1. 实验名称及目的

阿克曼底盘无人车模型代码生成及软硬件在环仿真：在 Matlab 将 Simulink 文件编译生成阿克曼底盘无人车的 DLL 模型文件；并对生成的阿克曼底盘无人车模型在 PX4 官方控制器下进行软硬件在环仿真测试，通过本例程熟悉平台阿克曼底盘无人车模型的使用。

2. 实验原理

CarAckerman.slx 是基于最小系统模版建立的阿克曼底盘无人车动力学模型

2.1. 模型参数介绍（参考 API.pdf 中 DLL/SO 模型与通信接口的重要参数部分）

1) 重要参数

CarAckerman_init.m 中定义了阿克曼底盘小车模型的各种参数：

三维显示样式（载具）

```
ModelParam_uavType = int16(50); %这个参数决定了载具的三维显示样式，需要和 RflySim3D 的 XML 文件中的 ClassID 相匹配
```

初始位姿参数

```
ModelInit_PosE=[0,0,0]; %用于设置飞机的初始位置，对应了 CopterSim 上的 X 和 Y 初始值。最小模板的 Z 初始值会由 terrainZ 输入接口限制为 0
ModelInit_AngEuler=[0,0,0]; %用于设定飞机的初始姿态。飞机姿态角的前两位（俯仰和滚转角）可以通过 ModelInit_AngEuler 参数来配置，但是偏航角需要在 CopterSim 中配置。
```

QGC 中显示的地图坐标和高度原点

```
ModelParam_GPSLatLong = [40.1540302 116.2593683];%飞机初始的纬度和精度，单位度。
ModelParam_envAltitude = -50;%原点的海拔高度，竖直向下为正，高于海平面填负值，单位米。
```

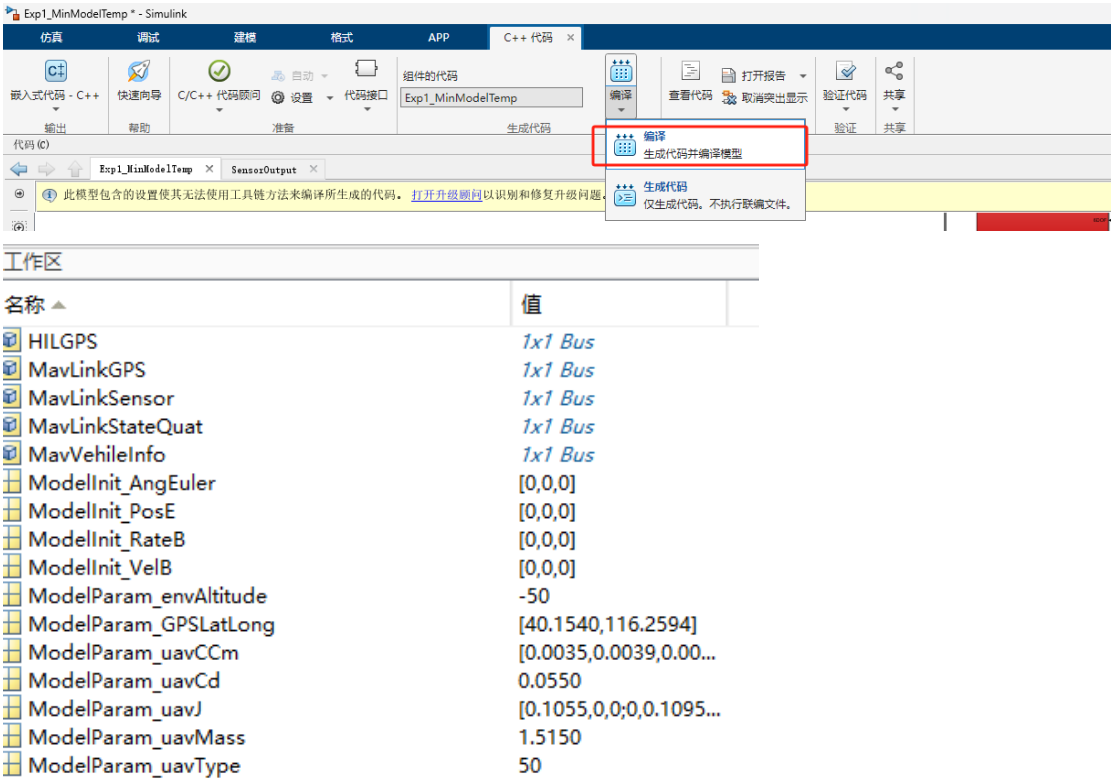
其余初始参数

```
ModelParam_uavMass=1.515; % 飞机质量:
ModelParam_uavJ= [0.0211*5,0,0;0,0.0219*5,0;0,0,0.0366*5]; % 转动惯量
ModelInit_VelB=[0;0;0];  % 速度向量，地球坐标系的 xyz 速度（z 向下为正）(m/s)
ModelInit_RateB=[0;0;0];                    % 体坐标系下飞机角速度 (rad/s)
%% 力和力矩模型参数
ModelParam_uavCd = 0.055;
ModelParam_uavCCm = [0.0035 0.0039 0.0034];
```

2) 参数调用过程

CarAckerman.slx 模型启动运行（编译）时会调用 CarAckerman_init.m

CarAckerman_init.m 中包含了模型的参数信息，本脚本会在 CarAckerman.slx 编译（编译所需环境配置参考 API.pdf 中的环境配置）时被调用将参数载入 MATLAB 工作空间。



GenerateModelDLLFile.p 是将 slx 模型转化为 DLL 模型文件的脚本，使用 RflySim 平台进行载具软硬件在环仿真时，需要将 DLL(windows 下)/SO（Linux 下）模型导入到 CopterSim，形成运动仿真模型，因此，在 Simulink 模型编译完成后，需要将模型对应的 C++文件打包成 DLL/SO 模型。

2.2. 输入信号（参考 API.pdf 中 DLL/SO 模型与通信接口的数据协议部分）

固定翼模型必须的 3 个输入数据包括飞控状态量输入、电机控制量和地形数据

1) 电机数据 inPwms

输入接口 inPWMs，16 维执行器控制量输入，已归一化到-1 到 1 尺度(通常电机是 0-1，

舵机是-1~1)，它的数据来自飞控回传的电机控制 MAVLink 消息 mavlink_hil_actuator_controls_t 的 controls，具体定义如下：

```
    typedef struct __mavlink_hil_actuator_controls_t {
    uint64_t time_usec; //时间戳，从开机后的时间，单位 ms
    uint64_t flags;  //标志位，用于显示当前的飞行状态
    float controls[16]; //控制量，16 维电机的控制量，发送到模型中，驱动飞机飞行
    uint8_t  mode;     // 模型，用于显示飞机当前的飞行模式和是否上锁等信息 })
mavlink_hil_actuator_controls_t;
```

软件在环仿真时，电机控制指令从 PX4 SITL 控制器通过 TCP 4561 系列端口以 MAVLink 协议发送到运动仿真模型的 inPWMs 接口，而硬件在环仿真时，该指令是从飞控通过串口以 MAVLink 协议发送到运动仿真模型的 inPWMs 接口。

2) 地形高度 terrainZ

最小模板默认 terrainz 值为 0，故只能使用平坦地形仿真

3) 自驾仪状态量输入 inCopterData

inCopterData 是 32 维 double 型数据，前 8 维存储 PX4 的状态，目前 1-6 维数据，依次为：

inCopterData(1)：PX4 的解锁标志位

inCopterData(2)：接收到的 RC 频道总数。当没有可用的 RC 通道时，该值应为 0。

inCopterData(3)：仿真模式标志位，0：HITL，1：SITL，2：SimNoPX4。

inCopterData(4)：CoperSim 中的 3D fixed 标志位。

inCopterData(5)：来自 PX4 的 VTOL_STATE 标志位。

inCopterData(6)：来自 PX4 的 LANDED_STATE 标志位。

2.3. 模型模块（参考 API.pdf 中的 Simulink 建模模板介绍）

这里为适配固定翼模型相对于最小模型模版，修改了模型参数（执行器初始参数等）和电机模块（转速动态响应）与力和力矩模块（驱动力、气动力、地面支撑和阻力等）两个自定义模块。

1) Motor Model 电机模块

在该模块中输入为 PWM 值（通过 inPWMs 接口获取），经过各电机的非线性动力学模型后得到各电机转速，该模块的输出分别为输入给力和力矩模型的电机转速（弧度每秒）；输入给 UE 的电机转速（转每分）

2) Force and Moment Model 力和力矩模块

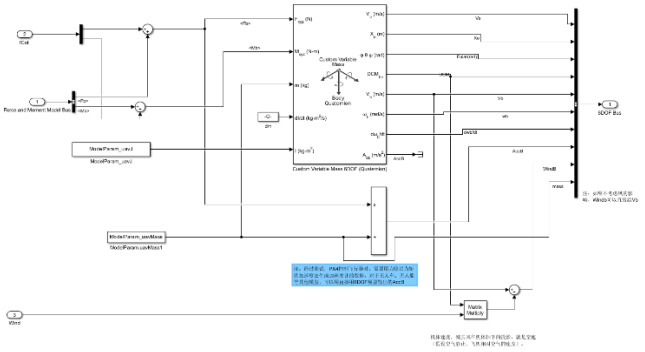
该模块输入为电机转速 MotorRads、飞机运动学姿态 6DOF 和地形高度输入 TerrainZ，输出为多旋翼合力、合力矩 Force and Moment Model Bus。

3) 6DOF 六自由度刚体运动学模块

用于描述无人机在空中运动时的姿态和位置变化 。考虑了无人机在三个坐标轴上的旋

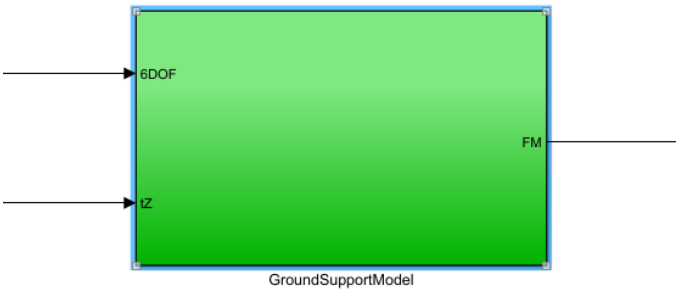
转运动（俯仰、横滚和偏航）以及机体与地球坐标系上的平移运动（前后、左右和上下）。

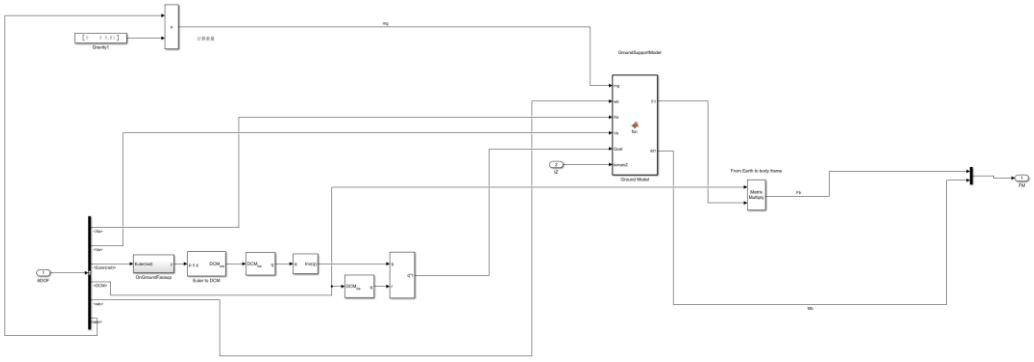
还可以根据实际需求对模型进行扩展，考虑更多的因素，如飞行器的非线性特性、气动力和惯性矩等。



4) GroundSupportModel 地面支撑模块

GroundSupportModel 地面支撑模块实际上是 PhysicalCollisionModel 碰撞检测模块的一个子模块，这里将所有物体简化为较为简单的基本几何体（例如圆柱体或者长方体）来计算其与地面之间的物理接触受力。



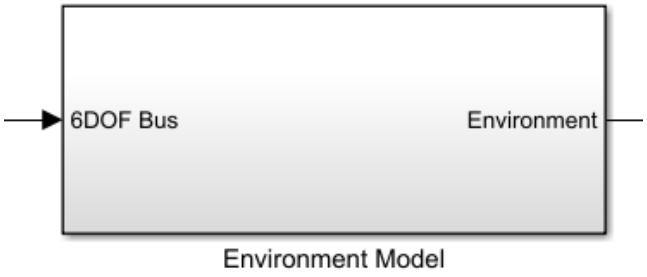


5) SensorOutput 传感器输出模块

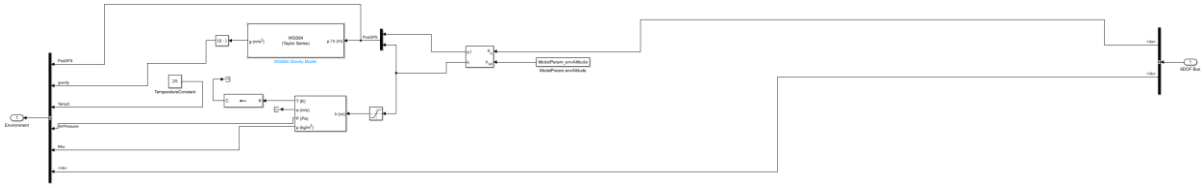
该模块中包括了环境模型、传感器模型和 GPS 模型

环境模型

环境模型对重力和大气压强对无人系统飞行产生的影响进行了模拟

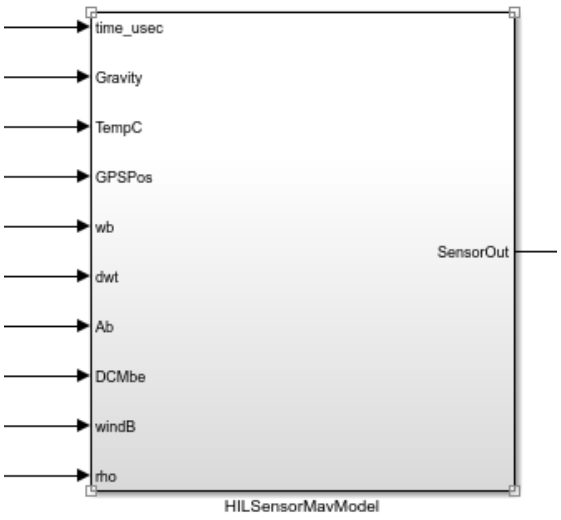


Environment Model

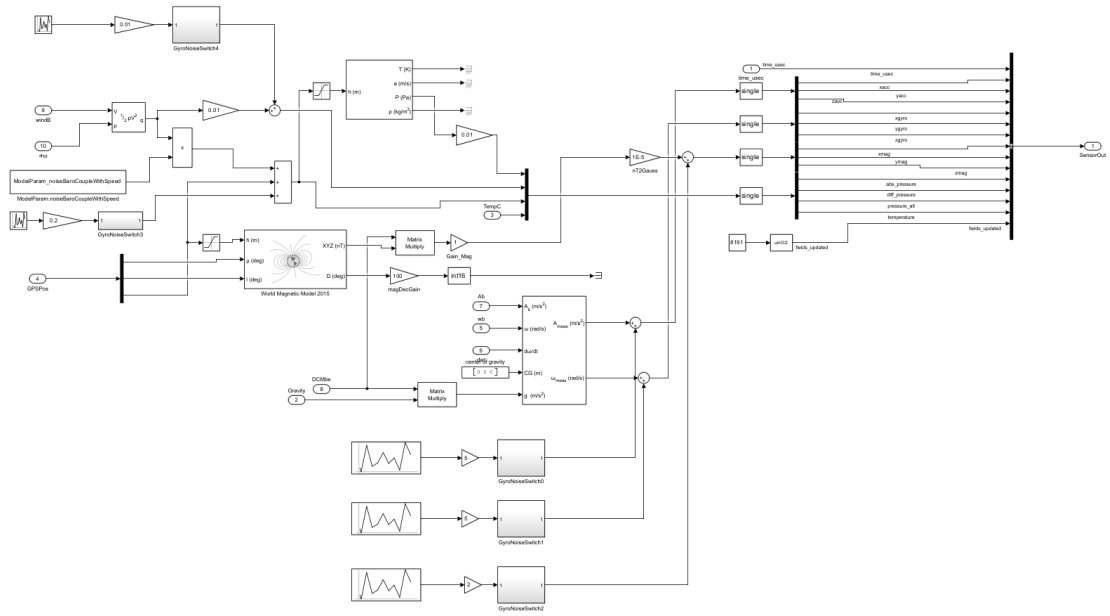


传感器模型

传感器模型中不仅对磁力计、惯性导航进行了建模，同时加入了噪声模拟

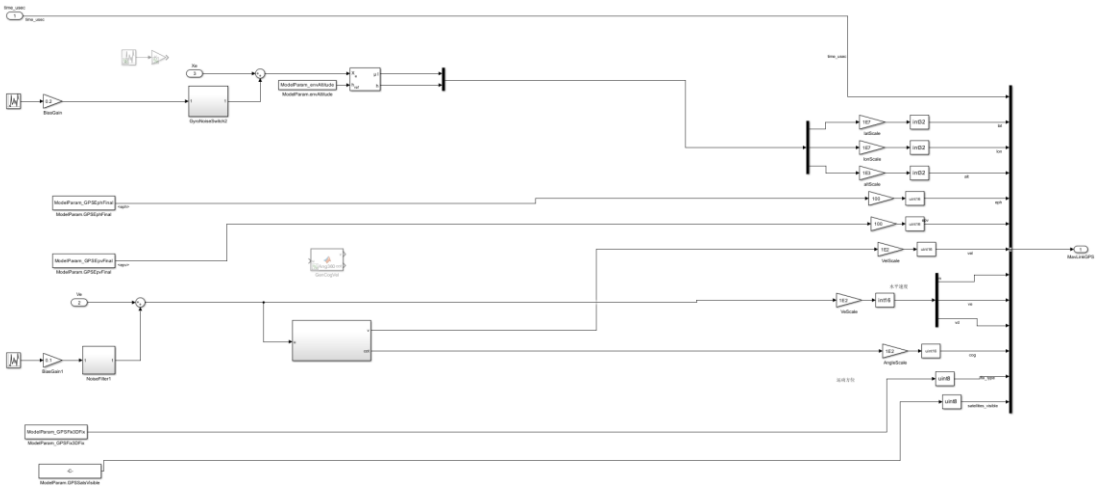


HILSensorMavModel



GPS 模型

GPS 模型用于计算 GPS 数据，在仿真时反馈回 PX4 控制器



6) 3DOutput 三维显示模块

该模块会将 CarAckerman_init.m 中的 ModelParam_uavType（三维显示 ID）、来自电机模型的 ActuatorToUE 以及来自 6DOF 模型的 6DOF Bus 的位置、速度、姿态和加速度等输出为 MavVehile3DInfo，并按协议对输入信息进行数据打包后通过该接口将数据发送至三维引擎

2.4. 输出信号（参考 API.pdf 中的数据协议）

CarAckerman_init.m 会调用 MavLinkStruct.mat 导入四个输出结构体 bus（MavLinkGPS、MavLinkSensor、MavLinkStateQuat 以及 MavVehileInfo）的定义。

```
load MavLinkStruct;
%load path;
```

最小模型模版包含了三个输出信号，分别是 MavHILSensor、MavHILGPS、MavVehile3DInfo。

1) MavHILSensor（传感器接口集合）

模型发送给 RflySim3D 的真实仿真数据，是平滑的理想值，这些数据可用于 Simulink 下的飞控与模型进行软件仿真测试。 对应了 MAVLink 的 mavlink_hil_sensor_t 消息，本结

构体包含了,加速度传感器的加速度值、陀螺仪传感器的角速度值、磁罗盘传感器的磁场值，气压和空速传感器的气压值等。这些传感器的值在仿真时由我们的模型提供，在真机飞行时由真实传感器芯片提供。

```
typedef struct __mavlink_hil_sensor_t {
 uint64_t time_usec; /*时间戳，单位毫秒 ms*/
 float xacc; /*机体坐标系 x 方向加速度，单位 m/s^2 */
 float yacc; /*机体坐标系 y 方向加速度，单位 m/s^2 */
 float zacc; /*机体坐标系 z 方向加速度，单位 m/s^2 */
 float xgyro; /*机体坐标系 x 方向角加速度，单位 rad/s */
 float ygyro; /*机体坐标系 y 方向角加速度，单位 rad/s */
 float zgyro; /*机体坐标系 z 方向角加速度，单位 rad/s */
 float xmag; /*机体坐标系 x 方向磁通量，单位 Gauss =T/10000*/
 float ymag; /*机体坐标系 y 方向磁通量，单位 Gauss =T/10000*/
 float zmag; /*机体坐标系 z 方向磁通量，单位 Gauss =T/10000*/
 float abs_pressure; /*绝对气压值，单位 millibar=100Pa*/
 float diff_pressure; /*相气压值，单位 millibar=100Pa*/
 float pressure_alt; /*气压解算高度值，单位 m*/
 float temperature; /*温度，单位摄氏度*/
 uint32_t fields_updated; /*传感器参数初始化标志位， bit 0 = xacc, bit 12: temperature, bit 31:
全部重新初始化 */
}) mavlink_hil_sensor_t;
```

2) MavHILGPS（GPS 接口）

模型发送给飞控的 GPS 数据值，它对应了 MAVLink 消息的 mavlink_hil_gps_t 结构体。输出信号中包含了经纬高、水平竖直精度、地速、北东地的速度、偏航角、定位状态和卫星数量等数据。 这些传感器的值在仿真时由我们的模型提供，在真机飞行时由真实 GPS 模块提供。

```
typedef struct __mavlink_hil_gps_t {
 uint64_t time_usec; /*时间戳，单位毫秒 ms*/
 int32_t lat; /*纬度(WGS84 地球模型)，单位度，再乘以 1E7*/
 int32_t lon; /*经度(WGS84 地球模型)，单位度，再乘以 1E7*/
 int32_t alt; /*高度 (AMSL 地球模型，而不是 WGS84)，单位 m，再乘以 1000 (向上为正)*/
 uint16_t eph; /*GPS 水平方向定位精度，单位 cm，如果不知道设为 65535*/
 uint16_t epv; /*GPS 竖直方向定位精度，单位 cm，如果不知道设为 65535*/
 uint16_t vel; /*GPS 地速，单位 cm/s，如果不知道设为 65535*/
 int16_t vn; /*GPS 地速朝北方向分量，单位 cm/s */
 int16_t ve; /*GPS 地速朝东方向分量，单位 cm/s */
 int16_t vd; /*GPS 地速朝下方向分量，单位 cm/s */
 uint16_t cog; /*运动方向，单位和范围 0~359.99 度，再乘以 100 degrees * 100，如果不知道设为 65535*/
 uint8_t fix_type; /*定位类型 0-1: no fix, 2: 2D fix, 3: 3D fix. */
 uint8_t satellites_visible; /*可见卫星数，如果不知道设为 255*/
}) mavlink_hil_gps_t;
```

注：GPS 数据的发送频率与真实传感器硬件基本相同为 10Hz，因此飞控的实时位置并不能靠 GPS 直接提供，需要与 IMU 等传感器进行融合滤波估计得到。

3) MavVehile3Dinfo（真实仿真数据输出）

模型发送给自驾仪的各种传感器数据的集合，对应了 MAVLink 的 mavlink_hil_sensor_t 消息。输出信号中包括了加速度传感器的加速度值、陀螺仪传感器的角速度值、磁罗盘传感器的磁场值，气压和空速传感器的气压值等。

```
struct SOut2Simulator {
    int copterID;  //飞机 ID，用于区分局域网内不同飞机
    int vehicleType;  //飞机样式，区分同种飞机（如四旋翼）下的不同样式（例如，大疆、AR.Drone）
    double runnedTime; //时间戳，当前时刻的时间，单位毫秒
    float VelE[3];   //速度向量，地球坐标系的 xyz 速度（z 向下为正），单位 m/s
    float PosE[3];   //位置向量，地球坐标系下的 xyz 方向（z 向下为正，单位 m，以起飞点为坐标原点
    float AngEuler[3];  //姿态角，飞机的欧拉角，定义于机体坐标系，单位弧度
    float AngQuatern[4]; //四元数，飞机姿态的四元数，定义于机体坐标系
    float MotorRPMS[8];  //电机转速，飞机的各个旋翼转速，单位转每分
    float AccB[3];      //加速度，飞机的运动加速度，单位 m/s^2
    float RateB[3];     //角速度，飞机的转动角速度，单位 rad/s
    double PosGPS[3];    //GPS 坐标，飞机的经纬高坐标，单位度、度、米
};
```

3. 实验效果

实现阿克曼底盘无人车 DLL 模型文件生成，以及完成阿克曼底盘无人车软硬件在环仿真。

4. 文件目录

文件夹/文件名称	说明
CarAckerman.slx	阿克曼底盘无人车模型文件。
CarAckerman_HITLRun.bat	硬件在环仿真批处理文件。
CarAckerman_SITLRun.bat	软件在环仿真批处理文件。
GenerateModelDLLFile.p	DLL 格式转化文件。
CarAckerman_init.m	动力学模型相关参数。
MavLinkStruct.mat	MavLink 数据结构体 mat 文件

5. 运行环境

序号	软件要求	硬件要求	
		名称	数量
1	Windows 10 及以上版本	笔记本/台式电脑①	1
2	RflySim 平台最新版	Pixhawk 6C②	1
3	MATLAB 2017B 及以上③	数据线	1

① 推荐配置请见：https://doc.rflysim.com/1.1InstallMethod.html

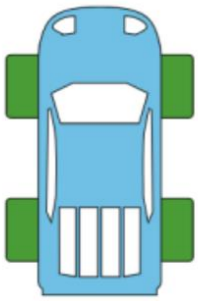
② 须保证平台安装时的编译命令为：px4_fmu-v6c_default，固件版本为：1.13.3。其他配套飞控请见：http://doc.rflysim.com/hardware.html

6. 实验步骤

两小车例程的区别 CarR1Diff 对应了差速小车（控制左右两侧轮子，进行差速转弯，可原地转弯后退），CarAckerman 对应了阿克曼底盘小车（控制后轮扭矩+方向盘，不能原地转弯，有转弯半径）

两个小车对应的机架类型。参考 PX4 的机架配置网页 https://docs.px4.io/main/en/airframes/airframe_reference.html#rover，可见目前的小车分为两类。其中，CarAckerman 对应了常规小车 Generic Ground Vehicle 和 CarR1Diff 对应了差速小车 Aion Robotics R1 UGV。

Rover



Name	
Generic Ground Vehicle	`SYS_AUTOSTART` = 50000 **Specific Outputs:** • **MAIN2**: steering • **MAIN4**: throttle
Aion Robotics R1 UGV	Maintainer: Timothy Scott `SYS_AUTOSTART` = 50003 **Specific Outputs:** • **MAIN0**: Speed of left wheels • **MAIN1**: Speed of right wheels

6.1. DLL 模型生成

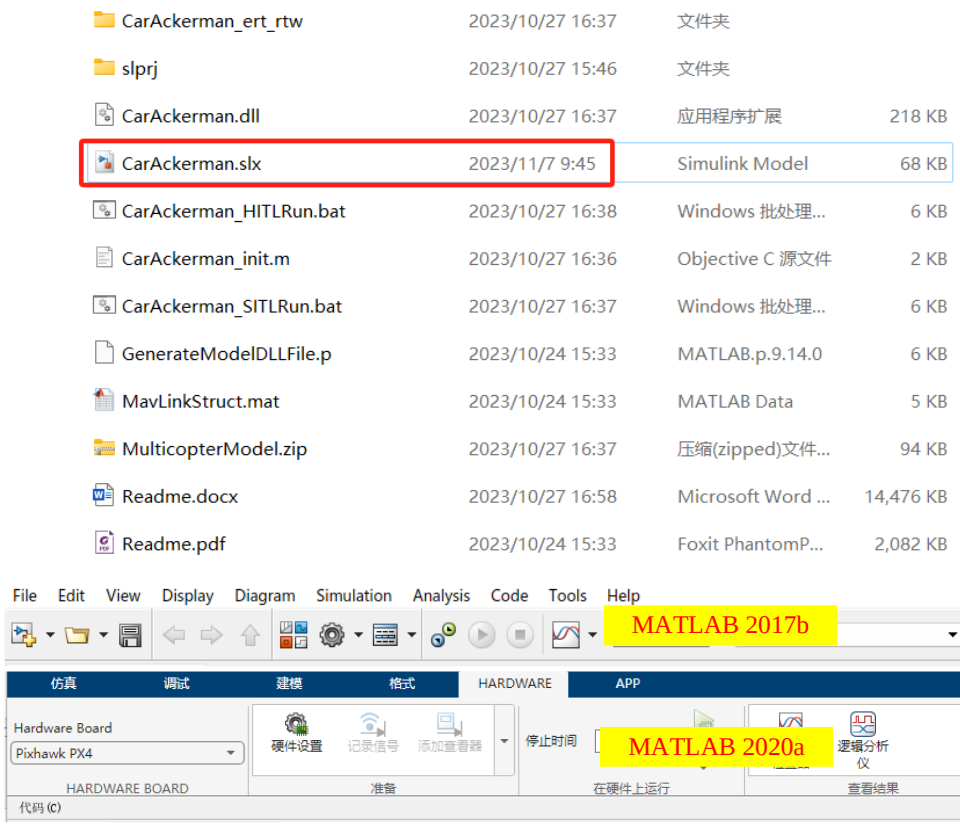
Step 1：

在 MATLBA 中打开“CarAckerman.slx” Simulink 文件，点击“Build Model”按钮。

与多旋翼模型先比，小车模型改动如下：

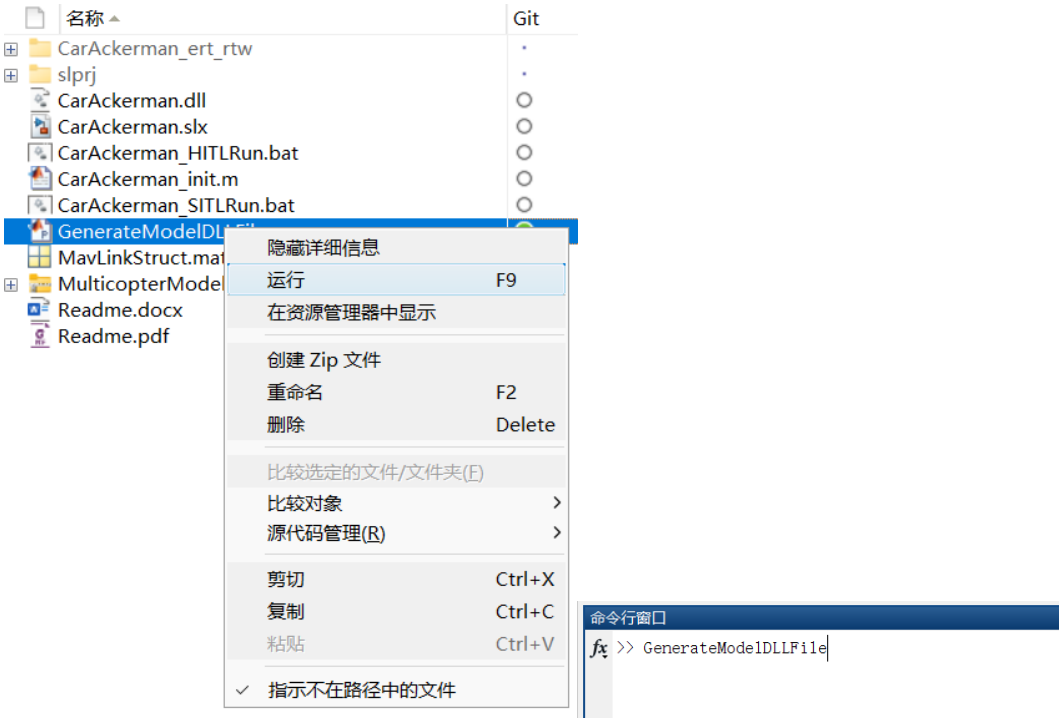
1）在 CarAckerman.slx 的 Motor Model 中，设置了只响应 2 和 4 号 PWM 信号，分别给了方向舵和驱动力矩，遵循 Generic Ground Vehicle 机架的混控通道。https://docs.px4.io/master/en/airframes/airframe_reference.html#rover。

2）在 CarAckerman.slx 的 Force and Moment Model/Ground Model 模型中，修改了地面模型，响应四个轮子的摩擦力，来驱动车前进。



Step 2：

编译完成后，右键 GenerateModelDLLFile.p 并点击运行（或者在 MATLAB 的命令行窗口中输入 GenerateModelDLLFile 后回车），即可以得到“AircraftMathworks.dll”的 DLL 模型文件。



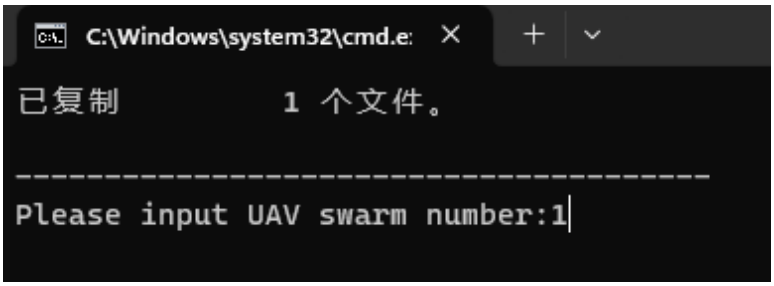
```
命令行窗口
   Copyright (C) Microsoft Corporation.  All rights reserved.

   /out:modeldllgen.exe
   /DLL
   /out:CarAckerman.dll
   modeldllgen.obj
   CarAckerman.obj
     正在创建库 CarAckerman.lib 和对象 CarAckerman.exp
   Compiling successfully, the CarAckerman.dll has been generated.
fx >>
```

6.2. 软件在环仿真

Step 1：

双击运行“CarAckerman_SITLRun.bat”批处理文件，在弹出的终端窗口中输入 1，启动一辆车的软件在环仿真。



注：在“CarAckerman_SITLRun.bat”软件在环的脚本文件中，需要设置对应无人车的 DLL 名：

```
REM Set use DLL model name or not, use number index or name string
REM This option is useful for simulation with other types of vehicles instead of multicopters
set DLLModel=CarAckerman
```

在 SimMode 处选择 CopterSim 中对应的软件在环仿真模式：

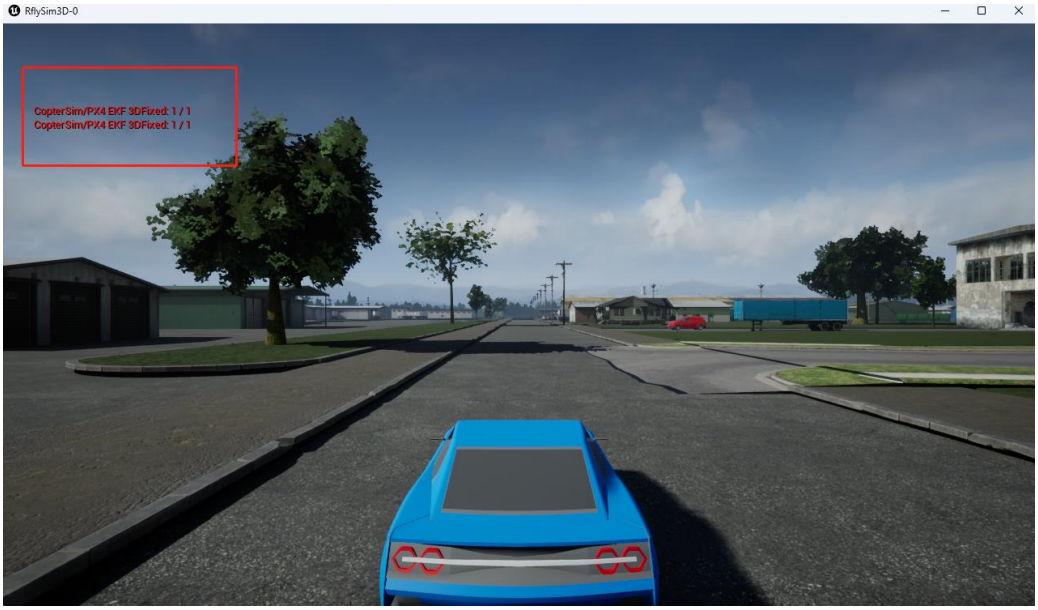
```
REM Set the simulation mode on CopterSim, use number index or name string
REM e.g., SimMode=2 equals to  SimMode=PX4_SITL_RFLY
set SimMode=2
```

在机架设置处设置无人车的对应机架，若不设置对应机架则仿真的默认机架则为四旋翼：

```
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 for supported airframes (Note: You can also create your airframe file here)
REM E.g., fixed-wing aircraft: PX4SitlFrame=plane; small cars: PX4SitlFrame=rover
set PX4SitlFrame=generic_ground_vehicle
```

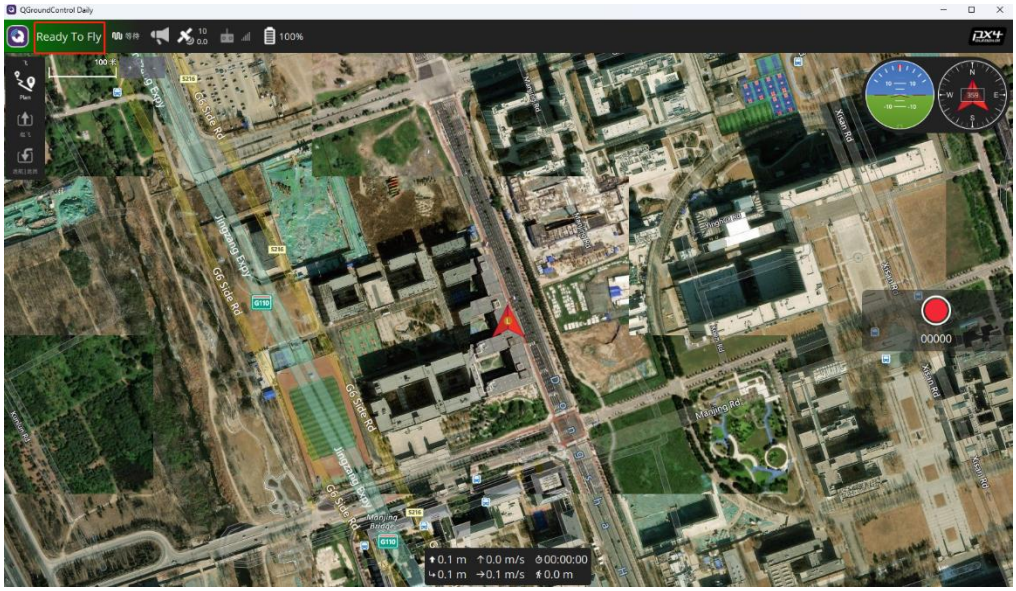
Step 2：

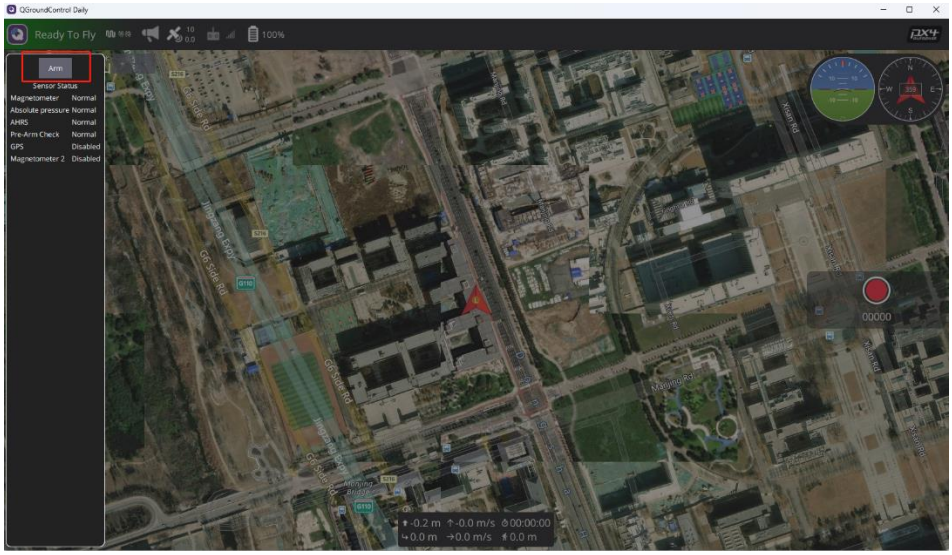
等待车辆初始化完成。

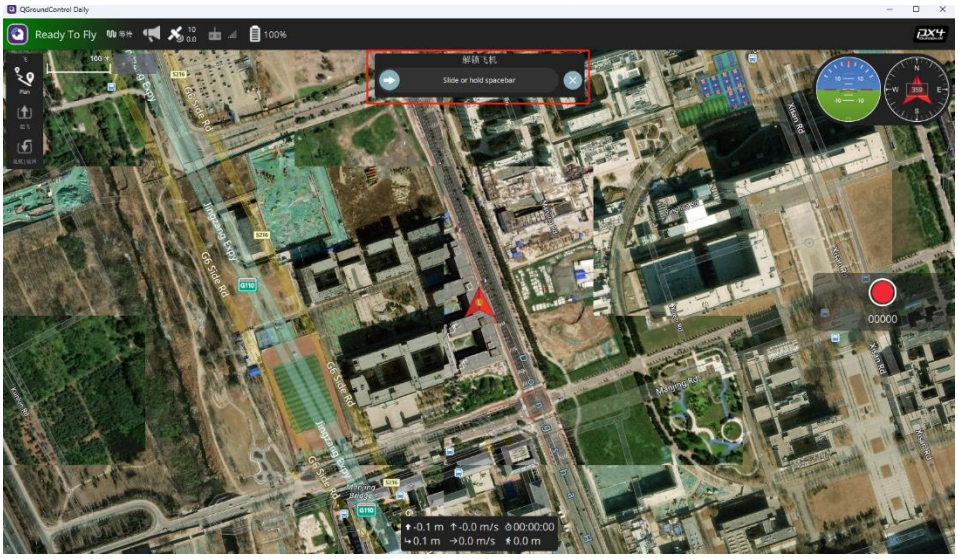


Step 3：

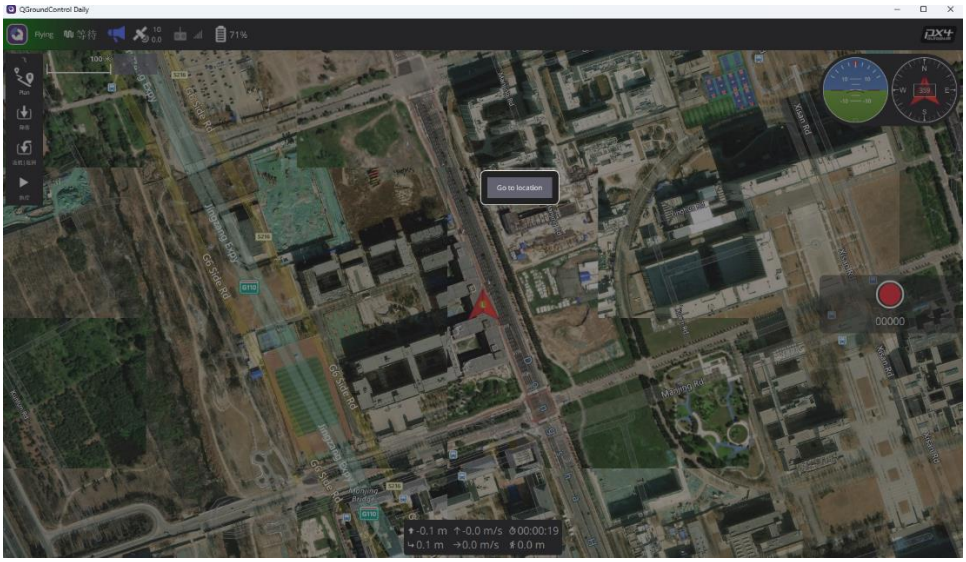
1、在 QGC 界面左上角点击“Ready To Fly”处进行解锁，最后滑动 QGC 上方解锁飞机进度条。



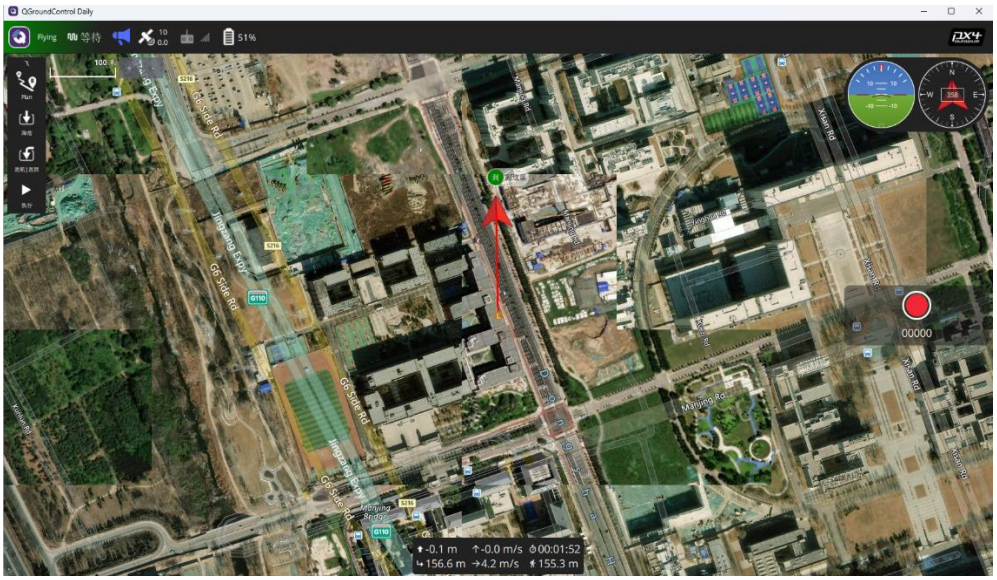




2、在 QGC 中点击地图后确认“Go to location”,滑动上方“将飞机移动到指定位置”完成目标位置确定。



3、在 UE4 与 QGC 中观察无人车的运动状态与运动轨迹。



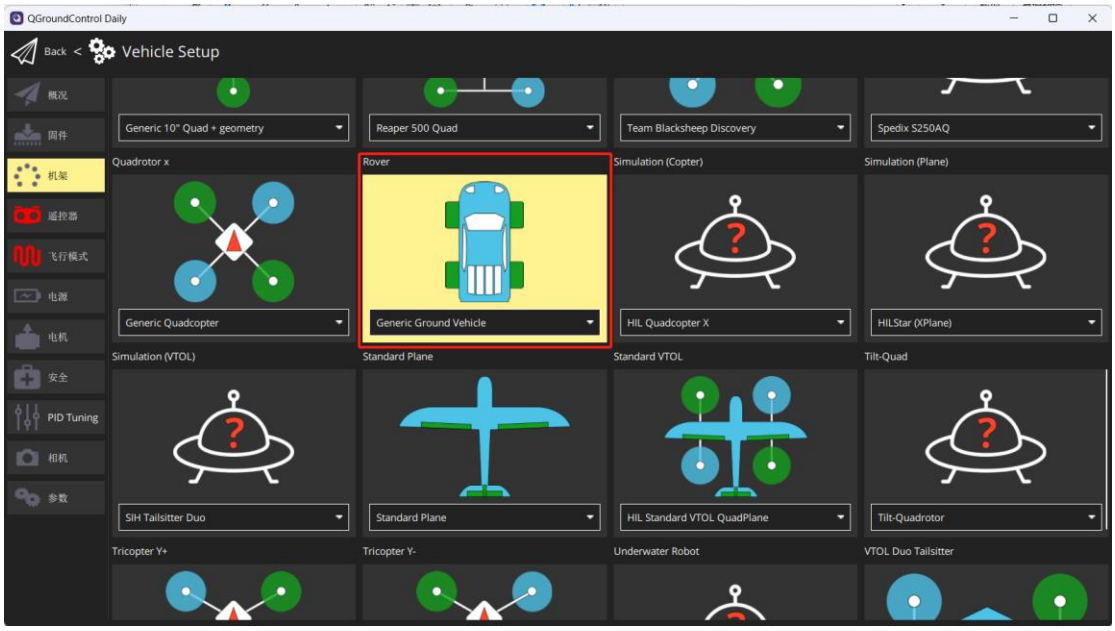
6.3. 硬件在环仿真

Step 1：

按下图所示将飞控与计算机连接。

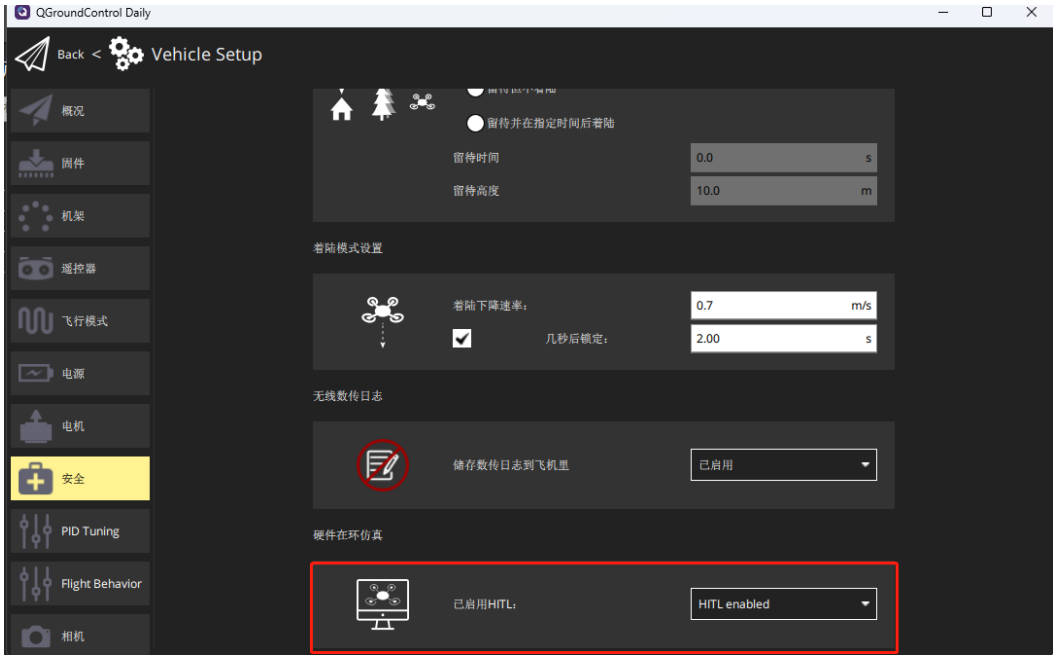
Step 2：

推荐使用 Pixhawk 6C 飞控进行硬件在环仿真，将飞控烧录至 1.13.3 固件版本，机架设置为“Generic Ground Vehicle”，点击 QGC 右上角的“应用并重启”。



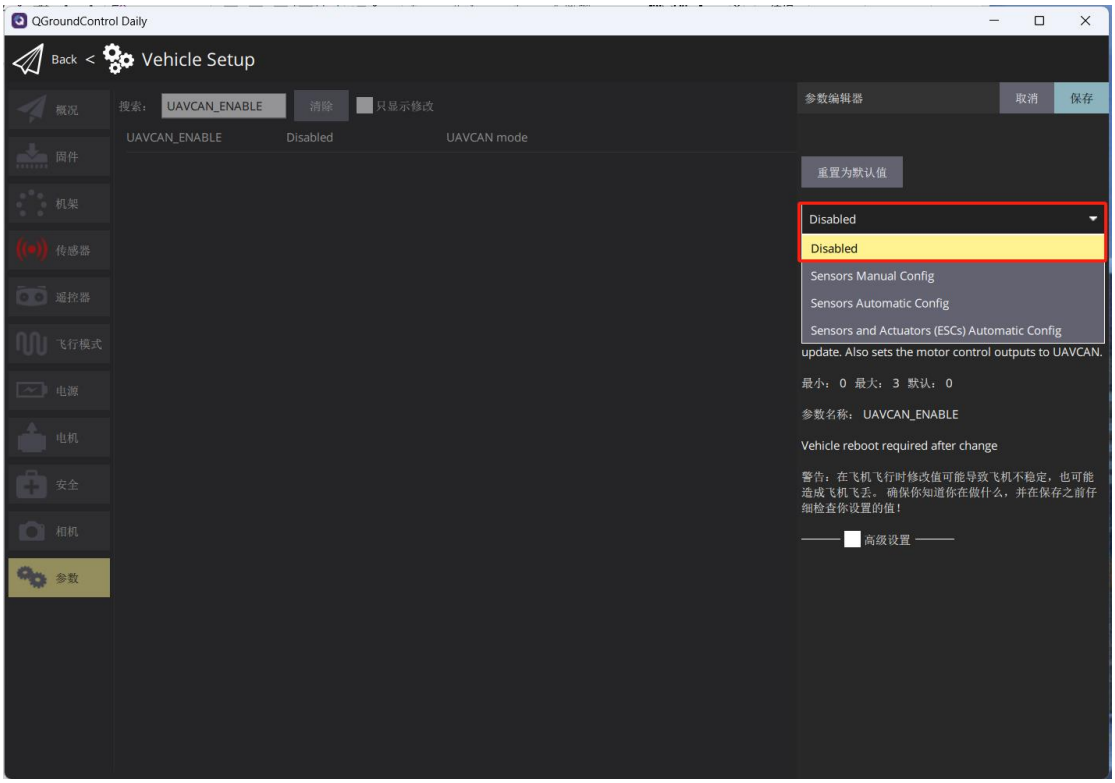
Step 3：

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

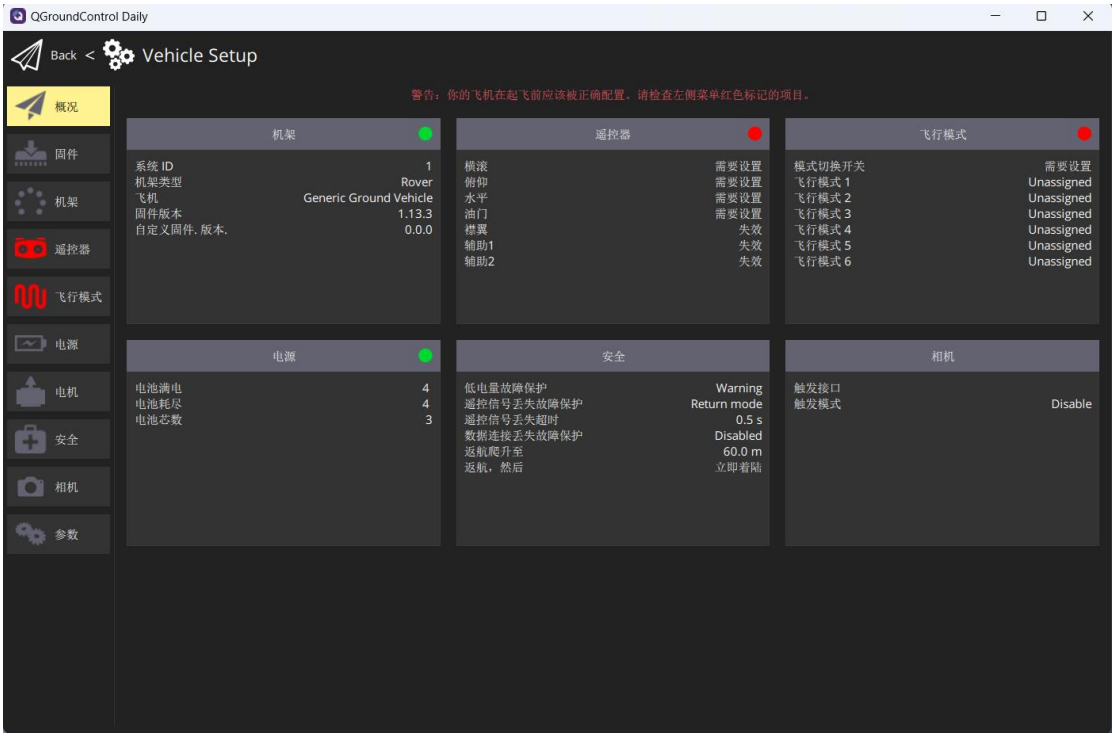


Step 4：

点击“参数”，在搜索栏中输入“UAVCAN_ENABLE”，在弹出框中设置为“Disabled”，保存后重新插拔飞控即可。



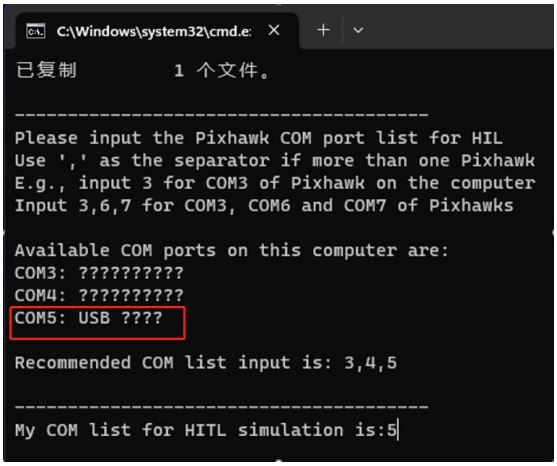
下图为完成硬件在环仿真相关配置后的示意图。



Step 5：

右键以管理员身份运行“CarAckerman_HITLRun.bat”批处理文件，在弹出的终端窗口中根据串口提示输入串口号 5，启动一辆无人车的硬件在环仿真。

CarAckerman_ert_rtw	2023/10/27 16:37	文件夹	
slprj	2023/10/27 15:46	文件夹	
CarAckerman.dll	2023/11/7 9:50	应用程序扩展	218 KB
CarAckerman.slx	2023/11/7 9:45	Simulink Model	68 KB
CarAckerman_HITLRun.bat	2023/10/27 16:38	Windows 批处理...	6 KB
CarAckerman_init.m	2023/10/27 16:36	Objective C 源文件	2 KB
CarAckerman_SITLRun.bat	2023/10/27 16:37	Windows 批处理...	6 KB
GenerateModelDLLFile.p	2023/10/24 15:33	MATLAB.p.9.14.0	6 KB
MavLinkStruct.mat	2023/10/24 15:33	MATLAB Data	5 KB
MulticopterModel.zip	2023/10/27 16:37	压缩(zipped)文件...	94 KB
Readme.docx	2023/10/27 16:58	Microsoft Word ...	14,476 KB
Readme.pdf	2023/10/24 15:33	Foxit PhantomP...	2,082 KB



注：在“CarAckerman_HITLRun.bat”硬件在环的脚本文件中，同样需要设置对应无人车的 DLL 名：

```
REM Set use DLL model name or not, use number index or name string
REM This option is useful for simulation with other types of vehicles instead of multicopters
set DLLModel=CarAckerman
```

在 SimMode 处选择 CopterSim 中对应的硬件在环仿真模式：

```
REM Set the simulation mode on CopterSim, use number index or name string
REM e.g., SimMode=0 equals to  SimMode=PX4_HITL
set SimMode=0
```

与软件在环仿真不同的是，在之前的配置准备环节中已经在 QGC 中设置了对应机架，所以在该脚本文件中不用设置机架。

Step 6：

之后测试步骤与软件在环仿真的 Step3 相同，运行之后观察阿克曼底盘无人车能否准确响应指令。

7. 参考文献

[1]. 无。

[2].

8. 常见问题

Q1：****

A1：****

无