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

Lesson 04 Experimental Process

Dr. Xunhua Dai, dai@buaa.edu.cn

School of Automation Science and Electrical Engineering,

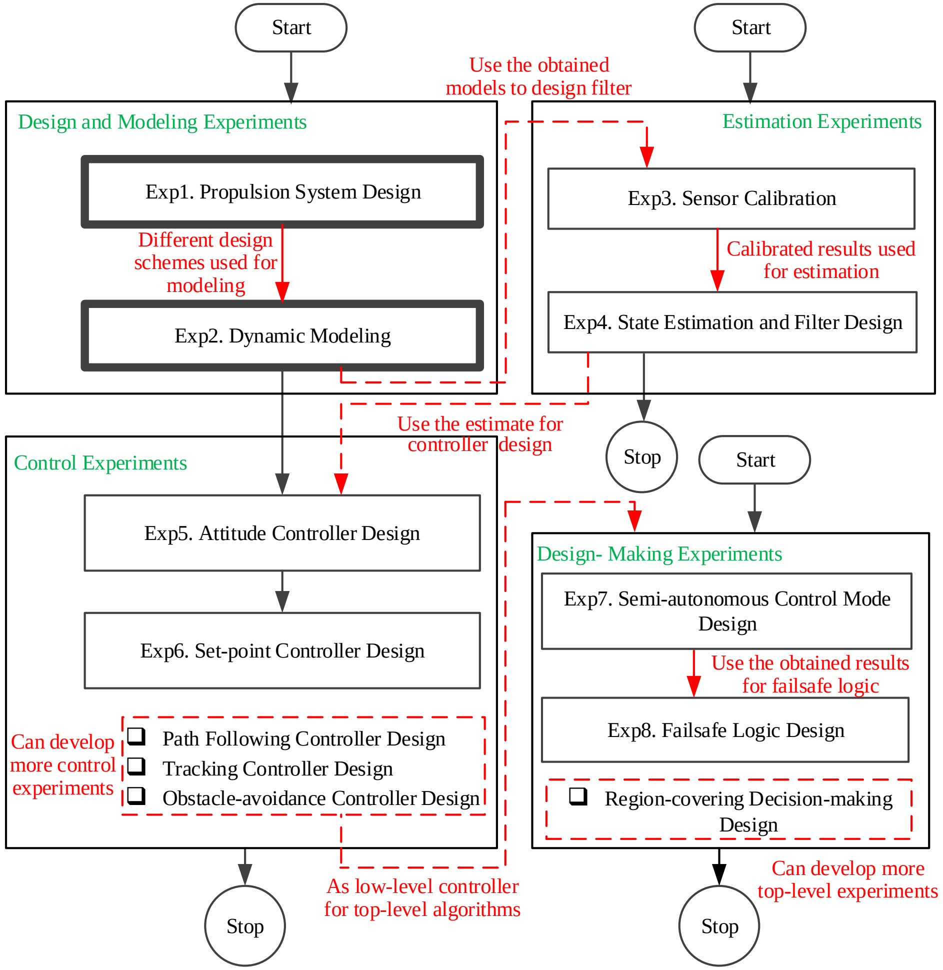
Beihang University, Beijing 100191, China.

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.



Start
Start
Use the obtained models to design filter
Design and Modeling Experiments
Estimation Experiments
Exp1. Propulsion System Design
Different design schemes used for modeling
Exp2. Dynamic Modeling
Exp3. Sensor Calibration
Calibrated results used for estimation
Exp4. State Estimation and Filter Design
Use the estimate for controller design
Stop
Start
Control Experiments
Exp5. Attitude Controller Design
Exp6. Set-point Controller Design
Can develop more control experiments
☐ Path Following Controller Design
☐ Tracking Controller Design
☐ Obstacle-avoidance Controller Design
Design- Making Experiments
Exp7. Semi-autonomous Control Mode Design
Use the obtained results for failsafe logic
Exp8. Failsafe Logic Design
☐ Region-covering Decision-making Design
Stop
As low-level controller for top-level algorithms
Stop
Can develop more top-level experiments

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)

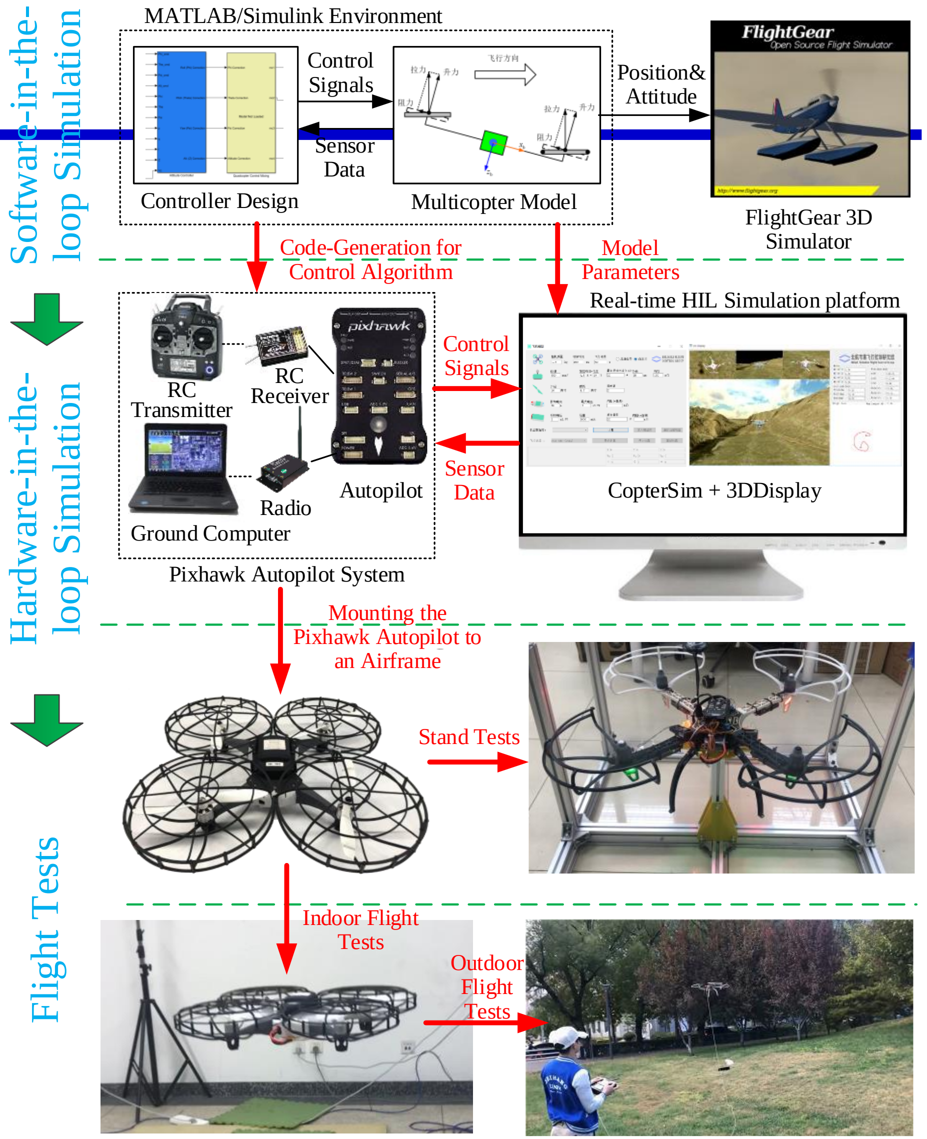

Software-in-the-loop Simulation
MATLAB/Simulink Environment
Control Signals
Sensor Data
Controller Design
Multicopter Model
Position& Attitude
FlightGear Open Source Flight Simulator
FlightGear 3D Simulator
Code-Generation for Control Algorithm
Model Parameters
Hardware-in-the-loop Simulation
RC Transmitter
RC Receiver
pixhawk
Autopilot
Radio
Ground Computer
Pixhawk Autopilot System
Control Signals
Sensor Data
Real-time HIL Simulation platform
CopterSim + 3DDisplay
Mounting the Pixhawk Autopilot to an Airframe
Stand Tests
Flight Tests
Indoor Flight Tests
Outdoor Flight Tests

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.

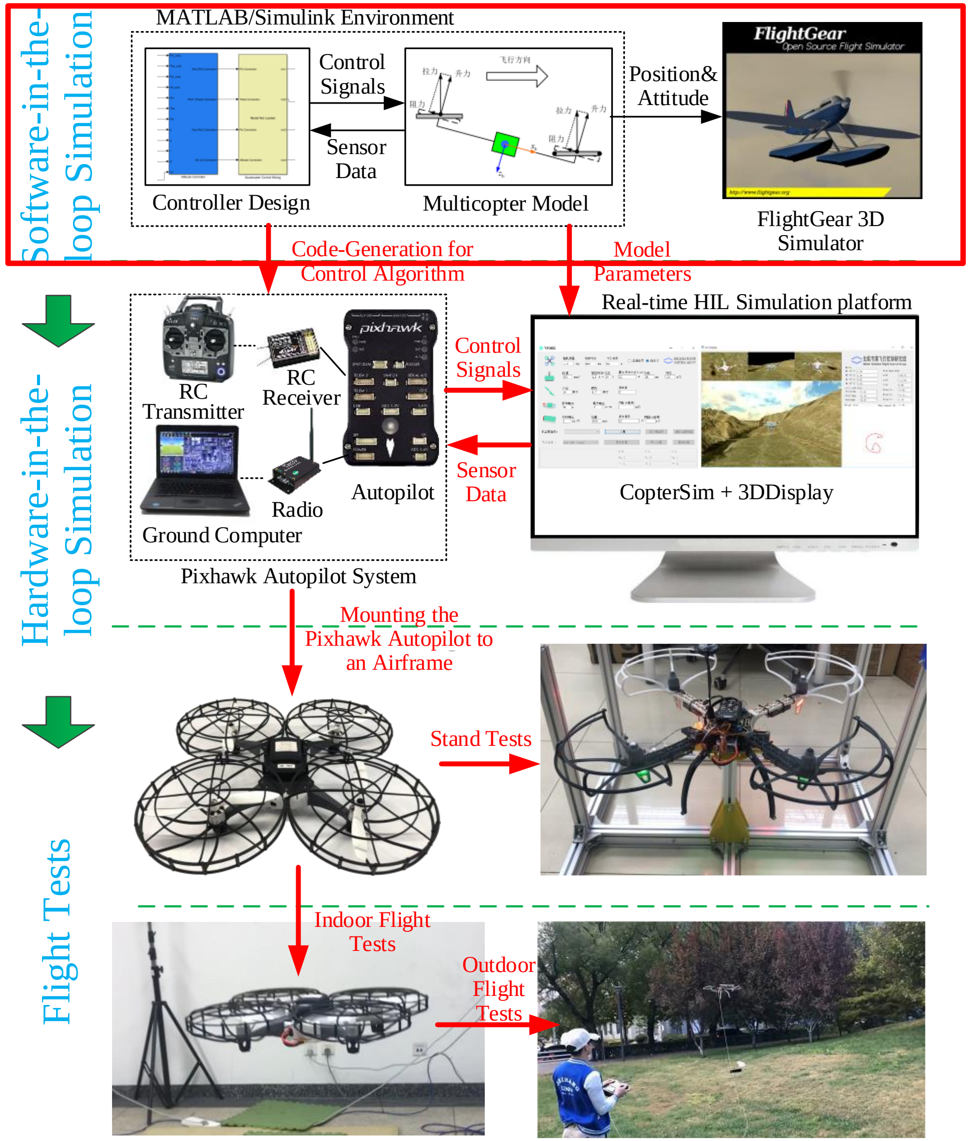

Software-in-the-loop Simulation
MATLAB/Simulink Environment
Control Signals
Sensor Data
Controller Design
Multicopter Model
Position& Attitude
FlightGear 3D Simulator
Code-Generation for Control Algorithm
Model Parameters
Real-time HIL Simulation platform
Hardware-in-the-loop Simulation
RC Transmitter
RC Receiver
Autopilot
Radio
Ground Computer
Pixhawk Autopilot System
Control Signals
Sensor Data
CopterSim + 3DDisplay
Mounting the Pixhawk Autopilot to an Airframe
Stand Tests
Flight Tests
Indoor Flight Tests
Outdoor Flight Tests

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.

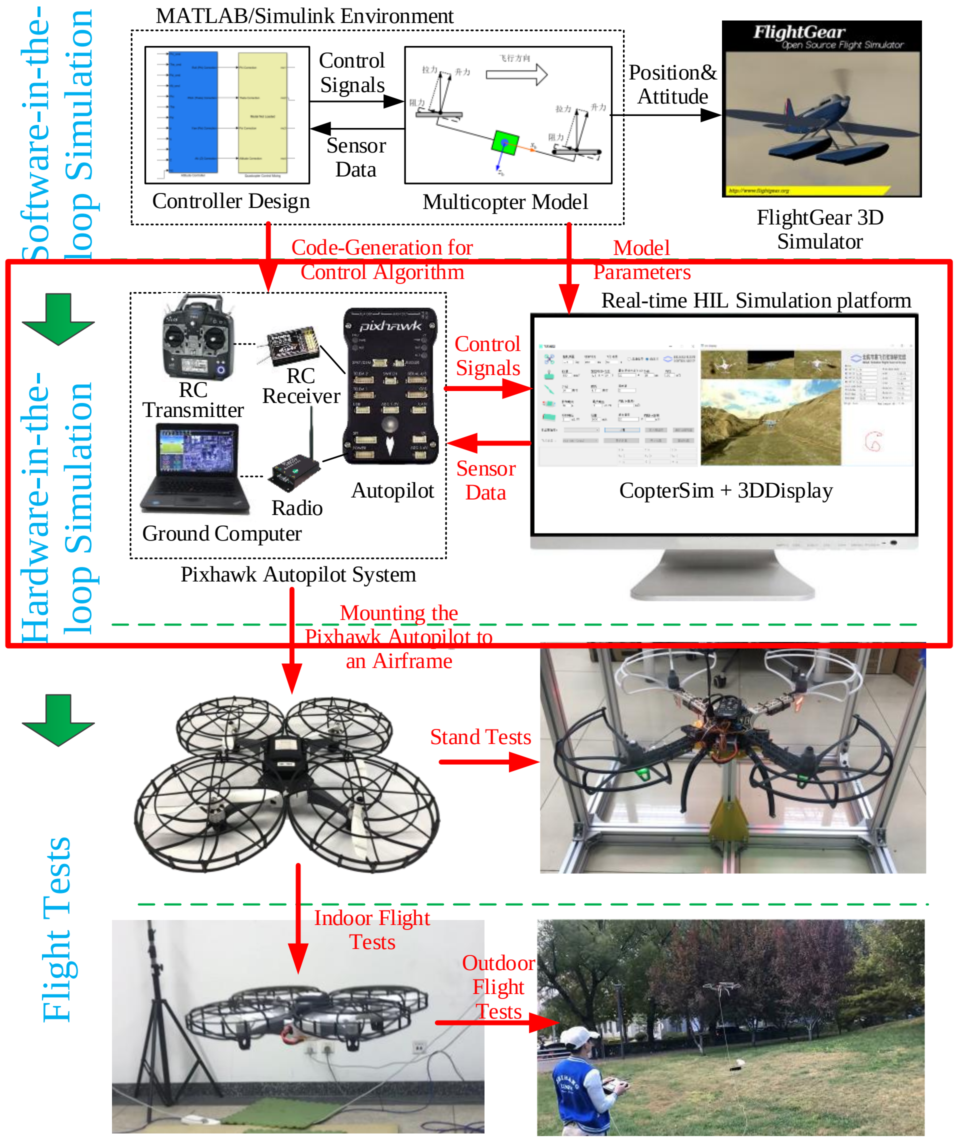

Software-in-the-loop Simulation
MATLAB/Simulink Environment
Control Signals
Sensor Data
Controller Design
Multicopter Model
Position& Attitude
FlightGear
Open Source Flight Simulator
FlightGear 3D Simulator
Code-Generation for Control Algorithm
Model Parameters
Hardware-in-the-loop Simulation
Real-time HIL Simulation platform
pixhawk
RC Transmitter
RC Receiver
Radio
Autopilot
Ground Computer
Pixhawk Autopilot System
Control Signals
Sensor Data
CopterSim + 3DDisplay
Mounting the Pixhawk Autopilot to an Airframe
Stand Tests
Flight Tests
Indoor Flight Tests
Outdoor Flight Tests

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.

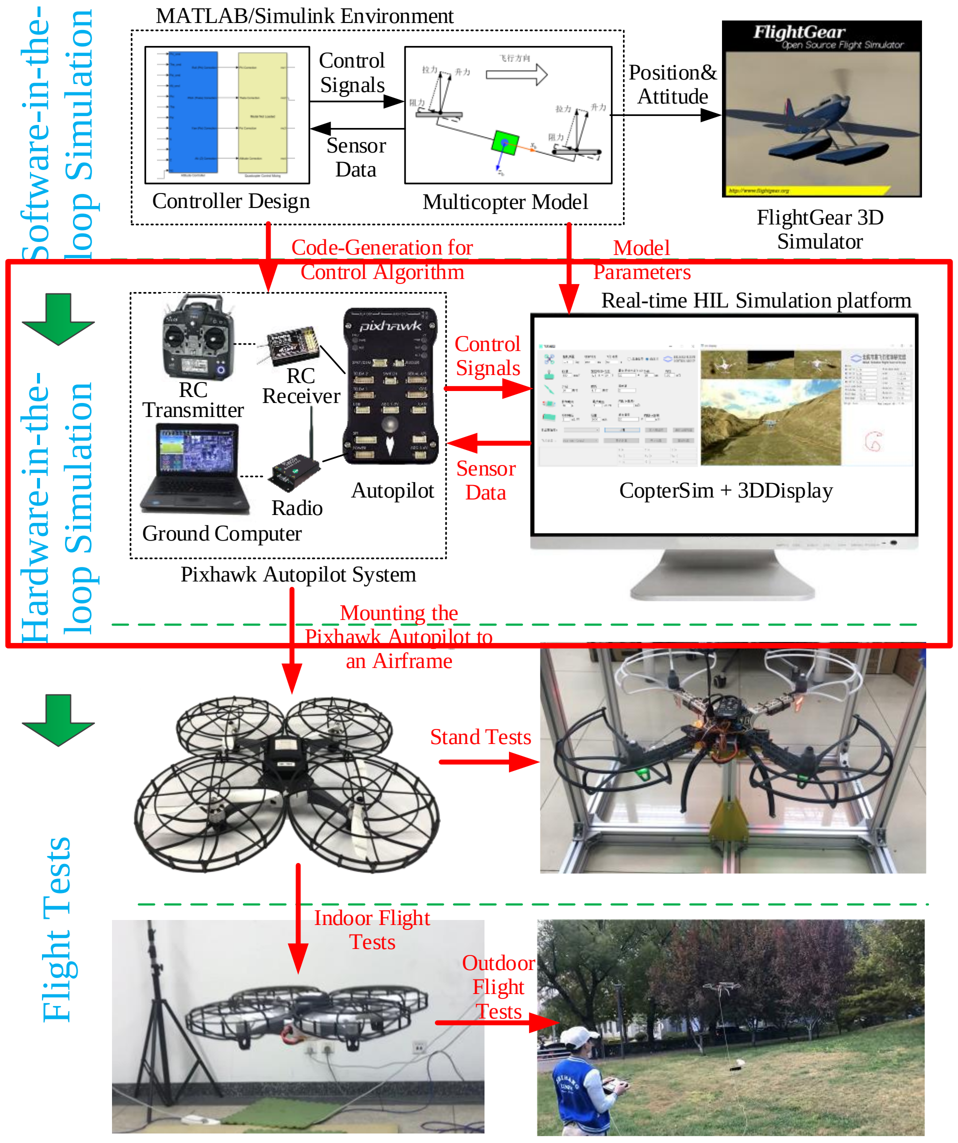
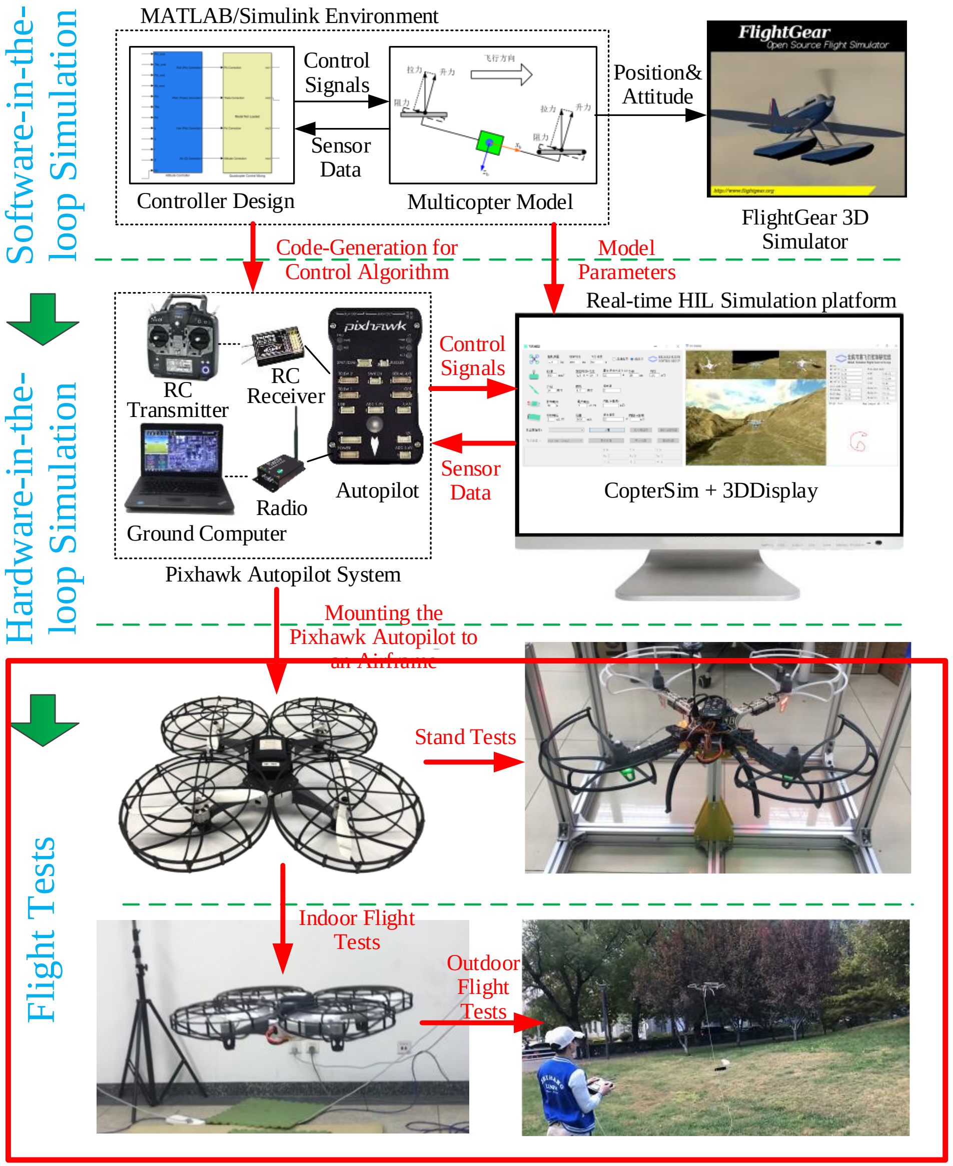

Software-in-the-loop Simulation
MATLAB/Simulink Environment
Control Signals
Sensor Data
Controller Design
Multicopter Model
Position& Attitude
FlightGear
Open Source Flight Simulator
FlightGear 3D Simulator
Code-Generation for Control Algorithm
Model Parameters
Hardware-in-the-loop Simulation
Real-time HIL Simulation platform
RC Transmitter
RC Receiver
pixhawk
Autopilot
Radio
Ground Computer
Pixhawk Autopilot System
Control Signals
Sensor Data
CopterSim + 3DDisplay
Mounting the Pixhawk Autopilot to an Airframe
Flight Tests
Stand Tests
Indoor Flight Tests
Outdoor Flight Tests

Fig. Overall experimental process

Experimental Process

(3) Flight Test Phase

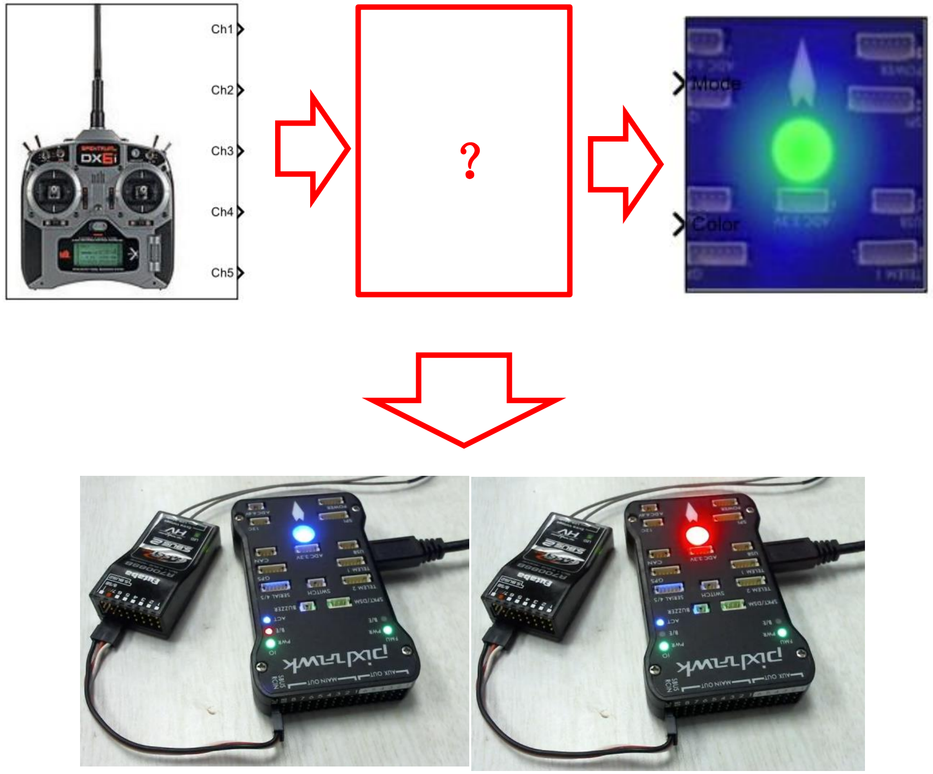
In this phase, the multicopter simulation model in CopterSim is further replaced with a real multicopter hardware system; the sensor data are replaced with signals from real sensor chips by sensing the multicopter motion and states and the control signals are directly output to the Pixhawk I/O ports 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.

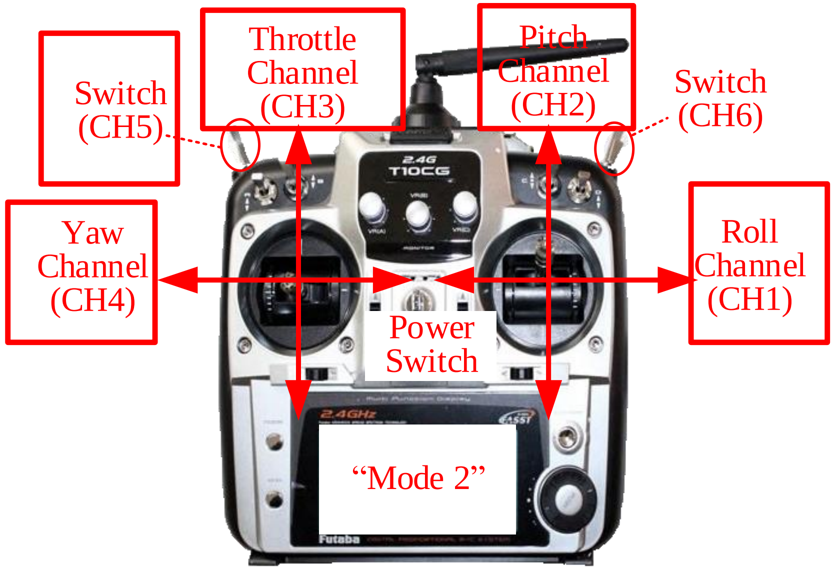

Software-in-the-loop Simulation
MATLAB/Simulink Environment
Control Signals
Sensor Data
Controller Design
Multicopter Model
Position& Attitude
FlightGear
Open Source Flight Simulator
FlightGear 3D Simulator
Code-Generation for Control Algorithm
Model Parameters
Hardware-in-the-loop Simulation
Real-time HIL Simulation platform
RC Transmitter
RC Receiver
pixhawk
Ground Computer
Radio
Autopilot
Pixhawk Autopilot System
Control Signals
Sensor Data
CopterSim + 3DDisplay
Mounting the Pixhawk Autopilot to an Airframe
Flight Tests
Stand Tests
Indoor Flight Tests
Outdoor Flight Tests

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.


Ch1
Ch2
Ch3
Ch4
Ch5
?
Mode
Color


Switch (CH5)
Throttle Channel (CH3)
Pitch Channel (CH2)
Switch (CH6)
Yaw Channel (CH4)
Roll Channel (CH1)
Power Switch
"Mode 2"

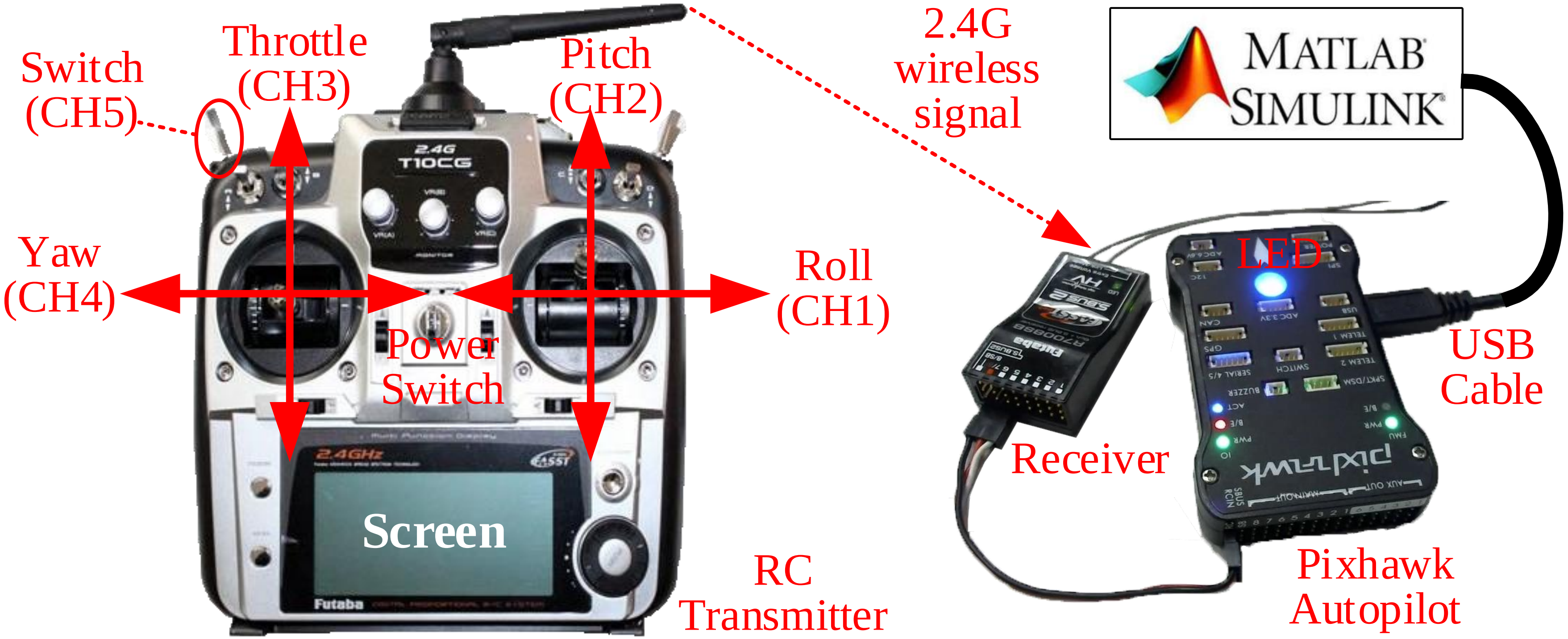
Throttle : control up-down movement
Pitch : control forward-backward
Yaw : control vehicle head direction
Roll : control left-right movement

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

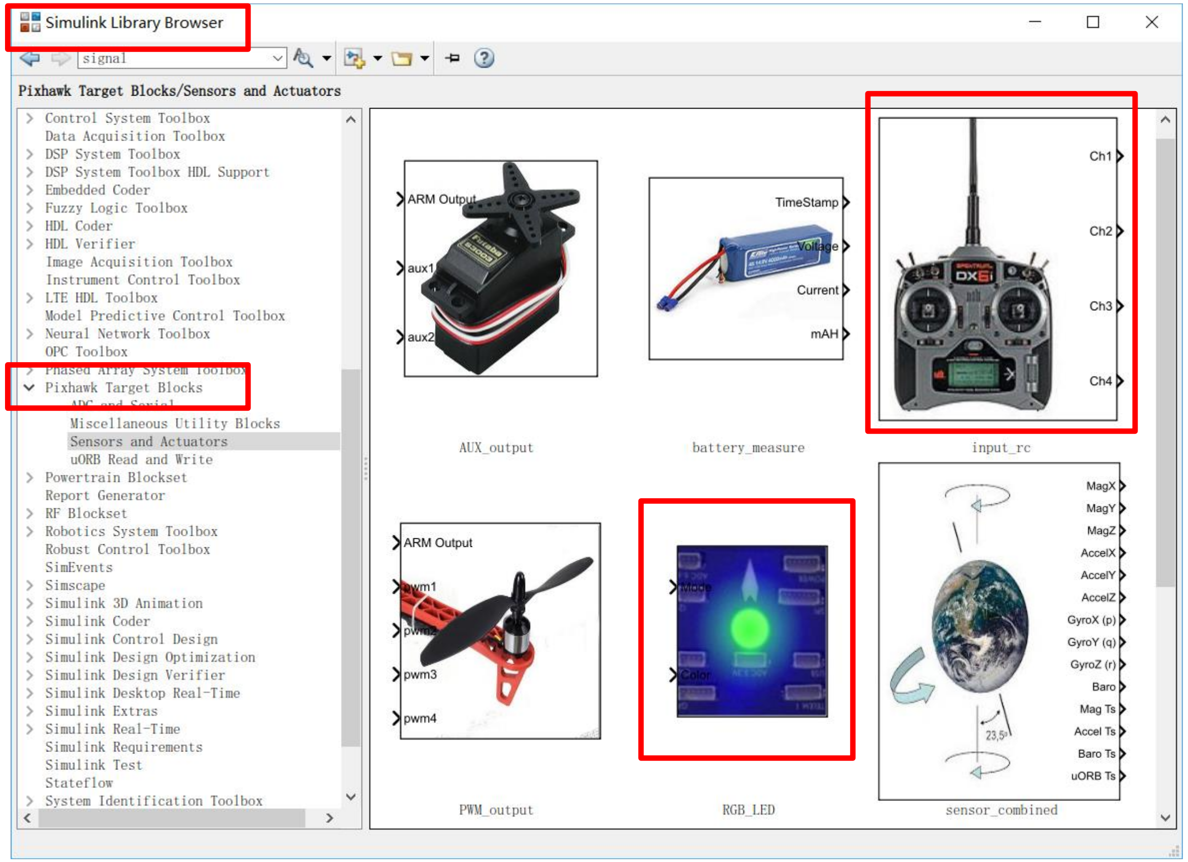


Switch (CH5)
Throttle (CH3)
Pitch (CH2)
2.4G wireless signal
MATLAB SIMULINK
Yaw (CH4)
Roll (CH1)
LED
Power Switch
USB Cable
Receiver
Screen
RC Transmitter
Pixhawk Autopilot

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”.

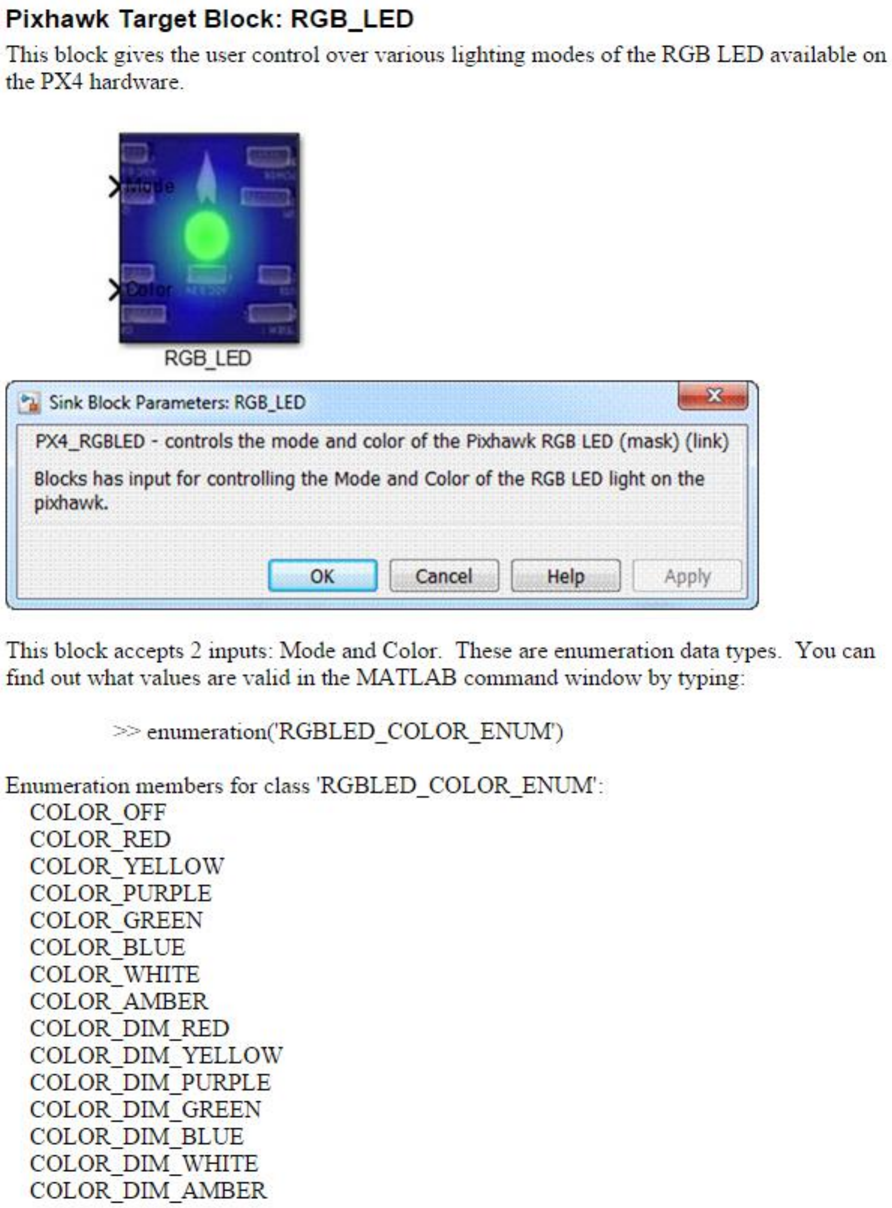

Simulink Library Browser
signal
Pixhawk Target Blocks/Sensors and Actuators
Control System Toolbox
Data Acquisition Toolbox
DSP System Toolbox
DSP System Toolbox HDL Support
Embedded Coder
Fuzzy Logic Toolbox
HDL Coder
HDL Verifier
Image Acquisition Toolbox
Instrument Control Toolbox
LTE HDL Toolbox
Model Predictive Control Toolbox
Neural Network Toolbox
OPC Toolbox
Phased Array System Toolbox
Pixhawk Target Blocks
ADC and Serial
Miscellaneous Utility Blocks
Sensors and Actuators
uORB Read and Write
Powertrain Blockset
Report Generator
RF Blockset
Robotics System Toolbox
Robust Control Toolbox
SimEvents
Simscape
Simulink 3D Animation
Simulink Coder
Simulink Control Design
Simulink Design Optimization
Simulink Design Verifier
Simulink Desktop Real-Time
Simulink Extras
Simulink Real-Time
Simulink Requirements
Simulink Test
Stateflow
System Identification Toolbox
ARM Output
aux1
aux2
AUX_output
TimeStamp
Voltage
Current
mAH
battery_measure
Ch1
Ch2
Ch3
Ch4
input_rc
ARM Output
pwm1
pwm2
pwm3
pwm4
PWM_output
Mode
Color
RGB_LED
MagX
MagY
MagZ
AccelX
AccelY
AccelZ
GyroX (p)
GyroY (q)
GyroZ (r)
Baro
Mag Ts
Accel Ts
Baro Ts
uORB Ts
sensor_combined

Procedure for LED Contro

□ 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.

PX4_RGBLED - controls the mode and color of the Pixhawk RGB LED (mask) (link)

Blocks has input for controlling the Mode and Color of the RGB LED light on the pixhawk.

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:

```
>> enumeration('RGBLED_COLOR_ENUM')
```

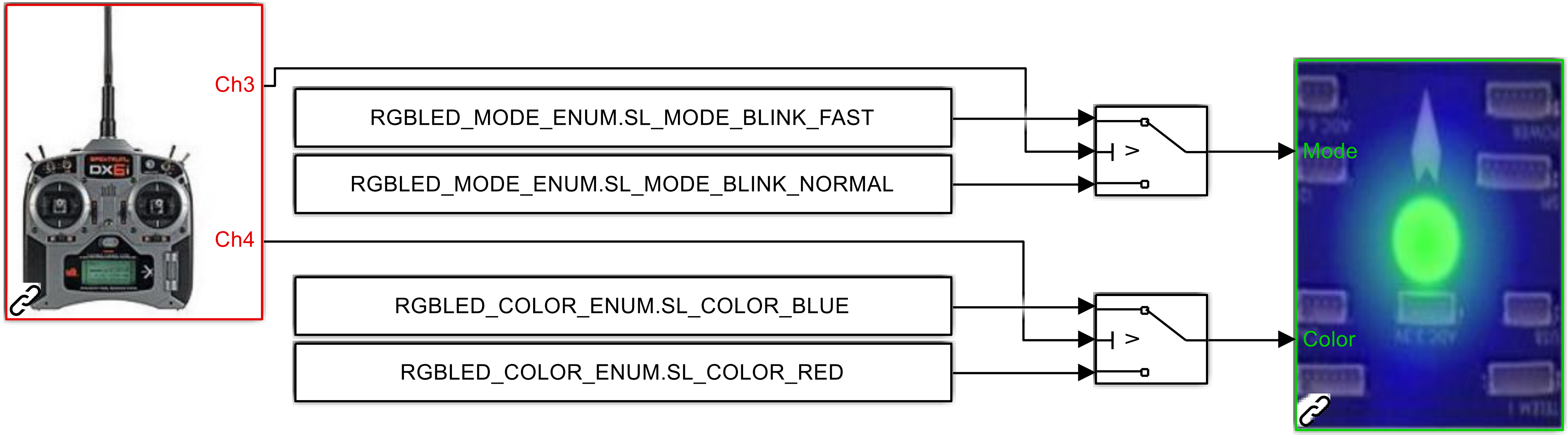
```
Enumeration members for class 'RGBLED_COLOR_ENUM':
  COLOR_OFF
  COLOR_RED
  COLOR_YELLOW
  COLOR_PURPLE
  COLOR_GREEN
  COLOR_BLUE
  COLOR_WHITE
  COLOR_AMBER
  COLOR_DIM_RED
  COLOR_DIM_YELLOW
  COLOR_DIM_PURPLE
  COLOR_DIM_GREEN
  COLOR_DIM_BLUE
  COLOR_DIM_WHITE
  COLOR_DIM_AMBER
```

图LED light module and its parameter setting box

Procedure for LED Control Experiment

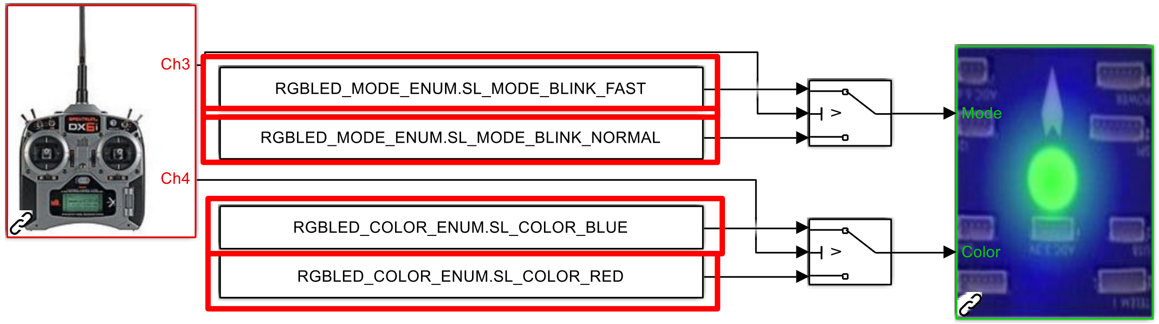
□ Design Control Model

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.


Ch3
Ch4
RGBLED_MODE_ENUM.SL_MODE_BLINK_FAST
RGBLED_MODE_ENUM.SL_MODE_BLINK_NORMAL
RGBLED_COLOR_ENUM.SL_COLOR_BLUE
RGBLED_COLOR_ENUM.SL_COLOR_RED
>
>
Mode
Color

Procedure for LED Control Experiment

☐ Design Control Model

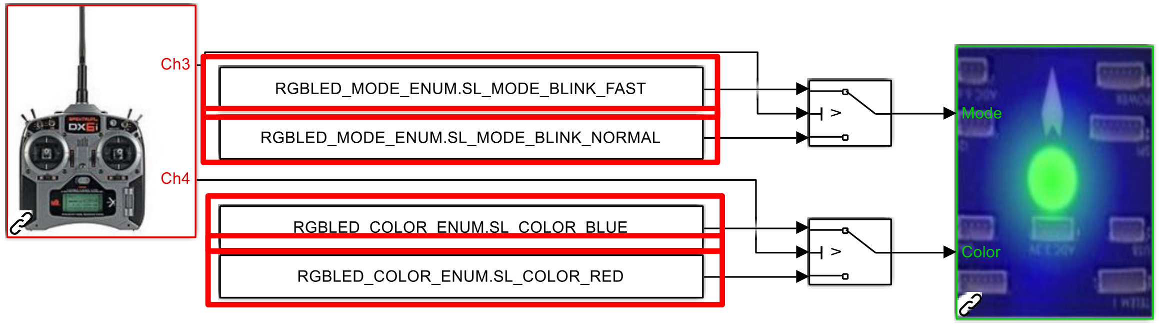

Ch3
Ch4
RGBLED_MODE_ENUM.SL_MODE_BLINK_FAST
RGBLED_MODE_ENUM.SL_MODE_BLINK_NORMAL
RGBLED_COLOR_ENUM.SL_COLOR_BLUE
RGBLED_COLOR_ENUM.SL_COLOR_RED
Mode
Color

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


Ch3
Ch4
RGBLED_MODE_ENUM.SL_MODE_BLINK_FAST
RGBLED_MODE_ENUM.SL_MODE_BLINK_NORMAL
RGBLED_COLOR_ENUM.SL_COLOR_BLUE
RGBLED_COLOR_ENUM.SL_COLOR_RED
Mode
Color

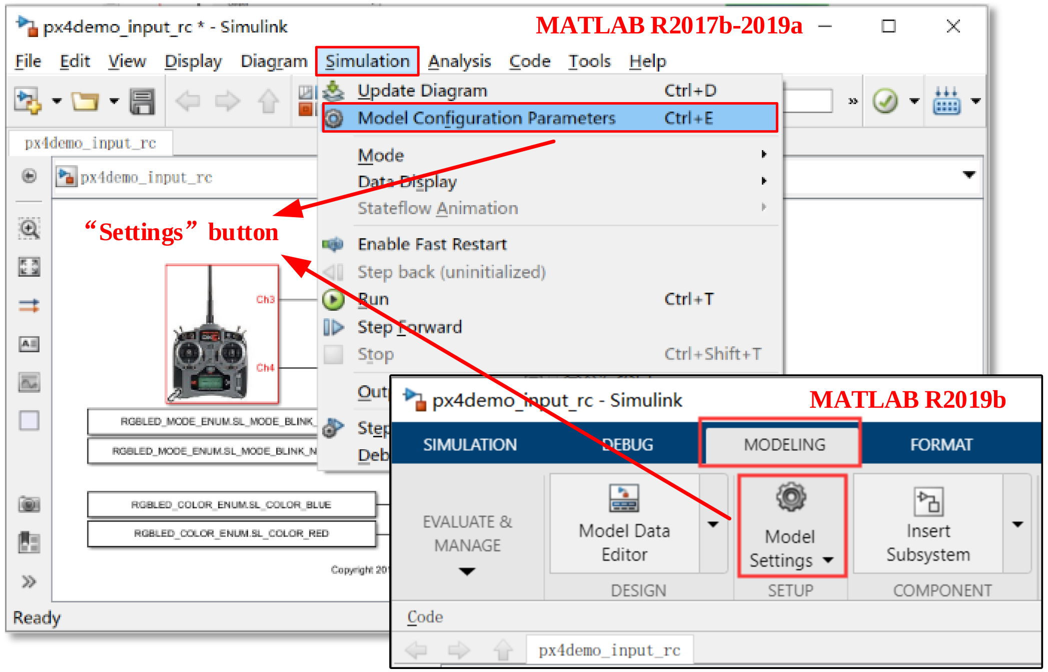
Use the CH4 channel of the RC transmitter to change the color of the LED light.

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

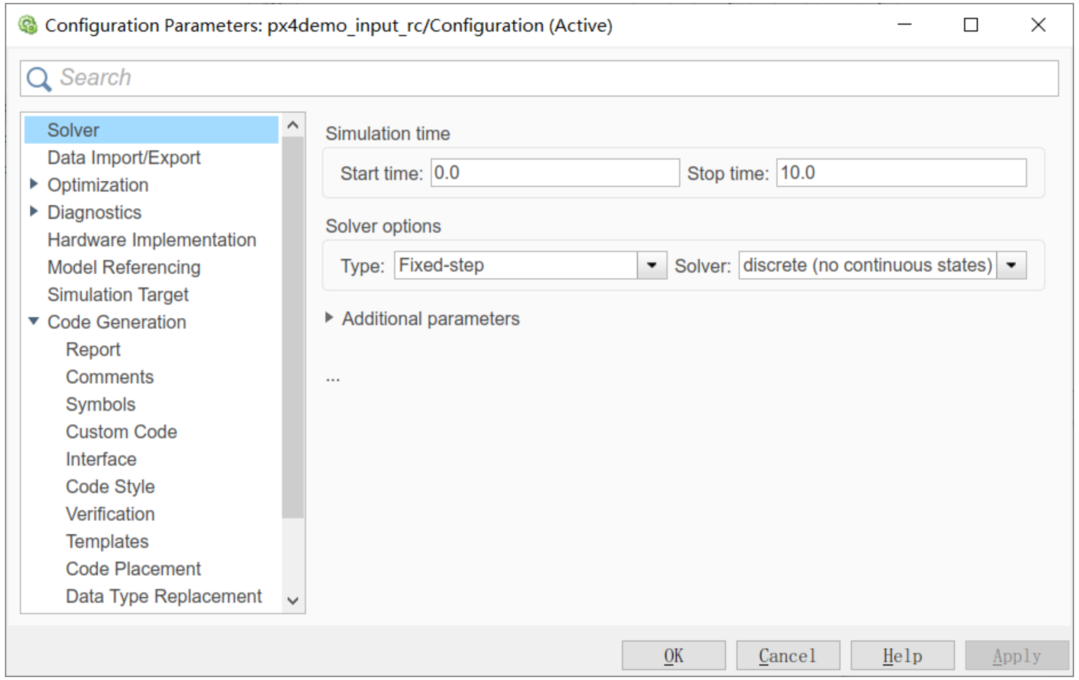
(1) For MATLAB 2017b-2019a, as shown in figure, click the “Simulation”-“Model Configuration Parameters” option on the Simulink menu bar to enter the Simulink setting dialog; for MATLAB 2019b and above, click the “Settings” button. The obtained setting dialog is presented in the figure on the right.


MATLAB R2017b-2019a

“Settings” button

MATLAB R2019b

px4demo_input_rc * - Simulink
File
Edit
View
Display
Diagram
Simulation
Analysis
Code
Tools
Help
Update Diagram Ctrl+D
Model Configuration Parameters Ctrl+E
Mode
Data Display
Stateflow Animation
Enable Fast Restart
Step back (uninitialized)
Run Ctrl+T
Step Forward
Stop Ctrl+Shift+T
px4demo_input_rc
Ch3
Ch4
RGBLED_MODE_ENUM.SL_MODE_BLINK
RGBLED_COLOR_ENUM.SL_COLOR_BLUE
RGBLED_COLOR_ENUM.SL_COLOR_RED
Ready
px4demo_input_rc - Simulink
SIMULATION
DEBUG
MODELING
FORMAT
EVALUATE & MANAGE
Model Data Editor
DESIGN
Model Settings
SETUP
Insert Subsystem
COMPONENT
Code


Configuration Parameters: px4demo_input_rc/Configuration (Active)

Simulation time

Start time: 0.0 Stop time: 10.0

Solver options

Type: Fixed-step Solver: discrete (no continuous states)

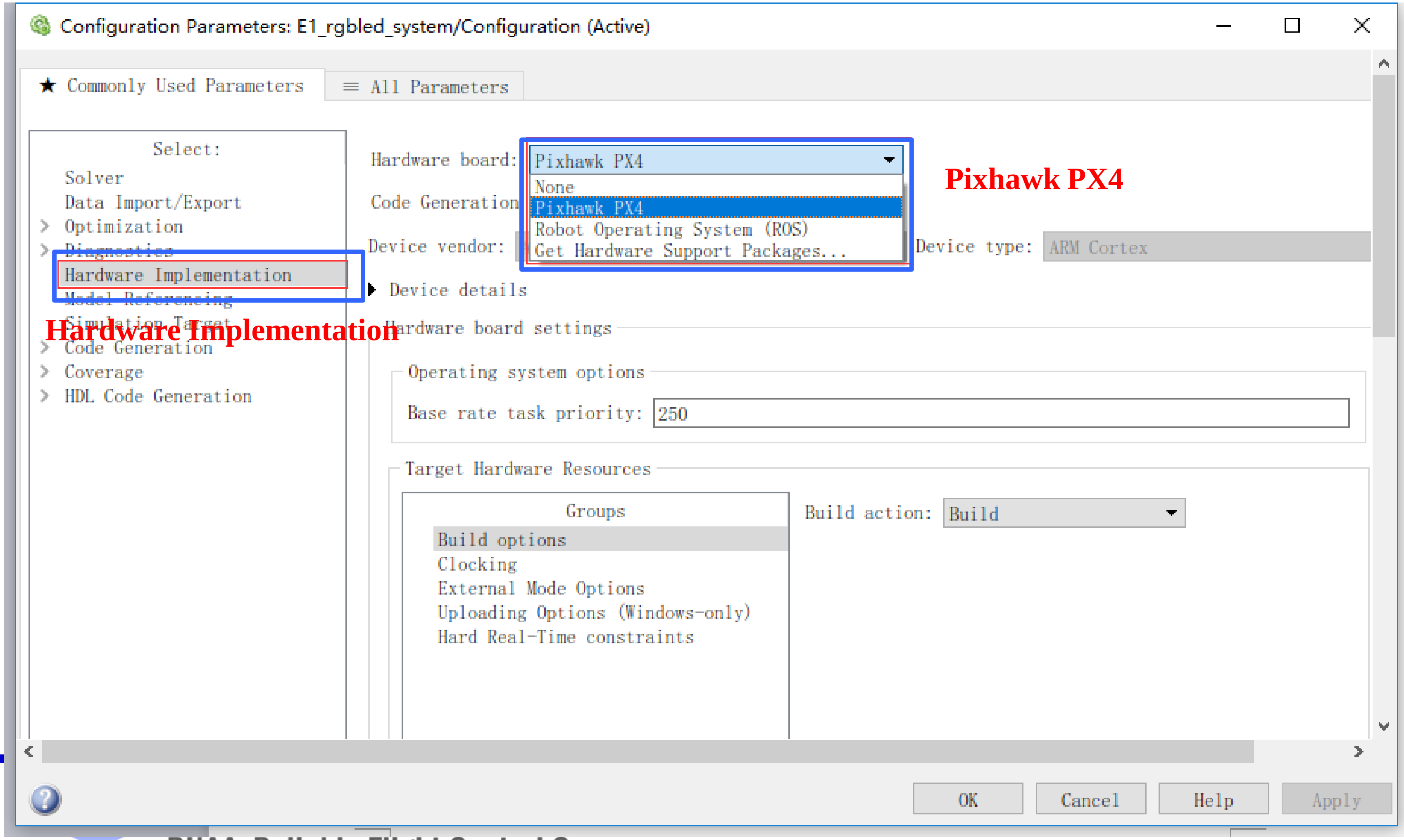
Additional parameters

Search
Solver
Data Import/Export
Optimization
Diagnostics
Hardware Implementation
Model Referencing
Simulation Target
Code Generation
Report
Comments
Symbols
Custom Code
Interface
Code Style
Verification
Templates
Code Placement
Data Type Replacement
OK
Cancel
Help
Apply

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”.



Configuration Parameters: E1_rgbled_system/Configuration (Active)

★ Commonly Used Parameters | ≡ All Parameters

Select:
- Solver
- Data Import/Export
- › Optimization
- › Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target
- › Code Generation
- › Coverage
- › HDL Code Generation

Hardware board: Pixhawk PX4
- None
- Pixhawk PX4
- Robot Operating System (ROS)
- Get Hardware Support Packages...

Code Generation

Device vendor: | Device type: ARM Cortex

▸ Device details

Hardware board settings

Operating system options

Base rate task priority: 250

Target Hardware Resources

Groups
- Build options
- Clocking
- External Mode Options
- Uploading Options (Windows-only)
- Hard Real-Time constraints

Build action: Build

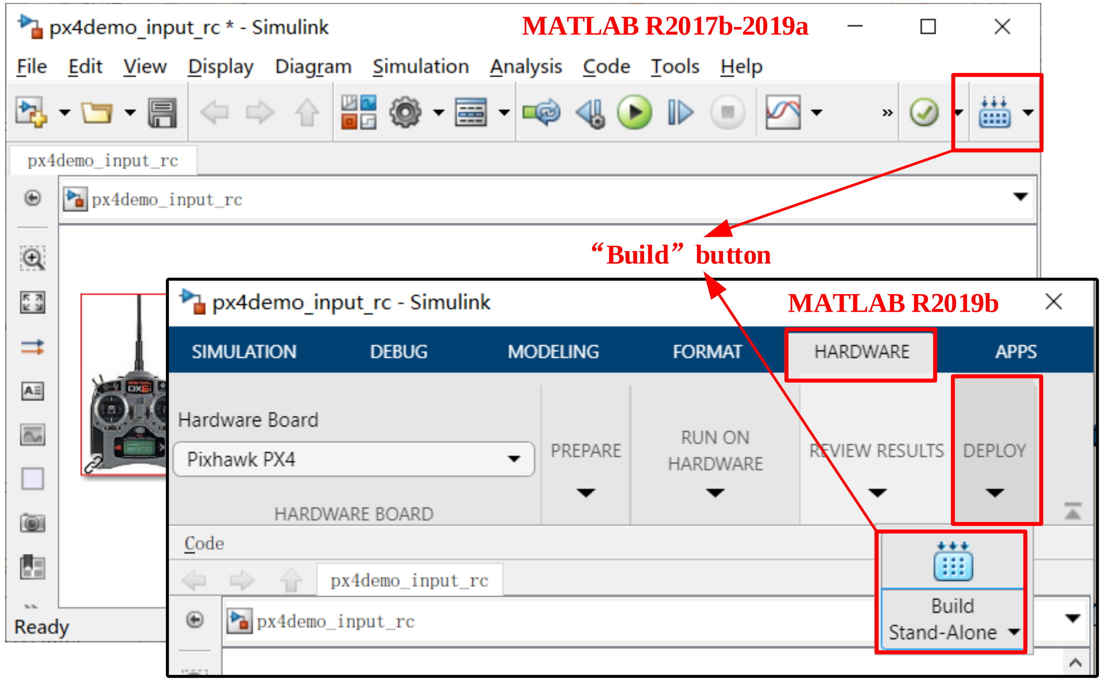
Hardware Implementation

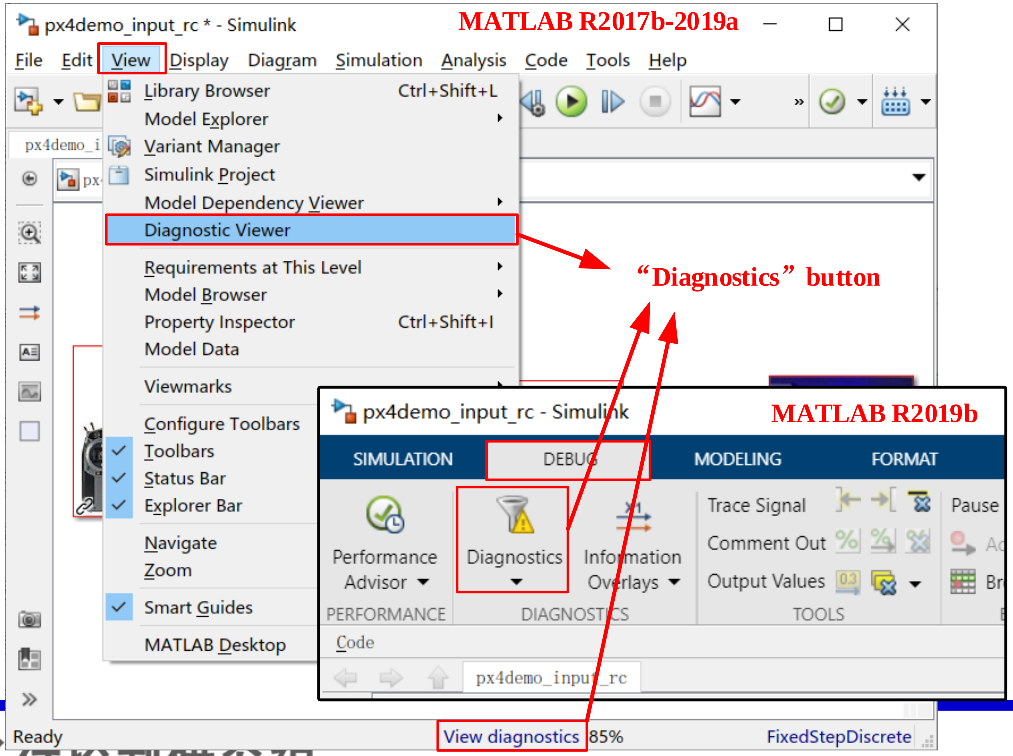
Pixhawk PX4

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.

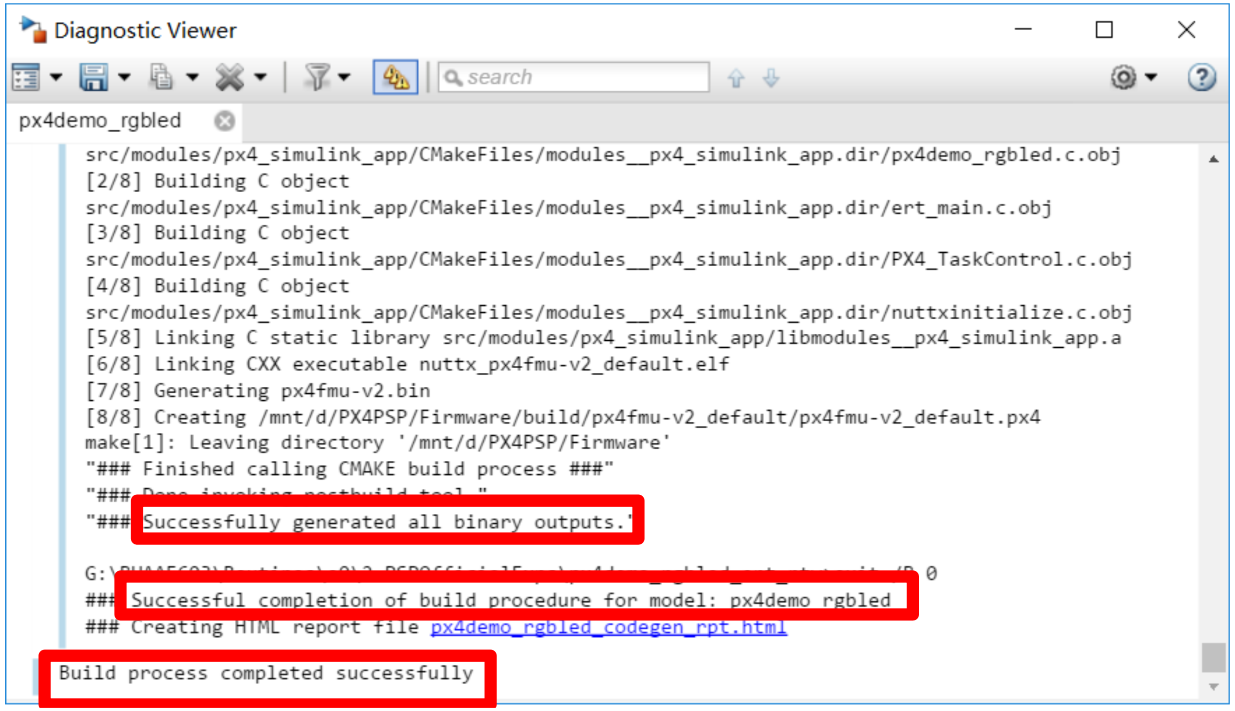

px4demo_input_rc * - Simulink
MATLAB R2017b-2019a
File Edit View Display Diagram Simulation Analysis Code Tools Help
px4demo_input_rc
“Build” button
Ready
px4demo_input_rc - Simulink
MATLAB R2019b
SIMULATION
DEBUG
MODELING
FORMAT
HARDWARE
APPS
Hardware Board
Pixhawk PX4
HARDWARE BOARD
PREPARE
RUN ON HARDWARE
REVIEW RESULTS
DEPLOY
Code
Build Stand-Alone


px4demo_input_rc * - Simulink
MATLAB R2017b-2019a
File Edit View Display Diagram Simulation Analysis Code Tools Help
Library Browser Ctrl+Shift+L
Model Explorer
Variant Manager
Simulink Project
Model Dependency Viewer
Diagnostic Viewer
Requirements at This Level
Model Browser
Property Inspector Ctrl+Shift+I
Model Data
Viewmarks
Configure Toolbars
Toolbars
Status Bar
Explorer Bar
Navigate
Zoom
Smart Guides
MATLAB Desktop
“Diagnostics” button
px4demo_input_rc - Simulink
MATLAB R2019b
SIMULATION
DEBUG
MODELING
FORMAT
Performance Advisor
Diagnostics
Information Overlays
Trace Signal
Comment Out
Output Values
Pause
PERFORMANCE
DIAGNOSTICS
TOOLS
Code
px4demo_input_rc
Ready
View diagnostics
85%
FixedStepDiscrete

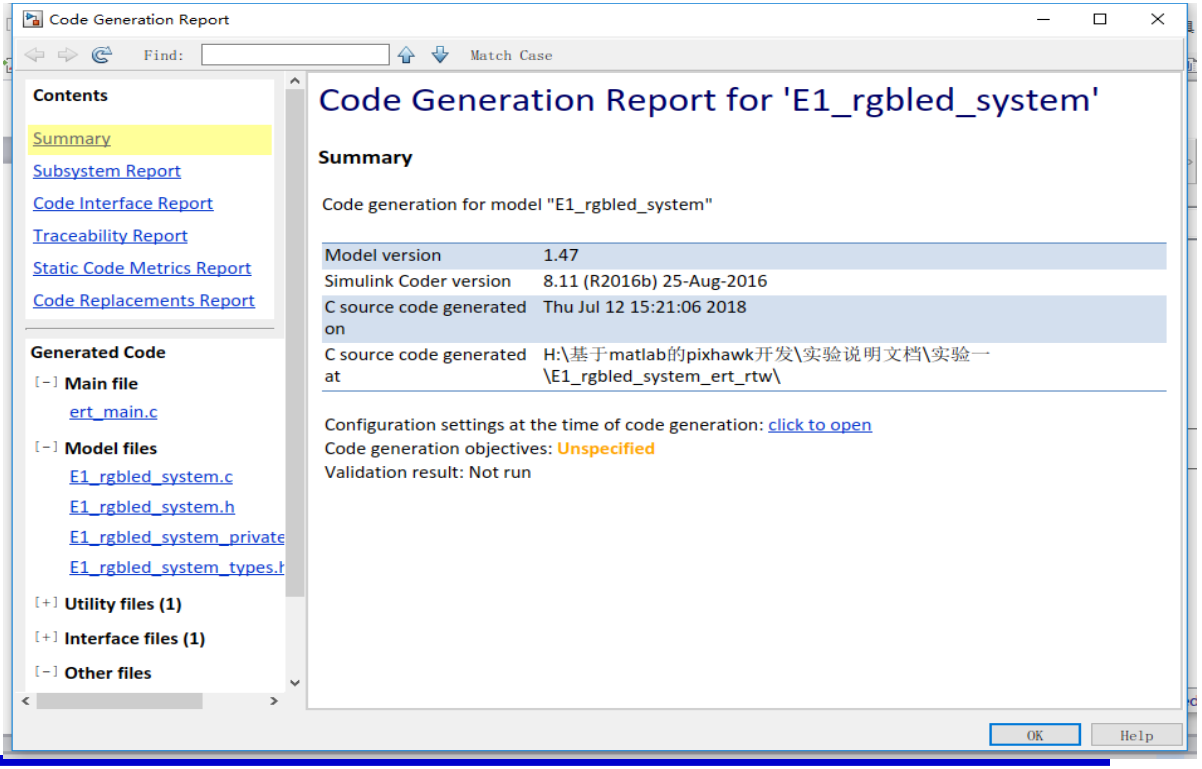
Procedure for LED Control Experiment

□ 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.


Diagnostic Viewer
search
px4demo_rgbled

```
src/modules/px4_simulink_app/CMakeFiles/modules__px4_simulink_app.dir/px4demo_rgbled.c.obj
[2/8] Building C object
src/modules/px4_simulink_app/CMakeFiles/modules__px4_simulink_app.dir/ert_main.c.obj
[3/8] Building C object
src/modules/px4_simulink_app/CMakeFiles/modules__px4_simulink_app.dir/PX4_TaskControl.c.obj
[4/8] Building C object
src/modules/px4_simulink_app/CMakeFiles/modules__px4_simulink_app.dir/nuttxinitialize.c.obj
[5/8] Linking C static library src/modules/px4_simulink_app/libmodules__px4_simulink_app.a
[6/8] Linking CXX executable nuttx_px4fmu-v2_default.elf
[7/8] Generating px4fmu-v2.bin
[8/8] Creating /mnt/d/PX4PSP/Firmware/build/px4fmu-v2_default/px4fmu-v2_default.px4
make[1]: Leaving directory '/mnt/d/PX4PSP/Firmware'
"### Finished calling CMAKE build process ###"
"### Done invoking postbuild tool."
"### Successfully generated all binary outputs."

G:\BUAAE603\Routines\e0\3_PSPOfficialFuncs\px4demo_rgbled_ert_rtw>exit /B 0
### Successful completion of build procedure for model: px4demo rgbled
### Creating HTML report file px4demo_rgbled_codegen_rpt.html

Build process completed successfully
```


Code Generation Report
Find:
Match Case
Contents
Summary
Subsystem Report
Code Interface Report
Traceability Report
Static Code Metrics Report
Code Replacements Report
Generated Code
[-] Main file
ert_main.c
[-] Model files
E1_rgbled_system.c
E1_rgbled_system.h
E1_rgbled_system_private
E1_rgbled_system_types.h
[+] Utility files (1)
[+] Interface files (1)
[-] Other files
OK
Help

Code Generation Report for 'E1_rgbled_system'

Summary

Code generation for model "E1_rgbled_system"

Model version	1.47
Simulink Coder version	8.11 (R2016b) 25-Aug-2016
C source code generated on	Thu Jul 12 15:21:06 2018
C source code generated at	H:\基于matlab的pixhawk开发\实验说明文档\实验一\E1_rgbled_system_ert_rtw\

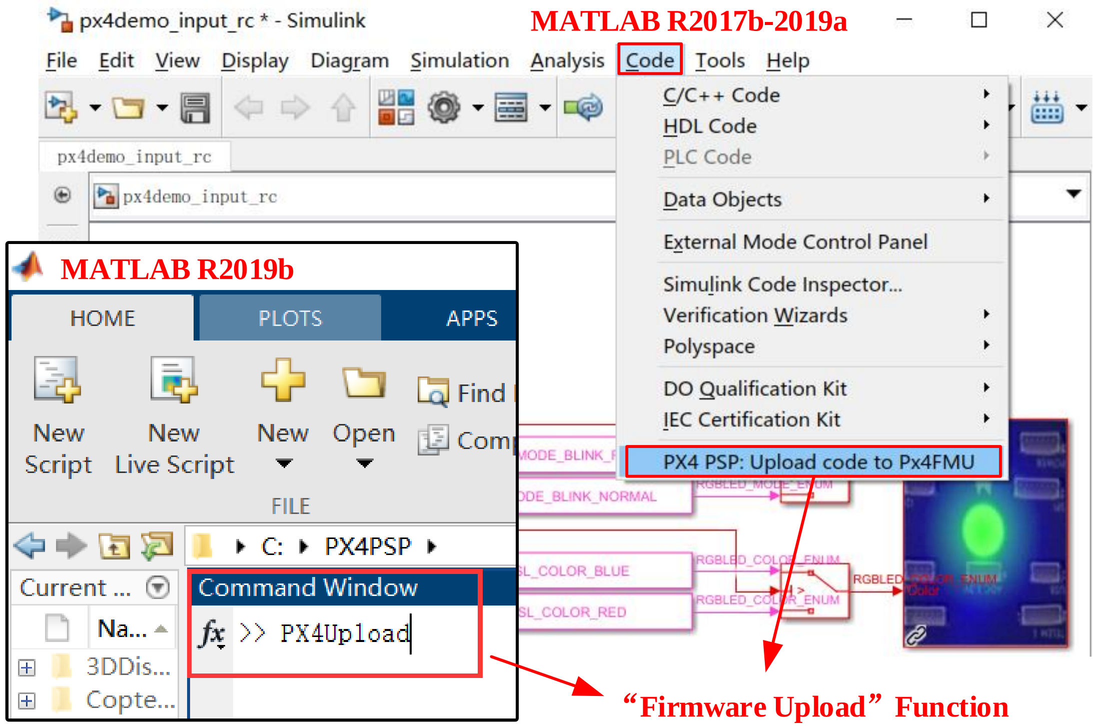
Configuration settings at the time of code generation: click to open
Code generation objectives: Unspecified
Validation result: Not run

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.



MATLAB R2017b-2019a

MATLAB R2019b

```
>> PX4Upload
```

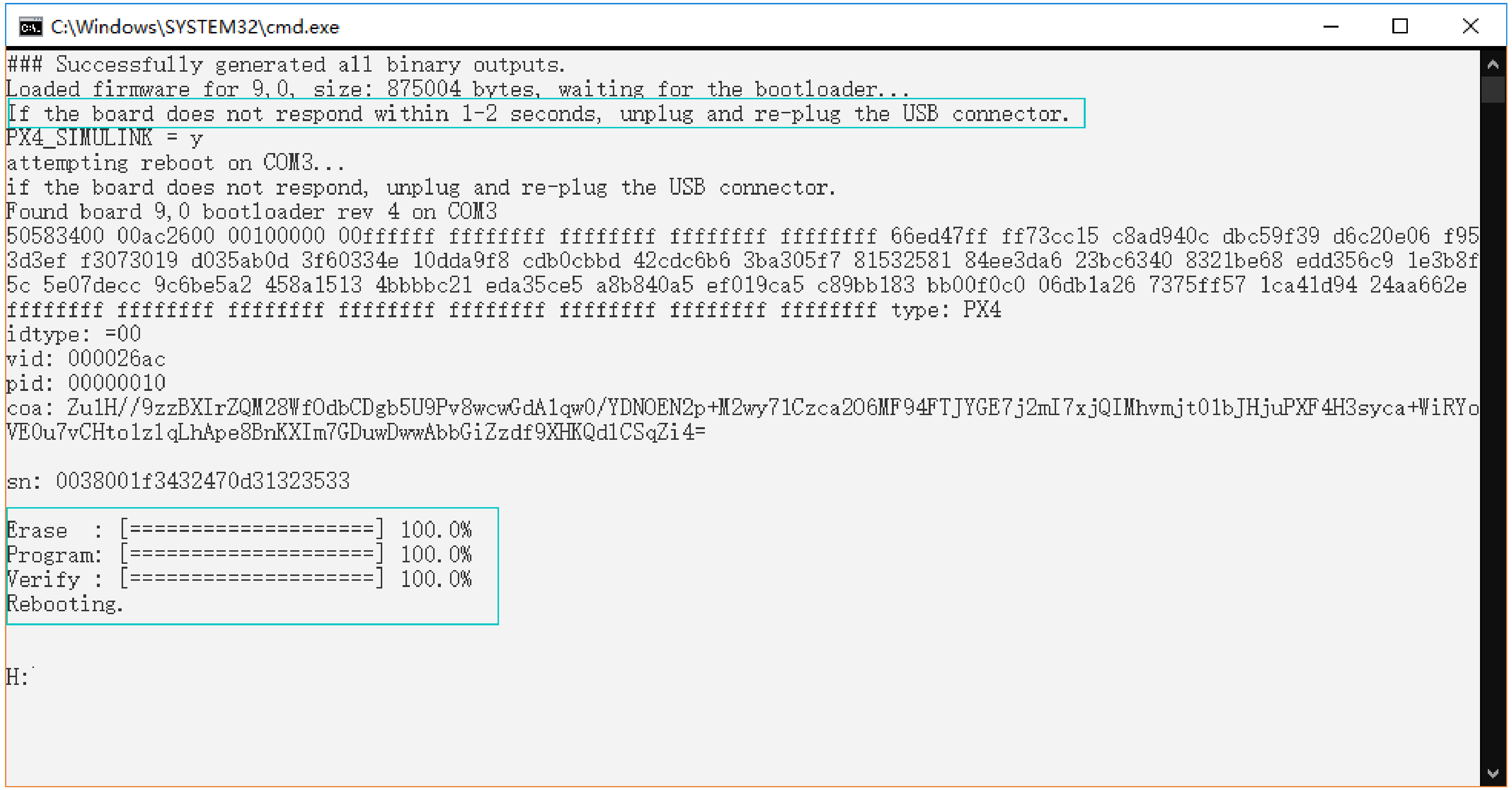
“Firmware Upload” Function

px4demo_input_rc * - Simulink
File
Edit
View
Display
Diagram
Simulation
Analysis
Code
Tools
Help
C/C++ Code
HDL Code
PLC Code
Data Objects
External Mode Control Panel
Simulink Code Inspector...
Verification Wizards
Polyspace
DO Qualification Kit
IEC Certification Kit
PX4 PSP: Upload code to Px4FMU
px4demo_input_rc
HOME
PLOTS
APPS
New Script
New Live Script
New
Open
Find
Com
FILE
C:
PX4PSP
Current ...
Na...
3DDis...
Copte...
Command Window

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.



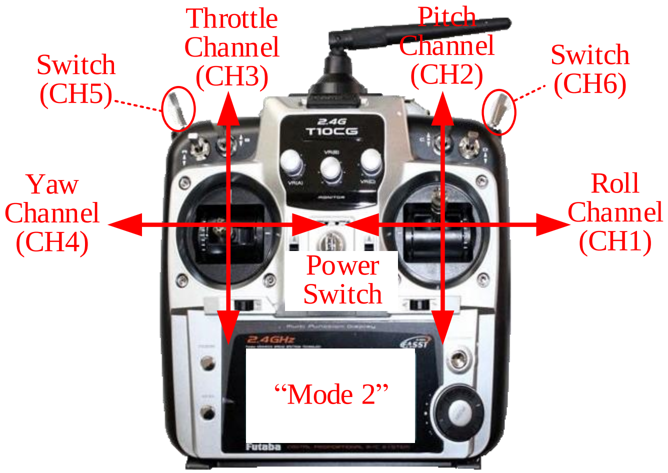
```
### Successfully generated all binary outputs.
Loaded firmware for 9,0, size: 875004 bytes, waiting for the bootloader...
If the board does not respond within 1-2 seconds, unplug and re-plug the USB connector.
PX4_SIMULINK = y
attempting reboot on COM3...
if the board does not respond, unplug and 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 edd356c9 1e3b8f
5c 5e07decc 9c6be5a2 458a1513 4bbbbc21 eda35ce5 a8b840a5 ef019ca5 c89bb183 bb00f0c0 06db1a26 7375ff57 1ca41d94 24aa662e
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff type: PX4
idtype: =00
vid: 000026ac
pid: 00000010
coa: Zu1H//9zzBXIrZQM28Wf0dbCDgb5U9Pv8wcwGdA1qw0/YDNOEN2p+M2wy71Czca2O6MF94FTJYGE7j2mI7xjQIMhvmjt01bJHjuPXF4H3syca+WiRYo
VE0u7vCHto1z1qLhApe8BnKXIm7GDuwDwwAbbGiZzdf9XHKQd1CSqZi4=

sn: 0038001f3432470d31323533

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


H:
```

Procedure for LED Control Experiment

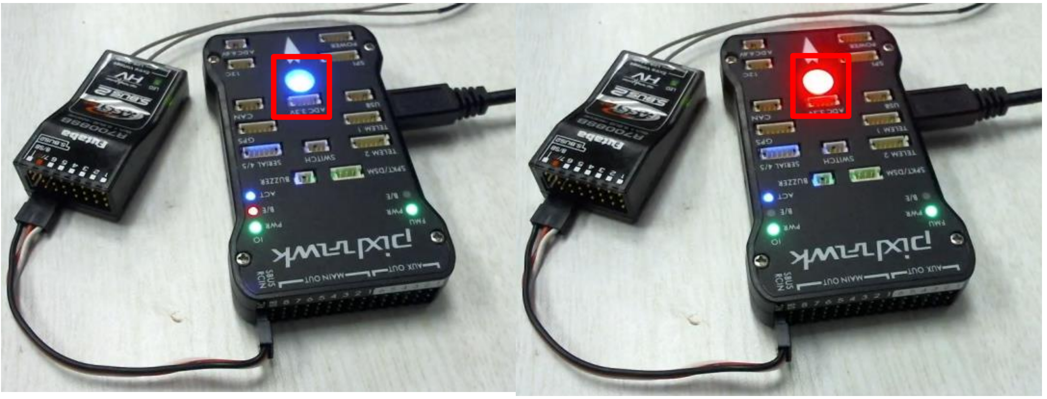
□ Experiment Result

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.


Switch (CH5)
Throttle Channel (CH3)
Pitch Channel (CH2)
Switch (CH6)
Yaw Channel (CH4)
Roll Channel (CH1)
Power Switch
"Mode 2"

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

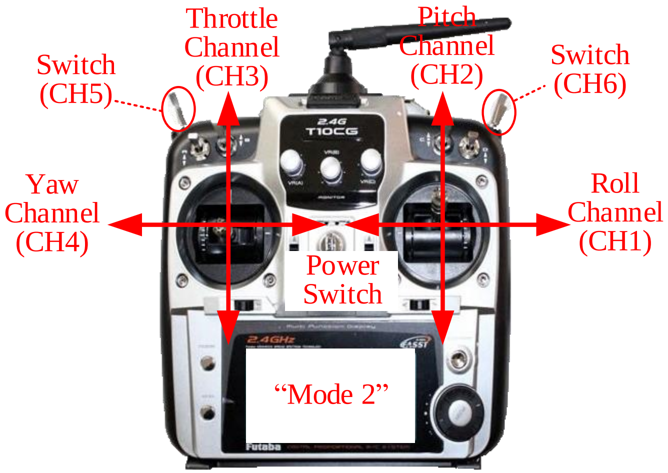


Procedure for LED Control Experiment

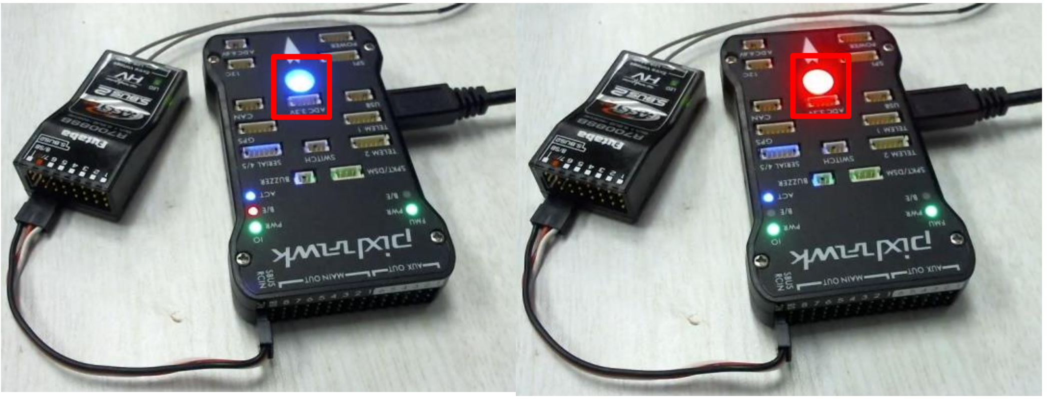
□ Experiment Result

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.

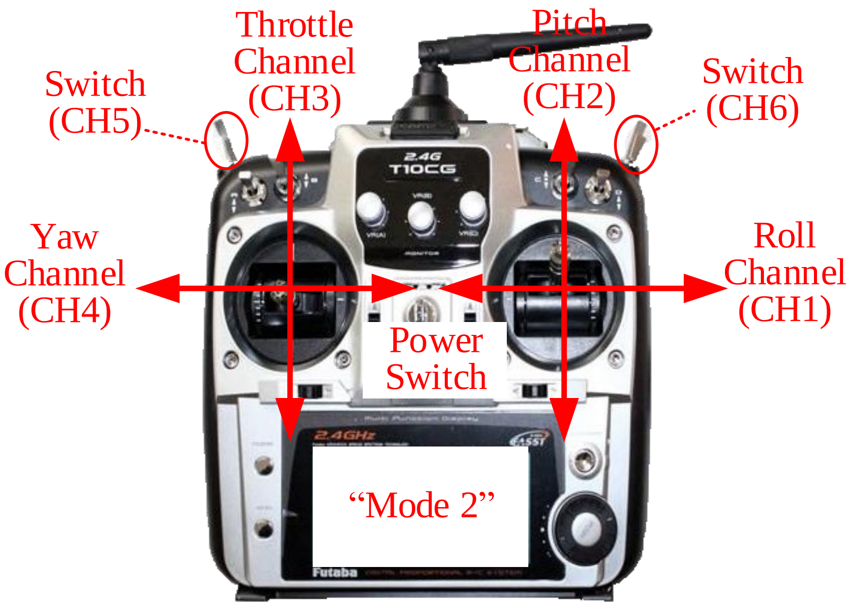

Throttle Channel (CH3)
Pitch Channel (CH2)
Switch (CH5)
Switch (CH6)
Yaw Channel (CH4)
Roll Channel (CH1)
Power Switch
"Mode 2"

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



Procedure of Attitude Contr

◻ Algorithm Design and SIL Simulatio

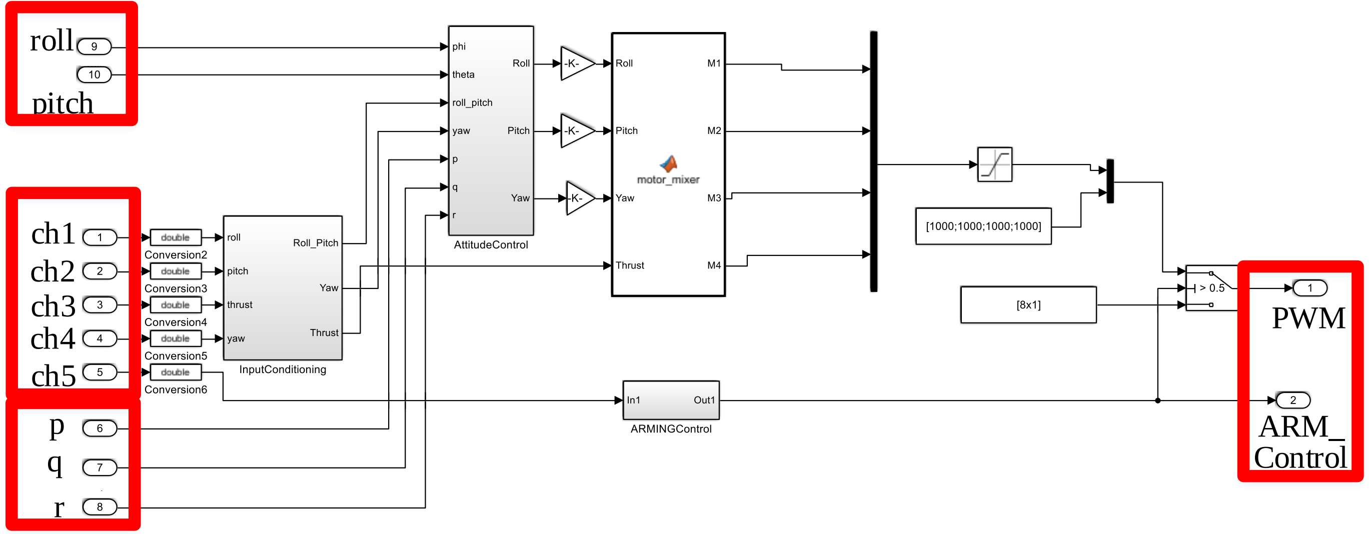

Switch (CH5)
Throttle Channel (CH3)
Pitch Channel (CH2)
Switch (CH6)
Yaw Channel (CH4)
Roll Channel (CH1)
Power Switch
"Mode 2"

(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.


roll 9
10 pitch
ch1 1
ch2 2
ch3 3
ch4 4
ch5 5
p 6
q 7
r 8
double
Conversion2
Conversion3
Conversion4
Conversion5
Conversion6
roll
pitch
thrust
yaw
Roll_Pitch
Yaw
Thrust
InputConditioning
phi
theta
roll_pitch
yaw
p
q
r
Roll
Pitch
Yaw
AttitudeControl
-K-
Roll
Pitch
Yaw
Thrust
motor_mixer
M1
M2
M3
M4
[1000;1000;1000;1000]
[8x1]
> 0.5
In1
Out1
ARMINGControl
1 PWM
2 ARM_Control

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

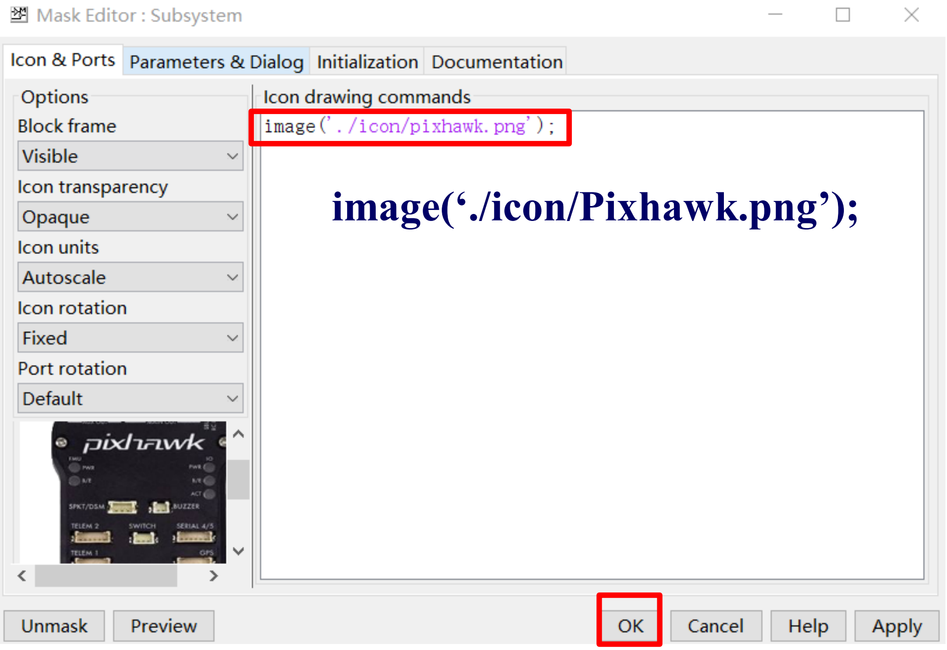
1. The throttle stick (CH3 on the RC transmitter) controls multicopters to perform the upward-and-downward movement.
2. Push up the pitch stick (CH2 < 1500) to control the multicopter flying forward.
3. Move the roll stick leftward (CH1 < 1500) to control the multicopter flying leftward.
4. Pull back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

Procedure of Attitude Con

◻ 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

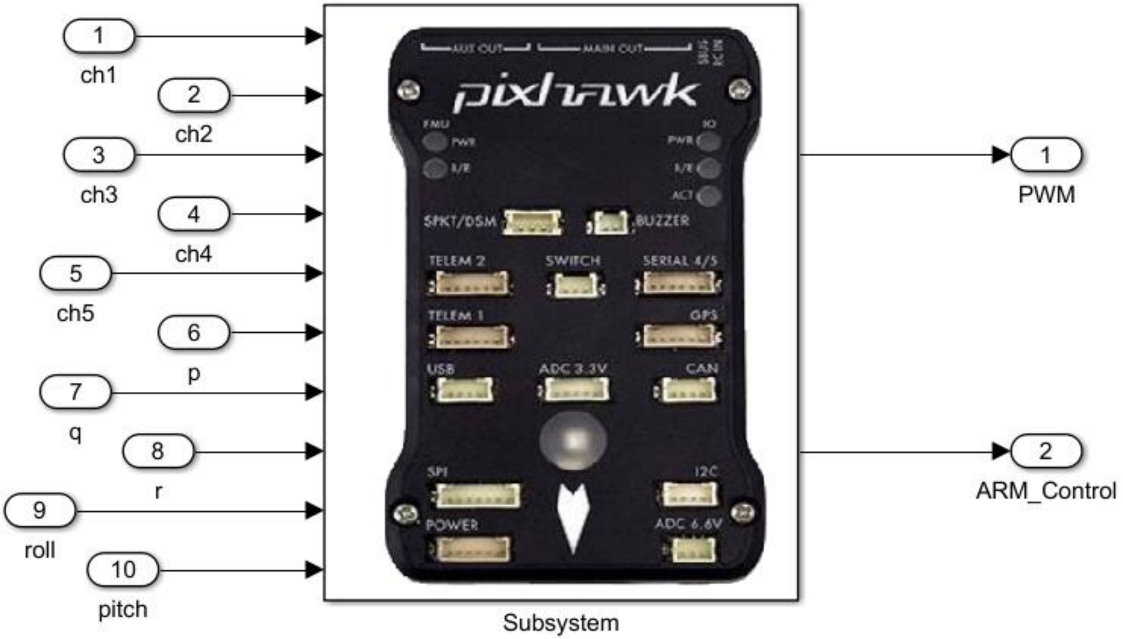

Mask Editor : Subsystem

Options
Block frame: Visible
Icon transparency: Opaque
Icon units: Autoscale
Icon rotation: Fixed
Port rotation: Default

Icon drawing commands:
```
image('./icon/pixhawk.png');
```

Icon & Ports
Parameters & Dialog
Initialization
Documentation
Unmask
Preview
OK
Cancel
Help
Apply

image(‘./icon/Pixhawk.png’);

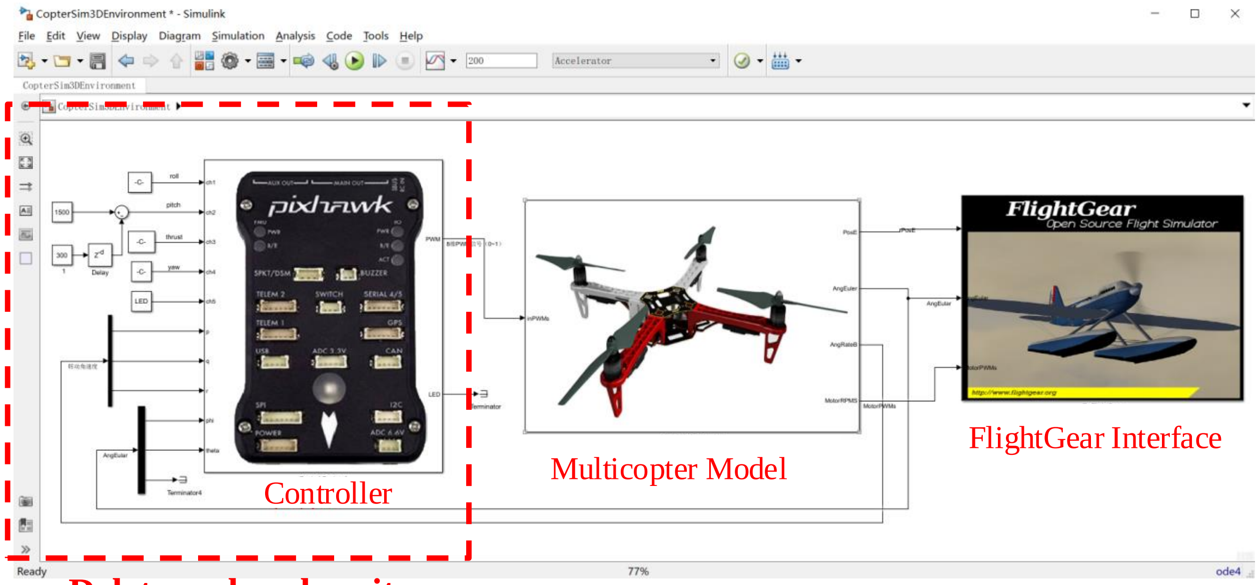

1
ch1
2
ch2
3
ch3
4
ch4
5
ch5
6
p
7
q
8
r
9
roll
10
pitch
pixhawk
1
PWM
2
ARM_Control
Subsystem

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.


CopterSim3DEnvironment * - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
200
Accelerator
CopterSim3DEnvironment
pixhawk
FlightGear
Open Source Flight Simulator
Ready
77%
ode4

Controller

Multicopter Model

FlightGear Interface

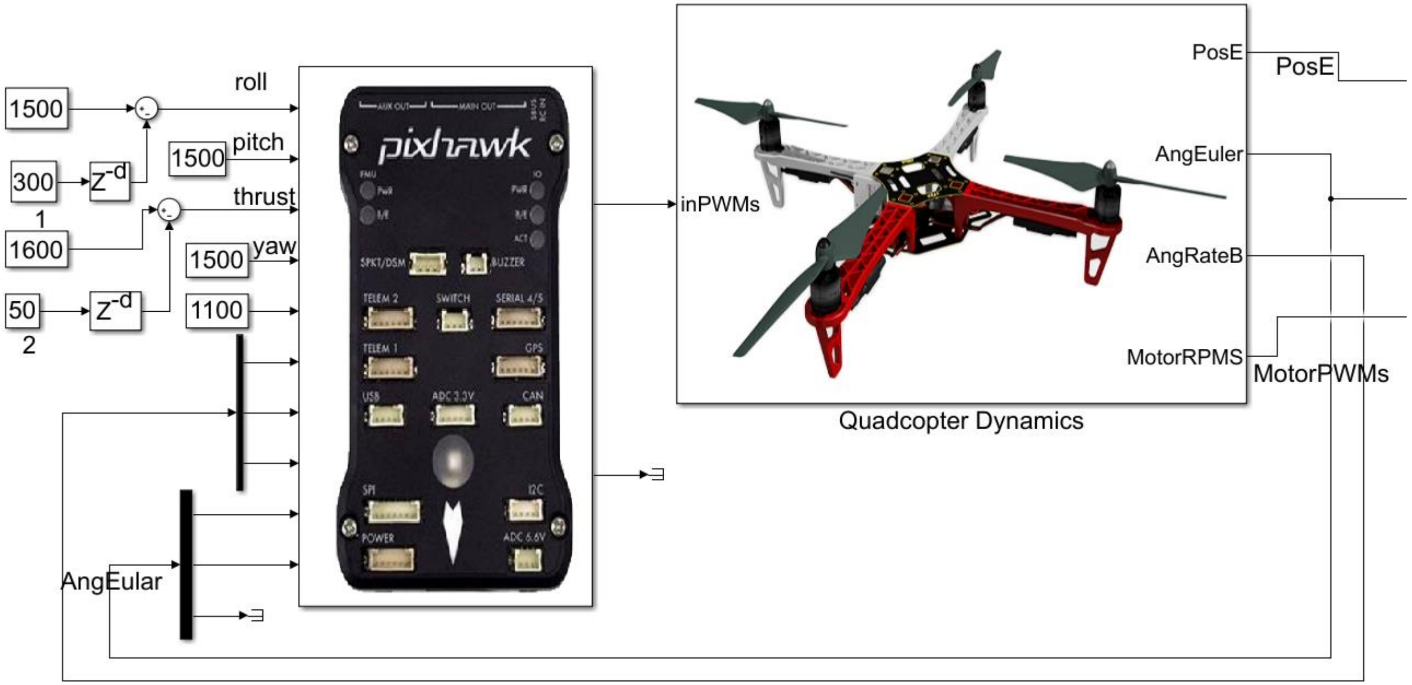
Delete and replace it

Procedure of Attitude Control Experiment

□ Algorithm Design and SIL Simulation Phase

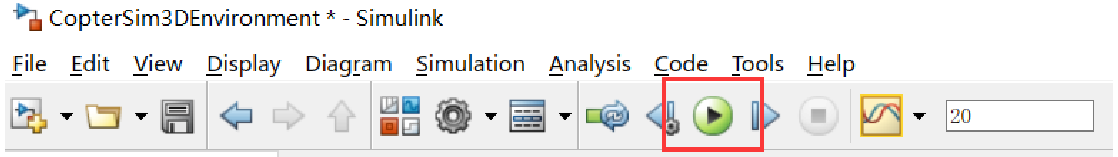
(4) Step 4: connect and configure the inputs and outputs

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


1500
roll
300
z^{-d}
1
1600
pitch
1500
thrust
50
z^{-d}
2
1500
yaw
1100
AngEular
pixhawk
inPWMs
PosE
PosE
AngEuler
AngRateB
MotorRPMS
MotorPWMs
Quadcopter Dynamics

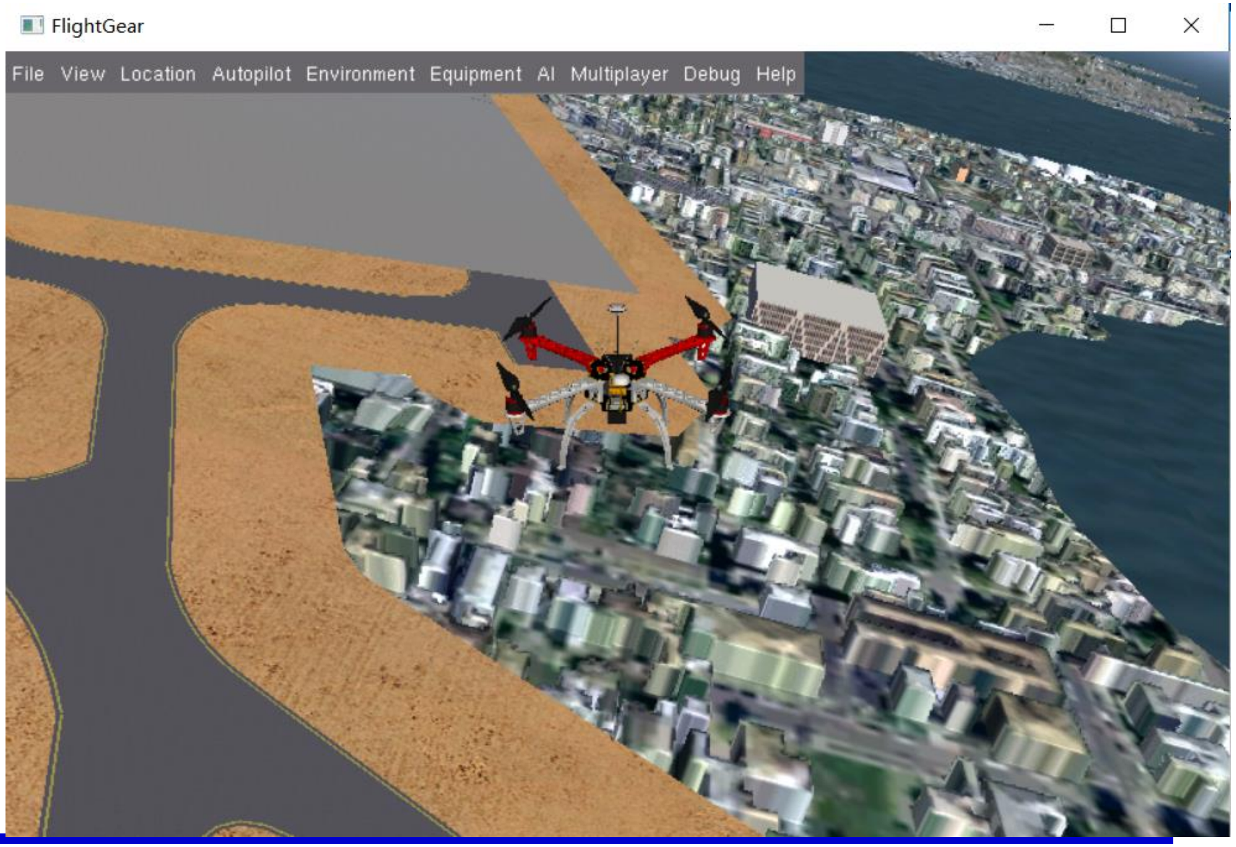
Procedure of Attitude Control Experiment

☐ Algorithm Design and SIL


CopterSim3DEnvironment * - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
20

(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 rolls left and flies leftward, indicating that the controller has achieved the expected requirements.

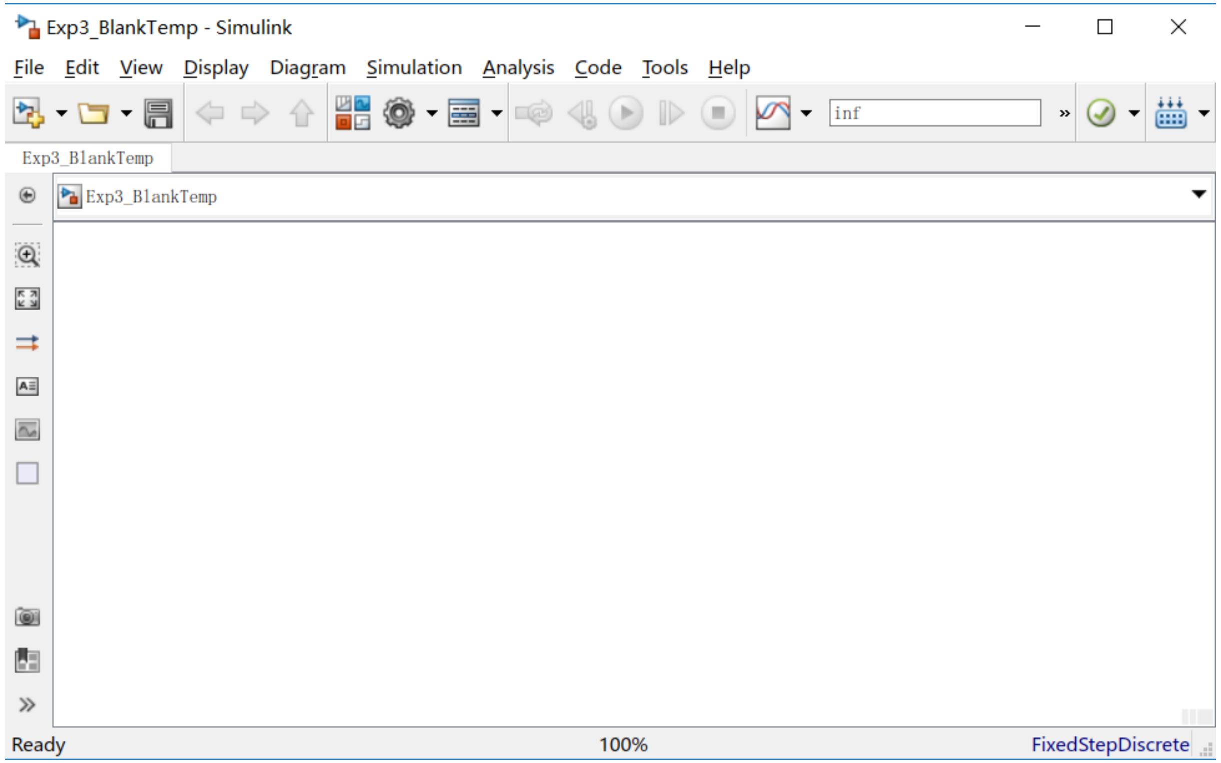

FlightGear
File View Location Autopilot Environment Equipment AI Multiplayer Debug Help

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


Exp3_BlankTemp - Simulink
File
Edit
View
Display
Diagram
Simulation
Analysis
Code
Tools
Help
inf
Exp3_BlankTemp
Exp3_BlankTemp
Ready
100%
FixedStepDiscrete

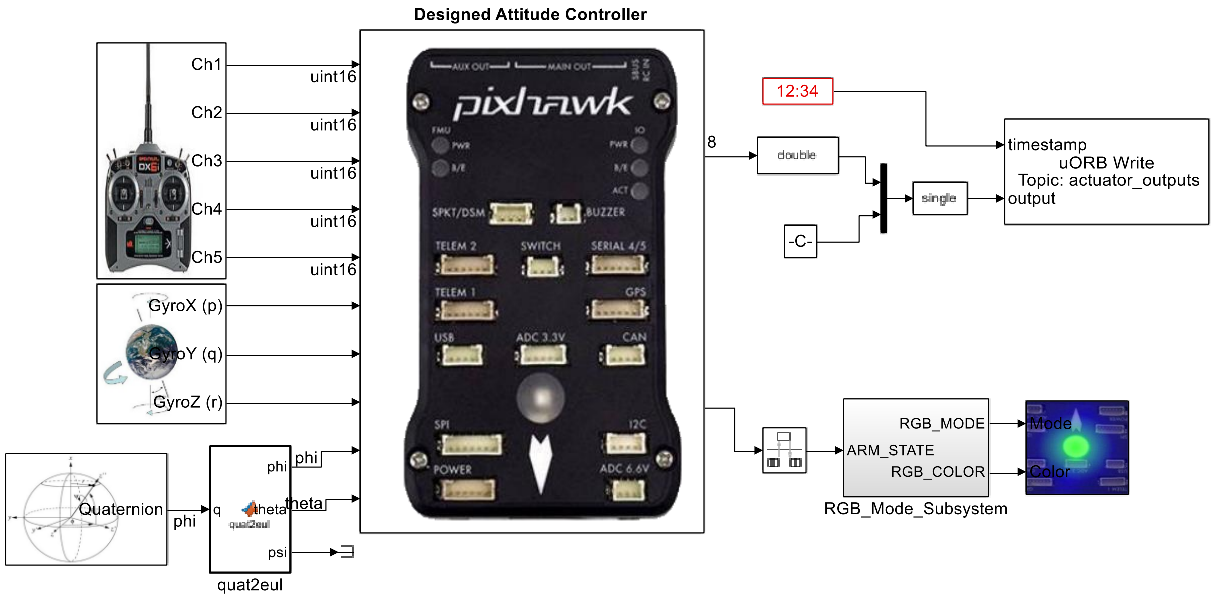
Procedure of Attitude Control Experiment

□ Code Generation and Configuration Phase

(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.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.


Designed Attitude Controller
Ch1
Ch2
Ch3
Ch4
Ch5
uint16
uint16
uint16
uint16
uint16
GyroX (p)
GyroY (q)
GyroZ (r)
Quaternion
q
phi
theta
psi
phi
theta
quat2eul
pixhawk
8
12:34
double
-C-
single
timestamp
uORB Write
Topic: actuator_outputs
output
RGB_MODE
ARM_STATE
RGB_COLOR
RGB_Mode_Subsystem
Mode
Color

Procedure of Attitude Control Experiment

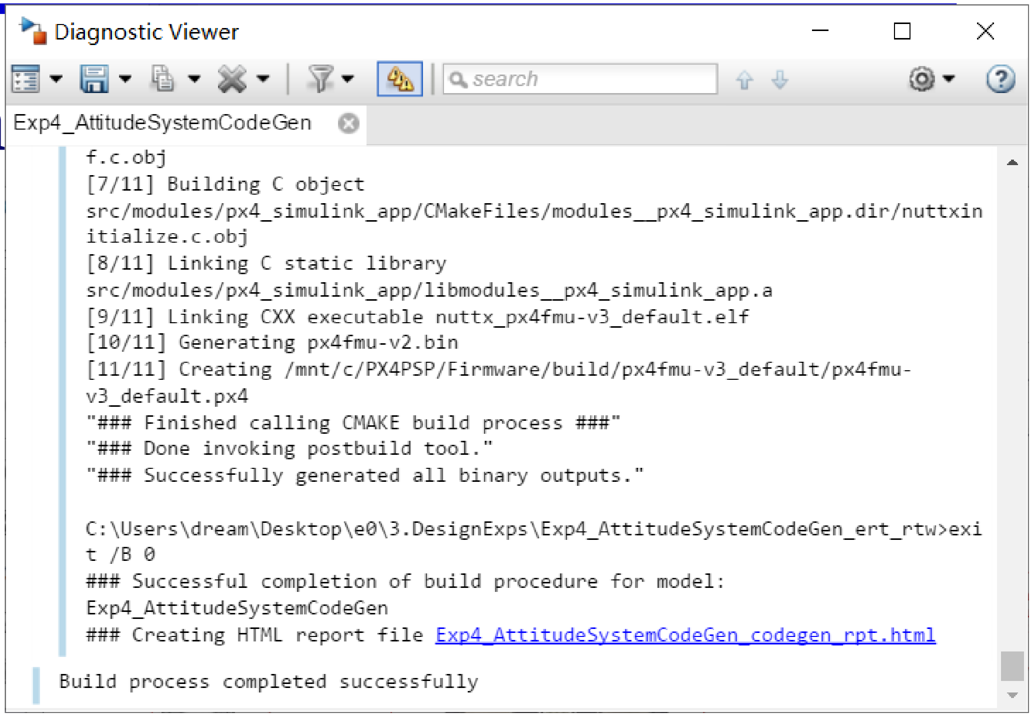
☐ Code Generation and Configura

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

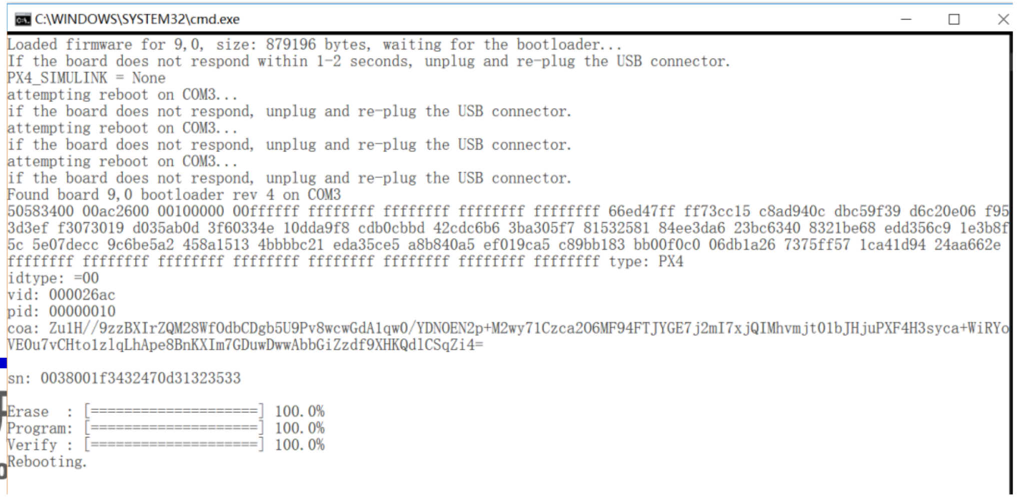
Click the “Build” button on the Simulink toolbar to automatically generate code and PX4 firmware file. The result on the upper-right figure shows a successful compilation process.

(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.



```
f.c.obj
[7/11] Building C object
src/modules/px4_simulink_app/CMakeFiles/modules__px4_simulink_app.dir/nuttxin
itialize.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
```



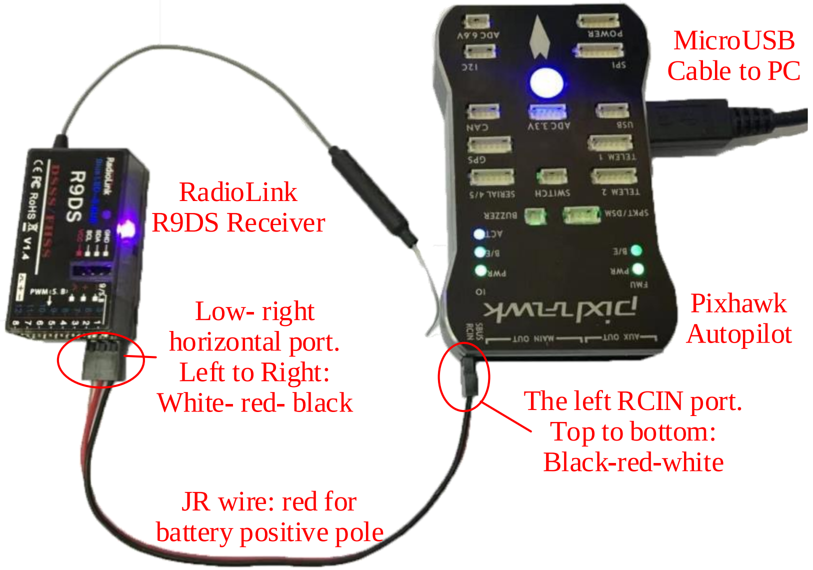
```
Loaded firmware for 9,0, size: 879196 bytes, waiting for the bootloader...
If the board does not respond within 1-2 seconds, unplug and re-plug the USB connector.
PX4_SIMULINK = None
attempting reboot on COM3...
if the board does not respond, unplug and re-plug the USB connector.
attempting reboot 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 edd356c9 1e3b8f
5c 5e07decc 9c6be5a2 458a1513 4bbbbc21 eda35ce5 a8b840a5 ef019ca5 c89bb183 bb00f0c0 06db1a26 7375ff57 1ca41d94 24aa662e
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff type: PX4
idtype: =00
vid: 000026ac
pid: 00000010
coa: Zu1H//9zzBXIrZQM28Wf0dbCDgb5U9Pv8wcwGdA1qw0/YDNOEN2p+M2wy71Czca206MF94FTJYGE7j2mI7xjQIMhvmjt01bJHjuPXF4H3syca+WiRYo
VE0u7vCHto1z1qLhApe8BnKXIm7GDuwDwwAbbGiZzdf9XHKQd1CSqZi4=

sn: 0038001f3432470d31323533

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

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.


MicroUSB Cable to PC
RadioLink R9DS Receiver
Pixhawk Autopilot
Low- right horizontal port. Left to Right: White- red- black
The left RCIN port. Top to bottom: Black-red-white
JR wire: red for battery positive pole

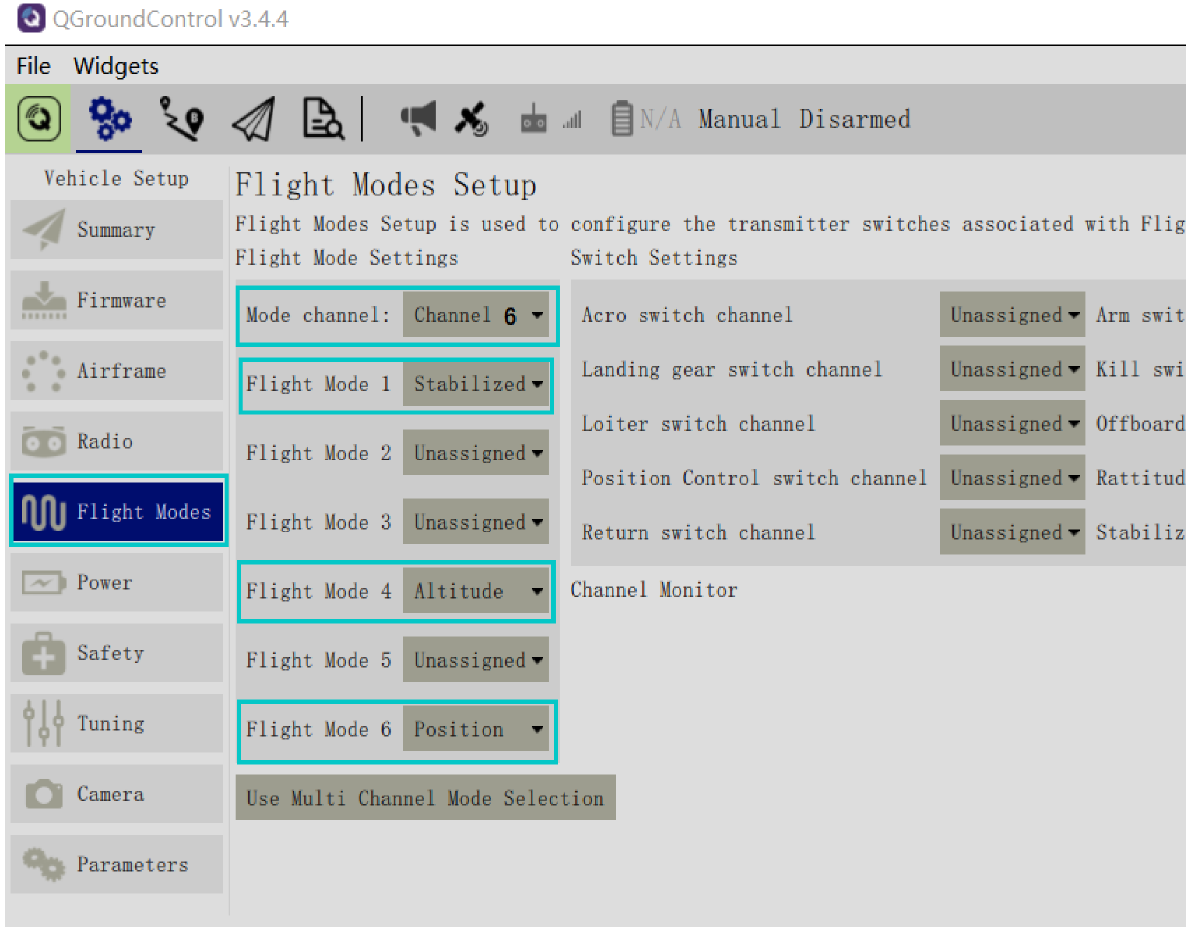
Procedure of Attitude Control Experiment

□ 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.


QGroundControl v3.4.4
File
Widgets
N/A
Manual
Disarmed
Vehicle Setup
Summary
Firmware
Airframe
Radio
Flight Modes
Power
Safety
Tuning
Camera
Parameters
Use Multi Channel Mode Selection

Flight Modes Setup

Flight Modes Setup is used to configure the transmitter switches associated with Flig

Flight Mode Settings

Mode channel: Channel 6
Flight Mode 1 Stabilized
Flight Mode 2 Unassigned
Flight Mode 3 Unassigned
Flight Mode 4 Altitude
Flight Mode 5 Unassigned
Flight Mode 6 Position

Switch Settings

Acro switch channel Unassigned
Landing gear switch channel Unassigned
Loiter switch channel Unassigned
Position Control switch channel Unassigned
Return switch channel Unassigned

Arm swit
Kill swi
Offboard
Rattitud
Stabiliz

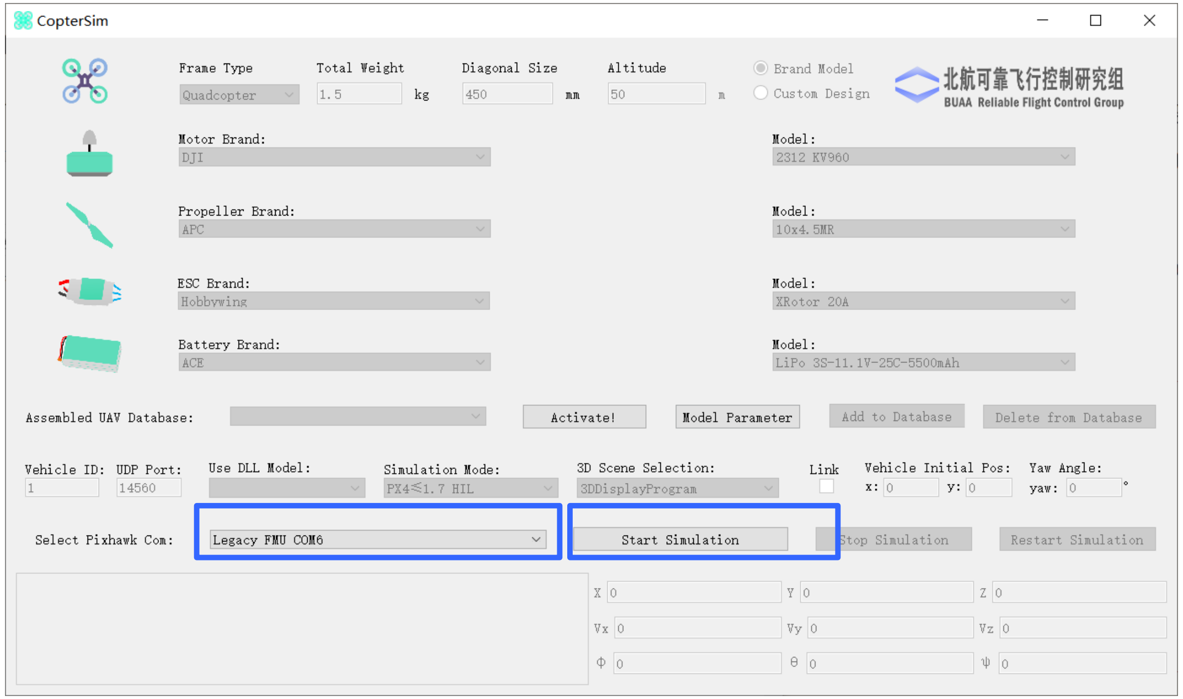
Channel Monitor

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.



CopterSim

Frame Type: Quadcopter
Total Weight: 1.5 kg
Diagonal Size: 450 mm
Altitude: 50 m
● Brand Model
○ Custom Design

Motor Brand: DJI
Model: 2312 KV960

Propeller Brand: APC
Model: 10x4.5MR

ESC Brand: Hobbywing
Model: XRotor 20A

Battery Brand: ACE
Model: LiPo 3S-11.1V-25C-5500mAh

Assembled UAV Database:

Vehicle ID: 1
UDP Port: 14560
Use DLL Model:
Simulation Mode: PX4≤1.7 HIL
3D Scene Selection: 3DDisplayProgram
Link ☐
Vehicle Initial Pos: x: 0 y: 0
Yaw Angle: yaw: 0 °

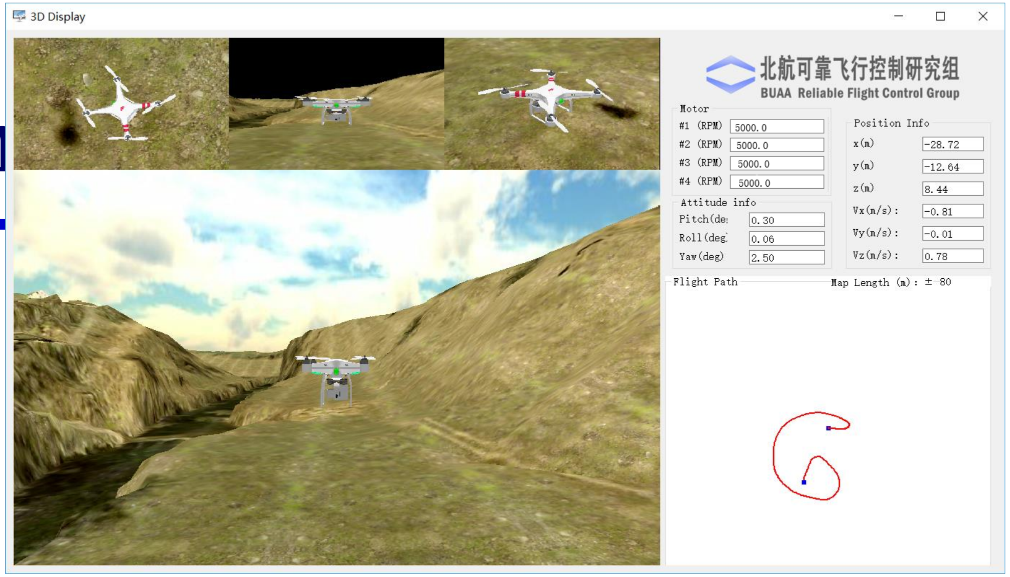
Select Pixhawk Com: Legacy FMU COM6

X 0 Y 0 Z 0
Vx 0 Vy 0 Vz 0
Φ 0 θ 0 ψ 0

北航可靠飞行控制研究组
BUAA Reliable Flight Control Group
Activate!
Model Parameter
Add to Database
Delete from Database
Start Simulation
Stop Simulation
Restart Simulation

Procedure of Attitude Con

☐ HIL Simulation Phase



Motor
#1 (RPM) 5000.0
#2 (RPM) 5000.0
#3 (RPM) 5000.0
#4 (RPM) 5000.0

Attitude info
Pitch(de 0.30
Roll(deg 0.06
Yaw(deg) 2.50

Position Info
x(m) -28.72
y(m) -12.64
z(m) 8.44
Vx(m/s): -0.81
Vy(m/s): -0.01
Vz(m/s): 0.78

Map Length (m): ± 80

(12) Step 12: 3DDisplay configuration

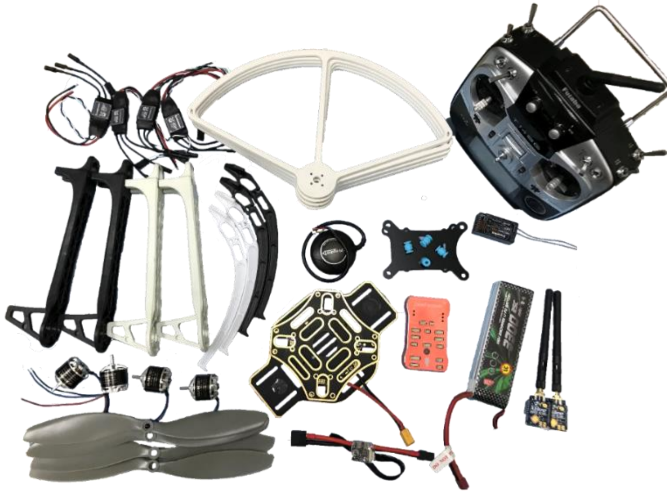
1. Double-click the 3DDisplay shortcut on the desktop to open it. This software does not require any configuration; it passively receives the flight attitude and trajectory information of multicopters sent by CopterSim and displays it in real-time.
2. Move the throttle stick on the RC transmitter to 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

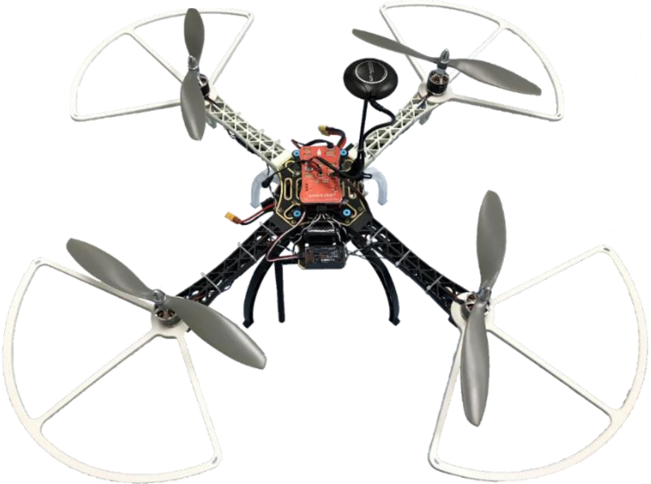
□ Flight Test Phase and Results

(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

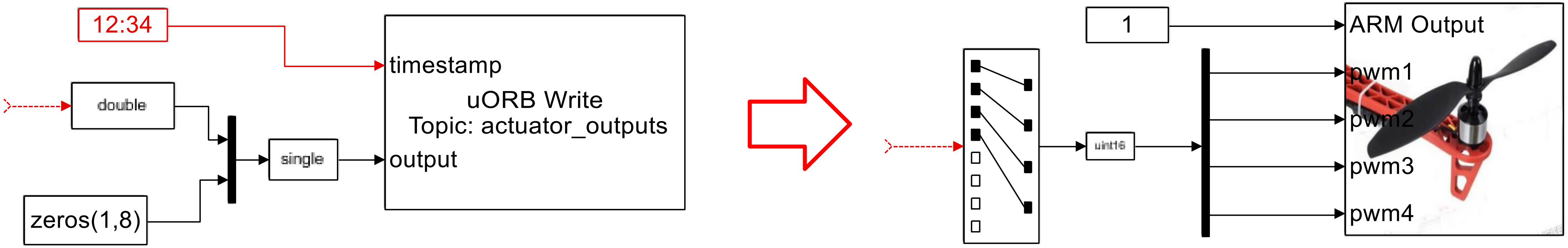


Procedure of Attitude Control Experiment

☐ 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.slx”. Then, click the Simulink “compile” button to compile the controller into PX4 firmware and upload it to Pixhawk.


12:34
double
zeros(1,8)
single
timestamp
uORB Write
Topic: actuator_outputs
output
1
uint16
ARM Output
pwm1
pwm2
pwm3
pwm4

Procedure of Attitude Control Experiment

☐ Flight Test Phase and Results

(15) Step 15: preparation for flight tests

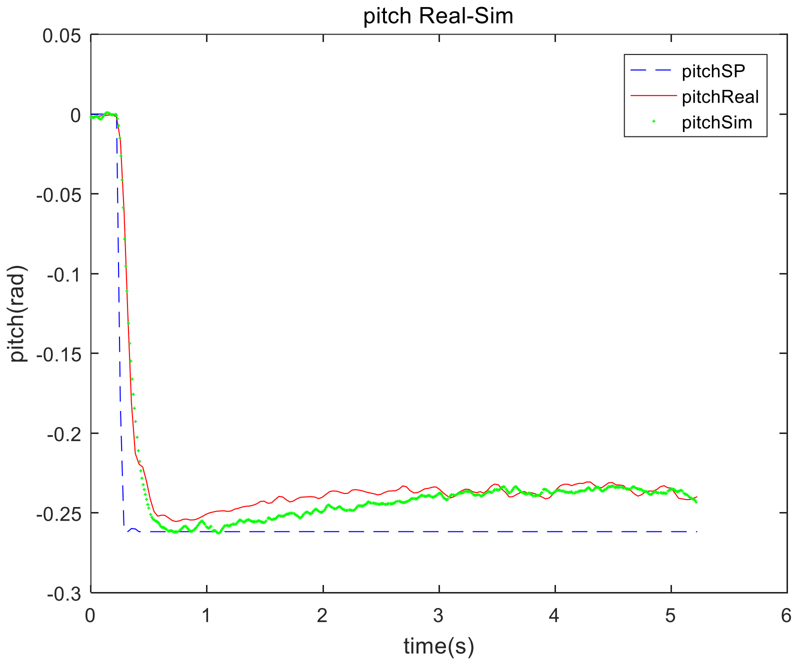
Owing to the lack of complete failsafe logic for the generated control algorithm, safety problems should be fully considered to prevent the uncertainty in outdoor flight tests. For example, the tests should be carried out in a relatively open area (such as a lawn) and on a nice day with good weather 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

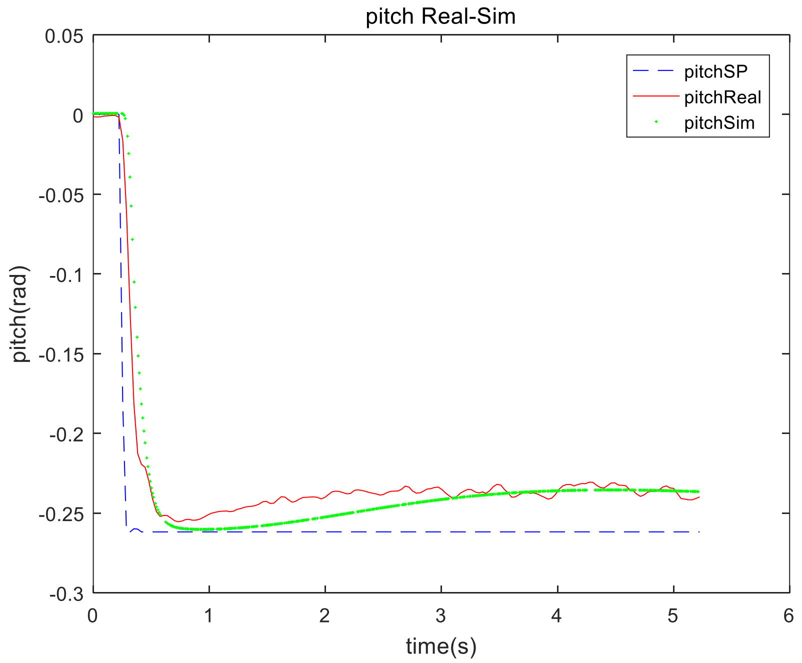
(16) Step 16: test results and analysis

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.

Thank You!