1.实验名称及目的

1.1 实验名称

故障注入与飞控日志时间戳获取实验

1.2 实验目的

掌握通过飞控日志（`vehicle_command_0.csv`）获取故障注入时间戳的方法；

1.3 关键知识点

故障注入机制

- 脚本通过调用 RflySimSDK: PX4MavCtrler类 封装的接口 sendSILIntFloat与 注入不同类型故障，通过SendFaultEvent将这些故障记录到飞控日志中 。
- 完整的故障注入机制参见 RflySimSDK: 一、故障标识符（FaultID）与故障参数

日志与时间戳解析

- PX4/QGC 自动记录各种类型的uORB消息；
- 当 `SendFaultEvent` 命令发送 `vehicle_command` 类型消息（command=31000，该消息不触发飞控动作），日志中会保存对应时间戳；
- 通过 RflySimSDK: QGCCtrlAPI类 中的 `ReqQgcLog()` 调用QGC下载 `.ulg`，再调用 `ulog2csv` 转换，得到 `vehicle_command_0.csv`；
- 在 CSV 中搜索 `command=31000` 结合故障id等即可定位特定故障的注入时刻。

脚本逻辑流程

- 初始化 MAVLink 循环与真值数据接收；
- 起飞至指定高度；
- 在设定时刻注入故障（电机效率、风扰动等）；
- 继续飞行，输出状态，直至实验结束；
- 自动请求 QGC 下载日志，并解析为 CSV 文件。

2.实验效果

完成实验后，用户应能够：

- 在仿真中观察无人机因电机故障坠毁或姿态异常；成功导出 QGC/飞控日志文件；
- 从 `vehicle_command_0.csv` 中解析出**故障注入的时间戳**；

3.文件目录

例程目录：[安装目录]\RflySimAPIs\7.RflySimPHM\0.ApiExps\e18_FaultLogTimestamp

文件夹/文件名称	说明
MulticopterModel.dll	包含电机故障模块的dll模型
MulticopterModelHITL.bat	硬件在环仿真批处理文件。
MulticopterModelSITL.bat	软件在环仿真批处理文件。
Python38Run.bat	Python环境启动脚本。
FaultInjectAPITest.py	实验主程序。

[!NOTE]
运行环境要求

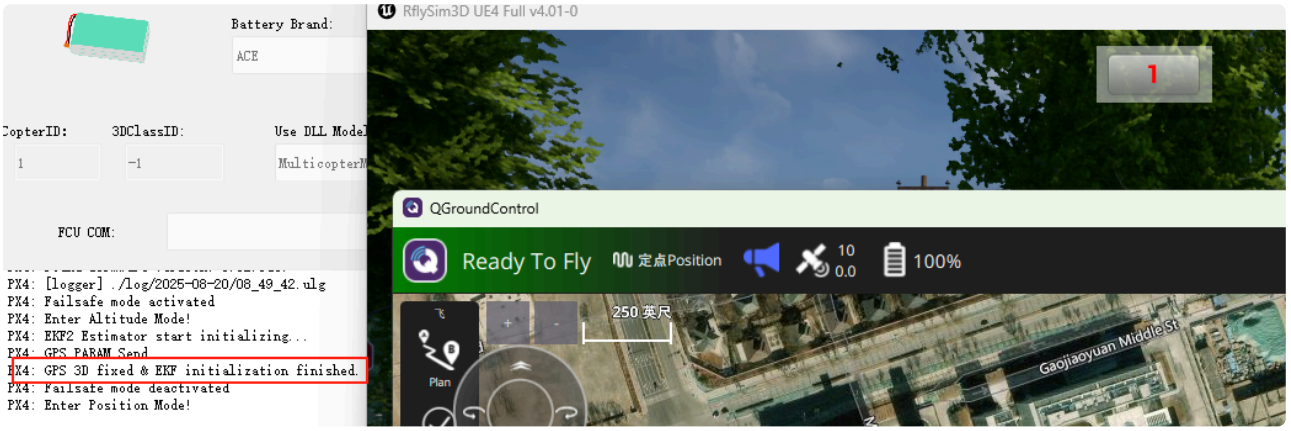
- **软件**：Windows 10+，RflySim 工具链
- **硬件**：普通笔记本/台式机
- **平台版本**：RflySim工具链4.01及以上

5.实验步骤

4.1 必做实验

Step 1：运行软件在环仿真

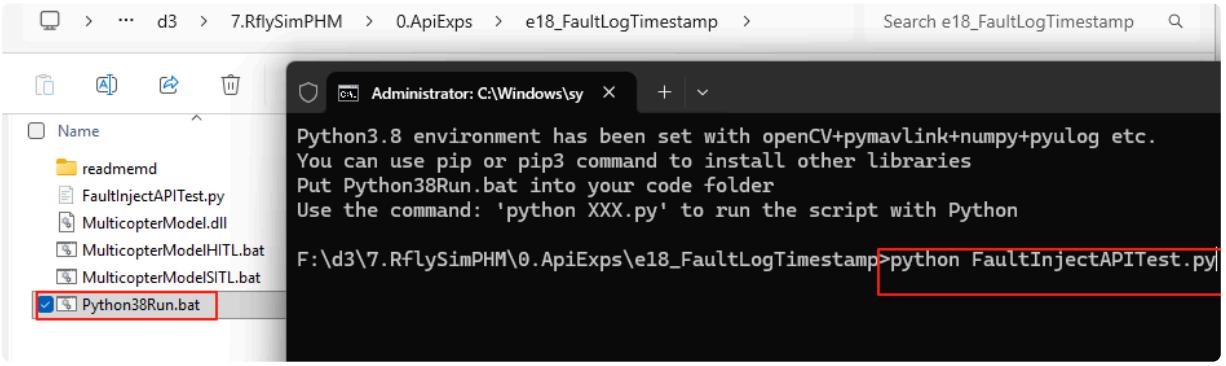
双击运行 `MulticopterModelSITL.bat`，启动无人机软件在环仿真，等待初始化完成。



Step 2：注入故障并获取日志

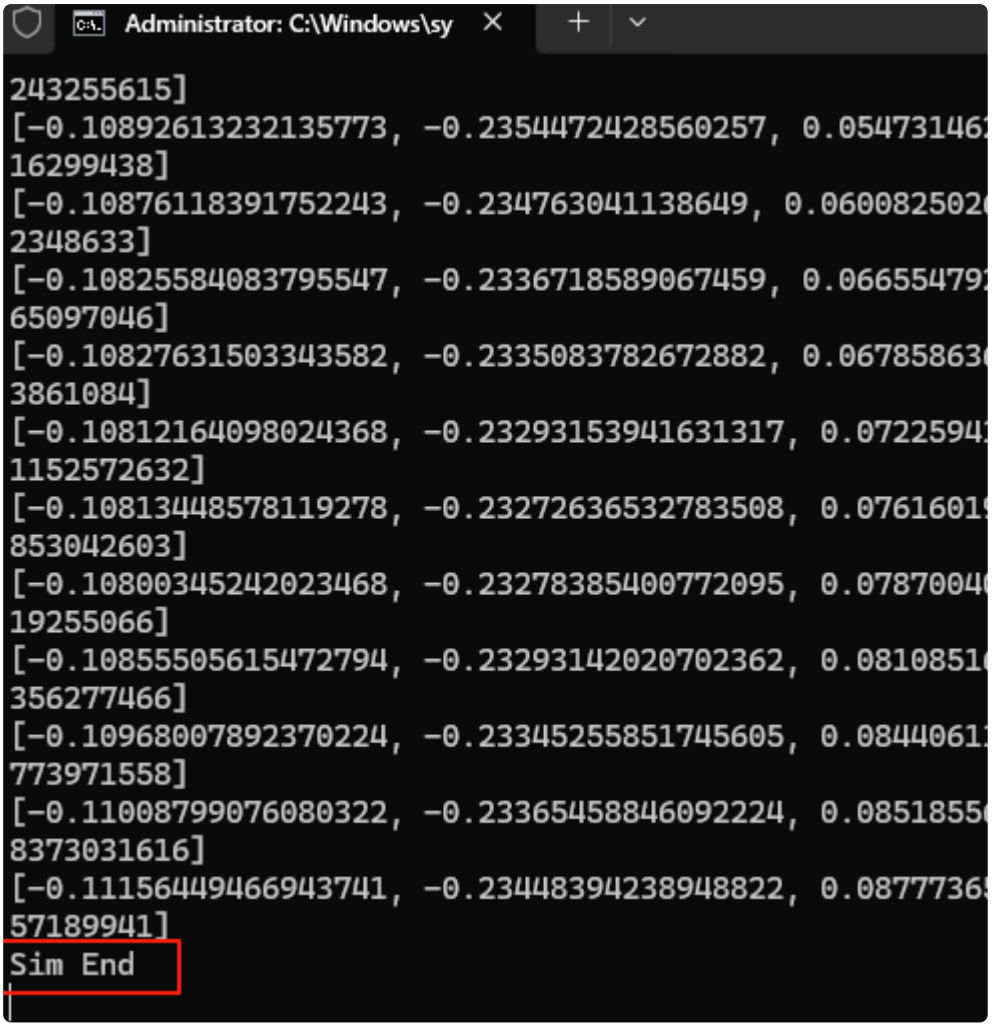
- 双击 `Python38Run.bat` 进入 Python 环境；
- 运行：

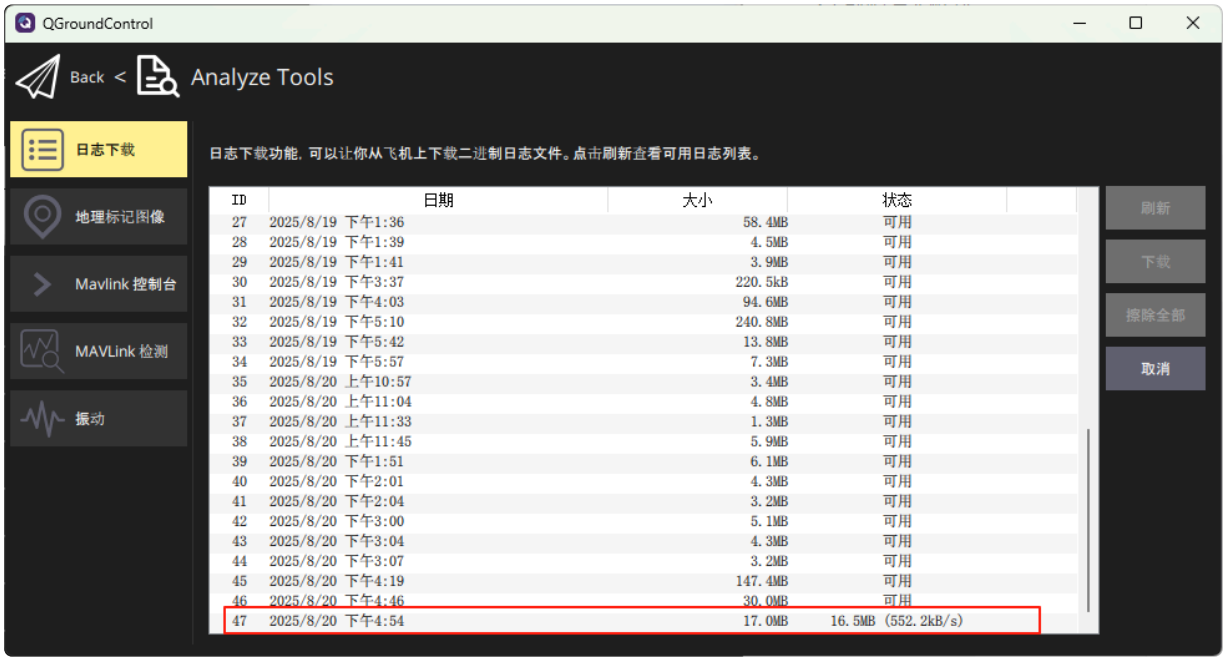
```
python FaultInjectAPITest.py
```



- 本例中无人机会被注入电机故障，飞行中断或坠毁。

- 等待仿真结束后，Python运行的终端会打印`Sim end`，此时会自动进入QGC日志界面开始下载日志

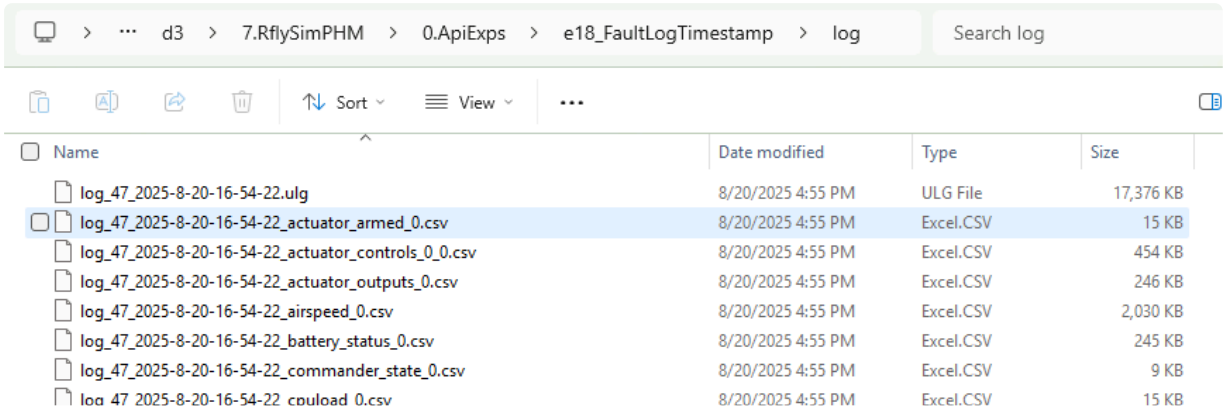




- 下载完成后会打印消息并将ulg文件复制到本实验目录并解析为csv

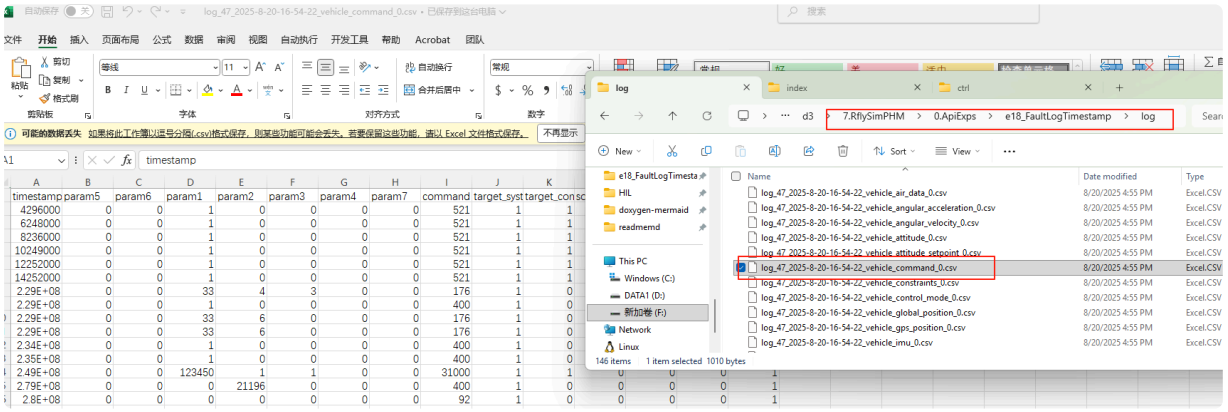
```
[-0.11156449466943741, -0.23448394238948822, 0.08777365088461
57189941]
Sim End
Got log file: log_47_2025-8-20-16-54-22.ulg
Download log log_47_2025-8-20-16-54-22.ulg successfully.
ulog2csv parse finished.

F:\d3\7.RflySimPHM\0.ApiExps\e18_FaultLogTimestamp>
```

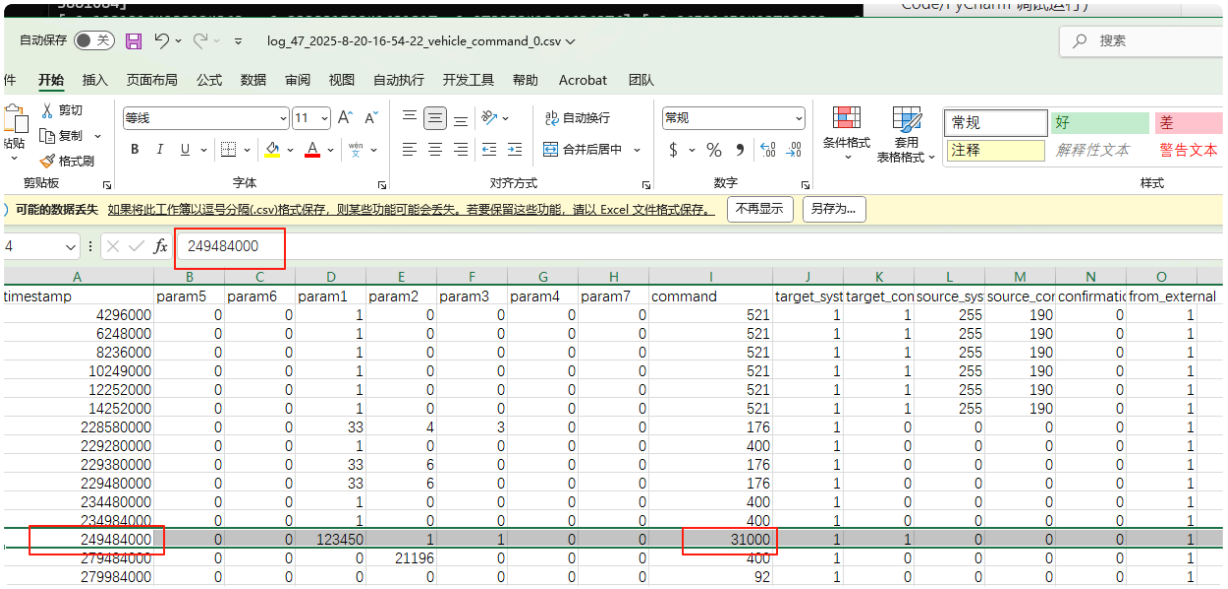


Step 3：获取日志时间戳

- 打开 后缀 `vehicle_command_0.csv` 的文件；



- 查找 `command=31000` 的行；
- 读取对应时间戳字段，作为**故障发生时间**。



timestamp	param5	param6	param1	param2	param3	param4	param7	command	target_syst	target_con	source_sys	source_cor	confirmatic	from_external
4296000	0	0	1	0	0	0	0	521	1	1	255	190	0	1
6248000	0	0	1	0	0	0	0	521	1	1	255	190	0	1
8236000	0	0	1	0	0	0	0	521	1	1	255	190	0	1
10249000	0	0	1	0	0	0	0	521	1	1	255	190	0	1
12252000	0	0	1	0	0	0	0	521	1	1	255	190	0	1
14252000	0	0	1	0	0	0	0	521	1	1	255	190	0	1
228580000	0	0	33	4	3	0	0	176	1	0	0	0	0	1
229280000	0	0	1	0	0	0	0	400	1	0	0	0	0	1
229380000	0	0	33	6	0	0	0	176	1	0	0	0	0	1
229480000	0	0	33	6	0	0	0	176	1	0	0	0	0	1
234480000	0	0	1	0	0	0	0	400	1	0	0	0	0	1
234984000	0	0	1	0	0	0	0	400	1	0	0	0	0	1
249484000	0	0	123450	1	1	0	0	31000	1	1	0	0	0	1
279484000	0	0	0	21196	0	0	0	400	1	0	0	0	0	1
279984000	0	0	0	0	0	0	0	92	1	0	0	0	0	1

4.2 选做实验（VS Code/PyCharm 调试运行）

- 配置 Python IDE（VS Code / PyCharm）；
- 打开 `FaultInjectAPITest.py` 源码；
- 在断点处逐行执行，观察 `SendMavCmdLong` 的参数传递过程；
- 对照 `PX4MavCtrlV4.py` 源码理解 MAVLink 封装逻辑。

扩展示例：可使用 `testPX4Mav.py` 中的 `TestSynModel`，结合 `SendSynAcc()` 指令实现加速度扰动后再注入故障，观察差异。

6.参考资料

- RflySim SDK 故障注入文档
- PX4 官方文档： MAVLink Command Protocol

6. 常见问题（FAQ）