



Development and Practice of Intelligent Unmanned Cluster Systems: A Full-Stack Development Case Study Based on the RflySim Platform

Lecture 5: Pose Control and Filter Estimation



Outline

- 1. Experimental Platform Configuration**
 - 2. Introduction to Key Interfaces**
 - 3. Basic Experimental Cases (Free Version)**
 - 4. Advanced Interface Experiments (Free Version)**
 - 5. Advanced Case Experiments (Free Version)**
 - 6. Extended Cases (Full Version)**
 - 7. Summary**
-



1.1 Reference Materials

- 全权, 戴训华, 王帅著. 多旋翼飞行器设计与控制实践[M]. 北京: 电子工业出版社, 2020
Quan Quan, Dai Xunhua, Wang Shuai, "Design and Control of Multirotor Aircraft: Practice"
[M], Published by Electronic Industry Press, 2020.



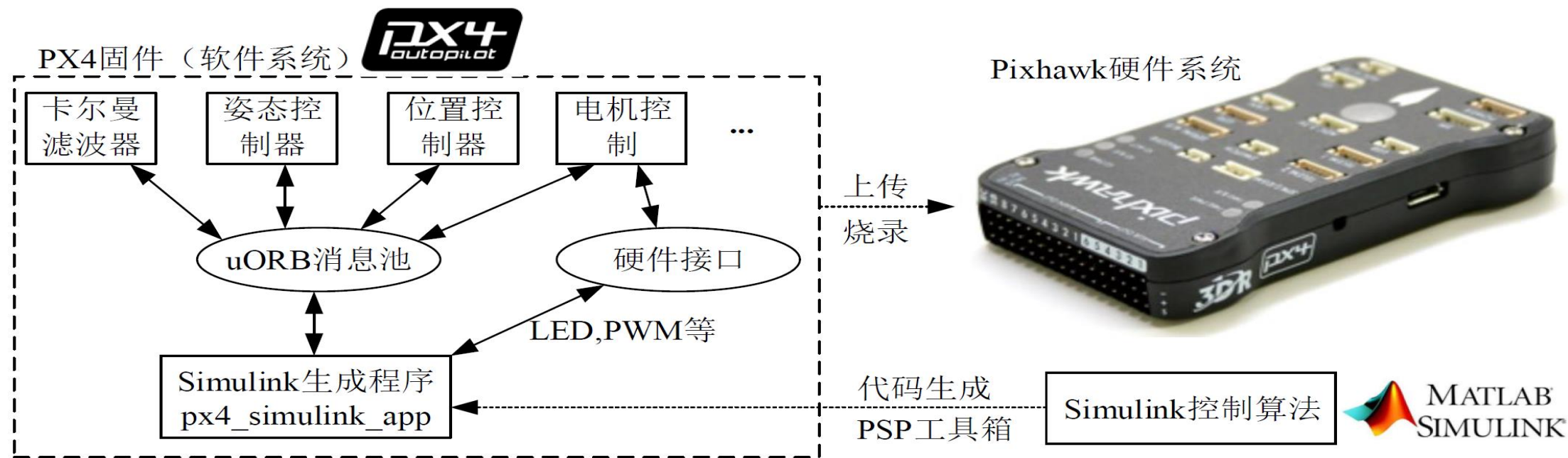
Note:

- The book 《多旋翼飞行器设计与控制实践》 on the left was released in 2020 and is a practical course focused on the development of flight control algorithms. It includes some theoretical knowledge and a series of experiments, enabling readers to quickly program their algorithms in Simulink and download them to Pixhawk for flight experiments.
- The book 《多旋翼飞行器设计与控制》 on the right was released in 2017 and primarily focuses on multicopter control theory.



1.2 Pixhawk/PX4/Simulink Code Generation Platform Architecture

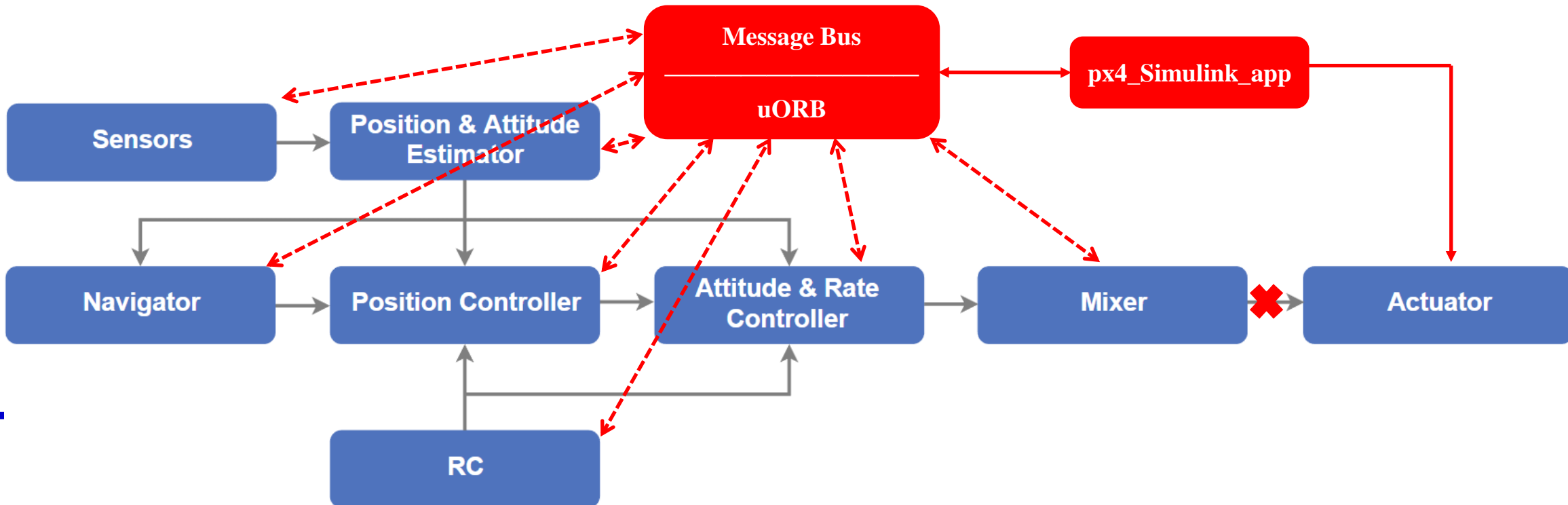
Pixhawk is the hardware (equivalent to a computer mainframe), PX4 is the flight control software (equivalent to the Windows operating system), Simulink generates controller code which is then compiled into firmware (equivalent to a system ISO image), uploaded to the Pixhawk hardware (similar to reinstalling the system), and the Simulink controller runs as a new thread (similar to a third-party app on a computer), operating independently of the official PX4 controller (similar to pre-installed software on the system), and running in parallel.





1.3 Why Disable PX4 Output

- PX4 adopts the uORB publish-subscribe message mechanism, allowing any app to retrieve and publish data from the uORB message pool.
- After Simulink code is generated and uploaded to Pixhawk, it creates an app named `px4_Simulink_app`. This app communicates with other apps through the uORB message pool mechanism.
- `Simulink_App` cannot access the motors simultaneously with the PX4 controller to avoid conflicts. Therefore, it is necessary to disable the official PX4 output.





1.4 Simulink Automatic Code Generation Configuration

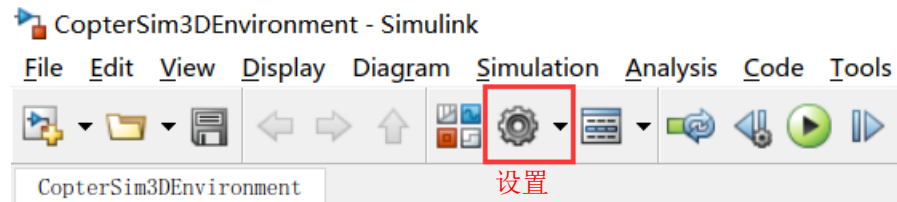
Creating a new blank slx routine file

1. Go to the Simulink settings page (For MATLAB 2019b and above, click the "Settings" button on the MODELING tab).

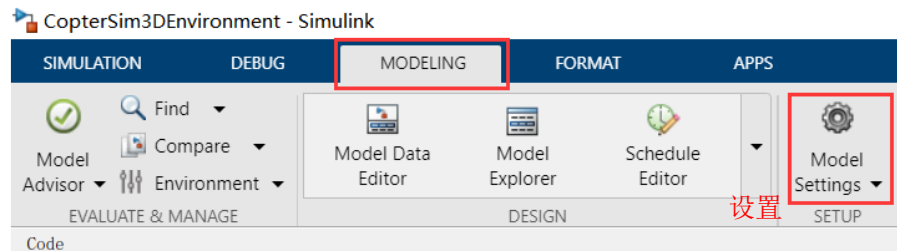
2. Select the Hardware board setting as Pixhawk PX4, which will automatically complete all the code generation settings required for this platform.

3. Custom task priorities can be set.

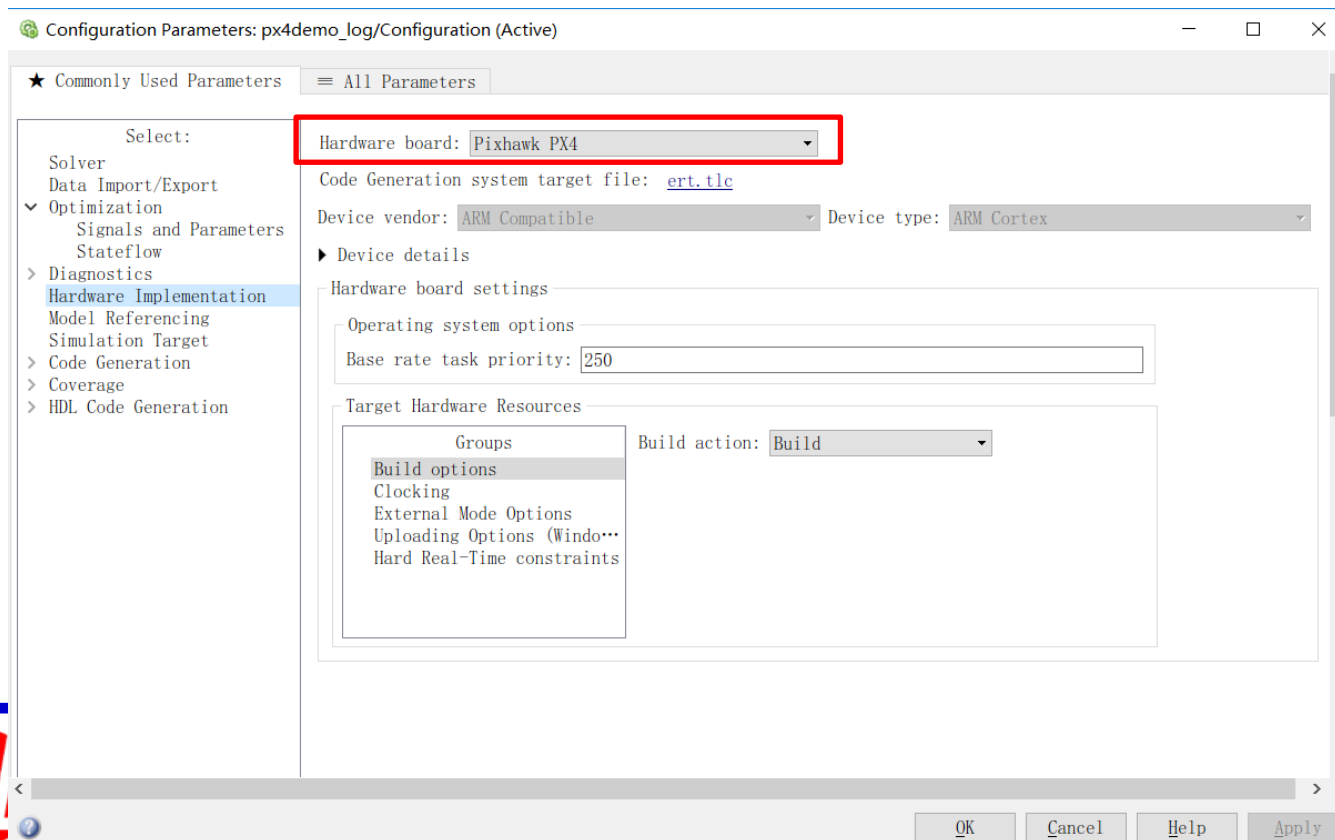
4. Configure compilation options.



(a) Simulink “设置”按钮 (MATLAB 2017b~2019a)



(b) Simulink “设置”按钮 (MATLAB 2019b及更高版本)





1.5 PX4PSP Toolbox Location

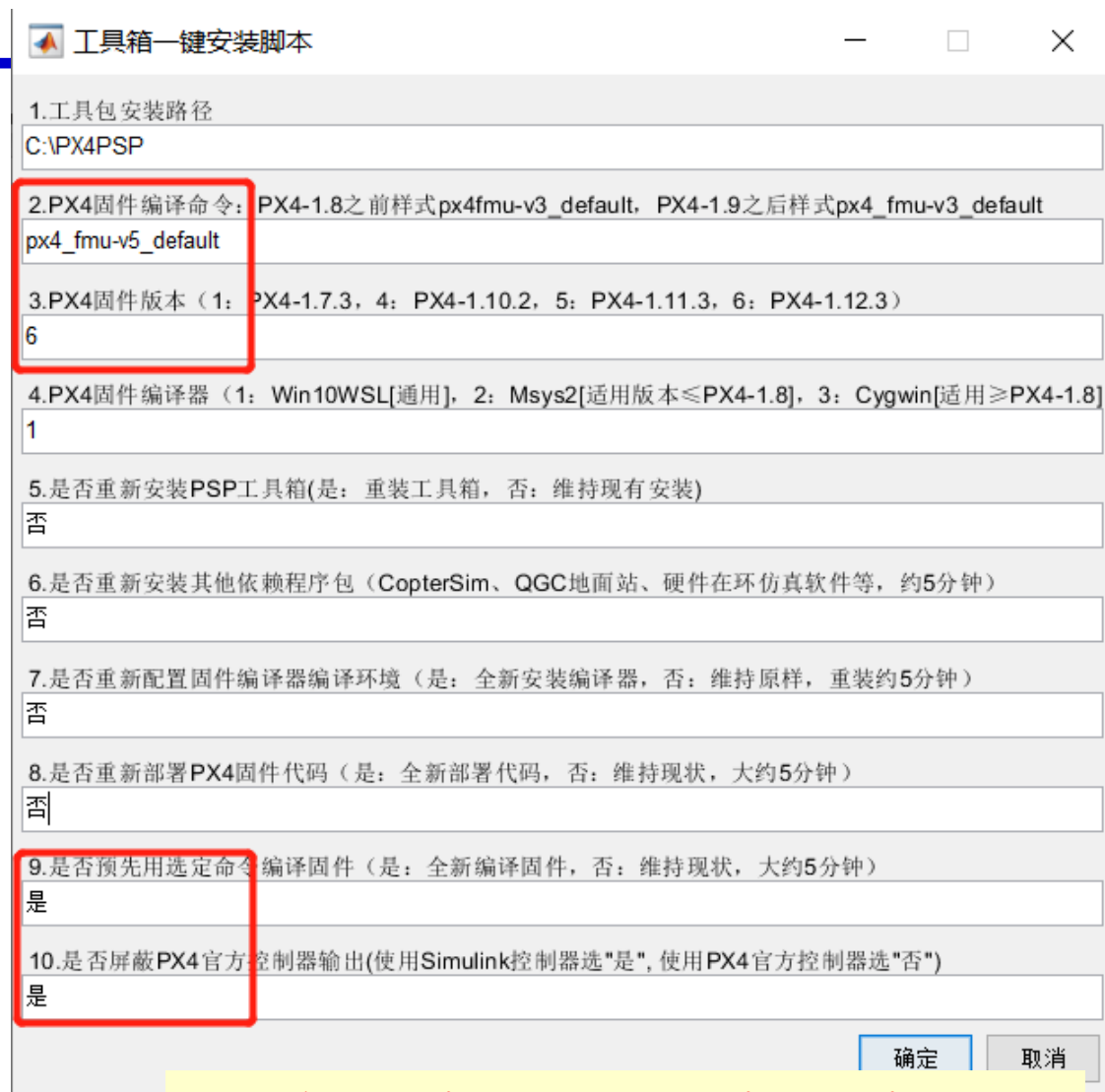
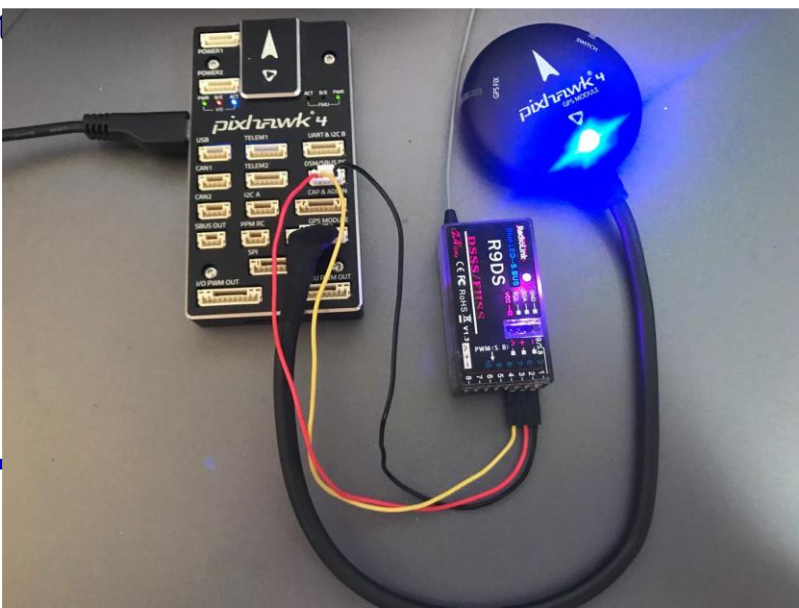
- Open any Simulink file, click the "Library Browser" button, and you will find the "Pixhawk Target Blocks" module library in it.
- The modules in this library can be considered as higher-level interfaces encapsulating the subscription or publication of data from the PX4 uORB pool. They include functions related to sensors, remote controllers, motors, serial ports, and more.
- Note: These modules do not contain internal models; data can only be obtained when the code is generated and compiled into PX4 firmware. If run directly in Simulink, the received data will be all zeros.
- For detailed usage of each module, refer to the **Basic Edition course** or the third tutorial.





1.6 Experimental Configuration - Recommended Installation Options

- Recommended Installation Options:
- It is recommended to use the latest Pixhawk 4 autopilot (refer to the image below), with the compilation command named px4_fmu-v5_default.
- Re-run the "OnekeyScript.p" script included in the installation package.
- Utilize the latest PX4 firmware version "6" - PX4-1.12.3.
- Other configurations are as shown in the image on the right



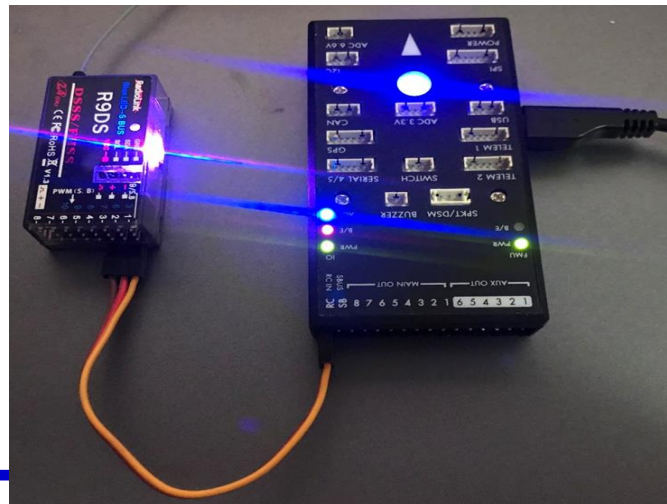
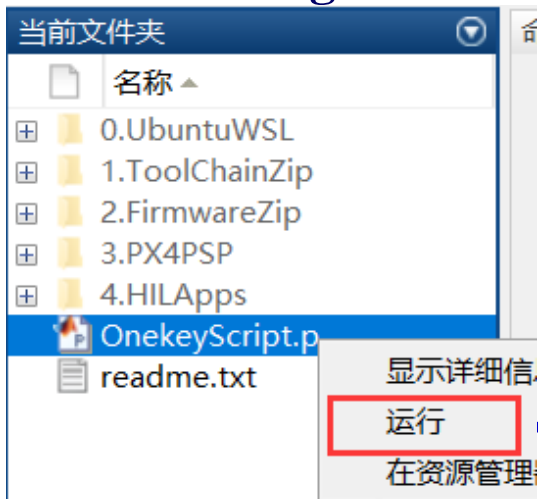
Note: Alternatively, you can quickly switch by entering the command "PX4CMD px4_fmu-v5_default" in MATLAB.



1.7 Experimental Configuration - Installation Options for Old Version of the Textbook

- If using the recommended Pixhawk 1 autopilot from the textbook (refer to the image below), the compilation command is `px4_fmu-v3_default`.
- Re-run the "OnekeyScript.p" script included in the installation package.
- Utilize the latest PX4 firmware version "6" - PX4-1.12.3, and compiler "1" - Win10WSL.
- Other configurations are as shown in the image on the right.

Note: If using the Pixhawk 1 autopilot recommended in the tutorial, it is advised to configure according to this page. Alternatively, you can follow the configuration method in the book, choosing PX4 1.7.3 version + Msys2 compiler (not recommended).



1.工具包安装路径	C:\PX4PSP
2.PX4固件编译命令: PX4-1.8之前样式px4fmu-v3_default, PX4-1.9之后样式px4_fmu-v3_default	px4_fmu-v3_default
3.PX4固件版本 (1: PX4-1.7.3, 4: PX4-1.10.2, 5: PX4-1.11.3, 6: PX4-1.12.3)	6
4.PX4固件编译器 (1: Win10WSL[通用], 2: Msys2[适用版本≤PX4-1.8], 3: Cygwin[适用≥PX4-1.8])	1
5.是否重新安装PSP工具箱(是: 重装工具箱, 否: 维持现有安装)	否
6.是否重新安装其他依赖程序包 (CopterSim、QGC地面站、硬件在环仿真软件等, 约5分钟)	否
7.是否重新配置固件编译器编译环境 (是: 全新安装编译器, 否: 维持原样, 重装约5分钟)	否
8.是否重新部署PX4固件代码 (是: 全新部署代码, 否: 维持现状, 大约5分钟)	否
9.是否预先选定命令编译固件 (是: 全新编译固件, 否: 维持现状, 大约5分钟)	是
10.是否屏蔽PX4官方控制器输出(使用Simulink控制器选"是", 使用PX4官方控制器选"否")	是

Note: Alternatively, you can quickly switch by entering the command "PX4CMD px4_fmu-v3_default" in MATLAB.



1.8 Experimental Configuration - Hardware Setup and Verification

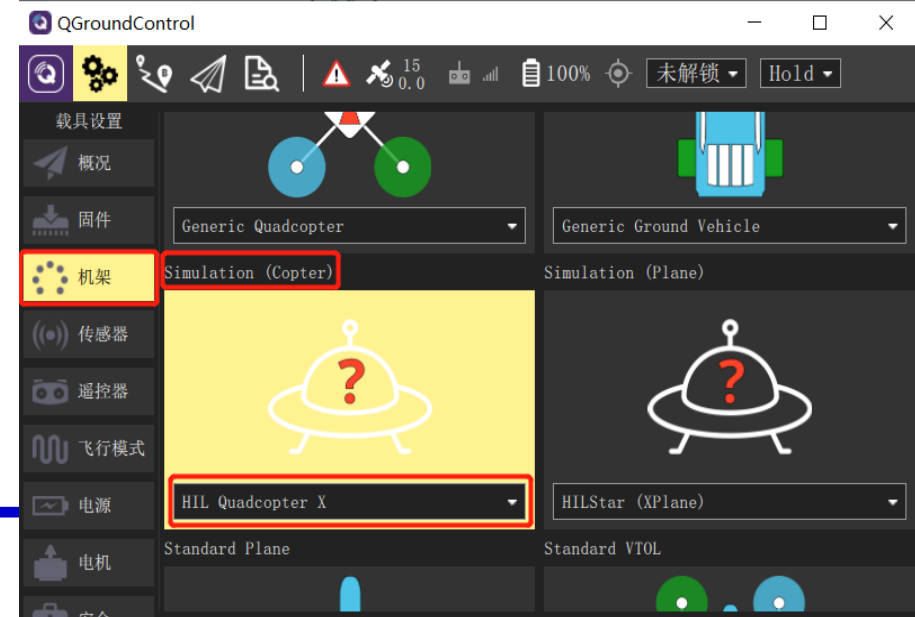
1、Please refer to the tutorial link for hardware setup:

https://doc.rflysim.com/hardware/3RC/AT9s_Pro.html 2、Confirm the following configurations:

- Ensure Pixhawk has been flashed with the latest official firmware version 1.12 in QGroundControl (QGC), and the LED is blinking normally.
- Correctly connect Pixhawk to the receiver, connect the remote controller to the receiver, open QGC ground station, and ensure that the movement signals of the remote controller joysticks can be observed.
- Properly configure the remote controller and calibrate it in QGC, ensuring that the lowest and highest positions meet the definitions in the tutorial link.
- Confirm that Pixhawk flight controller has been set to select HIL Quadcopter X frame in QGC.
- Ensure flight modes in QGC are configured according to the tutorial.



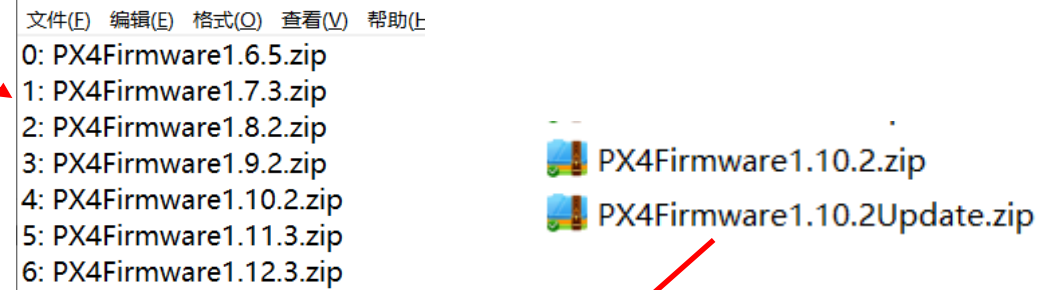
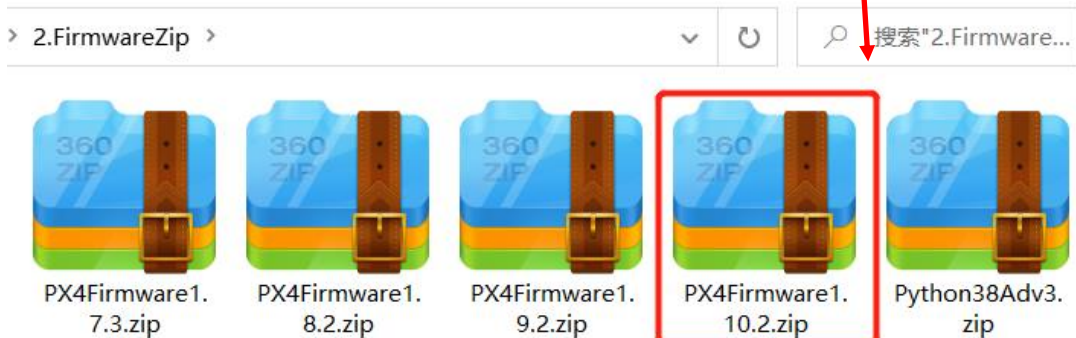
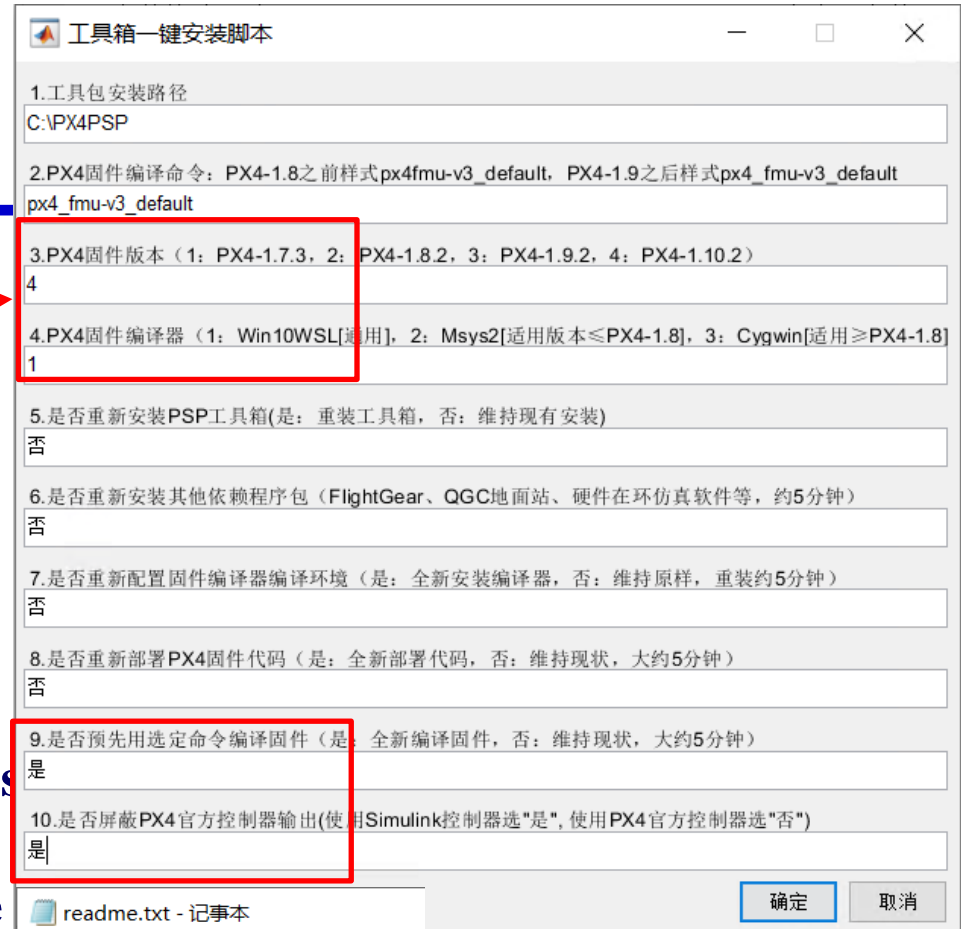
注：箭头表示PWM值增大的拨动方向





1.9 Choose a different firmware version.

- If you need to use your own PX4 firmware code, rename your code folder to "Firmware", compress it into a Firmware.zip file, and then rename it according to the rules in 2.FirmwareZip\readme.txt. Choose the desired firmware version.
- For example, if you have developed your code based on PX4 1.10, name it "PX4Firmware1.10.2.zip" and replace the corresponding file under the "2.FirmwareZip" folder. In the firmware version selection in the installation options on the right, choose "4".
- Whether to shield PX4 output project selection "Yes", the script will automatically complete all the necessary firmware modifications to adapt to this platform.



注: 也可以将PX4Firmware1.10.2.zip官方固件的增量文件打包并命名为“PX4Firmware1.10.2Update.zip”的格式放在2.FirmwareZip目录, 安装时会自动拷贝到固件文件夹。





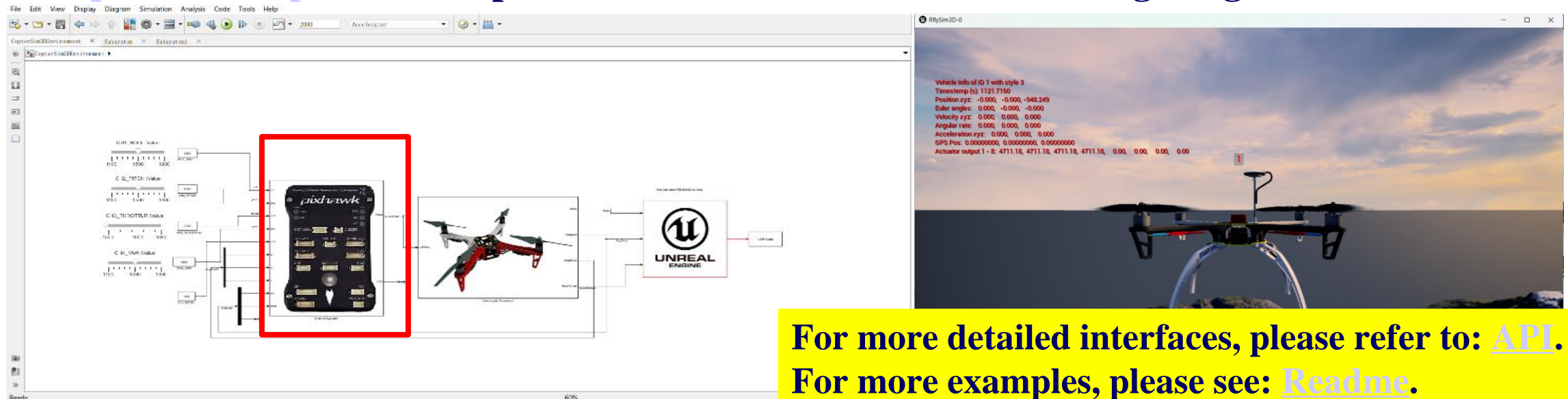
Outline

1. Experimental Platform Configuration
 2. Introduction to Key Interfaces
 3. Basic Experimental Cases (Free Version)
 4. Advanced Interface Experiments (Free Version)
 5. Advanced Case Experiments (Free Version)
 6. Extended Cases (Full Version)
 7. Summary
-



2.1 Software in-the-loop simulation experiment

As shown in the right figure, control the multirotor to the specified pitch and roll angles, maintaining and controlling the attitude. The controller responds to the control inputs from the remote control, and you can simulate the remote control input by dragging the Slider module on the left. For specific experimental operations, refer to the file [0.ApiExps\1.SoftwareSimulationExps\Readme.pdf](#). The experimental results are shown in the right figure.

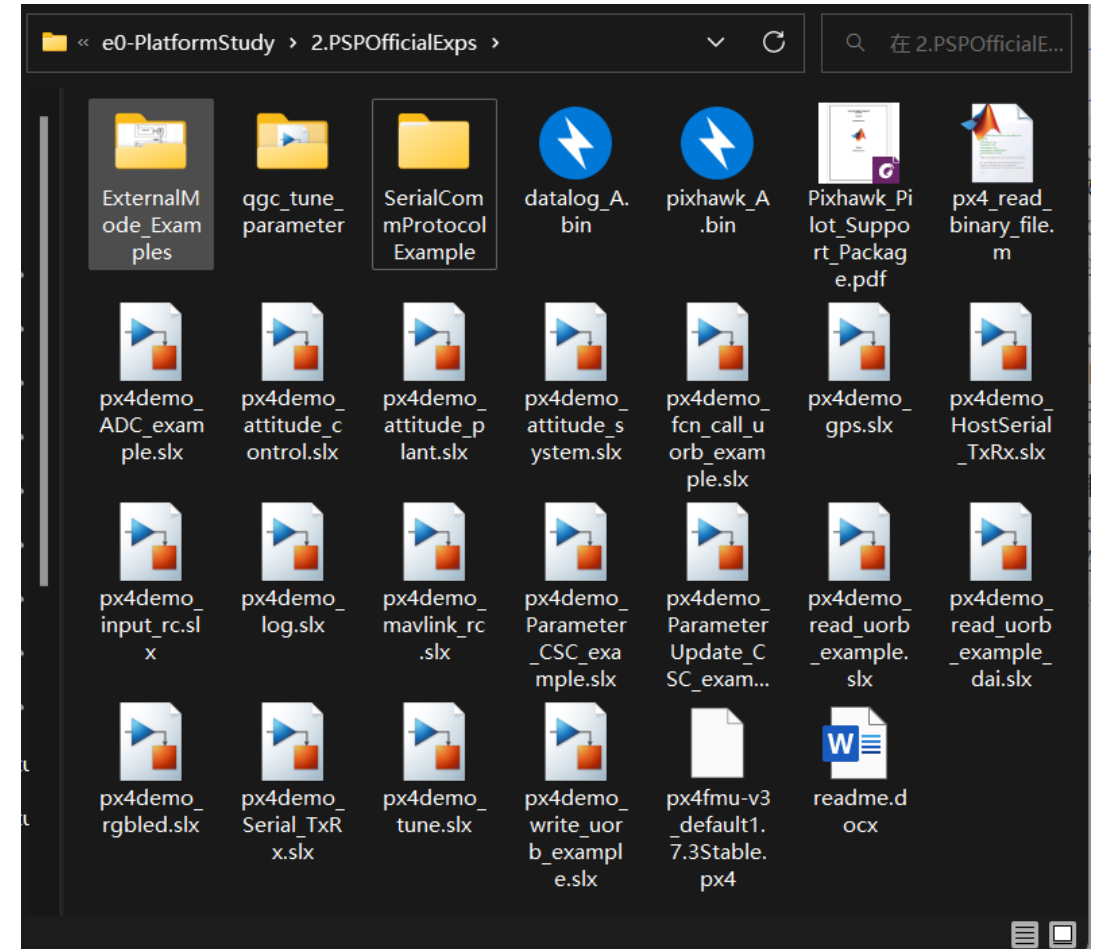


For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



2.2 Official Examples Experiment for Automatic Code Generation Toolbox

MATLAB provides relevant examples and documentation for the Pixhawk Pilot Support Package (PSP) in the document titled “Pixhawk_Pilot_Support_Package.pdf”. Users can learn to model, simulate, and validate flight control models in Simulink, and deploy them to PX4 hardware integrated in to the control system using the automatic code generation feature. For specific experimental operations, please refer to the file: [0.ApiExps\2.PSPOfficialExps\Readme.pdf](#).



For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



2.3 Attitude Control Design Experiment

This experiment designs a controller with inputs from remote control channels Ch1~Ch5 signals, angular velocity feedback AngRateB, and multirotor Euler angles (in radians). The entire process, from software in-the-loop simulation to automatic code generation, hardware in-the-loop simulation, and actual flight, is implemented by building a model in Simulink.

The specific operations of the experiment are detailed in the file: [0.ApiExps\3.DesignExps\Readme.pdf](#) .



软件在环RflySim3D显示

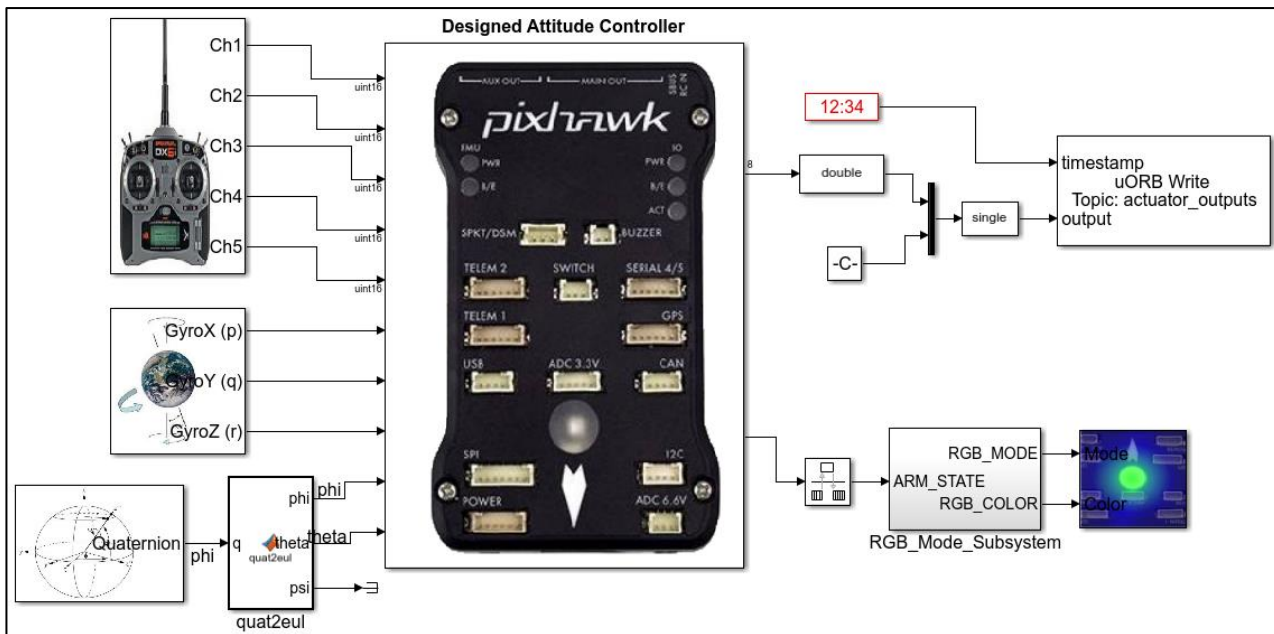


For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



2.Introduction to Key Interfaces

2.3 Attitude Control Design Experiment



硬件在环Simulink模型

```

Diagnostic Viewer
Exp4_AttitudeSystemCodeGen
f.c.obj
[7/11] Building C object
src/modules/px4_simulink_app/CMakeFiles/modules__px4_simulink_app.dir/nuttxin
initialize.c.obj
[8/11] Linking C static library
src/modules/px4_simulink_app/libmodules__px4_simulink_app.a
[9/11] Linking CXX executable nuttx_px4fmu-v3_default.elf
[10/11] Generating px4fmu-v2.bin
[11/11] Creating /mnt/c/PX4PSP/Firmware/build/px4fmu-v3_default/px4fmu-
v3_default.px4
"### Finished calling CMAKE build process ###"
"### Done invoking postbuild tool."
"### Successfully generated all binary outputs."

C:\Users\dream\Desktop\e0\3.DesignExps\Exp4_AttitudeSystemCodeGen_ert_rtw>exi
t /B 0
### Successful completion of build procedure for model:
Exp4_AttitudeSystemCodeGen
### Creating HTML report file Exp4_AttitudeSystemCodeGen_codegen_rpt.html

Build process completed successfully

```

```

C:\WINDOWS\SYSTEM32\cmd.exe
Loaded firmware for 9.0, size: 879196 bytes, waiting for the bootloader...
If the board does not respond within 1-2 seconds, unplug and re-plug the USB connector.
PX4_SIMULINK = None
attempting reboot on COM3...
if the board does not respond, unplug and re-plug the USB connector.
attempting reboot on COM3...
if the board does not respond, unplug and re-plug the USB connector.
attempting reboot on COM3...
if the board does not respond, unplug and re-plug the USB connector.
Found board 9,0 bootloader rev 4 on COM3
50583400 00ac2600 00100000 00ffffff ffffffff ffffffff ffffffff ffffffff 66ed47ff ff73cc15 c8ad940c dbc59f39 d6c20e06 f95
3d3ef f3073019 d035ab0d 3f60334e 10dda9f8 cdb0cbbd 42cdc6b6 3ba305f7 81532581 84ee3da6 23bc6340 8321be68 eed356c9 1e3b8f
5c 5e07decc 9c6be5a2 458a1513 4bbbbc21 eda35ce5 a8b840a5 ef019ca5 c89bb183 bb00f0c0 06db1a26 7375ff57 1ca41d94 24aa662e
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff type: PX4
idtype: -00
vid: 000026ac
pid: 00000010
coa: Zu1H/9zzBXIrZQM28Wf0dbCDgb5U9Pv8wGdA1qw0/YDNOEN2p+M2wy71Czca206MF94FTJYGE7j2mI7xjQIMhvmjt01bJHjuPXF4H3syca+WiRYo
VE0u7vCHto1z1qLhApe8BnKXIm7GDuwDwwAbbG1Zzdf9XHKQd1CSqZ14=
sn: 0038001f3432470d31323533

Erase : [=====] 100.0%
Program: [=====] 100.0%
Verify : [=====] 100.0%
Rebooting.

```

Simulink自动代码生成

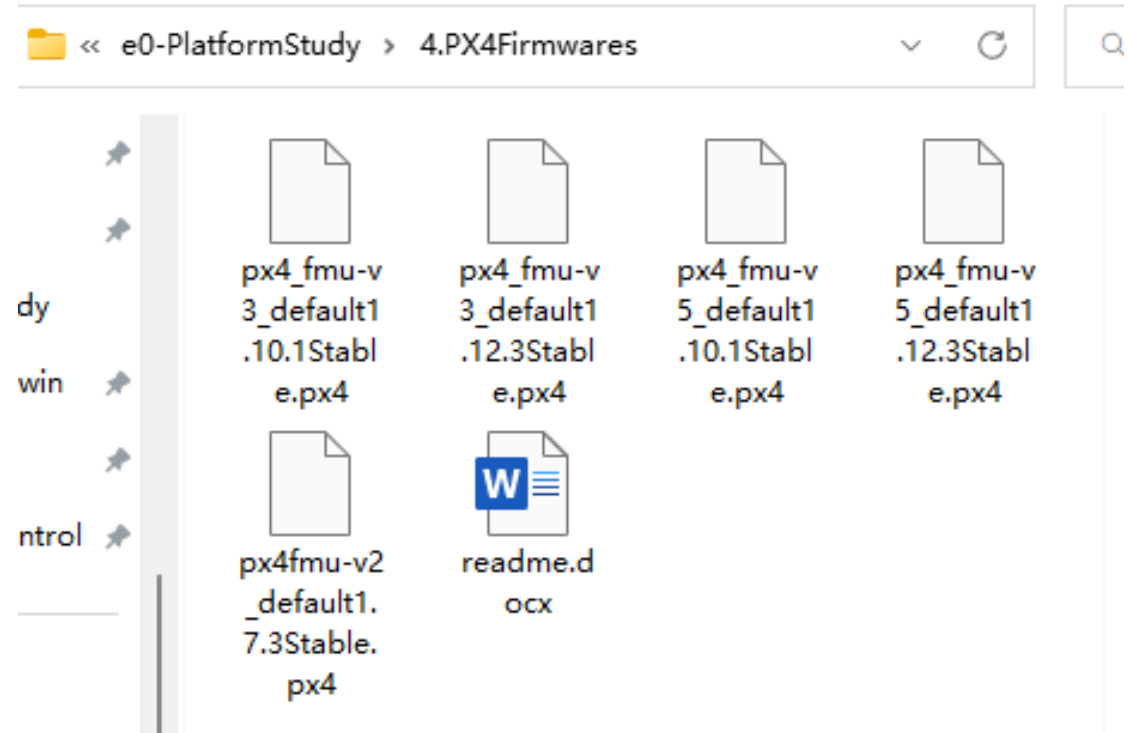
For more detailed interfaces, please refer to: [API](#).
 For more examples, please see: [Readme](#).



2. Introduction to Key Interfaces

2.4 Flight Controller Firmware Flashing Experiment

This experiment provides some flight controller firmware for conducting flight controller firmware flashing experiments using QGroundControl. For specific operating steps, please refer to the file: [0.ApiExps\4.PX4Firmwares\Readme.pdf](#) .



For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



2. Introduction to Key Interfaces

2.5 Log Data Recording

Using the binary logging module "binary_logger" to accomplish flight data writing and reading. For specific operating steps, please refer to the file [0.ApiExps\5.Log-Write-Read\Readme.pdf](#). The experimental results are as follows:

The screenshot displays a LabVIEW interface for data recording. On the left, a block diagram shows four waveform inputs connected to a 'double' data type, which is then connected to a 'data' input of a 'binary_logger' block. A 'z-20' block is also connected to the 'en' input of the 'binary_logger' block. A yellow callout box states: "Log File will be closed after 20 seconds of recording". The 'binary_logger' block is connected to a file path "/fs/microsd/log/pixhawk".

On the right, a file explorer window shows the contents of the "SDHC (E:) > log" directory. It contains folders for "sess011", "sess012", and "sess013", and files for "sess014" and "pixhawk_A.bin".

At the bottom, a command window shows the execution of the command: `>> [datapts, numpts]=px4_read_binary_file('pixhawk_A.bin')`. The results are displayed in a table:

名称	值
datapts	4x5000 double
numpts	5000

For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).

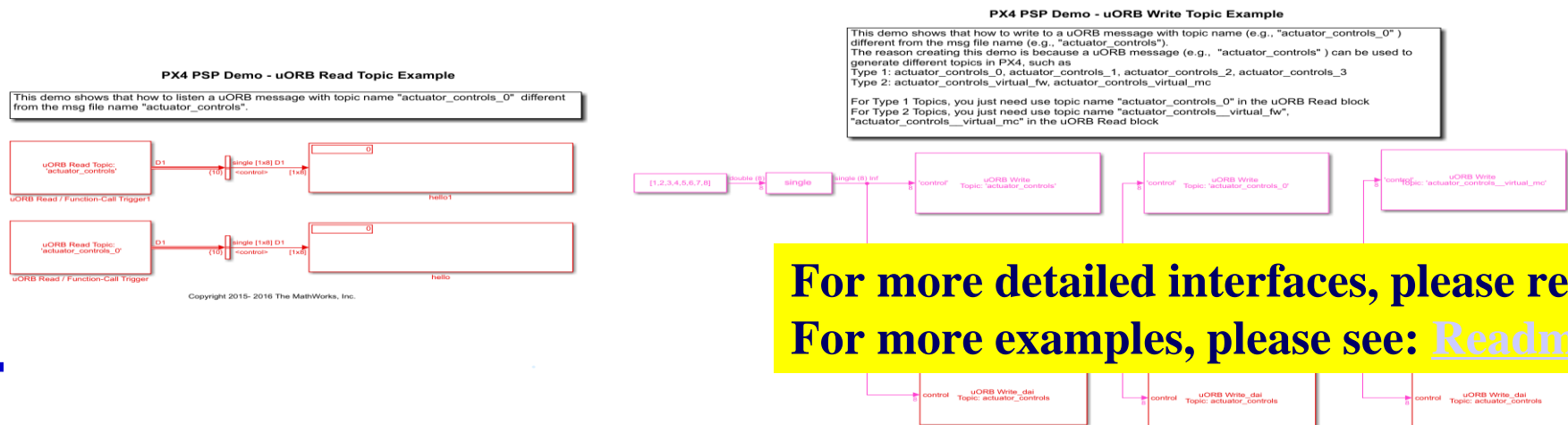




2. Introduction to Key Interfaces

2.6 uORB Read-Write Communication

The uORB messaging system in PX4 provides a powerful and convenient way for internal modules to interact with each other by allowing all modules to place data in a message pool, from which other modules can subscribe to the desired data. For specific operating steps, please refer to the file [0.ApiExps\6.uORB-Read-Write\Readme.pdf](#). The experimental results are as follows:



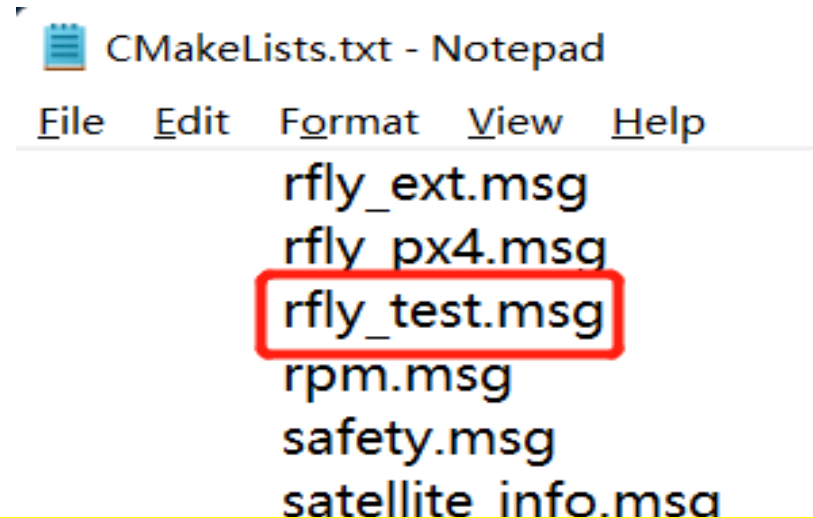


2. Introduction to Key Interfaces

2.7 Custom uORB Messages

By creating a custom uORB message, you can implement read and write functionality to familiarize yourself with and master PX4's uORB messaging system. For specific operating steps, please refer to the file [0.ApiExps\7.uORB-Create\Readme.pdf](#) . The experimental results are as follows:

```
rfly_test.msg X
D: > OneDrive > RflySimSource > RflySim
1  uint64 timestamp
2  float32[8] control
3  |
```



For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



2. Introduction to Key Interfaces

2.8 Feedback Prompt Messages

In flight control systems, it is often necessary to publish textual messages to reflect the current operational status of the system. This functionality can be achieved by sending uORB messages with "mavlink_log". For specific operating steps, please refer to the document [0.ApiExps\8.Mavlink-Msg-Echo\Readme.pdf](#) . The experimental results are as follows:

```
飞控选择 : Legacy FMU COM3  UDP Mode
UDP_Full
PX4: Simulink [Y] Radio 4 Value 1543
PX4: Simulink [Y] Radio 4 Value 1555 ◆◆◆◆◆ /◆ X
PX4: Simulink [Y] Radio 4 Value 1929
PX4: Simulink [Y] Radio 4 Value 1852
PX4: Simulink [Y] Radio 4 Value 1490
PX4: Simulink [Y] Radio 4 Value 1490
PX4: Simulink [Y] Radio 4 Value 1490
```

**For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).**



2. Introduction to Key Interfaces

2.9 External Communication of PX4 Controller

This example uses externally transmitted "rfly_ctrl" data as remote control inputs, while also sending received data to "rfly_px4" for external program feedback. For specific operating steps, please refer to the document [0.ApiExps\9.PX4CtrlExternalTune\Readme.pdf](#)

f. The experimental results are as follows:

For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).

The screenshot shows the QGroundControl interface with the MAVLink Inspector tool active. The interface displays real-time MAVLink messages. The following table summarizes the data shown in the inspector:

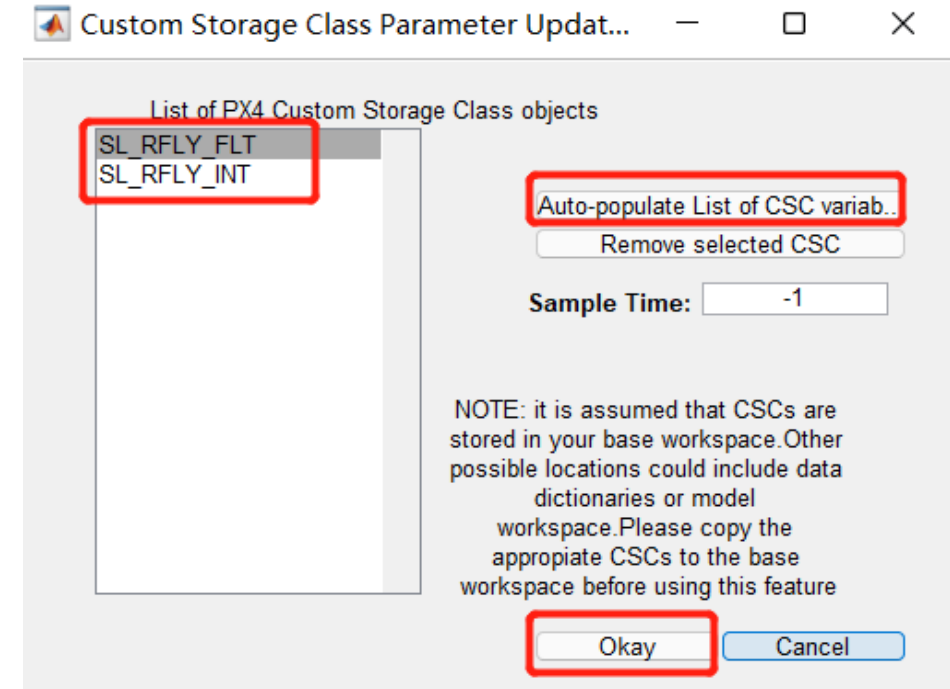
Message ID	Message Name	Frequency	Component	Count	Name	Value	Type
1	ACTUATOR_CONTROL_TARGET	30.0Hz	1	3242	time_usec	136000000	uint64_t
1	ALTITUDE	10.0Hz			group_mlx	123	uint8_t
1	ATTITUDE	50.7Hz			controls	1500, 1500, 1100, ...	float
1	ATTITUDE_QUATERNION	50.7Hz					
1	ATTITUDE_TARGET	8.0Hz					



2. Introduction to Key Interfaces

2.10 Real-Time Adjustment of Controller Parameters in QGC

During hardware-in-the-loop simulation and real-flight experiments, it is often necessary to observe the flight status in the QGroundControl (QGC) ground station and adjust controller parameters in real-time to achieve the best control performance for the aircraft. For specific operating steps, please refer to the document [0.ApiExps\10.QGC-Param-Tune\Readme.pdf](#). Partial experimental results are as follows:



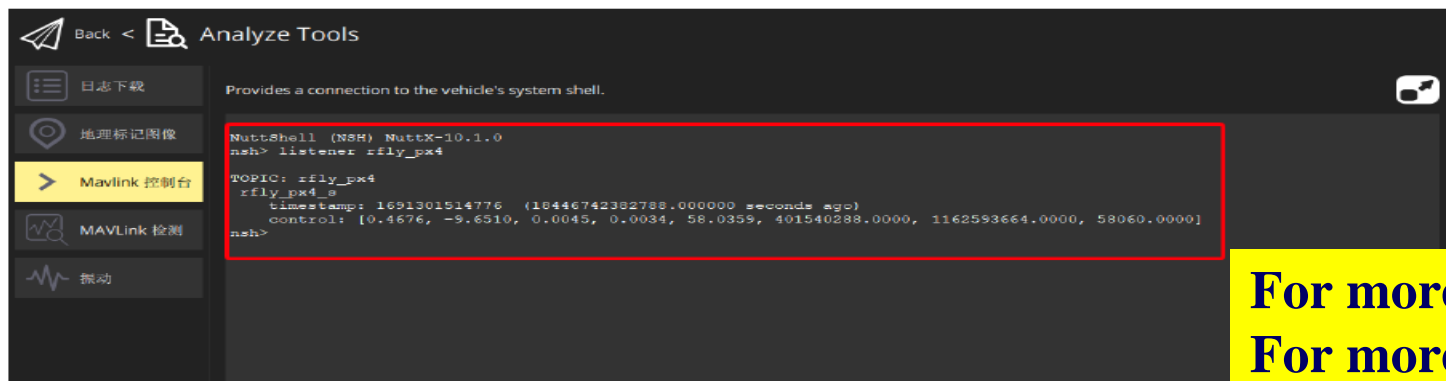
For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



2. Introduction to Key Interfaces

2.11 Sensor Data Reading Experiment

Through the low-level development interface of RflySim, accessible sensor data includes magnetometer, accelerometer, gyroscope, barometer, timestamp, and GPS data, among other information. This experiment will focus on acquiring partial data from the aforementioned sensors. By following this approach, a variety of sensor data can be subscribed to. For specific operating steps, please refer to the document [0.ApiExps\11.SenorDataGet\Readme.pdf](#). Partial experimental results are as follows:



```
NuttShell (NSH) NuttX-10.1.0
nsh> listener rfly_px4

TOPIC: rfly_px4_s
rfly_px4_s
timestamp: 1691301514776 (18446742382788.000000 seconds ago)
control: [0.4676, -9.6510, 0.0045, 0.0034, 58.0359, 401540288.0000, 1162593664.0000, 58060.0000]
nsh>
```

For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



2. Introduction to Key Interfaces

2.12 Autopilot CPU Usage Monitoring Experiment

When conducting low-level development on the RflySim platform, it is common to validate our developed algorithms on the autopilot hardware. However, when the algorithm model built in Simulink is automatically generated into autopilot firmware, the complexity of the algorithm and the rationality of model construction may lead to an overload of the autopilot system's CPU usage, causing experiment failure, as shown in the figure below. This experiment will demonstrate how to monitor the CPU usage of the autopilot system. For specific operating steps, please refer to the document [0.Api Exps\12.AutopilotCPUUsageGet\Readme.pdf](#). Partial experimental results are as follows:

```

PID COMMAND CPU(ms) CPU(%) USED/STACK PRIO(BASE) STATE FD
0 Idle Task 522 0.000 272/ 512 0 ( 0) READY 3
1 hwork 0 0.000 344/ 1260 249 (245) wsig 3
2 lpwork 0 0.000 344/ 1612 50 ( 50) wsig 3
3 init
4 wq:manager 0 0.000 400/ 1252 255 (255) wisem 4
5 netinit 1259 0.007 416/ 1564 49 ( 49) READY 3
6 Telnet daemon 0 0.000 528/ 2020 100 (100) wisem 0
393 log_writer_file 0 0.000 392/ 1164 60 ( 60) wisem 3
10 wq:hp_default 75 0.435 1100/ 1500 240 (240) wisem 4
20 dataman 0 0.000 768/ 1204 90 ( 90) wisem 4
22 wq:lp_default 3 0.000 832/ 1700 205 (205) READY 4
31 wq:I2C1 6 0.000 720/ 1460 246 (246) wisem 4
160 wq:nav_and_controllers 14 0.007 904/ 7196 241 (241) wisem 4
161 wq:rate_ctrl 11 0.000 856/ 1660 255 (255) wisem 4
163 commander 37 0.261 1288/ 3204 140 (140) wsig 6
164 commander_low_prio 0 0.000 716/ 2596 50 ( 50) wisem 6
172 mavlink_if0 27 0.261 1872/ 2924 100 (100) READY 4
275 mavlink_if1 16 0.274 1944/ 2936 100 (100) READY 4
277 mavlink_rov_if1 10 0.000 3644/ 5444 175 (175) wisem 4
295 wq:UART5 7 0.000 704/ 1396 233 (233) READY 4
360 wq:attitude_ctrl 0 0.000 432/ 1668 242 (242) wisem 4
367 navigator 1 0.000 960/ 1764 105 (105) wisem 5
383 logger 7 0.000 2160/ 3644 230 (230) wisem 3
397 mavlink_rov_if0 42 0.007 3824/ 5444 175 (175) wisem 6
396 PX4_Simulink_Task 3 0.000 952/ 2012 205 (205) wisem 10
398 <pthread> 15751 103.400 1176/ 2556 250 (250) wisem 10
399 mavlink_send 0 0.000 316/ 2020 100 (100) wisem 3
400 top 0 0.000 1452/ 2028 239 (235) RUN 3

Processes: 28 total, 7 running, 21 sleeping, max FDs: 15
CPU usage: 100.00% tasks, 0.00% sched, 0.00% idle
DMA Memory: 5120 total, 1024 used 1936 peak
Optime: 10.969s total, 0.522s idle
  
```

For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



2. Introduction to Key Interfaces

2.13 Experiment on Comparing Autopilot System Resource Usage Between M Function and S Function in Simulink

The flight control system of the PX4 firmware is based on the NuttX operating system. NuttX is a real-time embedded operating system (RTOS), known for its compact size and suitability for microcontroller environments. This experiment will flash Simulink models constructed separately by M Function and S Function. Through an analysis of the resource utilization of the autopilot system, it is observed that the Simulink model built with S Function consumes fewer autopilot resources. For detailed operating steps, please refer to the document [0.ApiExps\13.Simulink MS FuncVS\Readme.pdf](#). Partial experimental results are as follows:

```

PID  COMMAND  CPU(m)  CPU(%)  USED/STACK  PRIO(BASE)  STATE  FD
0  Idle Peak  35610  89.655  2827/ 512  0 ( 0)  READY  3
1  hpxwork  0  0.000  340/ 1628  248 (248)  wrstp  3
2  lpwork  0  0.000  340/ 1620  50 ( 50)  wrstp  3
3  init  0  0.000  2284/ 2332  100 (100)  wrsmn  3
4  wpmanager  0  0.000  427/ 1620  225 (225)  wrsmn  3
408  ekf2  2  0.013  764/ 2028  100 (100)  wrstp  3
22  wqisp_default  0  0.001  1132/ 1500  237 (237)  wrsmn  3
24  dtaman  26  0.006  785/ 1204  50 ( 50)  wrsmn  4
28  wqisp_default  10  0.061  804/ 1824  205 (205)  wrsmn  3
189  wqiuavcon  152  0.454  1796/ 3220  226 (226)  wrsmn  3
197  wuavcon_fw_srv  111  0.424  1972/ 6004  150 (150)  wrsmn  3
206  wjnav_and_controllers  36  0.202  1086/ 2244  242 (242)  wrsmn  3
207  wjnavs_ctrl  0  0.000  408/ 1836  225 (225)  wrsmn  3
209  commander  69  0.379  1440/ 3212  140 (140)  wrstp  5
217  mavlink_ifo  586  3.288  1948/ 2828  100 (100)  READY  6
261  mavlink_ifl  122  0.632  1860/ 2740  100 (100)  wrstp  4
262  mavlink_cov_ifl  22  0.131  1120/ 4476  175 (175)  wrsmn  4
351  px4io  34  0.151  544/ 1484  237 (237)  wrsmn  4
486  log_writer_file  0  0.000  364/ 1372  60 ( 60)  wrsmn  3
432  navigator  0  0.003  1052/ 1772  105 (105)  wrsmn  6
475  logger  29  0.142  2396/ 3644  220 (220)  wrsmn  3
493  optitrack  213  2.111  474/ 2664  250 (250)  wrsmn  3
492  PX4 Simulink Desk  0  0.000  700/ 2020  205 (205)  wrsmn  7
496  mavlink_cov_ifu  24  0.122  1452/ 4376  175 (175)  wrsmn  6
495  mavlink_shell  0  0.000  340/ 2028  100 (100)  wrsmn  3
500  top  70  0.401  1452/ 4084  237 (237)  RUN  3

Processes: 26 total, 3 running, 23 sleeping, max FDs: 16
CPU usage: 10.00% tasks, 0.30% sched, 89.68% idle
DMA Memory: 5120 total, 1024 used 1536 peak
Uptime: 37.432s total, 33.611s idle

```

```

PID  COMMAND  CPU(m)  CPU(%)  USED/STACK  PRIO(BASE)  STATE  FD
0  Idle Peak  114593  92.023  2827/ 512  0 ( 0)  READY  3
1  hpxwork  0  0.000  340/ 1628  248 (248)  wrstp  3
2  lpwork  0  0.000  340/ 1620  50 ( 50)  wrstp  3
3  init  0  0.000  2124/ 2332  100 (100)  wrsmn  3
4  wpmanager  0  0.000  427/ 1620  225 (225)  wrsmn  3
408  ekf2  1  0.012  764/ 2028  100 (100)  wrstp  3
22  wqisp_default  0  0.001  1132/ 1500  237 (237)  wrsmn  3
24  dtaman  26  0.006  785/ 1204  50 ( 50)  wrsmn  4
28  wqisp_default  5  0.040  784/ 1824  205 (205)  wrsmn  3
189  wqiuavcon  78  0.615  1796/ 3220  226 (226)  wrsmn  3
197  wuavcon_fw_srv  27  0.438  1972/ 6004  150 (150)  wrsmn  3
206  wjnav_and_controllers  18  0.205  1086/ 2244  242 (242)  wrsmn  3
207  wjnavs_ctrl  0  0.000  408/ 1836  225 (225)  wrsmn  3
209  commander  35  0.378  1440/ 3212  140 (140)  wrstp  5
217  mavlink_ifo  297  3.253  1636/ 2828  100 (100)  READY  6
261  mavlink_ifl  62  0.487  1724/ 2740  100 (100)  wrstp  4
262  mavlink_cov_ifl  11  0.129  1208/ 4476  175 (175)  wrsmn  4
351  px4io  17  0.154  544/ 1484  237 (237)  wrsmn  4
486  log_writer_file  0  0.000  364/ 1372  60 ( 60)  wrsmn  3
432  navigator  0  0.003  1052/ 1772  105 (105)  wrsmn  6
475  logger  13  0.149  2396/ 3644  220 (220)  wrsmn  3
493  optitrack  146  0.915  500/ 2664  250 (250)  wrsmn  3
492  PX4 Simulink Desk  0  0.000  684/ 2020  205 (205)  wrsmn  7
496  mavlink_cov_ifu  10  0.116  1452/ 4376  175 (175)  wrsmn  6
495  mavlink_shell  0  0.000  340/ 2028  100 (100)  wrsmn  3
500  top  35  0.397  1452/ 4084  237 (237)  RUN  3

Processes: 26 total, 3 running, 23 sleeping, max FDs: 16
CPU usage: 7.67% tasks, 0.30% sched, 92.02% idle
DMA Memory: 5120 total, 1024 used 1536 peak
Uptime: 123.013s total, 114.993s idle

```

For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



2. Introduction to Key Interfaces

2.14 Sensor Data Reading Experiment

Through the low-level development interface of RflySim, accessible sensor data includes magnetometer, accelerometer, gyroscope, barometer, timestamp, and GPS data, among other information. This experiment will focus on acquiring partial data from the aforementioned sensors. By following this approach, a variety of sensor data can be subscribed to. For specific operating steps, please refer to the document [0.ApiExps\14.SITLVeriGenCodeFirm\Readme.pdf](#). Partial experimental results are as follows:

```
Back < Analyze Tools
日志下载 Provides a connection to the vehicle's system shell.
地理标记图像
Mavlink 控制台
MAVLink 检测
震动

NuttShell (NSH) NuttX-10.1.0
nsh> listener rfly_px4

TOPIC: rfly_px4
rfly_px4_0
timestamp: 1691301514776 (18446742382788.000000 seconds ago)
control: [0.4676, -9.6510, 0.0045, 0.0034, 58.0359, 401540288.0000, 1162593664.0000, 58060.0000]
nsh>
```

For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



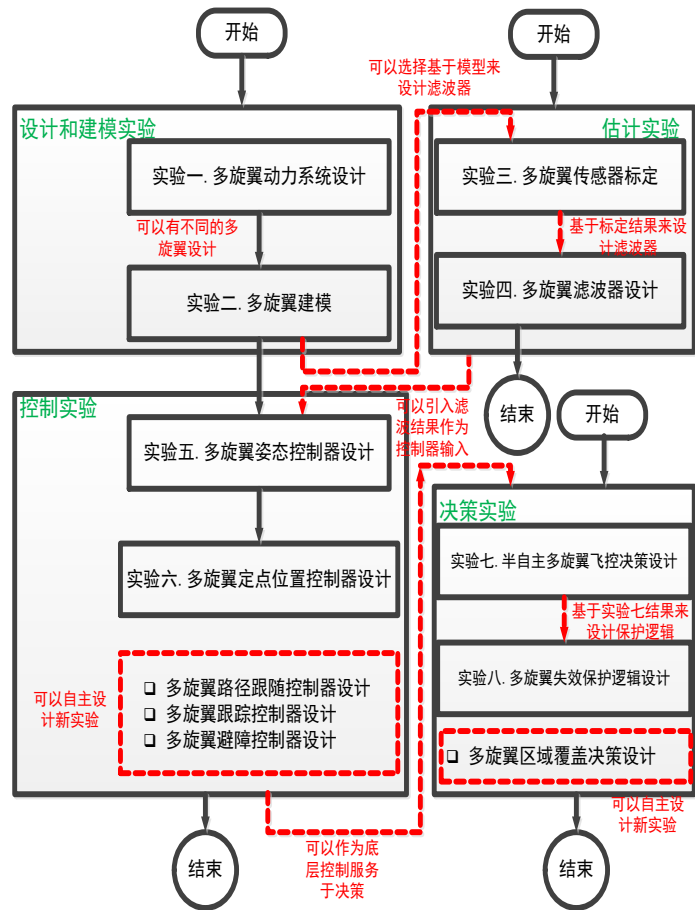
Outline

1. Experimental Platform Configuration
 2. Introduction to Key Interfaces
 3. **Basic Experimental Cases (Free Version)**
 4. Advanced Interface Experiments (Free Version)
 5. Advanced Case Experiments (Free Version)
 6. Extended Cases (Full Version)
 7. Summary
-



3. Basic Experimental Cases (Free Version)

3.0 Advanced Experiment Overview



- The routines provided on this platform ensure that each experiment or section thereof can **be completed independently**.
- To introduce diversity in the task objectives, our curriculum can be structured progressively. The progressive path can be divided into:
 - (1) Design and Modeling Experiment -> Control Experiment
 - (2) Design and Modeling Experiment -> Control Experiment -> Decision-making Experiment
 - (3) Design and Modeling Experiment -> Estimation Experiment -> Control Experiment -> Decision-making Experiment
- Different types of aircraft need to be designed, resulting in varying **models and different modeling approaches**. Similarly, **the design of control experiments varies**.
- Teachers may also choose to add **additional experiments at their discretion**.



3. Basic Experimental Cases (Free Version)

3.0 Advanced Experiment Overview

Open the routine, read and execute the program code, then observe, record, and analyze the data.

Guide the reader to modify the routine, run the modified program, and collect and analyze the data.

Based on the two aforementioned experiments, independently design for the given task.



3.Basic Experimental Cases (Free Version)

3.0 Advanced Experiment Overview

Table. Types, objectives, and content of the experiment

Aim	Basic experiment	Analysis experiment	Design the experiment
Familiar with development platform	✓	✓	✓
Be familiar with the analysis process	×	✓	✓
Familiar with design methods	×	×	✓
Carry out software-in-the-loop simulation	✓	✓	✓
Perform hardware-in-the-loop simulation	✓	✓	✓
Actual experimental test	×	×	✓



3. Basic Experimental Cases (Free Version)

3.1 Power system design experiment

The objectives of this lab are as follows:

- 1、 design a power system of that multi-rotor aircraft by use a multi-rotor flight evaluation website;
- 2、 According to the known information, the power system of the multi-rotor aircraft is designed and compared with the parameters generated by the multi-rotor flight evaluation website, and the effects of different cities, temperatures, propeller sizes and numbers on the hovering events of the multi-rotor aircraft are analyzed.

Please learn the advanced version of the course for the specific experimental principles, and see the file [1. BasicExps\e1-FlightEval\Readme.pdf](#) . PDF for the experimental operation steps.

For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



3. Basic Experimental Cases (Free Version)

3.1 Power system design experiment

The experimental results are as follows (in part):

详细信息

悬停性能 :	最大油门性能 :	整体性能 :
悬停时间 : 22.5 min.	飞行时间 : 7 min.	正常使用 : 17.8 min.
油门百分比 : 63.6 %	总升力 : 94.3 N	整机重量 : 4.56 kg
电调电流 : 6.69 A	电机电流 : 21.8 A	剩余载重 : 2.8 kg
电机转速 : 4623.5 rpm	电机转速 : 6716.3 rpm	最大起飞海拔 : 3.85 km
电机输出功率 : 132.2 W	电机输出功率 : 417.8 W	最大倾斜角度 : 51.7 °
电池输出电压 : 23.7 V	电池输出电压 : 22.9 V	最大平飞速度 : 12.4 m/s
电池输出电流 : 27.2 A	电池输出电流 : 87.3 A	单程飞行距离 : 8.5 km
能量效率 : 80.9 %	能量效率 : 79.8 %	抗风等级 : 4 级

Propeller size/inch	Hover time/min
10	17
9.4	16.5
9	15.9
8	14.5

Location	Elevation/m	Hover time/min
Shanghai	4	16.5
Beijing	43.5	16.5
Changsha	500	16.1
Lhasa	3658	13.5

Temperature	Hover time/min
0	17.1
10	16.8
20	16.6
30	16.3

For more detailed interfaces, please refer to: [API](#).

For more examples, please see: [Readme](#).



飞思灵



3. Basic Experimental Cases (Free Version)

3.2 Dynamic modeling experiment

The objectives of this lab are as follows:

1. The effects of multi-rotor mass, moment of inertia matrix, propeller thrust coefficient and propeller thrust coefficient on the multi-rotor flight performance are analyzed.
2. Build a complete multi-rotor aircraft model, and add a three-dimensional model of the quad rotor in RflySim3D.

Please learn the advanced version of the course for the specific experimental principle. See the file [1.BasicExps\e2-UavModeling\Readme.pdf](#) for the experimental operation steps.

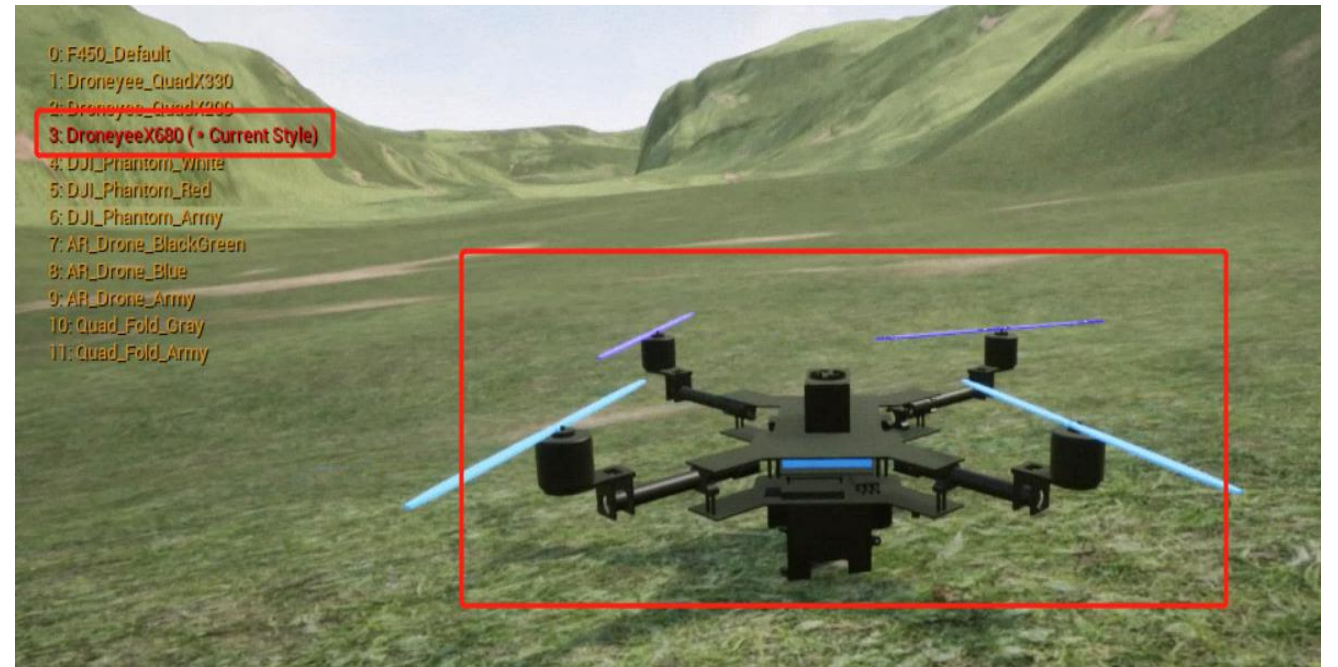
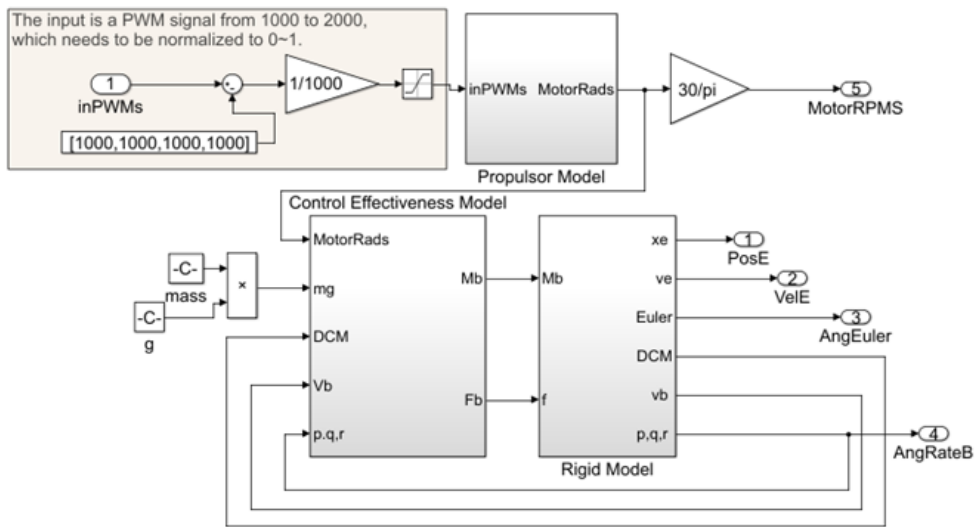
For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



3. Basic Experimental Cases (Free Version)

3.2 Dynamic modeling experiment

The experimental results are as follows (in part):



For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



3.Basic Experimental Cases (Free Version)

3.3 Sensor calibration experiment

The objectives of this lab are as follows:

1. Complete the calibration of acceleration according to the experimental steps.
2. According to the given error model of magnetometer, the data acquisition model of magnetometer is designed, and the optimal solution of model parameters is obtained by using the measured data and LM algorithm function to complete the calibration of magnetometer.

Please learn the advanced version of the course for the specific experimental principles, and see the file [1.BasicExps\e3-SensorCalib\readme.pdf](#) for the experimental operation steps.

For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



3. Basic Experimental Cases (Free Version)

3.3 Sensor calibration experiment

The experimental results are as follows (in part):

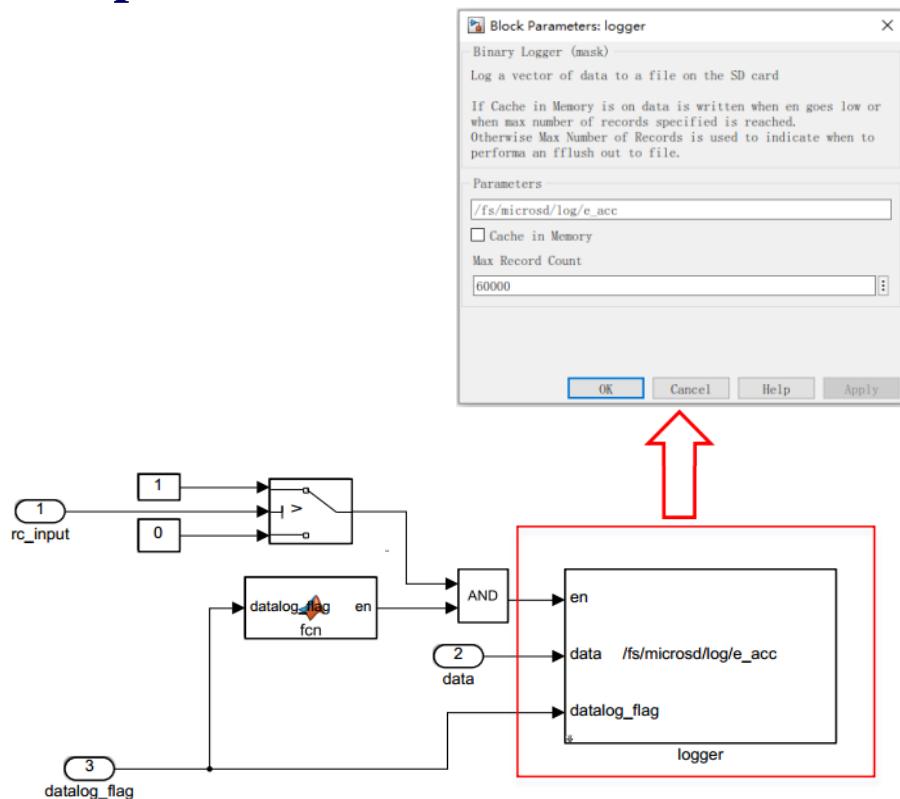


图. 加速度数据采集“binary_logger”模块

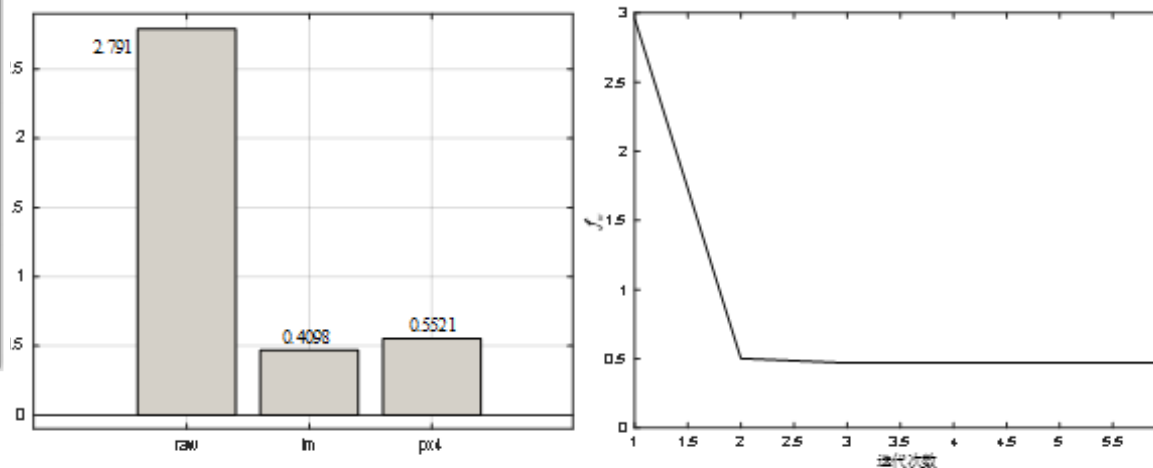


图. 磁力计校正对比值

For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



3. Basic Experimental Cases (Free Version)

3.4 Sensor calibration experiment

The objectives of this lab are as follows:

1. According to the data provided in the experiment, complete the complementary filtering, and compare it with the original data and the data calculated by Pixhawk's own filter to understand the advantages of the complementary filter.

2. Improve the parameters of the complementary filter and analyze the influence of the parameters of the complementary filter on the filtering effect.

3. Understand the principle of Kalman filter, design the Kalman filter to realize the filter, process the acceleration and angular velocity data, and draw the data map of the relevant attitude angle.

Please learn the advanced version of the course for the specific experimental principles, and see the file [1. BasicExps\e4-FilterDesign\Readme.pdf](#) for the experimental operation steps.

For more detailed interfaces, please refer to: [API](#).

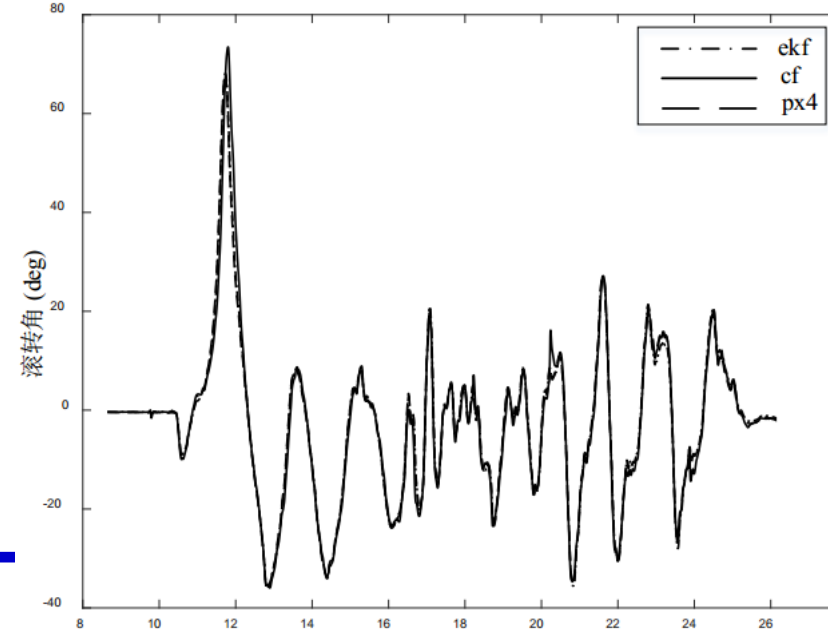
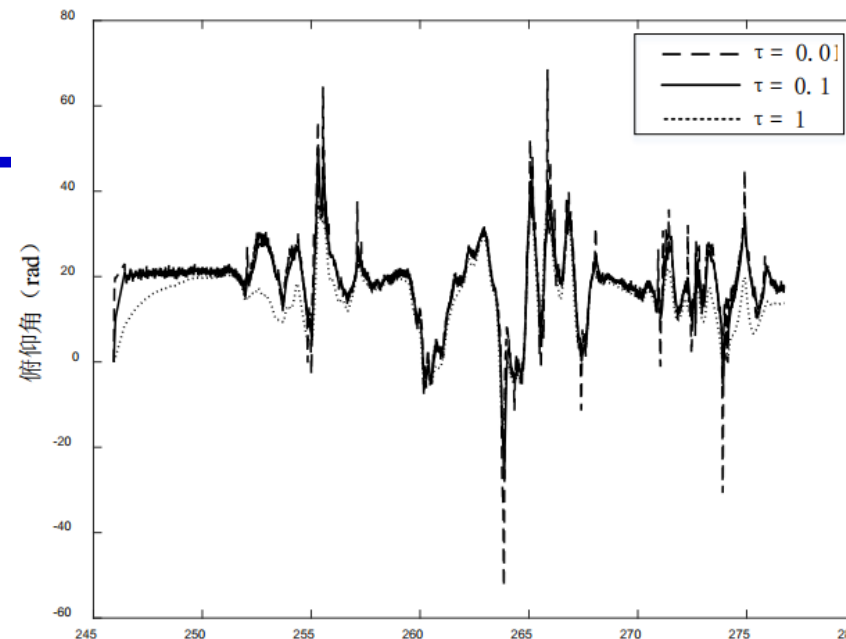
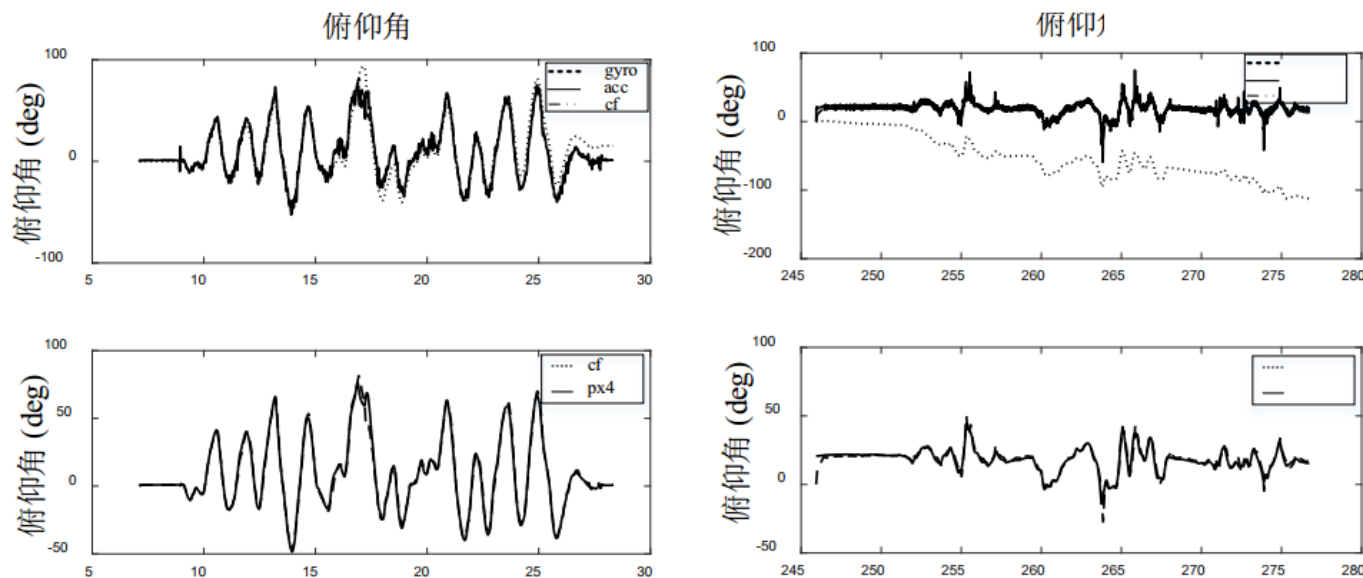
For more examples, please see: [Readme](#).



3. Basic Experimental Cases (Free Version)

3.4 Sensor calibration experiment

The experimental results are as follows (in part):



For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



3. Basic Experimental Cases (Free Version)

3.5 Attitude controller design experiment

The objectives of this lab are as follows:

1. Reproduce the Simulink simulation of the quadrotor aircraft, analyze the role of the control distributor; record the step response of the attitude, sweep the open-loop attitude control system to draw the Bode diagram, and analyze the stability margin of the closed-loop attitude control system; complete the hardware-in-the-loop simulation of the quadrotor aircraft.

2. Adjust the relevant parameters of PID controller to improve the control performance and record the overshoot and adjustment time to get a set of appropriate parameters; use the parameters after debugging to sweep the frequency of the system to draw the Bode diagram, observe the amplitude-frequency response and phase-frequency response curve of the system, and analyze its stability margin.

3. The transfer function model of the attitude control channel is established, and the correction controller is designed. The software and hardware simulation experiments and flight experiments are carried out with the controller designed by ourselves.

Please learn the advanced version of the course for the specific experimental principles, and see the file [1.BasicExps\e5-AttitudeCtrl\Readme.pdf](#) for the experimental operation steps.

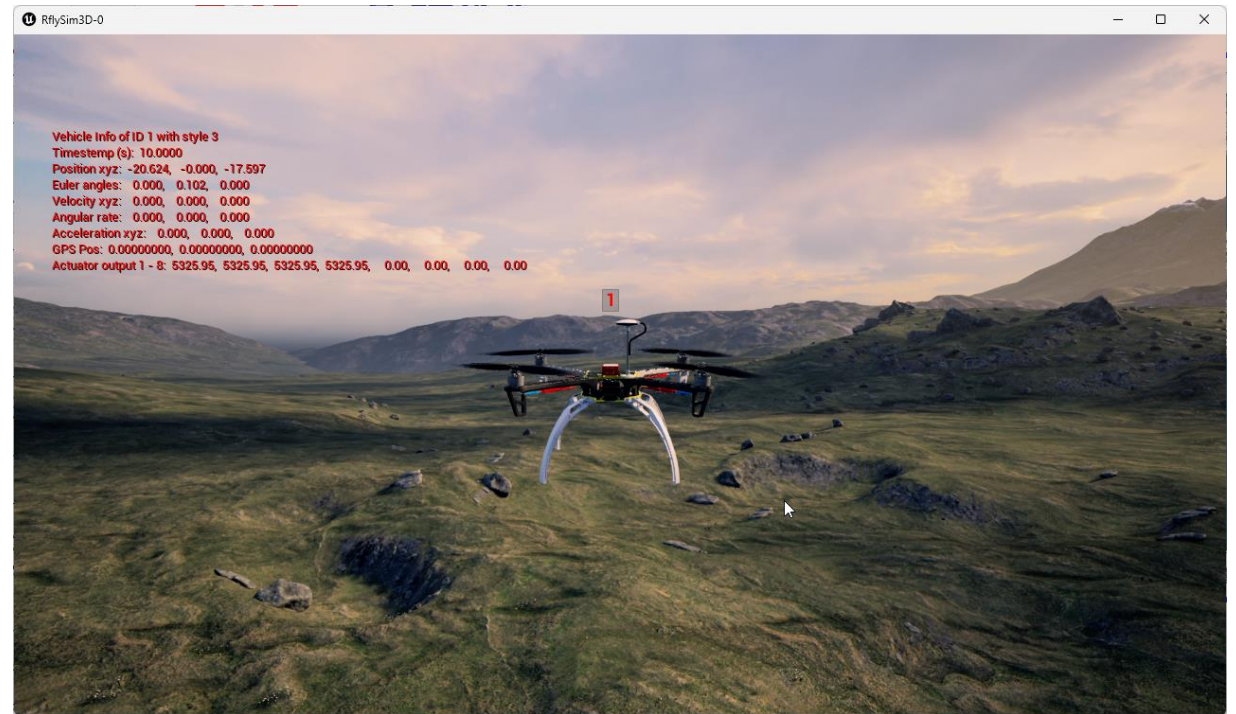
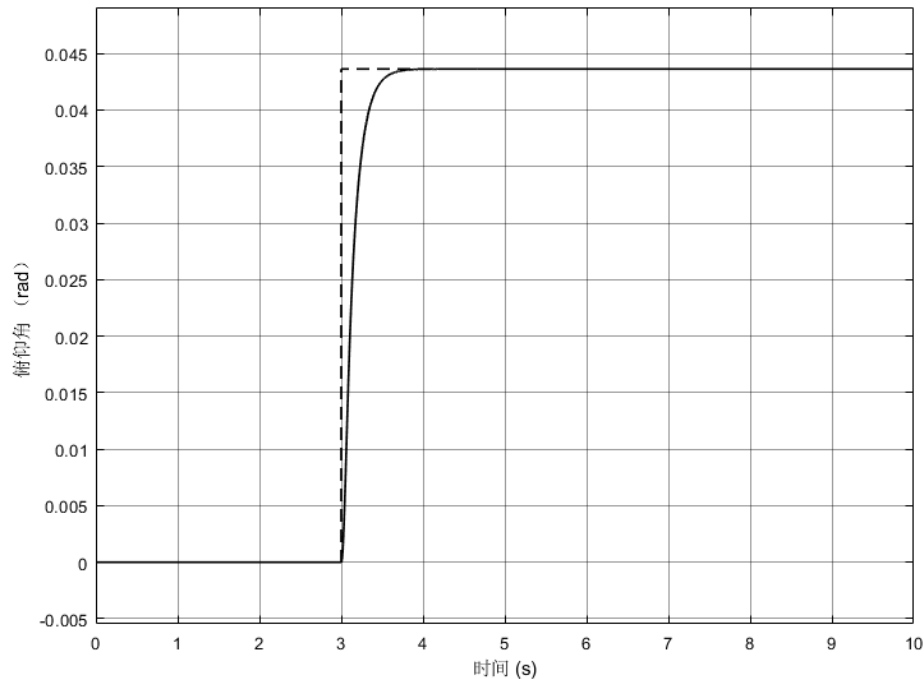
For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



3.Basic Experimental Cases (Free Version)

3.5 Attitude controller design experiment

The experimental results are as follows (in part):



For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



3. Basic Experimental Cases (Free Version)

3.6 Fixed-point position controller design experiment

The objectives of this lab are as follows:

1. Reproduce the quadrotor Simulink simulation to analyze the decoupling of the control action on the axis ${}_{O_b}x_b$ and the axis ${}_{O_b}y_b$; sweep the frequency of the system to draw the bode diagram to analyze the stability margin of the closed-loop position control system.

2. Adjust the relevant parameters of the PID controller to improve the control performance of the system. After obtaining satisfactory parameters, sweep the frequency of the system to draw the Bode diagram.

3. Establish the transfer function model of the position control channel, use the MATLAB "Control System Designer" to design the correction controller, and adjust the system error, relative margin and other parameters.

Please learn the advanced version of the course for the specific experimental principle. See the file [1.BasicExperiments\PositionCtrl\Readme.pdf](#) for the experimental operation steps.

For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



3. Basic Experimental Cases (Free Version)

3.6 Fixed-point position controller design experiment

The experimental results are as follows (in part):

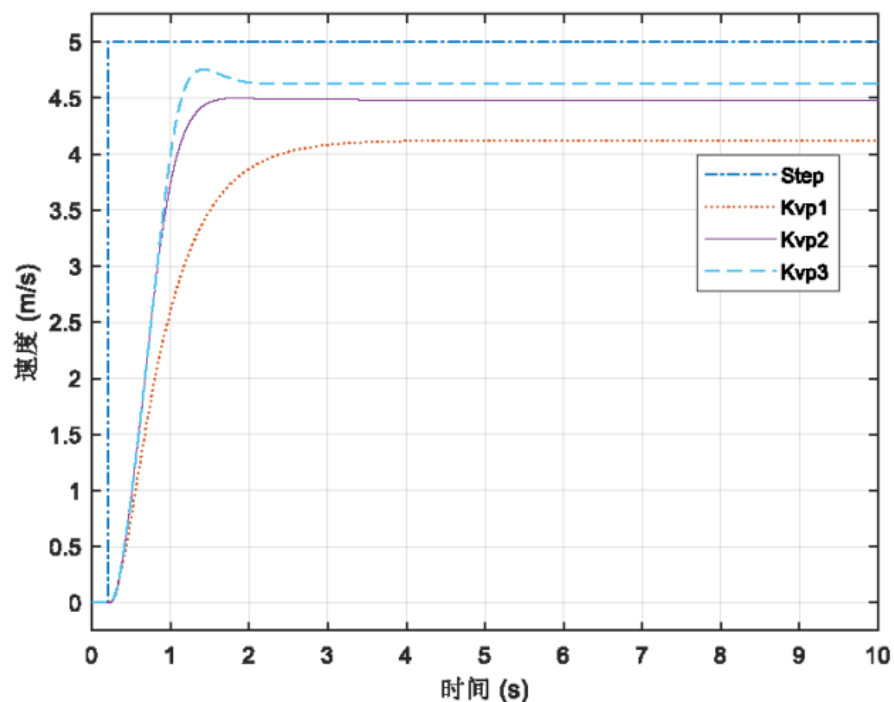
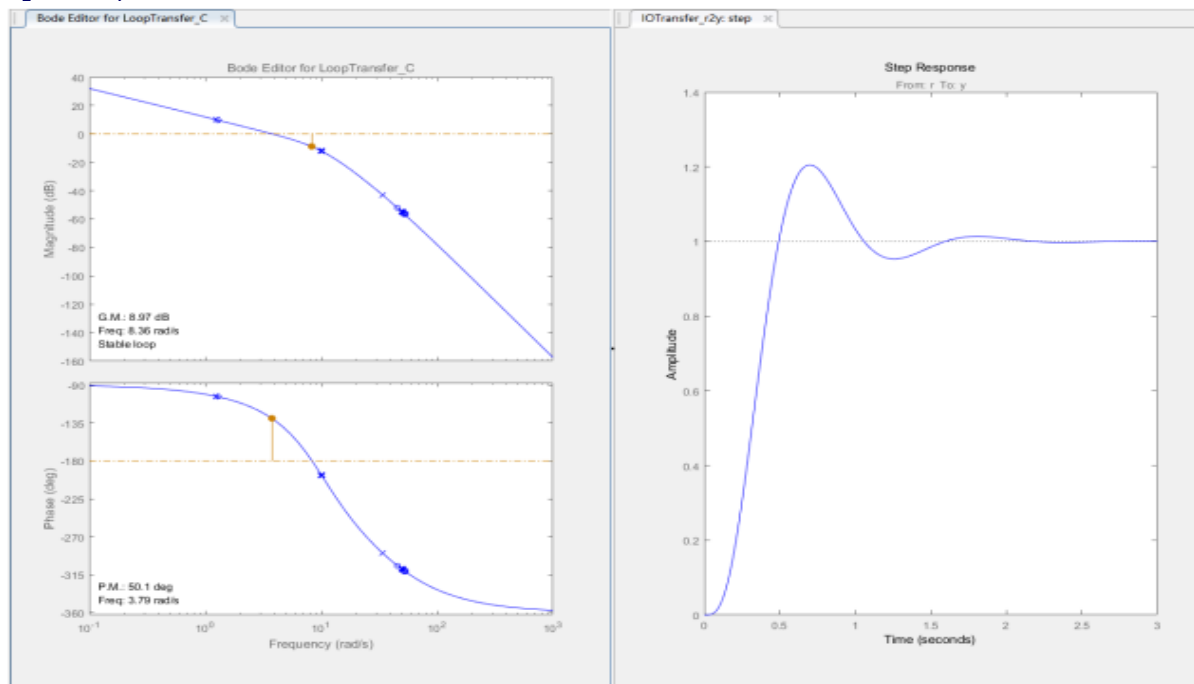


图. 不同比例项系数下的阶跃响应



For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



3. Basic Experimental Cases (Free Version)

3.7 Semi-autonomous control mode design experiment

The objectives of this lab are as follows:

1. On the controller design and simulation platform based on Simulink, the characteristics of the attitude and position response of the quadrotor are reproduced from the simulation experiment;
2. Change to fixed height mode on the basis of self-stabilization mode. According to the experimental analysis, the change of attitude and position output values of the multi-rotor in the fixed altitude mode is compared with that in the self-stabilized mode.
3. Change to fixed-point mode on the basis of self-stabilization mode. According to the experimental analysis, compared with the self-stabilization mode, the output values of the attitude and position of the multi-rotor in the fixed-point mode are changed, and the free switching of the three modes is realized by using a three-segment dial switch.

Please learn the advanced version of the course for the specific experimental principle. See the file [1.BasicExps\e7-SemiAutoCtrl\Readme.pdf](#) for the experimental operation steps.

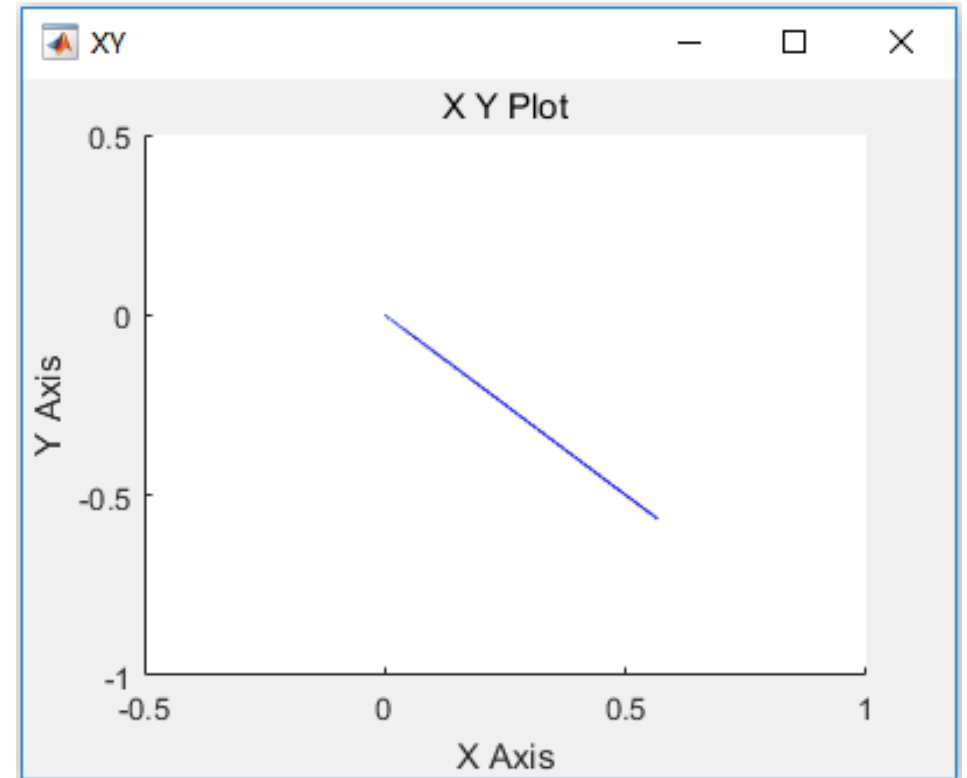
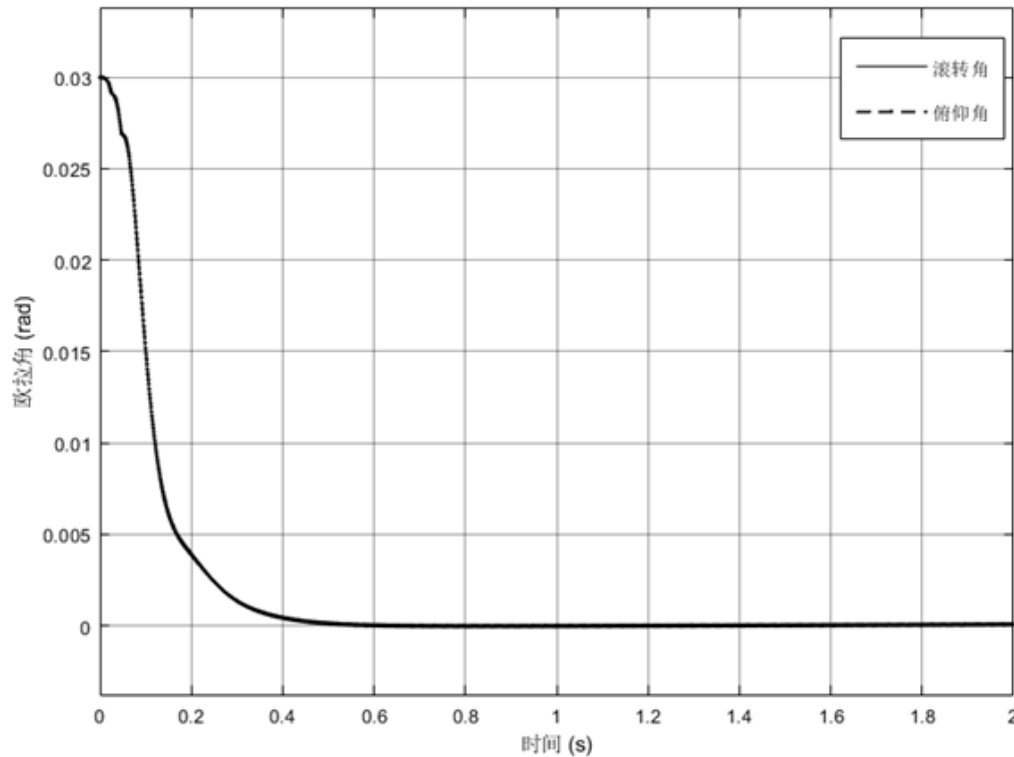
For more detailed interfaces, please refer to: [API](#).
For more examples, please see: [Readme](#).



3. Basic Experimental Cases (Free Version)

3.7 Semi-autonomous control mode design experiment

The experimental results are as follows (in part):



For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



3. Basic Experimental Cases (Free Version)

3.8 Fail-safe logic design experiment

The objectives of this lab are as follows:

1. In the Simulink simulation environment, in the manual mode, realize the return and landing of the aircraft, and record and analyze the simulation results.
2. On the basis of the basic experiment, the corresponding state transition is added to realize the return and landing of the aircraft in the manual mode, and the return and landing can be switched to each other.
3. On the basis of the previous experiment, add the remote control power failure and loss of contact event, complete the new mode and switching design, that is, add two States of failure return and failure landing, and complete the design of the state machine.

Please learn the advanced version of the course for the specific experimental principle. See the file [1.BasicExps\e8-FailsafeLogic\Readme.pdf](#) for the experimental operation steps.

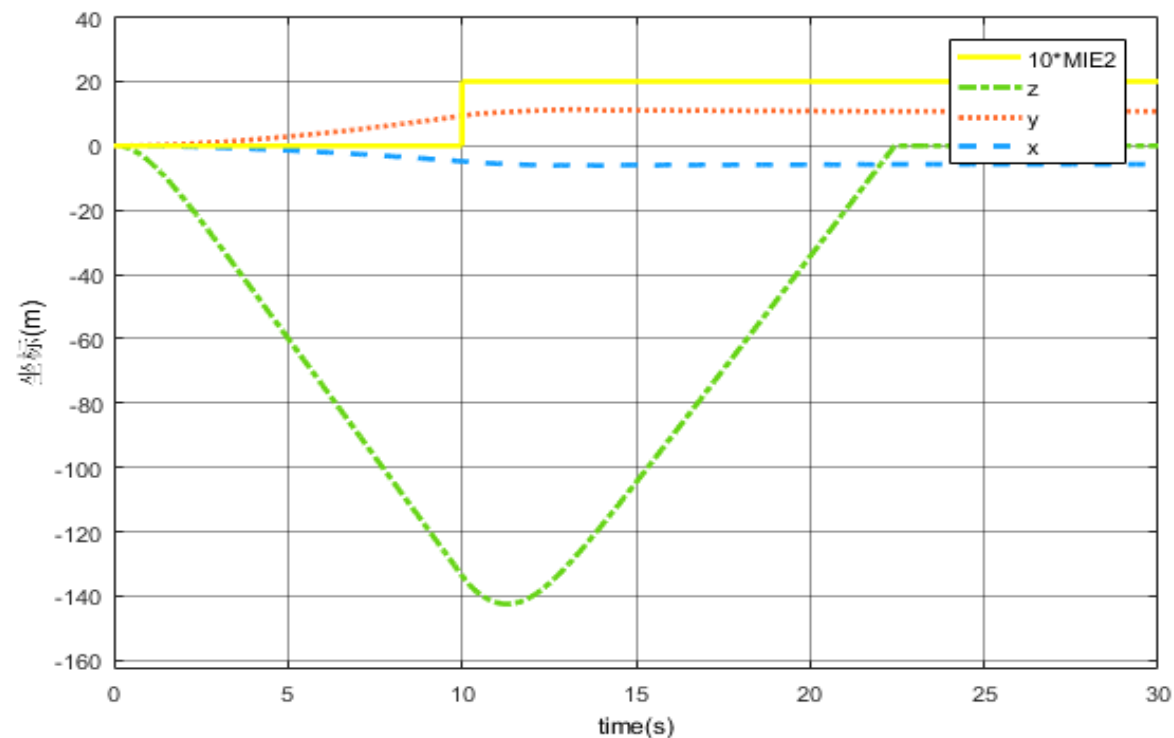
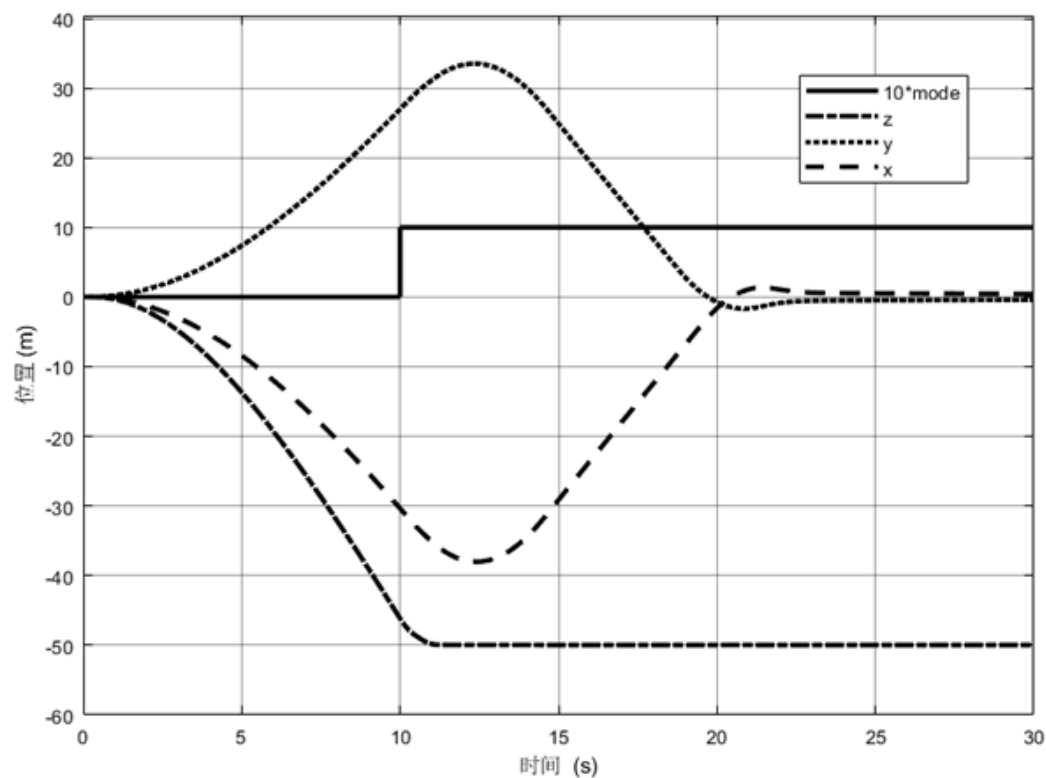
For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



3.Basic Experimental Cases (Free Version)

3.8 Fail-safe logic design experiment

The experimental results are as follows (in part):



For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



3. Basic Experimental Cases (Free Version)

3.9 PX4 Module Replacement Lab

The objectives of this lab are as follows:

In order to quickly replace some native modules (sensors, filters, attitude controllers, etc.) of the PX4 control software with the generated Simulink code, the experiment provides two methods to achieve this:

1. Open the " Firmware\src\modules\ekf2\ekf2_main.cpp "file, and manually comment out the module code to be shielded;
2. Modify the startup script file " Firmware\ROMFS\px4fmu_common\init.d\rcS " of the PX4 module, and comment out the module to be shielded.

Please learn the advanced version of the course for the specific experimental principles. The lab steps are shown in File [1.BasicExps\e9-ReplacePX4Module\Readme.pdf](#) .

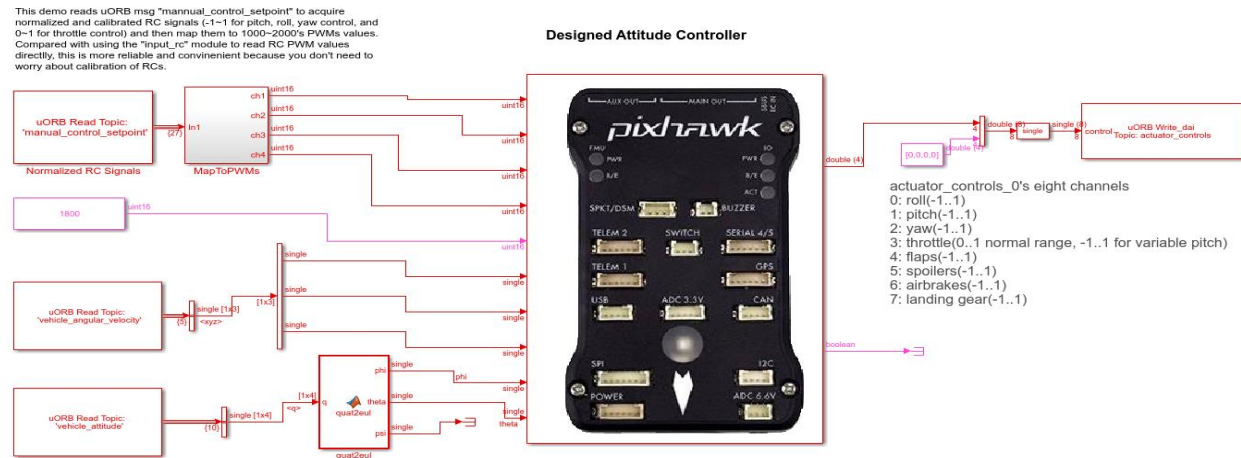
For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



3. Basic Experimental Cases (Free Version)

3.9 PX4 Module Replacement Lab

In this experiment, the "Position & Attitude Estimator" filter module "is shielded, and the section 2.3 (attitude controller design experiment) is modified to build the " Exp6 _ ReplacePX4AttitudeCtrler.slx "model, as follows:



Note: After the development of this experiment is completed, please be sure to return the modified code to the original place, so as not to affect the development of other functions.



For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



Outline

1. Experimental Platform Configuration
 2. Introduction to Key Interfaces
 3. Basic Experimental Cases (Free Version)
 - 4. Advanced Interface Experiments (Free Version)**
 5. Advanced Case Experiments (Free Version)
 6. Extended Cases (Full Version)
 7. Summary
-



4. Advanced Interface Experiments(Free Version)

4.1 User-defined shielding of any module output in PX 4

In this experiment, the relevant functions of PX 4 are shielded by replacing the uORB message " actuator_controls_0 " of the attitude angular rate ring in PX4 and replacing the modified CPP file. The experimental steps are described in detail. See the file [2.AdvExps/e0 AdvApiExps\1.CusMask PX4Code\Readme.pdf](#) . PDF for the specific operation steps. The (partial) experimental results are as follows:



For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



4. Advanced Interface Experiments(Free Version)

4.2 Rename PX 4 Application Name Experiment:

Based on the multi-process running status of PX 4 software system, the name of PX 4 application generated by MATLAB is PX 4 _ simulink _ a pp. This experiment can rename it and create a new application in PX 4 software system and compile it. See the file [2.AdvExps\20_AdvApiExps\2.RenamePX4App\Readme.pdf](#) for the specific operations. The (partial) experimental results are as follows:

```
>> PX4AppName 'rfly_simulink_app'  
Firmware目录中已存在rfly_simulink_app目录。  
当前的编译命令为: px4_fmuv5_default  
成功找到px4_fmuv5_default的cmake文件  
重命名完成。  
开始重新添加px4_simulink_app模版...
```

For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



4. Advanced Interface Experiments(Free Version)

4.3 Loading PX 4 application experiment:

The RflySim platform supports the loading of custom developed PX 4 applications. The PX 4 applications provided in this experiment can be directly loaded into the PX 4 software system for firmware compilation. See the file [2.AdvExps/e0 AdvApiExps/3.LoadPX4App/Readme.pdf](#) for the specific operation steps. The (partial) experimental results are as follows:

```
>> PX4AppLoad('C:\PX4PSP\rfly_simulink_app')
当前的编译命令为: px4_fmu-v5_default
Firmware目录中已存在rfly_simulink_app目录。
当前的编译命令为: px4_fmu-v5_default
成功找到px4_fmu-v5_default的cmake文件
重命名完成。
开始重新添加px4_simulink_app模版...
```

For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



4. Advanced Interface Experiments(Free Version)

4.4 Creating Multiple PX 4 Application Labs:

Based on the multi-process running status of PX4 software system, the PX4 application name generated by MATLAB code automatically is `px4 _ simulink _ app`, which can be renamed in this experiment, and then a new PX4 application can be generated by MATLAB code automatically. This allows multiple PX 4 applications to be created at the same time. See the file [2.AdvExps\e0 AdvApiExps\4.MultPX4App\Readme.pdf](#) for the specific operation steps. The (partial) experimental results are as follows:

```
>> PX4AppName 'rfly_simulink_app'  
Firmware目录中已存在rfly_simulink_app目录。  
当前的编译命令为: px4_fmu-v5_default  
成功找到px4_fmu-v5_default的cmake文件  
重命名完成。  
开始重新添加px4_simulink_app模版...
```

For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



Outline

1. Experimental Platform Configuration
 2. Introduction to Key Interfaces
 3. Basic Experimental Cases (Free Version)
 4. Advanced Interface Experiments (Free Version)
 5. Advanced Case Experiments (Free Version)
 6. Extended Cases (Full Version)
 7. Summary
-



5. Advanced Case Experiments (Free Version)

Under development...

For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



Outline

1. Experimental Platform Configuration
 2. Introduction to Key Interfaces
 3. Basic Experimental Cases (Free Version)
 4. Advanced Interface Experiments (Free Version)
 5. Advanced Case Experiments (Free Version)
 6. Extended Cases (Full Version)
 7. Summary
-



6. Extended Case (Full Version)

6.1 Experiment of ESO design:

The scheme is applicable to the aircraft control problem under disturbance and uncertainty. By decouple that aircraft into multiple cascaded SISO system comprising a nominal model and an uncertainty comprising model/identification uncertainties, control mismatch, and external disturbances. In order to estimate the system state and the total uncertainty, an extended state observer (ESO) is designed. Using the output of ESO, the controller compensates the total uncertainty online. See File [3.CustExps\e1_ESO-CtrlExp\Readme.pdf](#) for specific operation steps.

For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



6. Extended Case (Full Version)

6.2 Optimal control experiment based on reinforcement learning:

The optimal control problem is decomposed into two phases: approximate policy evaluation and policy promotion by using the model-based reinforcement learning method and the approximate policy iteration algorithm. In the approximate policy evaluation phase, a linear approximator is used to approximate the value function, and the system model and Bellman equation are used to update the parameters of the approximator. In the policy lifting phase, a linear-structured approximator is used to approximate the optimal control policy, and the value function and the system model are used to update the parameters of the approximator. The two phases alternate until convergence to a near optimal solution. Based on this, the experiment first observes the uncertainty of the aircraft model based on the extended state observer and compensates it, then approximates the optimal value function of the compensated system by using the model-based reinforcement learning optimal control method, and then determines the optimal control law, and then designs the safety feedback item for the optimal control law based on the control barrier function. The forward invariance of the safety set of the closed-loop system is guaranteed. See File [3.CustExps/e2_RL-CtrlExp/Readme.pdf](#) for specific operation steps.



For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



6. Extended Case (Full Version)

6.3 Model Compensated Controller (MCC) Design Experiment:

All the experiments in this folder are model compensation controller (MCC) design experiment routines, and the traditional Extended State Observer (ESO) is abandoned in MCC. A Compensation Function Observer (CFO) with higher accuracy is used to estimate the complex disturbance or fast time-varying disturbance with high accuracy, and the estimation of the total disturbance is fed back to the controller to realize the high-precision tracking control of the UAV system. This folder contains the quadrotor controller design routines for attitude, altitude, position, and semi-bootstrap modes. See File [3.CustExps\e3_MCC-CtrlExp\Readme.pdf](#) for specific operation steps.

For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



6. Extended Case (Full Version)

6.4 Design experiment of ADRC controller:

All the experiments in this folder are quadrotor-based ADRC design experiment routines, a model-free control method suitable for designing controllers for plants with unknown dynamics and internal and external disturbances. This algorithm only needs to approximate the dynamic characteristics of the controlled object, and can design a controller with robust anti-disturbance function and no overshoot. This folder contains the quadrotor controller design routines for attitude, altitude, position, and semi-bootstrap modes. See File [3.CustExps\e4 ADRC-CtrlExp\Readme.pdf](#) for specific operation steps.

For more detailed interfaces, please refer to: [API](#). For more examples, please see: [Readme](#).



Outline

- 1. Experimental Platform Configuration**
- 2. Introduction to Key Interfaces**
- 3. Basic Experimental Cases (Free Version)**
- 4. Advanced Interface Experiments (Free Version)**
- 5. Advanced Case Experiments (Free Version)**
- 6. Extended Cases (Full Version)**
- 7. Summary**



7. Summary

This session primarily focuses on the development course of flight control algorithms, divided into two parts: basic experiments and advanced experiments. This is to help students quickly become familiar with the theoretical design of multirotors, simulation using the RflySim platform, and the development process of physical control.

The basic experiments mainly involve learning the simulation process based on the RflySim platform, focusing on software-in-the-loop and hardware-in-the-loop simulations. The advanced experiments follow a learning path from theoretical design and modeling experiments for multirotors, estimation experiments, control experiments, to decision-making experiments.

For any inquiries, please visit <https://doc.rflysim.com/> for more information.



More tutorials for
RflySim



Scan the code for inquiries and
communication.



FeiSi RflySim Technical
Exchange Group



Thank you!