Multicopter Design and Control Practice ——A Series Experiments Based on MATLAB and Pixhawk

Lesson 03 Experimental Platform Usage

Dr. Xunhua Dai, Associate Professor,
School of Computer Science and Engineering,
Central South University, China;
Email: dai.xh@csu.edu.cn ;
https://faculty.csu.edu.cn/daixunhua

Outline

1. Experimental Platforms

2. Controller Design and Simulation Platform

3. PSP Toolbox

4. Pixhawk Hardware System

5. HIL Simulation Platform

6. Summary

Experimental Platforms

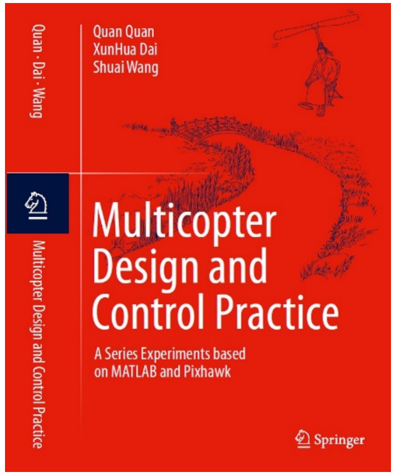
☐ Platform Composition

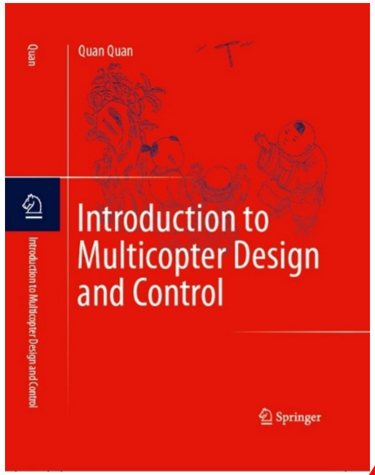
Simulink-based Controller Design and Simulation Platform

MATLAB SIMULINK

Pixhawk Autopilot System

Experimental Instruction Package





Multicopter Hardware System

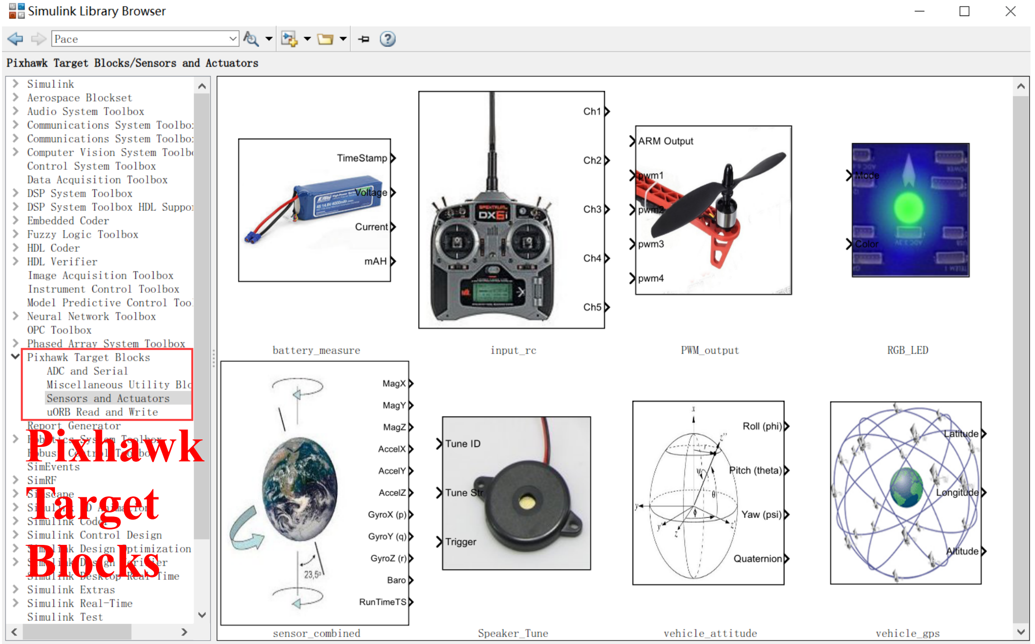
HIL Simulation Platform

Experimental Platforms

☐ Platform Composition

(1) Simulink-based Controller Design and Simulation Platform

- A high-fidelity nonlinear model
- Modular controller design
- A real-time 3D display for the flight status
- Automatic code generation and upload function



Controller design and simulation

MATLAB SIMULINK

FlightGear

Automatic code generation and firmware compiling

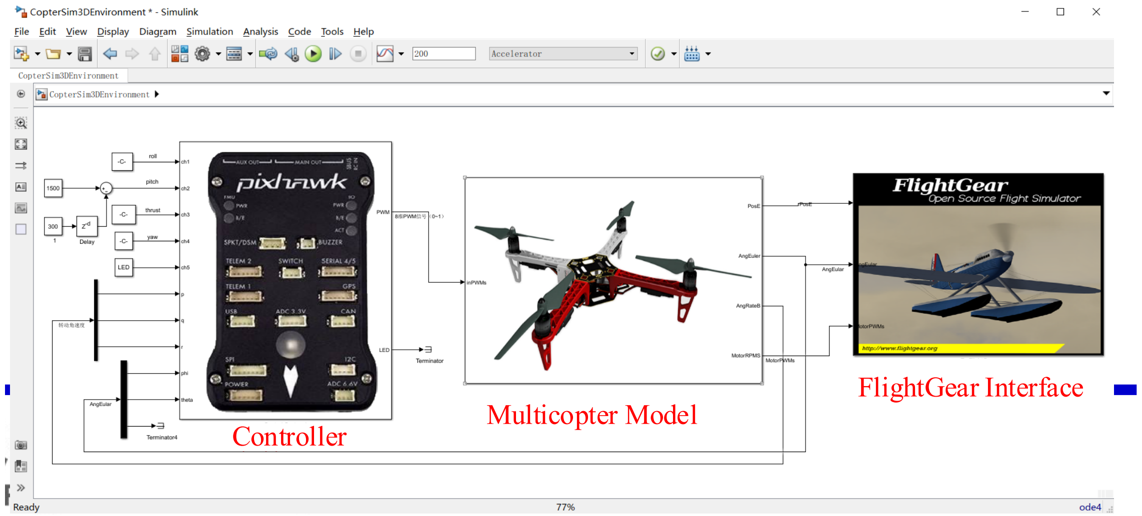
Simulink PSP toolbox

PX4 firmware source code

WSL/Msys2/ Cygwin compiling tool

Code view and modification

eclipse JUNO

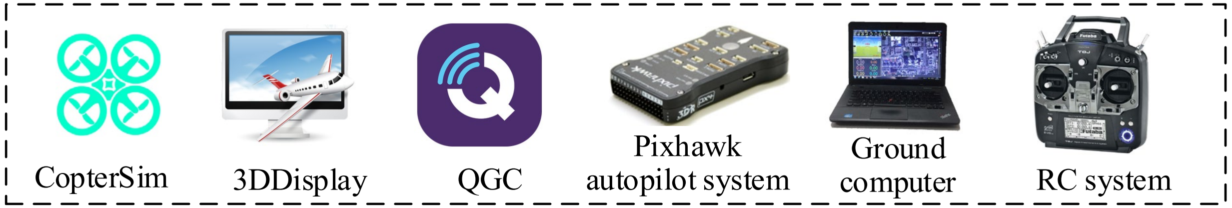


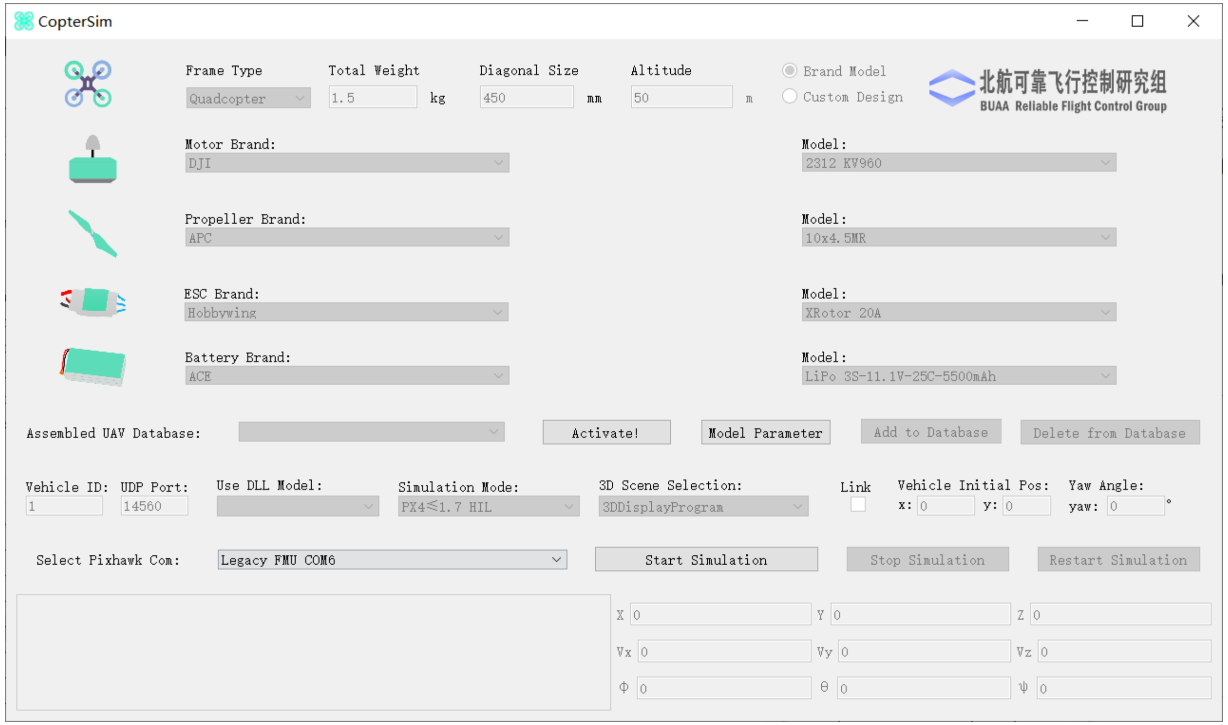
Experimental Platforms

□ Platform Composition

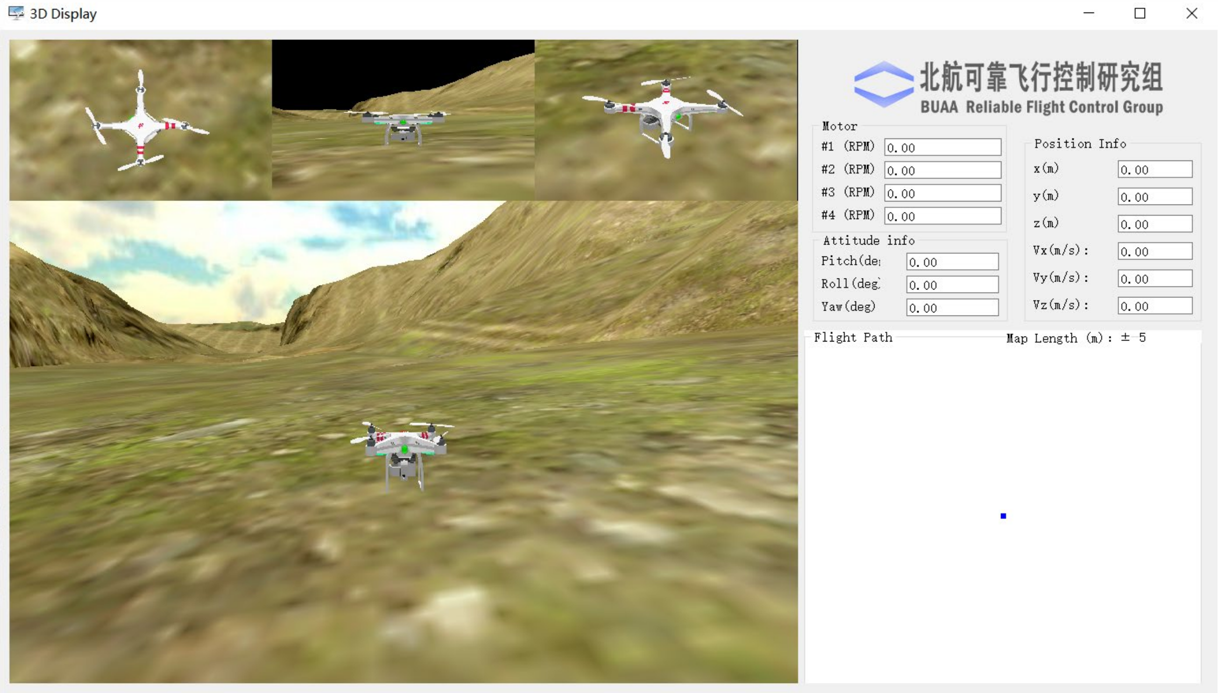
(2) HIL Simulation Platform

HIL simulation





(a) CopterSim: real-time motion simulation software



(b) 3DDisplay: 3D visual display software

北航可靠飞行控制研究组
BUAA Reliable Flight Control Group

Experimental Platforms

□ Platform Composition

(3) Pixhawk Autopilot System

1) Pixhawk Autopilot

2) RC transmitter and receiver

3) A microUSB cable

4) Ground Control Station (GCS)

Experimental Platforms

□ Platform Composition

(4) Multicopter Hardware System

1) Airframe system

2) Propulion system

3) External sensors

4) Test stand

Experimental Platforms

□ Platform Composition

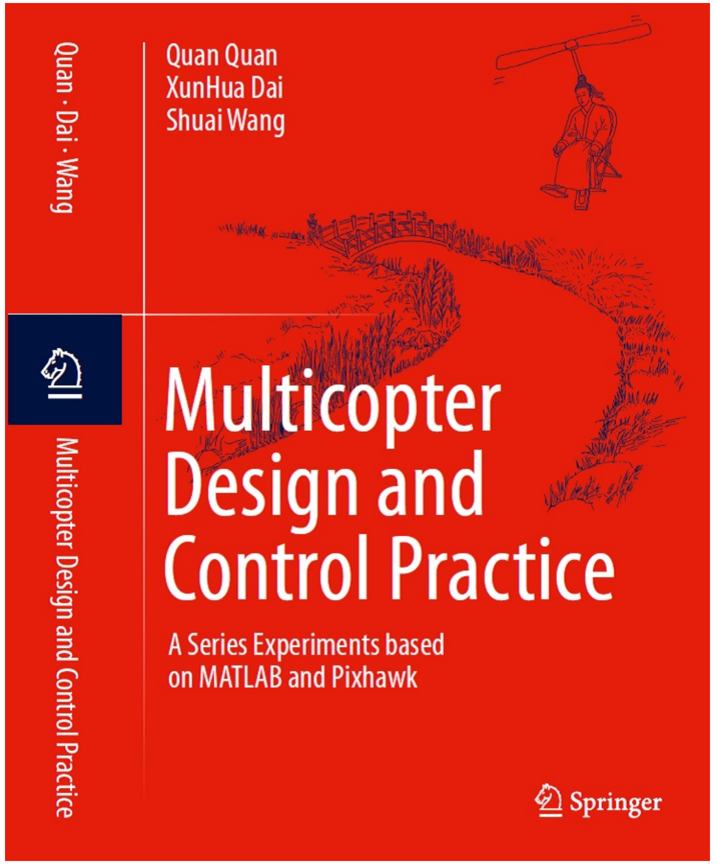
(5) Experimental Instruction Package

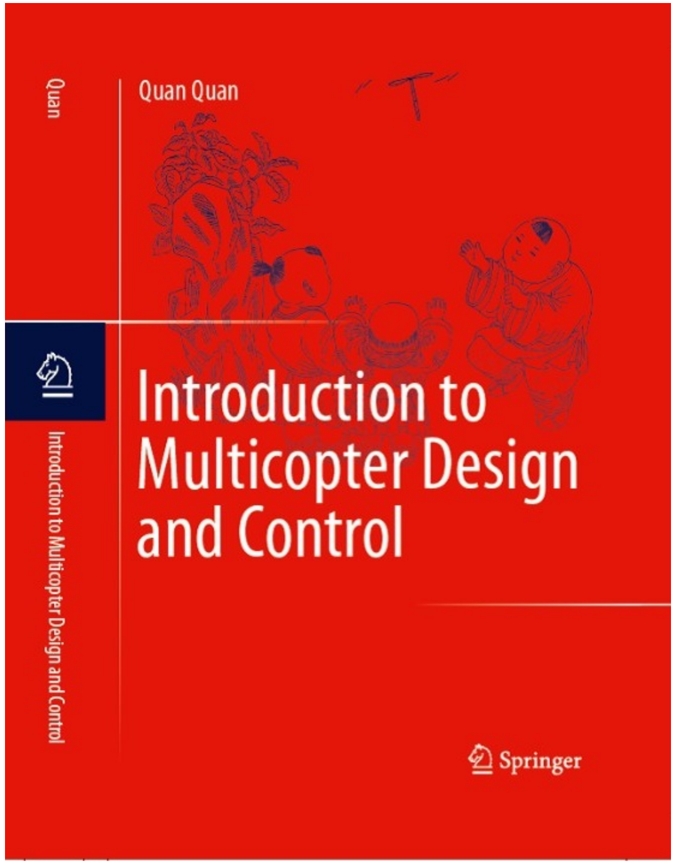
1) Textbook

2) Experiment instructor

3) Source code examples

4) Video tutorials





Experimental Platforms

□ Platform Advantage

- This experimental platform provides interfaces for multicopter controller design in MATLAB/Simulink
- Readers (beginners, students, or engineers) can develop a rapid design and verify the control algorithms by SIL simulation.
- Platform also provides a code generation function to generate Simulink controllers to C/C++ code.
- Platform also provides a HIL simulation platform for preliminary simulation tests on a Pixhawk autopilot system that may help in eliminating potential problems that may exist in flight tests.
- After all the tests are completed, indoor and outdoor flight tests can be carried out by assembling the Pixhawk autopilot onto a real multicopter hardware system. The performance of the designed controllers can be evaluated through experimental tests.

Controller Design and Simulation Platform

This book provides a high-fidelity simulation environment based on Simulink/FlightGear. The main source code file is presented in "e0\1.SoftwareSimExps\CopterSim3DEnvironment.slx".

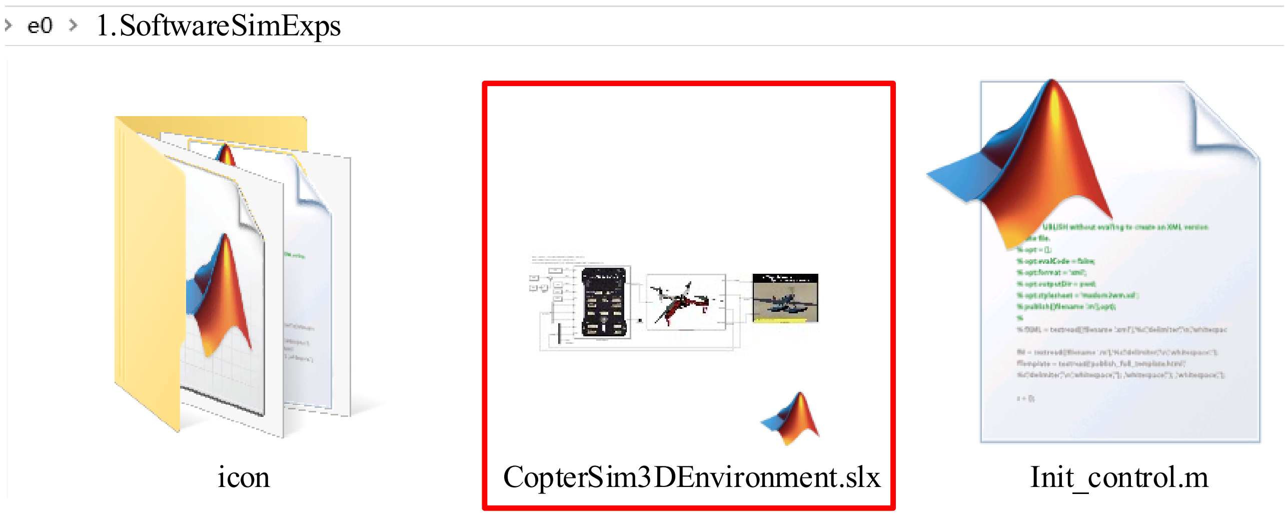


Fig. SoftwareSimExps file diagram

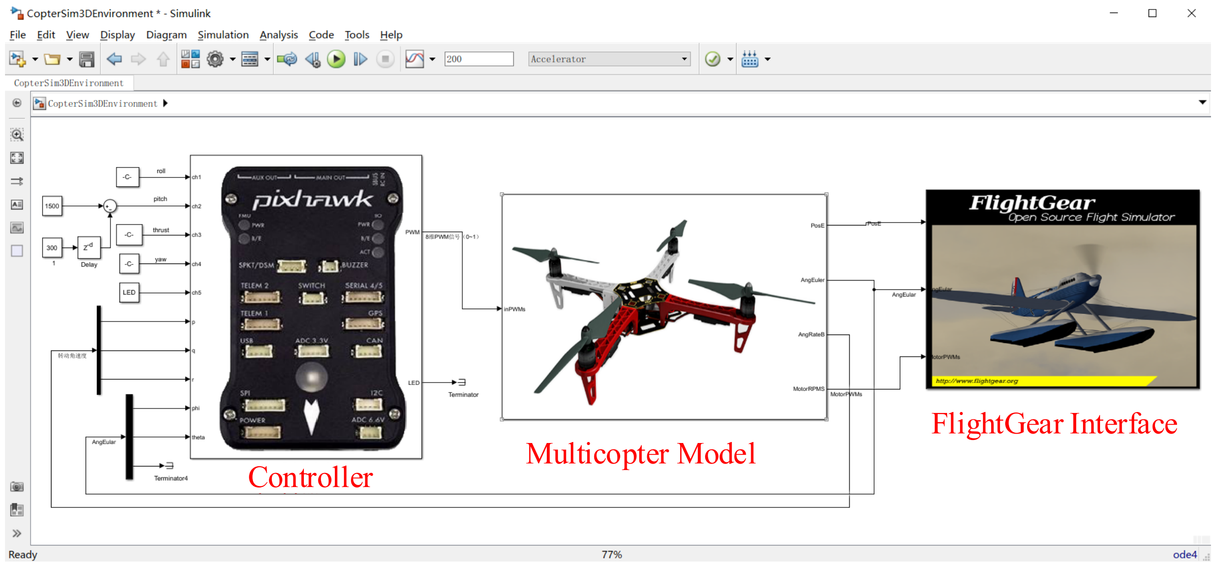
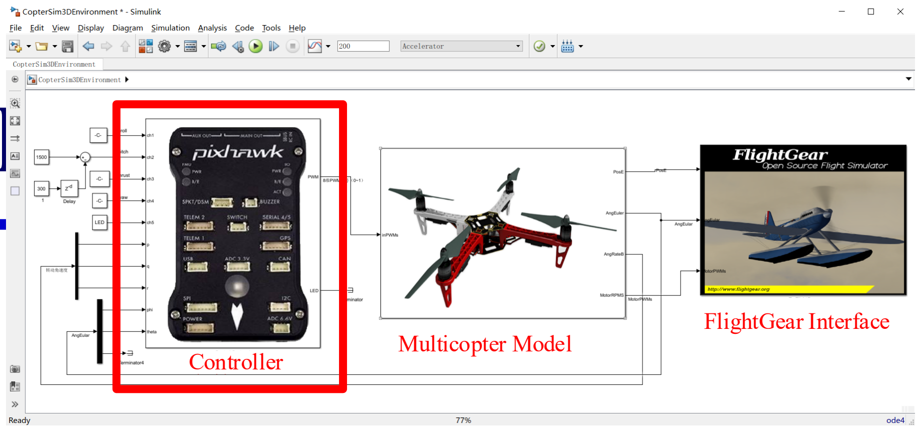


Fig. Simulink SIL simulation example

Controller Design and Simul



☐ Controller

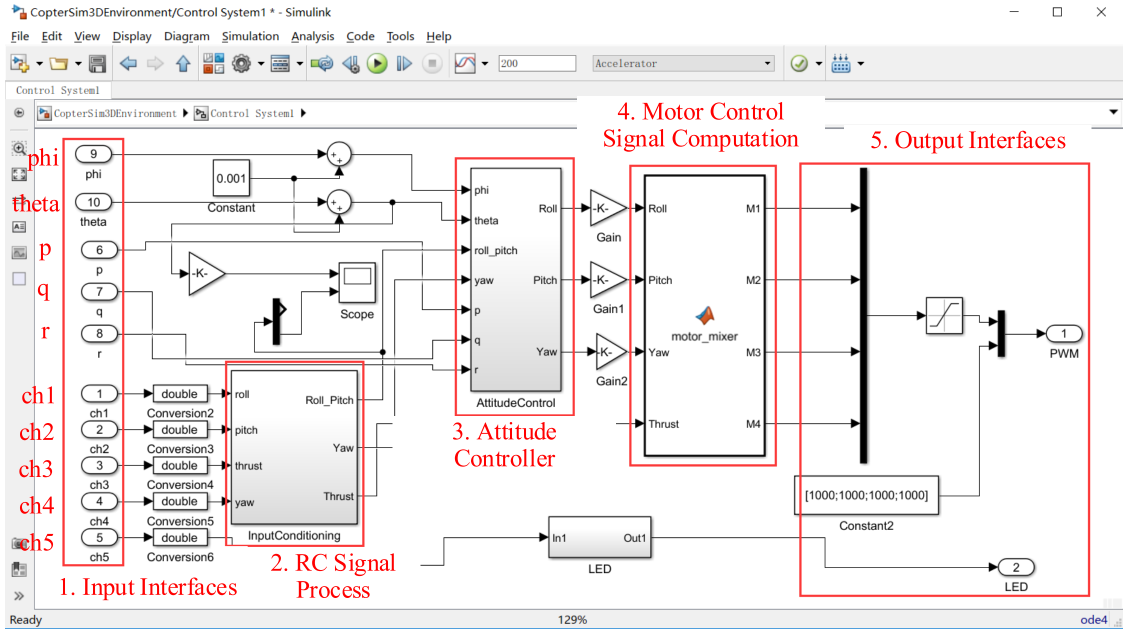


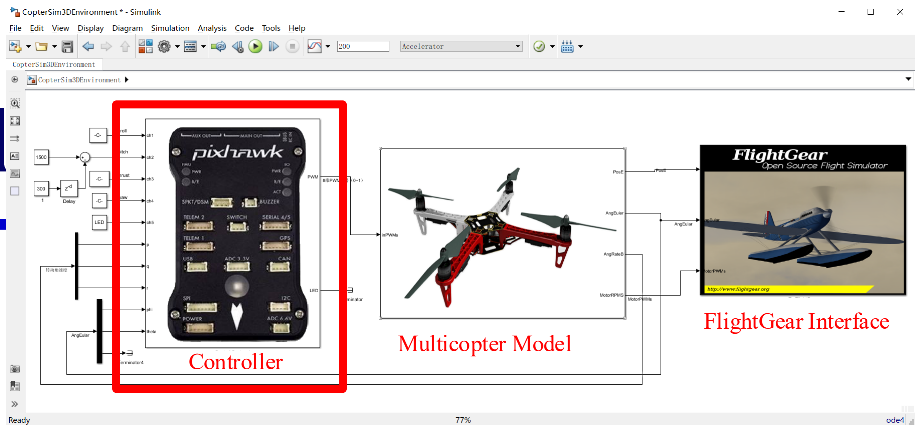
Fig. Internal structure of “Controller” subsystem

As shown in left figure, this example shows a simple attitude controller for pitch and roll angles. The controller receives the control signals from the RC transmitter and controls the multicopter to achieve the desired pitch and roll angles.

1. The “Input Interfaces” module receives the RC transmitter signals and the multicopter state estimation signals
2. The “RC Signal Process” module maps the five-channel signals of the RC transmitter to the desired roll and pitch angle values.
3. The “Attitude Controller” module computes the desired force and torque values to control the multicopter to the desired attitude.

Controller Design and Simul

□ Controller



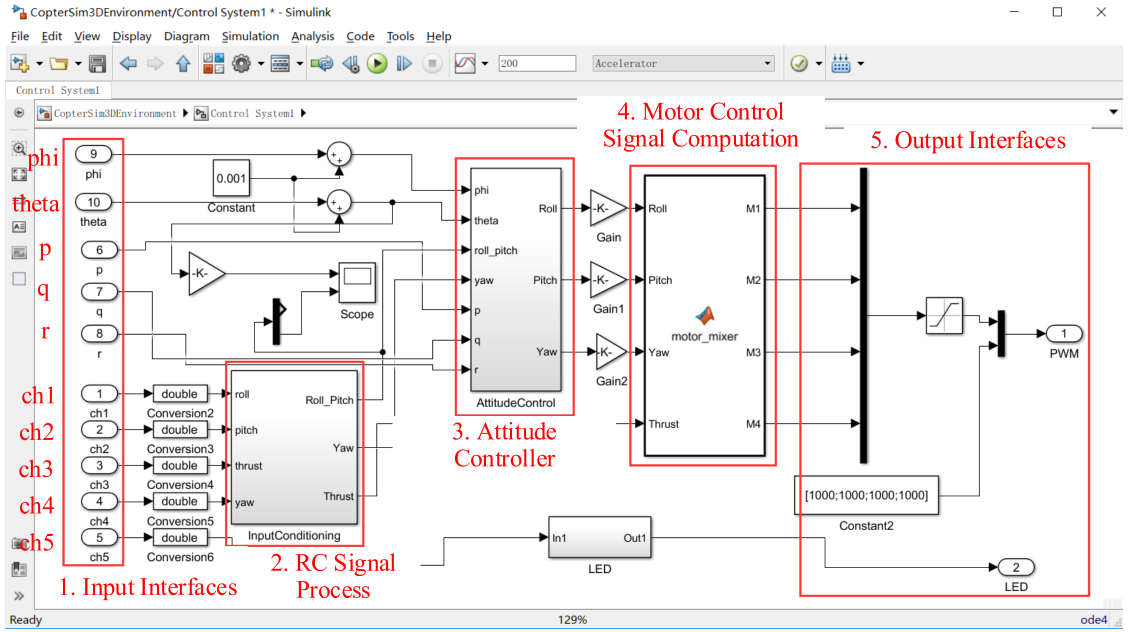
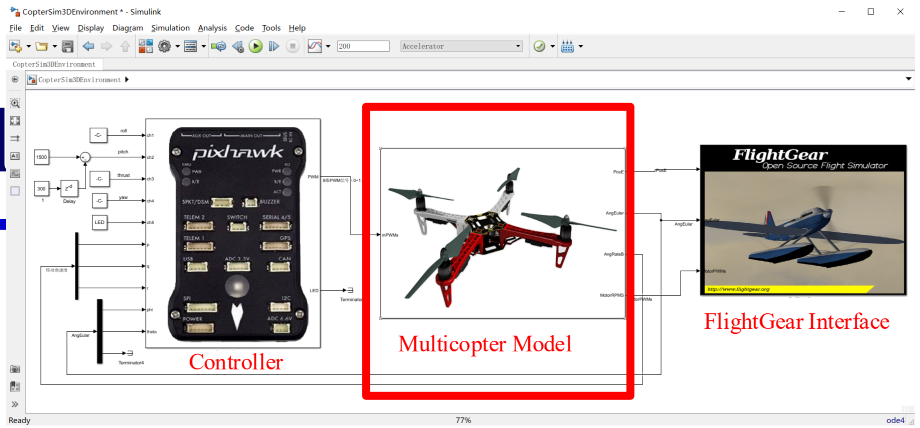


Fig. Internal structure of "Controller" subsystem

4. The "Motor Control Signal Computation" module maps the force and torque values to the control signals (ranging from 1000 to 2000) for the four motors
5. The "Output Interfaces" module fills the remaining 4-dimensional control signals and generates an 8-dimensional PWM signal (there are eight PWM output ports on Pixhawk) ranging from 1000 to 2000.

Controller Design and Simul

◻ Multicopter Model



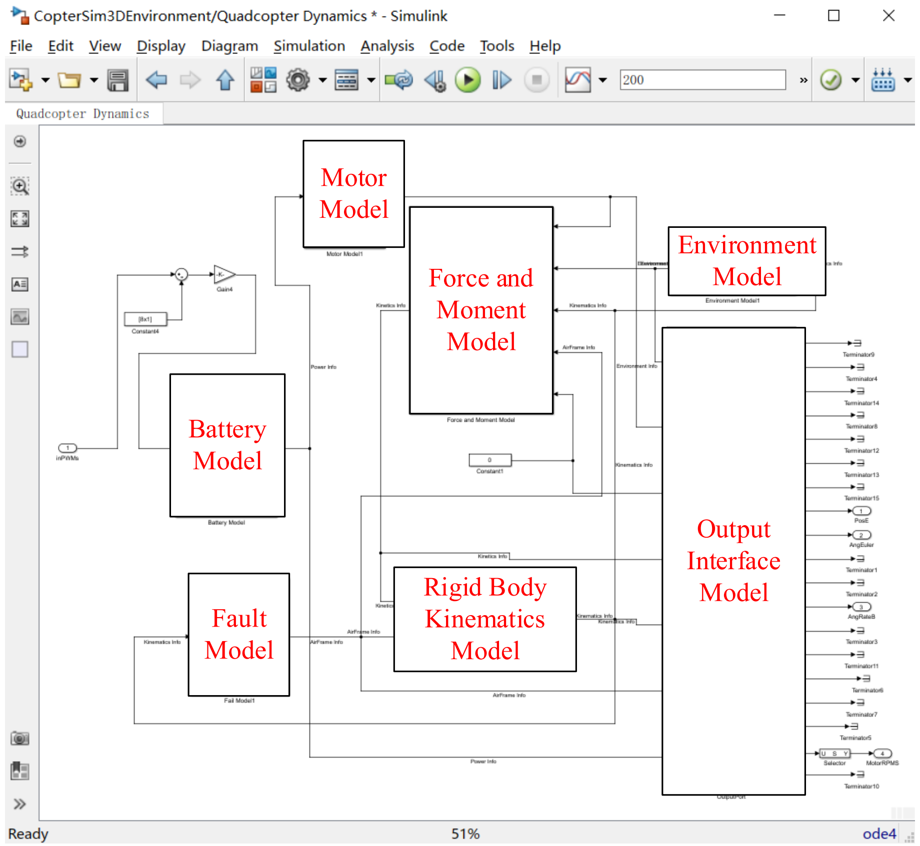


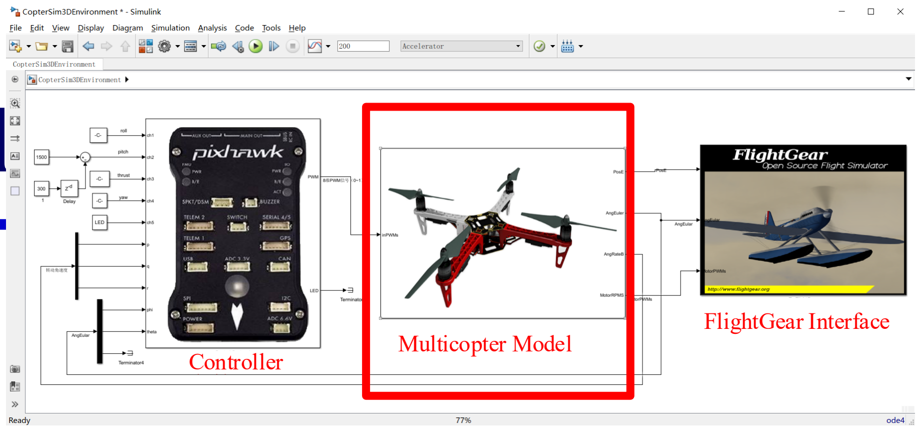
Fig. Internal structure of “Multicopter Model” subsystem

1) Inputs： motor PWM controls

2) Outputs: flight state and sensor signals

- “Motor Model” module: it simulates the motor dynamics.
- “Force and Moment Model” module: it simulates all external forces and moments acting on the body, such as the propeller thrust, fuselage aerodynamics, gravity, and ground supporting force.
- “Rigid Body Kinematics Model” module: it calculates the vehicle kinematics of the multicopter, such as speed, position, and attitude.

Controller Design and Simul



☐ Multicopter Model

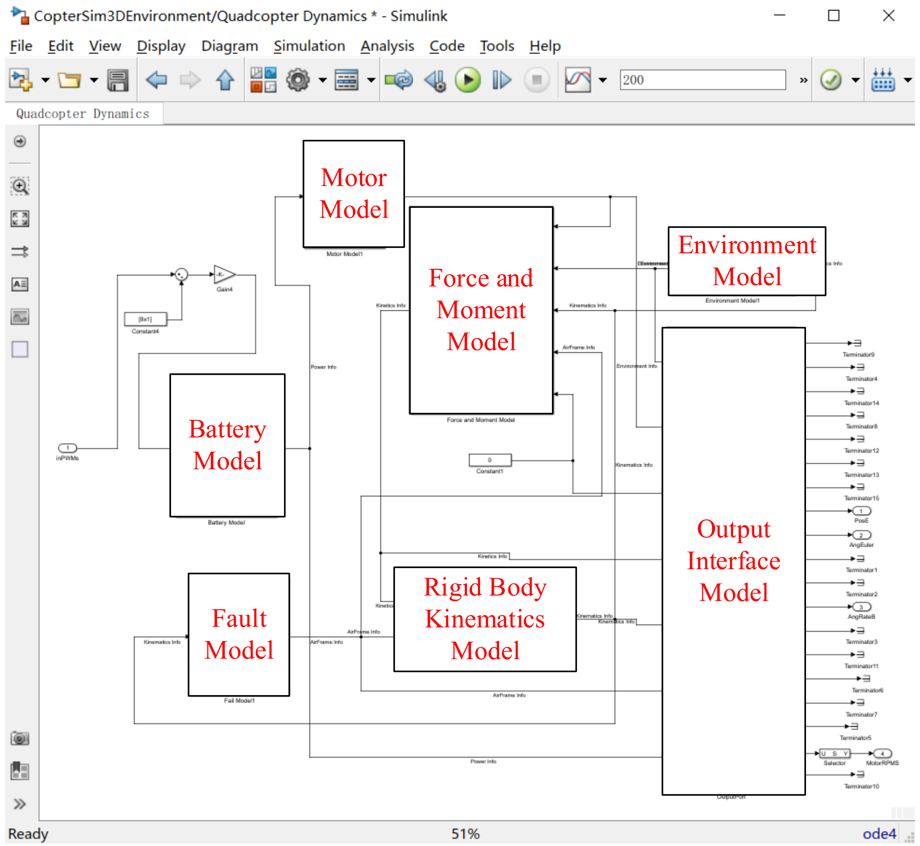
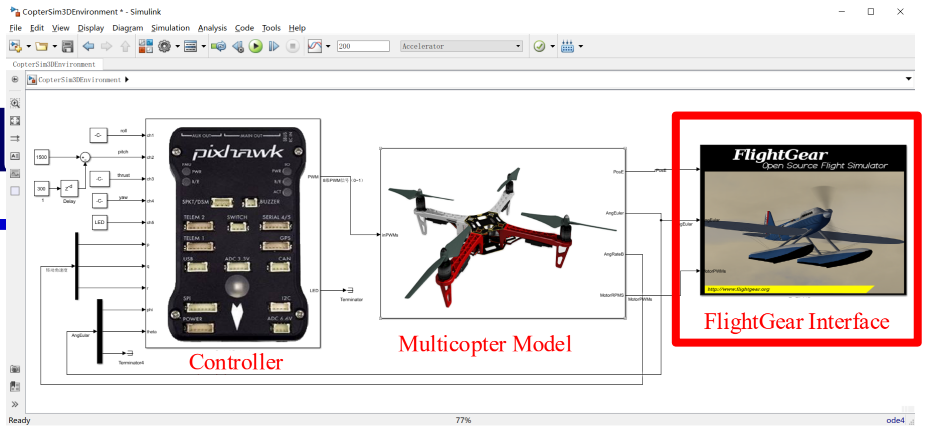


Fig. Internal structure of "Multicopter Model" subsystem

- "Environmental Model" module: it calculates the environmental data, such as gravitational acceleration, air density, wind disturbances, and geomagnetic field
- "Fault Model" module: it is mainly used to inject model uncertainties (related to mass and moment of inertia) as well as faults.
- "Battery Model" module: it simulates the discharge process of the battery.
- "Output Interface Model" module: it packs the output signals in the desired format.

Controller Design and Simul



☐ FlightGear Interface

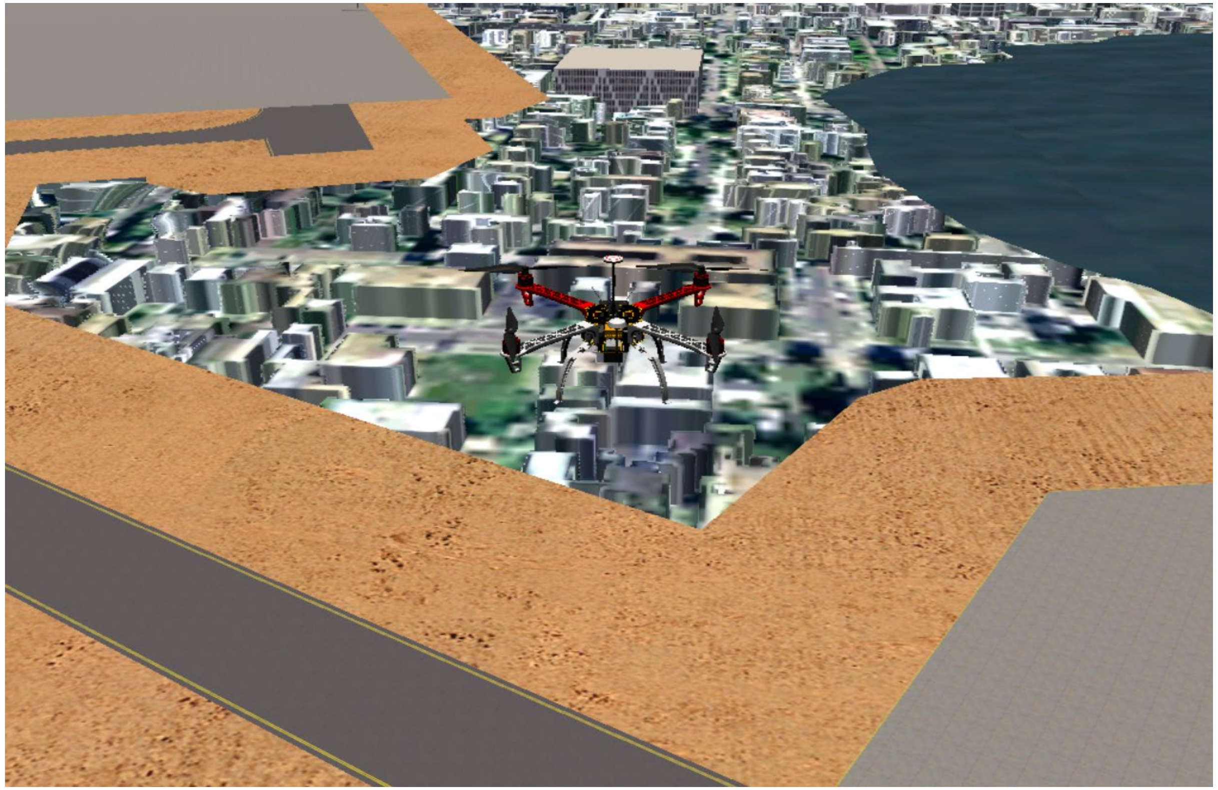
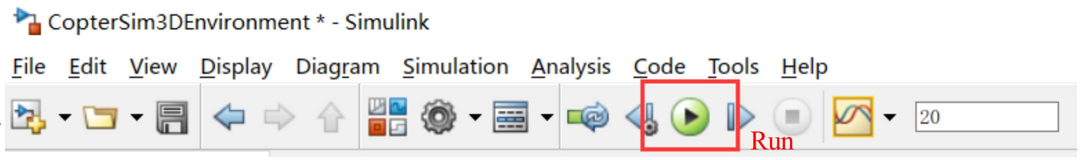


Fig. A quadcopter in FlightGear

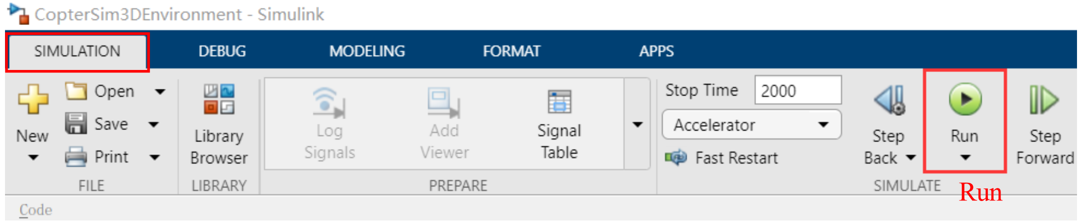
1) Function

"FlightGear Interface" subsystem has three input ports representing the multicopter position, Euler angles, and motor PWM signals, respectively. This subsystem sends multicopter flight state information to FlightGear to observe the flight attitude and trajectory of the multicopter in a 3D scene.

Controller Design and S



(a) Simulink "Run" button on MATLAB 2017b-2019a



(b) Simulink "Run" button on MATLAB 2019b and above

□ FlightGear Interface

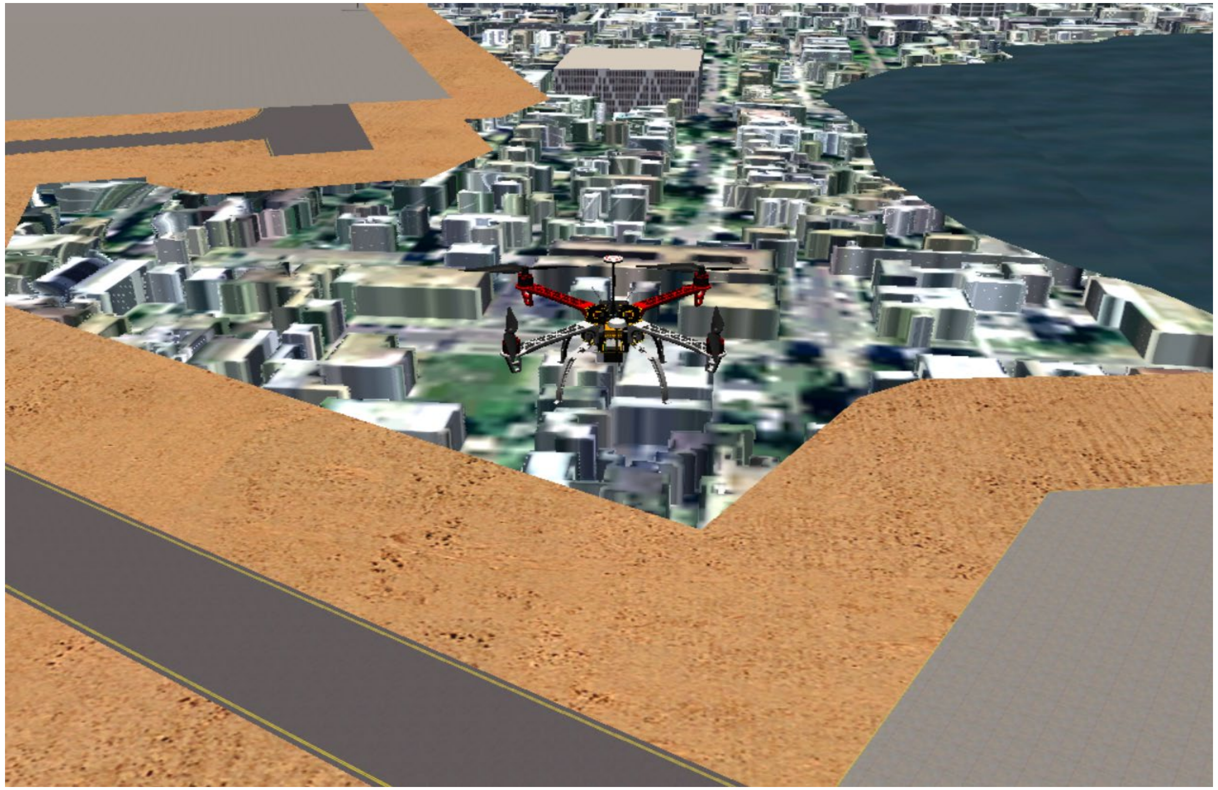


Fig. A quadcopter in FlightGear

2) Effect

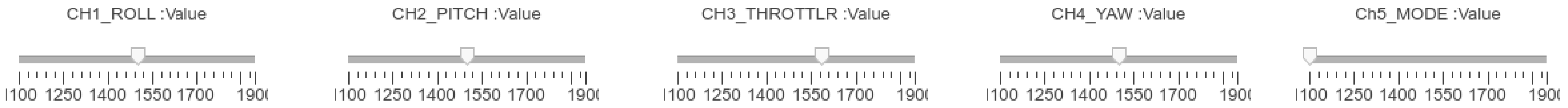
Note: if FlightGear cannot display properly, you can open 3DDisplay to observe the flight effect

1. Double-click the FlightGear-F450 shortcut on the desktop to open FlightGear;

2. Click the "Run" button

on the Simulink toolbar to run the "CopterSim3DEnvironment.slx" file;

3. Then, the multicopter takes off vertically from the ground. You can slide the slider modules in Simulink to simulate control the roll, pitch, yaw, throttle channels of the controller with an RC system.

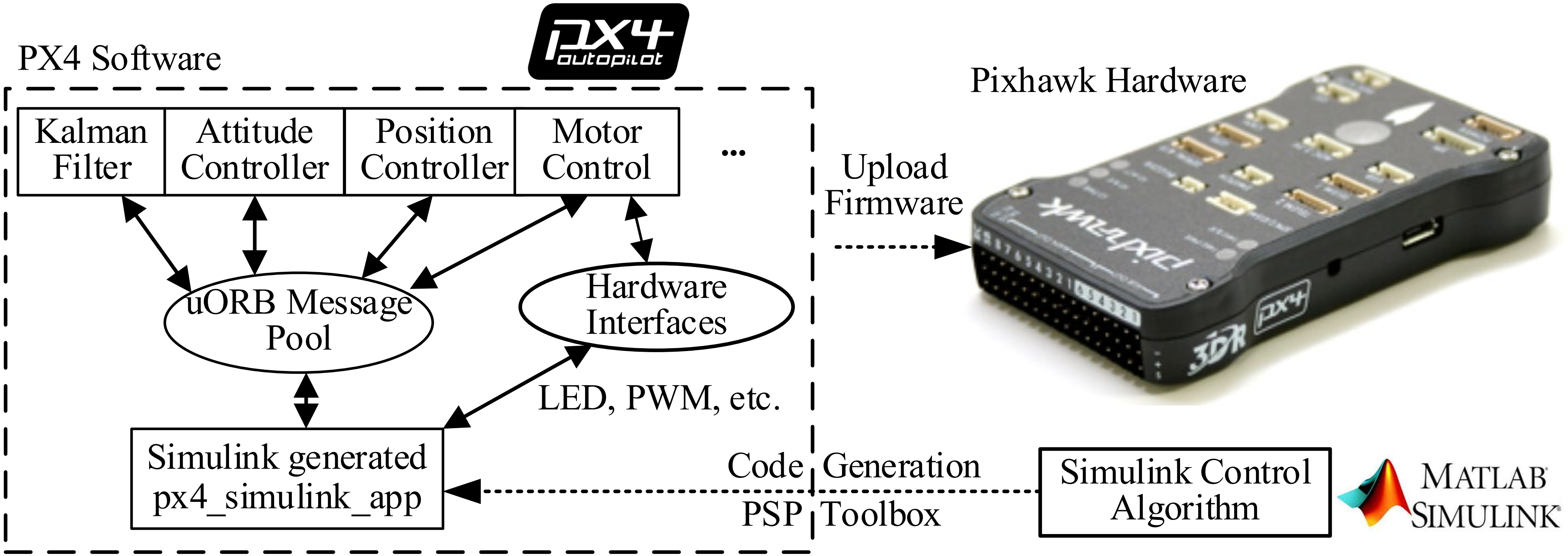


PSP Toolbox

Pixhawk Pilot Support Package (PSP) Toolbox s a Simulink toolbox officially released by Mathworks.

This toolbox can use the Embedded Coder in Simulink to automatically compile and deploy Simulink model autopilot algorithms to Pixhawk hardware systems.

The figure below shows the relationship among the PSP toolbox, the PX4 software, and the Pixhawk hardware.



PSP Toolbox

☐ Main Functions

1. The toolbox can simulate and test different multicopter models and flight control algorithms in Simulink and then automatically deploy the algorithms to the Pixhawk autopilot.

2. The toolbox provides many practical examples, including LED control, RC data process, and attitude controller.

3. The toolbox provides many interface modules to access the Pixhawk hardware and software components.

4. It automatically records flight data from sensors, actuators, and controllers deployed by themselves.

5. It can subscribe and publish uORB topic messages. All messages in the PX4 autopilot software are temporarily stored in a uORB message pool. The subscription function can read topics of interest from the message pool, and the publishing function can publish specific topics to the message pool for other modules.

PSP Toolbox

◻ Relationship with Pixhawk system

1. The structure of a Pixhawk autopilot system includes two parts: the Pixhawk hardware (similar to the computer hardware) and the PX4 software (similar to operating system and applications on a computer).

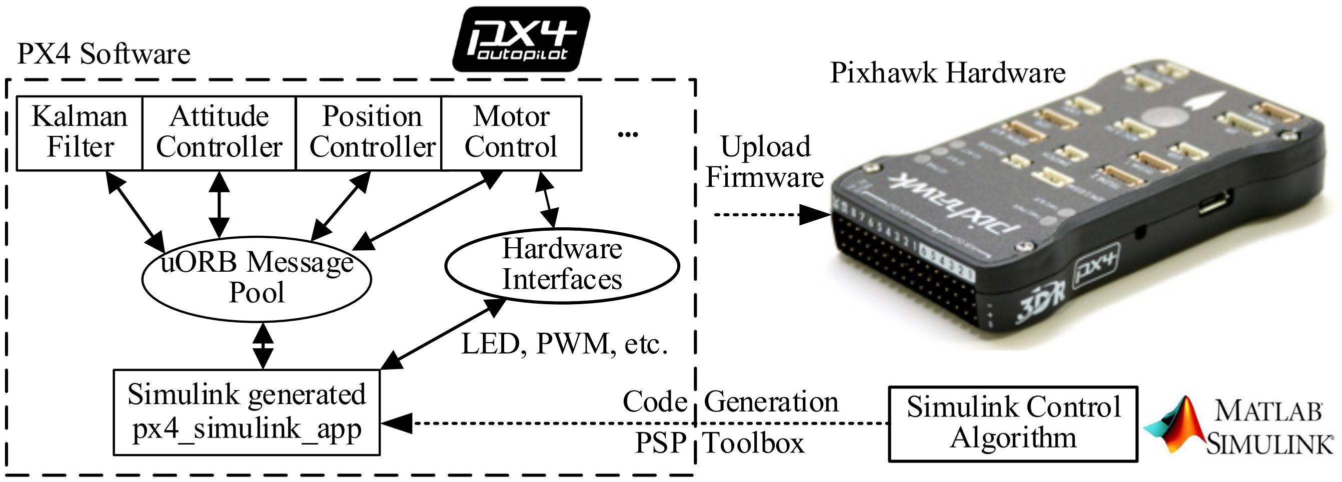
2. The PX4 software system can be further divided into several small modules, which run independently in parallel multi-thread. Each module exchanges data with other modules through uORB messages.

3. After the algorithm code is generated by the PSP toolbox, it is embedded into the PX4 software system. This will not affect the operation of the native control modules in the PX4 software. Instead, a new independent module named “px4_simulink_app” will be created to run in parallel with other modules.

5. The native modules in the PX4 software may assess the same hardware outputs as the generated “px4_simulink_app” module, which may cause read and write conflicts. Therefore, in the one-key installation script, the hardware output accessing codes of the PX4 native modules have to be blocked. This will ensure that only the “px4_simulink_app” module can send motor control signals.

PSP Toolbox

□ Process to generate code



(a) The PSP toolbox generates the C/C++ code from the control algorithm designed in Simulink.

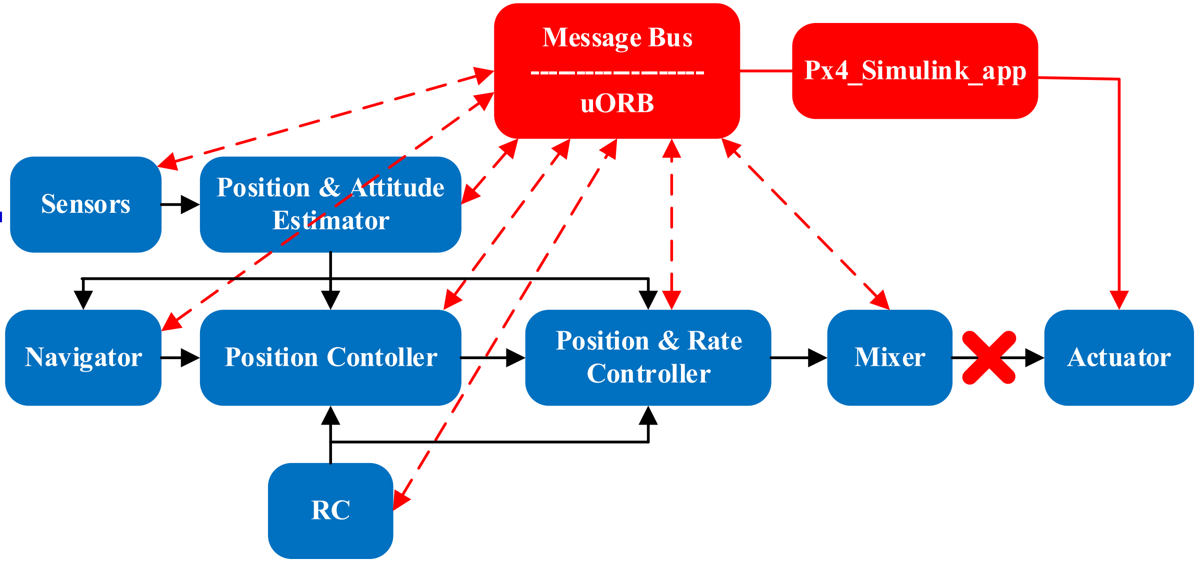
(b) The obtained algorithm code is imported into the PX4 source code to generate a "px4_simulink_app" independent of other modules.

(c) The PSP toolbox calls the compiling toolchain (Win10WSL, Msys2, or Cygwin) to compile all the code into a ".px4" PX4 firmware file (similar to a software installation package).

(d) Upload the obtained firmware file to the Pixhawk hardware; then, the Pixhawk autopilot can execute the PX4 software with the generated algorithm code.

PSP Toolbox

☐ Advanced extension features



Note: Another feasible way is to block the module in the startup script "Firmware\ROMFS\px4fmu_common\init.d\rcS"

The generated Simulink code can also be used to replace some of the native modules (sensors, filters, attitude controllers, etc.) of the PX4 software. However, the PX4 software code needs to be manually modified to block the output interface of the corresponding native module. For example, if readers want to use Simulink to replace the attitude filter module (input sensor data, output filtered attitude data) of the PX4 software, they should manually block uORB publishing code of the "vehicle_attitude" msg in the "Position & Attitude Estimator" module. The detailed steps are described next.

(a) Open the "Firmware\src\modules\ekf2\ekf2_main.cpp" file (corresponding to the code of the extended Kalman filter module).

(b) Search the text "_att_pub", and comment out the lines containing "publish" and "att", and replace them with "UNUSED(att);". The UNUSED is used to suppress warning of "variable not used" of compiler.

(c) Create a Simulink controller to publish "vehicle_attitude" message with "uORB Write" PSP module.

PSP Toolbox

□ Simulink Pixhawk Target Blocks

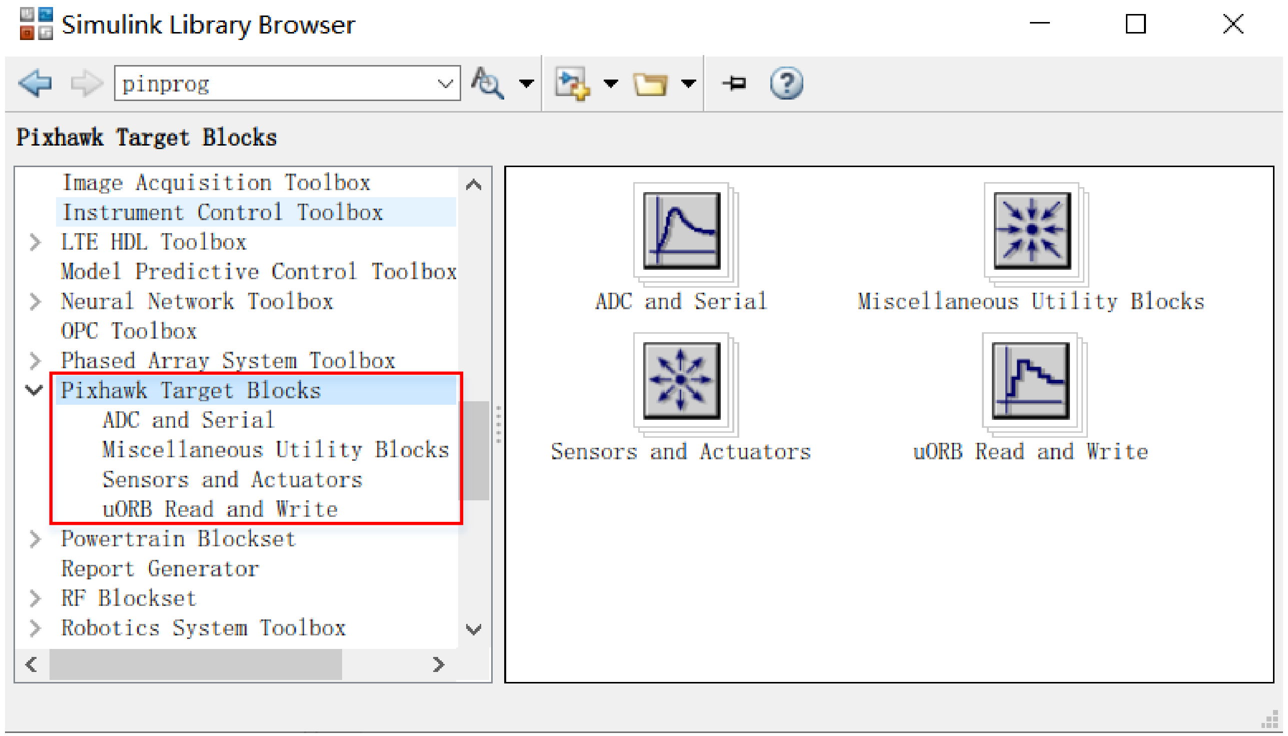


Fig. Simulink Pixhawk Target Blocks of PSP toolbox

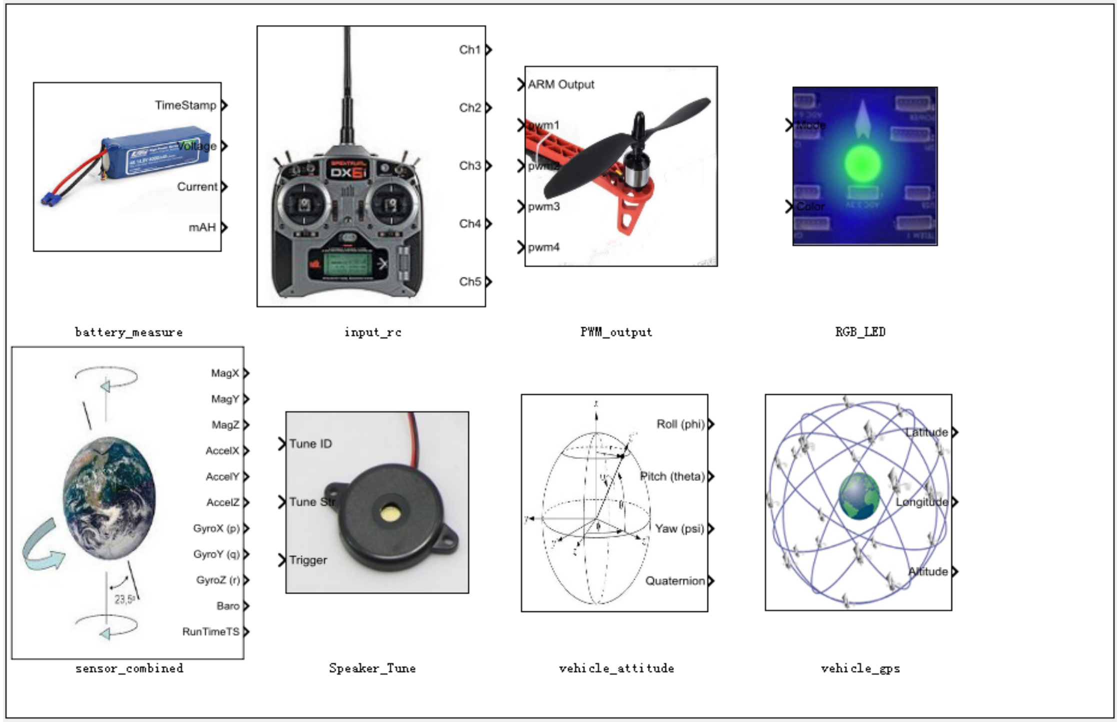


Fig. interfaces of Sensors and Actuators

PSP Toolbox

□ Simulink Pixhawk Target Blocks

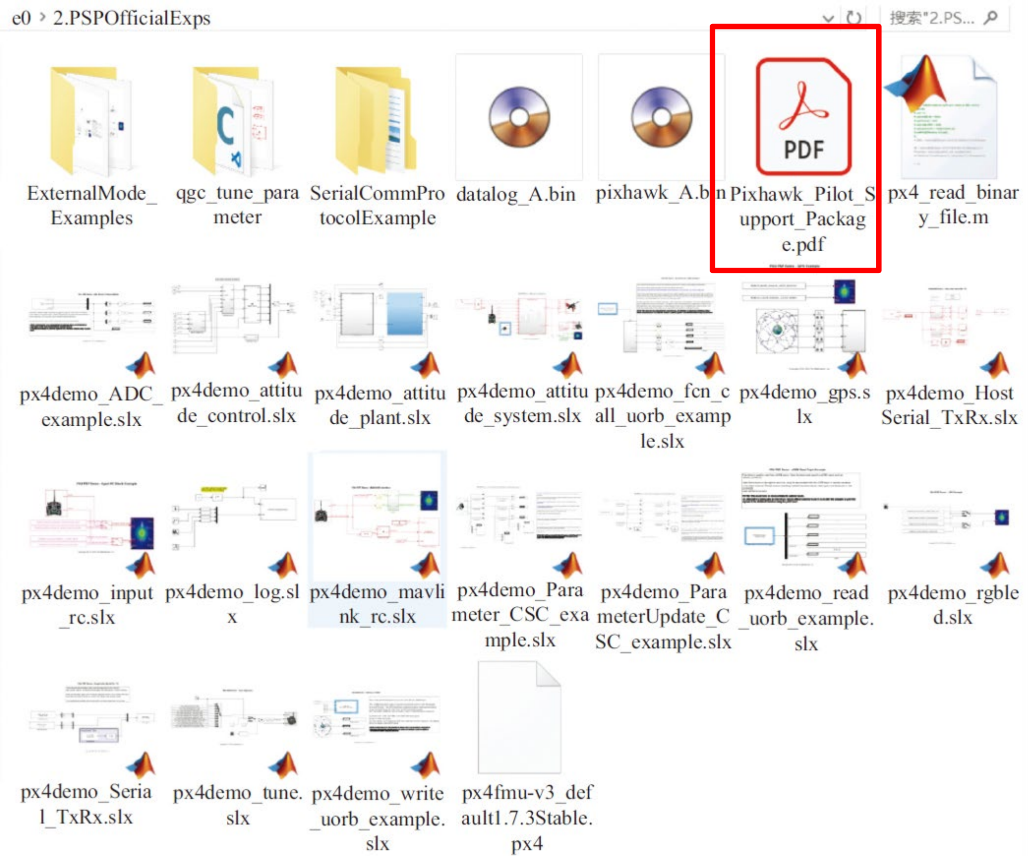
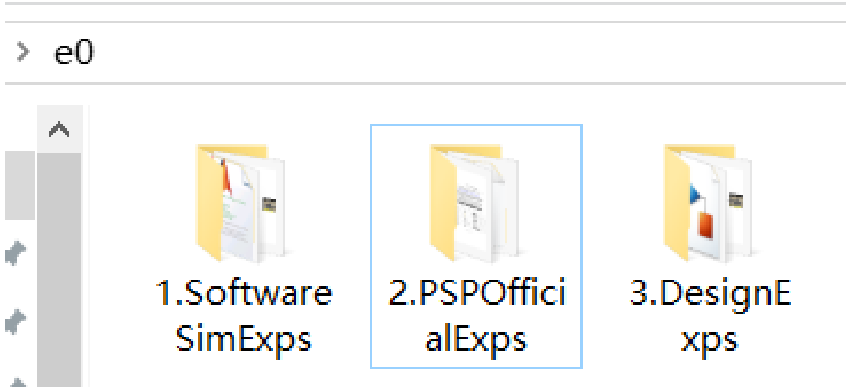


Fig. Official examples and manuals for PSP toolbox



the PSP toolbox also provides many examples (see folder “e0\2.PSPOfficialExps”) with an official manual (see document “e0\2.PSPOfficialExps\Pixhawk_Pilot_Support_Package.pdf” for details) for readers to be quickly familiar with functions and usage methods of PSP toolbox.

☐ Modules in PSP Toolbox

PWM output module: it is used to send PWM signals to PX4IO ports to control the motor. The PWM update frequency and the number of output channels can be configured in the setting box.

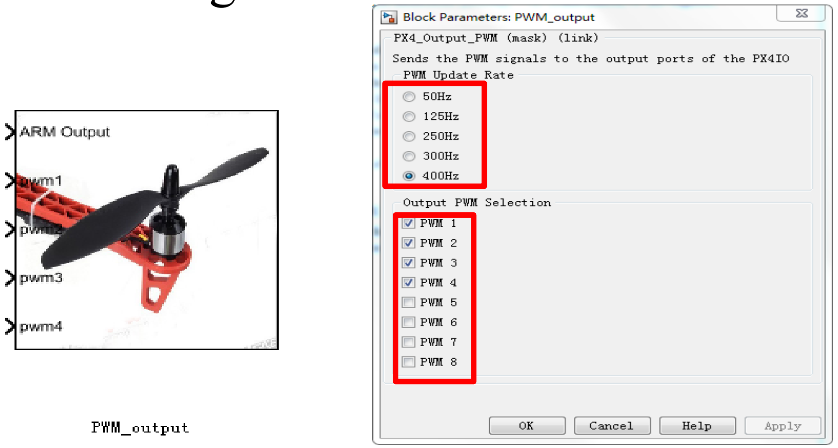


Fig. PWM output module and its parameter

RC input module: It is convenient to select RC channels and other information to be used by Simulink. The definition and application of each option can be viewed by clicking the "help" button of the box or by consulting the official PDF document. The PSP toolbox also provides an example (see file "e0\2.PSPOfficialExps\px4demo_input_rc.slx") to show how to use this module.

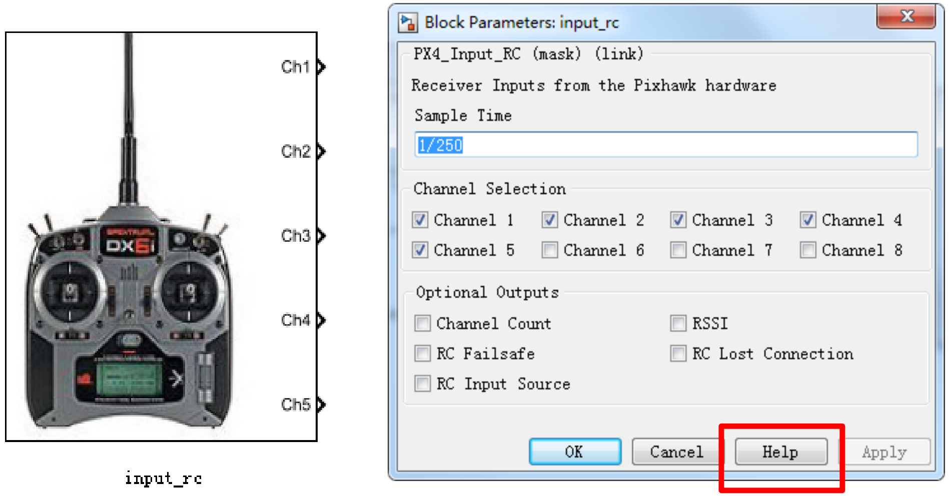


Fig. RC input module and its parameter setting

□ Modules in PSP Toolbox

Buzzer module: it is used when the buzzer is required to make a warning sound. There is an example (see file “PX4PSP\examples\px4demo_tune.slx”) for detailed information.

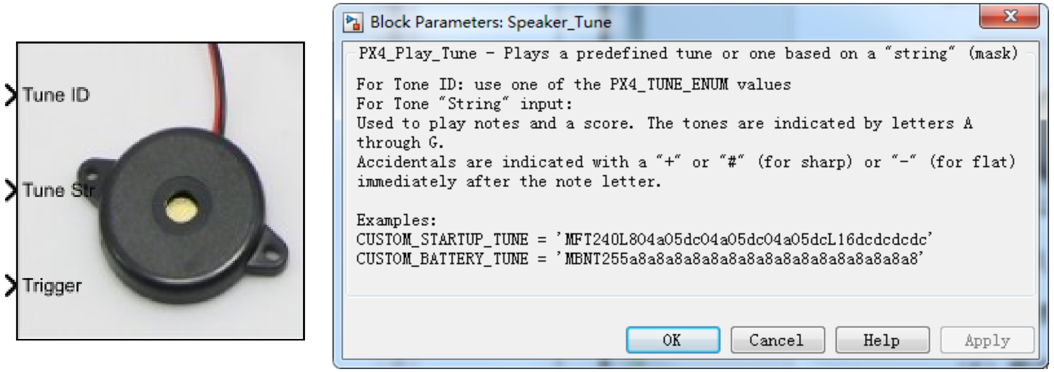


Fig. Buzzer module and its parameter setting box

RGB_LED module: this module can control the blink mode and color of the LED on Pixhawk. The module receives two inputs, namely “Mode” and “Color” representing the mode and color of the LED. The PSP toolbox provides an example (see file “e0\2.PSPOfficialExps\px4demo_rgbled.slx”) to study this module.

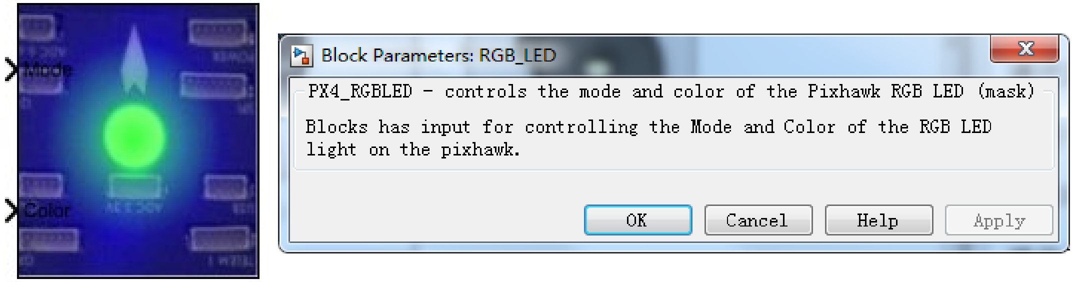


Fig. LED light module and its parameter setting box

□ Modules in PSP Toolbox

Sensor combination module: This module can access the sensor data available in the Pixhawk autopilot, which can then be used for controller design in Simulink. Available sensor data include magnetometers, accelerometers, gyroscopes, barometers, and timestamps. The PSP toolbox also provides an example (see file "e0\2.PSPOfficialExps\px4demo_attitude_system.slx") to study this module.

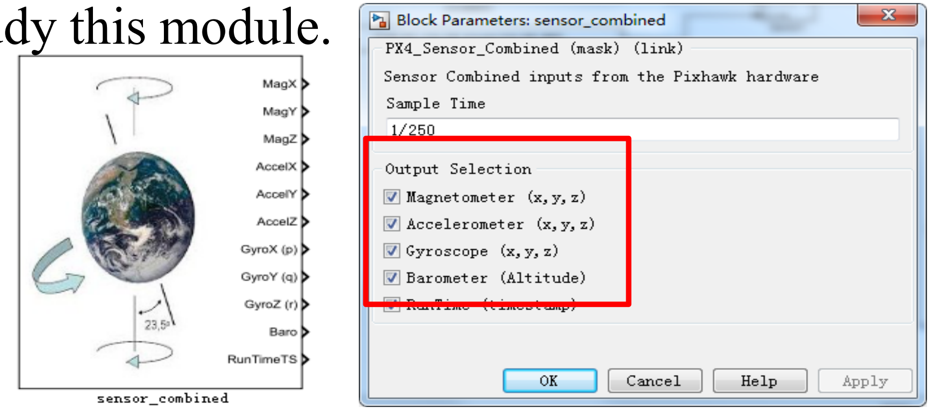


Fig. Sensor combination module and its parameter setting

Attitude data module: it provides an interface to access the attitude estimate (Euler angles and quaternion). The PSP toolbox also provides an example (see file "e0\2.PSPOfficialExps\px4demo_attitude_system.slx") to study this module.

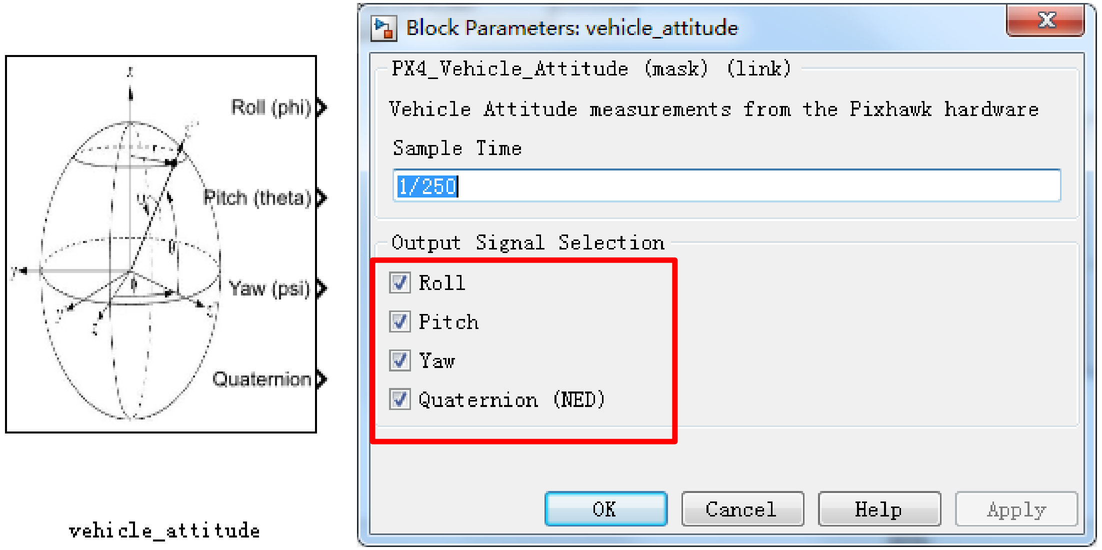


Fig. Attitude data module and its parameter setting

PSP Toolbox

The PSP toolbox also provides an example (see file “e0\2.PSPOfficialExps\px4demo_gps.slx”) to study this module.

☐ Modules in PSP Toolbox

GPS data module: it can be used to access the Pixhawk GPS data, which are achieved by subscribing to the uORB topic “vehicle_gps”. Therefore, in practical operation, it is necessary to ensure that the GPS module is inserted into the Pixhawk hardware and then works.

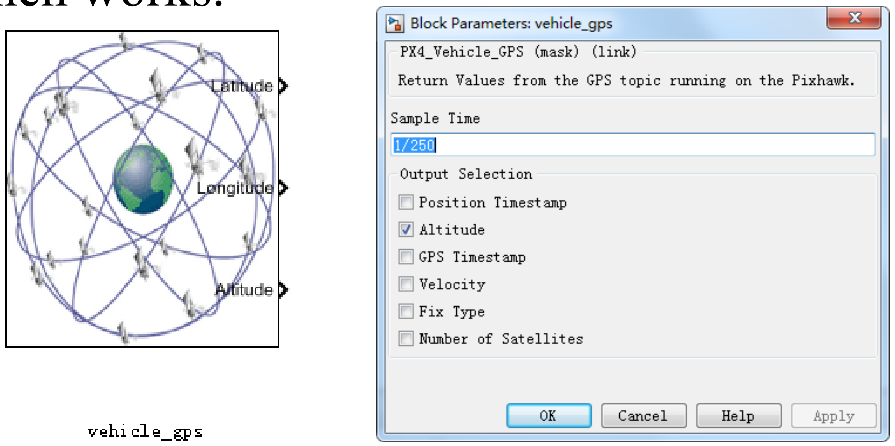


Fig. GPS data module and its parameter setting

Battery data module: it can be used to obtain the real-time status of the battery. It is implemented by subscribing to the uORB topic “battery_status”. Therefore, in practical operation, it is necessary to ensure that the power module is inserted into the Pixhawk hardware and then works correctly.

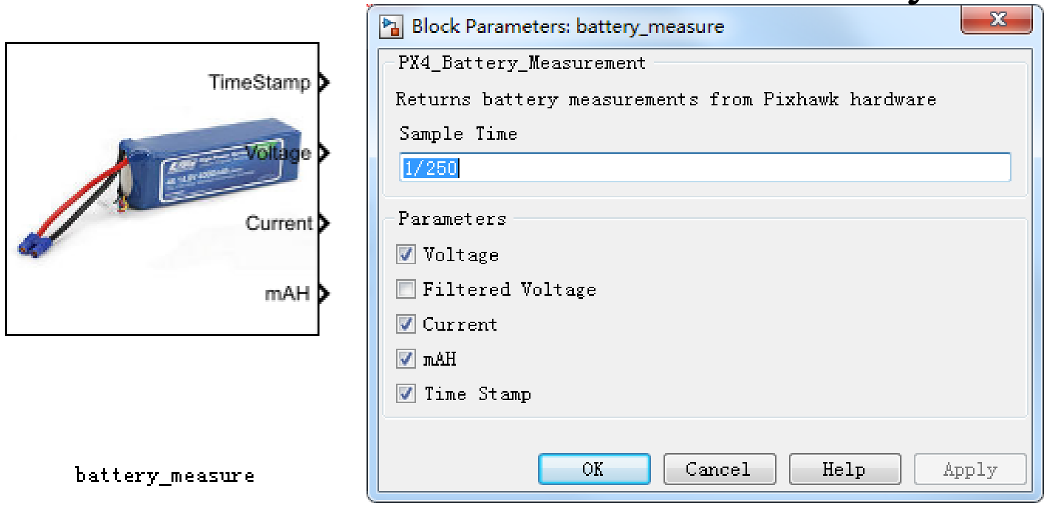


Fig. Battery data module and its parameter setting

PSP Toolbox

In fact, almost all modules in PSP toolbox can be released by uORB module. We can use uORB module to access and publish all data in Pixhawk/PX4, which can help us achieve more complex functions. Please see https://dev.px4.io/master/en/middleware/uorb.html

☐ Modules in PSP Toolbox

uORB modules. These modules, presented in left figure, are used to read or write uORB messages from the PX4 autopilot software. All the uORB message topics supported by the PX4 autopilot are listed in the directory “Firmware\msg” of the software package installation directory (the default directory is “C:\PX4PSP”). Double-click the “uORB Write” module in the lower-left figure, then the obtained parameter setting box of the “uORB Write” module is presented in lower-right figure where the uORB topic name and the message variables to be sent can be configured.

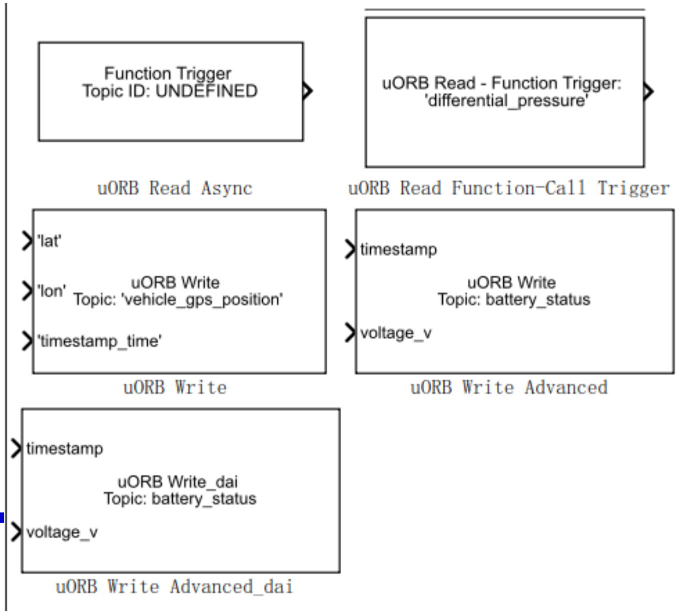


Fig. uORB read & write modules

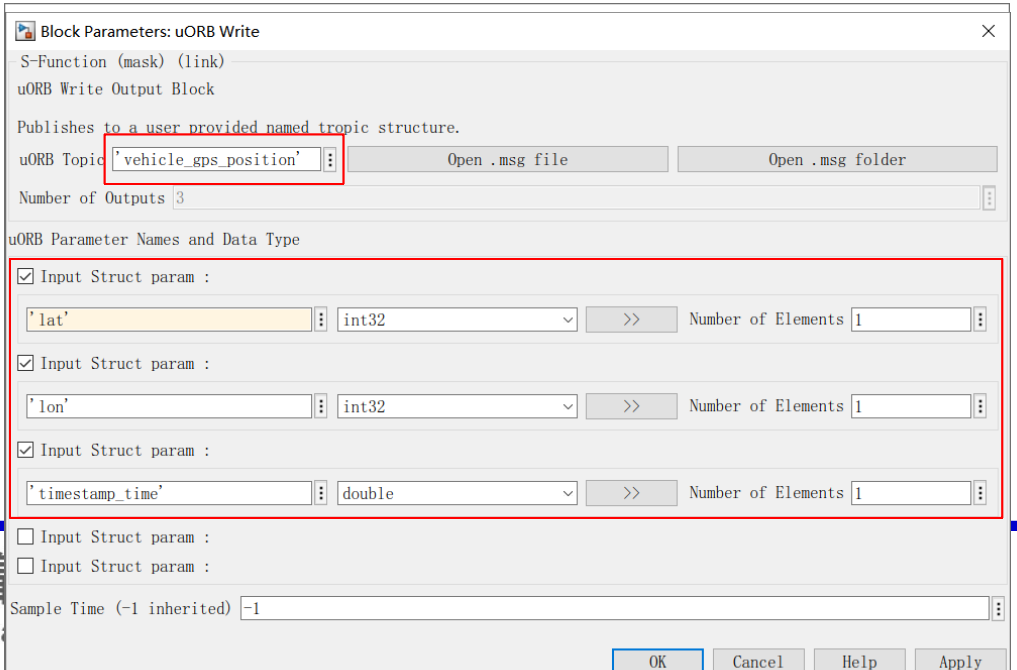
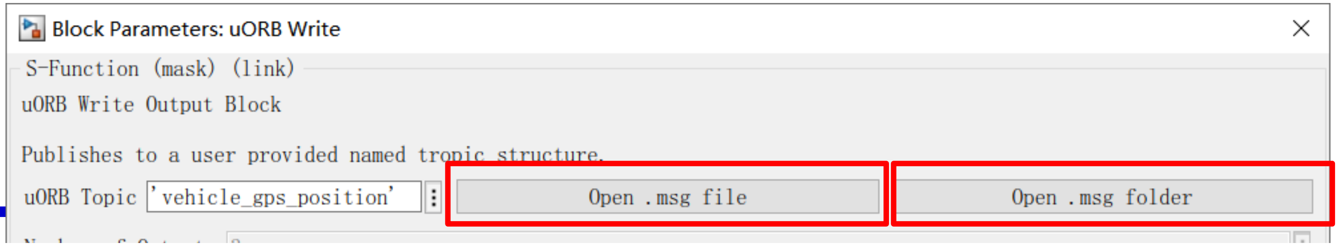


Fig. “uORB Write” module parameter setting box



Modules in PSP Toolbox

Clicking the "Open .msg file" button in upper-right figure yields the content of the select ".msg" file (see the lower-left figure), and clicking the "Open .msg folder" button yields the list of all supported uORB messages (See the lower-right figure).

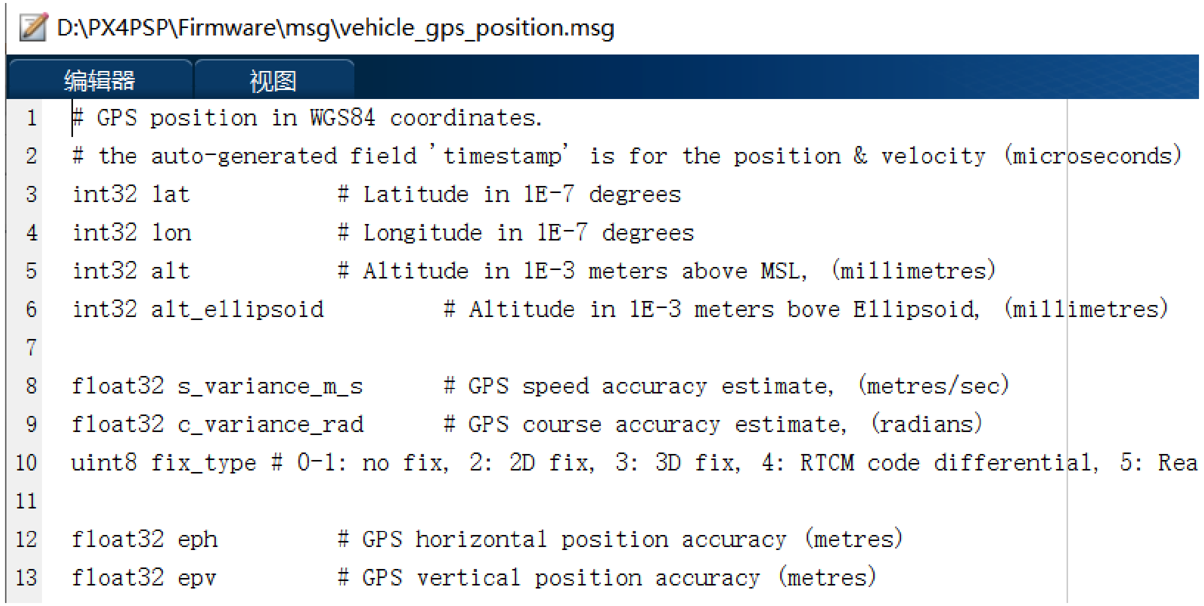


Fig. uORB message file

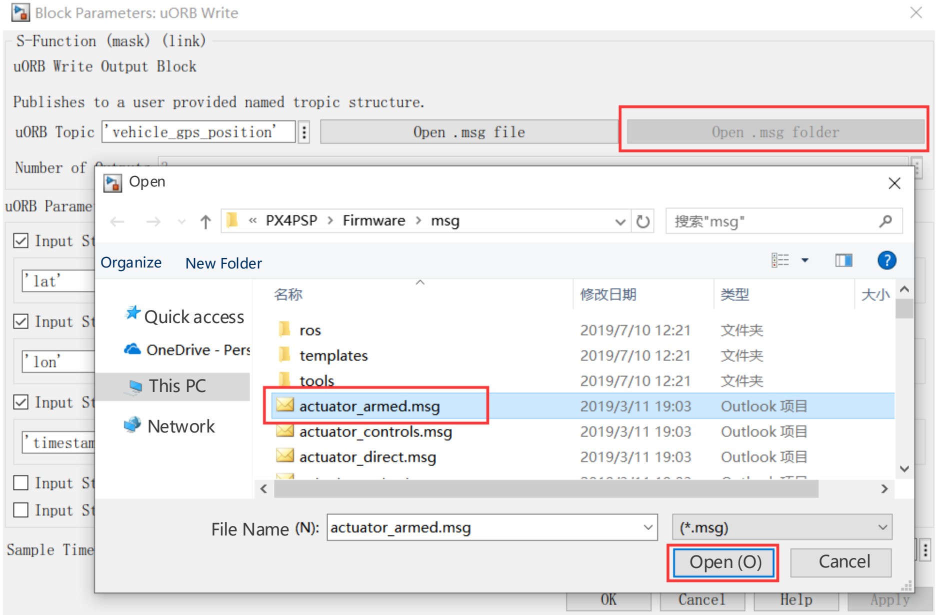


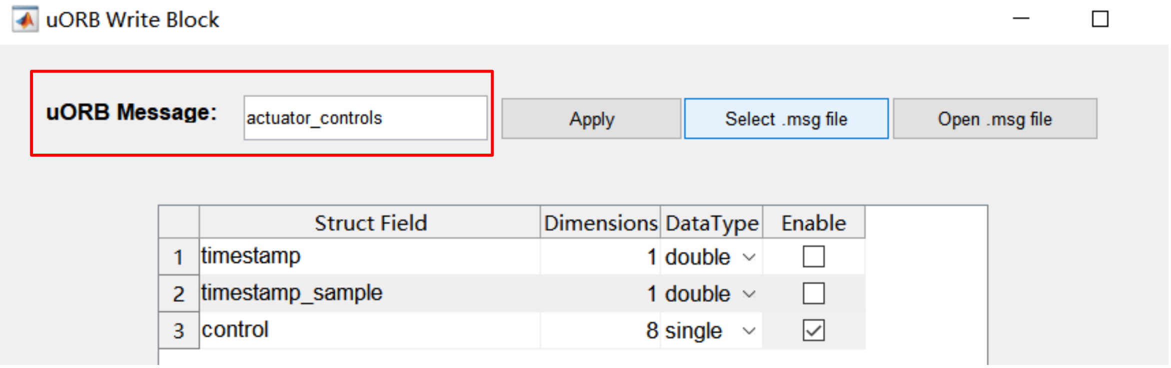
Fig. Pop-up window of "Open .msg folder" button

PSP Toolbox

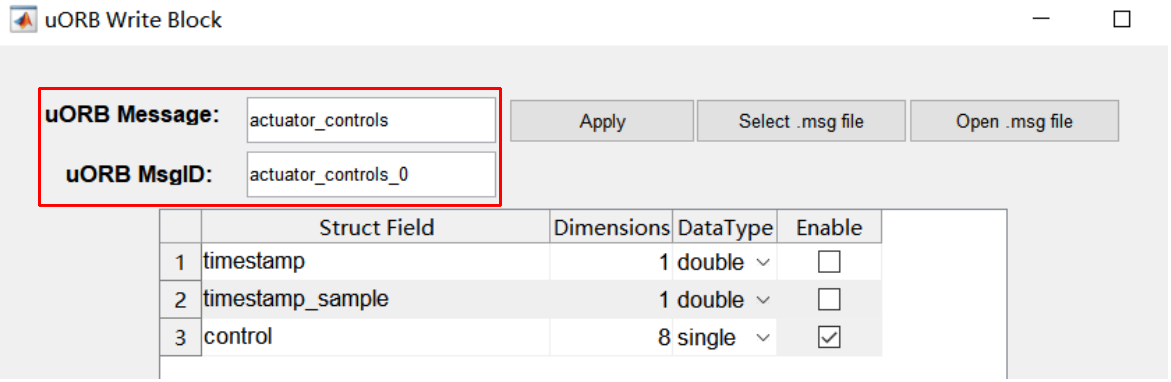
☐ Modules in PSP Toolbox

All modules (PWM output, RGB_LED, etc.) mentioned in this section are implemented at the underlying code by reading and writing uORB messages. Theoretically, by using the “uORB Read and Write” modules, all messages and intermediate variables used in the PX4 autopilot can be accessed by Simulink. This simplifies the implementation of more advanced functions for controller design. The PSP toolbox also provides two examples (see file “e0\2.PSPOfficialExps\px4demo_fcn_call_uorb_example.slx”, and file “e0\2.PSPOfficialExps\px4demo_write_uorb_example.slx”) to study this module.

There are two modules for “uORB Write” operation to choose. The suffix “_dai” allows sending extended messages (such as actuator_controls_0) based on a uORB message base class (such as actuator_controls).



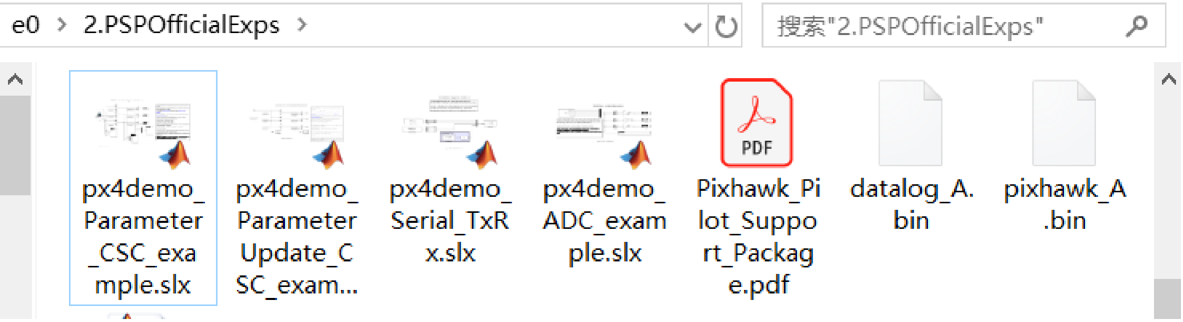
(a) uORB Write Advanced module. uORB MsgID is the name of “.msg” file.



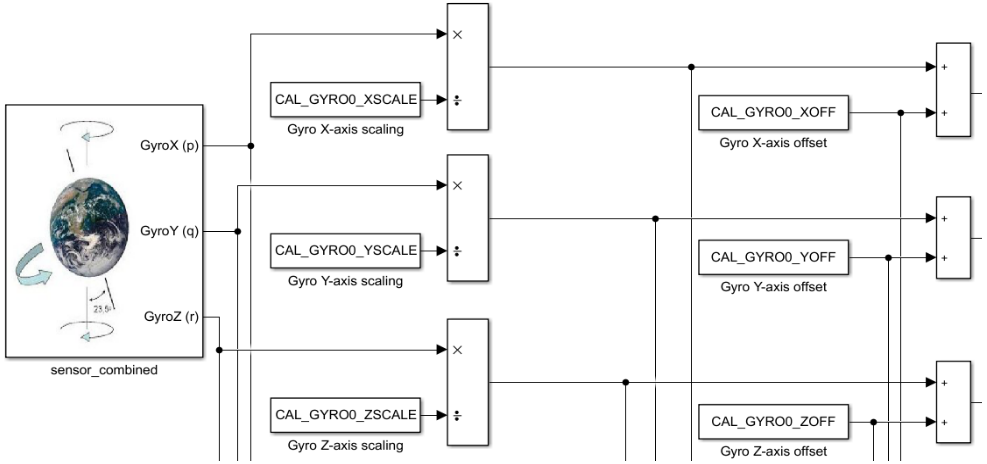
(b) uORB Write Advanced_dai module. uORB MsgID can be set different from the name of the “.msg” file.

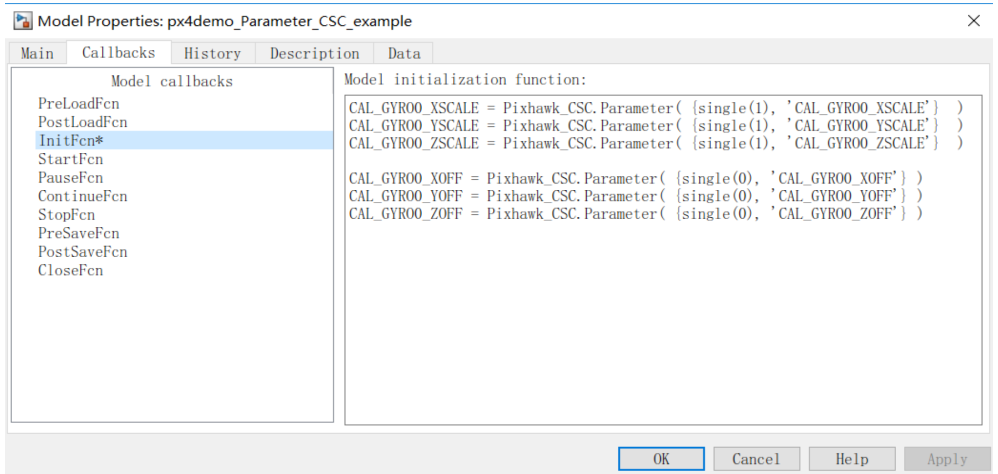
PSP Toolbox

☐ Modules in PSP Toolbox



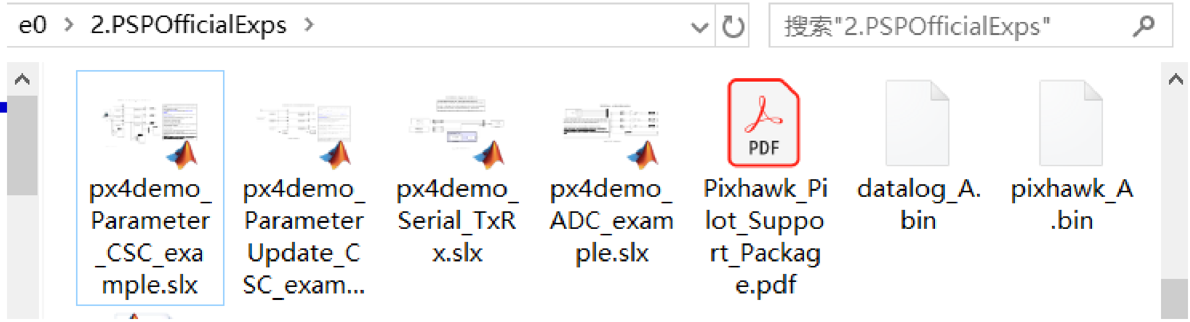
For the sake of convenience for controller parameter tuning in flight tests, the PSP Toolbox also provides interfaces to access the PX4 internal parameters. In this way, the parameters of the controller generated by Simulink can be tuned online in the GCS software, instead of modifying the controller parameters in Simulink, generating code, and uploading the firmware file again. An example of how to access the PX4 internal parameters is presented in file "e0\2.PSPOfficialExps\px4demo_Parameter_CSC_example.slx".



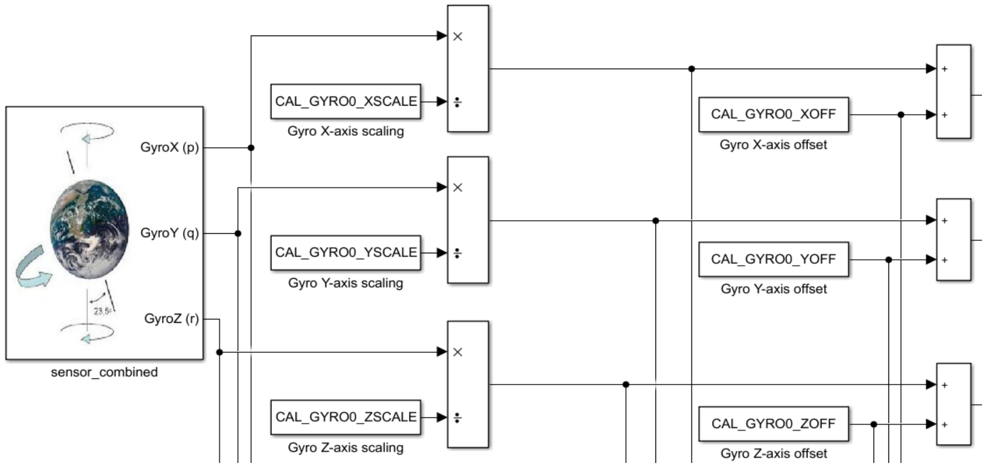


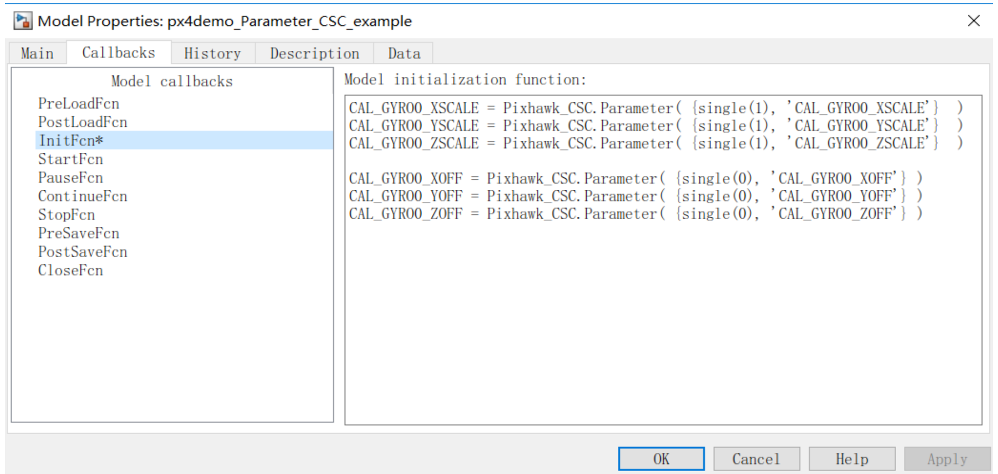
PSP Toolbox

□ Modules in PSP Toolbox



PX4 internal parameter access is realized by using the function "Pixhawk_CSC.Parameter({*, *})", which needs to be called in the Simulink initialization function (click "Simple" - "Model Properties" - "Callbacks" - "InitFcn" in the Simulink menu bar). For the examples shown in the lower-left figure, the corresponding parameter initialization script is shown in the figure on the lower right.





PSP Toolbox

□ Code-Generation Configuration

1) Preparation of controller for code generation

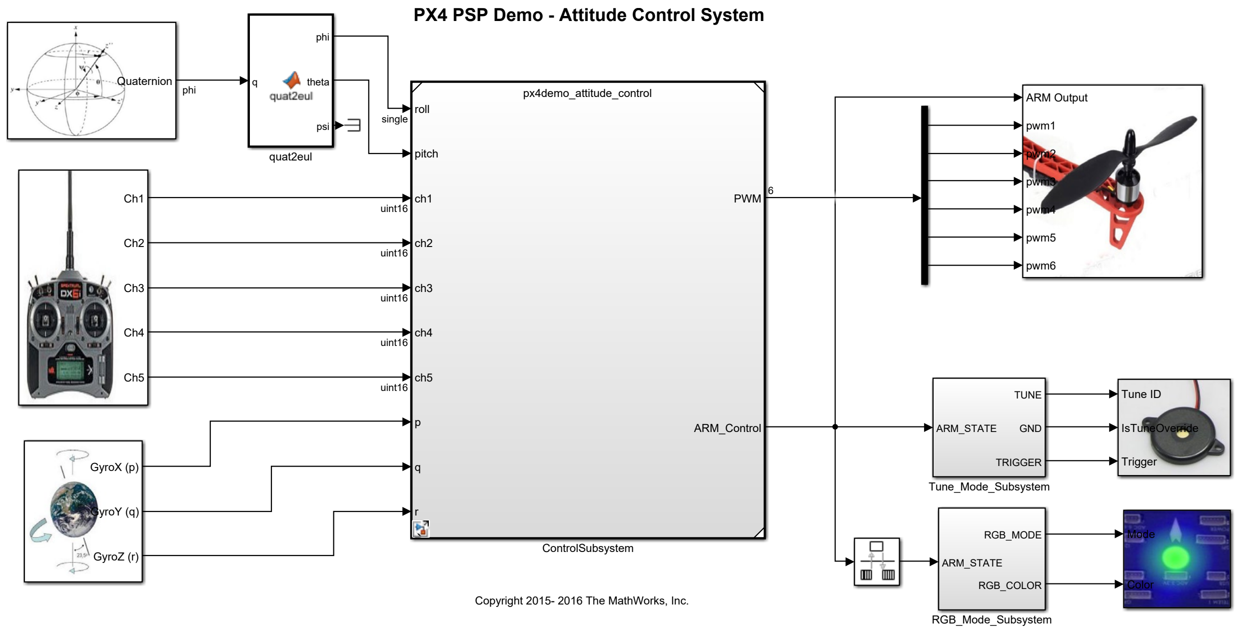


Fig. Example of Simulink controller connecting with PSP modules

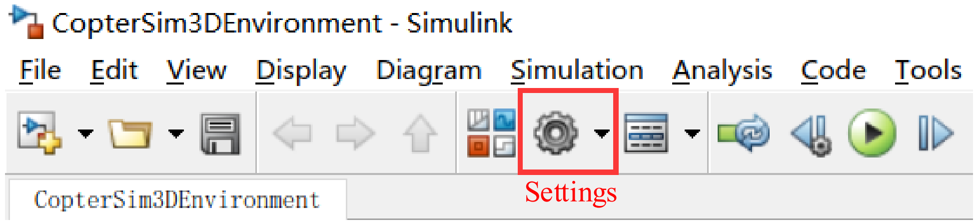
1. design a controller in Simulink and verify it with SIL simulations
2. Copy the verified controller to a Simulink file.
3. Connect the input and output ports of the controller subsystem with the input (e.g., combined sensor module and RC input module) and output (e.g., PWM module and uORB modules) interface modules in the PSP library
4. An example of the obtained Simulink controller file is presented in the figure on the left. The example file is available in "e0\2.PSPOfficialExps\px4demo_attitude_system.slx".

PSP Toolbox

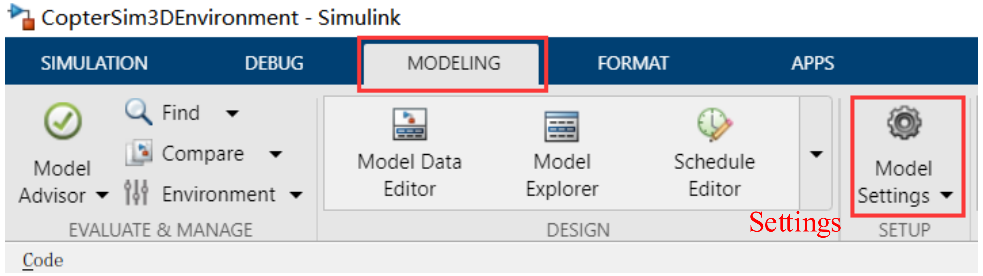
□ Code-Generation Configuration

2) Select target hardware

The new created Simulink file must be configured to support the code generation function of the PSP toolbox. Simulink setting panel can be opened by clicking “setting” button in the Simulink menu bar. Then, go to the “Hardware Implementation” tab and select the “Pixhawk PX4” item in the pull-down menu of the “Hardware board” option.



(a) Simulink “Settings” button on MATLAB 2017b~2019a



(b) Simulink “Settings” button on MATLAB 2019b and above

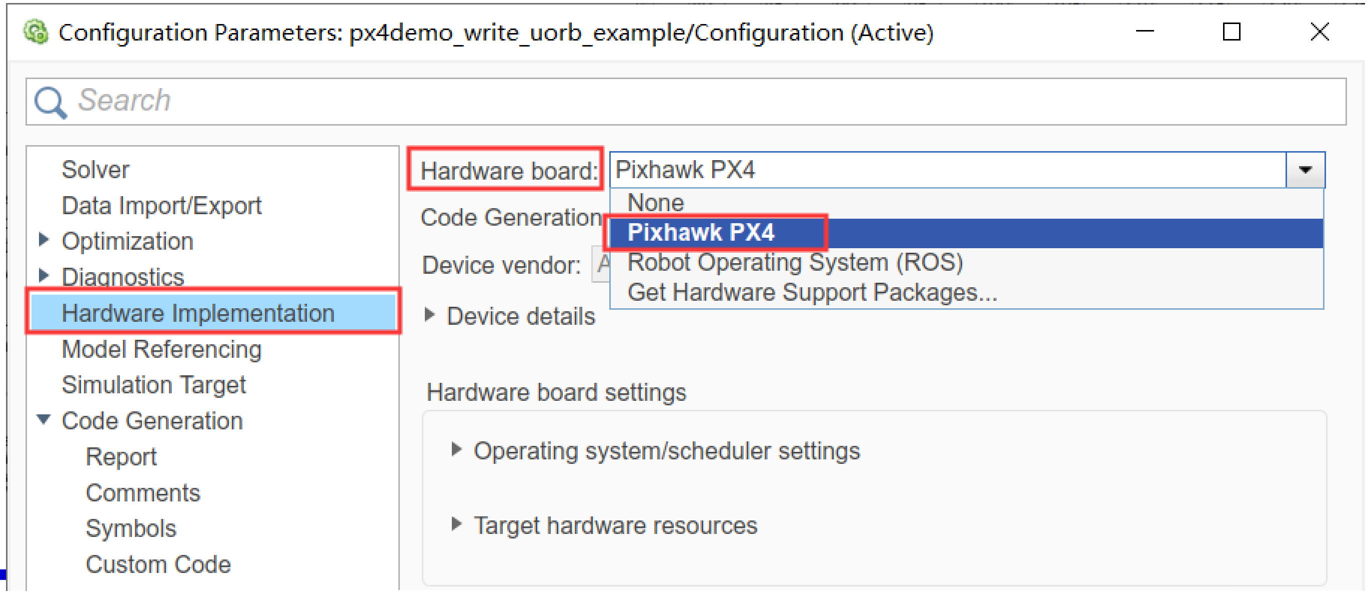


Fig. Simulink “Settings” button for different MATLAB versions

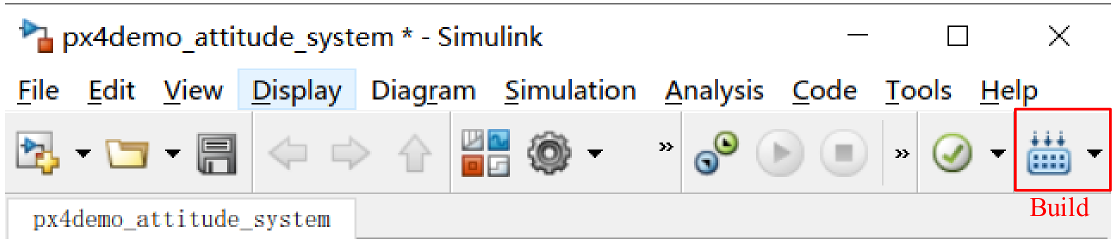
Fig. Select target hardware in setting panel

2.Soft and hardware

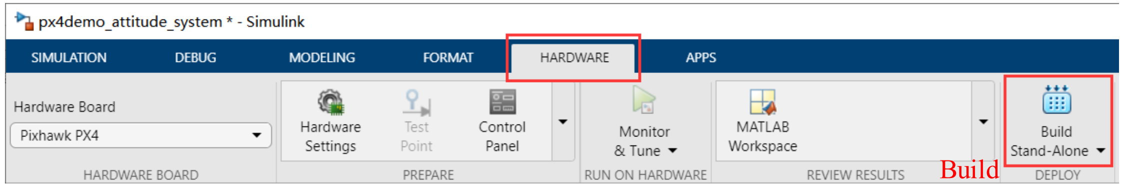
□ Code-Generation Configuration

3) Source code compilation and firmware generation

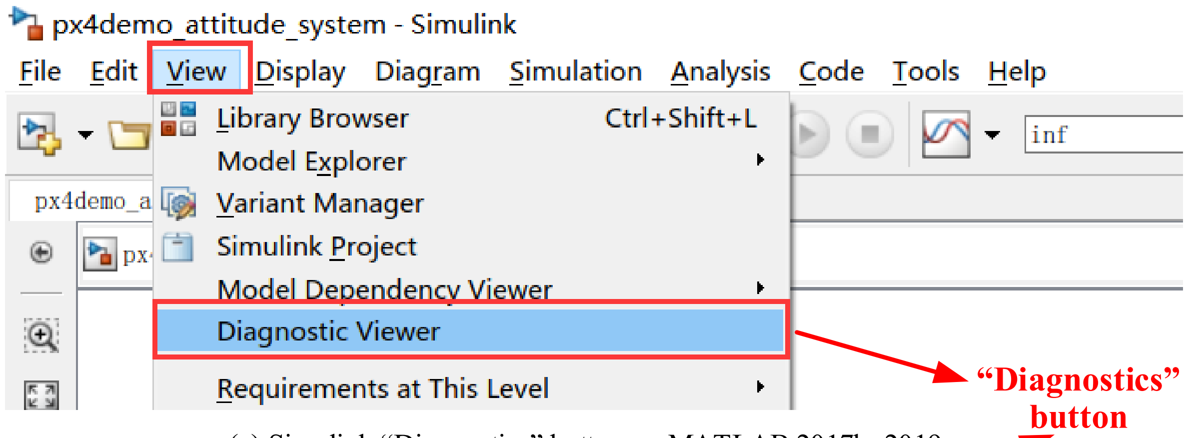
Click the “Build” button. Then, the code generation and compiling process can also be observed by clicking the “Diagnostics” button in Simulink.



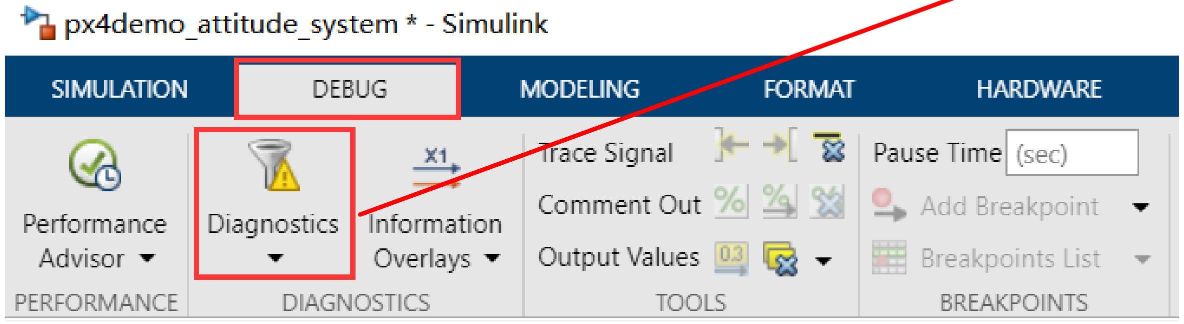
(a) Simulink “Build” button on MATLAB 2017b~2019a



(b) Simulink “Build” button on MATLAB 2019b and above



(a) Simulink “Diagnostics” button on MATLAB 2017b~2019a

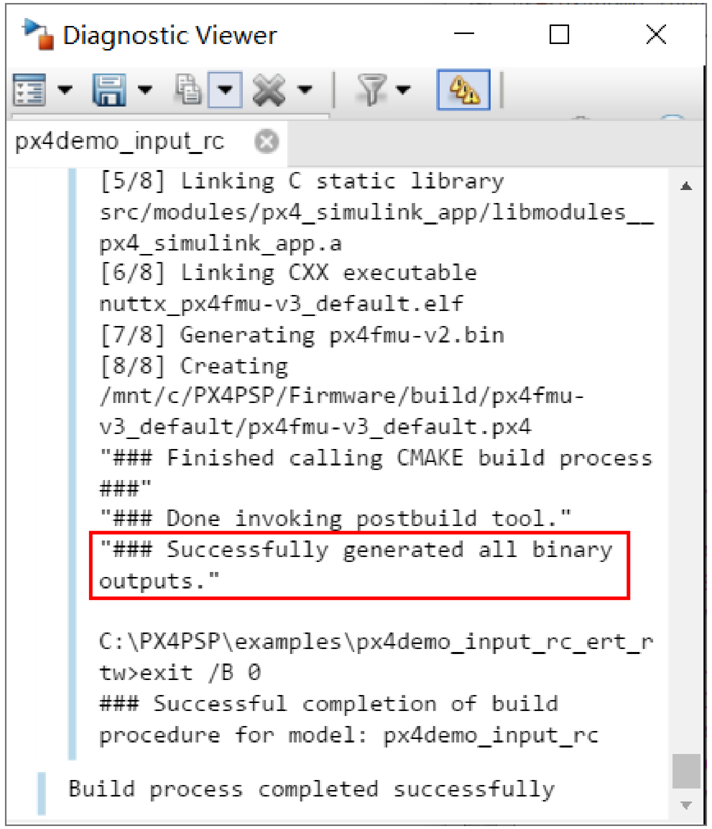


(b) Simulink “Diagnostics” button on MATLAB 2019b and above

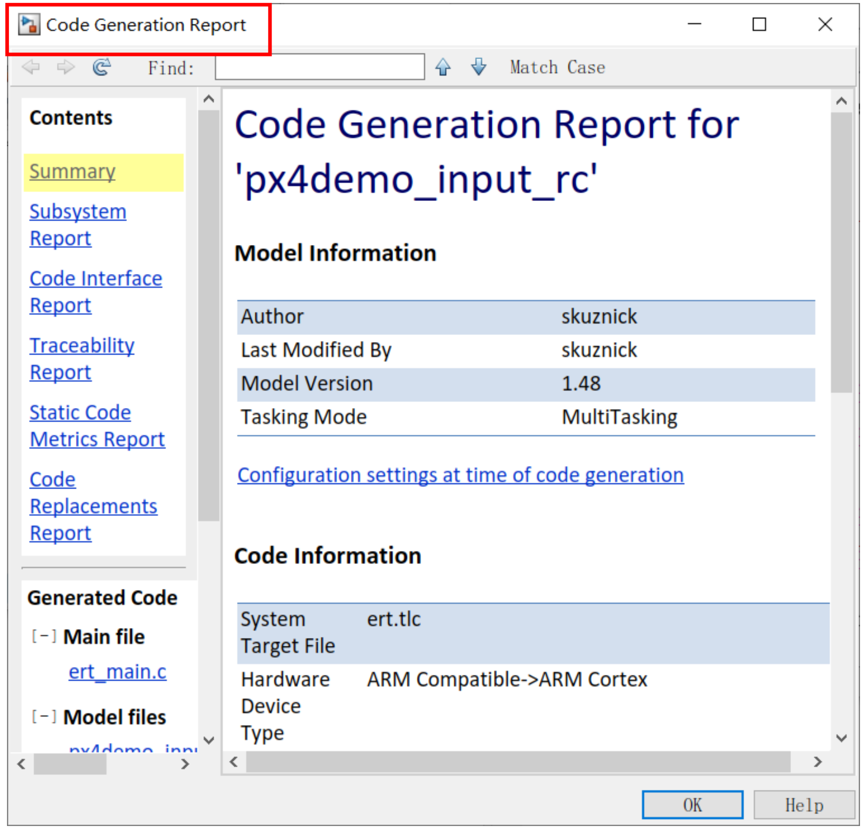
□ Code-Generation Configuration

4) Compiling process and results

A successful compiling process in the "Diagnostic Viewer" dialog is shown in Fig. (a), where the compiling progress is finished with the following text "Successfully generated all binary outputs". It can also be observed in Fig. (b) that a "Code Generation Report" document will pop up after the compiling process is finished.



(a) Compilation progress

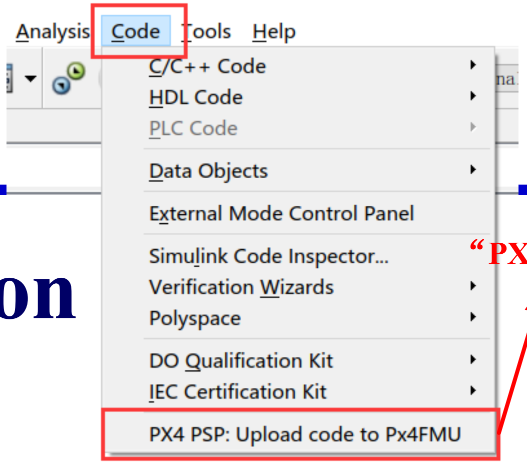


(b) Code generation report

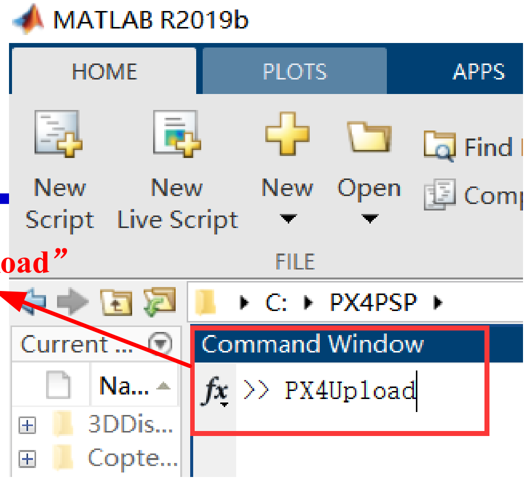
PSP Toolbox

□ Code-Generation Configuration

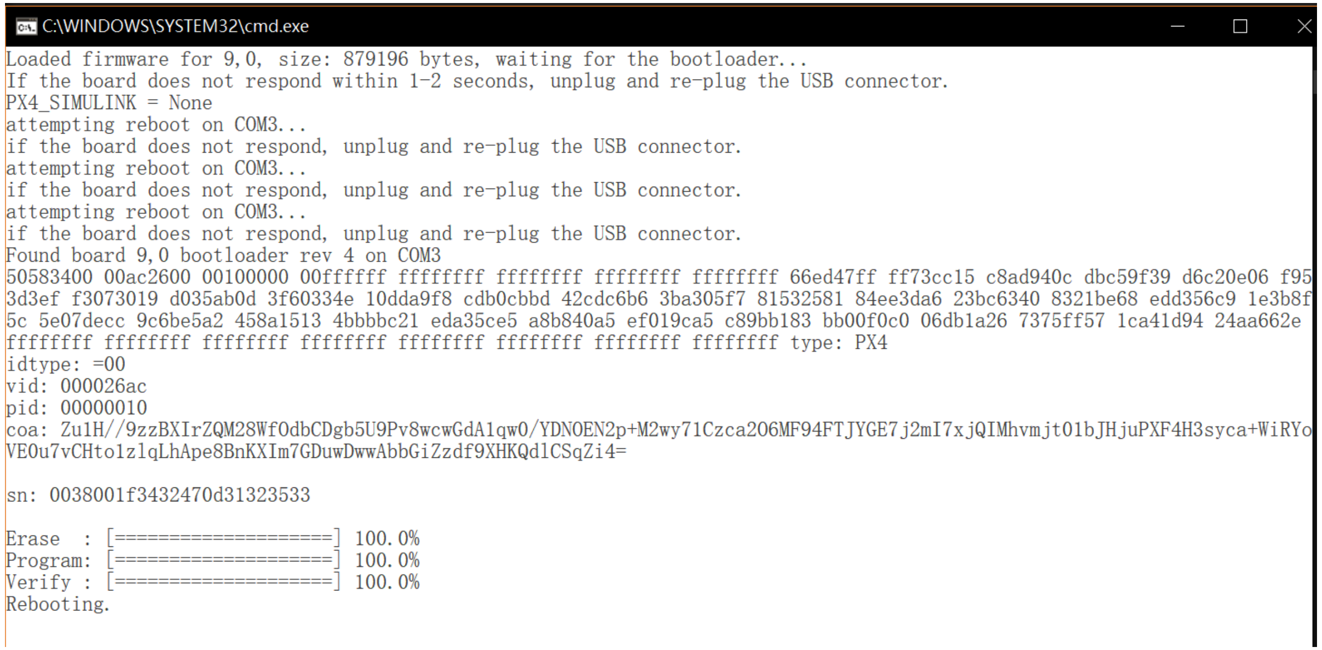
5) Upload PX4 firmware to Pixhawk hardware



(a) Simulink "PX4Upload" button for MATLAB 2017b~2019a



(b) MATLAB "PX4Upload" command for MATLAB 2019b and above

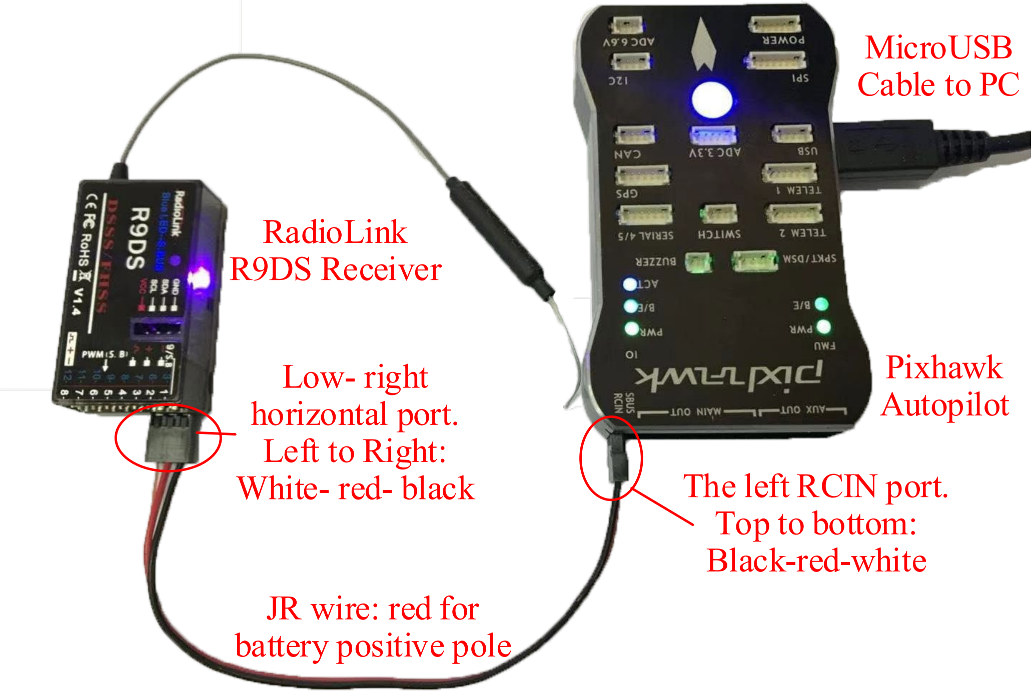


1. Use a USB cable to connect the MicroUSB (on the Pixhawk hardware) with the USB port on the computer.
2. For MATLAB 2017b-2019a, click "Code" - "PX4 PSP: Upload code to Px4FMU" on the Simulink menu bar, then the firmware will be automatically uploaded to the Pixhawk autopilot; for MATLAB 2019b and above, readers can input the "PX4Upload" in MATLAB.
3. Sometimes, the Pixhawk autopilot has to be re-plugged to start the firmware uploading process.

Pixhawk Hardware System

□ Hardware Composition and Connection

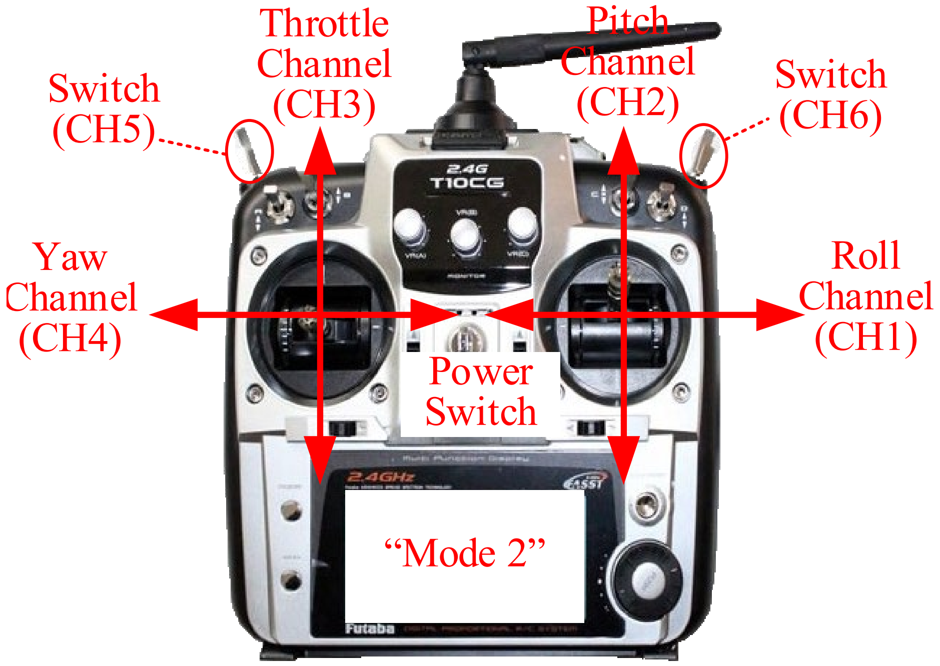
Hardware components required by this book include an RC transmitter, an RC receiver, a JR signal cable (connecting the Pixhawk autopilot and the RC receiver), a Pixhawk autopilot (Pixhawk 1 is recommended for studying, and higher hardware versions are recommended for outdoor flight tests), and a MicroUSB cable (connecting the computer and the Pixhawk hardware for power supply and data transmission).



Pixhawk Hardware System

□ Basic Operation Method for RC Transmitter

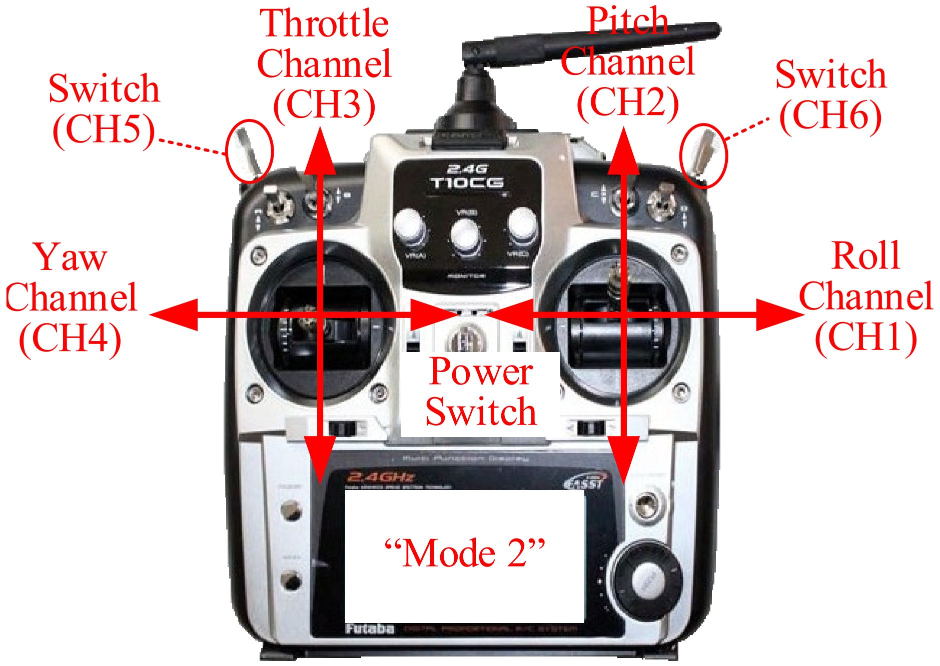
1. The RC transmitter used in this book should be set to "Mode 1". The throttle and yaw channels are controlled by the left stick on the RC transmitter, whereas the roll and pitch channels are controlled by the right stick.
2. The roll, pitch, throttle, and yaw channels correspond to the CH1 to CH4 of the RC receiver respectively; the upper-left and upper-right three-position switch corresponds to CH5 and CH6 for triggering the autopilot to switch flight modes or enable other functions.



Throttle : control up-down movement
Pitch : control forward-backward
Yaw : control vehicle head direction
Roll : control left-right movement

Basic Operation Method for RC Transmitter

3. The throttle stick (CH3) and the pitch stick (CH2) moves from the bottom position to the top position, and the corresponding PWM value received by Pixhawk changes from 1100 to 1900.
4. The roll stick (CH1) and the yaw stick (CH4) move from the left position to the right position, and the corresponding PWM values change from 1100 to 1900.
5. The upper-left switch (CH5) and the upper-right switch (CH6) move from the top position (the farthest position from the user), middle position, and bottom position (the closest position from the user), and the corresponding PWM values change from 1100, 1500, to 1900.



Throttle : control up-down movement
Pitch : control forward-backward
Yaw : control vehicle head direction
Roll : control left-right movement

Pixhawk Hardware System

QGroundControl

☐ Uploading Firmware through QGC

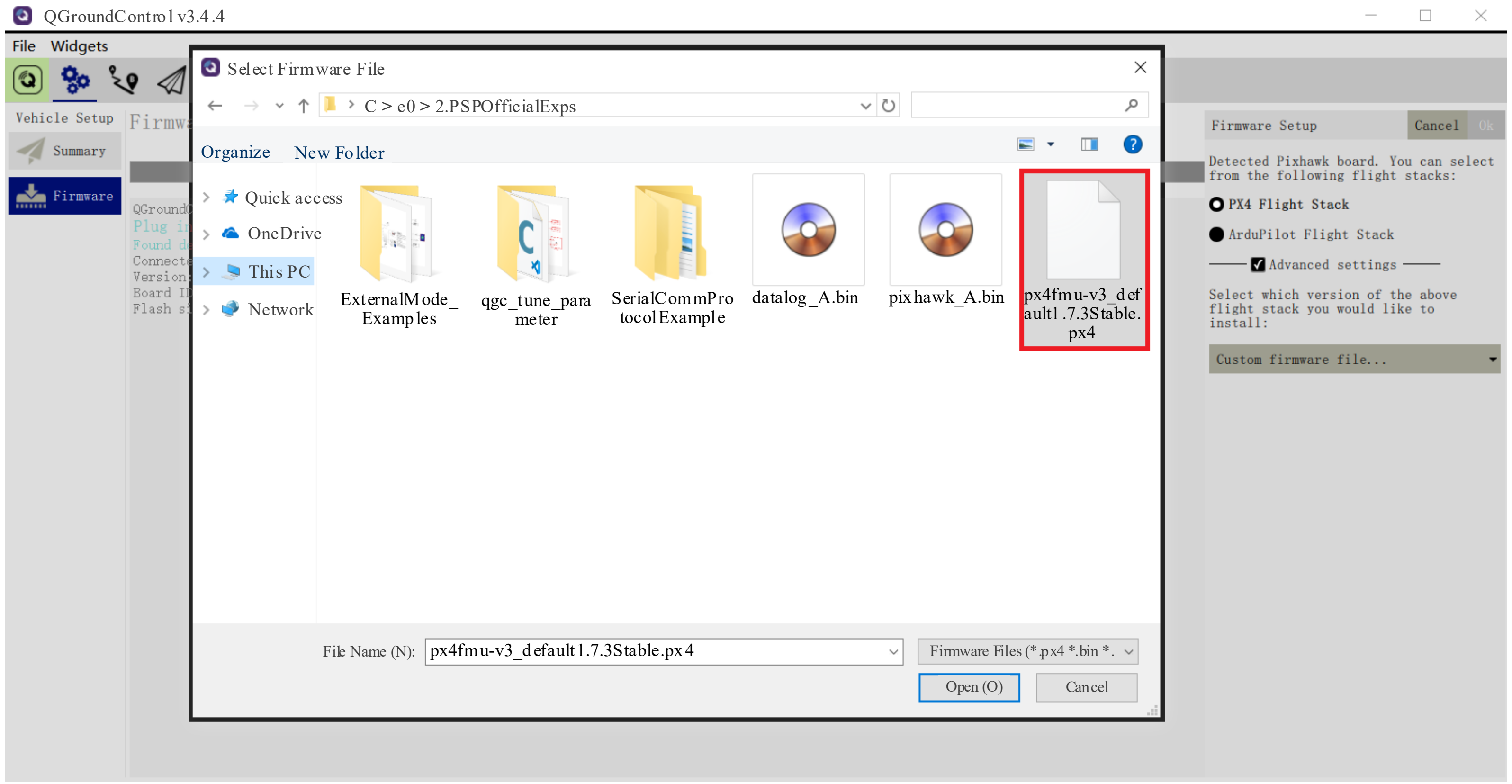
Open QGroundControl and do the following steps.

1. Click the “Settings” button (the gear icon on the toolbar in the figure) to enter QGC setting page.
2. Click the “Firmware” tab, and then connect the Pixhawk hardware with a USB cable. QGC will automatically detect the Pixhawk autopilot. (Note: the subsequent operations are for Pixhawk 1 hardware, so if you are using other Pixhawks, just click “OK” button to download and use the official firmware, and then you can skip the following steps);
3. Click the “Advanced settings” checkbox.
4. Click on the “Standard Version (stable)” tab.

Pixhawk Hardware System

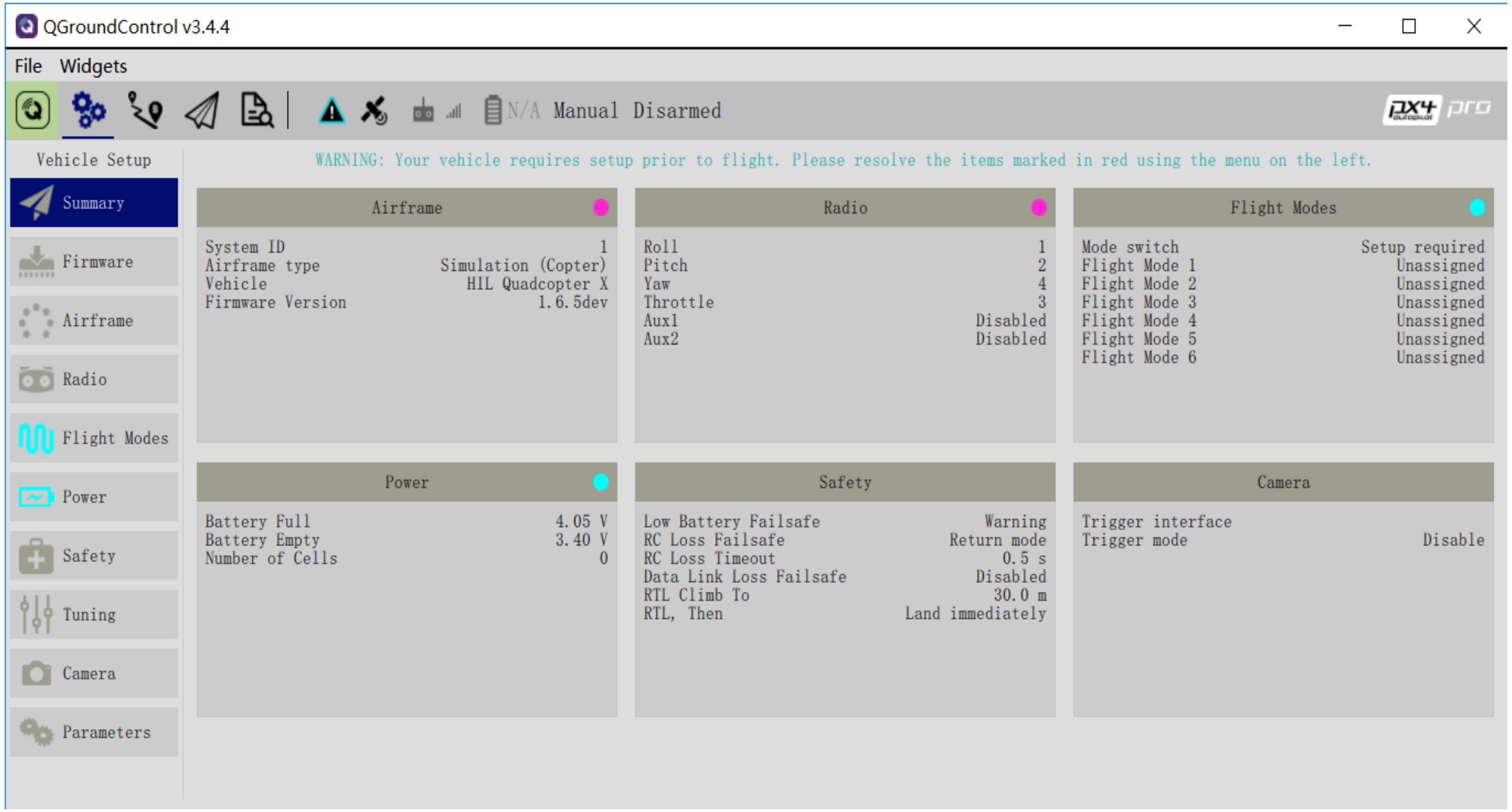
□ Uploading Firmware through QGC

5. Select the "Custom firmware file .." option in the pop-up menu.
6. Click the "OK" button.
7. Select file "\e0\2.PSPOfficialExps\px4fmu-v3_default-1.7.3Stable.px4" in the pop-up file selection window, and click the "Open" button. Then, QGC will upload and burn the firmware into the Pixhawk hardware. (Note: this firmware file is only for Pixhawk 1 with 2M flash or Cube, for other Pixhawk hardware please download desired .px4 firmware at https://github.com/PX4/Firmware/releases/tag/v1.7.0)



□ Pixhawk HIL Simulation Mode

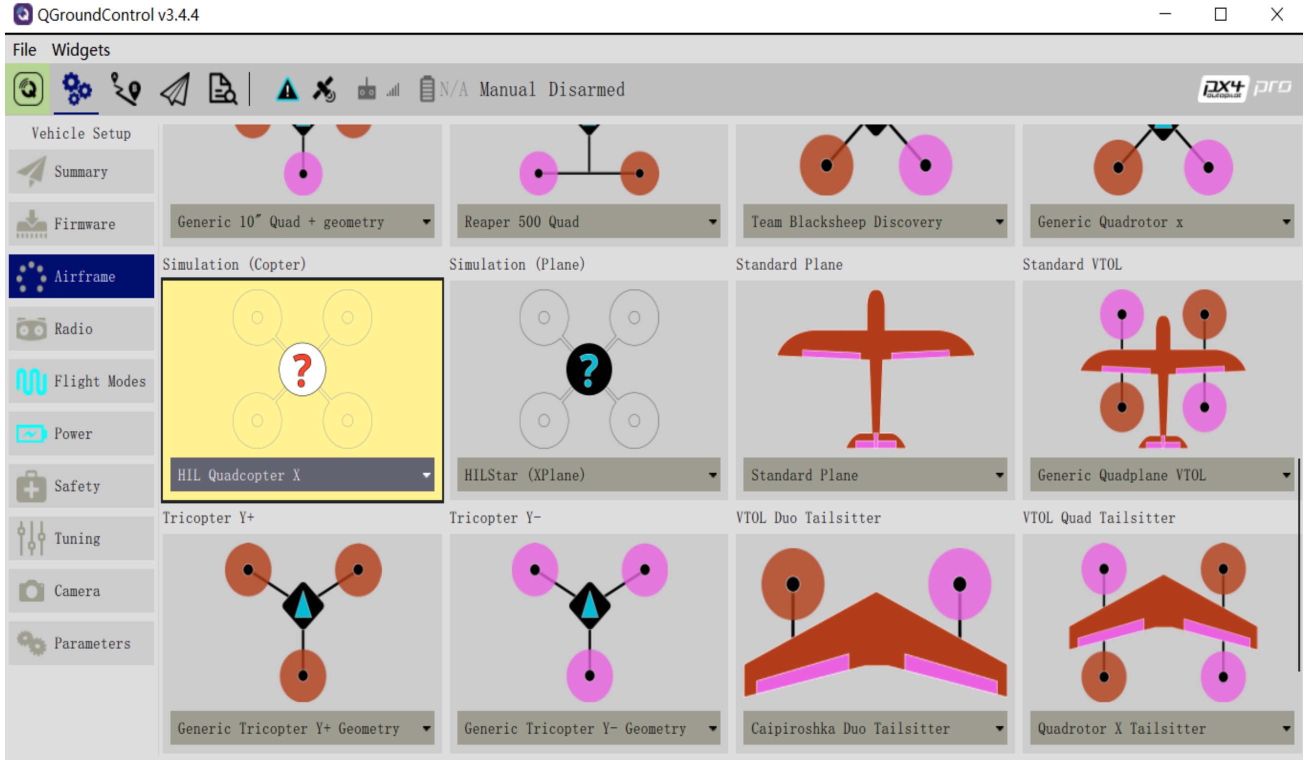
1) Open the QGC software, and connect the Pixhawk autopilot with a USB cable. After QGC automatically recognize the Pixhawk autopilot and create a connection for parameter setting and data transmission, the UI of QGC can be seen.



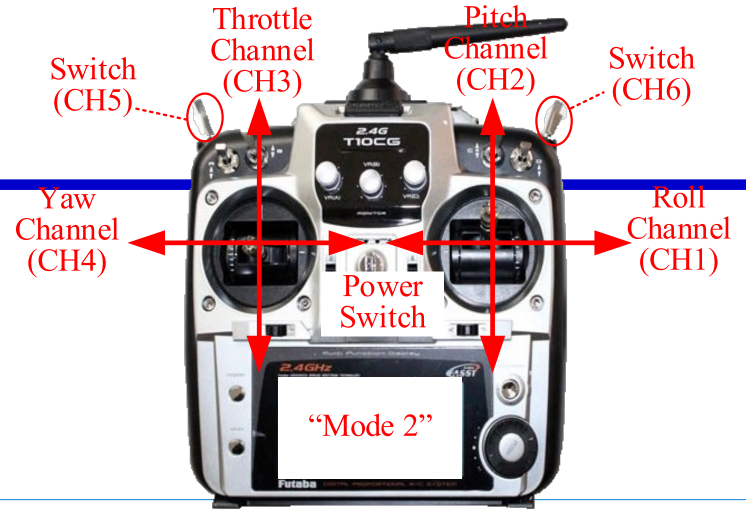
□ Pixhawk HIL Simulation Mode

2) Click the “Airframe” tab (the third item) on the QGC setting page to confirm that the “HIL Quadcopter X” airframe mode is selected by default. This setting is critical for subsequent HIL simulation. Otherwise, a manual setup will be required. The method is :

- Select the “HIL Quadcopter X” airframe icon.
- Click the “Apply and Restart” button in the upper right corner of the UI. Then, the autopilot will restart to make the new airframe available.
- Wait for a few seconds; QGC will connect to the autopilot again and will check whether the setting is correctly established.

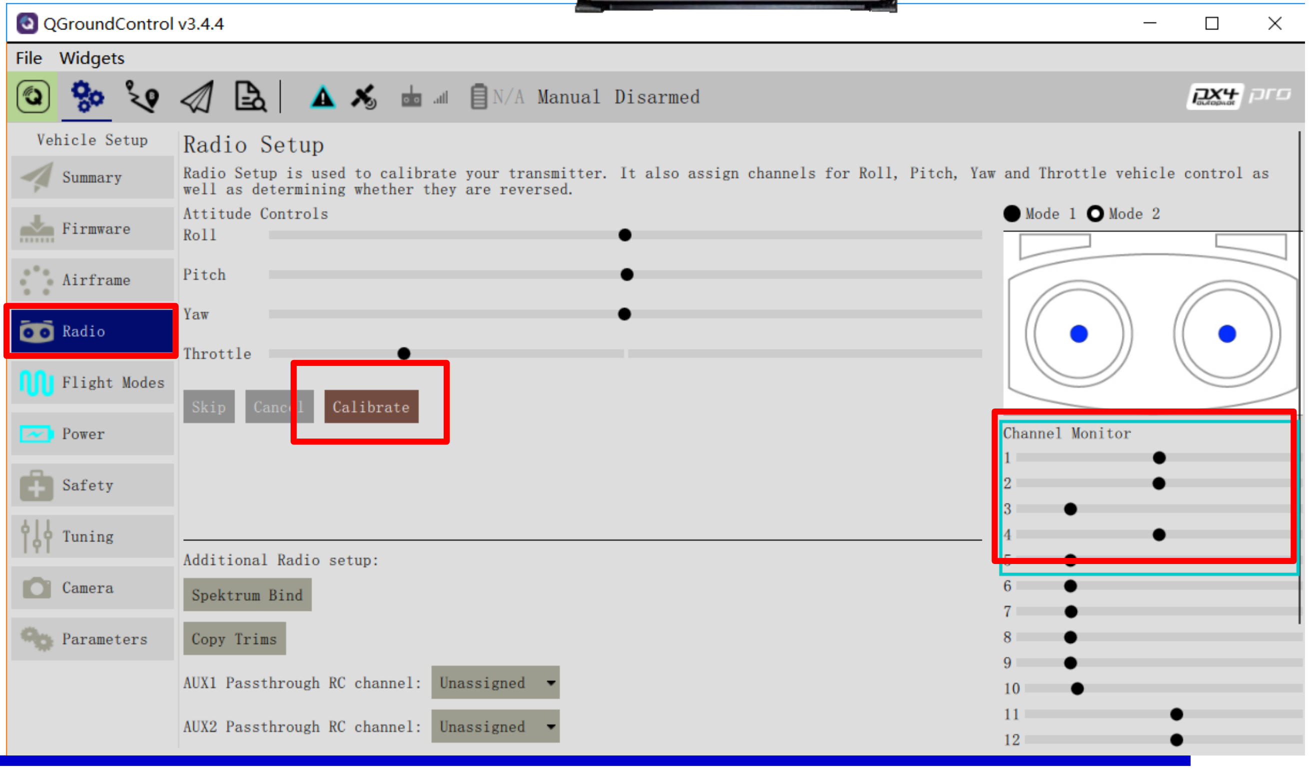


Pixhawk Hardware System



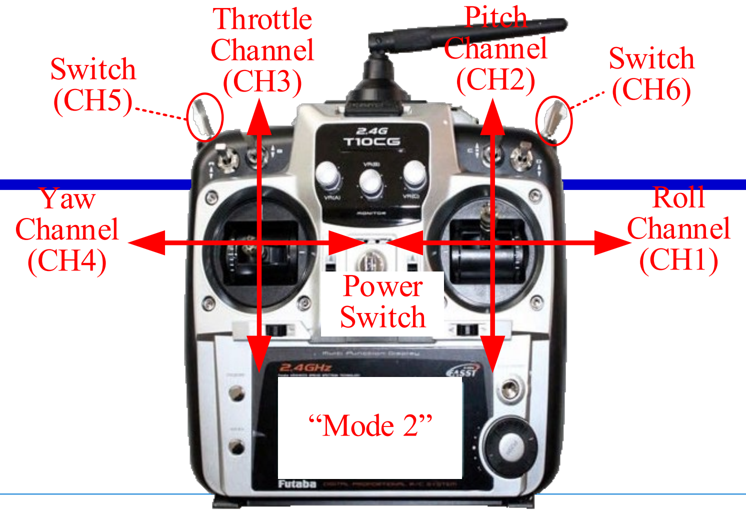
◻ RC Configuration & Calibration

1. connect the Pixhawk autopilot and the RC receiver.
2. connect the Pixhawk autopilot with the computer.
3. Next, turn on the RC transmitter, and open QGC
4. click the "Radio" item on the QGC setting page
5. Move the sticks of the RC transmitter and observe the trend of channels 1-6 on the Channel Monitor page on QGroundControl
6. The first and the fourth sliders in the figure on the right should move from left to right (the PWM values change from 1100 to 1900).

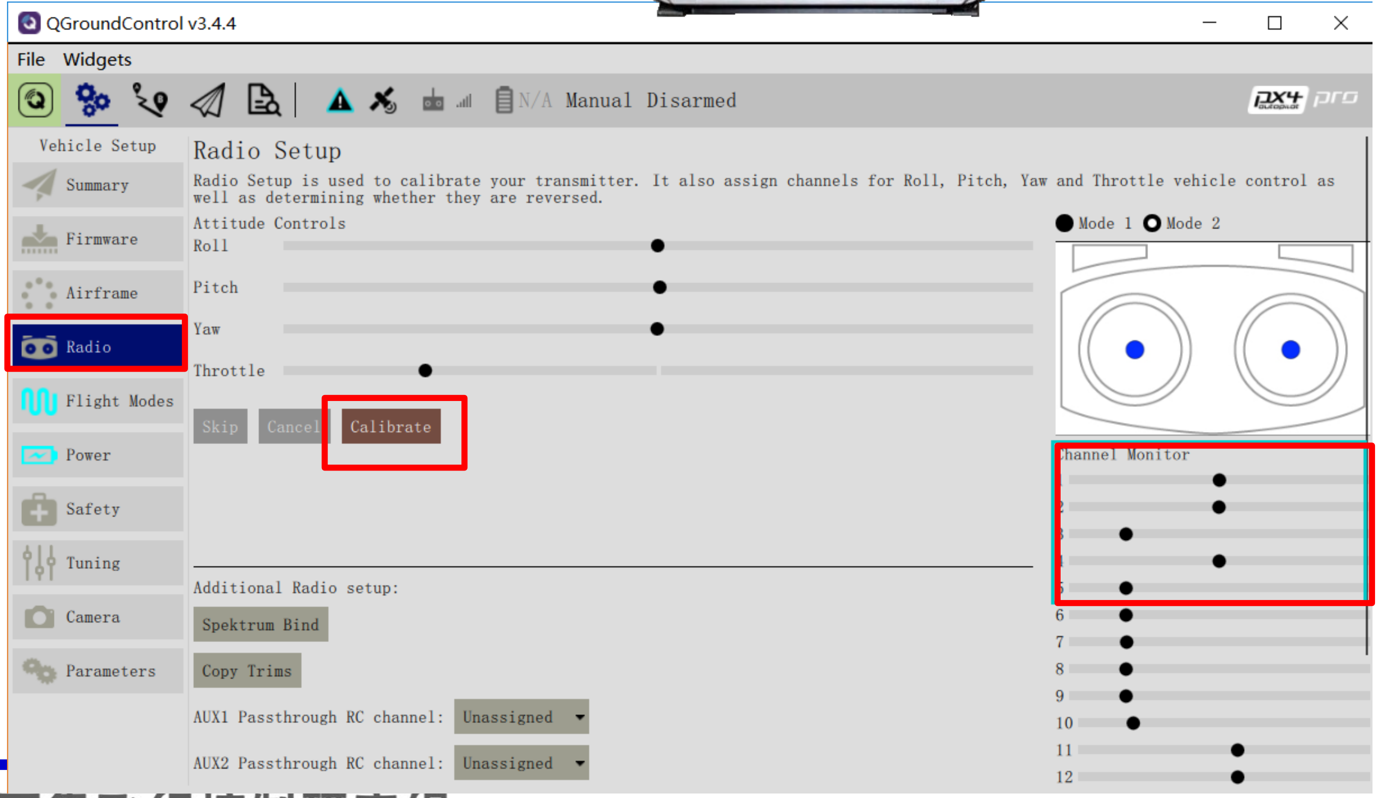


Pixhawk Hardware System

☐ RC Configuration & Calibration



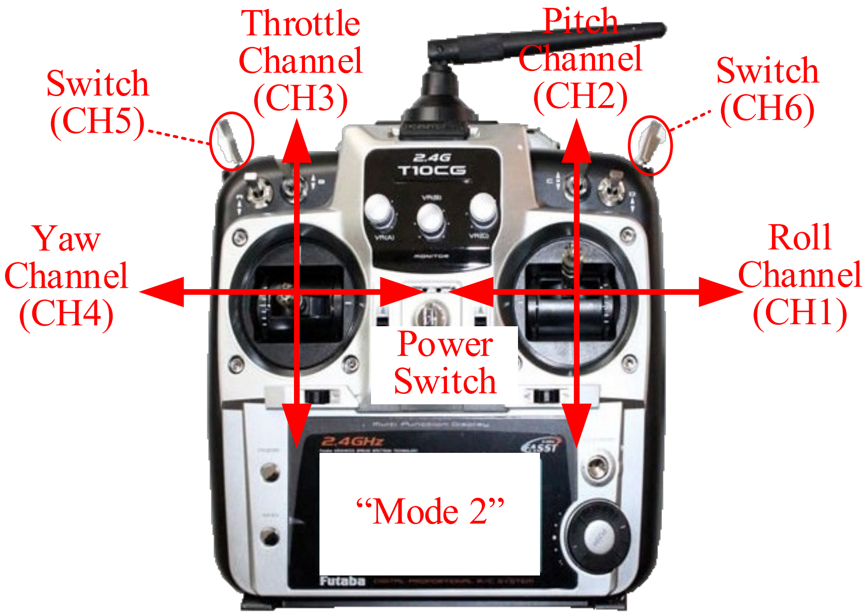
7. The third slider on the bottom-right region should move from left to right (the PWM value changes from 1100 to 1900).
8. If the above rules are not satisfied, it means that the RC transmitter is not set correctly, so the RC transmitter should be re-configured.
9. If the RC is configured correctly, click the "Calibrate" button and complete the RC calibration by moving the sticks according to the instructions on QGC.

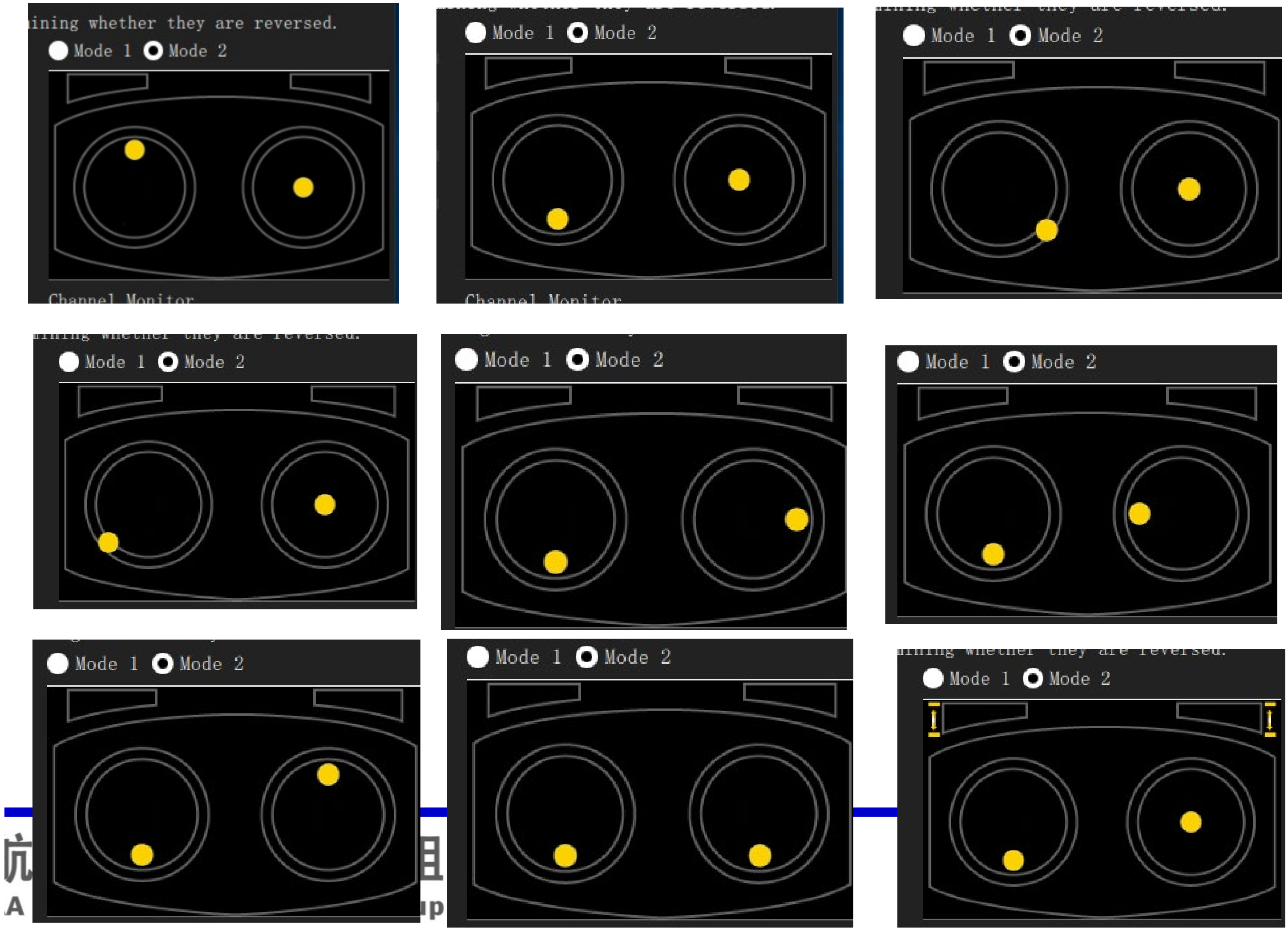


Pixhawk Hardware System

□ RC Configuration & Calibration

10. Click QGC's "Calibrate" – "Next" button, then move the RC's sticks according to the figures on the right (just follow the hints on QGC) to finish the RC calibration

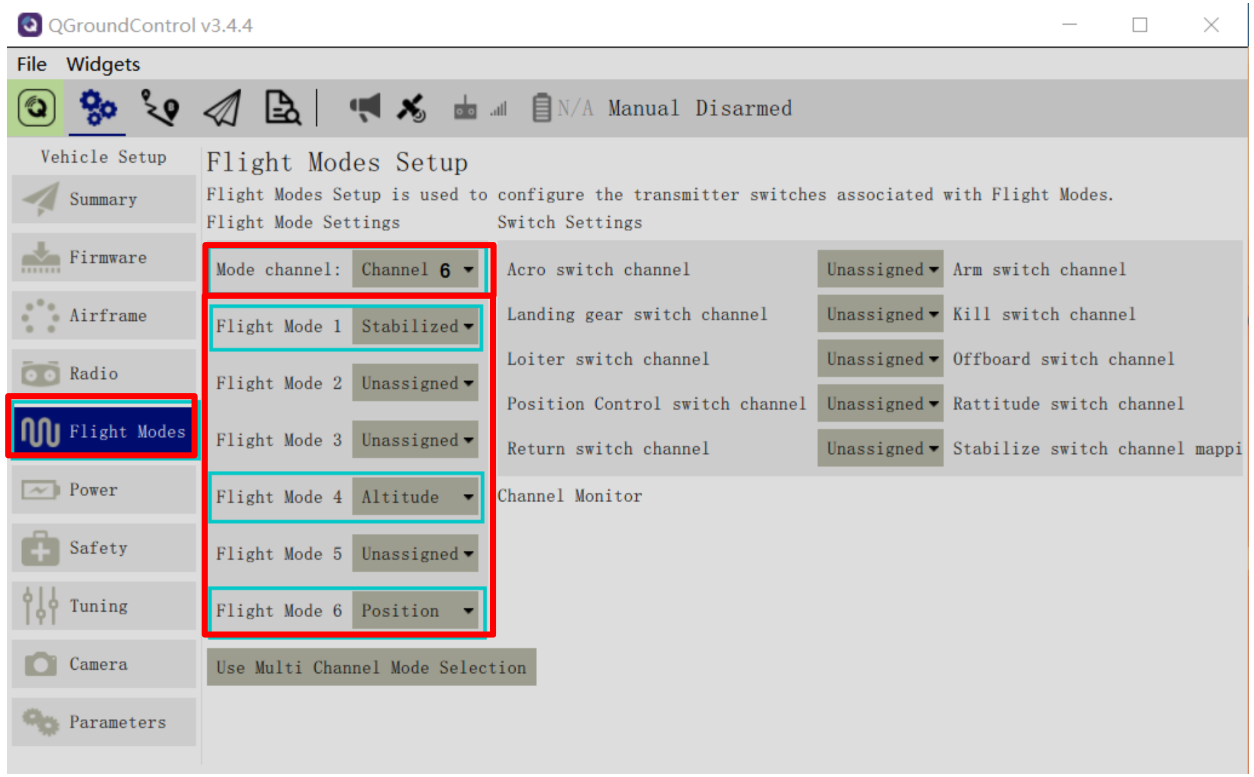




Pixhawk Hardware System

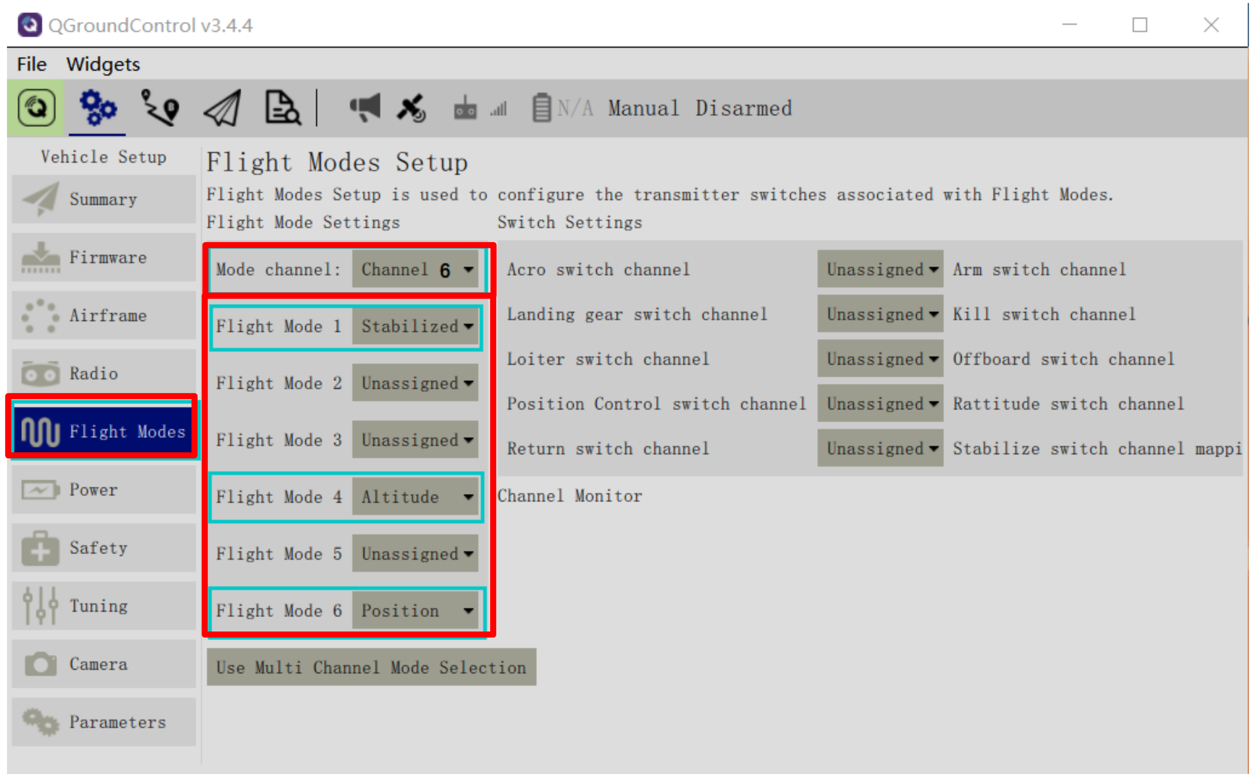
□ Flight Mode Settings

1. After the RC transmitter is successfully calibrated, enter the “Flight Modes” setting page (see the figure on the right) and select “Mode Channel” as the previously tested CH6 channel. Since the CH6 channel is a three-position switch, the top position (the farthest position from the user), middle position, and bottom position (the closest position from the user) of the switch correspond to “Flight Mode 1, 4, 6” in figure.



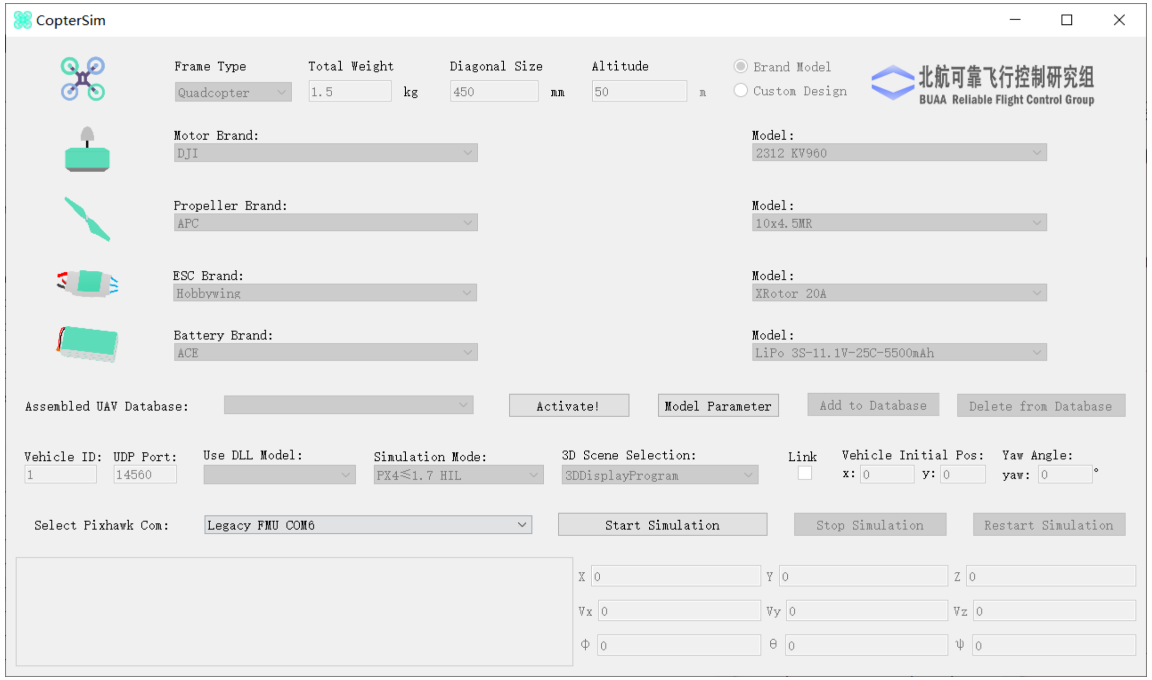
□ Flight Mode Settings

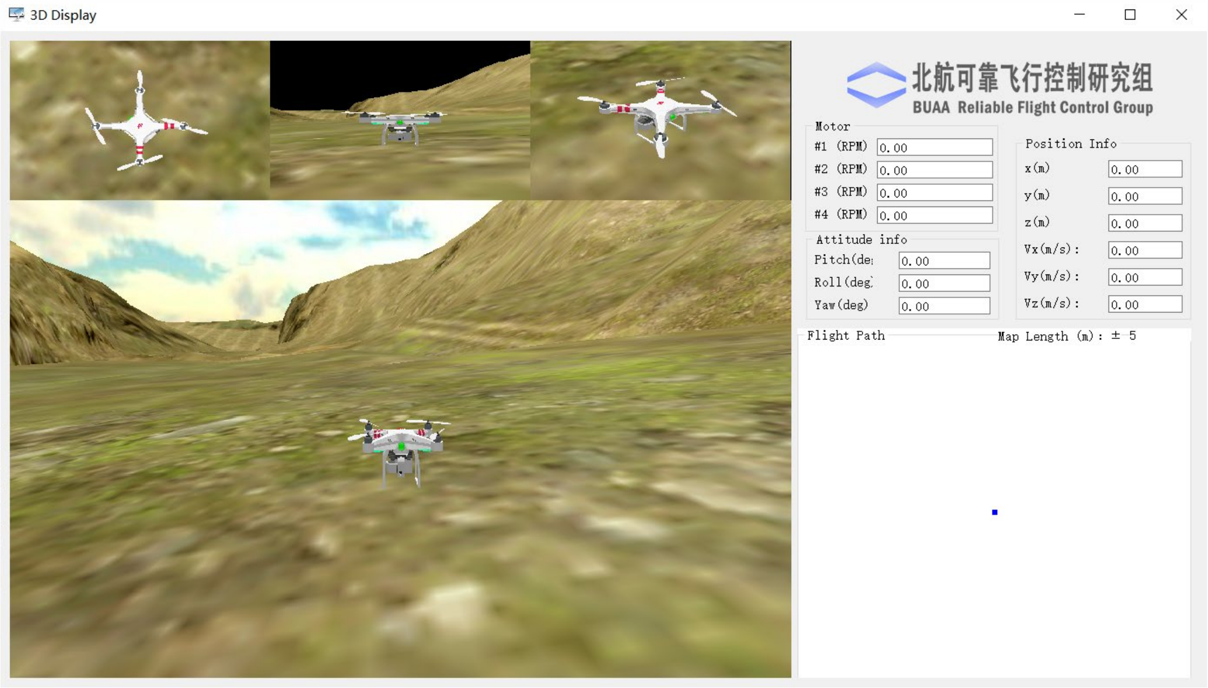
2. Associate these three modes to “Stabilized” (the stabilized mode, only including attitude control), “Altitude” (the altitude hold mode, including attitude and altitude control), and “Position” (the loiter mode, including attitude and position control).
3. In subsequent HIL simulations, you can experience different control effects by switching between different modes.



HIL Simulation Platform

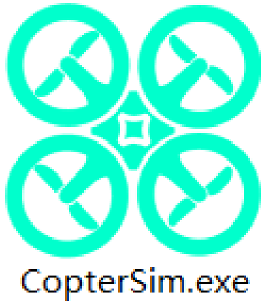
The HIL simulation platform includes a Real-time Motion Simulation Software — CopterSim and a 3D Visual Display Software — 3DDisplay.



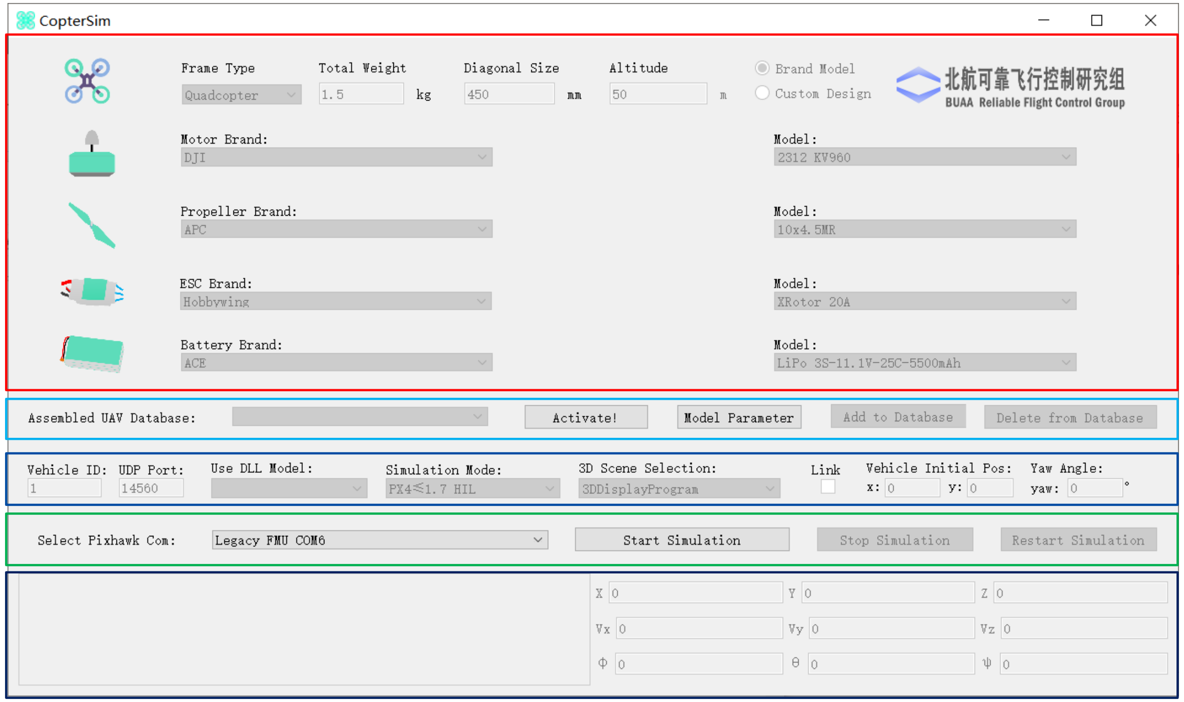


HIL Simulation Platform

□ CopterSim



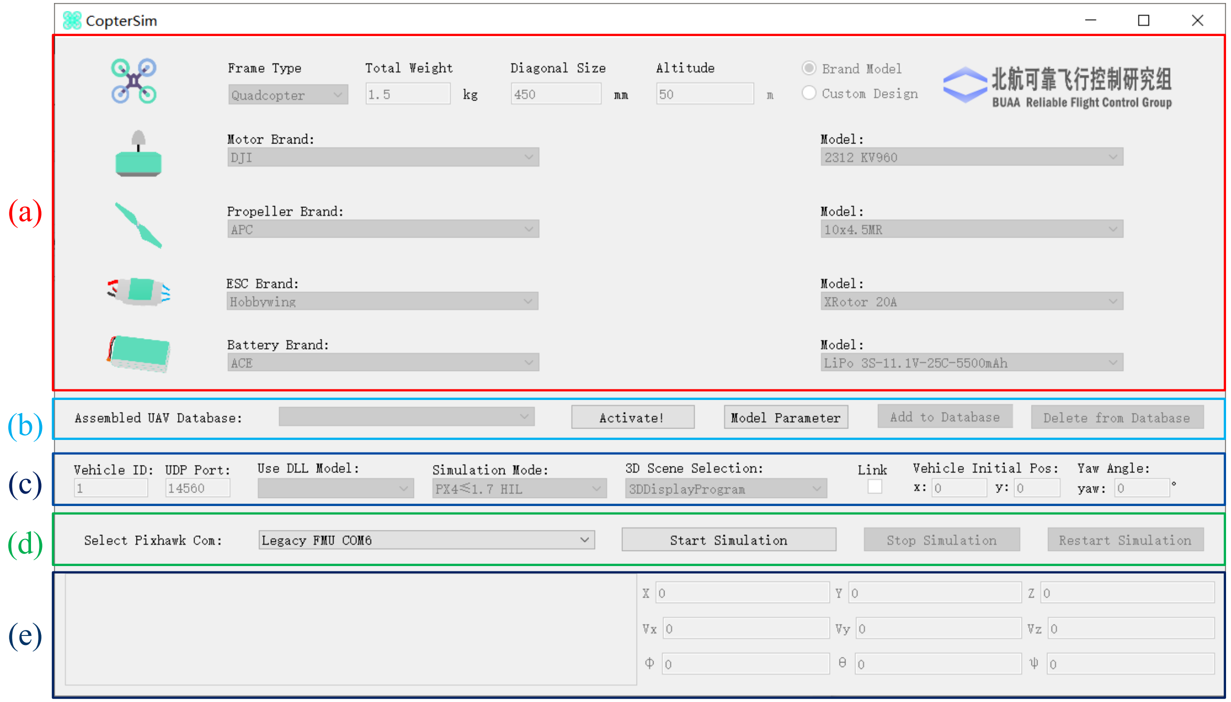
(a)
(b)
(c)
(d)
(e)



1) Double-click the CopterSim shortcut on the Windows desktop to open the CopterSim software, whose UI is presented in the figure on the left. The default simulation model and parameters are the same as for the Simulink multicopter model used in the SIL simulation system. This is because the CopterSim is developed based on the code generation technique with the Simulink multicopter model. CopterSim needs to run on a x64 Windows computer platform with a serial port and a MicroUSB cable to communicate with the Pixhawk autopilot.

HIL Simulation Platform

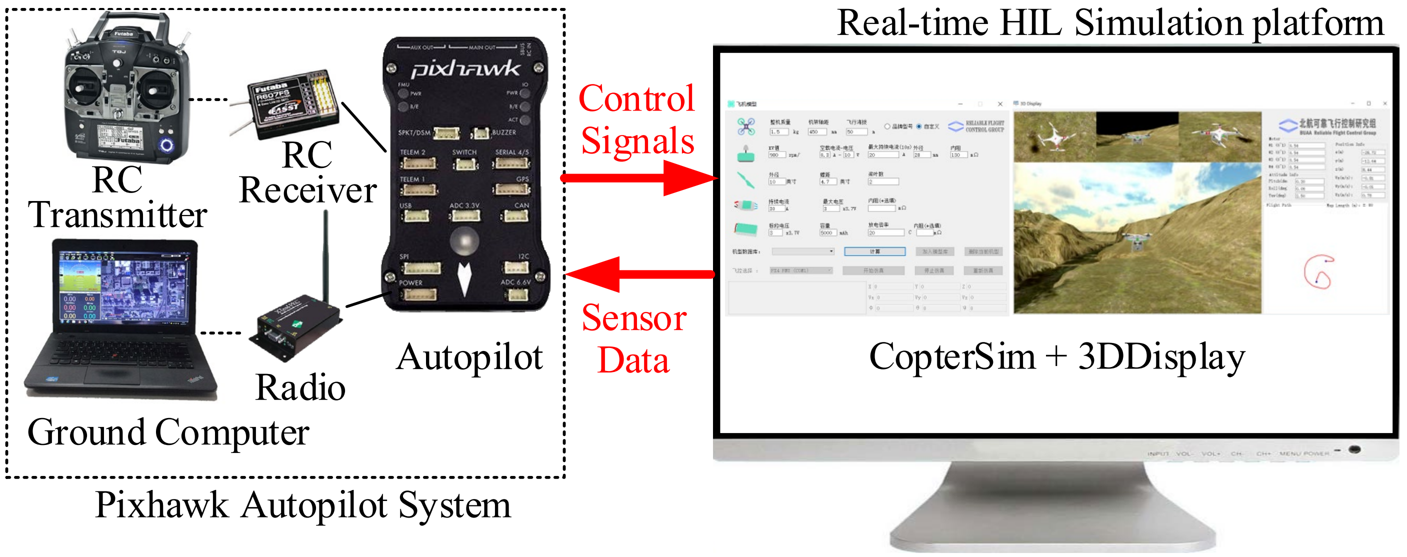
□ CopterSim



2) As shown in the figure on the left, the UI of CopterSim is divided into two parts. The upper part, presented in Fig. (a), is the input interface to design a multicopter by selecting popular components on the market. The lower part presented in Figs. (b)-(e) is the interface to connect with the autopilot for HIL simulation. Note that CopterSim enables by default only the basic functions required by this book. Registration is required to use many other practical functions, such as swarm simulation, high-fidelity UE4 scenes, and HIL simulations for other aerial vehicles (e.g., fixed-wing aircraft).

HIL Simulation Platform

◻ CopterSim



Principle of HIL simulation: CopterSim sends sensor data to the Pixhawk autopilot, and then the autopilot solves the motor PWM control signal and returns it to CopterSim. As a result, the Pixhawk autopilot can perform real-time control on the simulated multicopter in CopterSim, as well as control a real multicopter. Meanwhile, CopterSim will send the attitude and position information of the multicopter to the local network through the UDP protocol, and the 3DDisplay receives the multicopter flight information to complete the corresponding real-time 3D scene display.

HIL Simulation Platform

□ CopterSim

(b)

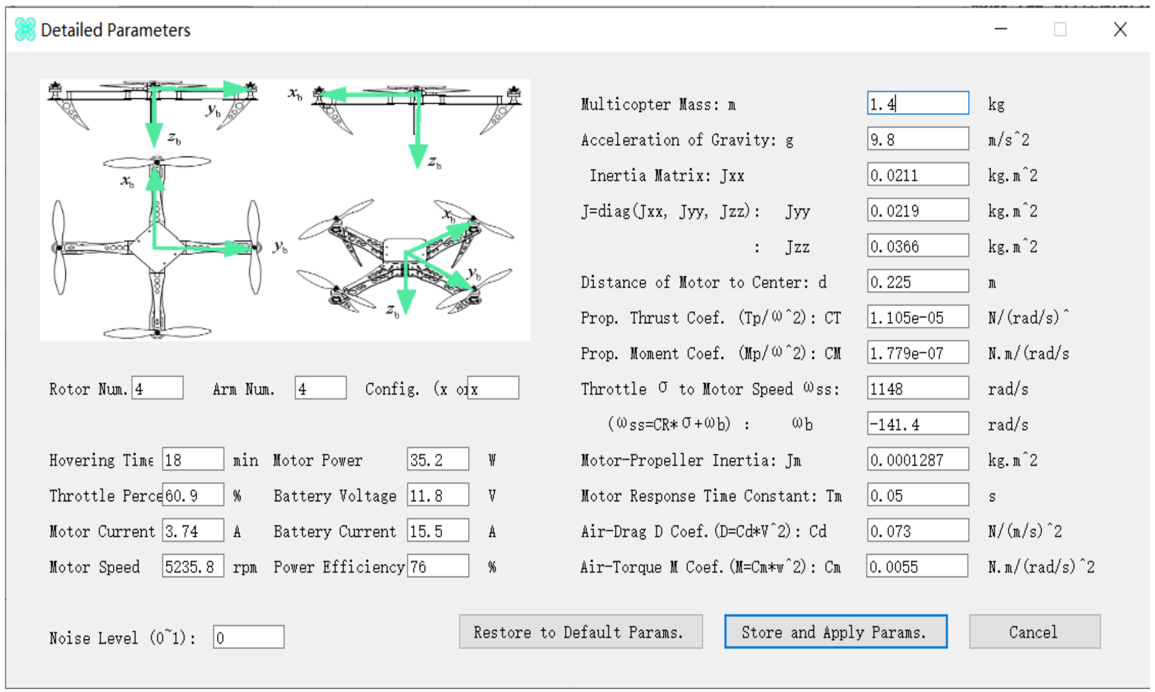



Fig. Model parameter configuration dialog

1. Click the "Model Parameter" button in the middle of the CopterSim UI . The model parameter configuration dialog in the figure on the right will pop up; the model parameters stored in the previous simulation will be displayed here.
2. The parameter dialog mainly includes two parts: the hover information (hover endurance, throttle, output power, motor speed, etc.) and the basic multicopter parameters (total mass, the moment of inertia, size, thrust coefficient, and drag coefficient).

Fig. 模型配置错误提示框

HIL Simulation Platform

☐ CopterSim

(b)

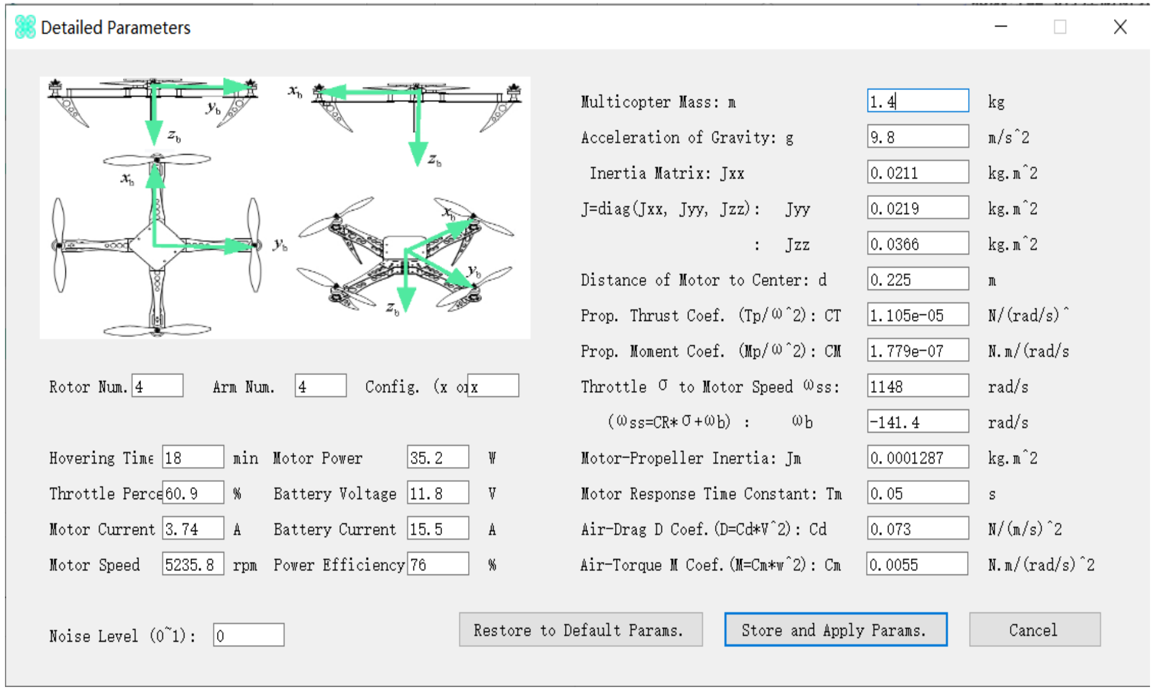


Fig. Model parameter configuration dialog

3. Clicking the "Restore to Default Params" button will restore the model parameters to the default values; clicking the "Save and Apply Params" button will store the current parameters to the database for subsequent HIL simulations.
4. A noise level between 0-1 or larger than 1 can also be selected to represent the noise level of actual sensors. This enables the possibility of testing the anti-interference ability of the designed control algorithms.

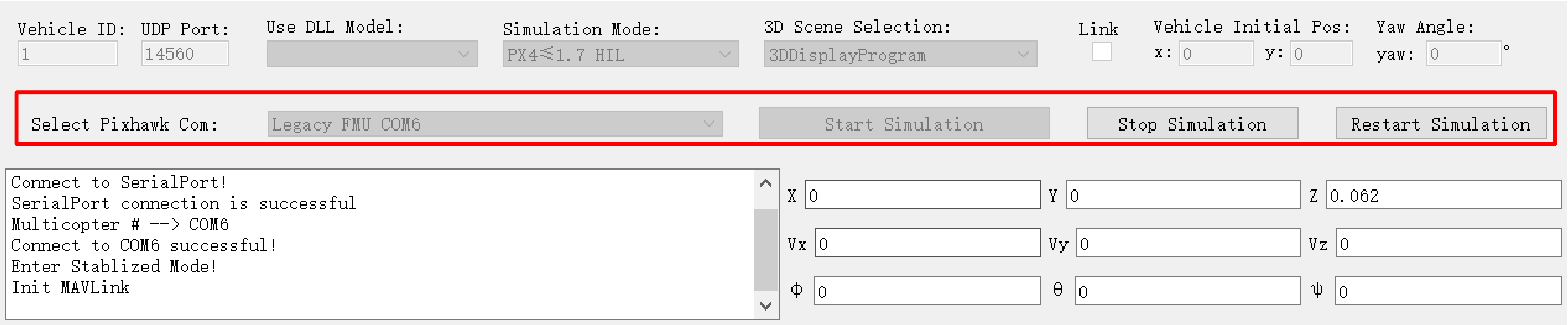
Fig. 模型配置错误提示框

HIL Simulation

Note 1: If you are using Pixhawk to perform HIL simulation, please plug Pixhawk in the computer and wait for about ten seconds until boot process to complete. Then, you can press “Start Simulation” on CopterSim.

Note 2: After the simulation start, you should also wait for about ten seconds until the message box showing “ EKF initialization finished”. Then you can use RC to arm and control the drone.**

☐ CopterSim



Start and Stop Simulation: After the multicopter parameters and the noise level are configured, connect the Pixhawk autopilot with the computer. The serial port of the Pixhawk autopilot will be listed in the “Select Pixhawk Com” drop-down menu. Select the Pixhawk serial port (usually described by the text “FMU”), and click the “Start Simulation” button to start the HIL simulation. clicking the “Stop Simulation” button will stop the HIL simulation, and clicking the “Restart Simulation” will re-initialize the multicopter position and states to their initial values.

HIL Simulation Platform

◻ 3DDisplay

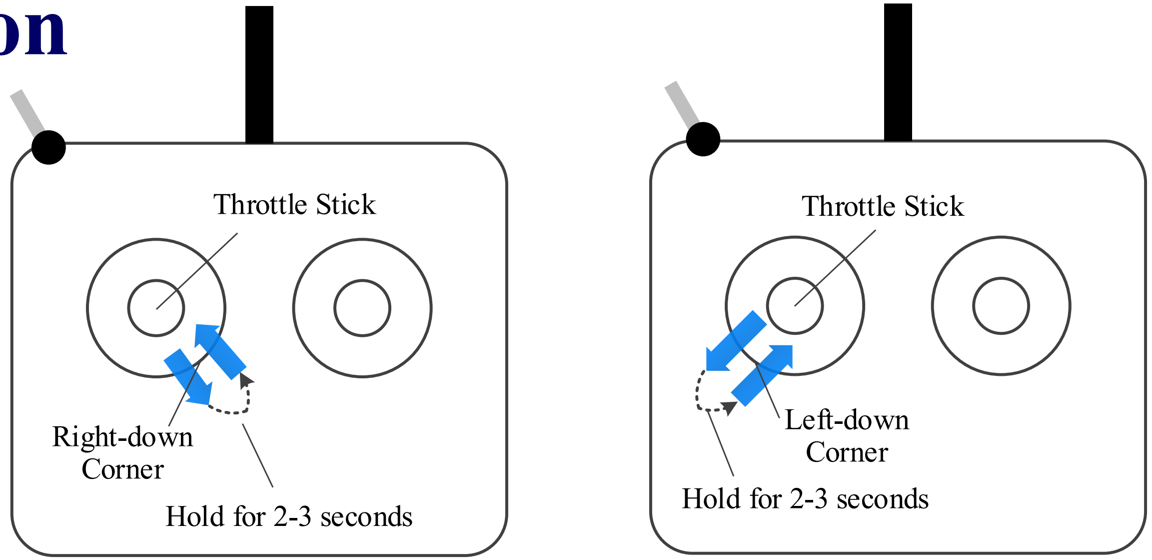
3DDisplay.exe

Fig. Main UI of 3DDisplay software

Double-click the 3DDisplay shortcut on the Windows desktop to open the 3DDisplay software. As shown in figure on the left, the "3D Scene Viewer" on the left side of the 3DDisplay UI presents the current flight status of the multicopter in the 3D scene. The basic flight parameters are displayed in the upper right window of the 3DDisplay UI, including motor speed, position, and attitude information. The flight trajectory of the multicopter is displayed on the lower right window of the 3DDisplay UI.

□ Flight Tests with HIL Simulation

In the HIL simulation platform, when controlling a real multicopter, it is convenient to control the simulated multicopter with a real RC transmitter to perform basic actions, such as arming, taking off, manual flight, landing, etc. The detailed steps are described next.



(a) Arm action (b) Disarm action

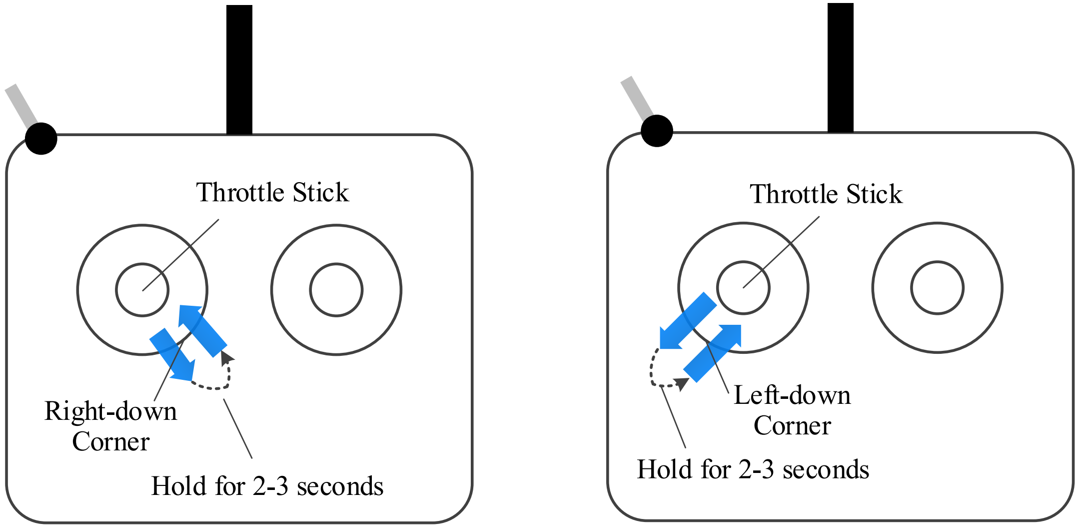
1. Push up the POWER switch to turn on the RC transmitter.

2. Correctly connect the computer with the Pixhawk hardware system (including the Pixhawk autopilot and the RC receiver) and start the HIL simulation in CopterSim according to the procedure mentioned above.

3. As shown in Fig. (a), arm the Pixhawk autopilot by moving the left-hand stick on the RC transmitter (CH3) to the lower-right corner for 2-3 seconds.

□ Flight Tests with HIL Simulation

4. Pixhawk is successfully armed when its LED turns from slow flashing to always on Higher Pixhawk hardware (e.g., Pixhawk 2/3/4/5) starts to discard LED module, so an external I2C LED module is required to observe the lighting effect, and the CopterSim print message “Detect Px4 Armed” is received from Pixhawk. If arming Pixhawk fails, please disconnect all hardware and software and repeat the above steps.



(a) Arm action (b) Disarm action

5. Pull up the left-hand stick on the RC transmitter (CH3) for the multicopter to take off and fly up to a certain altitude. Next, vertically move the left-hand stick to verify the vertical motion control of the multicopter.

6. Horizontally move the left-hand stick on the RC transmitter (CH4) to verify the yaw angle motion control of the multicopter.

□ Flight Tests with HIL Simulation

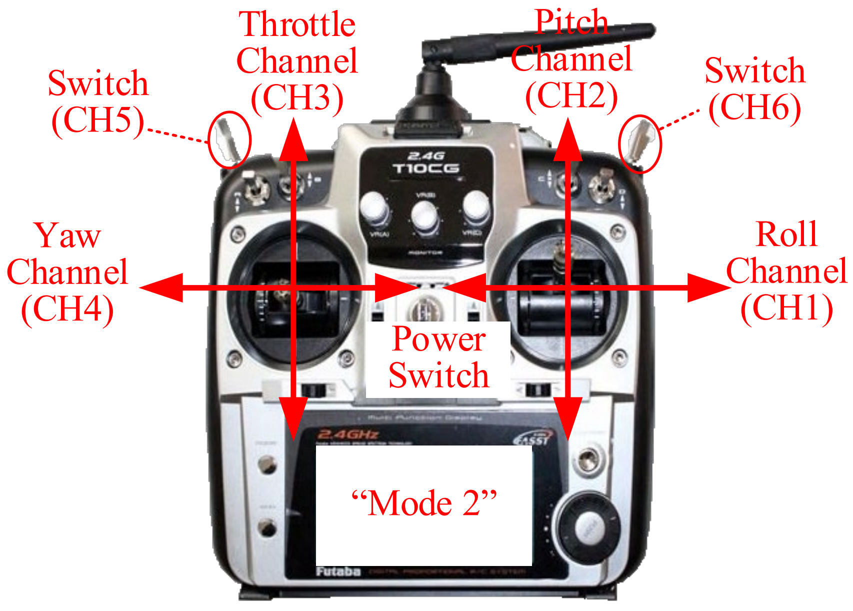
7. Vertically move the right-hand stick on the RC transmitter (CH2) to verify the pitch angle control as well as the forward and backward motion control of the multicopter.

8. Horizontally move the right-hand stick on the RC transmitter (CH1) to verify the roll angle control as well as the left and right motion control of the multicopter.

9. Change the position of the top-right switch on the RC transmitter (CH6) to verify the mode switching control of the multicopter.

10. Pull down the left-hand stick on the RC transmitter (CH3) to land the multicopter to ground.

11. Move the left-hand stick on the RC transmitter (CH3) to the lower-left corner for 2-3 seconds to disarm the Pixhawk.



Throttle : control up-down movement
Pitch : control forward-backward
Yaw : control vehicle head direction
Roll : control left-right movement

12. Click the "Stop Simulation" button on the CopterSim UI to stop the HIL simulation. Then, disconnect all software and hardware connections between the computer and Pixhawk.

Resource

All course PPTs, videos, and source code will be released on our website

https://rflysim.com/en

For more detailed content, please refer to the textbook:

Quan Quan, Xunhua Dai, Shuai Wang. *Multicopter Design and Control Practice*. Springer, 2020

https://www.springer.com/us/book/9789811531378

If you encounter any problems, please post question at Github page

https://github.com/RflySim/RflyExpCode/issues

If you are interested in RflySim advanced platform and courses for rapid development and testing of UAV Swarm/Vision/AI algorithms, please visit:

https://rflysim.com/en/4_Pro/Advanced.html

Thank you!