基于smolagents智能体框架的无人机任务执行

1. 实验目的

利用代码生成智能体（如 smolagent）结合 LLM 的程序合成能力，探索自然语言如何驱动无人机完成复杂、多步骤任务，实现"语言即任务代码"的目标。用户使用一句或几句话的自然语言描述任务，系统借助 smolagents 中的 CodeAgent（代码生成智能体），由大语言模型（LLM）生成完整的 Python 控制逻辑，驱动无人机完成任务。

2. 实验要求

- 软件要求：Windows 10及以上版本；RflySim工具链[1]。
  > 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

- 硬件要求：笔记本/台式电脑1台[2]。
  > ①：推荐配置请见：https://rflysim.com/

3.实验地址

例程目录：[安装目录]\RflySimAPIs\6.RflySimExtCtrl\3.CustExps\e2.AI_AgentUAVCtrl

./SITL/ServerFile/main.py：主程序，执行由 LLM 生成的任务逻辑，包括初始化、状态监控、异常处理
./SITL/ServerFile/OpenAI_api_Mavlink_Agent.py：LLM 交互模块，以 CodeAgent 方式向大语言模型发起提示并接收 Python 代码成果
./SITL/ServerFile/volcEngineLLM.py：使用火山引擎的 LLM 接口调用模块（与 OpenAI 平行，具备用例覆盖）
./SITL/ServerFile/Description.py：提示词配置，用于构造对话或代码生成任务的自然语言模板，定义任务步骤、意图

./SITL/ServerFile/Communication_Mavlink.py：封装 MAVLink 协议通信逻辑，用于路径、起飞、降落等命令的发送与异常反馈接收
./SITL/ServerFile/Coordinate_Transformation.py：处理无人机在全局经纬度与本地坐标系之间的转换，供路径规划模块使用

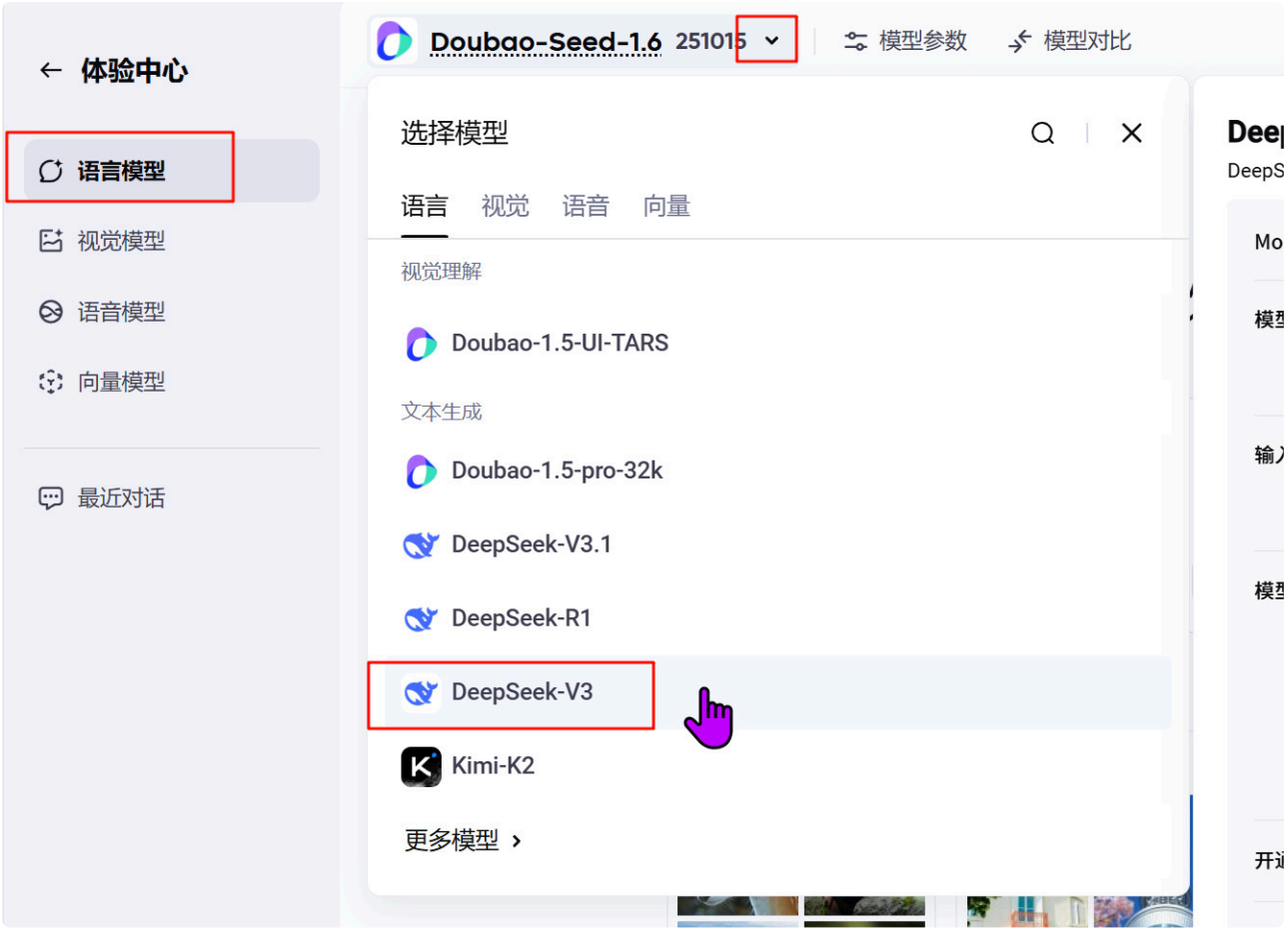
4. 实验内容或步骤

4.1 步骤1：获取大语言模型接口API模型密钥

①进入火山模型官网 https://www.volcengine.com，点击上方工具栏的"大模型"中的"豆包大模型"，需要登录时请自动登录账号。

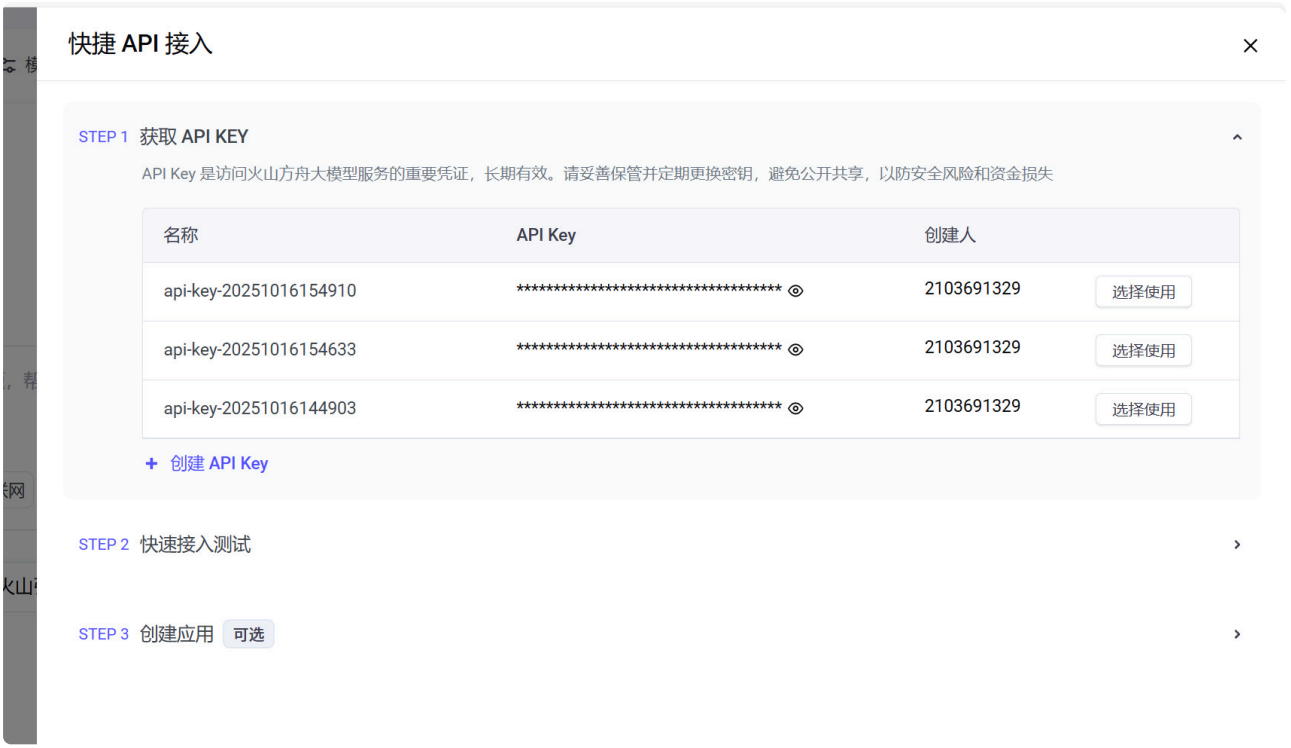
②点击左侧"体验中心"。

在左侧选择"语言模型"，上方打开下拉框将模型更改为DeepSeek-V3。

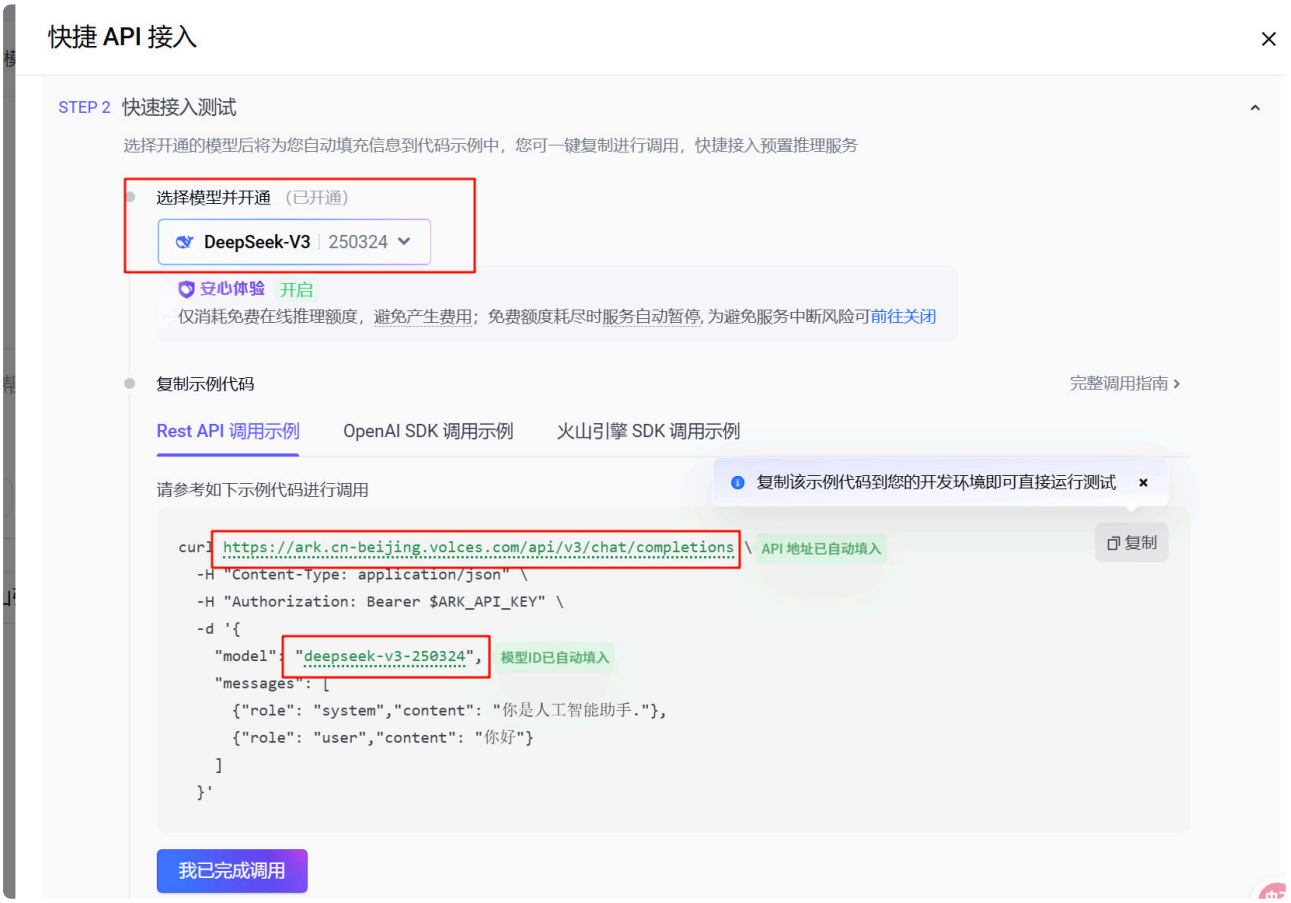


点击右侧"API"接入。

③在弹出的对话框中，按照步骤依次获取APIKEY、接入测试。其中，若无下图显示显示的相关API Key，请自行点击创建。并将此处的API KEY需要记录下来后续步骤中需要使用。



接入测试中，需要切换到DeepSeek-V3模型并开通，示例代码选择为"Rest API调用示例"，此处需要记录下下图中的curl和model值，后续实验步骤中需要使用。



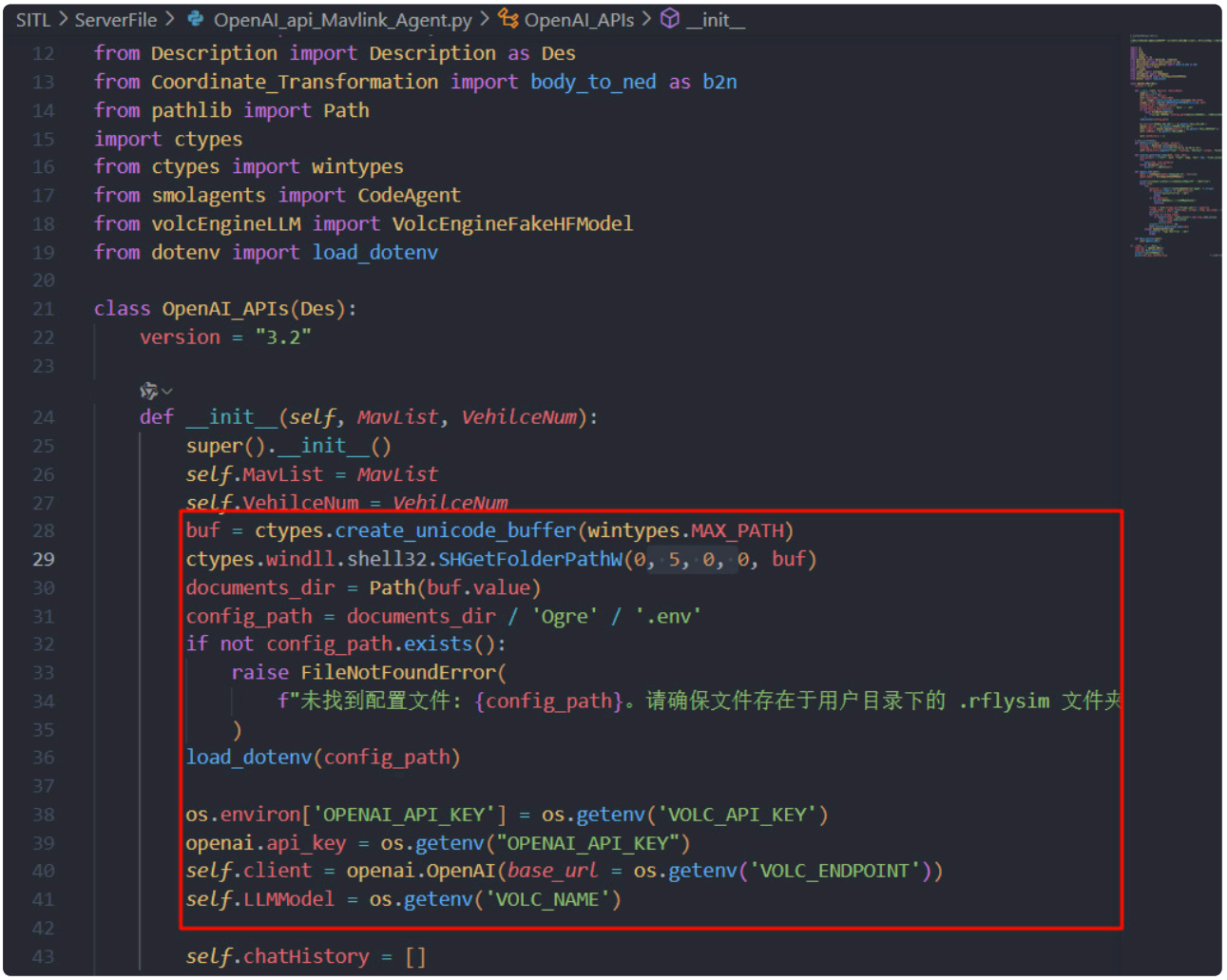
4.2 步骤2：API接口调用配置

本实验中需要将密钥按照格式写入"*\文档\Ogre.env"，该文件配置和详细说明可见实验[RflySim安装目录]\RflySimAPIs\6.RflySimExtCtrl\0.ApiExps\e20.LLMUsage\1.CloudAPIInvocation。打开`\文档\Ogre\.env`，依次按照如下创建变量并输入上一步中记录的相关值：

```
VOLC_API_KEY=45f6ff5*******  # 上一步中记录的API KEY
VOLC_NAME=deepseek-v3-250324 # 上一步中记录的model值
VOLC_PROVIDER=volcengine     # 暂时没使用到
VOLC_AGENT_ENDPOINT=https://ark.cn-beijing.volces.com/api/v3/chat/completions # 上一步中记录的curl值
```

创建完成后，保存本文件后关闭。

本实验中SITL\ServerFile\volcEngineLLM.py代码会自动调用"*\文档\Ogre.env"文件中的相关变量。需要注意的是本实验的`volcEngineLLM.py`使用的是Rest API 调用方式，需要特别注意上面的curl值

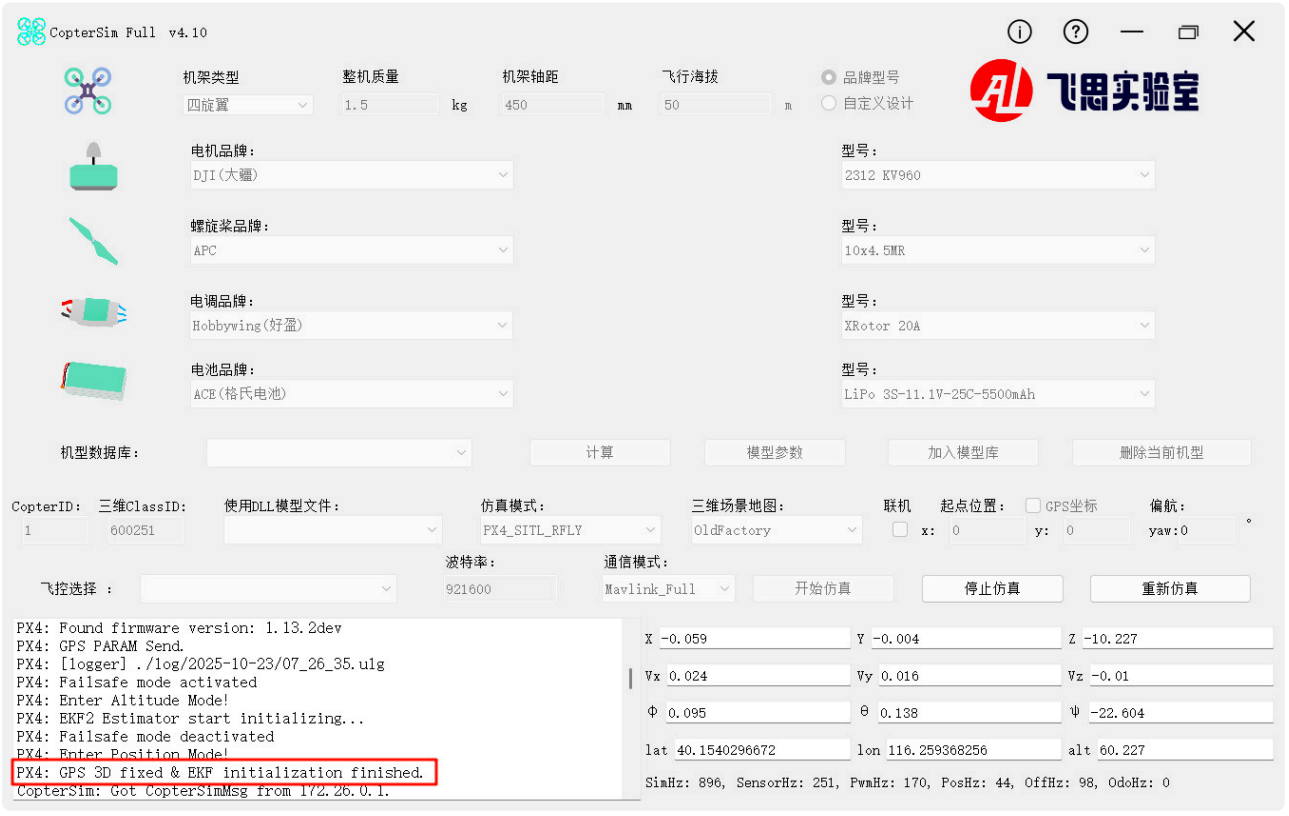


SITL > ServerFile > volcEngineLLM.py > ...

```python
import requests
import re,os
from pathlib import Path
import ctypes
from ctypes import wintypes
from dotenv import load_dotenv

class VolcEngineFakeHFModel:
    def __init__(self):
        buf = ctypes.create_unicode_buffer(wintypes.MAX_PATH)
        ctypes.windll.shell32.SHGetFolderPathW(0, 5, 0, 0, buf)
        documents_dir = Path(buf.value)
        config_path = documents_dir / 'Ogre' / '.env'
        if not config_path.exists():
            raise FileNotFoundError(
                f"未找到配置文件: {config_path}。请确保文件存在于用户目录下的 .rflysim 文件夹
            )
        load_dotenv(config_path)
        self.api_key = os.getenv('VOLC_API_KEY')
        self.api_url = os.getenv('VOLC_ENDPOINT')
        self.model_id = os.getenv('VOLC_NAME')
```

4.3 步骤3：启动软件在环仿真

双击启动RflyUdpMavlinkRealSim.bat批处理文件，等待仿真环境初始化完成。脚本将会启动 1 个 QGC 地面站， 1 个 CopterSim 、1 个 RflySim3D 软件， 等待CopterSim软件下侧日志栏必须打印出 GPS 3D fixed & EKF initialization finished 字样代表初始化完成。



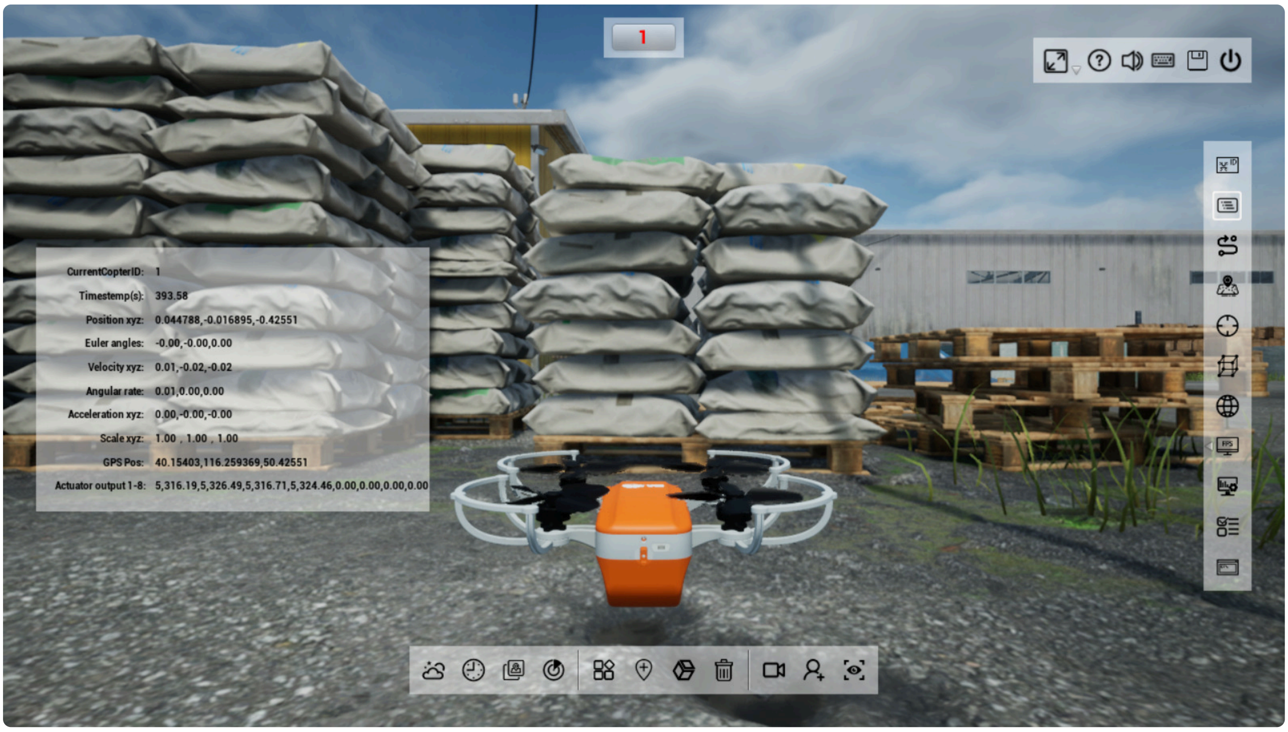
4.4 步骤4：运行控制程序

打开Visual Studio Code(或者其他的Python IDE 软件,需要保证该类软件的配置成功(可见:[RflySim安装目录]\RflySimAPIs\1.RflySimIntro\2.AdvExps\e3.PythonConfig),打开到本实验的路径下,打开程序 `ServerFile\main.py` ;并运行代码。

在终端输入："场外模式"，程序会调用大模型生成一段控制无人机开机起飞的代码，并运行。这样，无人机就起飞了

```
PS E:\AI标品例程\样品例程-6\1.软件在环实验\ServerFile> & C:/PX4PSP/Python310_backup/python.exe e
No Redis labs
HostIP is 192.168.31.32
Start listening CopterSim heartbeat Msg ...
Got time msg from CopterSim # 1 , not on this PC
Got time msg from CopterSim # 2 , not on this PC
Got time msg from CopterSim # 1 , running on this PC
End listening CopterSim heartbeat.
Got 2 CopterSim on the LAN.
智能体模式选择已启动，输入exit或quit则退出控制程序

请输入你的控制模式指令：场外模式
```

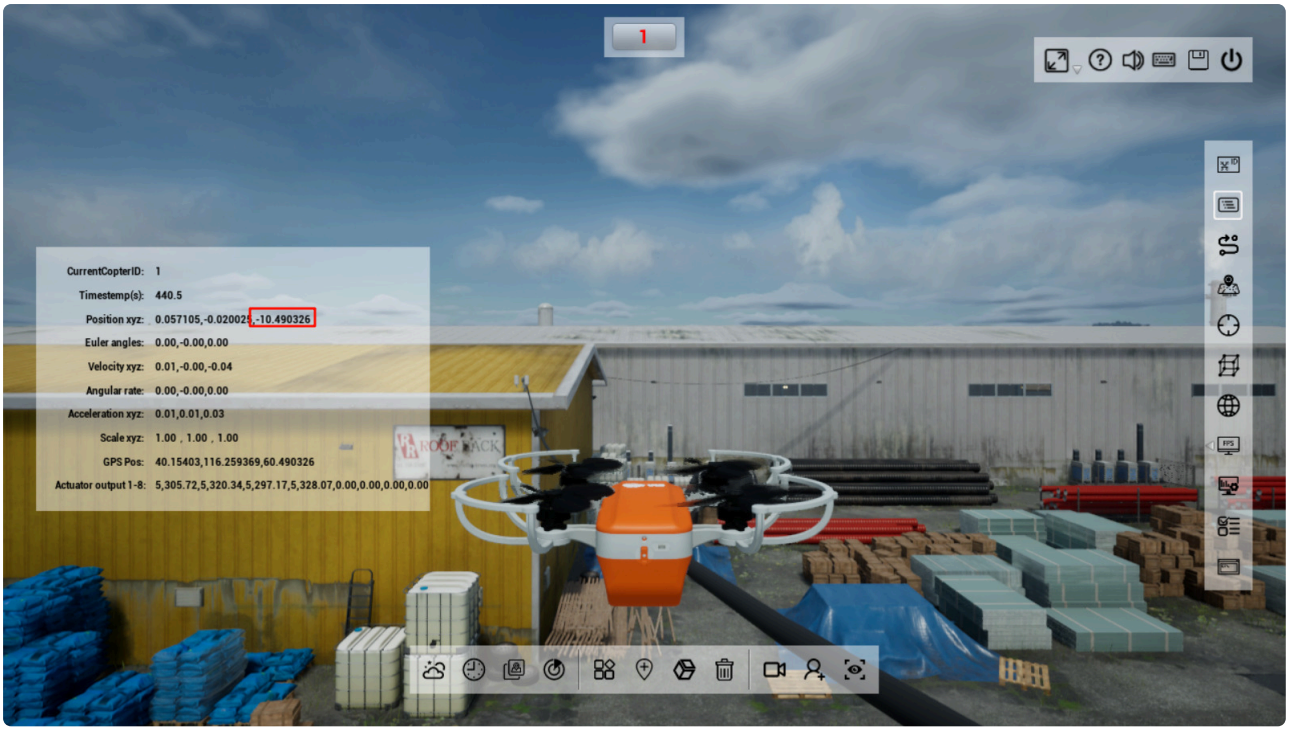
```
火山方舟 API 调用成功。
 - Executing parsed code:
  import time

  # Initialize offboard mode
  self.MavList[0].initOffboard()
  time.sleep(5)

  # Send initial position command (0, 0, -0.5, 0) in NED coordinates
  self.MavList[0].SendPosNED(0, 0, -0.5, 0)
  time.sleep(5)

  final_answer("Offboard mode initialized successfully. Drone is now in flight mode.")
```



接着输入指令："上升10米"，同样的，通过大模型生成了新的控制无人机的代码，使得无人机能够完成指令任务。

```
final_answer("Offboard mode initialized successfully. Drone is now in flight mode.")
< < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < < <
执行成功：Offboard mode initialized successfully. Drone is now in flight mode.

请输入你的控制模式指令：上升10米
```

```
火山方舟 API 调用成功。
─ Executing parsed code: ─────────────────────────────────────
  import time

  # Get current position
  current_x = self.MavList[0].uavPosNED[0]
  current_y = self.MavList[0].uavPosNED[1]
  current_z = self.MavList[0].uavPosNED[2]

  # Calculate new Z position (rise by 10 meters)
  new_z = current_z - 10

  # Send the new position command
  self.MavList[0].SendPosNED(current_x, current_y, new_z, self.MavList[0].uavAngEular[2])

  # Wait for the drone to reach the new position
  # Wait for the drone to reach the new position
  time.sleep(5)
```



5. 关键知识点

5.1 关键知识点1：任务自然语言解析与约束表达

用户在 Description.py 中配置提示词模板，将用户的自然语言问题转成结构化、约束清晰的任务描述。例如："请起飞至高度 10 米，再导航到目标区域并拍摄目标物"，在模板中会被细化为"步骤 1：起飞到 10m；步骤 2：飞到给定坐标；步骤 3：拍照并返回结果"等，引导 LLM 明确动作顺序、坐标系、单位等信息。

5.2 关键知识点2：配置层与环境变量读取

本实验将大模型的访问配置集中放在 *\文档\Ogre.env 中，例如 `VOLC_API_KEY`、`VOLC_NAME`、`VOLC_ENDPOINT` 等。OpenAI_api_Mavlink_Agent.py 和 volcEngineLLM.py 在运行时通过 `python-dotenv` /os.environ 加载这些环境变量，用于构造 HTTP 请求的 URL、认证头（密钥）和 `model` 字段。这样既保证了安全性（代码中不写死密钥），又方便在不同模型和供应商之间切换。

5.3 关键知识点3：LLM 调用封装与 CodeAgent 代码生成

在 OpenAI_api_Mavlink_Agent.py 或 volcEngineLLM.py 中，分别封装了"走 OpenAI/兼容接口"和"走火山引擎 DeepSeek-V3 接口"的两套调用逻辑：前者面向 OpenAI 风格的 Chat Completions 服务，后者面向火山引擎豆包/DeepSeek 系列模型。二者都会从 .env

中读取密钥、模型名和端点地址，按各自厂商的协议组装 HTTP 请求，并统一封装成可被 CodeAgent 调用的 `model`，例如：

```
agent = CodeAgent(tools=[...], model=OpenAI(...) or LiteLLMModel(...))
result = agent.run(natural_language_task)
```

CodeAgent 会驱动 LLM 输出完整的 Python 控制代码，而不仅是自然语言说明。生成的代码调用 Coordinate_Transformation.py 实现局部/全局坐标换算，调用 Communication_Mavlink.py 通过 MAVLink 控制无人机通讯指令等，也可以使用 OpenCV 等库进行图像处理和状态判断。

5.4 关键知识点4：主程序执行、反馈与多步智能体

生成的控制代码由 main.py 主程序加载并执行。main.py 会：

- 初始化仿真环境与通信模块，导入 Communication_Mavlink.py、Coordinate_Transformation.py 等工具模块；
- 在本地执行由 CodeAgent 生成的 Python 代码，实际下发起飞、移动、悬停等指令；
- 在每步执行后通过通信模块获取无人机状态反馈，并在终端打印或记录日志；
- 若发生异常（如连接中断、指令错误、坐标异常），触发异常回退或安全处理逻辑（如终止任务、降落等）。

部分复杂任务可以采用多步智能体（multi-step agent）模式：先由 LLM 规划任务步骤，再逐步生成和修正子任务代码，每一步执行后再将结果回传给 LLM，以"执行—观察—再推理"的闭环方式提升代码的鲁棒性和任务完成质量。

6.参考资料

1. smolagents官方文档 - Hugging Face提供的smolagents框架详细文档，包含CodeAgent的使用方法和原理介绍。
2. Building Code Agents with Hugging Face smolagents - deeplearning.ai提供的关于构建代码智能体的课程。
3. PX4 MAVLink开发者指南 - PX4飞控系统中MAVLink协议的详细说明文档。
4. RflySim - 本实验使用的无人机仿真平台官方网站，提供相关文档和教程。
5. 火山引擎大模型服务平台 - 本实验使用的LLM服务提供商，提供大模型API接入文档。
6. OpenAI API文档 - 与本实验兼容的OpenAI API接口文档，可用于替代实现。

7.常见问题

Q1：无法连接到火山引擎API，提示认证失败

A1：请检查以下几点：
Q1：确认在`*\文档\Ogre\.env`文件中正确配置了VOLC_API_KEY、VOLC_NAME、VOLC_ENDPOINT等参数。确保API Key有效且已在火山引擎控制台正确创建。检查网络连接是否正常，能否访问火山引擎API地址。

Q2：无人机无法正常起飞或执行指令

A2：请检查以下几点：

- 确认仿真环境已完全启动，CopterSim软件日志中显示"GPS 3D fixed & EKF initialization finished"。
- 检查输入的自然语言指令是否清晰明确，避免模糊表达。
- 查看控制台输出，确认生成的代码是否符合预期。

Q3：生成的控制代码不符合预期或执行出错

A3：这是由于LLM的随机性和理解偏差导致的，可以尝试以下解决方法：

- 修改自然语言指令的表述方式，使其更加明确具体。
- 在描述任务时增加约束条件，如坐标系说明、单位说明等。
- 如果多次尝试仍不能得到正确的代码，可以考虑手动调整生成的代码。

Q4：Python环境配置问题，无法运行程序

A4：请确保：

- 已正确安装Python 3.7及以上版本。
- 安装了必要的依赖库，包括openai、numpy、python-dotenv、requests、scipy等。
- 在VS Code中正确选择了Python解释器路径。

Q5：坐标转换结果不正确

A5：请注意坐标系的定义：

- 室外NED导航系：X轴指向北，Y轴指向东，Z轴指向地心（地面为Z=0，高于地面为负数）。
- 机体系：x轴指向机头方向，y轴指向机身右方，z轴指向机身下方。
- 角度单位应使用弧度制而非角度制。

Q6：程序运行时报错"未找到配置文件"

A6：请确认以下文件存在：

- `*\文档\Ogre\.env` 文件已创建并正确配置。
- 确保路径中没有中文或特殊字符干扰。

---

1. https://rflysim.com/ ↩
2. 推荐配置请见： https://rflysim.com/doc/zh/HowToInstall.pdf ↩