Development and practice of intelligent Unmanned Swarm System Full-stack Development Case based on RflySim Toolchain

4 Vehicle motion modeling and simulation

飞思实验室

Outline

飞思实验室

Fesi LABS

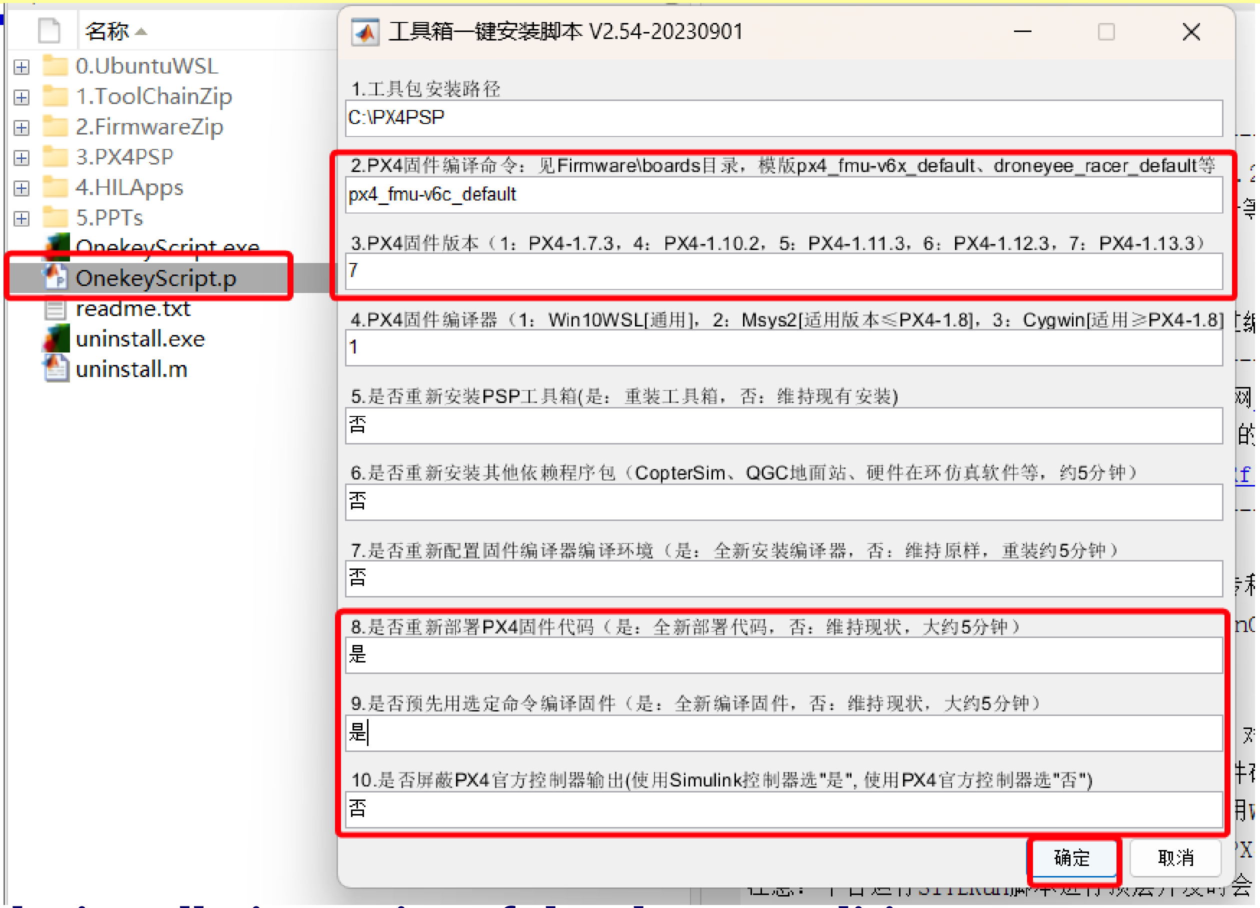
RflySim tutorial

1. Experimental pla

Note: This platform supports the simulation of different multi-rotor models by configuring parameters on CopterSim; It also supports the development of aircraft models in Simulink, and through the DLL model import way, realize the simulation of different multi-rotor, even fixed wing, car, boat and other diverse vehicles.

1.1 Pixhawk 6C flight control simulation environment restoration

- Re-run the one-click installation script "OnekeyScript.p"
- PX4 firmware edit command: "px4_fmu-v6c_default"
- Make sure the current PX4 firmware version is 7, which means PX4 version is: 1.13.3.
- Choose: "Yes", "yes", "no" whether to redeploy the px4 firmware code (8), whether to pre-compile the firmware with the selected command (9), and whether to block the official PX4 control output (10).



- If the above conditions are met, simply exit the installation script. If the above conditions are not met, you need to configure the appropriate options, click OK, and adjust the environment. For other flight control configuration, just modify the PX4 firmware edit command (2) to complete the simulation environment restoration.

飞思实验室

1. Experimental platform configu

Note: The main interface of CopterSim can select appropriate accessories and weight data to simulate different aircraft models, but please remember to restore the model parameters according to your own steps after each simulation to avoid affecting the subsequent experimental results.

1.2 CopterSim model parameter reduction

- In the desktop RflyTools folder, open the CopterSim shortcut and pop up the main interface.
- Click the "Model Parameters" button to enter the model configuration page.
- Click "Restore default parameters" and then "Store and Use Parameters" to restore the custom aircraft model data.
- Note: The above steps can be restored before by modifying the CopterSim main interface to configure different multi-rotor models or noise levels.
- Note: The above methods will not affect the parameters and noise of the DLL model, and the two systems are independent of each other.

1. Configuration of experi

Note 1: The example in this section needs to use the official firmware of PX4, any version can be used, here we choose the latest firmware

Note 2: If you do not have Pixhawk autopilot hardware on hand, you can skip the flight control hardware configuration content and directly perform software in the loop simulation

1.3 Restore flight control firmware

The example in this section requires the official firmware of PX4. If you are using a custom controller generated by Simulink, follow the steps on this page to restore the firmware:

1) Open QGC ground station software and disconnect Pixhawk;

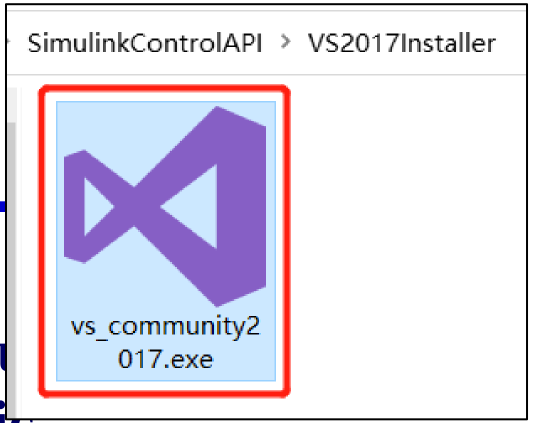
2) As shown on the right, click the gear icon in the toolbar to enter the vehicle setting page, and then click the "Firmware" TAB to enter the firmware burning page;

3) Connect the Pixhawk autopilot to the computer with a USB cable, and the software will automatically identify the Pixhawk hardware, as shown in the right picture, the firmware configuration window will pop up on the right side of the interface, check the first item "PX4 ***", and then click "OK", QGC starts to download automatically (requires networking, if you can't connect to the Internet, please refer to the next page to use local firmware) and install the latest PX4 firmware to Pixhawk;

1. Experimental platfor

Note: Many of the examples in this section will require the VS compiler, which is recommended to be installed in advance.

Note: The VS compiler can also be used in other versions, MATLAB can recognize it



1.4 Install the Visual Studio compiler

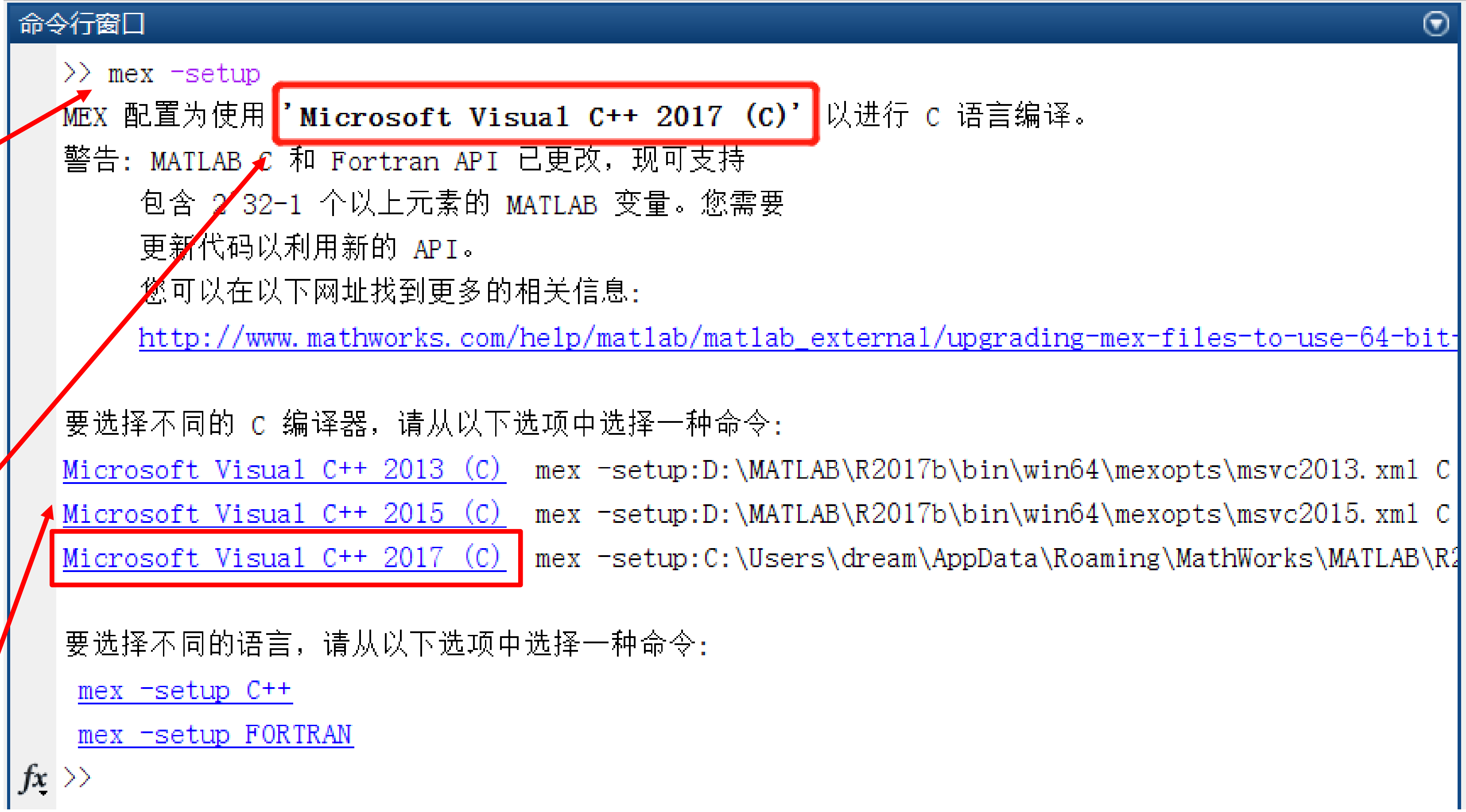
- The Visual Studio compiler will be used in many parts of the course, su the use of the MATLAB S-Function Builder module, and the automatic generation of C/C ++ model code by Simulink
- It is recommended to install Visual Studio 2017. The online installation steps are as follows:
- Double-click on "RflySimAPIs\SimulinkContro "

- This course content only needs to check "Desktop Development in C++" in the right picture.
- Note: A higher version of MATLAB can also install VS2019, but MATLAB can only recognize Visual Studio below its own version, so MATLAB 2017b will not recognize VS2019.
- Note: Please do not change the default VS installation directory (e.g. install to disk D), MATLAB will not recognize it.
- Cannot use Mingw compiler, need VS

1. Experiment platform configuration

1.5 MATLAB compiler installation confirmation

- In MATLAB's command line window, enter the command "mex-setup"
- In general, the VS 2017 compiler will be automatically recognized and installed, as shown in the picture on the right showing "MEX configuration uses' Microsoft Visual C++ 2017 'for compilation" indicates that the installation is correct
- If there are other compilers, this page can also switch to select other compilers such as VS 2013/2015

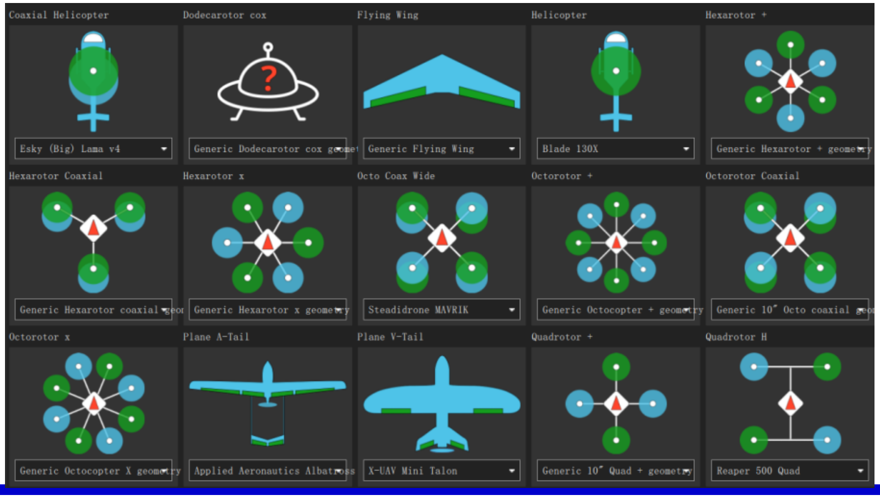


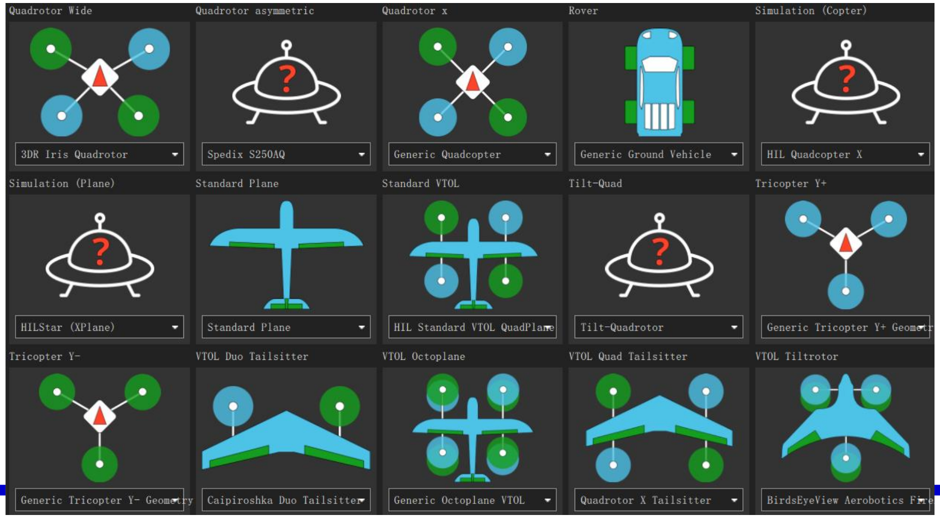
飞思实验室

1. Experiment platform configuration

1.6 Hardware/software in the loop simulation models supported by the platform

- The RflySim platform supports hardware/software in the loop simulation of any PX4 controllable model.
- All supported models can be viewed from the Airframe (rack) page of QGroundControl, as shown below.
- At present, the RflySim platform includes rotary-wing and fixed-wing models, and other models need to be built by users in Simulink.





1. Experimental platform

Note: The following steps can be automatically implemented in the bat script, here only need to know the implementation process.

1.7 Introduction to software/hardware in the loop simulation configuration

- Hardware-in-the-loop simulation process: CopterSim configuration model parameters or Simulink import DLL model QGC configuration Pixhawk enter corresponding rack QGC configuration enter hardware-in-the-loop simulation mode CopterSim start simulation
- Software in the loop simulation process: CopterSim configuration model parameters or Simulink import aircraft model configuration rack file in PX4 source code selected rack style in bat startup script one-click start software in the loop
  - Configuration file copy process in PX4 source code: "Firmware\ROMFS\px4fmu_common\init.d\airframes" folder to copy the machine files to "Firmware\ROMFS\px4fmu_common\ init.d-POSIx", For example, "6001_hexa_x" for hexacopter X and "2100_standard_plane" for fixed wing.
  - bat start script modification: Copy a copy of the SITLRun.bat file and change the model PX4SitlFrame to the non-digital part of the configuration file, for example, "set PX4SitlFrame=hexa_x" for the six rotor, "set PX4SitlFrame=standard_plane" for fixed wing, similar for other models.
  - Choose dedicated terrain: OldFactory terrain with flat runway is recommended for fixed-wing take-off. Three modifications are needed: Select the venue "SET /a UE4_MAP=OldFactory", x-coordinate "SET /a ORIGIN_POS_X=-250", and y-coordinate "SET /a ORIGIN_POS_Y=-119" to initialize onto the runway.

飞思实验室

Outline

1 Experiment platform configuration

2. Introduction of key interfaces (vehicle modeling template)

3. Basic Experimental Case (fr

4. Advanced Interface Experime dition)

5. Advanced Case Experiment (C ion)

6. Extended Case Studies (Full

7. Summary

Fesi LABS

RflySim tutorial

2.0 Vehicle modeling template and related interface experiments

The RflySim platform provides two sets of unified vehicle modeling templates, which are divided into the minimum system template and the maximum system template. The related interfaces and differences of the templates are summarized in the visible files: *\PX4PSP\ RflySimAPIs \4.RflySimModel\Readme.pdf, where the minimum system template contains the minimum required input and output interfaces, and the relevant verification experiment is visible: *\PX4PSP\ RflySimAPIs \4.RflySimModel\1.BasicExps\e1_MinModelTemp\Readme.pdf,

The largest system template contains more additional features, and the relevant validation experiment is shown in the following file: *\PX4PSP\RflySimAPIs\4.RflySimModel\2.AdvExps\e1_MaxModelTemp\Readme.pdf,

At the same time, this section also describes some of the vehicle model development process interface, related experiments are visible in the folder PX4PSP\ RflySimAPIs \4.RflySimModel\0.ApiExps

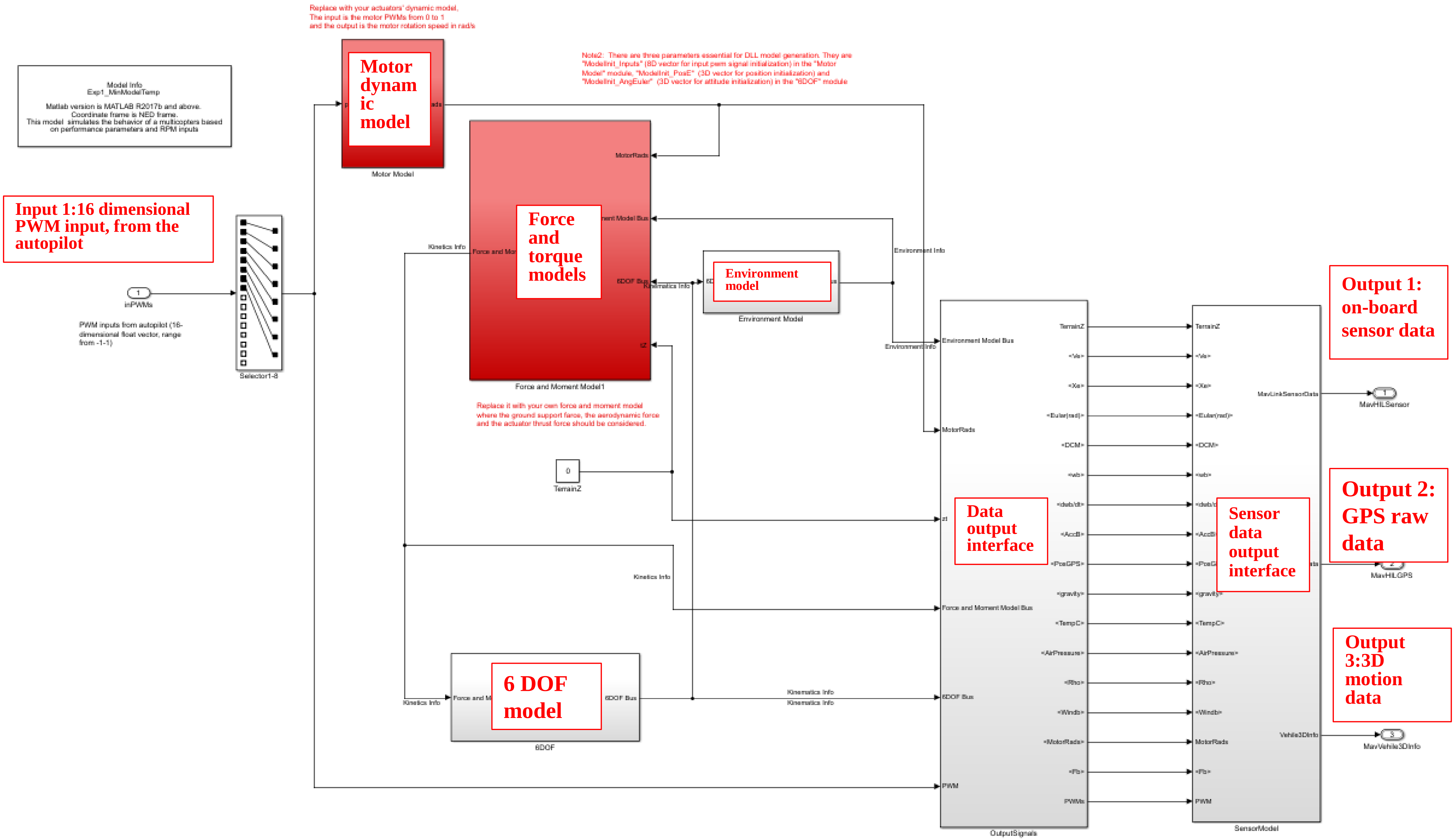
2. Vehicle modeling templates and related interface experiments

2.1 Minimum system model template

- 16 dimensional normalized actuator control inputs.
- It contains the acceleration value of the acceleration ser
gyroscope sensor, the magnetic field value of the magnetic compass sensor, the pressure value of the air pressure and airspeed sensor, and many other sensor data.
- The initial position and attitude of the aircraft can be customized, so as to be suitable for vertical takeoff aircraft such as missiles, and set the appropriate pitch and roll values.
- The real simulation data sent by the model to RflySim3D are smooth ideal values, unlike sensors and GPS modules that add noise and vibration.

Note: The input signal of the Advanced Experience version only supports the first 6 dimensional input (the data behind is all 0), and the full 16 dimensional input is supported in the Personal Advanced version and above.

More details on the minimum system template can be found at: PX4PSP\ RflySimAPIs \4.RflySimModel\1.BasicExps\e1_MinModelTemp\Readme.pdf -



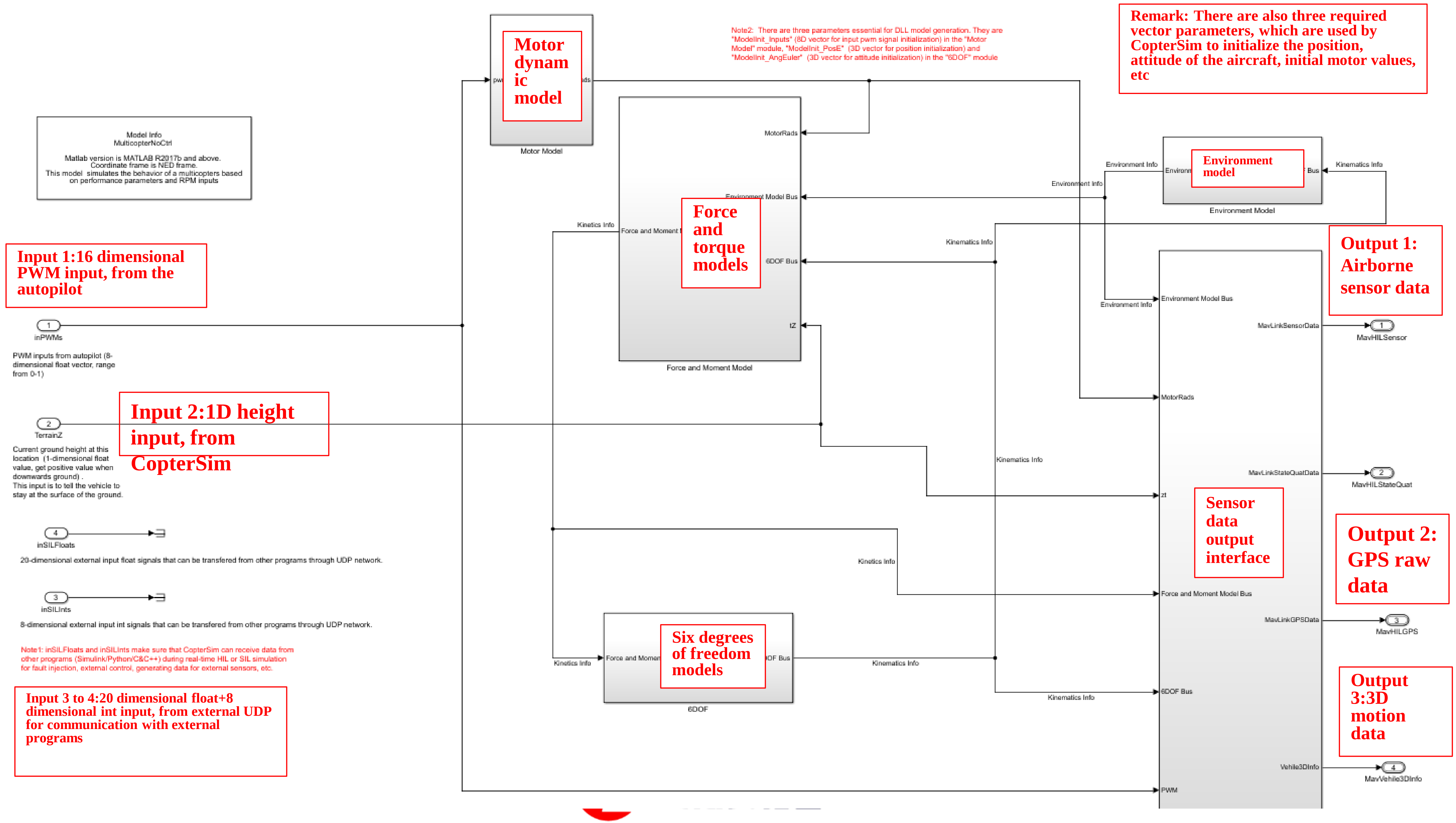
Replace with your actuators' dynamic model,
The input is the motor PWMs from 0 to 1
and the output is the motor rotation speed in rad/s
Model Info
Exp1_MinModelTemp
Matlab version is MATLAB R2017b and above.
Coordinate frame is NED frame.
This model simulates the behavior of a multicopters based on performance parameters and RPM inputs
Motor dynamic model
Motor Model
Note2: There are three parameters essential for DLL model generation. They are "ModelInit_Inputs" (8D vector for input pwm signal initialization) in the "Motor Model" module, "ModelInit_PosE" (3D vector for position initialization) and "ModelInit_AngEuler" (3D vector for attitude initialization) in the "6DOF" module
Input 1:16 dimensional PWM input, from the autopilot
inPWMs
PWM inputs from autopilot (16-dimensional float vector, range from -1-1)
Selector1-8
MotorRads
Force and torque models
Kinetics Info
Force and Moment Model1
Replace it with your own force and moment model where the ground support force, the aerodynamic force and the actuator thrust force should be considered.
Environment model
Environment Model
Environment Info
TerrainZ
Data output interface
OutputSignals
Sensor data output interface
SensorModel
Environment Model Bus
Force and Moment Model Bus
6DOF Bus
Kinematics Info
PWM
PWMs
MavLinkSensorData
MavHILSensor
MavHILGPS
Vehile3DInfo
MavVehile3DInfo
Output 1: on-board sensor data
Output 2: GPS raw data
Output 3:3D motion data
6 DOF model
6DOF

2. Vehicle modeling template and related interface experiments

2.2 Maximum system model template (only advanced collection version and above)

- **Retain the full functionality of the minimum system template.**
- **4. Add four input signal structures as follows: TerrainZ (1-D terrain height), inFloatsCollision (20-D collision engine), inSILInts (8-D integer) and inSILFloats (20-D floating point) were used to implement fault injection and external sensors.**
- **A new output signal structure named ExtToUE4PX4 is added to transmit other sensors or necessary data to the flight controller.**
- **A new fault motor was added for random fault injection, which simulated the crash of the plane.**

More details on the maximum system templates are available at: PX4PSP\ RflySimAPIs \4.RflySimModel\2.AdvExps\e1_MaxModelTemp\Readme.pdf

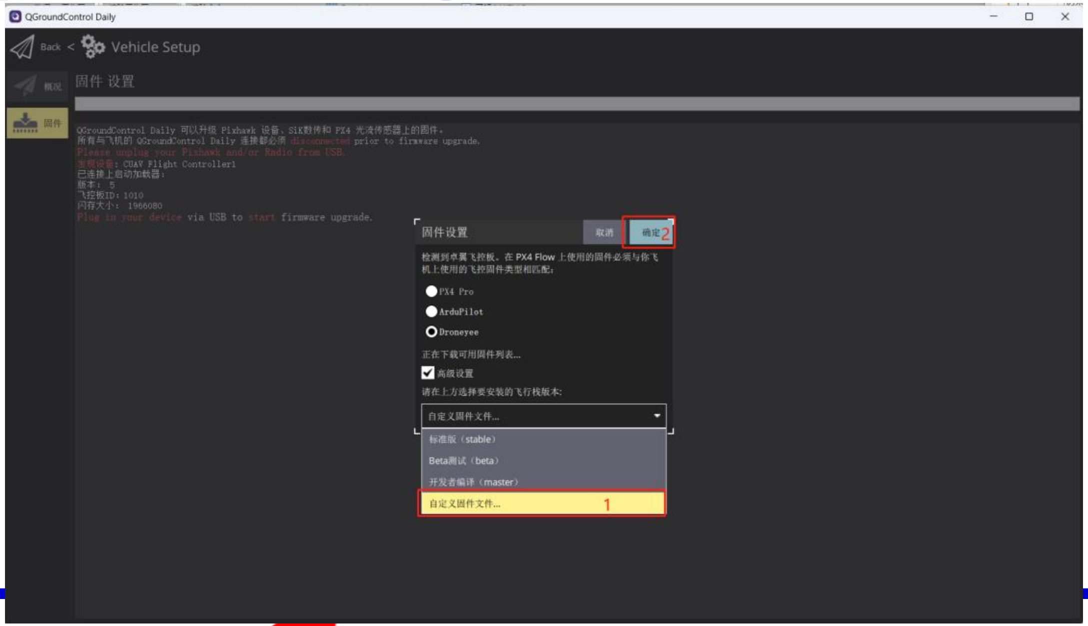


Motor dynamic model
Motor Model
Model Info
MulticopterNoCtrl
Matlab version is MATLAB R2017b and above.
Coordinate frame is NED frame.
This model simulates the behavior of a multicopters based on performance parameters and RPM inputs
Note2: There are three parameters essential for DLL model generation. They are "ModelInit_Inputs" (8D vector for input pwm signal initialization) in the "Motor Model" module, "ModelInit_PosE" (3D vector for position initialization) and "ModelInit_AngEuler" (3D vector for attitude initialization) in the "6DOF" module
Remark: There are also three required vector parameters, which are used by CopterSim to initialize the position, attitude of the aircraft, initial motor values, etc
MotorRads
Environment Model Bus
6DOF Bus
tZ
Force and torque models
Force and Moment Model
Kinetics Info
Environment Info
Kinematics Info
Environment model
Environment Model
Input 1:16 dimensional PWM input, from the autopilot
1
inPWMs
PWM inputs from autopilot (8-dimensional float vector, range from 0-1)
Input 2:1D height input, from CopterSim
2
TerrainZ
Current ground height at this location (1-dimensional float value, get positive value when downwards ground) .
This input is to tell the vehicle to stay at the surface of the ground.
4
inSILFloats
20-dimensional external input float signals that can be transfered from other programs through UDP network.
3
inSILInts
8-dimensional external input int signals that can be transfered from other programs through UDP network.
Note1: inSILFloats and inSILInts make sure that CopterSim can receive data from other programs (Simulink/Python/C&C++) during real-time HIL or SIL simulation for fault injection, external control, generating data for external sensors, etc.
Input 3 to 4:20 dimensional float+8 dimensional int input, from external UDP for communication with external programs
Six degrees of freedom models
Force and Moment
6DOF
Output 1: Airborne sensor data
Output 2: GPS raw data
Output 3:3D motion data
Sensor data output interface
Environment Model Bus
MotorRads
zt
Force and Moment Model Bus
6DOF Bus
PWM
MavLinkSensorData
MavLinkStateQuatData
MavLinkGPSData
Vehile3DInfo
1
MavHILSensor
2
MavHILStateQuat
3
MavHILGPS
4
MavVehile3DInfo

2. Vehicle modeling template and related interface experiments

2.3 Flight control firmware generation experiment on RflySim platform

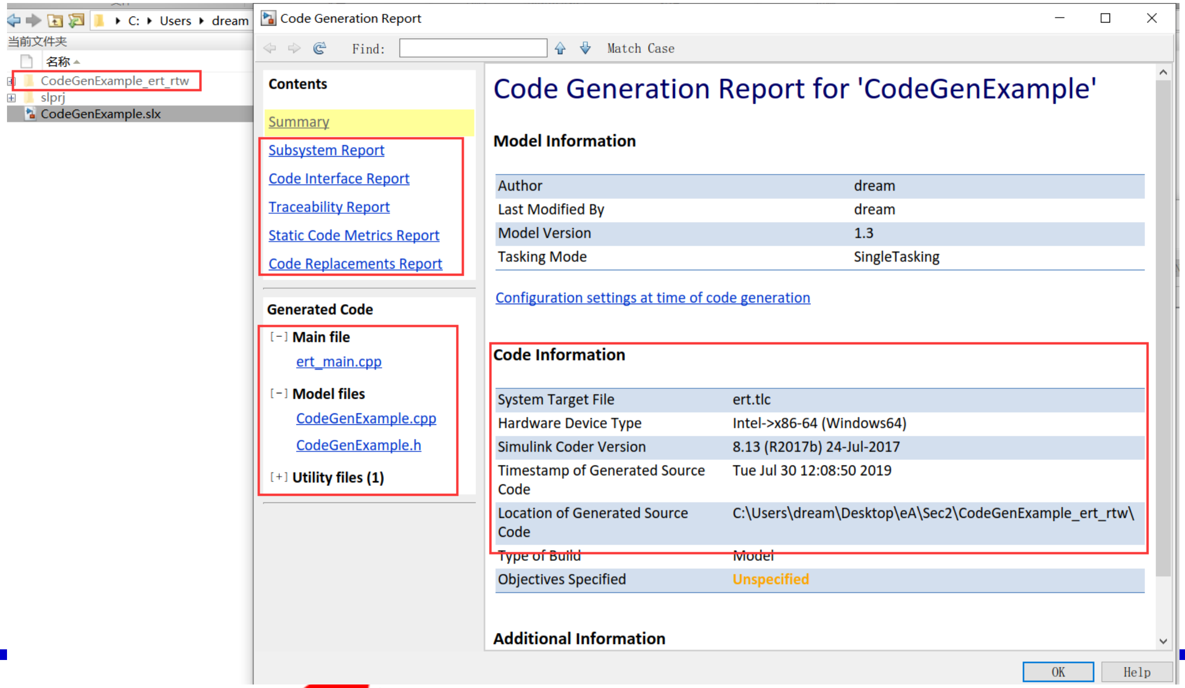
Through this routine, users can understand how to configure the one-click installation script of the platform, how to use the platform to complete the underlying self-developed controller firmware generation and native firmware generation. The detailed experimental operation steps are as follows: *\PX4PSP\RflySimAPIs\4.RflySimModel\1.BasicExps\e0_ModelAPIUsage\1.PX4FirmwareGen\Readme.pdf.



2. Vehicle modeling template and related interface experiments

2.4 RflySim platform generated C/C++ code experiments independently

This routine is used to introduce how to automatically generate C/C++ code for Simulink models. See *\PX4PSP\RflySimAPIs\4.RflySimModel\1.BasicExps\e0_ModelAPIUsage\ 2.UserDefinedC++\Readme.pdf for the detailed experimental operation steps.



飞思实验室

Outline

1 Experiment platform configuration

2. Introduction of key interfaces

3. Basic Experimental Case (free version)

4. Advanced Interface Experiment (Personal Edition)

5. Advanced Case Experiment (Collective Edition)

6. Extended Case Studies (Full version)

7. Summary

Fesi LABS

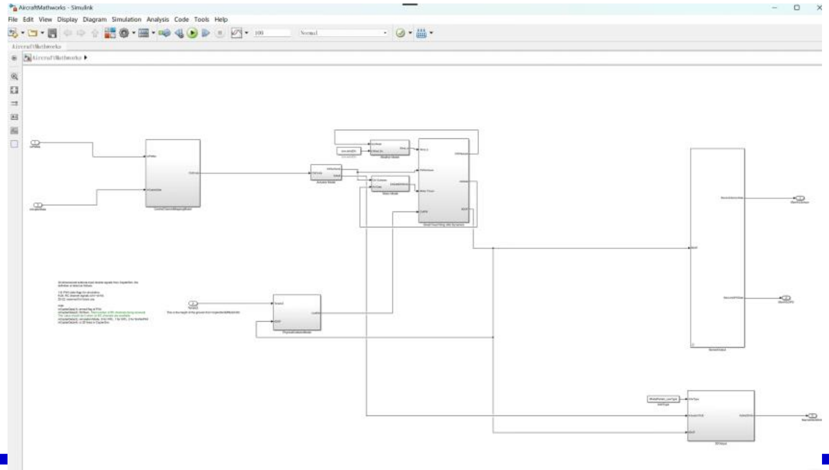
RflySim tutorial

3. Basic Experiment (platform fixed wing)

3.1 DLL generation and SIL/HIL experiment of fixed-wing aircraft model

The fixed-wing Simulink model was compiled and DLL model files were generated in Matlab, and the hardware-software in the loop simulation of the fixed-wing UAV was carried out by uploading the route through QGC.

Specific experimental operation steps, please see: * PX4PSP/RflySimAPIs / 4. RflySimModel / 1. BasicExps e2_FixWingModelCtrl/Readme. PDF.



飞思实验室

3. Basic Experiments (Other vehicles)

3.2 Code generation of Ackermann chassis unmanned vehicle model and software and hardware in the loop simulation experiment

In Matlab, the Simulink files were compiled to generate the DLL model files of Ackermann chassis unmanned vehicle. The software and hardware in the loop simulation test of the generated Ackman chassis unmanned vehicle model was carried out, and the use of the platform Ackman chassis unmanned vehicle model was familiar with through this routine. The detailed experimental operation steps are as follows: *\PX4PSP\ RflySimAPIs \4.RflySimModel\1.BasicExps\e3_CarAckermanModeCtrl\Readme.pdf.

飞思实验室

3. Basic Experiments (Other vehicles)

3.3 Code generation of differential unmanned vehicle model and simulation experiment of software and hardware in the loop

The Simulink files were compiled to generate the DLL model files of the differential unmanned vehicle in Matlab. The software and hardware in the loop simulation test was carried out on the generated differential unmanned vehicle model, and the use of the platform differential unmanned vehicle model was familiar with through this routine. The detailed experimental operation steps are as follows: *\PX4PSP\ RflySimAPIs \4.RflySimModel\1.BasicExps\e4_CarR1DiffModelCtrl\Readme.pdf.

Outline

1 Experiment platform configuration

2. Introduction of key interfaces

3. Basic Experimental Case (free version)

4. Advanced Interface Experiment (Personal Edition)

5. Advanced Case Experiment (Collective Edition)

6. Extended Case Studies (Full version)

7. Summary

Fesi LABS

RflySim tutorial

飞思实验室

4. Advanced interface experiments

4.0 Overview of advanced interfaces

Compared with the introduction of key interfaces in Section 2, this subsection mainly introduces some advanced interface experiments during the development of vehicle models, as follows:

Name of experiment	Routine path
Read state estimates for external communication experiments	0.ApiExps\3.ExtCtrlAPI-UDP20100\Readme.pdf
Reading simulation truth data for external communication experiments	0.ApiExps\4.ExtCtrlAPI-UDP30100\Readme.pdf
Obtaining platform rfly_px4 uORB message for external communication experiments	0.ApiExps\5.ExtCtrlAPI-UDP40100\Readme.pdf
ExtToUE4 interface validation experiment	0.ApiExps\6.ExtToUE4\Readme.pdf
ExtToPX4 interface validation	0.ApiExps\7.ExtToPX4\Readme.pdf

Advanced interface experiments

4.0 Overview of advanced interfaces

Name of experiment	Routine path
Max system model OutCopterData interface validation experiment	0.ApiExps\9.OutCopterData\readme.pdf
Fault module dynamic modification parameter experiment	0.ApiExps\10.FaultParamsDynMod\Readme.pdf
Physics engine validation experiment for InFloatsCollision	0.ApiExps\11.InFloatsCollision\Readme.pdf
InSILInts and InSILFloats interface experiments	0.ApiExps\12.InSILInts&Floats\Readme.pdf

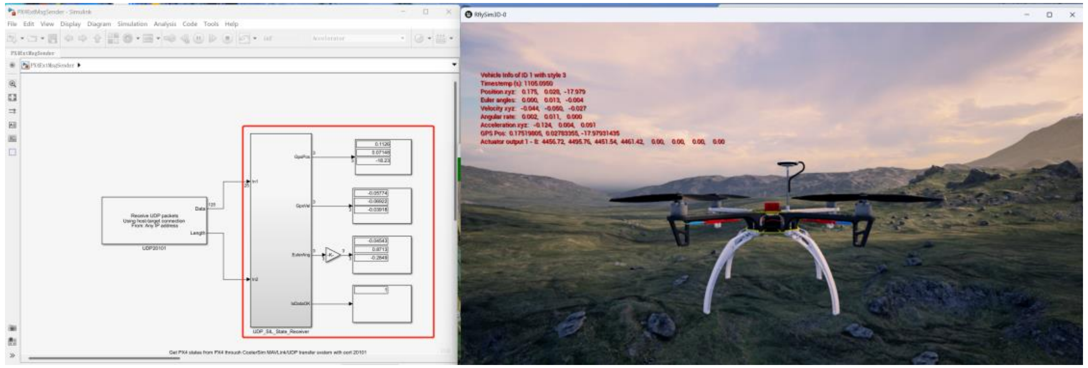
4. Advanced interface experiments

inCopterData input interface experiment	**In the model routines of the platform, in addition to several necessary input and output interfaces that serve the basic functions of the platform, there are also some input interfaces that can send some more detailed vehicle simulation information, among which inCopterData is the 32-dimensional input interface that CopterSim sends to DLL models. Among them, dimensions 1 to 8 represent PX4 status flags in simulation; In this experiment, inCopterData (5) is used to make users familiar with the use of the input interface.**	0.ApiExps\14.inCopterData\1.PX4_State_flags\Readme.pdf
inCopterData input interface experiment	Dimension 9~24 is the channel signal of the remote control in the simulation; In this experiment, inCopterData (9~24) is used to make users familiar with the use of the input interface.	0.ApiExps\14.inCopterData\2.RC_channel_signals\Readme.pdf
inCopterData input interface experiments	Among them, 25~32 dimensions can listen to rfly_px4 messages; The experiment is designed by inCopterData (25~32) to make users familiar with the use of the input interface.	0.ApiExps\14.inCopterData\3.rfly_px4\Readme.pdf

Advanced interface experiments

4.1 Read State Estimates for External Communication Experiments (Personal Advanced only)

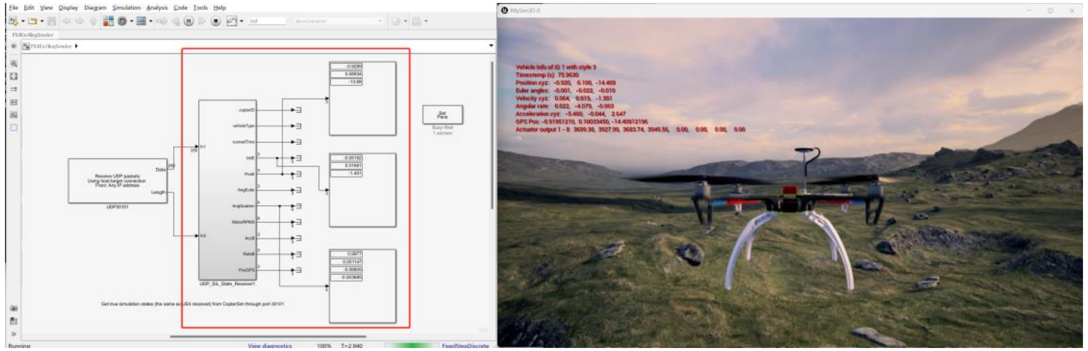
When using the RflySim platform, you can perform software/hardware in-the-loop simulation in UDP_Full mode, and receive PX4 internal state estimates by listening to UDP20100 series ports. The detailed experimental operation steps are as follows: PX4PSP\ RflySimAPIs \4.RflySimModel\0.ApiExps\3. extCTRLAPpi-UDP20100 \Readme.pdf.



4. Advanced interface experiments

4.1 Reading simulation Truth Data for External Communication Experiments (Personal Advanced Edition and above only)

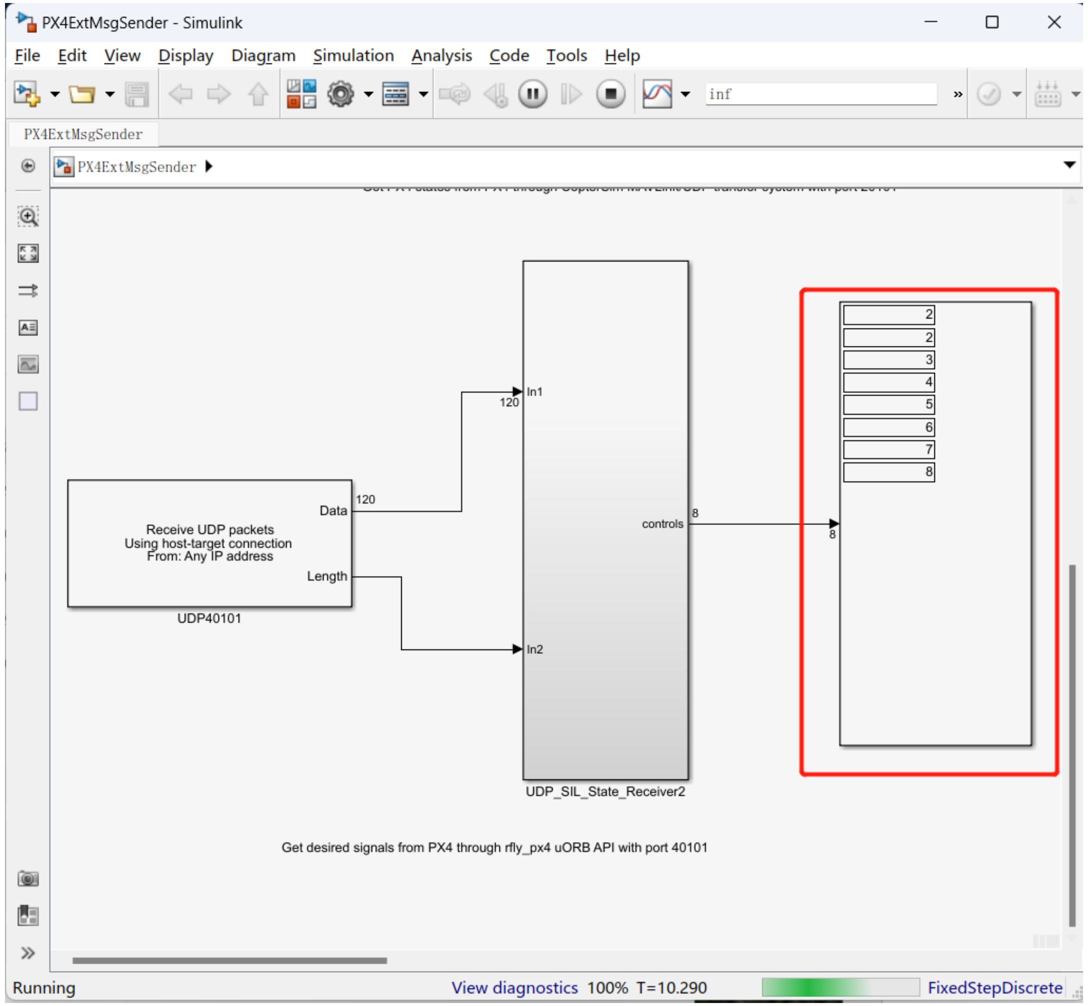
When using the platform (UDP/MAVLink mode) to perform software /hardware in-the-loop simulation, the CopterSim flight simulation real data can be received by listening to the UDP30100 series port. For detailed experimental operation steps, please refer to: PX4PSP\ RflySimAPIs \4.RflySimModel\0.ApiExps\4.ExtCtrlAPI-UDP30100\Readme.pdf -.



4. Advanced interface experiments

4.1 Obtaining Platform rfly_px4 uORB messages for external communication experiments (Personal Advanced Edition and above only)

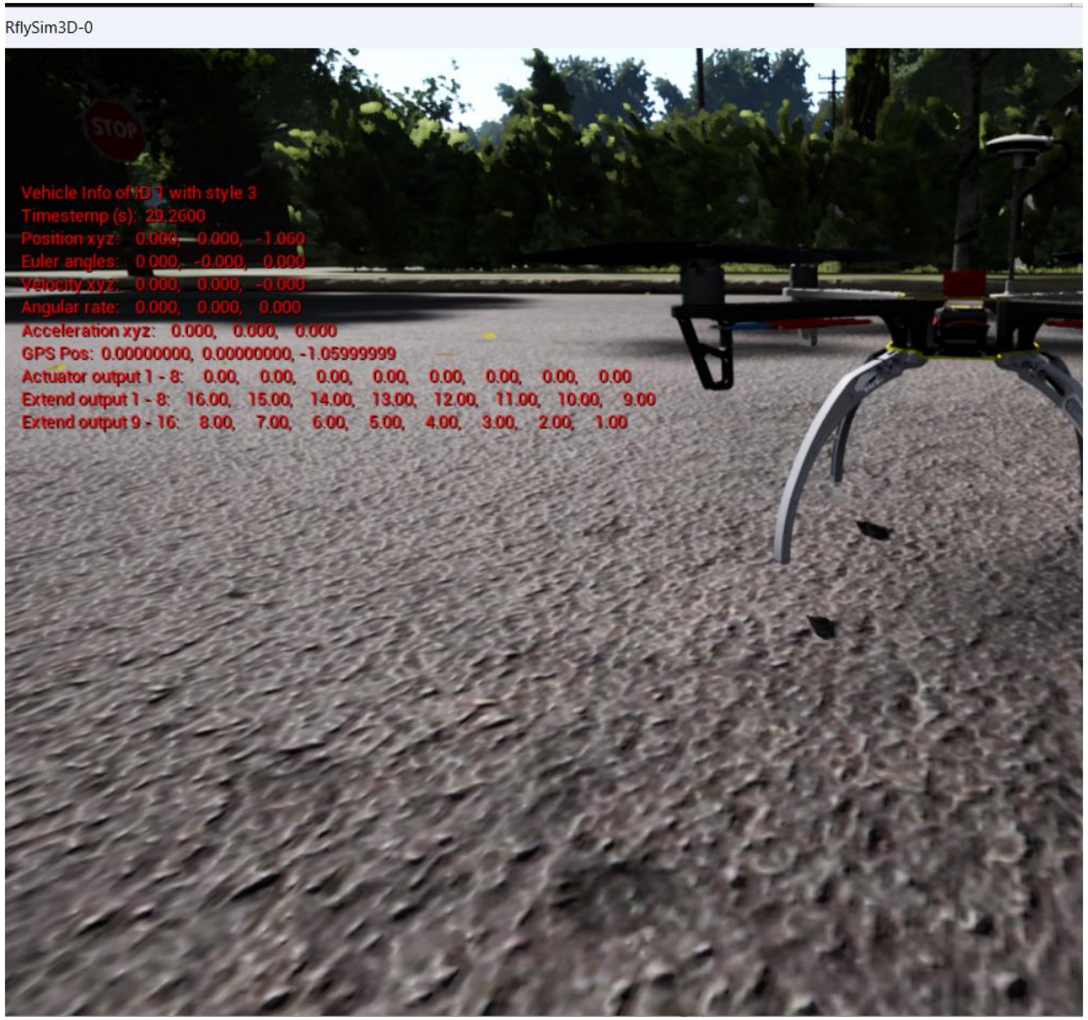
When subscribing to rfly_px4 uORB messages and using the platform maximum template for hardware-in-the-loop simulation, you can receive rfly_px4 messages by listening on UDP40100 series ports. The detailed experimental operation steps are shown in: PX4PSP\ RflySimAPIs \4. RflySimModel\0.ApiExps\5.ExtCtrlAPI-UDP40100\Readme.pdf -.



4. Advanced interface experiments

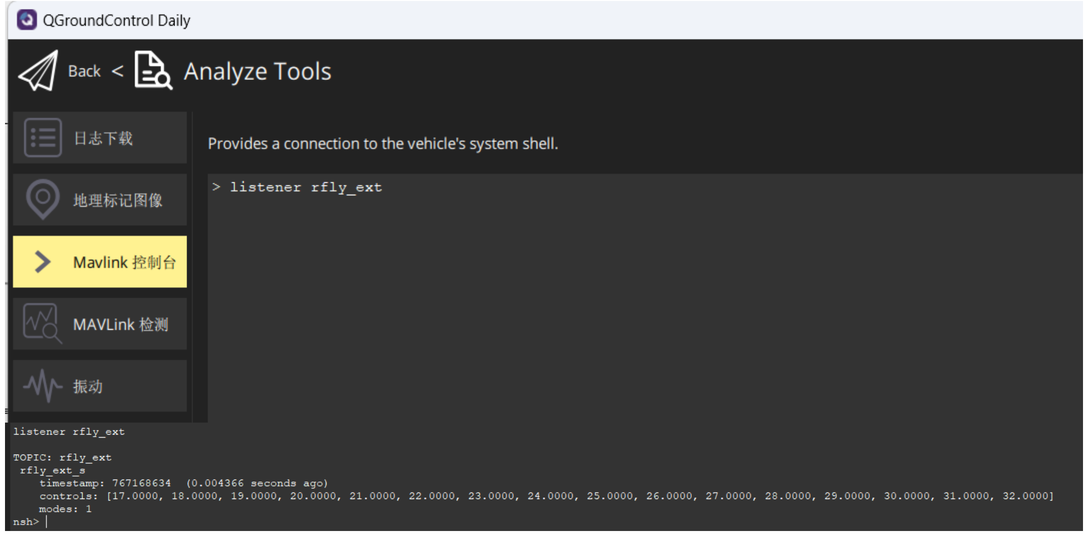
4.2 ExtToUE4 Interface Verification Experiment (Personal Advanced Edition and above only)

This routine allows users to customize the data sent to the ExtToUE4 interface of the largest model, which facilitates the development and debugging of the model. The detailed experimental operation steps are shown in: PX4PSP\ RflySimAPIs \4.RflySimModel\0.ApiExps\6.ExtToUE4\Readme.pdf -.



4.3 ExtToPX4 Interface Verification (Personal Advanced only)

This routine allows the user to customize the data sent to the ExtToPX4 interface of the largest model, which is the uORB message rfly_ext sent to the PX4 for transferring other sensors or necessary data to the flight controller to facilitate model development and debugging. The detailed experimental operation steps are shown in: PX4PSP\ RflySimAPIs \4.RflySimModel\0.ApiExps\7.ExtToPX4\Readme.pdf -.



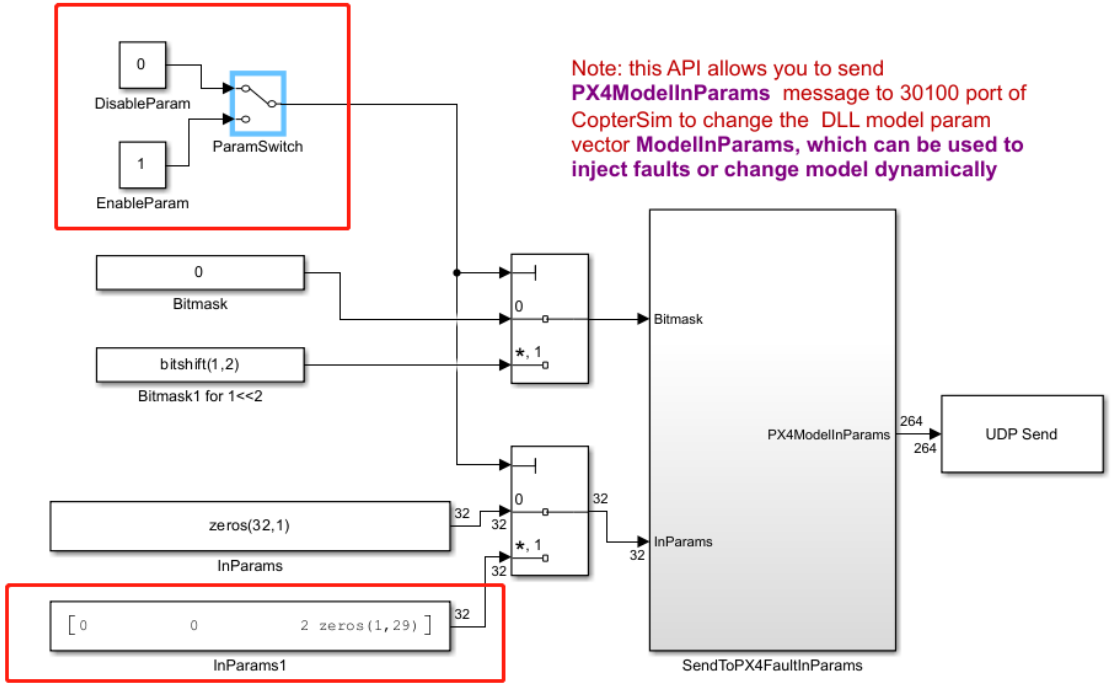
4. Advanced interface experiments

4.4 Maximum System Model OutCopterData Interface Verification Experiment (Personal Advanced Edition and above only)

This routine shows users how to use the outCopterData interface in the Max system model, which allows custom recording of 32-dimensional data during simulation. See PX4PSP\ RflySimAPIs \4.RflySimModel\0.ApiExps\9.OutCopterData\readme.pdf - For detailed experimental instructions.

4.5 Fault module dynamic modification parameter experiment (only personal advanced version or above)

Be familiar with the principle and process of dynamic parameter modification of FaultInParam, the largest system model of the platform. When using RflySim platform to simulate software and hardware in the loop, the maximum system model will receive FaultInParam data, the port number is 30100 series, and the third parameter in FaultInParam is related to the motor output. Therefore, the parameters can be dynamically modified to make the motor output all 0 to achieve the landing of the aircraft. For detailed experimental operation steps, please refer to: PX4PSP\ RflySimAPIs \4.RflySimModel\0.ApiExps\10.FaultParamsDynMod\Readme.pdf -.



4. Advanced interface experiments

4.6 InFloatsCollision Physics Engine Validation Experiment (Personal Premium only)

Be familiar with the use of the collision model port of inFloats Collision, the largest model of the platform. When using RflySim platform for hardware/software in-loop simulation, inFloatsCollision in the maximum system model reserves ports for the collision model, through which UDP data from RflySim3D can be obtained. Therefore, users can implement the function of physics engine through the inFloatsCollision interface. The detailed experimental instructions are available at: PX4PSP\ RflySimAPIs \4.RflySimModel\0.ApiExps\11.InFloatsCollision\Readme.pdf -.

4. Advanced Interface Experiment

4.7 InSILInts and InSILFloats Interface Experiment (Only personal Premium edition or above)

Familiar with inSILInts and inSILFloats interface, the biggest system model of the platform. The inSILInts and inSILFloats interfaces in the largest system model will receive external UDP struct data with port number 30100 series when using RflySim platform for hardware/software in-loop simulation. Therefore, users can use inSILInts and inSILFloats interfaces to implement additional functions, such as fault injection. See PX4PSP\ RflySimAPIs \4.RflySimModel\0.ApiExps\12.InSILInts&Floats\Readme.pdf -.

Outline

5. Advanced Case Experiment (multi-rotor simulation)

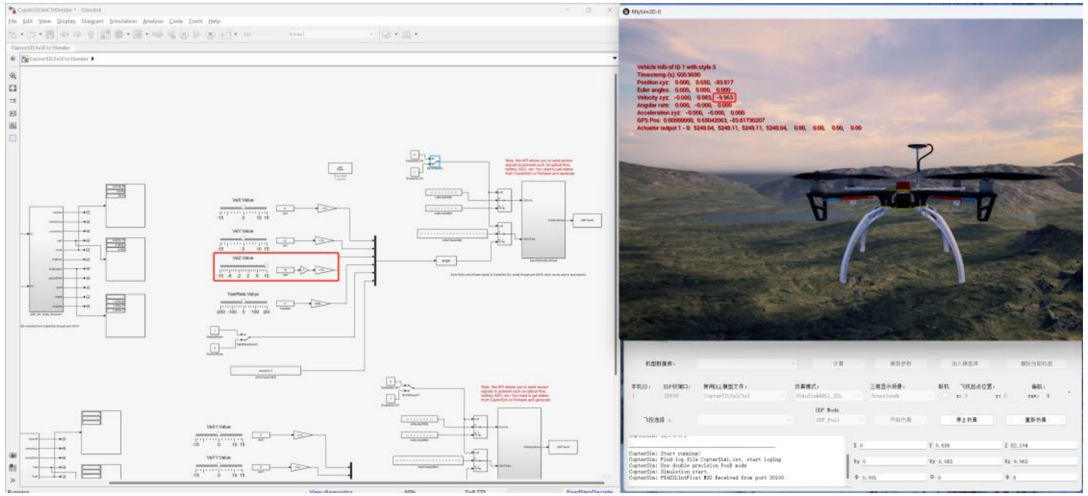
5.0 Overview of multi-rotor simulation experiments

This section describes the comprehensive model and software and hardware in the loop simulation experiments of several multi-rotor models, and the specific experiments are as follows:

Name of Experiment
Quadrotor integrated model simulation verification experiment (only advanced collection version above)
DLL Generation and SIL/HIL experiment of quadrotor model (Advanced collection version or above only)
Quadrotor model DLL generation and SIL/HIL experiments (including collision detection) (Advanced Collection and above only)
DLL Generation and SIL/HIL experiments for six-rotor models (Advanced Collection edition and above only)

5. Advanced Case study (multi-rotor simulation)

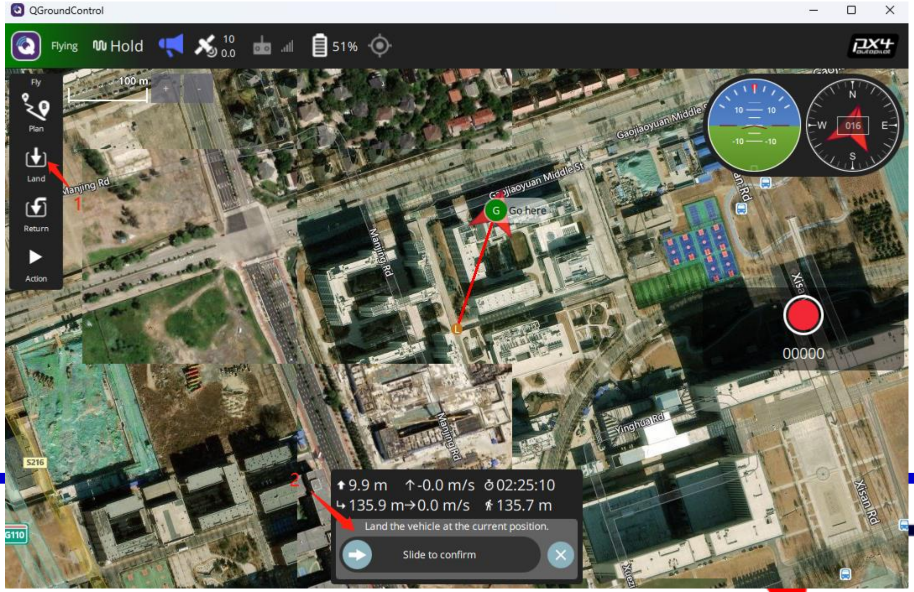
- 5.0 Quadrotor comprehensive model simulation verification Experiment (only Advanced collection version and above)
- Based on the DLL model of Simulink, the quadrotor controller is designed based on MATLAB/Simulink, and the controller an d the Dll model are put in the same slx file. According to the specific input and output interfaces, an aircraft overall simulation cl osed loop is formed, that is, the synthesis model. After the synthesis model is obtain ed, the top level control is realized by exte rnal control method. See * \PX4PSP\ Rfly SimAPIs \4.RflySimModel\2.AdvExps\e2_ MultiModelCtrl\1.CopterSimSILNoPX4\ Readme.pdf for details.



5. Advanced Case study (Multi-rotor simulation)

5.0 Quadrotor model DLL Generation and SIL/HIL Experiments (Advanced Collection version and above only)

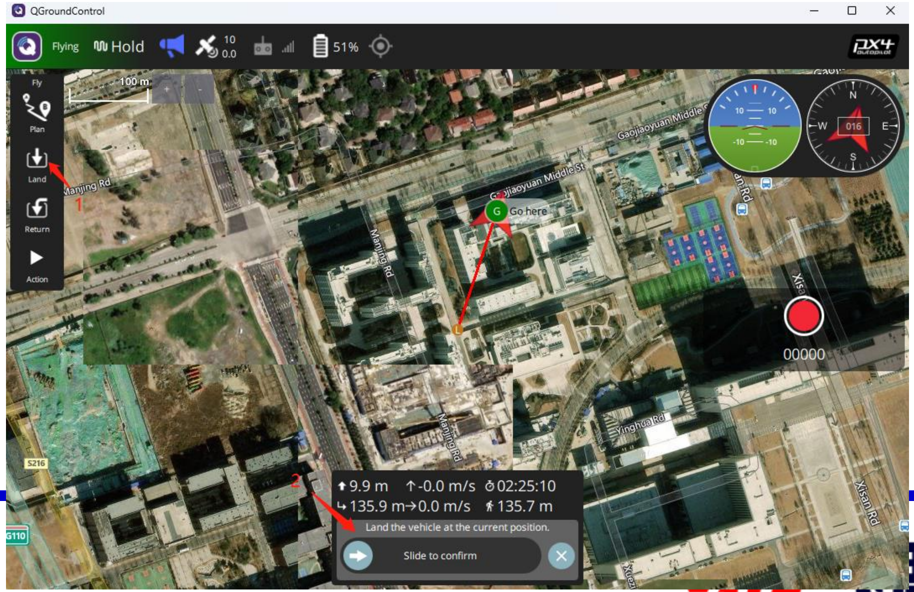
The Simulink file was compiled to generate the quadrotor DLL model file in Matlab. The generated quadrotor model is tested by software and hardware in the loop simulation, and the use of the quadrotor model of the platform is familiar with through this routine. The detailed experimental steps are as follows: * \PX4PSP\ RflySimAPIs \4.RflySimModel\2.AdvExps\e2_MultiModelCtrl\2.MultiModelCtrl\Readme.pdf.



5. Advanced Case Study (Multi-rotor simulation)

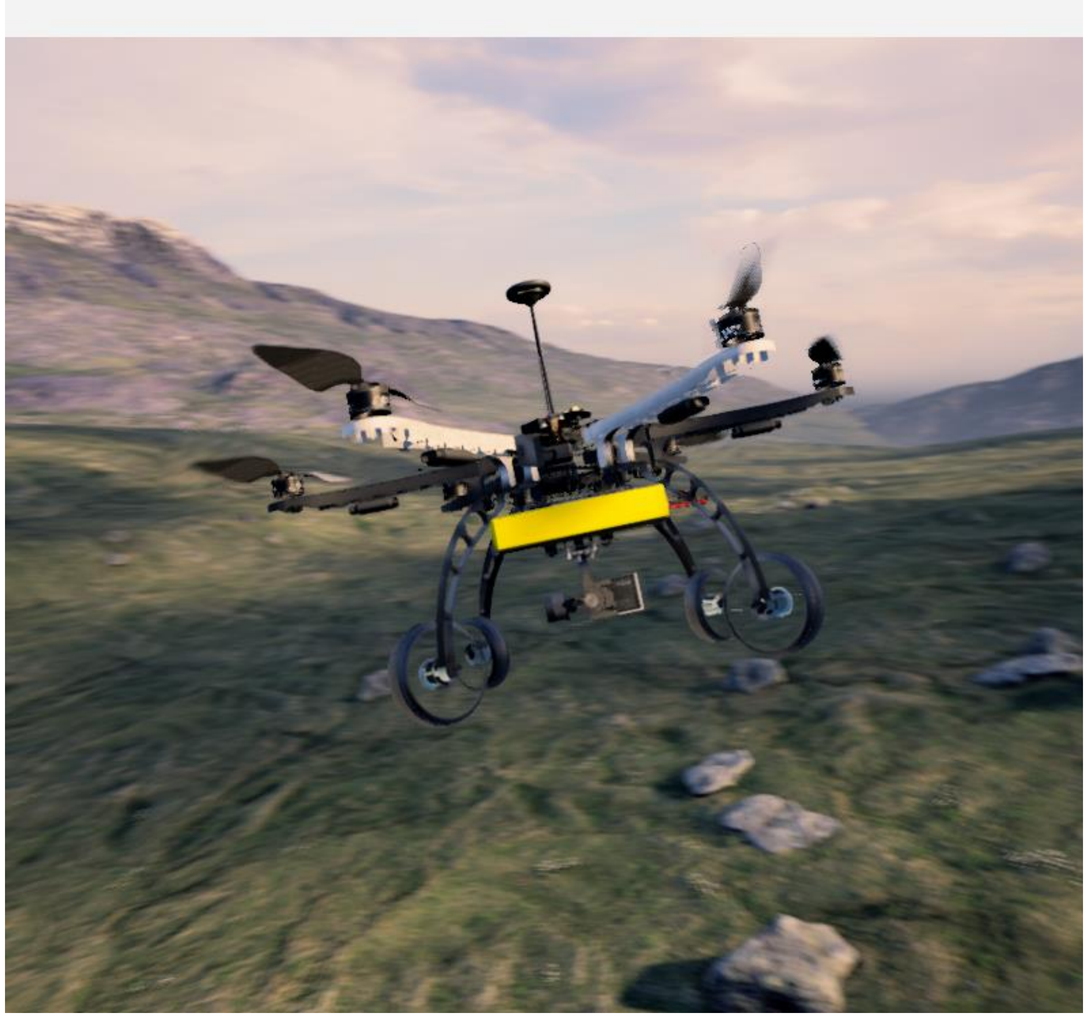
5.0 Quad rotor model DLL Generation and SIL/HIL experiments (including collision detection) (Advanced Collection only)

The Simulink file was compiled to generate the quadrotor DLL model file in Matlab. The generated quadrotor model is tested by software and hardware in the loop simulation, and the use of the quadrotor model of the platform is familiar with through this routine. The detailed experimental steps are as follows: * \PX4PSP\ RflySimAPIs \4.RflySimModel\2.AdvExps\e2_MultiModelCtrl\3.MultiModelCtrlColl\Readme.pdf.



5. Advanced Case Study (Multi-rotor simulation)

- **5.0 DLL Generation and SIL/HIL Experiment of six-rotor model (Advanced Collection version and above only)**
- **The Simulink files were compiled in Matlab to generate the DLL model file s of the six-rotor. The generated six-ro tor model was simulated and tested in the loop, and the use of the platform si x-rotor model was familiar with throu gh this routine. The detailed experime ntal steps are as follows: * \PX4PSP\RflySimAPIs \4.RflySimModel\2.Adv Exps\e2_MultiModelCtrl\4.HexModel Ctrl\Readme.pdf.**



5. Advanced Case Experiment (Platform fixed-wing simulation)

5.1 Overview of platform fixed-wing simulation experiment

This section takes the fixed wing as an example to explain several interface experiments and software and hardware in the loop simulation experiments of the fixed wing. The specific experiments are as follows:

Name of the Experiment
Fixed-wing aircraft model DLL Generation and SIL/HIL experiments (including collision detection)
Fixed-wing waypoint control experiment
Fixed-wing flight experiment with fixed pitch Angle
Fixed-wing speed/altitude/yaw interface validation experiment

5. Advanced Case Experiment (Platform fixed-wing simulation)

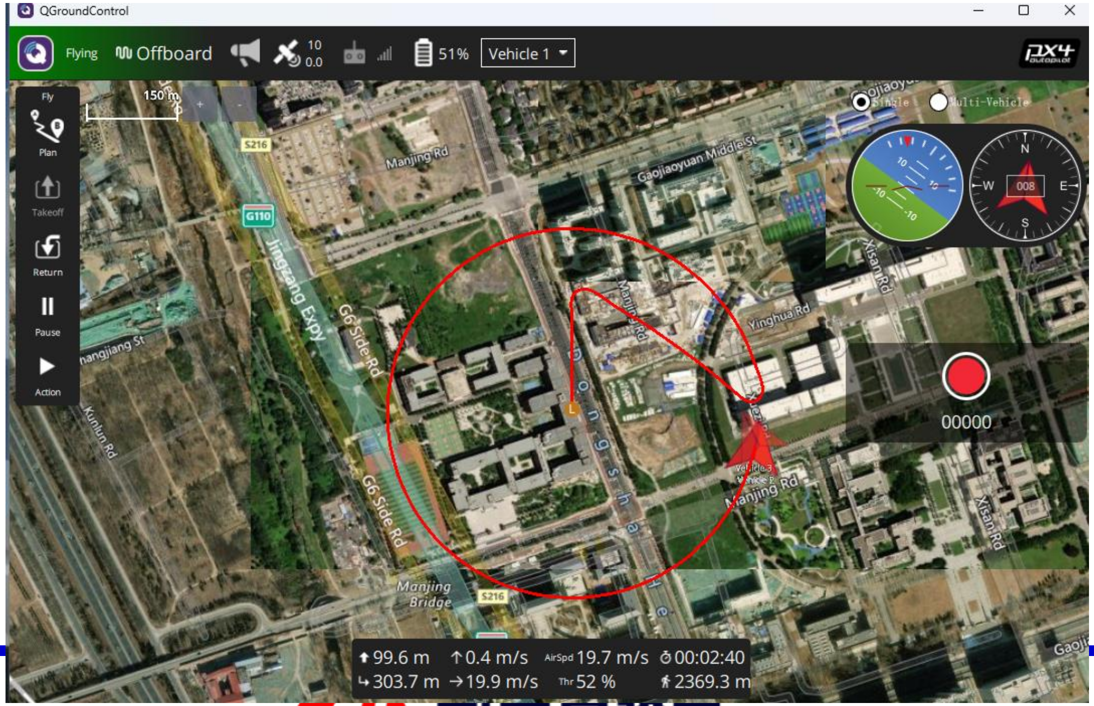
5.1 Fixed-wing aircraft model DLL generation and SIL/HIL experiments (including collision detection) (only Advanced collection version and above)

The Simulink file was compiled to generate the fixed-wing DLL model file in Matlab; The generated fixed wing model was tested by software and hardware in the loop simulation, and the use of the fixed wing model of the platform was familiar with through this routine. The detailed experimental operation steps are as follows: *\PX4PSP\ RflySimAPIs \4.RflySimModel\2.AdvExps\e3_FWingModelCtrl\1.FixWingModelCtrlColl\Readme.pdf.

5. Advanced Case Experiment (Platform fixed-wing simulation)

5.1 Fixed-wing Waypoint Control Experiment (Advanced Collection version or above only)

This routine uses the fixed wing control interface of the platform to make the fixed wing fly to the desired waypoint during the software/hardware in-loop simulation. See *\PX4PSP\ RflySimAPIs \4.RflySimModel\2.AdvExps\e3_FWingModelCtrl\2.FWPosCtrlAPI\Readme.pdf for detailed experimental operation steps.



5.1 Flight Experiment with fixed wing at fixed pitch Angle (Advanced Set version or above only)

This routine controls the pitch Angle of the fixed wing through the platform fixed wing control interface, so that the fixed wing flies forward with a fixed pitch Angle of 10°. See *\PX4PSP\ RflySimAPIs\4.RflySimModel\2.AdvExps\e3_FWingModelCtrl\3.FWAttCtrlAPI\Readme.pdf for detailed experimental operation steps.

5. Advanced Case Experiment (Platform fixed-wing simulation)

5.1 Fixed-wing speed/altitude/yaw interface verification Experiment (only Advanced Collection version or above)

The routine is in the form of Simulink/Python, and through the platform fixed wing interface, the fixed wing can fly according to the desired command in the process of software and hardware in the loop simulation. Python program for specific experimental operation steps, please see: *\PX4PSP\ RflySimAPIs \4.RflySimModel\2.AdvExps\e3_FWingModelCtrl\4.VelAltYawCtrlAPI_Py\Readme.pdf.

The detailed experimental steps of Simulink program are as follows: *\PX4PSP\ RflySimAPIs \4.RflySimModel\2.AdvExps\e3_FWingModelCtrl\5.VelAltYawCtrlAPI_Mat\Readme.pdf

5.2 Overview of other vehicle simulation

This section mainly introduces several other common vehicle modeling ideas, including: vertical takeoff aircraft, Ackermann chassis unmanned vehicle, differential unmanned vehicle, quadrotor tail type vertical takeoff UAV, and explains the interface experiments and software and hardware in the loop simulation experiments of several model development. The specific experiments are as follows:

Name of the Experiment
DLL generation and SIL/HIL experiment of high precision VTOL aircraft
Software/hardware in the loop simulation experiment of position of Ackermann chassis unmanned vehicle
Software/hardware in the loop simulation experiment of Ackman chassis unmanned vehicle speed
Software/hardware in the loop simulation experiment of differential unmanned vehicle speed
Software/hardware in the loop simulation experiment of quadrotor tail-mounted UAV

5. Advanced Case experiment (simulation of other vehicles)

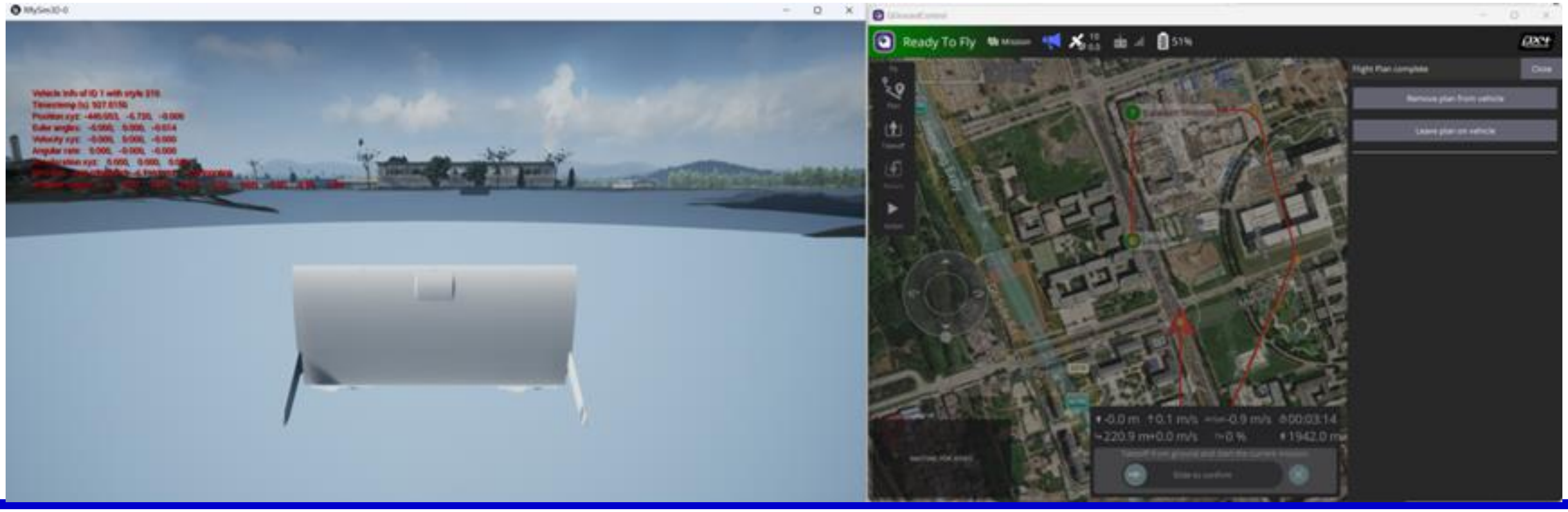
5.2 High-precision VTOL aircraft DLL Generation and SIL/HIL experiment (only Advanced collection version or above)

The Simulink files were compiled to generate the DLL model files of the VTOL aircraft in Matlab. The generated VTOL aircraft model is simulated by hardware and software in the loop, and the modeling and use of VTOL aircraft are familiar with this routine. The detailed experimental operation steps are as follows: * \PX4PSP\ RflySimAPIs \4.RflySimModel\2.AdvExps\e4_VTOLModelCtrl\1.VTOLModelCtrl\Readme.pdf.

5. Advanced Case study (Other vehicle simulation)

5.2 Simulation of software and hardware in the loop for quadrotor tail-mounted UAV (only advanced collection version or above)

This routine describes how to use the platform quadrotor tail mount vertical UAV for hardware and software in the loop simulation. Under the platform hardware and software in-the-loop simulation, the process of takeoff, mode switching, forward flight, return and landing of the quadrotor tail mount UAV is controlled by uploading track through QGC. Detailed experimental operation steps are as follows: * \PX4PSP\ RflySim APIs \4.RflySimModel\2.AdvExps\e4_VTOLModelCtrl\2.TailsitterModelCtrl\Readme.pdf.



5. Advanced Case study (Other vehicle simulation)

5.2 Ackermann chassis unmanned vehicle position software/hardware in the loop simulation experiment (only advanced collection version or above)

The routine is in the form of Simulink/Python, the software and hardware are in the loop simulation mode, and the position control of single or multiple unmanned vehicles on Ackermann chassis is realized through the platform position control interface. Python program for specific experimental operation steps, please see: *\PX4PSP\ RflySimAPIs \4.RflySimModel\ 2.AdvExps\e5_CarAckermanCtrl\2.CarAckermanPosCtrl_Py \Readme.pdf.

The detailed experimental steps of Simulink program are as follows: *\PX4PSP\ RflySimAPIs \4.RflySimModel\ 2.AdvExps\ e5_CarAckermanCtrl \3.CarAckermanPosCtrl_Mat \Readme.pdf.

5. Advanced Case Experiments (Other vehicle simulations)

5.2 Ackermann chassis unmanned vehicle speed software/hardware in the loop simulation experiment (only advanced collection version or above)

The routine is in the form of Simulink/Python, the software and hardware are in the loop simulation mode, and the speed control of single or multiple unmanned vehicles on Ackermann chassis is realized through the platform speed control interface. Python program for specific experimental operation steps, please see: *\PX4PSP\ RflySimAPIs \4.RflySimModel\ 2.AdvExps\e5_CarAckermanCtrl\4.CarAckermanVelCtrl_Py \Readme.pdf.

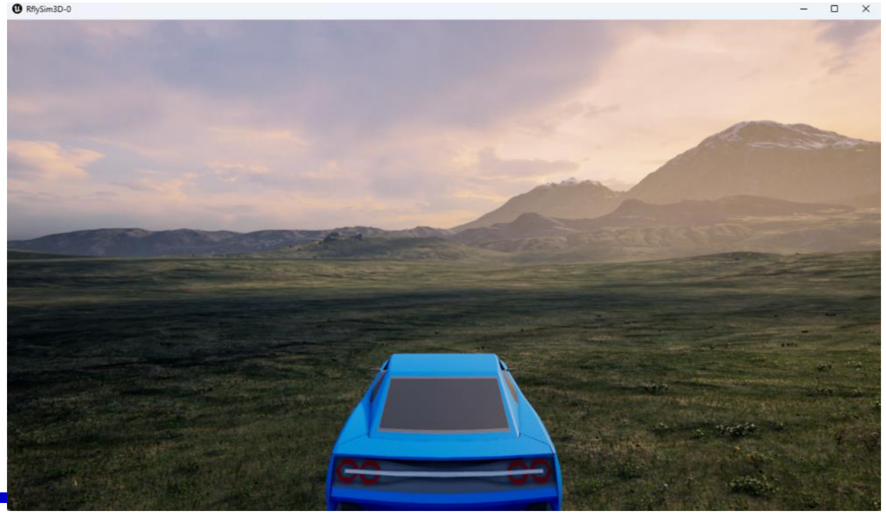
The detailed experimental steps of Simulink program are as follows: *\PX4PSP\ RflySimAPIs \4.RflySimModel\ 2.AdvExps\ e5_CarAckermanCtrl\5.CarAckermanVelCtrl_Mat \Readme.pdf.

5. Advanced Case Experiments (Other vehicle simulations)

5.2 Differential unmanned vehicle position software/hardware in the loop simulation experiment (only advanced collection version or above)

The routine is in the form of Simulink/Python, and the software and hardware are in loop simulation mode to realize the position control of single/multiple differential unmanned vehicles through the platform position control interface. Please see the Python program for detailed experimental operation steps: *\PX4PSP\ RflySimAPIs \4.RflySimModel\ 2.AdvExps\ e6_CarR1DiffCtrl\2.CarR1DiffPosCtrl_Py \Readme.pdf.

The detailed experimental steps of Simulink program are as follows: *\PX4PSP\ RflySimAPIs \4.RflySimModel\ 2.AdvExps\ e6_CarR1DiffCtrl\3.CarR1DiffPosCtrl_Mat \Readme.pdf.

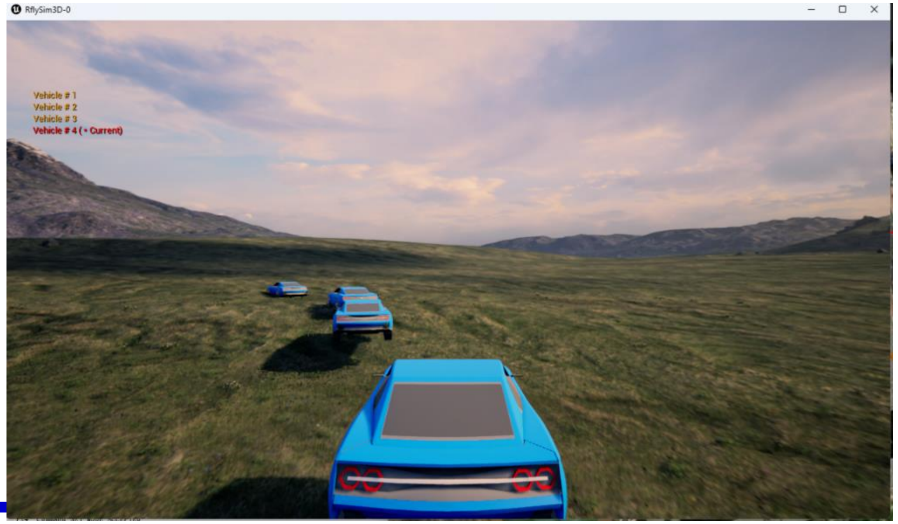


5. Advanced Case study (Other vehicle simulation)

5.2 Differential unmanned vehicle speed software/hardware in the loop simulation experiment (only advanced collection version or above)

The routine is in the form of Simulink/Python, the software and hardware are in the loop simulation mode, and the speed control of single/multiple differential unmanned vehicles is realized through the platform speed control interface. Python program for specific experimental steps, please see: *\PX4PSP\ RflySimAPIs \4.RflySimModel\ 2.AdvExps\ e6_CarR1DiffCtrl\4.CarR1DiffVelCtrl_Py \Readme.pdf.

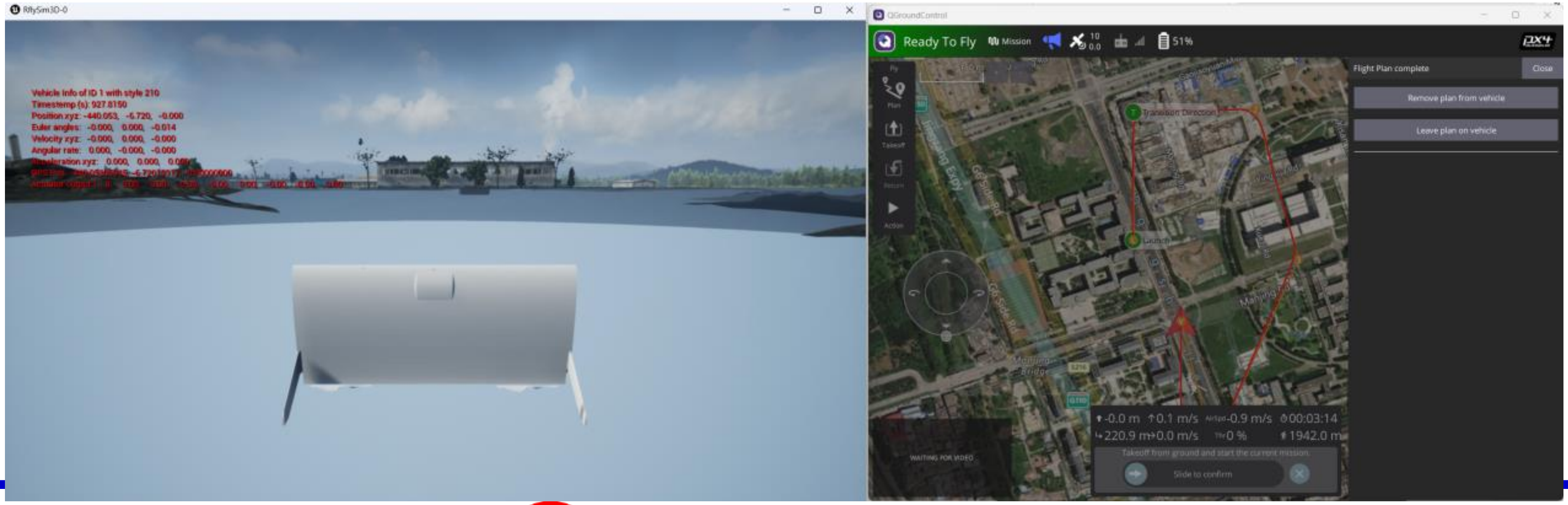
The detailed experimental steps of Simulink program are as follows: *\PX4PSP\ RflySimAPIs \4.RflySimModel\ 2.AdvExps\ e6_CarR1DiffCtrl\5.CarR1DiffVelCtrl_Mat \Readme.pdf.



5. Advanced Case study (Other vehicle simulation)

5.2 Simulation experiment of software and hardware in the loop for quadrotor tail-mounted UAV (only advanced collection version or above)

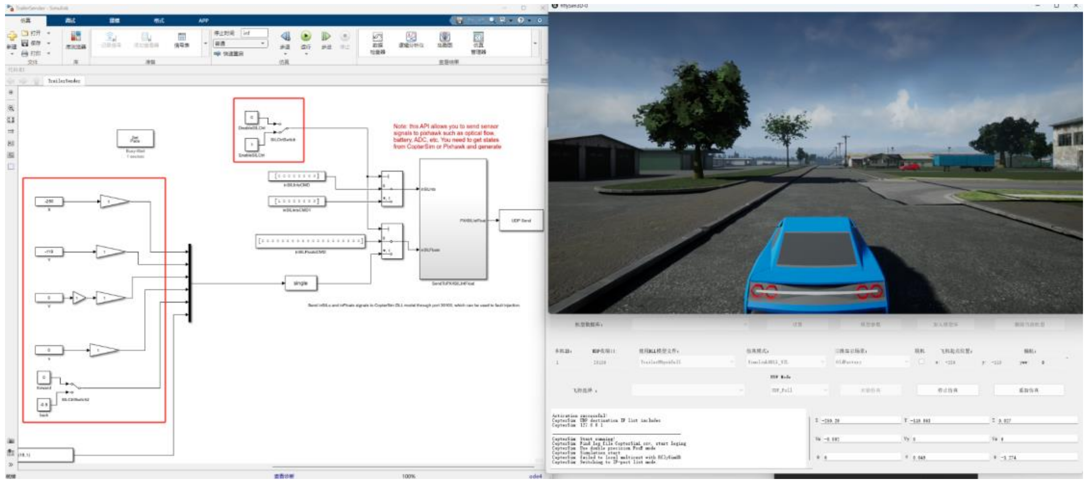
This routine describes how to use the platform quadrotor tail mount vertical UAV for hardware and software in the loop simulation. See * \PX4PSP\ RflySimAPIs \4.RflySimModel\2.AdvExps\e7_TailsitterModelCtrl\1.TrailerModelCtrl\Readme.pdf for detailed experimental operation steps.



5. Advanced Case study (Other vehicle simulation)

5.2 Simulation and Verification of unmanned vehicle comprehensive model (only advanced collection version or above)

Based on the Dll model of Simulink, an unmanned vehicle controller is designed base d on MATLAB/Simulink, and the controller and the Dll model are placed in the same slx f ile. According to the specific input and output interfaces, an unmanned vehicle overall simul ation closed loop is formed, that is, the synthe sis model. After the synthesis model is obtain ed, the top level control is realized by externa l control method. See: * \PX4PSP\ RflySimA PIs \4.RflySimModel\2.AdvExps\e7_Tailsitter ModelCtrl\2.TrailerNoPX4\Readme.pdf.



Outline

1 Experiment platform configuration

2. Introduction of key interfaces

3. Basic Experimental Case (free version)

4. Advanced Interface Experiment (Personal Edition)

5. Advanced Case Experiment (Collective Edition)

6. Extended Interfaces and Cases (Full version)

7. Summary

6 Expand the case

Model inCtrlExt series interface experiments	**The RflySim platform provides a wealth of model input and output interfaces to realize complex functions. This routine introduces the use of the inCtrlExt series input interface of models.**	**0.ApiExps\13.inCtrlExt\Readme.pdf**
inFromUE input interface experiment	In the model routines of the platform, in addition to several necessary input and output interfaces that serve the basic functions of the platform, there are also some input interfaces that can send some more detailed vehicle simulation information. inFromUE is the 32-dimensional double data sent by UE to the model, which is used to deal with the interaction between the scene and the model.	0.ApiExps\15.inFromUE\Readme.pdf

Outline

1 Experiment platform configuration

2. Introduction of key interfaces

3. Basic Experimental Case (free version)

4. Advanced Interface Experiment (Personal Edition)

5. Advanced Case Experiment (Collective Edition)

6. Extended Case Studies (Full version)

7. Summary

7 Summary

- **RflySim platform adopts the model-based design concept. It takes the model as the core, and uses a lot of automatic code generation technology, which greatly improves the development efficiency of unmanned vehicle system. Therefore, this lecture describes the interface experiments related to RflySim platform model development, model development experiments and SIL/HIL experiments of various vehicles in a step-by-step way.**

If in doubt, please visit https://doc.rflysim.com/ for more information.

RflySim for more tutorials

Scan code consultation and communication

Fesi RflySim technology exchange group

Thank you!