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

Lesson 04 Experimental Process

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

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

Outline

1. Experimental Process
2. Experimental Procedure for LED Control Experiment
3. Experimental Procedure of Attitude Control Experiment
4. Summary

Experimental Process

- The experimental courses include propulsion system design, modeling, filtering, control, and decision-making.
- each experimental course includes three step-by-step experiments, i.e., a basic experiment, an analysis experiment, and a design experiment.

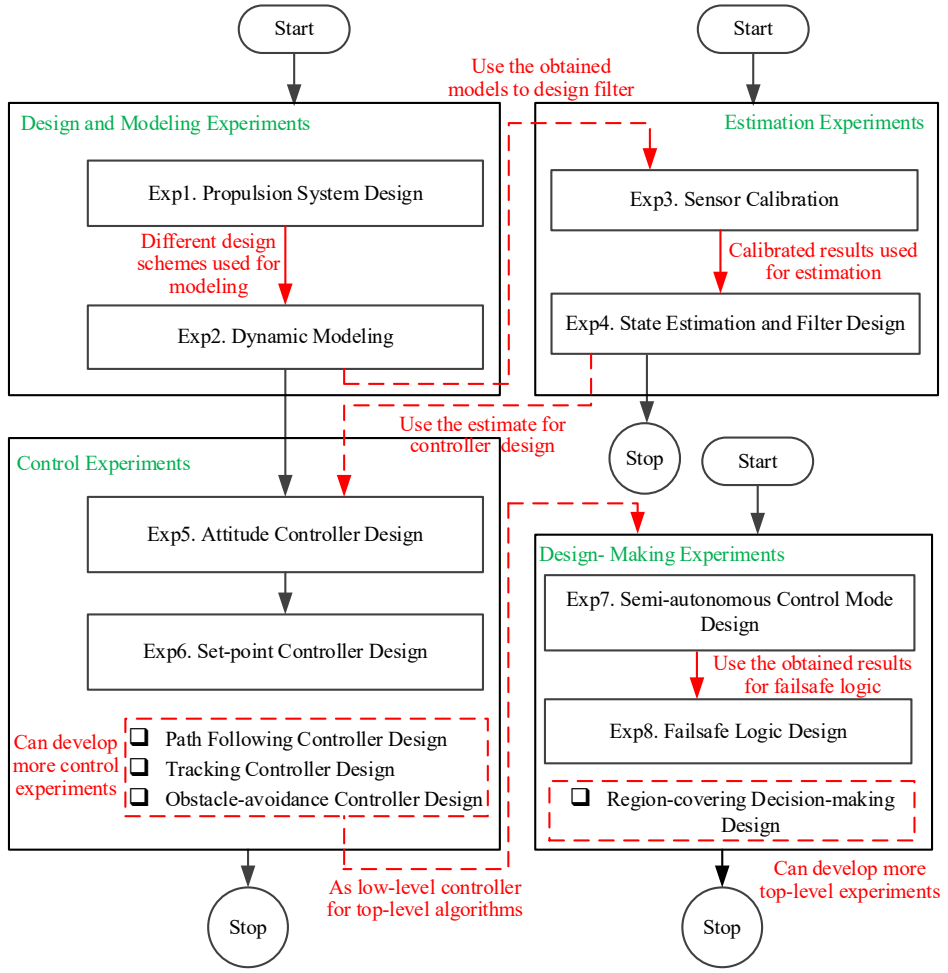


Fig. Progressive routes for eight experiments

Experimental Process

(1) Basic experiment.

Open the given code example, read, and run its source code to observe, record, and analyze the results.

(2) Analysis experiment

Modify the given code example and then run it to collect and analyze the data.

(3) Design experiment

Based on the above two experiments, complete the given design task independently.

For the basic experiments and the analysis experiments, this book provides source code examples to ensure that all readers can complete them easily. Through the above two experiments, the readers will acquire a deep understanding of the theoretical and practical methods. In design experiments, the readers will independently design and verify algorithms by referring to the code examples offered in basic experiments and analysis experiments. The three step-by-step experiments constitute a learning ladder from shallow to deep, which is convenient for readers to reach the final experimental goal.

Experimental Process

Each experiment generally includes the following three phases

1. Simulink-based Algorithm Design and SIL Simulation Phase
2. HIL Simulation Phase
3. Flight Test Phase

(Considering the possible risks, the practical flight, may not be included in the experimental content)

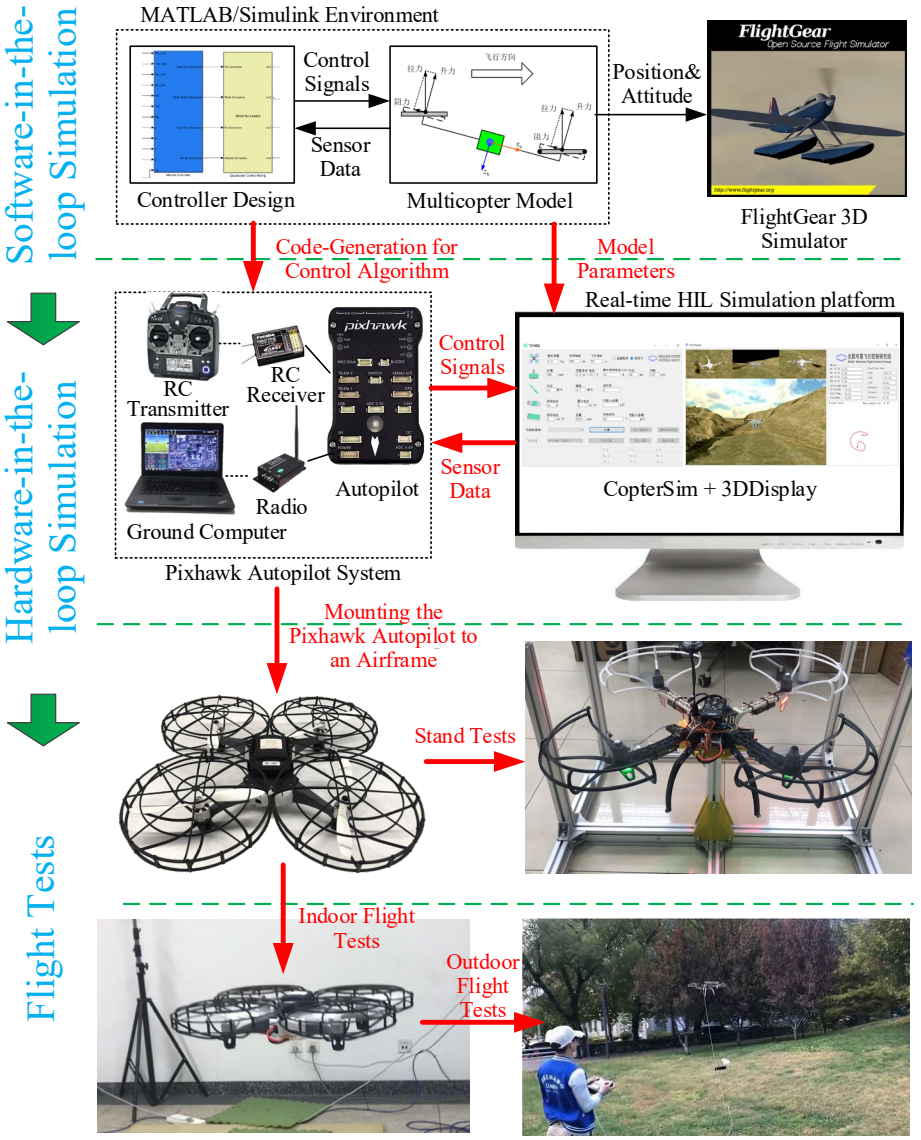


Fig. Overall experimental process

Experimental Process

(1) Simulink-based Algorithm Design and SIL

The entire phase is performed in the MATLAB environment. Readers can design control algorithms in Simulink based on the given multicopter simulation model. The input and output ports of the designed controller and the multicopter simulation model should be correctly connected to ensure that the data transmission is consistent with the real multicopter control system. Similar to a real multicopter system, the multicopter simulation model sends sensor data or estimated states (e.g., attitude, angular rate, position, and speed) to the controller, and the controller sends back the motor PWM control signals to the multicopter simulation model. This sets a closed-loop HIL simulation system. In this phase, readers can analyze the control performance by observing the simulation results, and then modify the controller or re-design it according to the desired performance requirements.

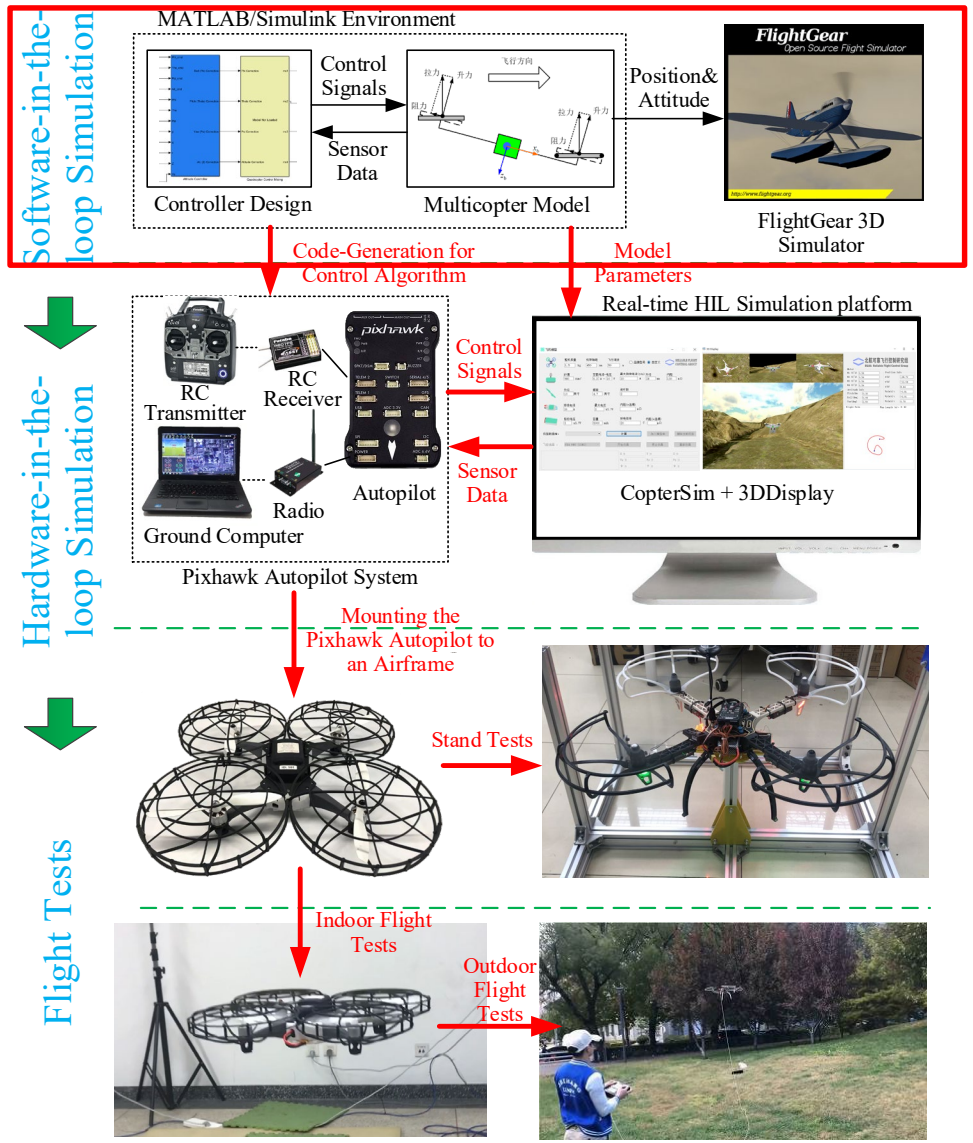


Fig. Overall experimental process

Experimental Process

(2) HIL Simulation Phase

In this phase, readers will import the multicopter model from Simulink to CopterSim, upload the control algorithms from Simulink to a real Pixhawk with the code generation technology, and replace the Simulink virtual data connection with a real USB cable. CopterSim sends sensor data (e.g., accelerometer, gyroscope, and electronic compass) to the Pixhawk system via the USB cable; the PX4 autopilot software in the Pixhawk system filters the sensor data and estimates the multicopter states; next, it sends the estimated multicopter states to the designed controller via the internal uORB message bus; the controller computes the motor PWM control signals and sends them back to the CopterSim via the USB cable to establish a closed-loop HIL simulation system.

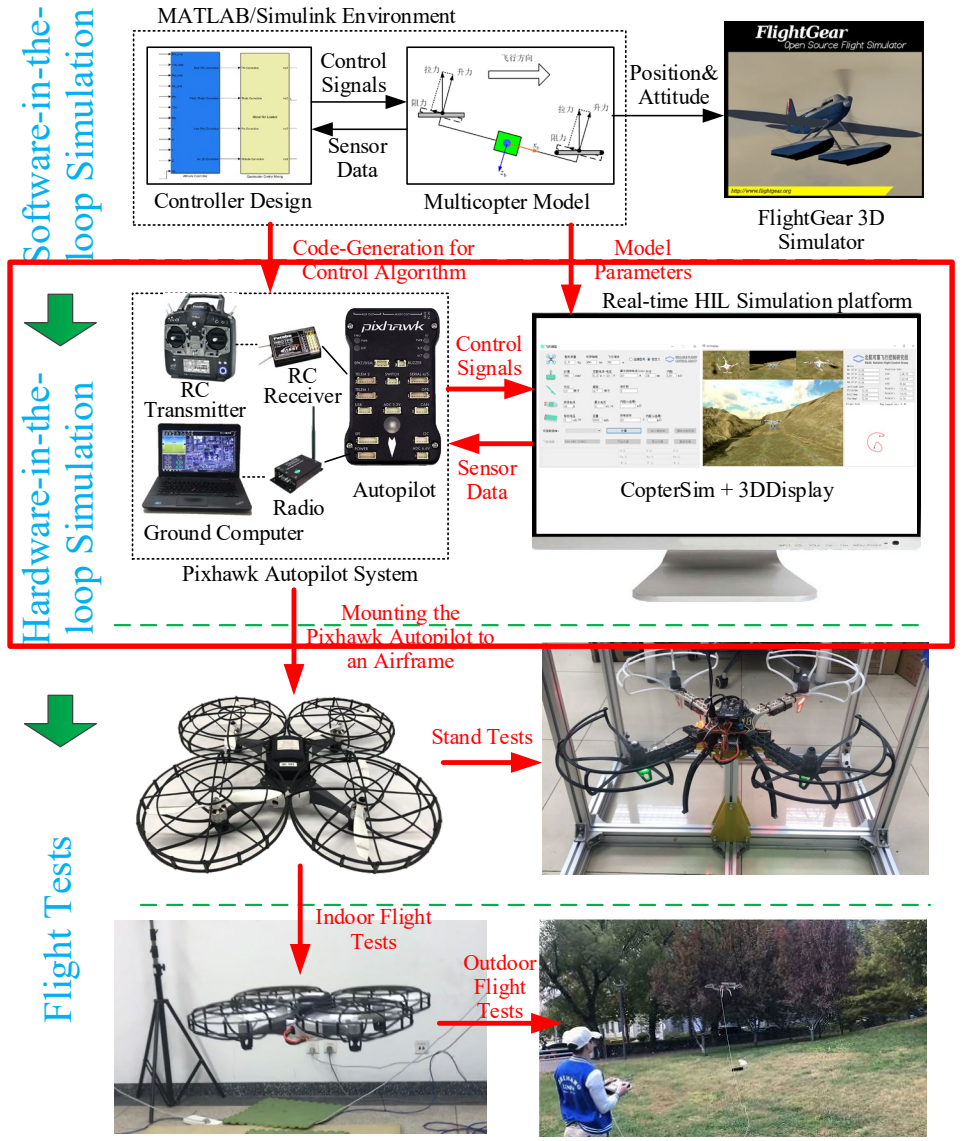


Fig. Overall experimental process

Experimental Process

(2) HIL Simulation Phase

Compared with a SIL simulation system, the multicopter simulation model in CopterSim is running with the same clock speed as the real world to ensure the real-time requirement, and the control algorithms are running in a real embedded system, namely the Pixhawk autopilot, which is better emulates a practical multicopter system. Note that the data transmission delay is inevitable in a real communication system, and the operating environments of the simulation model, as well as the control algorithms in the HIL simulation system, are difficult to be consistent with the SIL simulation system. Therefore, the controller parameters may need to be slightly tuned based on the simulation results. In any case, this is consistent with the system development process of a real multicopter system.

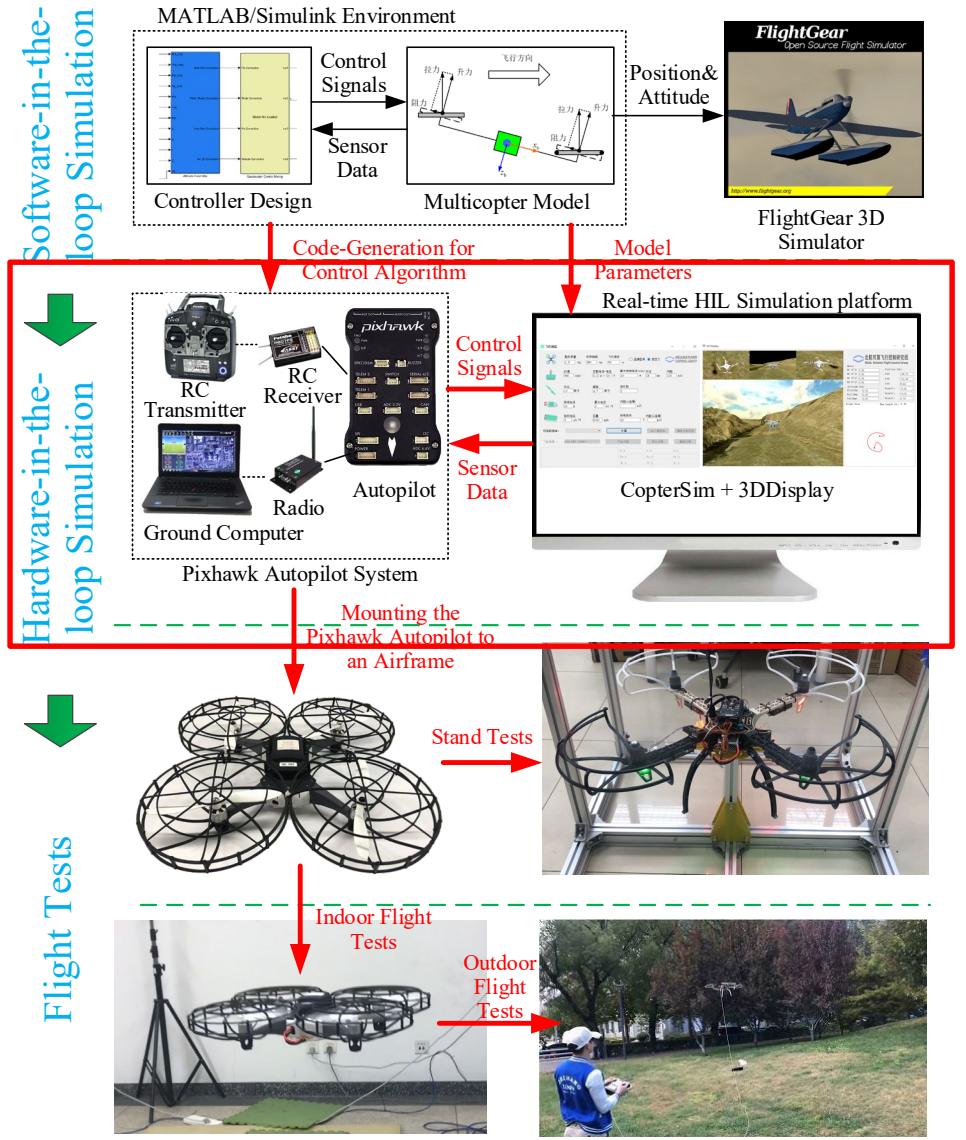
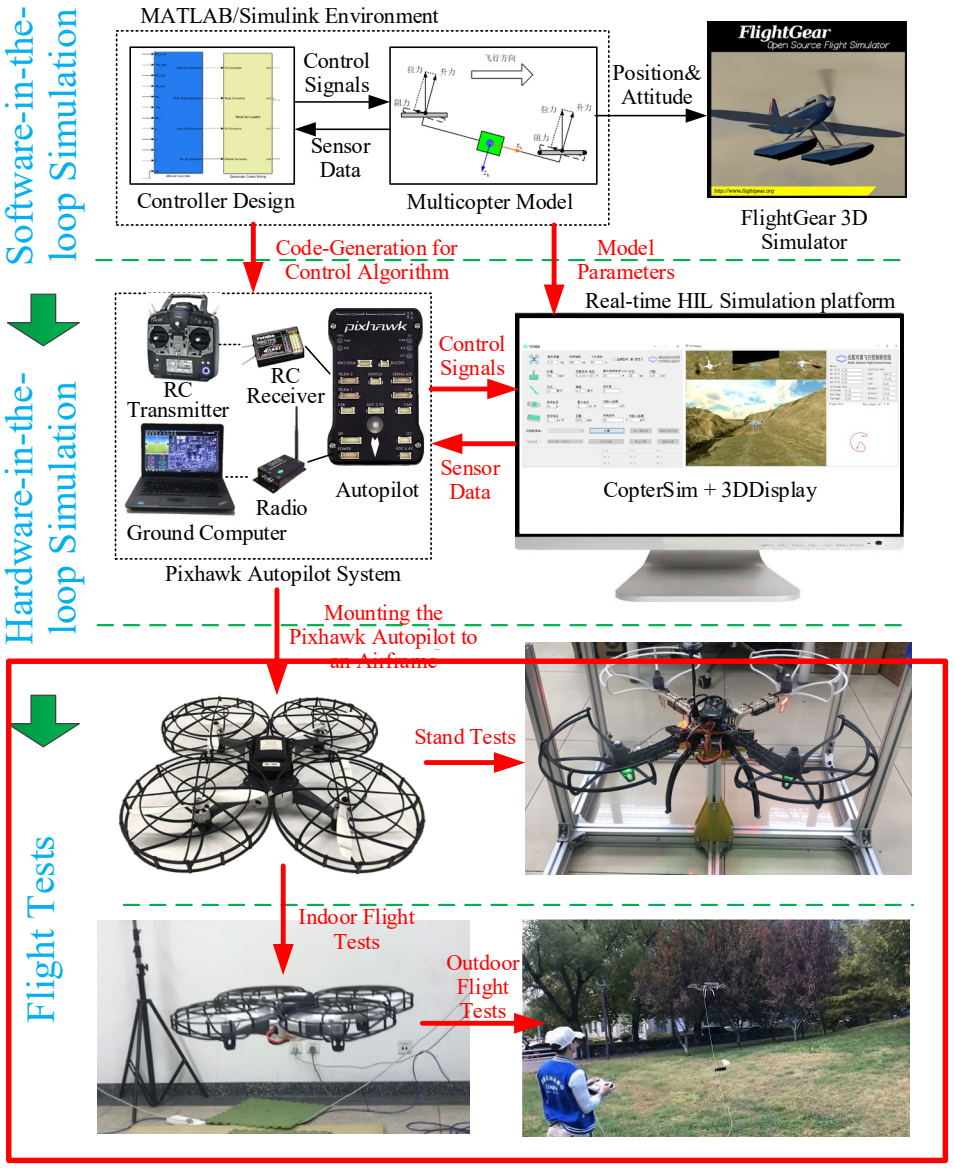


Fig. Overall experimental process

Experimental Process

(3) Flight Test Phase

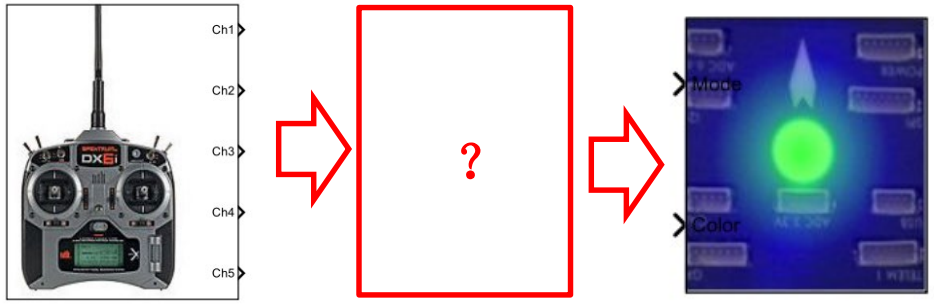
In this phase, the multicopter simulation model in CopterSim is further replaced with a real multicopter hardware system; the sensor data are replaced with signals from real sensor chips by sensing the multicopter motion and states and the control signals are directly output to the Pixhawk I/O ports to control the motors. Note that it is difficult to ensure that the simulation model used in both the SIL simulation and the HIL simulation is identical to the dynamics of a real multicopter system. Consequently, the controller parameters may need further slight tuning according to the experimental results.

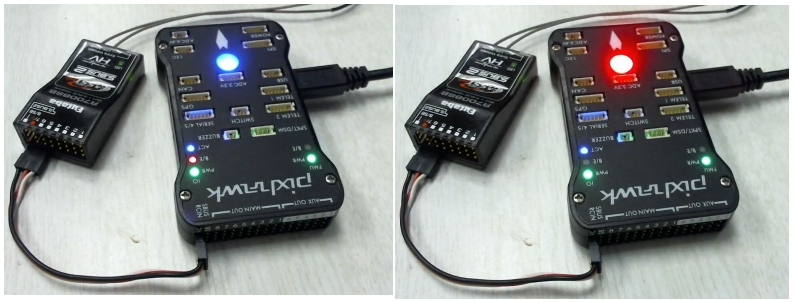


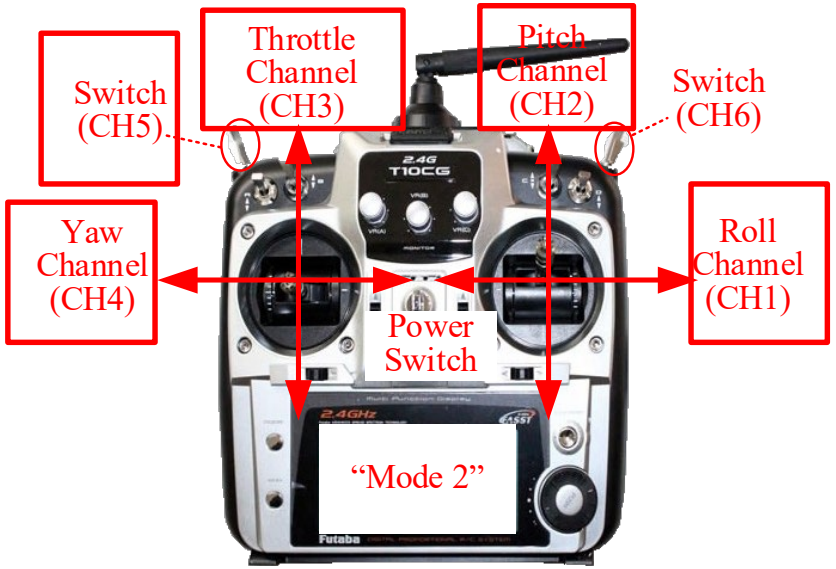
Procedure for LED Control Experiment

Experiment Objectives

Use two channels from CH1 to CH5 of the RC transmitter to control the LED light on Pixhawk in two different colors and two different modes.







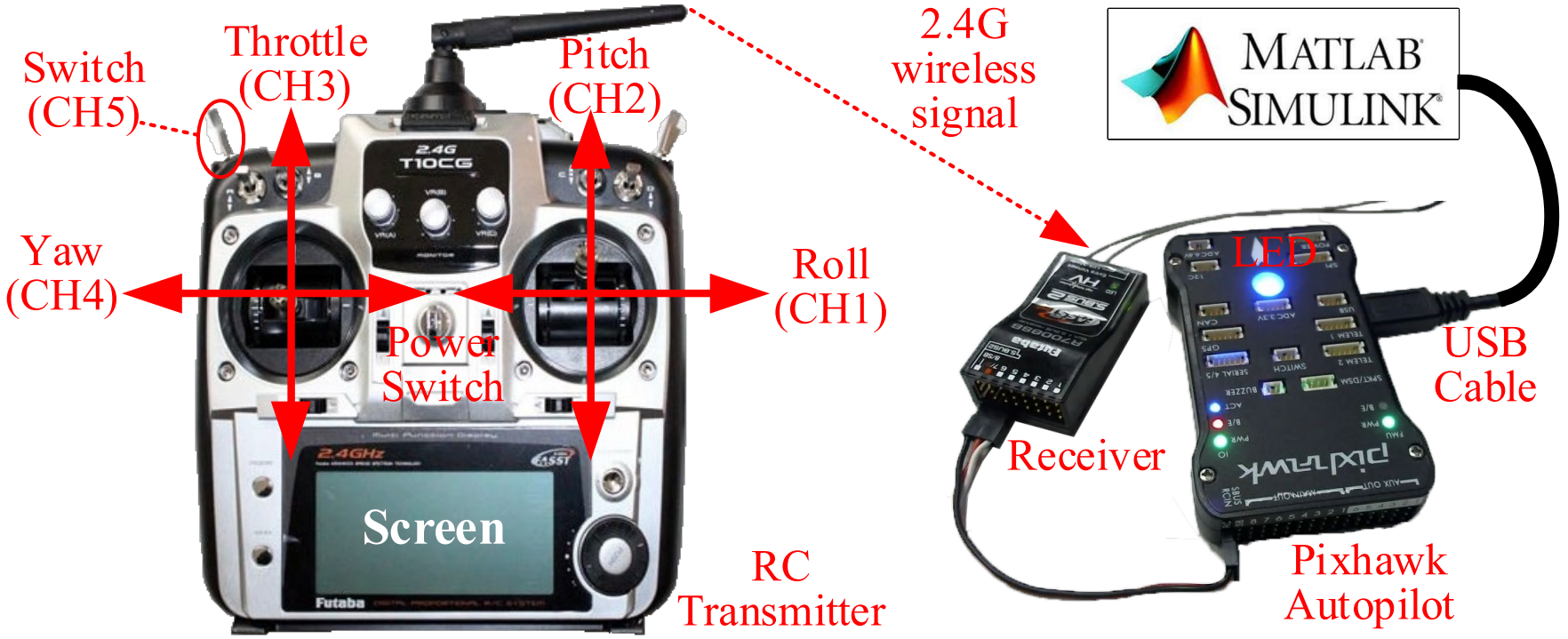
Throttle : control up-down movement
Pitch : control forward-backward
Yaw : control vehicle head direction
Roll : control left-right movement

Procedure for LED Control Experiment

☐ Experiment Objectives

Use two channels from CH1 to CH5 of the RC transmitter to control the LED light on Pixhawk in two different colors and two different modes.

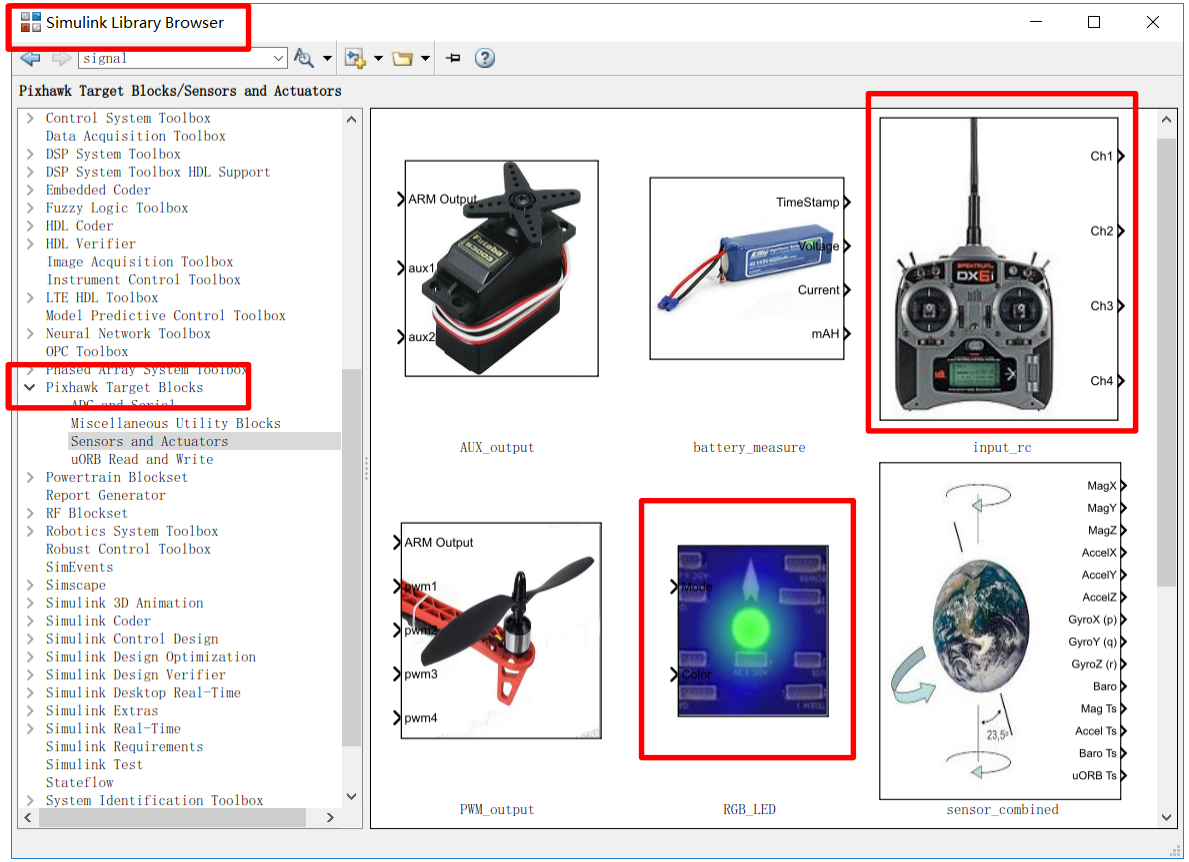
Hardware connection diagram



Procedure for LED Control Experiment

☐ Design Control Model

(1) Create and open a new Simulink model file. find the “RGB_LED” module in the “Pixhawk Target Blocks” toolbox, which is in the “Library Browser” of Simulink, and drag it to the new created Simulink file. The RC transmitter module “input_rc” is also dragged into the Simulink as the input signals to control the LED light. A Simulink example file completing the whole process of configuration is available in “e0\2.PSPOfficialExps\px4demo_input_rc.slx”.



Procedure for LED Control

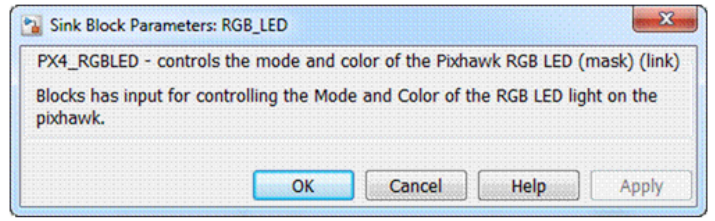
☐ Design Control Model

(2) See the "RGB_LED" module description. Double-click the "RGB_LED" module and click "help" to view the predefined control enumeration variables. This module can be used to control the patterns and colors of LED lights on Pixhawk.

Note: These variables have been registered as MATLAB global parameters when installing the PSP toolbox so that they can be directly applied. For example, an LED light fast blinking mode can be obtained by transferring the variable "RGBLED_MODE_ENUM.SL_MODE_BLINK_FAST" with the "Constant" module to the "Mode" port of the LED module (the upper input port of the LED module)

Pixhawk Target Block: RGB_LED

This block gives the user control over various lighting modes of the RGB LED available on the PX4 hardware.



This block accepts 2 inputs: Mode and Color. These are enumeration data types. You can find out what values are valid in the MATLAB command window by typing:

RGBLED_MODE_ENUM

Value
SL_MODE_OFF (0)
SL_MODE_ON (1)
SL_MODE_DISABLED (2)
SL_MODE_BLINK_SLOW (3)
SL_MODE_BLINK_NORMAL (4)
SL_MODE_BLINK_FAST (5)
SL_MODE_BREATHE (6)

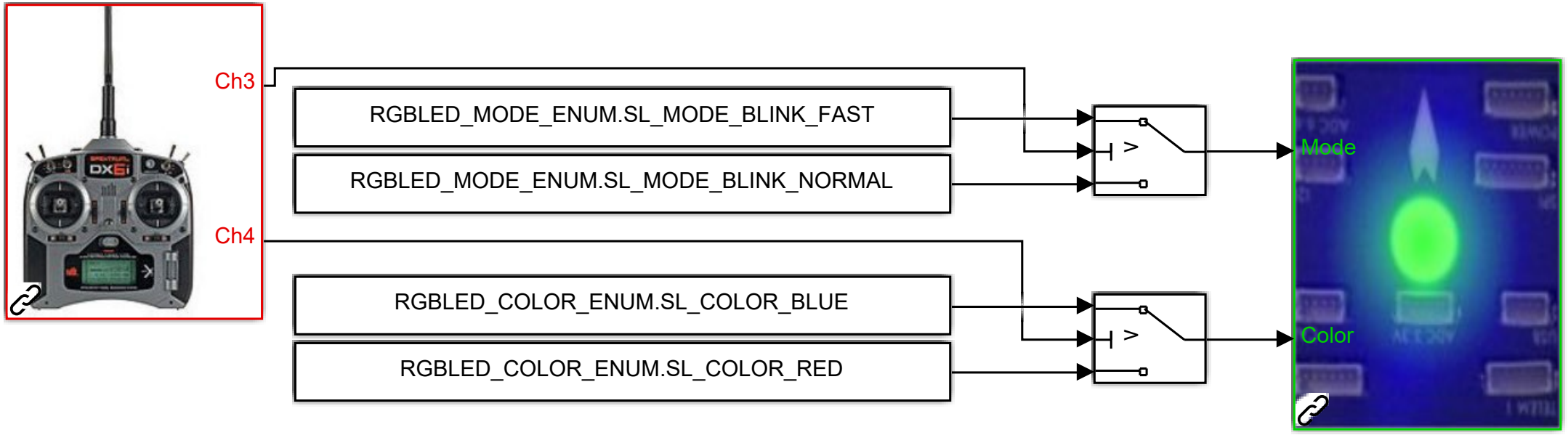
RGBLED_COLOR_ENUM

Value
SL_COLOR_OFF (0)
SL_COLOR_RED (1)
SL_COLOR_GREEN (2)
SL_COLOR_BLUE (3)
SL_COLOR_YELLOW (4)
SL_COLOR_PURPLE (5)
SL_COLOR_AMBER (6)
SL_COLOR_CYAN (7)
SL_COLOR_WHITE (8)

Procedure for LED Control Experiment

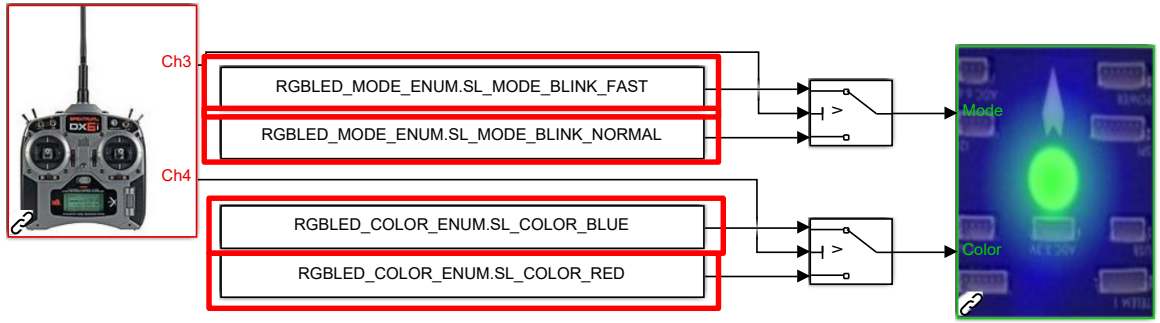
☐ Design Control Model

3）Controller design. The data range of the PWM signals received by Pixhawk is 1100-1900. Here, 1500 is selected as the Switch quantity of Switch module. We will use two channels of the RC transmitter in this experiment to control the mode and color of the LED light.



Procedure for LED Control Experiment

☐ Design Control Model

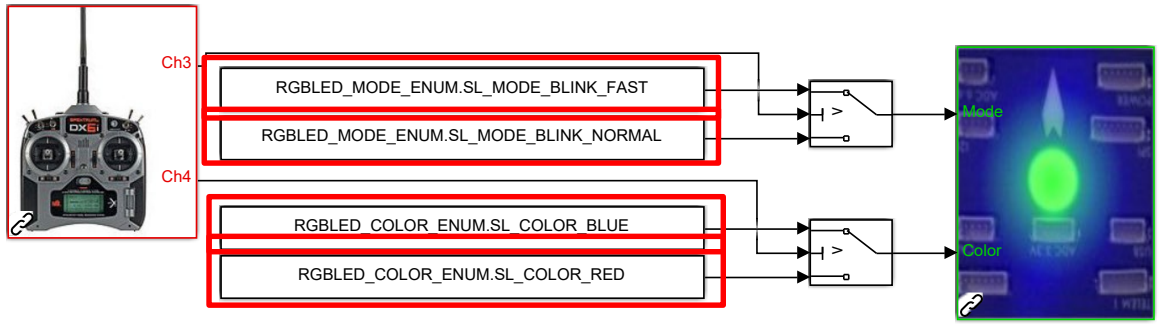


Use the CH3 channel of the RC to change the blink mode of the LED light.

1. When the throttle stick is turned to the upper side, i.e., when the PWM value of the CH3 channel is higher than 1500 microseconds (abbreviated as CH3>1500), then the “Mode” port receives an “RGBLED_MODE_ENUM.SL_MODE_BLINK_FAST” variable corresponding to a fast blinking mode;
2. when CH3 ≤ 1500, the “Mode” port receives the “RGBLED_MODE_ENUM.SL_MODE_BLINK_NORMAL” variable corresponding to a slow blinking mode.

Procedure for LED Control Experiment

☐ Design Control Model



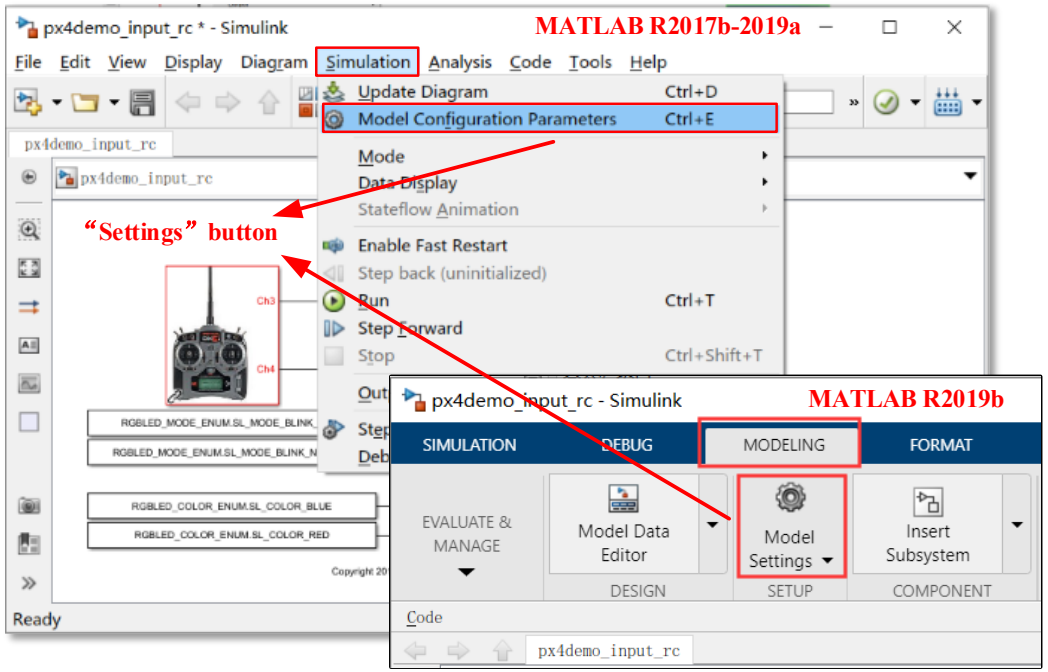
Use the CH4 channel of the RC transmitter to change the color of the LED light.

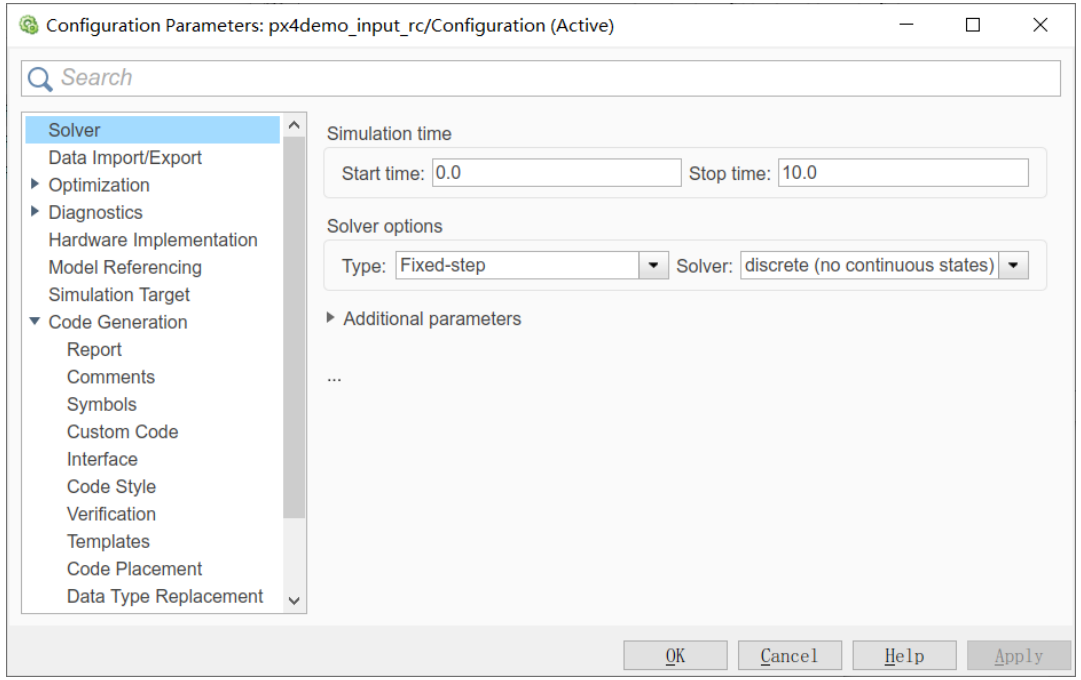
1. When CH4>1500, the “Color” port receives an “RGBLED_COLOR_ENUM.COLOR_RED” variable corresponding to red color;
2. when CH4 ≤ 1500, the “Color” port receives an “RGBLED_COLOR_ENUM.COLOR_BLUE” variable corresponding to blue color.

Procedure for LED Control Experiment

☐ Controller Code Generation and Firmware Uploading

(1) For MATLAB 2017b-2019a, as shown in figure, click the "Simulation"-"Model Configuration Parameters" option on the Simulink menu bar to enter the Simulink setting dialog; for MATLAB 2019b and above, click the "Settings" button. The obtained setting dialog is presented in the figure on the right.

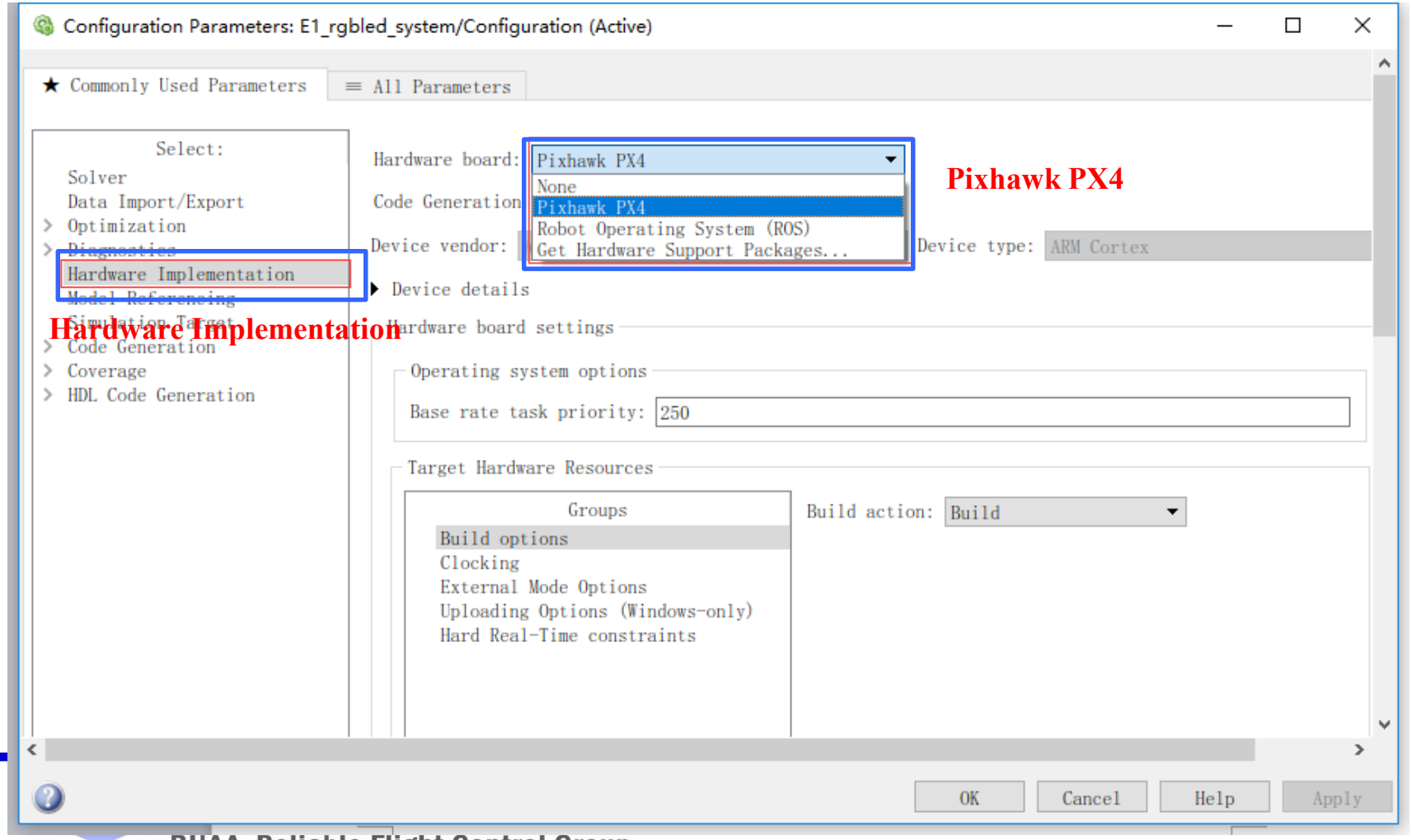




Procedure for LED Control Experiment

☐ Controller Code Generation and Firmware Uploading

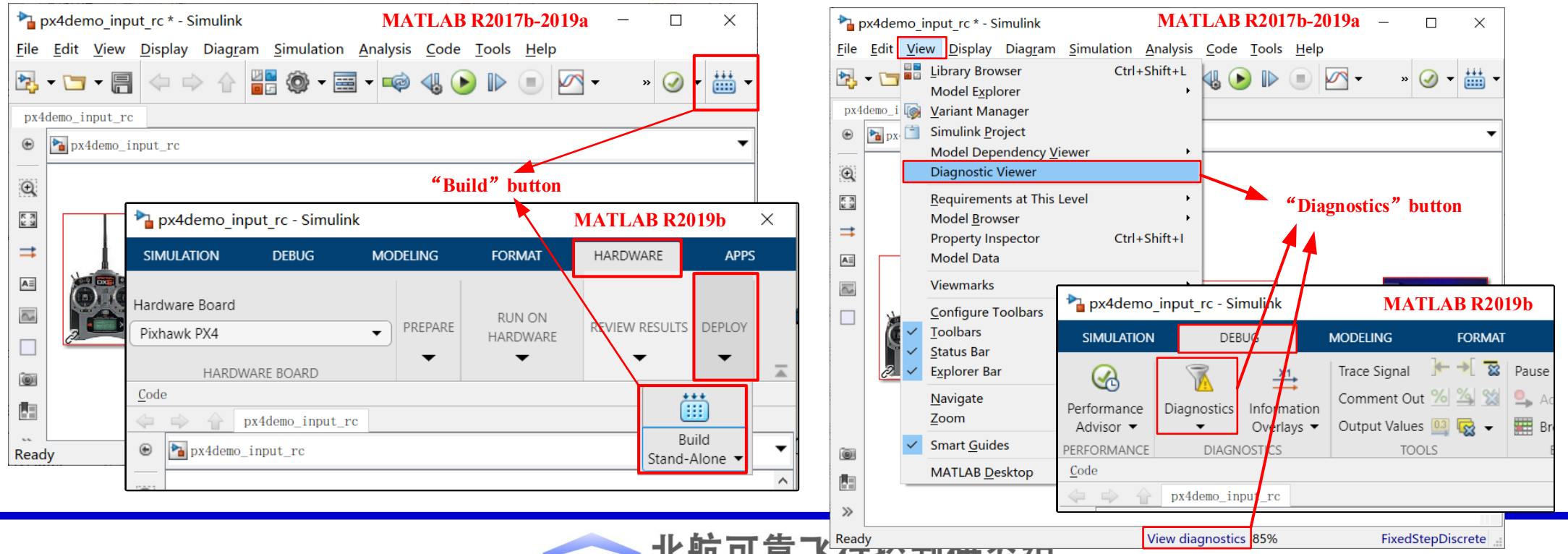
(2) Select the target hardware. As shown in Figure, set the option "Hardware Implementation" - "Hardware Board" to "Pixhawk PX4".



Procedure for LED Control Experiment

☐ Controller Code Generation and Firmware Uploading

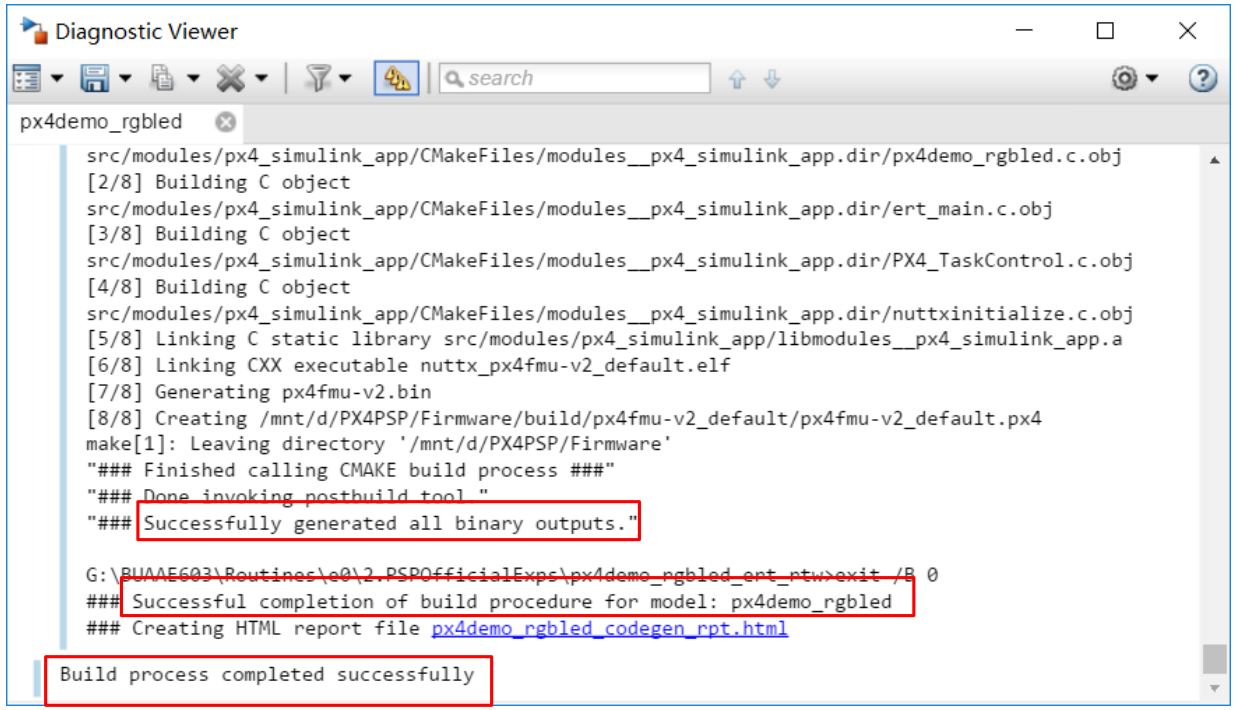
(3) Compile the model. As shown in lower-left figure, compile the designed controller into the PX4 firmware file by clicking the "Build" button on the Simulink toolbar. Then, click the "Diagnostics" button can open the Diagnostics Viewer window.

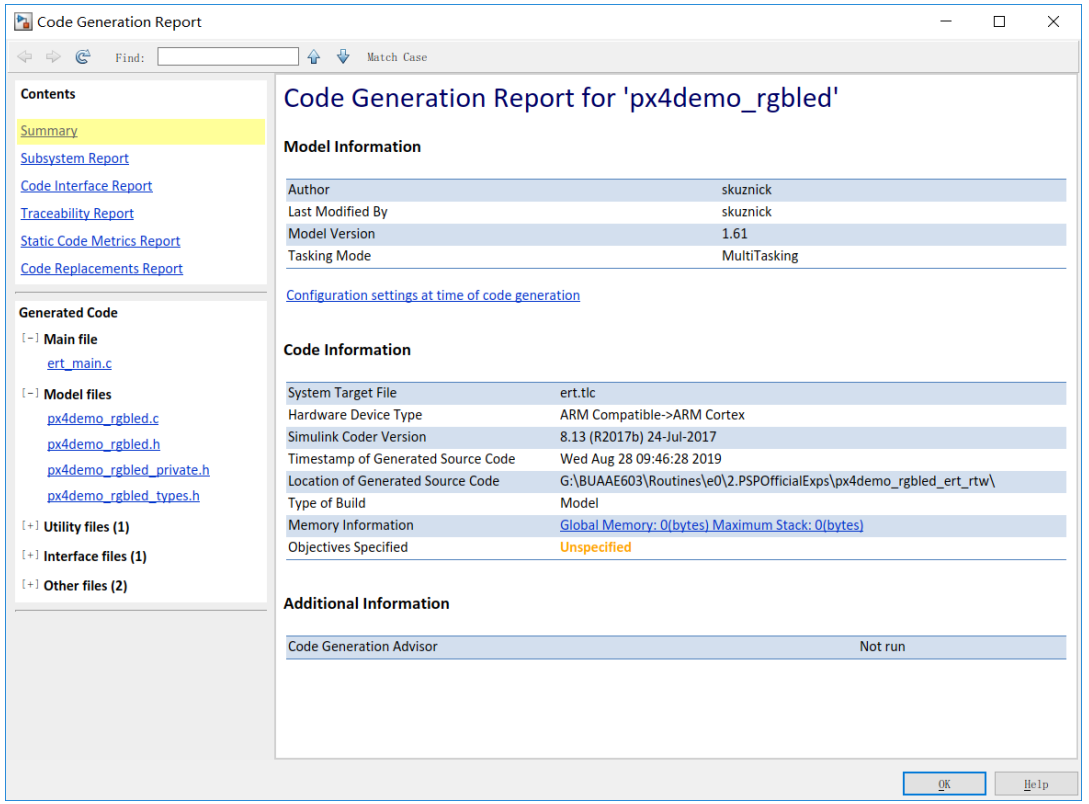


Procedure for LED Control Experiment

☐ Controller Code Generation and Firmware Uploading

The compiling process can be observed in the Diagnostics Viewer window. A code generation report shown in lower-right figure can also be obtained.



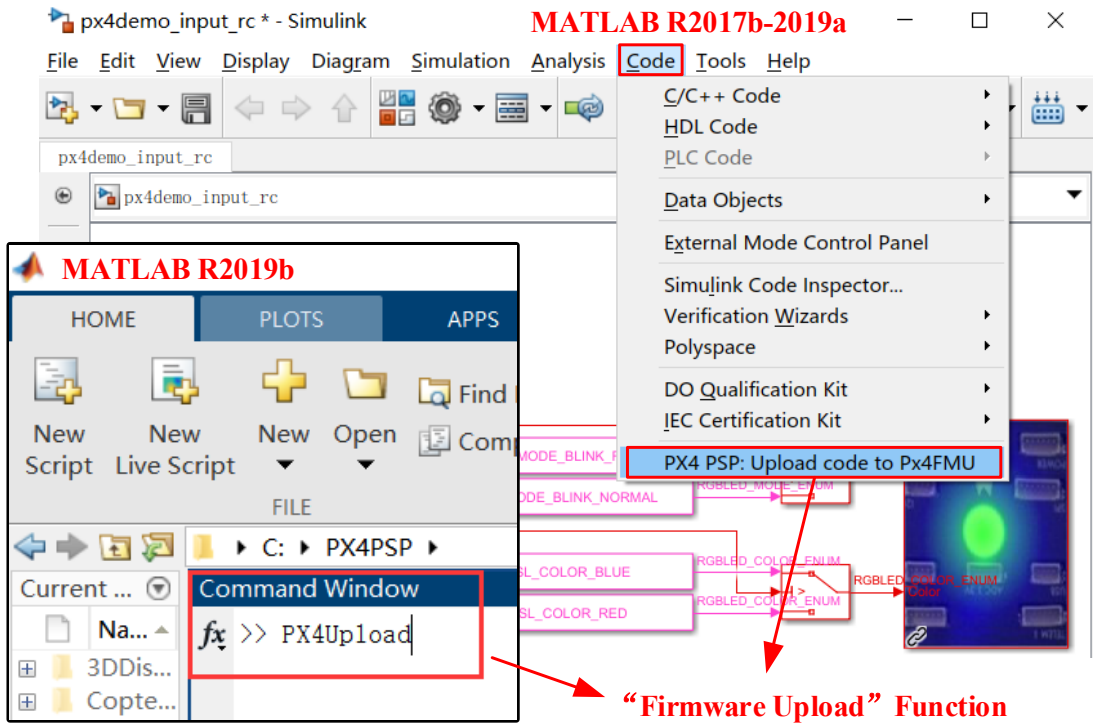


Procedure for LED Control Experiment

☐ Controller Code Generation and Firmware Uploading

(4) Upload the firmware. Use the one-key upload function provided by the PSP toolbox to upload the PX4 firmware file. The specific steps are presented as follows.

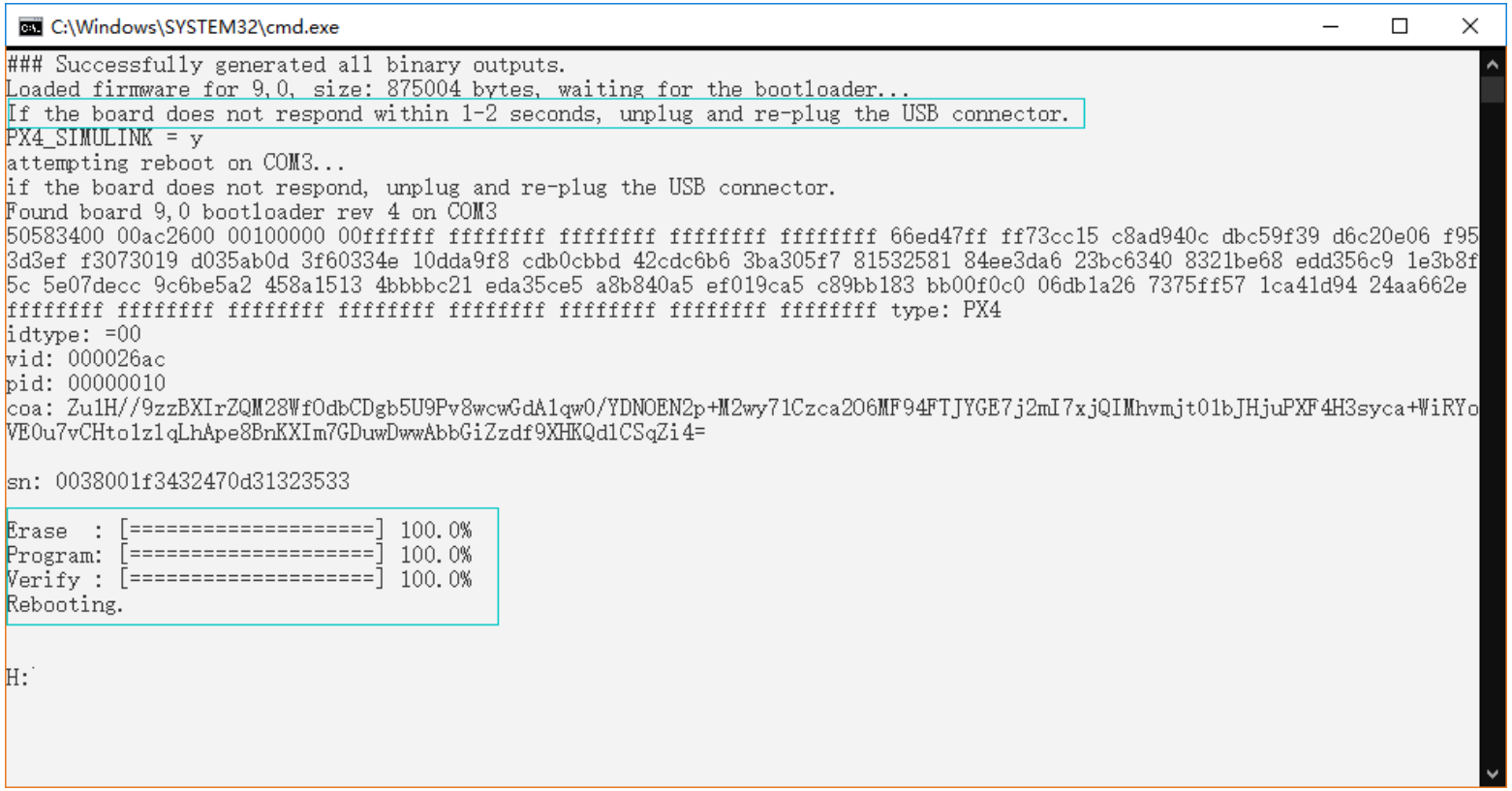
1. Connect the MicroUSB port of the Pixhawk hardware with the computer by using a USB cable.
2. For MATLAB 2017b-2019a, as shown in the figure on the right, click the "Code"-"PX4 PSP: Upload code to Px4FMU" in the Simulink menu bar; For MATLAB 2019b and above, input the "PX4Upload" command in the MATLAB "Command Window" according to the figure to upload the firmware.



Procedure for LED Control Experiment

☐ Controller Code Generation and Firmware Uploading

3. Simulink will automatically recognize the Pixhawk autopilot and start to upload and deploy the PX4 firmware file. The uploading and deploying process is successfully completed when the progress bar reaches 100%. Note that Pixhawk may need to be re-plugged in some cases.

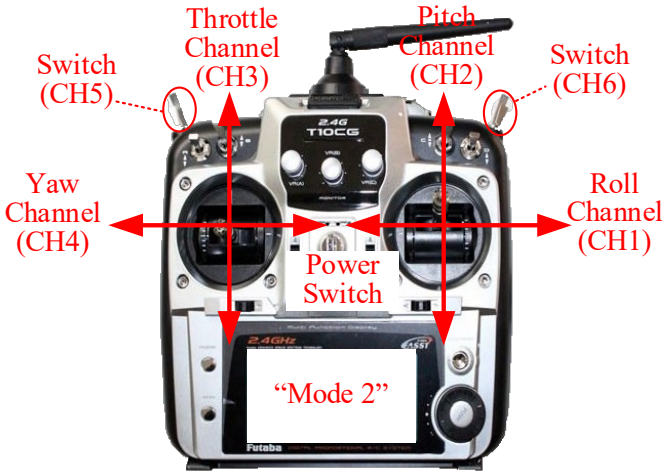


Procedure for LED Control Experiment

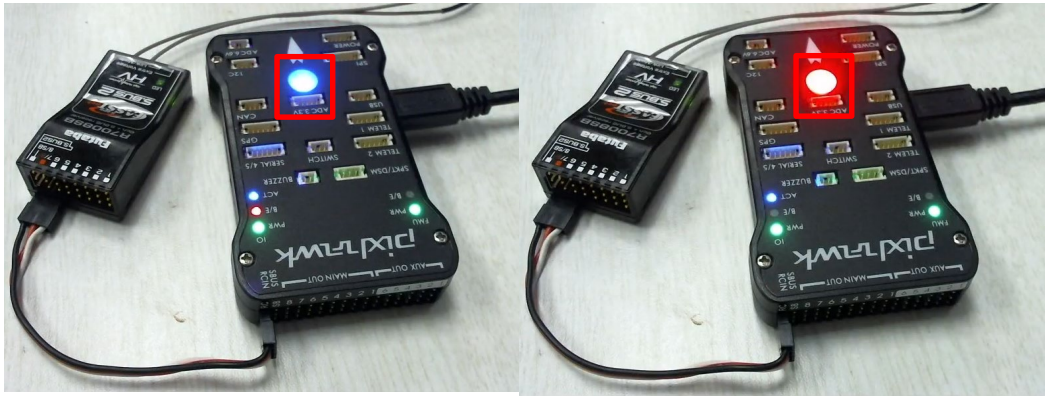
☐ Experiment Result

By default, when the RC transmitter does nothing, the LED light is slowly blinking in blue. Do the following steps to verify the experimental results.

1. When the left-hand stick of the RC transmitter shown in upper-right figure is placed in the upper-right position (CH3>1500 and CH4>1500), the LED light on Pixhawk is quickly blinking in blue.
2. When the throttle stick of the RC transmitter is placed in the upper-left position (CH3>1500 and CH4<1500), the LED is quickly blinking in red.



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



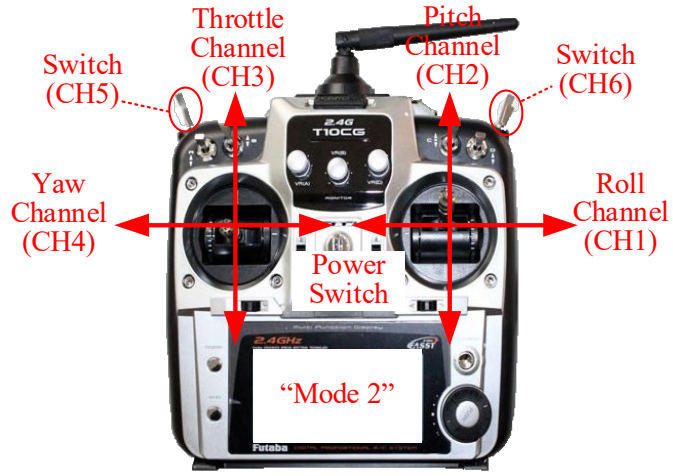
Procedure for LED Control Experiment

☐ Experiment Result

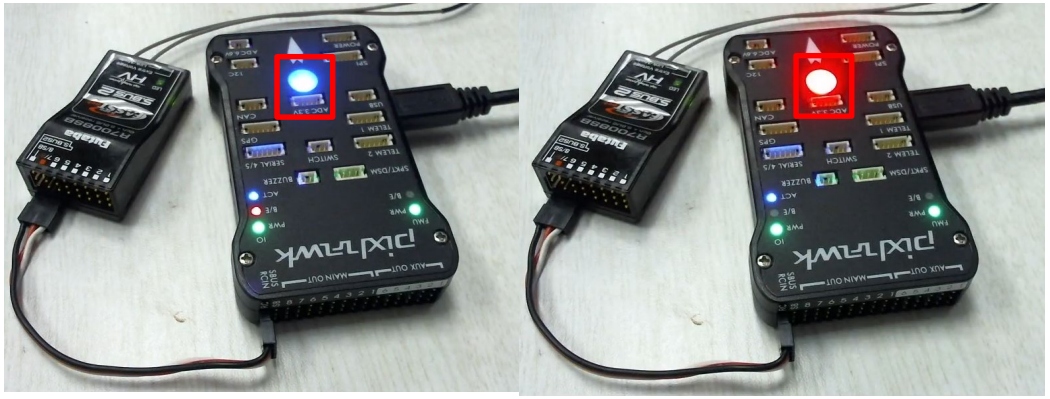
By default, when the RC transmitter does nothing, the LED light is slowly blinking in blue. Do the following steps to verify the experimental results.

3. When the throttle stick of the RC transmitter is placed in the lower-left position (CH3<1500 and CH4<1500), the LED is slowly blinking in red.

4. When the throttle stick of the RC transmitter is placed in the lower-right position (CH3<1500 and CH4>1500), the LED is slowly blinking in blue.



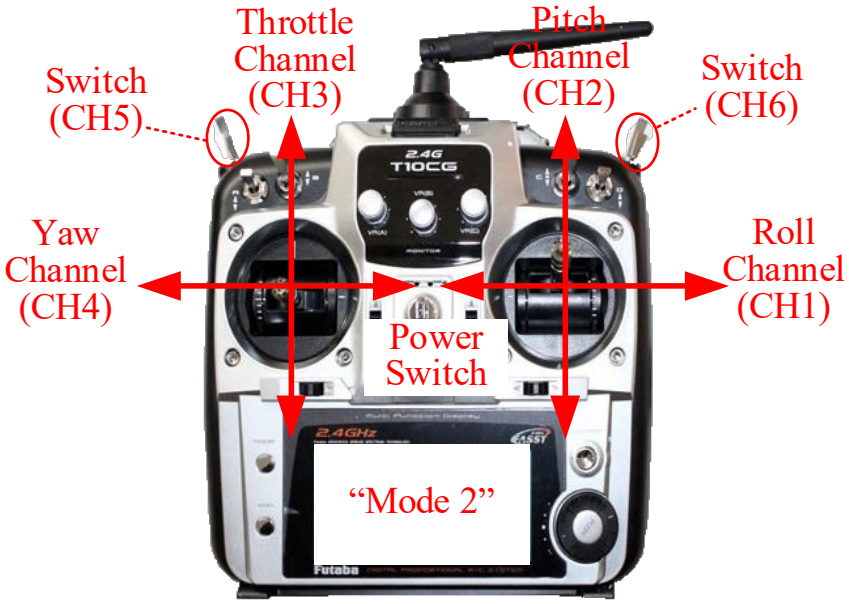
Throttle : control up-down movement
Pitch : control forward-backward
Yaw : control vehicle head direction
Roll : control left-right movement



Noteworthy, you may need an external I2C LED module to observe LED effect for Pixhawk 2 (Cube), and an external GPS module with LED to observe the effect for Pixhawk 4 (fmu-v5)

Procedure of Attitude Contr

☐ Algorithm Design and SIL Simulatio

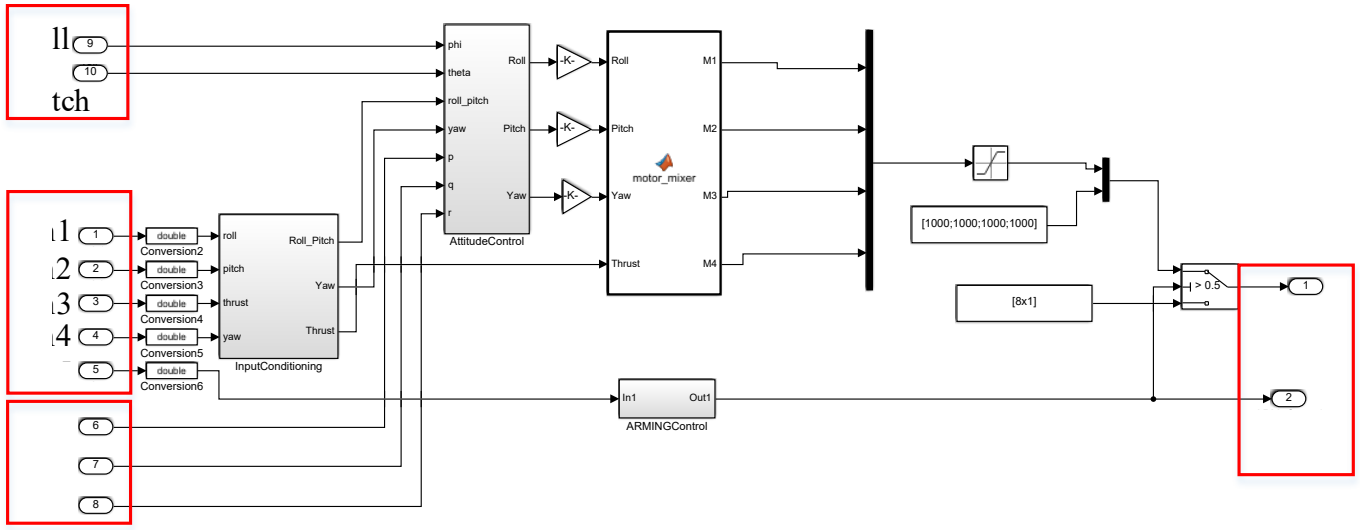


(1) Step 1: controller design

Create a new Simulink file and design a multicopter attitude controller in it. The design requirements for the controller are in the following.

1) Input data

1. CH1-CH5 channel signals of the RC transmitter (see the upper-right figure), which correspond to the "ch1"-"ch5" input ports in the figure on the right. The actual data approximately range from 1100 to 1900, so calibration or dead zones are required in processing the RC data.



For simplicity, an example of a well-designed attitude controller is available in "e0\3.DesignExps\Exp1_AttitudeController.slx".

Procedure of Attitude Control Experiment

☐ Algorithm Design and SIL Simulation Phase

2. Multicopter angular velocity (corresponding to the “p”, “q”, “r” input ports in figure, unit: rad/s). The above three inputs represent the velocity rotating around the x-axis of the body, the velocity rotating around the y-axis of the body, and the velocity rotating along the z-axis of the body.
3. Multicopter Euler angles (unit: rad). Here, the roll angle and the pitch angle (corresponding to the “roll” and “pitch” input ports) are mainly considered, and the yaw control is temporarily not considered in this experiment.

2) Output data

1. PWM control signals of four motors (corresponding to the “PWM” output port. The data range is 1000-2000.
2. Identifier for the armed state (corresponding to the “ARM_Control” output port. The data type is Boolean.

Procedure of Attitude Control Experiment

☐ Algorithm Design and SIL Simulation Phase

3) Expected effects

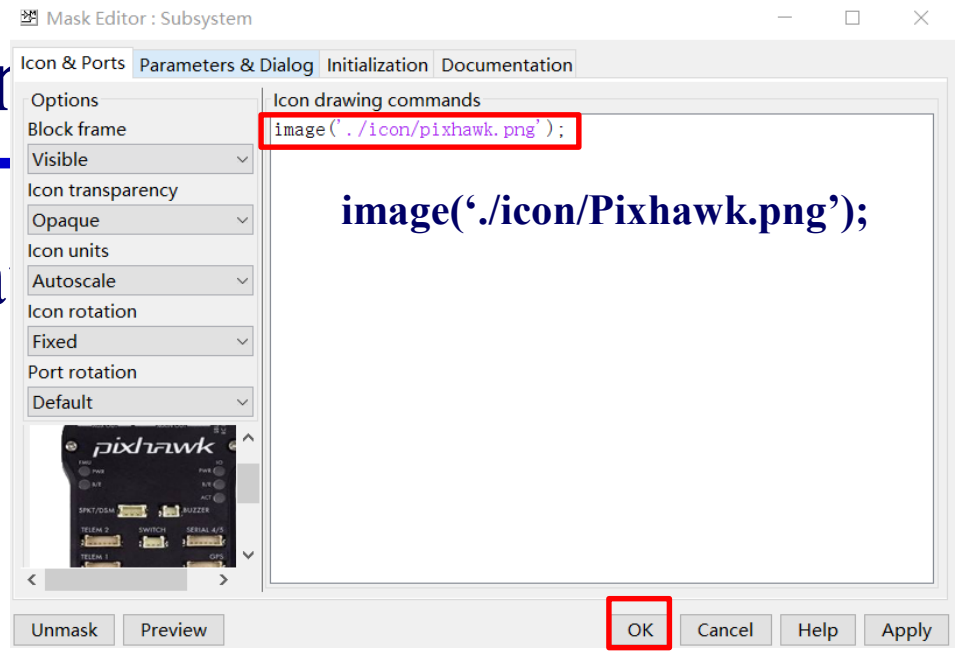
1. The throttle stick (CH3 on the RC transmitter) controls multicopters to perform the upward-and-downward movement.
2. Push up the pitch stick (CH2 < 1500) to control the multicopter flying forward.
3. Move the roll stick leftward (CH1 < 1500) to control the multicopter flying leftward.
4. Pull back (or down) the three-position switch (CH5 > 1500) to disarm the multicopter.

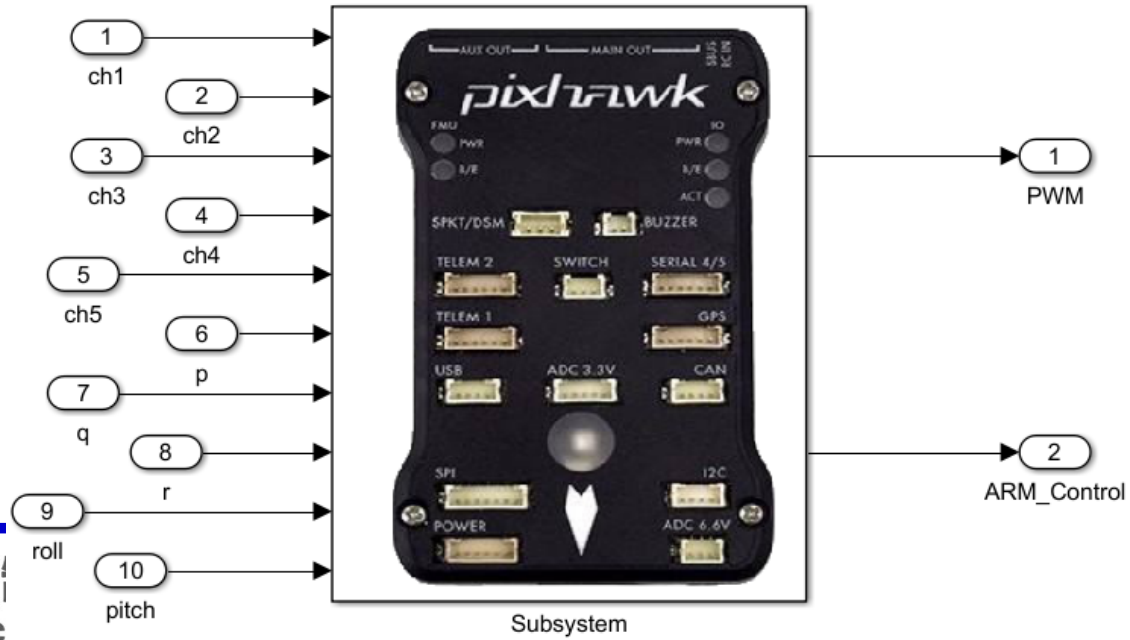
Procedure of Attitude Cor

☐ Algorithm Design and SIL Simula

(2) Step 2: generate the controller subsystem

1. Select all the components with the mouse (or simultaneously press the keys CTRL + A), and right-click the mouse, choosing “Create Subsystem From Selection” to pack the controller to a subsystem in Simulink.
2. Right-click the obtained subsystem and click “Mask” - “Create Mask” to open the mask setting .
3. Then, enter text “image(‘./icon/Pixhawk.png’);” in the “Icon drawing commands” input box.
4. Finally, click the “OK” button and adjust the positions of the input and output ports of the obtained subsystem to get a subsystem



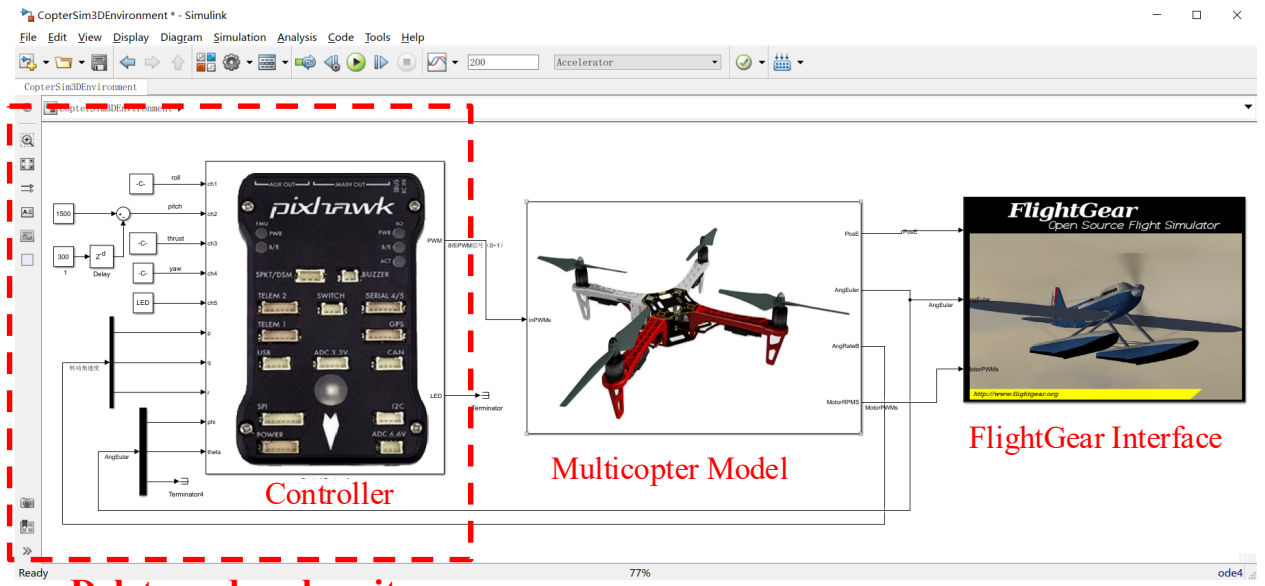


Procedure of Attitude Control Experiment

☐ Algorithm Design and SIL Simulation Phase

(3) Step 3: integrate the controller with the model

Open the Simulink file for SIL simulation, i.e., "e0\1.SoftwareSimExps\CopterSim3DEnvironment.slx" used in the previous chapter; delete its original controller subsystem (remember to create a backup); then, copy the new controller subsystem obtained in Step 2 to replace it.

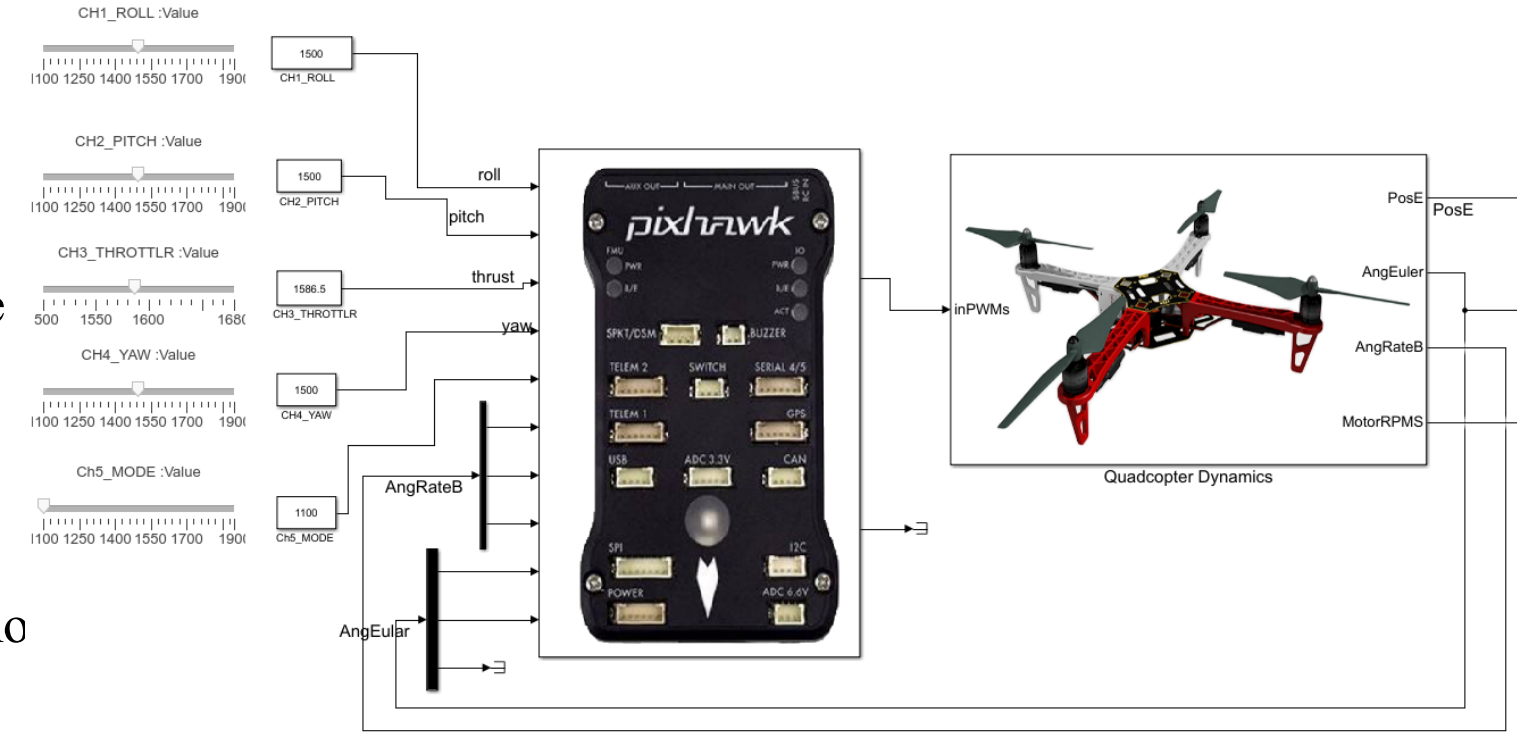


Procedure of Attitude Control Experiment

☐ Algorithm Design and SIL Simulation Phase

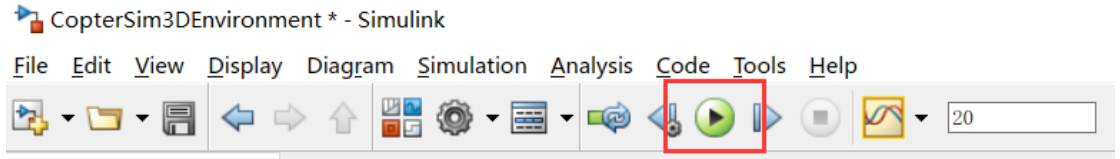
(4) Step 4: connect and configure the inputs and outputs

Given that the RC transmitter signals cannot be obtained in the SIL simulation phase, readers can use the constant values to replace them or use the MATLAB functions to simulate the corresponding RC transmitter actions. An example is also available in "e0\3.DesignExps\Exp2_ControlSystemDemo slx" whose controller and practical RC transmitter signals have been connected



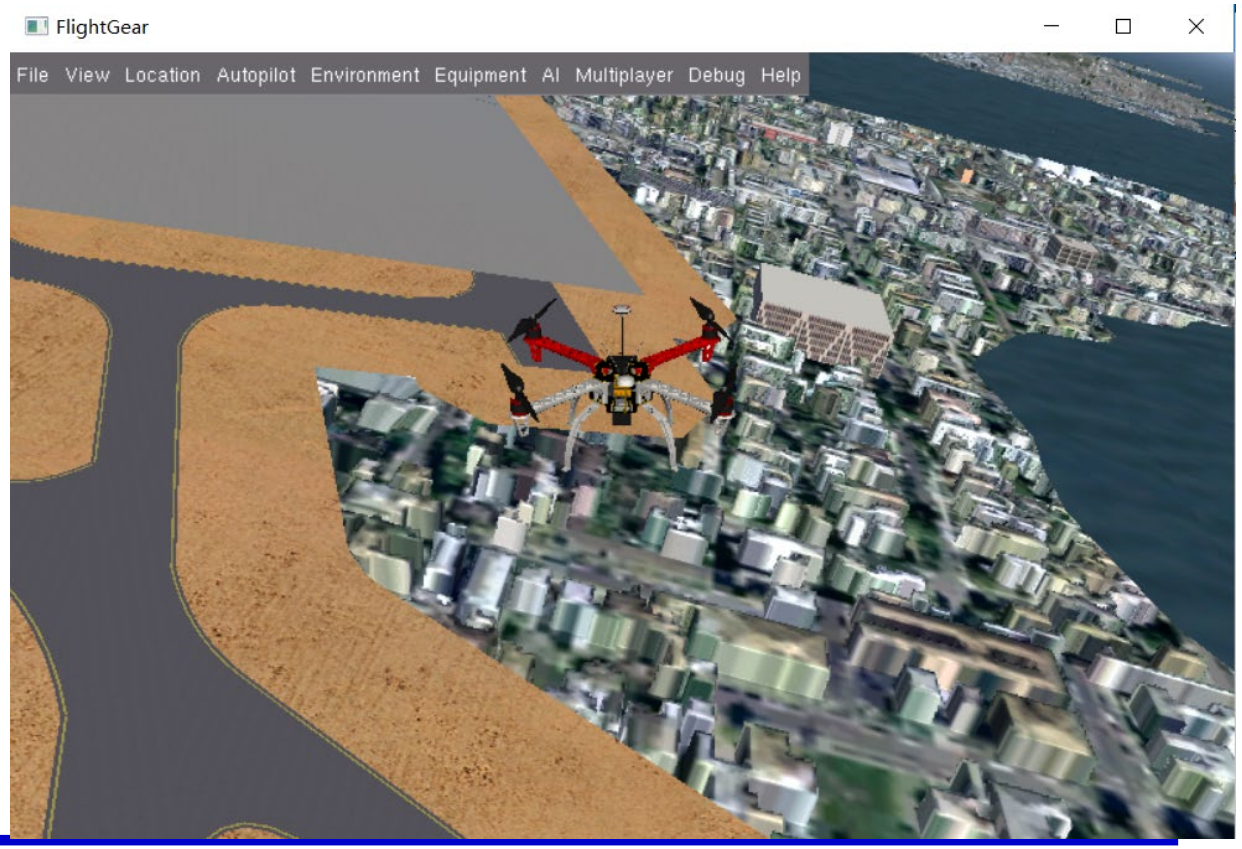
Procedure of Attitude Control Experiment

☐ Algorithm Design and SIL



(5) Step 5: start a joint simulation

Click the FlightGear-F450 shortcut on the Windows desktop to open FlightGear, and click on the “Start Simulation” button on the Simulink UI to start the simulation. Then, it can be observed in FlightGear that a quadcopter climbs up for some time and then you can move the siders in Simulink to simulate using RC to control a multicopter.

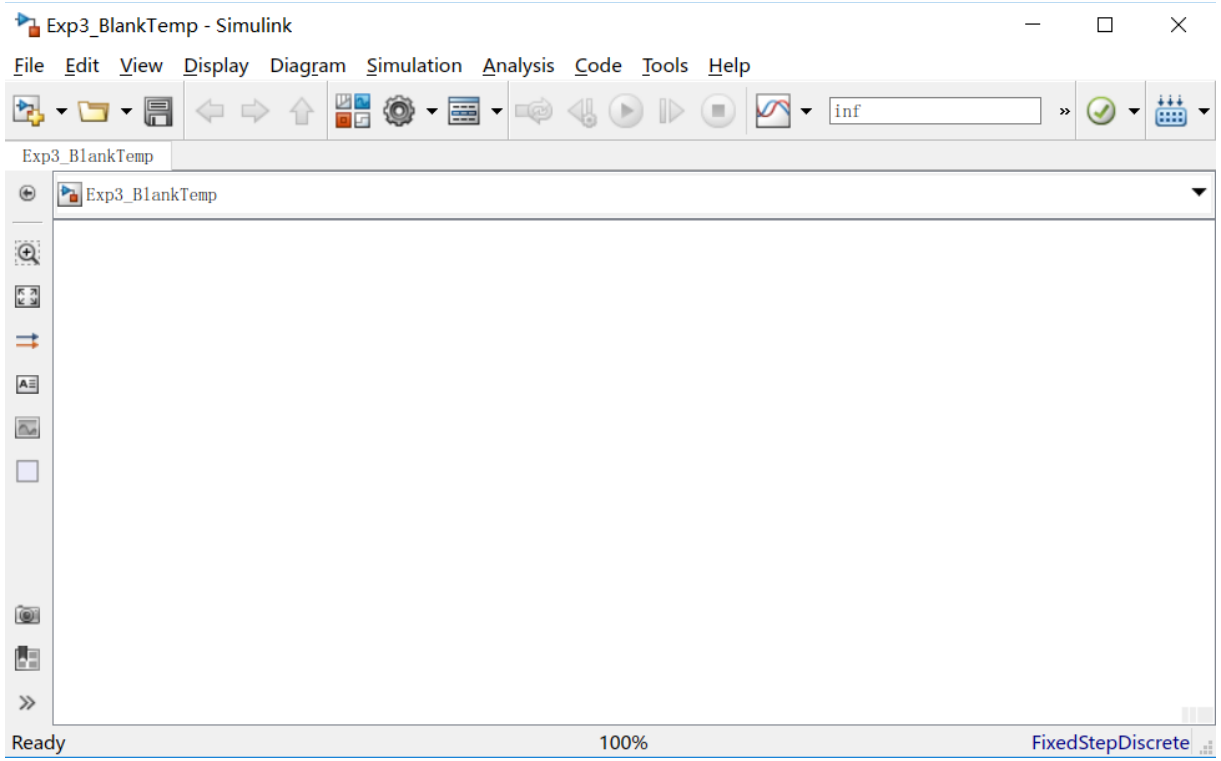


Procedure of Attitude Control Experiment

☐ Code Generation and Configuration Phase

(6) Step 6: configuration of the code generation environment

After finishing the SIL simulation in Simulink, copy the obtained controller subsystem to file "e0\3.DesignExps\Exp3_BlankTemp.slx" (this file has been configured with all the settings required for code generation). Readers can also create a blank Simulink file and configure it



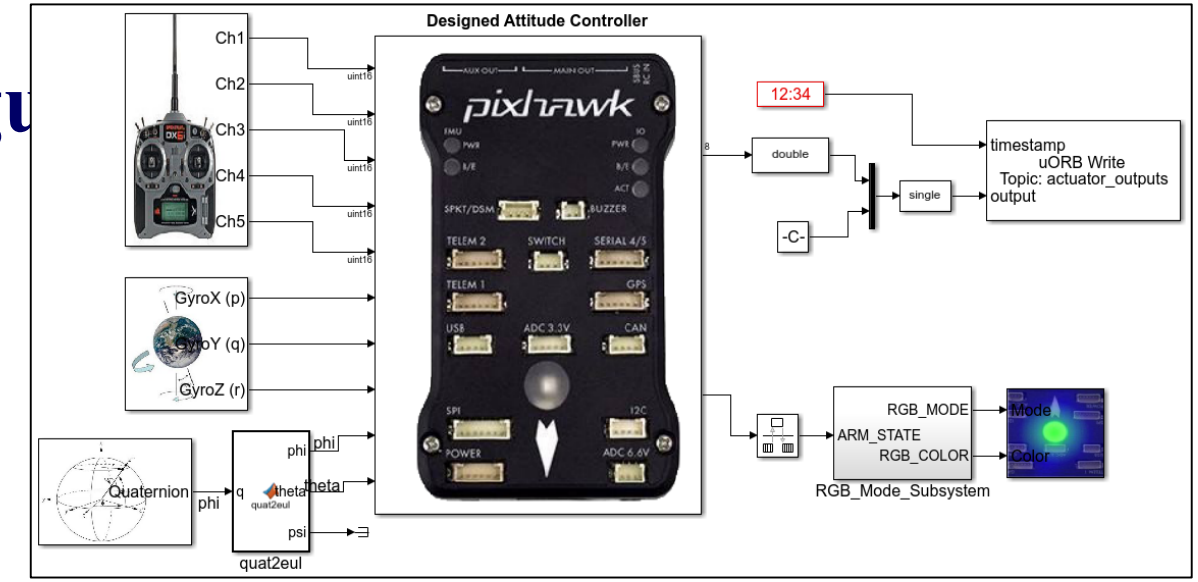
Procedure of Attitude Control Experiment

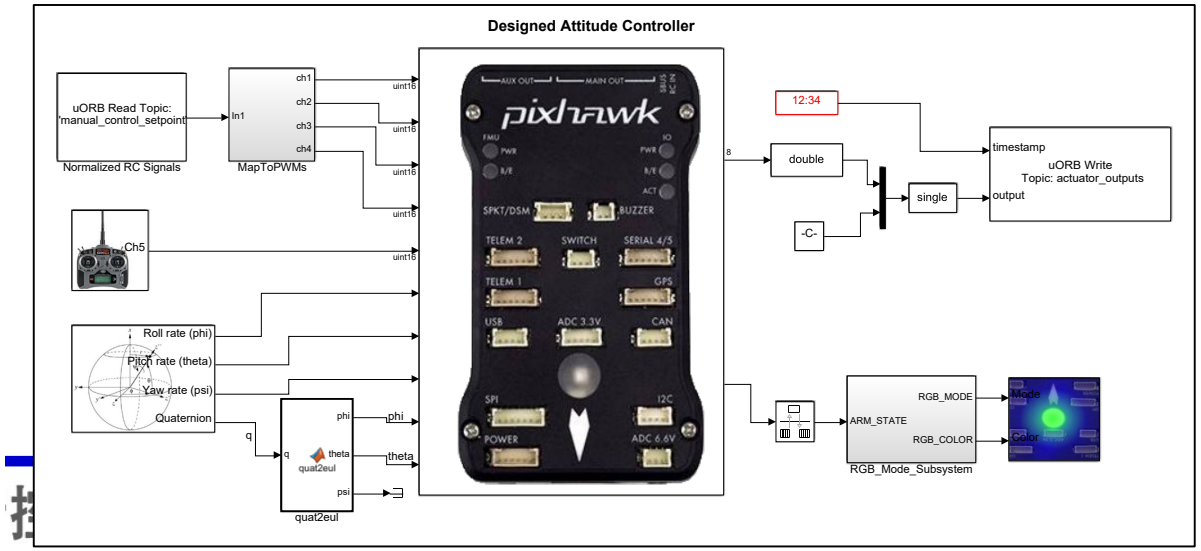
☐ Code Generation and Configu

(7) Step 7: connect the controller to the PSP modules

Extract the corresponding I/O interfaces from the Simulink PSP module library and connect it to the controller obtained in Step 6. A complete example is available in "e0\3.DesignExps\Exp4_AttitudeSystemCodeGen_old.slx". Note that the motor control signals should be sent the uORB message of "actuator_outputs" to the "uORB Write" module instead of the PWM output module. This is because the controller is currently used for HIL simulation instead of actual flight tests.

Besides, this demo reads RC raw data, and you can also choose the demo "Exp4_AttitudeSystemCodeGen.slx" uORB message "mannual_control_setpoint" to acquire calibrated and normalized RC signals, so you don't need to worry about calibration or incompatibility problems for different RC systems.





Procedure of Attitude Control Experiment

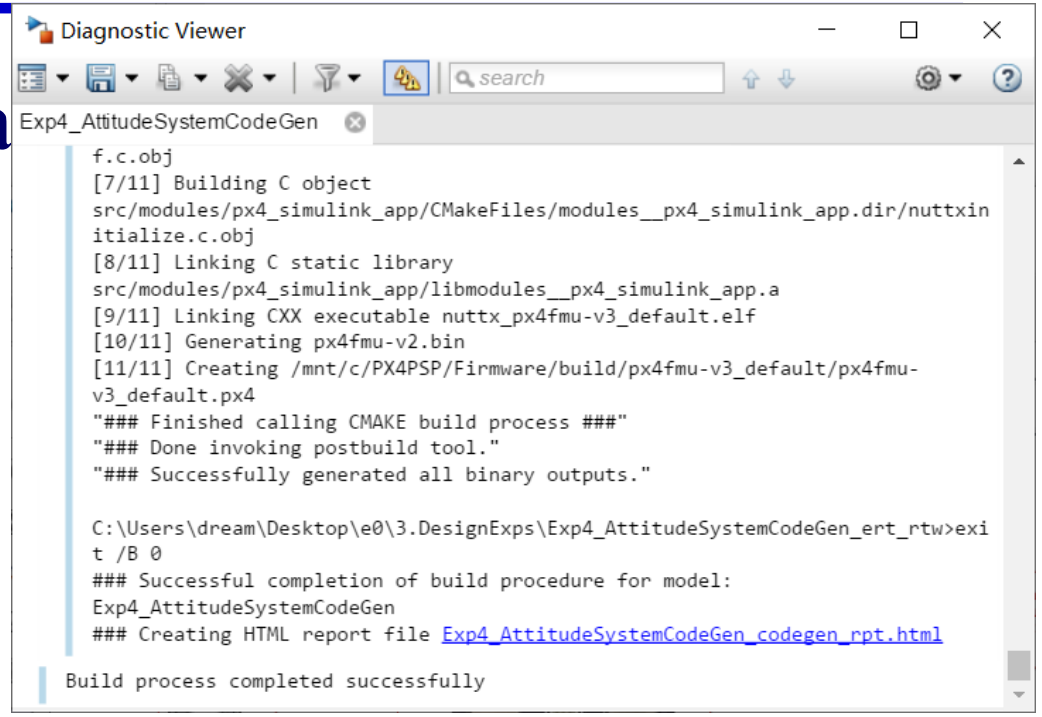
☐ Code Generation and Configura

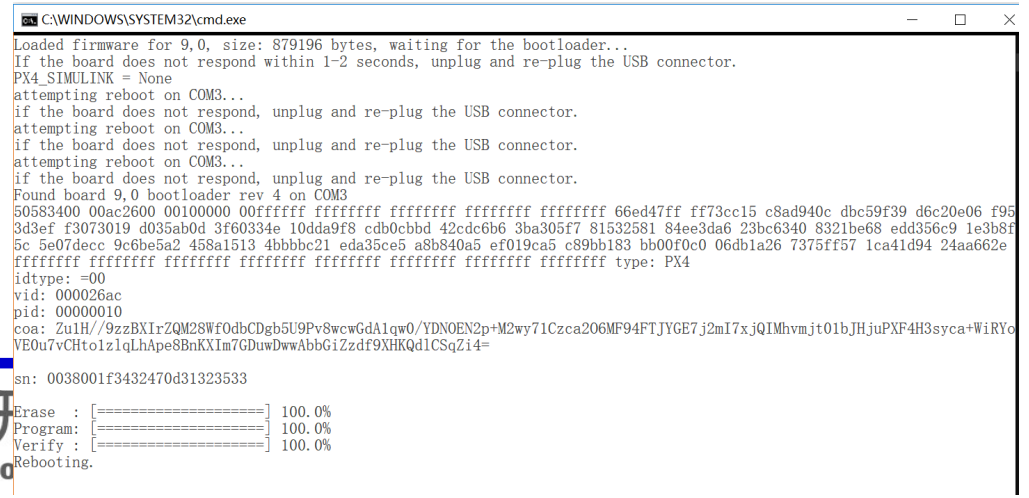
(8) Step 8: generate code and compile the firmware

Click the “Build” button on the Simulink toolbar to automatically generate code and PX4 firmware file. The result on the upper-right figure shows a successful compilation process.

(9) Step 9: upload the firmware

Connect the computer to the Pixhawk autopilot with a USB cable, then use the “PX4Upload” function to upload the firmware to the Pixhawk. A successful uploading result is shown in the lower-fight figure.



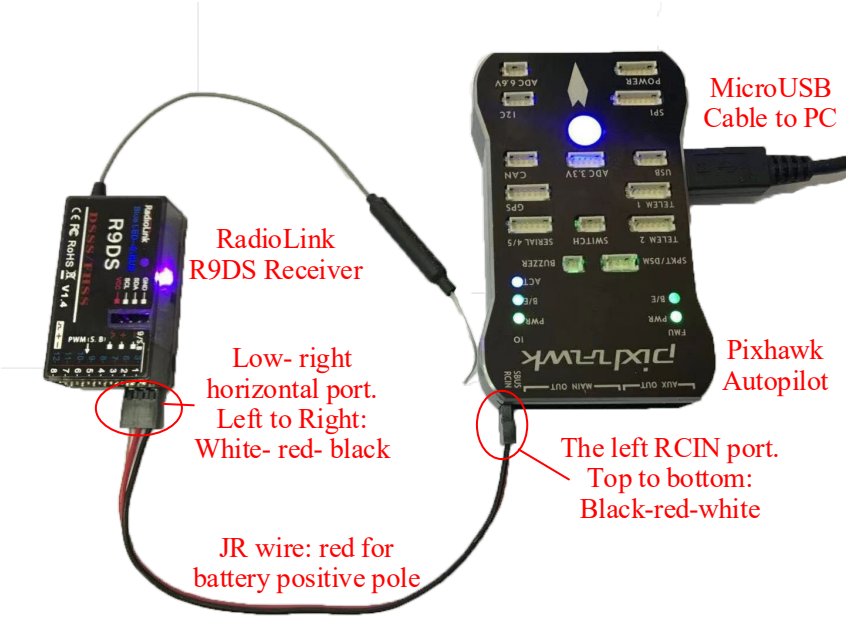


Procedure of Attitude Control Experiment

☐ HIL Simulation Phase

(10) Step 10: hardware system connection

As shown in the picture, connect the RC receiver to Pixhawk with a three-color JR cable, then connect Pixhawk to the computer via a USB cable. At this point, readers can observe that the LED on Pixhawk lights up and blinks slowly, the LED on the RC receiver is blue and white (this is for RadioLink and green light for Futaba receiver). Then, turn on the RC transmitter, readers can observe that the LED light on the Pixhawk blinks quickly for a few seconds, indicating that the RC transmitter data has been successfully received. If there is no change for the Pixhawk LED light, indicating that the connection between the RC transmitter and the receiver is not correct, readers should check and confirm the hardware connection.



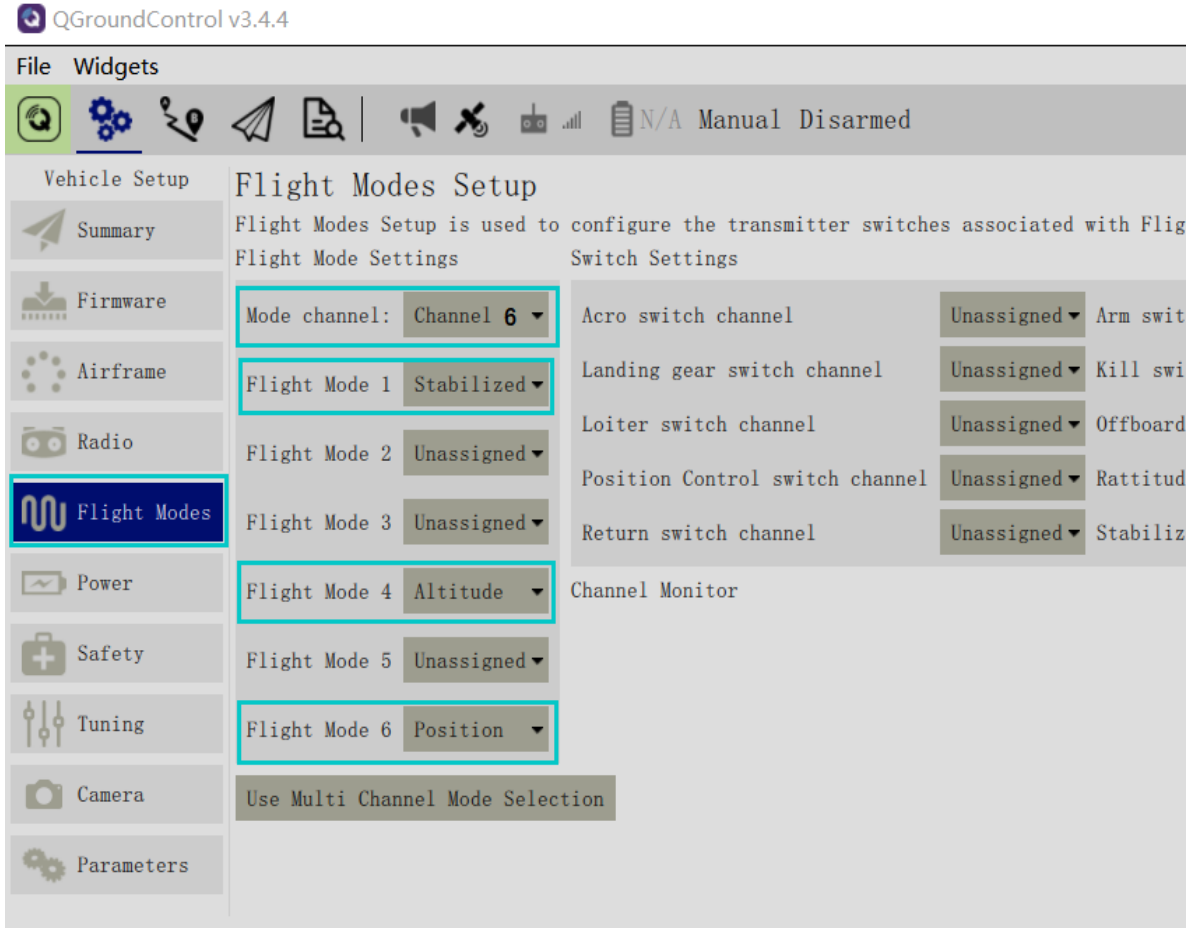
Procedure of Attitude Control Experiment

☐ HIL Simulation Phase

(11) Step 11: CopterSim configuration

1) Open the QGC software, and then connect the Pixhawk hardware with a USB cable.

1. Click “Airframe” on the QGC setting page to confirm that the “HIL Quadcopter X” airframe mode is selected by default.
2. Click “Flight Modes” to ensure the “Model channel” is not Channel 5, which can avoid the occupancy of CH5 channel by PX4’s own mode switch algorithm.
3. Close the QGC software.

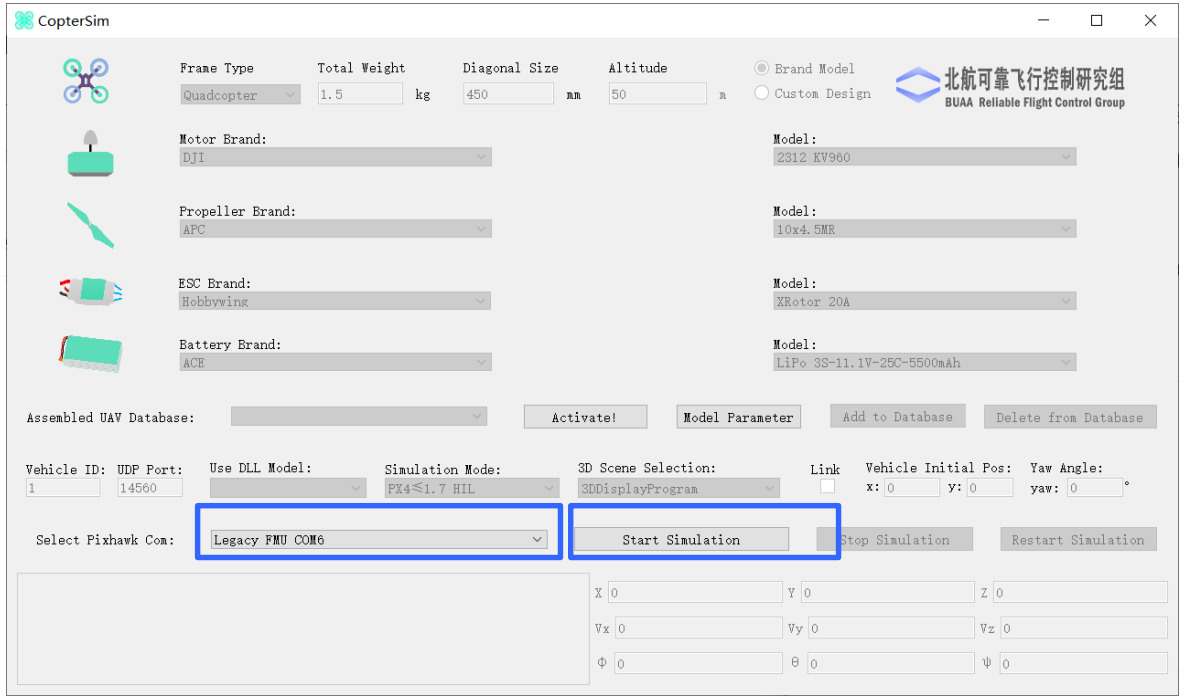


Procedure of Attitude Control Experiment

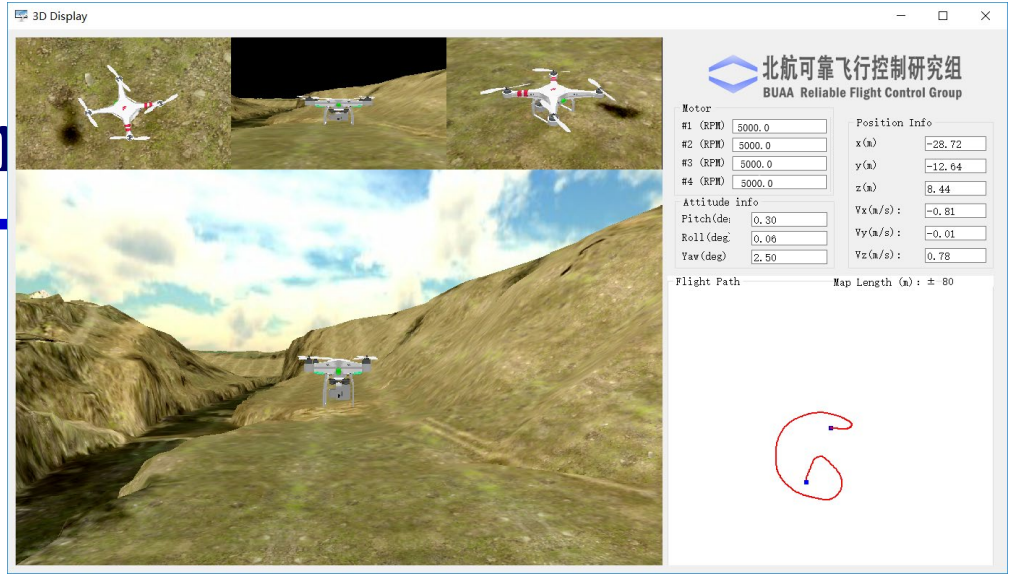
☐ HIL Simulation Phase

(11) Step 11: CopterSim configuration

2) Double-click the CopterSim shortcut on the Windows desktop to open it. There is no need to change the model parameters. Select the serial port of the Pixhawk autopilot in the “Select Pixhawk Com” drop-down box (usually in the format “*** FMU COM*”), click the “Start Simulation” button to start HIL simulation. As shown in the figure, the message returned by the Pixhawk autopilot will be printed on the lower-left box of the CopterSim UI.



Procedure of Attitude Con



☐ HIL Simulation Phase

(12) Step 12: 3DDisplay configuration

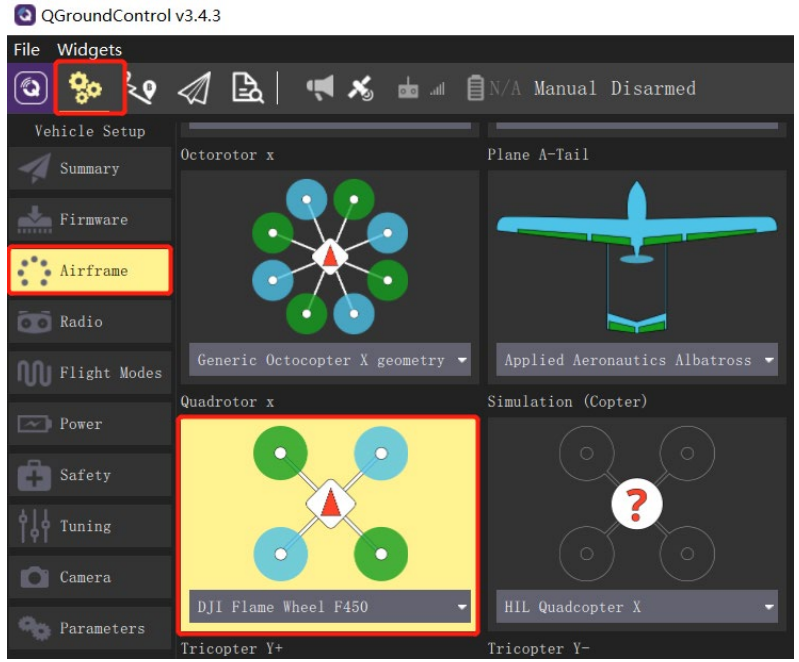
1. Double-click the 3DDisplay shortcut on the desktop to open it. This software does not require any configuration; it passively receives the flight attitude and trajectory information of multicopters sent by CopterSim and displays it in real-time.
2. Move the throttle stick on the RC transmitter to the lower-right corner for three seconds to disarm the Pixhawk, and pull down (or back) the CH5 stick to the rearming position to disarm the designed controller. Then, the RC transmitter is free to control the multicopter to complete the corresponding action. As shown in the figure, the multicopter attitude and position can be observed on the left side of the 3DDisplay interface. The real-time flight data are observed in the upper-right region, whereas the multicopter trajectory is observed in the lower-right region of the 3DDisplay UI.

Procedure of Attitude Control Experiment

☐ Flight Test Phase and Results

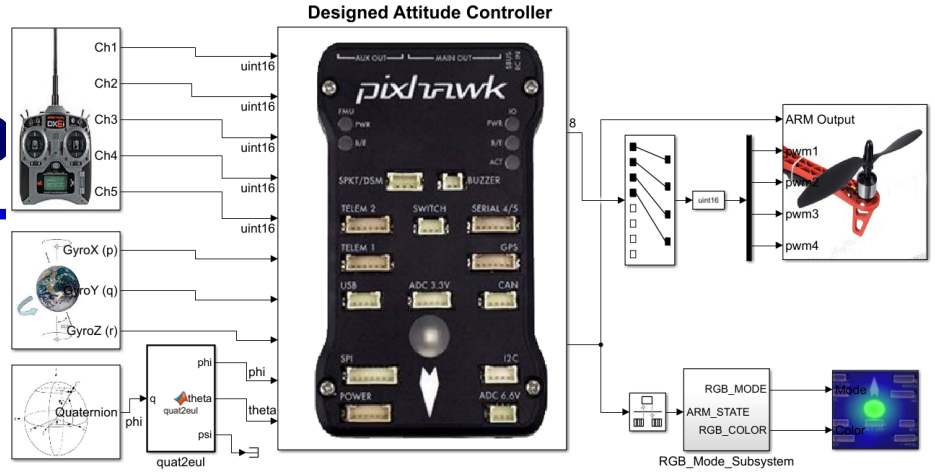
(13) Step 13: mount Pixhawk onto a multicopter airframe

The multicopter used in the outdoor flight tests is an F450 quadcopter.The parameters of the multicopter are accurately measured and identified by the system identification methods to ensure that the multicopter simulation model is consistent with the dynamics of the real multicopter system. For outdoor flight tests, the airframe of Pixhawk should be changed from “HIL Quadcopter X” to “DJI Flame Wheel F450” in QGC. All sensors should also be calibrated in QGC.



F450 airframe

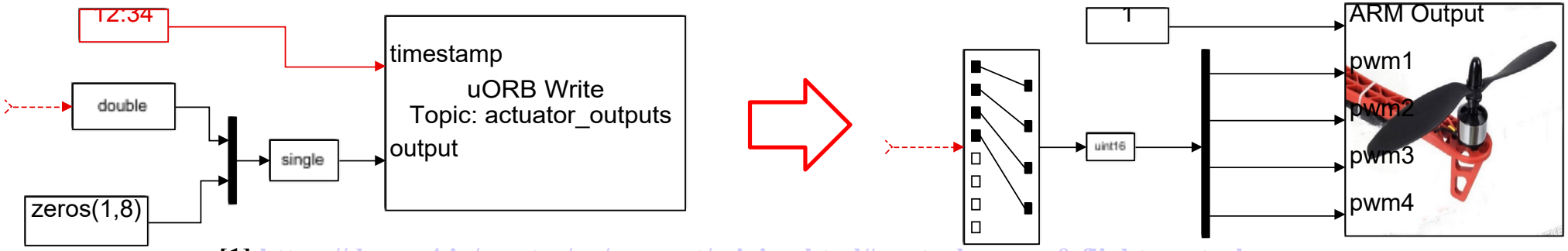
Procedure of Attitude Co



☐ Flight Test Phase and Results

(14) Step 14: modify the Simulink controller

Open the Simulink controller file and change the "uORB Write" module to the "PWM_out" module provided by the PSP toolbox. An example with the modified motor output is presented in "e0\3.DesignExps\Exp5_AttitudeSystemCodeGenRealFlight_old.slx" (or choose the demo without suffix "_old" to change the RC data source). Then, click the Simulink "compile" button to compile the controller into PX4 firmware and upload it to Pixhawk. Note that, this demo does not need to arm the drone with the RC throttle stick, but it need the switch the CH5 stick (sometime you need to switch it twice) to arm the drone. Note: for some autopilot hardware that do not support PWM_out module, you can use the demo with suffix "_all.slx" to send control signals (normalized thrust and torque by the uORB message actuator_controls_0 [1]) to the PX4 mixer to control the motor indrectly.



[1] https://dev.px4.io/master/en/concept/mixing.html#control-group-0-flight-control

Procedure of Attitude Control Experiment

☐ Flight Test Phase and Results

(15) Step 15: preparation for flight tests

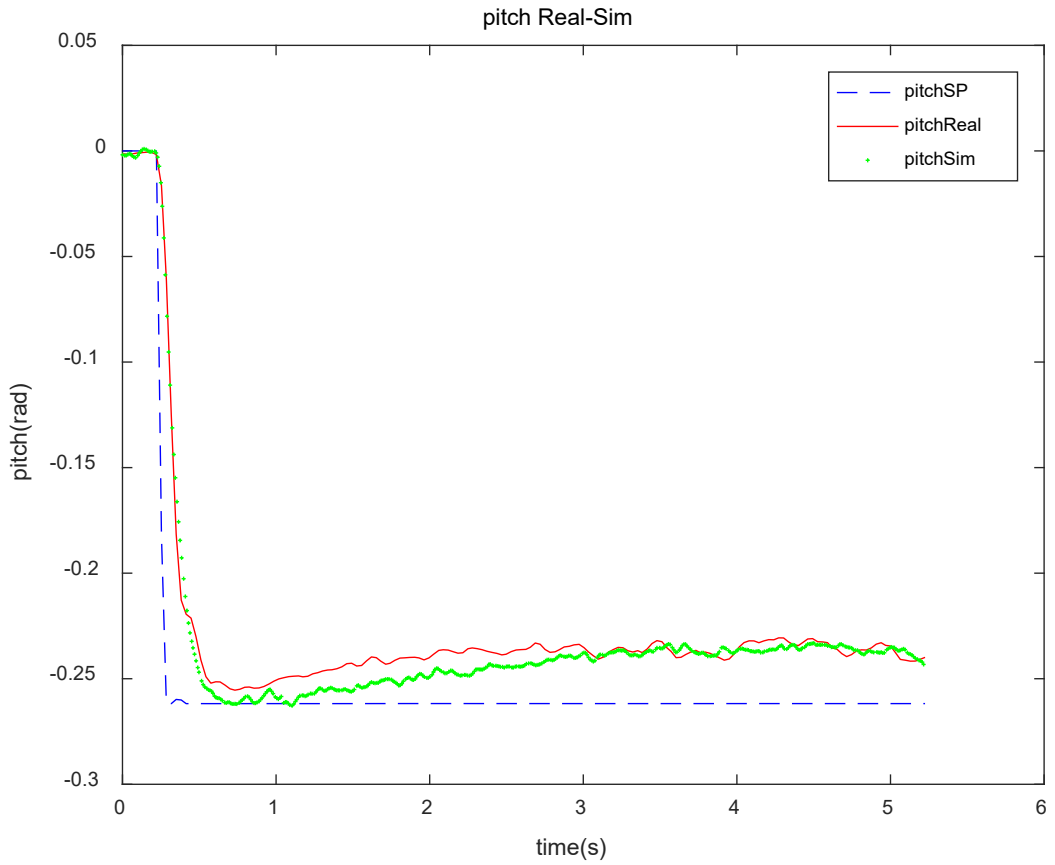
Owing to the lack of complete failsafe logic for the generated control algorithm, safety problems should be fully considered to prevent the uncertainty in outdoor flight tests. For example, the tests should be carried out in a relatively open area (such as a lawn) and on a nice day with good weather and low wind speed. When the above conditions are met, connect Pixhawk with the battery, and press the safety switch on Pixhawk for more than three seconds. Then, control the multicopter with an RC transmitter to verify the actual performance of the controller.

Procedure of Attitude Control Experiment

☐ Flight Test Phase and Results

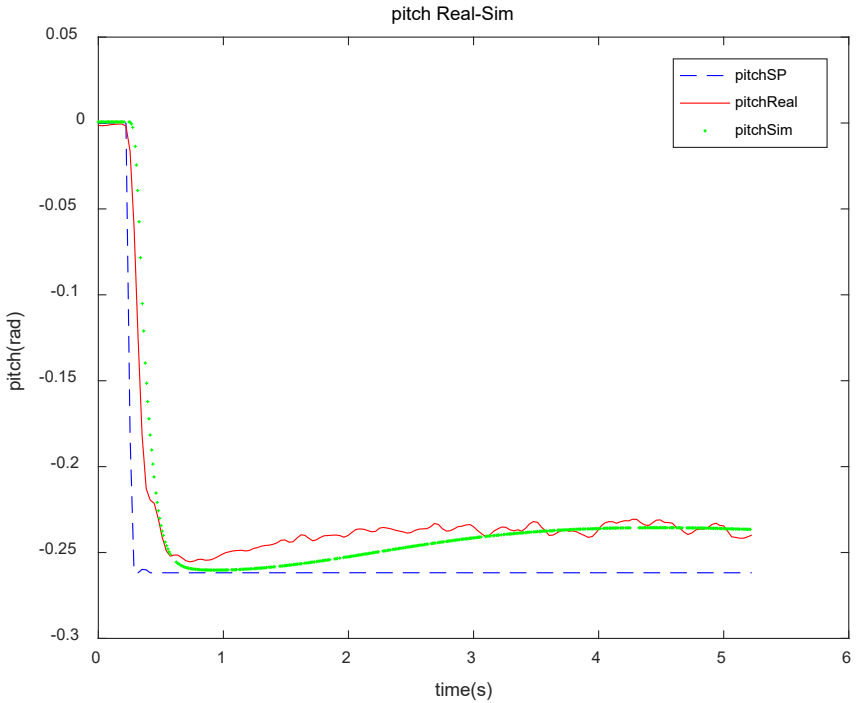
(16) Step 16: test results and analysis

The figure on the right presents the HIL simulation results when the sensor noise level in CopterSim is set to 1.0. It can be observed that the step response curves from the HIL simulation and the outdoor flight are close to each other in terms of dynamic processes and noise levels.



Procedure of Attitude Con

☐ Flight Test Phase and Results



The figure is the HIL simulation result when the noise level in CopterSim is set to 0. It can be seen that the response curve of the HIL simulation is very smooth with no noise disturbance, and the dynamic process and outdoor flight results are slightly different. Note that because the airframe “HIL Quadcopter X” is not exactly the same as the airframe “DJI Flame Wheel F450” used in outdoor flight, there are differences in the controller parameters. In addition, the data transmission speed of the Pixhawk serial port may fluctuate during the HIL simulation, thereby affecting the real-time performance. Still, the error between their response curves is accepted. Simultaneously, the aerodynamics of the multicopter in outdoor flight is very complicated, and a simplified aerodynamic model is used in the multicopter simulation model. Therefore, it is acceptable to have a certain error in the final angular steady-state response curve.

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!