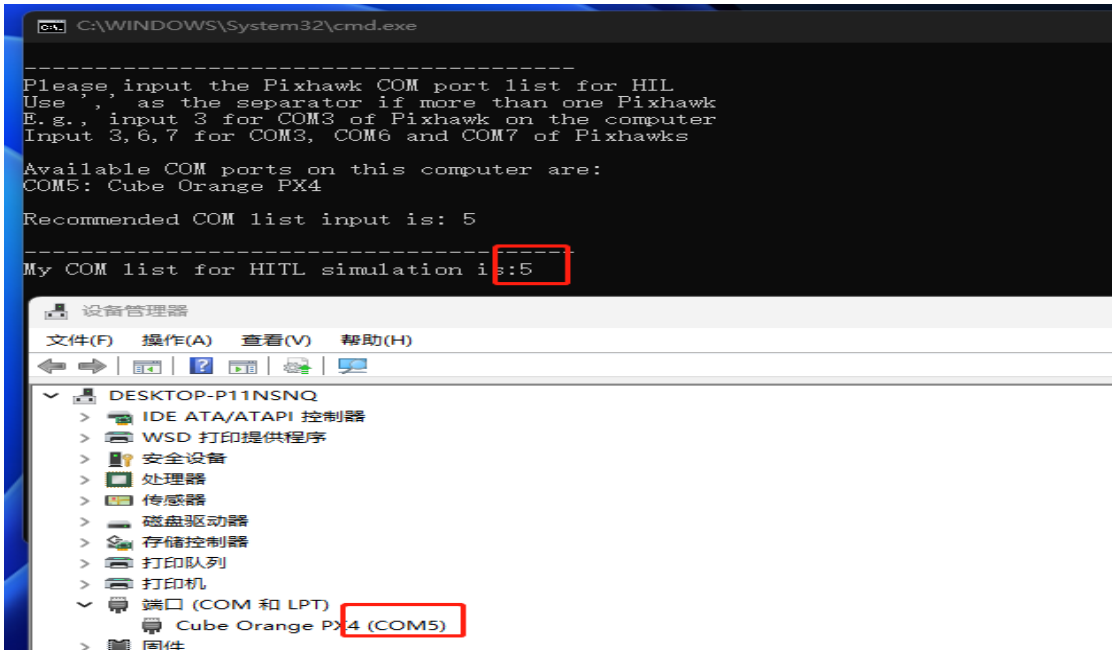


2.编译完成后打开 QGC 进程烧录

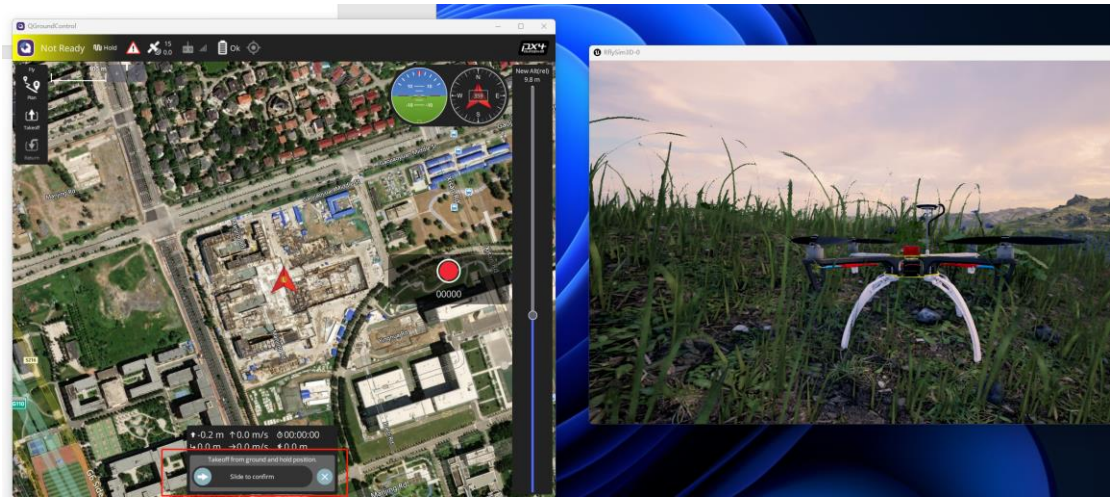


4.用管理员权限打开 PythonSender 文件中的 bat 文件

输入连接飞控板的端口数字



5.在 QGC 中点击起飞飞机



6. 打开 PythonSender 文件中的 PythonSender.py 文件

(G) 运行(R) 终端(T) 帮助(H) PythonSender.py - PythonSender - Visual Studio Code

PythonSender.py × PX4MavCtrlV4.py

PythonSender > PythonSender.py > ...

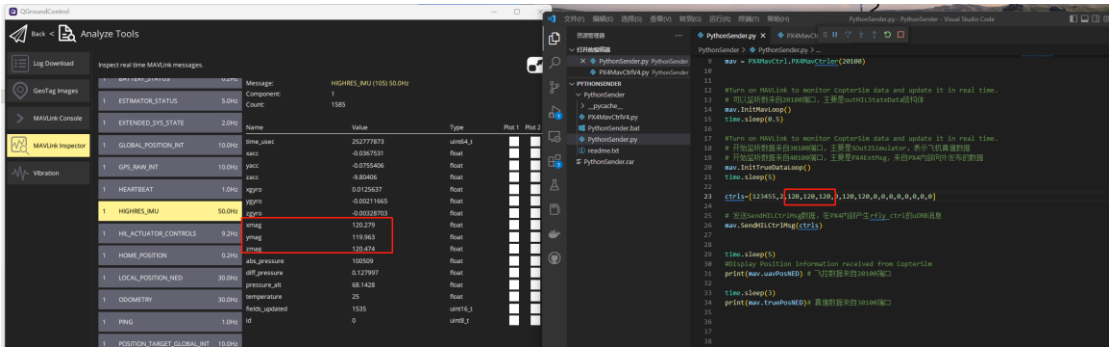
```
9   mav = PX4MavCtrl.PX4MavCtrler(20100)
10
11
12  #Turn on MAVLink to monitor CopterSim data and update it in real time.
13  # 可以监听数来自20100端口，主要是outHILStateData结构体
14  mav.InitMavLoop()
15  time.sleep(0.5)
16
17  #Turn on MAVLink to monitor CopterSim data and update it in real time.
18  # 开始监听数据来自30100端口，主要是SOut2Simulator，表示飞机真值数据
19  # 开始监听数据来自40100端口，主要是PX4ExtMsg，来自PX4内部向外发布的数据
20  mav.InitTrueDataLoop()
21  time.sleep(5)
22
23  ctrls=[123450,2,120,120,120,0,120,120,0,0,0,0,0,0,0,0]
24
25  # 发送SendHILCtrlMsg数据，在PX4内部产生rfly_ctrl的uORB消息
26  mav.SendHILCtrlMsg(ctrls)
27
28
29  time.sleep(5)
30  #Display Position information received from CopterSim
31  print(mav.uavPosNED) # 飞控数据来自20100端口
32
33  time.sleep(3)
34  print(mav.truePosNED)# 真值数据来自30100端口
35
```

进行故障注入的参数（重点）:

红色框为我们在 msg 文件中设置的 16 位控制数，其中蓝色划线为 ID 端口

类型	ID 号	备注
地磁	123455	第 0 位是 ID，第一位是选择位（若为 1，则是赋值方式，若为 2，则是叠加方式），2，3，4 位是参数
GPS	123456	第 0 位是 ID，第一位是选择位，2，3，4 位是参数
遥控器	123457	第 0 位是 ID，第一位是选择位，后面是参数，根据遥控输出个数定
电机	123450	第 0 位是 ID，第一位是选择位，后面是参数，参数个数根据电机输出个数定
加速度计	123542	第 0 位是 ID，第一位是选择位，后面是参数，参数设定是 X Y Z

7.地磁故障注入



8.GPS 故障