5.6 Control Development and Testing

Chapter Outline

Low-Level Control Development and Experimental Validation

Chapter 5
Filtering, State Estimation,
and Low-Level Control

5.1 Filtering and Low-Level Control

5.2 Sensor Calibration and Estimation

5.3 Controller Design and Allocation

5.4 Control Performance and Tuning

5.5 PX4 Architecture and Interfaces

5.6 Control Development and Testing

5.7 Other Vehicles and Advanced Cases

5.8 Exercises and Classroom Discussion

5 Units in This Section

5.6.1 Control Algorithm Development and Testing
Textbook: Sec. 5.2.2.1

5.6.2 Position Control and Semi-Autonomous Modes
Textbook: Secs. 5.1.2.4, 5.3.1

5.6.3 Controller Development and Code Generation
Textbook: Sec. 5.2.2.2

5.6.4 Software- and Hardware-in-the-Loop Simulation
Textbook: Sec. 5.2.2.2

5.6.5 Real-Vehicle Experiments and Result Review
Textbook: Secs. 5.2.2.3, 5.1.4

5.6.1 Control Algorithm Development and Testing: Objectives

Each Stage Adds Evidence for Validation

Main example: attitude control in 3.DesignExps; add the position outer loop in stages.

Stage	Input → Output	Record at This Stage
Algorithm Model	Targets, States → Control Requests	Units, Directions, Limits, Feedback Sources
Code Generation	Control Model → Target Program	Model Hash, Configuration, Build Report
In-Loop Simulation	Program, Simulated States → Response	Errors, Update Intervals, Computation Time
Physical Testing	Real Observations → Real Actions	Vehicle Conditions, Protections, Actual Outputs

Firmware: installed default v1.12.3; this example recommends the v1.14 series. See “Control Outputs Across Versions” in slide section 5.5.

5.6.1 Control Algorithm Development and Testing: Controller Overview

Inputs, Control Chain, and Arming Bypass

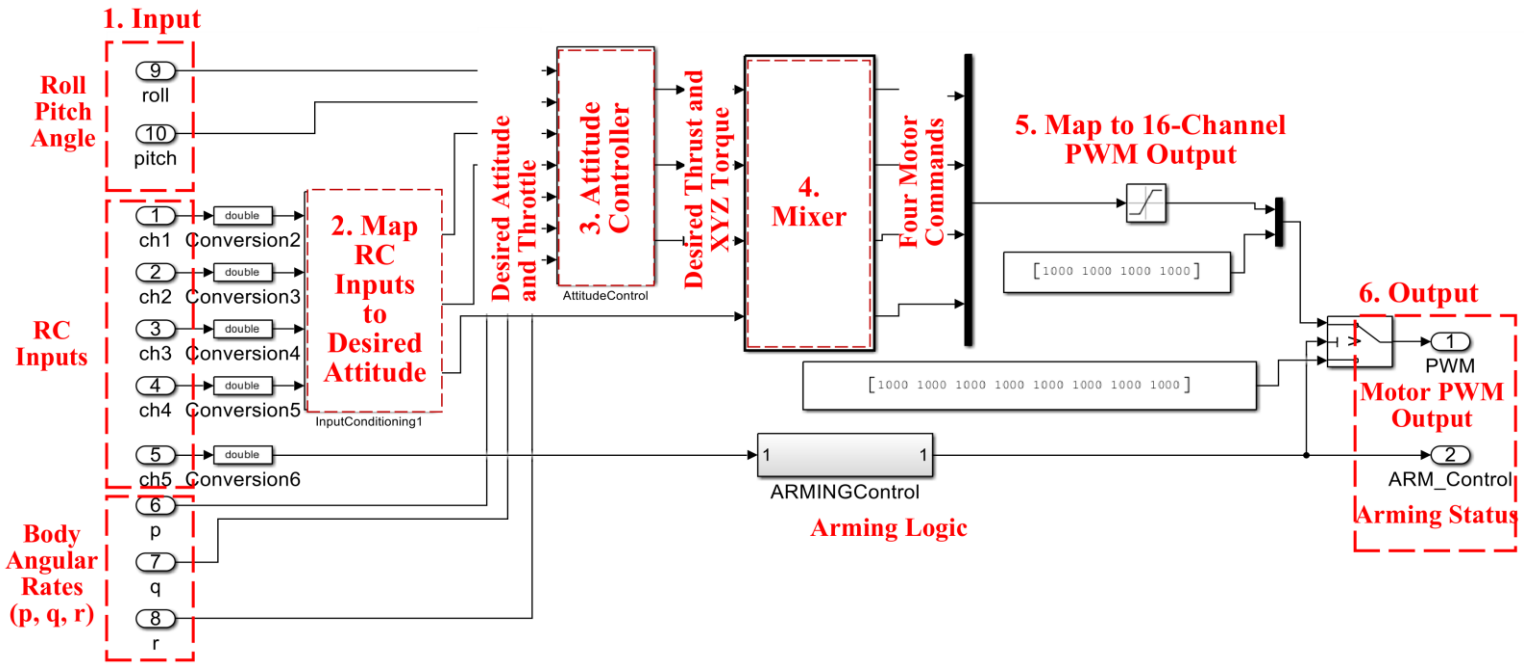


Fig. 5.7 Quadcopter Attitude Controller Architecture

CH5 uses the arm bypass. Exp1_AttitudeController.slx outputs 8 pulse-width modulation (PWM) channels. The figure labels 16.

5.6.1 Control Algorithm Development and Testing: Inputs and Outputs

Port Units, Directions, and Dimensions

Port	Meaning and Units	Exp1 Model Rules
CH1～CH4	Roll/Pitch/Throttle/Yaw Rate 1100～1900 μs	CH1/2/4 Deadband ±40 μs CH3 Also Has a Deadband
roll, pitch	Roll and Pitch Euler Angles Units: rad	Angle Feedback No Yaw-Angle Input
p, q, r	Body Forward/Right/Down Angular Rates Units: rad/s	r Enters yaw_rate Yaw-Rate Loop Is Present
CH5 **ARM_Control**	Pulse-Width Input Boolean Armed Output	CH5 ≥ 1500 μs Armed Output Is true
PWM	8 Motor and Auxiliary Pulse Widths Units: μs	First 4: 1000～2000 μs Last 4: 1000 μs

Source: Exp1_AttitudeController.slx. Below 1500, CH1 commands left and CH2 forward, following the textbook convention.

5.6.1 Control Algorithm Development and Testing: Roll Calculation

Roll Target into the Proportional–Integral (PI) Loop

Given: CH1=1660 μs, angle and angular rate both 0, initial integral contribution 0, deadband ±40 μs.

$$\phi_{\mathrm{d}} = 0.5\frac{1660-1500}{400} = 0.2\ \mathrm{rad}$$

Roll-Angle Target → **Attitude PI** → **Angular-Rate Request**

$$u_{\mathrm{att}} = \frac{0.6301386251}{1.2}(0.2-0) = 0.105023$$

÷400 normalizes 1100～1900 μs to ±1. ×0.5 gives 0.5 rad at full stick. Use the original value outside the deadband.

kFactor=1.2 scales gains across channels. The initial integral contribution is 0, so the first step contains only the proportional term.

5.6.1 Control Algorithm Development and Testing: Rate Calculation

Normalized Rate Request to Mixer Input

Input: attitude-loop output 0.105023, angular rate 0 rad/s, initial integral contribution 0.

Angular-Rate Request − Normalized Feedback → **Angular-Rate PI roll_rate** → **Output Scaling Gain 1/10**

$$u_{\text{rate}} = \frac{1.3153383656}{1.2}\left(u_{\text{att}} - \frac{0}{3}\right) = 0.115117$$

÷3: Normalize angular-rate feedback by 3 rad/s before comparing it with the request.
×1/10: Multiply the roll_rate output by 0.1 for the mixer, giving a roll request of 0.011512.
Static first-step example with zero initial conditions. Update the integral contribution afterward using forward Euler.

5.6.1 Control Algorithm Development and Testing: Output Calculation

One-Step Check of Four-Motor Output Changes

Mixer inputs: roll $R = 0.011512$, pitch $P = 0$, yaw $Y = 0$, throttle $T = 0.5$.

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \end{bmatrix} = 1000 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + 1000 \begin{bmatrix} T + P - R + Y \\ T - P + R + Y \\ T + P + R - Y \\ T - P - R - Y \end{bmatrix} \quad (\mu s)$$

Top View; Nose ↑; ΔPWM (μs)

M3 Front Left Clockwise +11.51	M1 Front Right Counterclockwise −11.51
M2 Rear Left Counterclockwise +11.51	M4 Rear Right Clockwise −11.51

$$\mathrm{PWM}_{1:4} = [1488.49,\ 1511.51,\ 1511.51,\ 1488.49]\ \mu s$$

Left ↑, Right ↓ → Roll Right

Add a 1000 μs bias to each element, then limit to 1000～2000 μs. Auxiliary outputs 5～8 and all disarmed outputs are 1000 μs.

Quad X numbering: M1 front right, M2 rear left, M3 front left, M4 rear right. Throttle 0.5 is for the channel check only.

5.6.1 Control Algorithm Development and Testing: Digital Closed Loop

Ground Truth, Estimated Feedback, and 3D Display

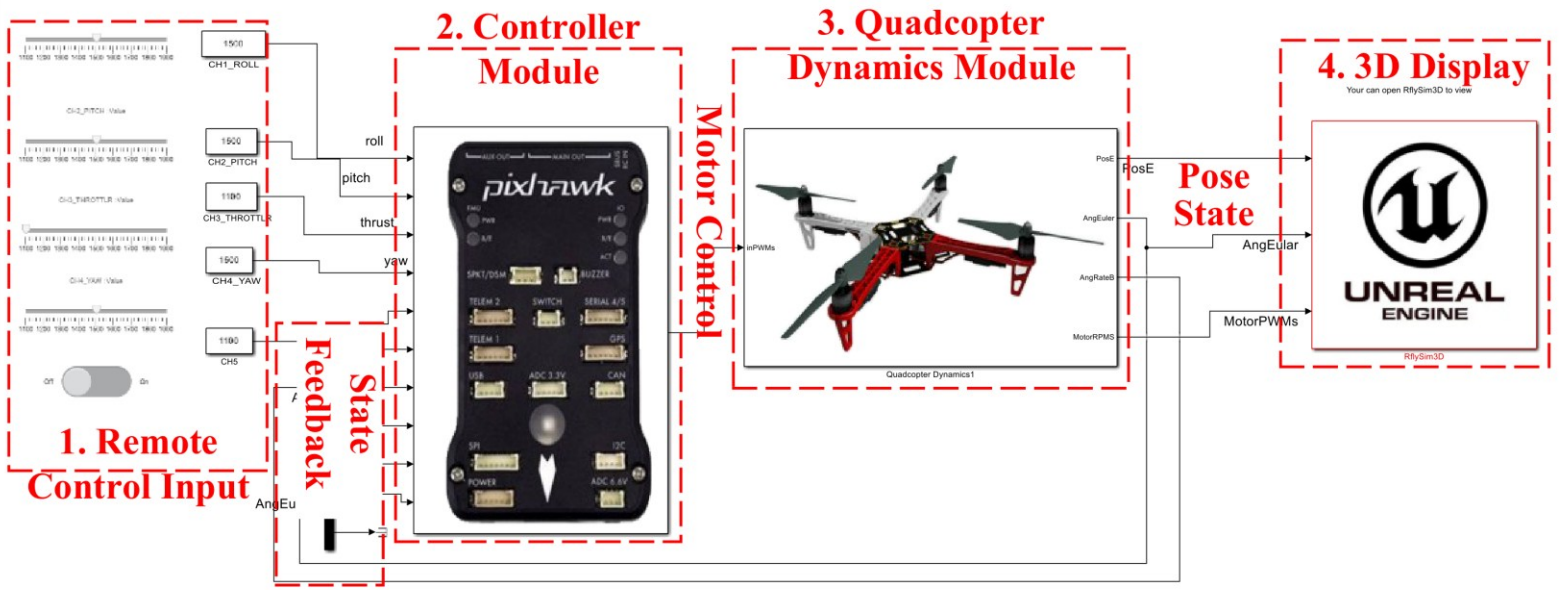


Fig. 5.8 Closed-Loop Simulation of the Quadcopter Controller and Dynamic Model

Ground-truth feedback isolates control; simulated sensors plus estimation test both estimation and control.

Autopilot estimates include interface effects and delays; 3D displays pose only. Pixhawk photo: icon.

5.6.1 Control Algorithm Development and Testing: Startup and Arming

Model Simulation: Startup, Then Arming

Entry: [安装目录]/RflySimAPIs/5.RflySimFlyCtrl/0.ApiExps/3.DesignExps/Readme.pdf

1. Save work → Run Init_control.m (clears workspace; loads parameters and icon path). Open Exp2_ControlSystemDemo.slx. Run only the controller and vehicle models.
2. Start disarmed: CH1/2/4=1500 μs, CH3=1100 μs, CH5=1100 μs. CH5<1500 gives ARM_Control=false. Save these inputs and parameters first.
3. Start RflySim3D → Click "运行" (Run) in the model. Check simulation time and states. Time must advance continuously, with attitude and angular-rate updates. Stop and restore the baseline if updates stop or become abnormal.
4. In simulation, set CH5=1600 μs → Check ARM_Control=true. Keep CH1/2/4 centered. Increase CH3 gradually only after arming, then establish a steady baseline.

Model loop only, with no physical autopilot or motors. Button labels depend on the current UI. Check actual CH5 and Boolean output.

5.6.1 Control Algorithm Development and Testing: Testing and Shutdown

Baseline Testing, Shutdown, and Restoration

Exp2 armed in model simulation: center CH1/2/4; save input and parameter baselines first.

1. Raise CH3 slowly; once airborne with near-zero vertical velocity and steady attitude, record CH3/PWM.
   Throttle baseline is vehicle-dependent; the static 0.5 checks channel logic only.
2. Change only CH1: 1500 → 1560 → 1500 μs. Keep CH2/4 at 1500 μs.
   Record targets, attitude, angular rates, 8-dimensional PWM, and timestamps on one time base.
3. Expected: correct direction, decay after target removal, bounded outputs, continuous time.
   Stop/restore for reversed direction, abnormal values, sustained saturation, or no steady baseline.
4. Normal shutdown: center attitude inputs → CH3 to 1100 → CH5=1100 to disarm.
   Confirm disarmed outputs → stop model → close 3D. Save data and input records.

Judge the same small command using its data. Use the 3D view for direction, and data to assess closed-loop quality.

5.6.1 Control Algorithm Development and Testing: Interpreting Results

Direction in 3D, Quality in Data from the Same Action

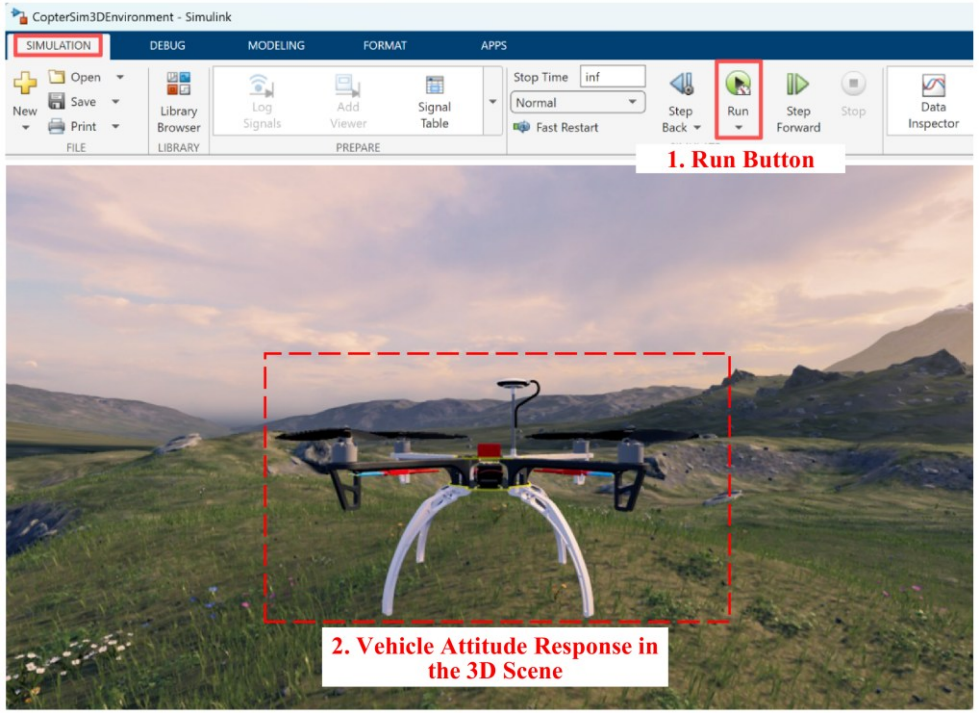


Fig. 5.9 Co-Simulation of Simulink and RflySim3D

Three Data Groups: Same Action

Target

Step time, amplitude, removal time.

State

Attitude, angular rates, and feedback source.

Output

Mixer requests, PWM, limit status.

3D: direction, ground contact, overturning.
Metrics: Deck 5.4, “Step-Response Metrics.”

Chapter Outline

Low-Level Control Development and Experimental Validation

Chapter 5
Filtering, State Estimation, and Low-Level Control

5.1 Filtering and Low-Level Control
5.2 Sensor Calibration and Estimation
5.3 Controller Design and Allocation
5.4 Control Performance and Tuning
5.5 PX4 Architecture and Interfaces
5.6 Control Development and Testing
5.7 Other Vehicles and Advanced Cases
5.8 Exercises and Classroom Discussion

5 Units in This Section

5.6.1 Control Algorithm Development and Testing
Textbook: Sec. 5.2.2.1

5.6.2 Position Control and Semi-Autonomous Modes
Textbook: Secs. 5.1.2.4, 5.3.1

5.6.3 Controller Development and Code Generation
Textbook: Sec. 5.2.2.2

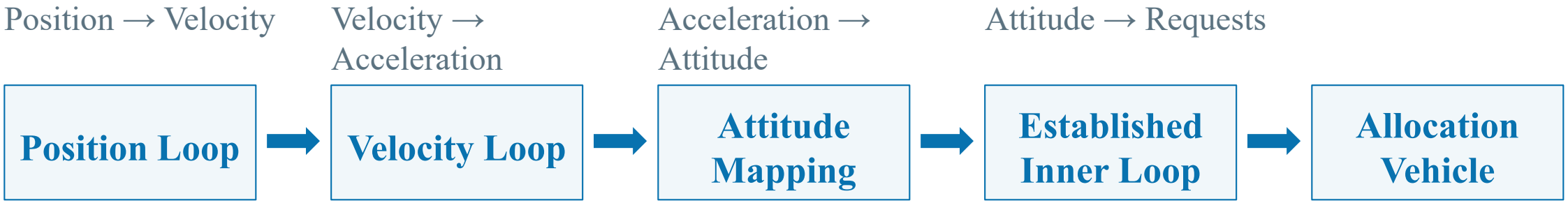
5.6.4 Software- and Hardware-in-the-Loop Simulation
Textbook: Sec. 5.2.2.2

5.6.5 Real-Vehicle Experiments and Result Review
Textbook: Secs. 5.2.2.3, 5.1.4

5.6.2 Position Control and Semi-Autonomous Modes: Position Outer Loop

Adding Position Outer Loops and Feedback

Prerequisites: a stable attitude inner loop and valid position and velocity feedback. Frames: north-east-down (NED) and body forward-right-down.



Add records of position, velocity, and desired attitude. Retain inner-loop and actuator data.

Installed example: 1.BasicExps/e6-PositionCtrl/e6.1/Sim

Init_control.m opens PosControl_Sim.slx at the end. This example loads its own vehicle parameters and inner-loop gains.

All relative entries are under [安装目录]/RflySimAPIs/5.RflySimFlyCtrl.

5.6.2 Position Control and Semi-Autonomous Modes: Position Model

Acceleration Limits and Tilt for a 1 m Step

Offline 1D case: 1 m target, initial position and velocity 0, ideal attitude inner loop. NED north axis, yaw 0, near hover. Hold acceleration constant within each sample interval.

$$a_k = \mathrm{clip}[k_x(1 - x_k) - k_v v_k, -1{,}1] \ \mathrm{m/s^2}$$

Position and Position Gain: x_k (m), $k_x = 2\ \mathrm{s^{-2}}$

Velocity and Velocity Damping: v_k (m/s), $k_v = 2\ \mathrm{s^{-1}}$

Acceleration and Sampling: a_k ($\mathrm{m/s^2}$), $\Delta t = 0.002$ s

clip limits the acceleration request to ± 1 m/s². Convert acceleration into the tilt target: $\theta_\mathrm{d} = -\arctan(a_k/g)$ (NED, yaw 0: northward acceleration requires nose-down, or negative pitch).

Continue the same recursion to 10 s, then compare the full results for position gains 2 and 6.

5.6.2 Position Control and Semi-Autonomous Modes: One Position Step

First-Step Acceleration, Position, Velocity, and Tilt

Continue the 1D model: substitute the 1 m target and zero initial conditions for the first 0.002 s sample.

$$a_0 = \text{clip}(2 \times 1 - 2 \times 0, -1, 1) = 1 \ \text{m/s}^2$$

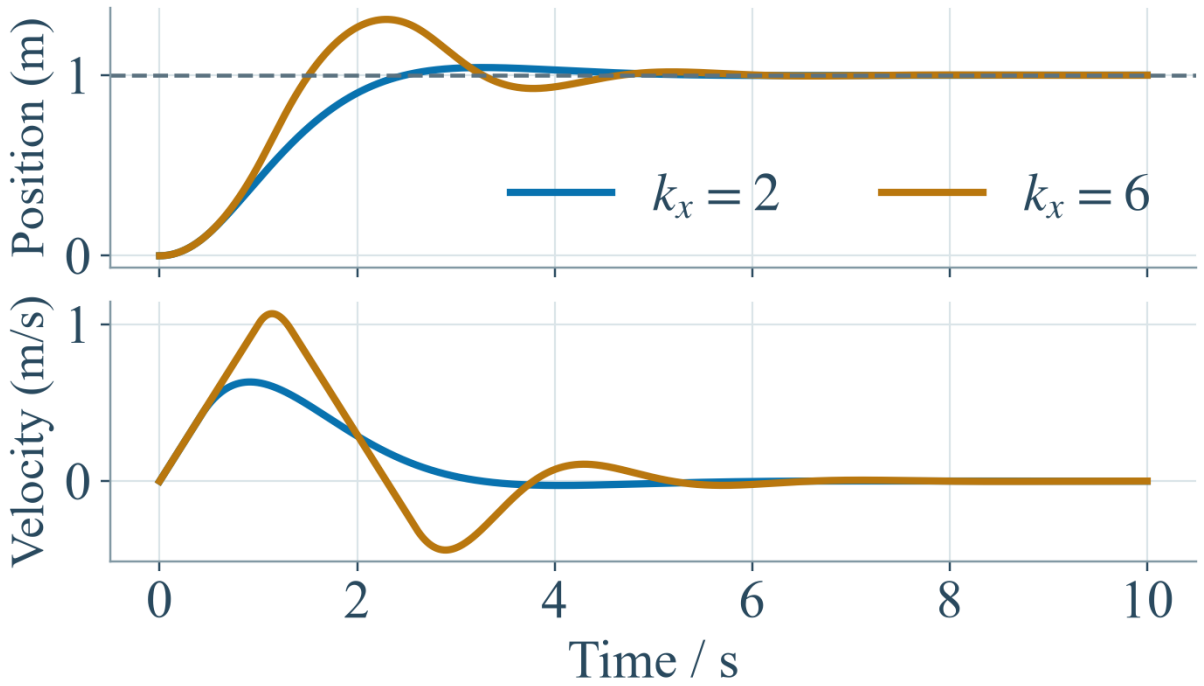
$$x_1 = 0 + 0 \times 0.002 + \frac{1}{2}(1)(0.002)^2 = 0.000002 \ \text{m}$$

$$v_1 = 0 + 1 \times 0.002 = 0.002 \ \text{m/s}, \quad \theta_\text{d} = -\arctan(1/9.81) = -5.8204^\circ$$

The first step advances only 0.000002 m. Repeat the updates to 10 s for the complete position response.

5.6.2 Position Control and Semi-Autonomous Modes: Full Position Results

An Aggressive Outer Loop Can Increase Overshoot and Saturation



Metric	Gain 2	Gain 6
Peak / m	1.0423	1.3088
Overshoot / %	4.23	30.88
2% Settling / s	4.326	4.450
Saturation / s	0.416	2.260
Final Value / m	1.000067	0.999904
Tilt / °	5.8204	5.8204

Figure: 1 m step (1D teaching calculation)

2% settling: stays within 0.98～1.02 m through 10 s. Saturation duration includes both limits. Tilt is the maximum absolute value.

5.6.2 Position Control and Semi-Autonomous Modes: Semi-Autonomous Modes

Stick Targets Depend on the Mode

CH6 uses a three-position switch for three modes; input targets and required feedback change.

Mode	Target Expressed by the Input	Holding and Feedback Requirements
Stabilize	Desired Roll/Pitch + Total Thrust	Valid Attitude and Angular Rates Centered Sticks Do Not Hold Position
Altitude Hold	Desired Attitude + Vertical Motion	Valid Altitude and Vertical Velocity Center Holds Altitude
Position Hold	Horizontal + Vertical Motion	Valid Position and Velocity Initialize Hold Target at Switching

Installed example: 1.BasicExps/e7-SemiAutoCtrl/Readme.pdf
e7.1/Sim/StabilizeControl_Sim.slx (simulation). e7.4/ModeSwitch_FLY.slx (real flight, CH6 switching).

5.6.2 Position Control and Semi-Autonomous Modes: Simulation Examples

Position Decoupling and Stabilize Mode Drift

Entries: 1.BasicExps/e6-PositionCtrl/e6.1/Readme.pdf; e7-SemiAutoCtrl/e7.1/Readme.pdf

1. e6.1/Sim: Run Init_control.m. PosControl_Sim.slx opens automatically.
   Open RflySim3D first, then click Run. Change ch1～ch4 during simulation.
2. ch1→y, ch2→x, ch3→altitude, ch4→yaw. Double-click Scope2 for position.
   Expected: changing only ch2 mainly changes x position, showing x/y channel decoupling.
3. e7.1/Sim: Run Init_control.m. StabilizeControl_Sim.slx opens automatically.
   Constant disturbance at the attitude output; desired roll/pitch=0. Click Run.
4. Expected: roll and pitch approach 0 while horizontal position and altitude drift. Stabilize holds attitude.
   Finish: stop simulation. Oscillation: first reduce PI gains, then check stability margins.

Both use Simulink only, with no autopilot. HIL directory: hardware-in-the-loop (HITL; HIL in the textbook). See “HITL Startup.”

5.6.2 Position Control and Semi-Autonomous Modes: Switching and Outer-Loop Diagnosis

Outer-Loop Diagnosis with an Inner-Loop Baseline

Observation	Compare Records	Interpretation and Action
Desired Tilt Stays at a Limit	Position/Velocity Targets Desired Attitude and Limits	Check Targets/Gains, Disturbances, and Capability Limits
Mild Command Vehicle Oscillates	Desired/Actual Attitude Angular Rates and Delay	Check inner-loop tracking/timing Keep the outer-loop baseline
Drift after Centering	Target Initialization, Velocity, Position Validity	Check Mode Meaning and States Pause Position-Hold Testing
Output Jumps at Switching	Switch Time, Integral Contribution, Control Requests and Outputs	Stop the Test Restore target/parameter baselines

Keep the inner loop and vehicle at baseline. Finish: baseline target → disarm → stop the model.

Chapter Outline

Low-Level Control Development and Experimental Validation

Chapter 5
Filtering, State Estimation, and Low-Level Control

5.1 Filtering and Low-Level Control
5.2 Sensor Calibration and Estimation
5.3 Controller Design and Allocation
5.4 Control Performance and Tuning
5.5 PX4 Architecture and Interfaces
5.6 Control Development and Testing
5.7 Other Vehicles and Advanced Cases
5.8 Exercises and Classroom Discussion

5 Units in This Section

5.6.1 Control Algorithm Development and Testing
Textbook: Sec. 5.2.2.1

5.6.2 Position Control and Semi-Autonomous Modes
Textbook: Secs. 5.1.2.4, 5.3.1

5.6.3 Controller Development and Code Generation
Textbook: Sec. 5.2.2.2

5.6.4 Software- and Hardware-in-the-Loop Simulation
Textbook: Sec. 5.2.2.2

5.6.5 Real-Vehicle Experiments and Result Review
Textbook: Secs. 5.2.2.3, 5.1.4

5.6.3 Controller Development and Code Generation: Model to Firmware

Separate Locations for Controller and Vehicle Model

Vehicle Dynamics → **Model / Dynamics Library (DLL)** → **Simulation Computer**

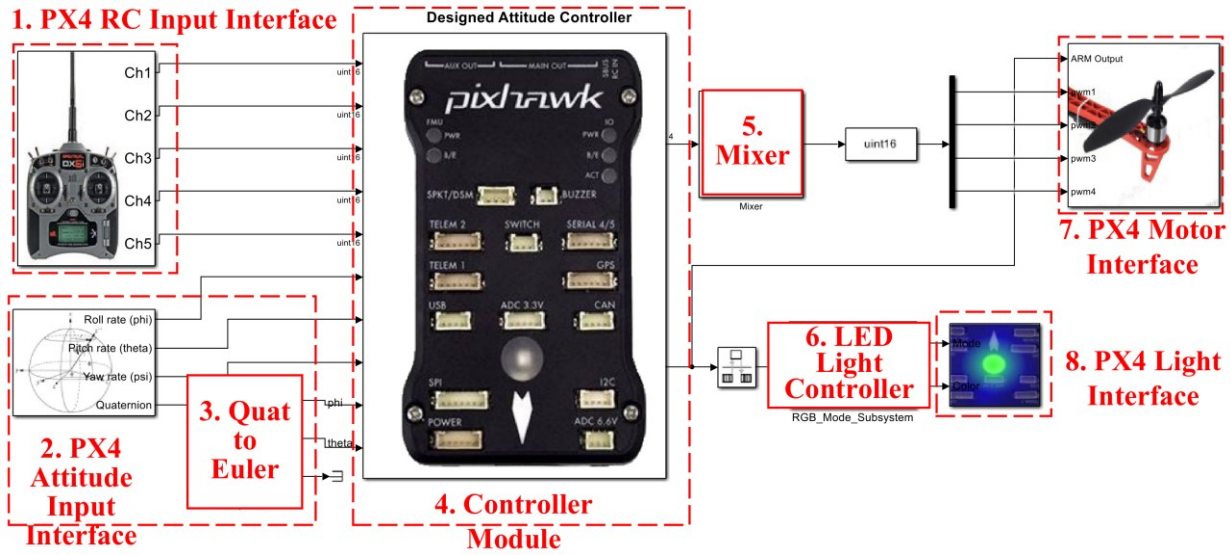
Simulate motion: actuator commands → position, attitude, velocity, and simulated observations. DLL: dynamic-link library.

Controller Subsystem → **Generate C → Build Target Program** → **SITL Process / Physical Autopilot**

Execute control: targets, estimated states → control requests. Run the target program in software-in-the-loop (SITL; SIL in the textbook) or on a physical autopilot.

5.6.3 Controller Development and Code Generation: Algorithm Porting Template

Controller Porting into the PSP Template



Blank template:
Exp3_BlankTemp.slx
Connected example in Fig. 5.10:
Exp4_AttitudeSystemCodeGen.slx
Port: copy the controller into ④ and connect ①～③ and ⑤⑦ by port
Check: dimensions, units, types, and step size 1/250 s

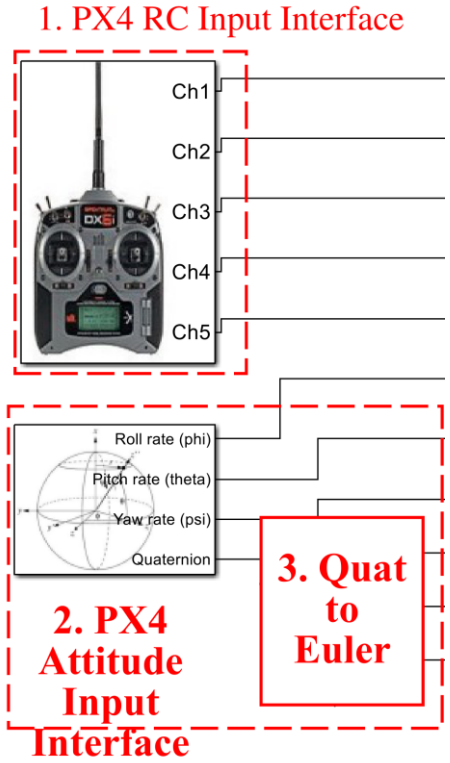
Fig. 5.10 Quadcopter HIL Framework

① Remote control; ② Angular rates/quaternion; ③ Quaternion to Euler angles; ④ Control; ⑤ Mixing; ⑥⑧ Lights; ⑦ Motors.

See “Template Interfaces” for details. Photos are interface-block icons. Physical hardware depends on the validation stage.

5.6.3 Controller Development and Code Generation: Template Interfaces

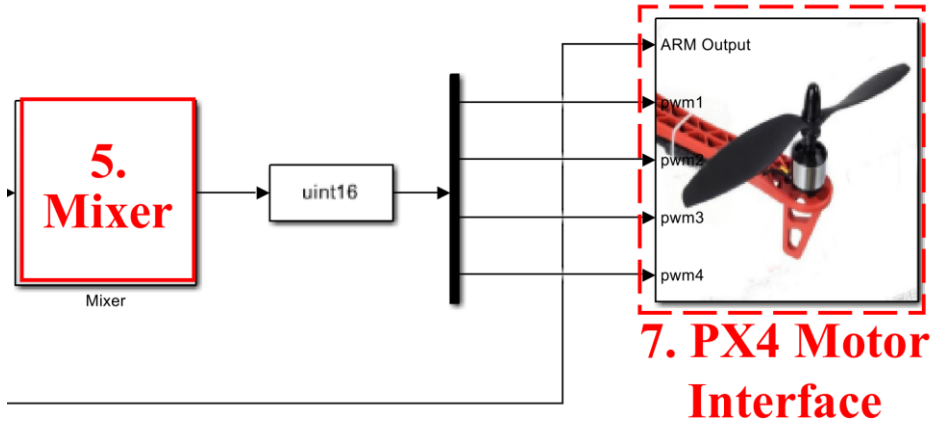
Remote-Control and Attitude Inputs, Motor Outputs



① CH1～CH5
uint16 Pulse Widths, μs

② Angular Rates and Quaternion
Roll/Pitch/Yaw rate:
p, q, r, in rad/s

③ Quaternion → Euler Angles
phi, theta, in rad



④→⑤: 4-D Control Requests Enter Mixer
uint16: Integer Pulse Widths to pwm1～pwm4
ARM Output: Arming Signal, Also Sent to Lights ⑥

Fig. 5.10 (Details): Left, Inputs ①～③; Right, Mixer ⑤ to Motor Interface ⑦

5.6.3 Controller Development and Code Generation: Code Generation Interface

Hardware Tab, Diagnostics, and Build Report

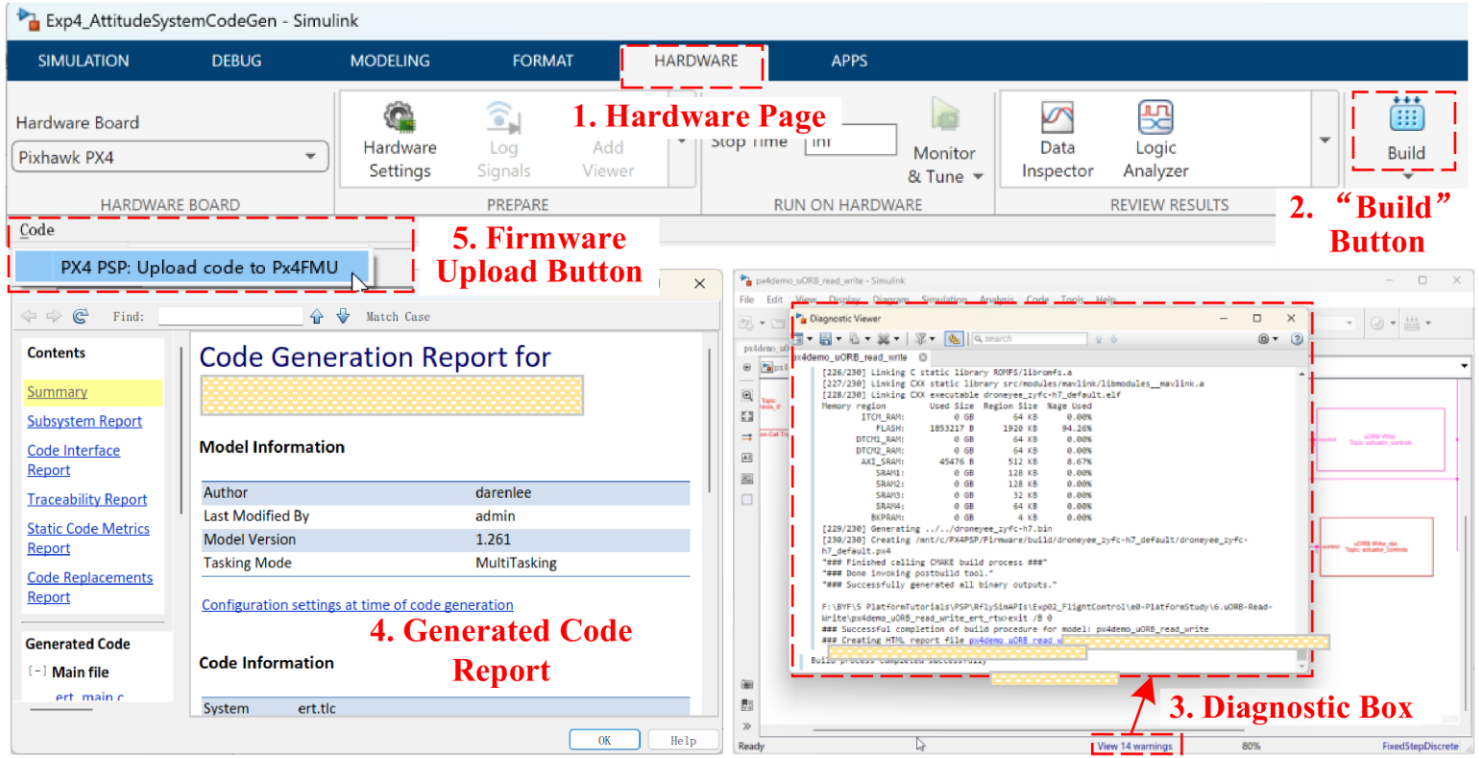


Fig. 5.11 Flowchart of Automatic Code Generation in Simulink

① Hardware Tab
② Build
③ Diagnostics
④ Generated Code Report
⑤ Upload Firmware

The report is a textbook example. Build and upload are separate actions. Use PX4Upload to upload.

5.6.3 Controller Development and Code Generation: Discrete Implementation

Faster Sampling, Smaller Integral Increments

I is the integral contribution to output, including Ki, rather than the raw error integral.
Constant error $e = 0.2$ rad, integral gain $K_i = 0.4\ \mathrm{s}^{-1}$, initial value $I_0 = 0$.

$$I_{k+1} = I_k + K_i e_k \Delta t$$

Frequency	Per-Step Substitution	Accumulation over the Same 1 s
250 Hz	$0.4 \times 0.2 \times 0.004 = 0.00032$ rad	$250 \times 0.00032 = 0.08$ rad
500 Hz	$0.4 \times 0.2 \times 0.002 = 0.00016$ rad	$500 \times 0.00016 = 0.08$ rad

Using 0.004 s at 500 Hz gives $I_{500} = 500 \times 0.00032 = 0.16$ rad, doubling the integral contribution.

Discrete PI follows the sample step, independently of continuous solver ode4. Also check sampling, delay, and computation budget.

5.6.3 Controller Development and Code Generation: Generation Configuration

Configuration for Computation, Timing, and Target

Configuration	Model Simulation Exp2	Generation Model Exp4
Base Step Size	0.001 s	1/250 s = 0.004 s
Hardware / Code Target	None / grt.tlc	Pixhawk PX4 / ert.tlc
Controller Implementation	Discrete PI, Forward Euler, Inherited Sampling	Discrete PI, Forward Euler, Inherited Sampling
Report and Solver	Report Off. Not a Firmware Entry.	Report On. Solver Field: ode4.

grt.tlc generates general simulation code. ert.tlc generates embedded target code. ode4 solves only continuous parts.

Example Readme: build command px4_fmu-v6x_default, recommended v1.14 series. Match the target to the board.

5.6.3 Controller Development and Code Generation: Build Evidence

Model, Target, and Warning Checks before Acceptance

Entry: Exp4_AttitudeSystemCodeGen.slx → Hardware → Build.

Stage Artifact	Evidence to Save	Where to Return on Failure
Model Check	Model Name, Initialization Parameters, Library Dependencies	Missing Variables or Library Blocks: Initialization/Dependencies
Code Generation	ert.tlc, Target Board, Code Report	Configuration Conflict: Generation Configuration
Target Build	First Error, Key Warnings, Resource Budget	Toolchain/Target: Match Version and Board
Binary Output	Firmware Path, SHA Hash, Version	No Artifact: Retain the Previous Baseline

Upload conditions: no terminating build errors, matching board, isolated actuators. See “HITL Startup” for upload steps.

Chapter Outline

Low-Level Control Development and Experimental Validation

Chapter 5
Filtering, State Estimation, and Low-Level Control

5.1 Filtering and Low-Level Control
5.2 Sensor Calibration and Estimation
5.3 Controller Design and Allocation
5.4 Control Performance and Tuning
5.5 PX4 Architecture and Interfaces
5.6 Control Development and Testing
5.7 Other Vehicles and Advanced Cases
5.8 Exercises and Classroom Discussion

5 Units in This Section

5.6.1 Control Algorithm Development and Testing
Textbook: Sec. 5.2.2.1

5.6.2 Position Control and Semi-Autonomous Modes
Textbook: Secs. 5.1.2.4, 5.3.1

5.6.3 Controller Development and Code Generation
Textbook: Sec. 5.2.2.2

5.6.4 Software- and Hardware-in-the-Loop Simulation
Textbook: Sec. 5.2.2.2

5.6.5 Real-Vehicle Experiments and Result Review
Textbook: Secs. 5.2.2.3, 5.1.4

5.6.4 Software- and Hardware-in-the-Loop Simulation: Validation Levels

Model, SITL, and HITL Validation Layers

Level	Control Program / Vehicle	Additional Evidence
Model Simulation	Simulink Algorithm / Computer Model	Equations, Directions, Limits, Connections
Software-in-the-Loop (SITL)	PX4 Program Built for Software / Computer Model	Processes, Message Loop, Firmware Closed Loop
Hardware-in-the-Loop (HITL)	PX4 Program on Physical Autopilot / Computer Model	Hardware Execution Time, Communication Timing
Real-Vehicle Experiment	Physical Autopilot Program / Physical Vehicle	Physical Sensors, Actuators, Environment

In this HITL setup, motor and aerodynamic behavior remain simulated. “SIL” in the 3.DesignExps example name refers to model simulation.

PX4 HITL Documentation (v1.14): https://docs.px4.io/v1.14/en/simulation/hitl

5.6.4 Software- and Hardware-in-the-Loop Simulation: SITL Startup

Build Control Code, Then Start Firmware SITL

Attitude SITL entry: 0.ApiExps/14.SITLVeriGenCodeFirm/2.AttitudeCtrlCodeGen/Readme.pdf

1. 3.DesignExps: Init_control.m → Exp5_AttitudeSystemCodeGen2.slx.
   Exp5 connects to a HITL-specific normalization block for simulation only. Hardware → Build.
2. MATLAB: PX4SitlSet replaces SITL firmware with a build containing this generated code.
   Wait for the build; save diagnostics. On failure, stop and exclude the run from the results.
3. [安装目录]/RflySimAPIs/SITLRun.bat → Enter 1 (1 simulated vehicle); wait for initialization.
   Enable the QGroundControl (QGC) virtual joystick. Arm in simulation; raise throttle slowly and record a steady baseline.
4. Apply a small attitude target from the baseline. Record target, states, requests, PWM, and timestamps.
   Stop/restore if abnormal. For exit and PX4SitlRec restoration, see “SITL Shutdown.”

Startup requires administrator privileges and terminates running CopterSim processes. This example requires the Full Edition.

5.6.4 Software- and Hardware-in-the-Loop Simulation: SITL Shutdown

Stop the Loop, Then Restore SITL

Normal Shutdown Order

1 Return to Baseline and Stop Recording

Center targets, reduce throttle to baseline, and disarm. Stop sending targets and collecting logs.

2 Exit Simulation Processes

Follow the SITL terminal prompts. Confirm firmware processes have exited and serial/network resources are free.

3 Restore the Original Environment

MATLAB: PX4SitlRec (Personal Edition and above). Confirm the baseline program works again.

Failure: stop requests → save first error/raw logs → exit → restore environment → restore original parameters.

The 5.01 exit script stops CopterSim, QGC and RflySim3D by process name, affecting parallel tasks.

5.6.4 Software- and Hardware-in-the-Loop Simulation: HITL Airframe

Airframe Selection Sets the Simulated Mode

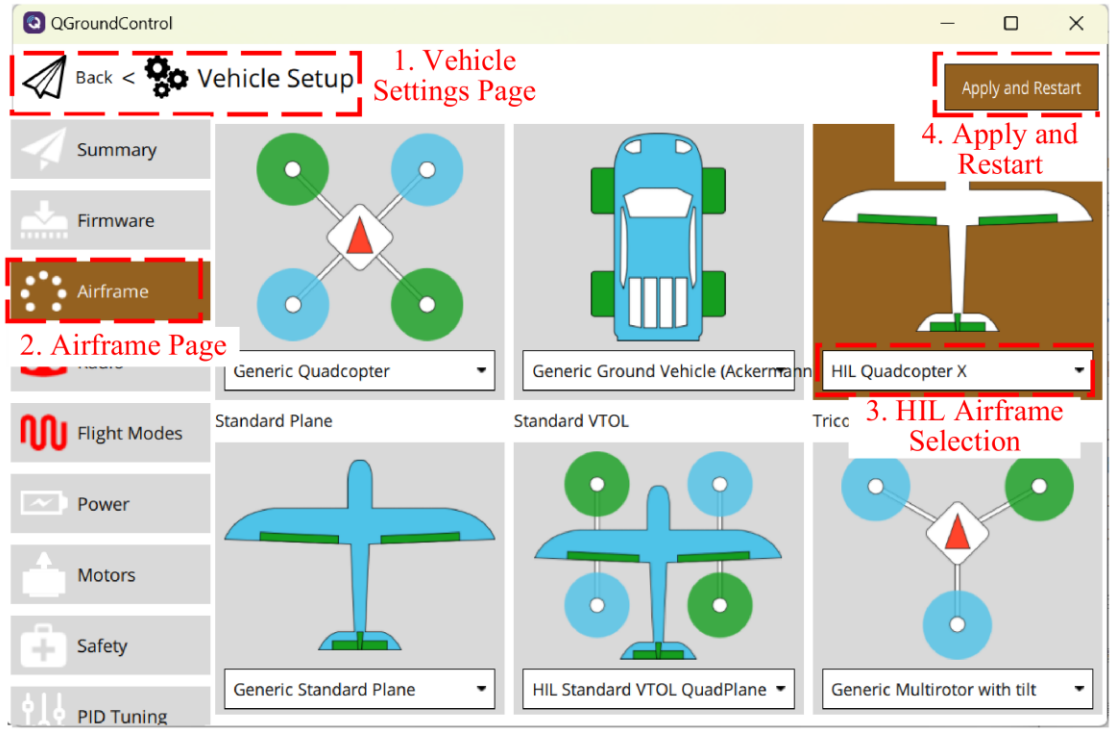


Fig. 5.12 Configuring a HIL Airframe in QGroundControl

① Vehicle Setup → ② Airframe

③ HIL Quadcopter X

④ Apply and Restart

v1.12.3: HIL Quadcopter X ID 1001 automatically sets SYS_HITL=1.

Physical autopilot + simulated motion, with physical actuators isolated. Select the matching real-vehicle airframe separately.

5.6.4 Software- and Hardware-in-the-Loop Simulation: HITL Startup

Upload, Confirm the Airframe, Then Start HITL

0.ApiExps/3.DesignExps/Readme.pdf (Sec. 4.3) → Exp4_AttitudeSystemCodeGen.slx

1. Successful build → Connect the autopilot by USB → Run PX4Upload in MATLAB.
   Wait for programming and Verify to pass, then check the new firmware after restart. Alternative: PX4 PSP: Upload code to Px4FMU.
2. In QGC, select HIL Quadcopter X (see “HITL Airframe”).
   In Flight Modes, set Emergency Kill switch=CH5; calibrate remote control as prompted.
3. Reconnect autopilot; run [安装目录]/RflySimAPIs/HITLRun.bat; enter serial port number.
   Wait for CopterSim (EKF: extended Kalman filter): GPS 3D fixed & EKF initialization finished.
4. Arm with CH5: LED on; throttle gives visible takeoff. Follow “HITL Acceptance.”
   Finish: center inputs, disarm → press any key in HITLRun; all three applications close.

Firmware is replaced; first remove propellers/check board. Select the real airframe in QGC before real flight.

No COM port: try another data cable/USB port. No init: check port number, exit QGC, rerun.

5.6.4 Software- and Hardware-in-the-Loop Simulation: HITL Acceptance

Evidence for Connections, States, Directions, and Timing

Continue from “HITL Startup.” After CopterSim reports EKF initialization complete, check the four stages in order.

Preparation	States	Limited Test	Shutdown and Restoration
Action: check HITL airframe and parameters, isolate actuators **Pass:** no port conflicts, matching configuration	**Action:** wait for EKF initialization **Pass:** continuous timestamps. Stop if updates stop.	**Action:** small attitude target. Check CH5 and channel directions. **Pass:** all directions correct. Stop for saturation or timeout.	**Action:** center, disarm, press a key to exit HITLRun **Pass:** ports released, logs saved, baseline restored

An LED, serial connection, or 3D view proves only a local state. Also check direction, timing, and closed-loop response.

5.6.4 Software- and Hardware-in-the-Loop Simulation: Consistency across Stages

Align Tasks, State Sources, and Clocks

Fixed Conditions	Synchronized Records	Quantities to Compare
Same Target and Initial Conditions	Target, Attitude, Angular Rates	Error, Peak, Settling Time
Same Parameters and Constraints	Control Requests, Outputs, Limits	Saturation Duration, Allocation Residual
Explicit State Sources	Ground Truth/Simulated Observations/Estimated States	Estimation Bias, Validity Flags
Aligned Time Base	Sample/Receive Times, Execution Time	Update Intervals, Age, Timeouts

Observation age uses one time base; see Deck 5.5, “Control Period and Time Budget.”

$$\Delta t_{\text{age}} = t_{\text{recv}} - t_{\text{sample}} = 1.020 - 1.000 = 0.020\ \text{s} = 20\ \text{ms}$$

Observation Age Differs from End-to-End Control Delay.

Chapter Outline

Low-Level Control Development and Experimental Validation

Chapter 5
Filtering, State Estimation, and Low-Level Control

5.1 Filtering and Low-Level Control
5.2 Sensor Calibration and Estimation
5.3 Controller Design and Allocation
5.4 Control Performance and Tuning
5.5 PX4 Architecture and Interfaces
5.6 Control Development and Testing
5.7 Other Vehicles and Advanced Cases
5.8 Exercises and Classroom Discussion

5 Units in This Section

5.6.1 Control Algorithm Development and Testing
Textbook: Sec. 5.2.2.1

5.6.2 Position Control and Semi-Autonomous Modes
Textbook: Secs. 5.1.2.4, 5.3.1

5.6.3 Controller Development and Code Generation
Textbook: Sec. 5.2.2.2

5.6.4 Software- and Hardware-in-the-Loop Simulation
Textbook: Sec. 5.2.2.2

5.6.5 Real-Vehicle Experiments and Result Review
Textbook: Secs. 5.2.2.3, 5.1.4

5.6.5 Real-Vehicle Experiments and Result Review: Real-Vehicle Prerequisites

Physical Differences Still to Check

Item	Confirm before Testing	Pass or Stop Conditions
Vehicle and Sensors	Airframe, Inertia, Directions, Calibration	Valid States Recheck after Changes
Actuator Chain	Channel Order, Range, Response Direction	Isolation/Bench Conditions Met Stop on Abnormal Behavior
Task-Required States	Attitude or Position, Velocity, and Quality Flags	Stop the Affected Task on Failure
Test Level	**Offline** **Bench** **Limited** **Real Flight**	Responsible Person, Site, Takeover and Termination Procedures

Establish an official-firmware baseline before real-vehicle tests. Recheck airframe and parameters when leaving HITL.

5.6.5 Real-Vehicle Experiments and Result Review: Control Source Switching

Targets, Internal States, and Outputs at Takeover

Use official PX4 for safe real-vehicle fallback; keep total output continuous across switching.

Before Takeover	At Switching	After Fallback
Valid states, aligned targets Prealign the Incoming Integral Contribution Total Output Jump ≈0	CH8≥1500: mode 0→2 Record Control Source and Output Calculate the Output Jump	QGC reports return to PX4 Check Feedback and Targets Save failure/recovery records

Slide section 5.5: “Control Source Switching Example” (v1.13) gives steps. See “Output Authority and Arbitration” for masking.

RePX4Block: mode=0 is official PX4; mode=2 masks only PX4 torque/thrust.

$u_{I,0} = u_{\mathrm{PX4}} - u_P$: preset the integral contribution for output continuity (see “Control Source Switching Calculation” in slide section 5.5).

5.6.5 Real-Vehicle Experiments and Result Review: Switching Rehearsal

Control Source Switching Rehearsal in HITL

Entry: 0.ApiExps/21.SwitchPX4Simulink/Readme.pdf (requires v1.13 firmware).

1. Prepare: build, rebuild firmware and upload per Deck 5.5, “Control Source Switching Example.” Start HITLRun; await EKF initialization. Confirm airframe and CH5 kill switch in QGC.
2. Baseline: arm/take off in QGC on official PX4 (mode=0); establish steady flight. Synchronize recording of control source, control requests, actual outputs and timestamps.
3. Switch: CH8≥1500 selects Simulink, mode=2. Switch back to PX4, mode=0. At every switch, confirm that QGC messages match the recorded control source.
4. One time base: measure total output jump at switching and post-fallback response. If the jump exceeds the preset threshold or PX4 fails to take over, diagnose in HITL before real-vehicle testing.

For switching problems, use “Control Source Switching Example” in slide section 5.5. Finish by disarming and pressing a key to exit HITLRun.

5.6.5 Real-Vehicle Experiments and Result Review: Log Review

Tracing Targets, States, Requests, and Outputs

Target	State	Control Request	Actuator Command
Check: value, units, time, control source **Causes:** wrong task; target not initialized at switching	**Check:** validity, innovation, timestamps **Causes:** estimation failure, delay, inconsistent frames	**Check:** error, integral contribution, limits **Causes:** aggressive parameters; integral buildup; conflicting publishers	**Check:** channels, direction, amplitude, response **Causes:** allocation, actuator or vehicle differences

Hypothetical: request spike + invalid state flag → check states/timing first → repeat the same task.

Collect Filter Data (e4.1; Entry in 5.2)

Upload log_data.slx: Replaces Autopilot Firmware

CH5 Up Starts, Down Stops

e4_A.bin → Attitude_estimator0.m

Control Logs (5.Log-Write-Read)

px4demo_log.slx → .bin

px4demo_log_qgc.slx → .ulg

PX4ReadLog Imports into MATLAB

5.6.5 Real-Vehicle Experiments and Result Review: Result Records

Evidence for Each Completed Stage

Completed example: offline northward 1 m case, with an ideal attitude inner loop.

Record Item	Baseline or Result	Change and Restoration
Model and Input	Step 0.002 s; 10 s; Initial Values 0	Acceleration Limit ±1 m/s² Unchanged
Parameters	Position Gain 2; Velocity Damping 2	Position Gain 6; Velocity Damping 2
Full Results	Peak 1.0423 m; Overshoot 4.23%	Peak 1.3088 m; Overshoot 30.88%
Saturation/End	0.416 s; Final 1.000067 m	2.260 s; Final 0.999904 m
Evidence Scope	Independent Low-Dimensional Offline Calculation	Platform Tests Require Separate Raw Logs

Restore gain 2; keep both datasets. Platform data: versions and model/parameter hashes.

Class Discussion

HITL Oscillation: What to Check First

Scenario: a small roll target works in the model but oscillates in HITL. Find evidence for each check.

Suggested Check	Available Evidence	First Action
Sampling and Delay	Target Updates Every 4 ms Feedback Updates Every 20 ms	Check Periods and Data Freshness
Frames and Units	Model Uses rad Hardware Input Not Converted to rad	Check Unit Conversion and Axis Directions
Outputs and Authority	Reversed PWM Direction Two Modules Publish Simultaneously	Check Mixing, Channels, and Publisher Ownership
Computation Budget **Model Differences**	Execution Timeout Mismatched Actuator Inertia	Check Execution Time/Vehicle Baseline Then Tune

Criteria: data and stop conditions for every check. Restore the baseline first, then vary one factor at a time.

Class Exercises

New Inputs for the Control Chain and Integral Step

Exercise 1 Remote-Control Pulse Width to Motor PWM

Given: CH1=1420 μs, roll −0.02 rad, p=0, initial integral contribution 0; T=0.5, P=Y=0. Use Exp1 gains. Find ϕ_d, u_{att}, rate-loop output, and four motor PWM values. Determine roll direction.

Exercise 2 Integral Step after a Sample-Rate Change

Change the teaching discrete PI from 250 Hz to 400 Hz. Constant error 0.1 rad, integral gain 0.5 s^{-1}, initial integral contribution 0.

If the code still uses 0.004 s, what is the integral contribution after 1 s? What is the correct value?

Exercise 3 Evaluating Evidence

In HITL, the LED is lit and QGC reports armed. Does this establish correct control direction?

If not, what further evidence is needed?

Response Guidance

Show substituted values and units at each step; justify roll direction by which side's motor outputs increase.

Reference Answers

Requests, Motor Changes, and the Integral Step

Exercise 1

$\phi_d = 0.5 \times (1420 - 1500)/400 = -0.1$ rad (use the original value outside the deadband)
Attitude loop: $u_{\text{att}} = 0.525116 \times (-0.1 + 0.02) = -0.042009$
$u_{\text{rate}} = 1.096115 \times (-0.042009 - 0/3) = -0.046047$, $\times 0.1$ gives $R = -0.004605$
$\text{PWM}_{1:4} = [1504.60,\ 1495.40,\ 1495.40,\ 1504.60]\ \mu s$: right increases, left decreases → roll left.

Exercise 2

Correct step 1/400=0.0025 s: 0.5×0.1×0.0025=0.000125 rad per step. 400 steps give 0.05 rad.
Wrong step 0.004 s: 0.0002 rad per step, giving 0.08 rad in 1 s, or 1.6 times the correct value.

Exercise 3

No. The LED and armed status only show that outputs are enabled. Apply a small CH1 step and check roll direction and PWM changes.

Section Summary

One Task through Development, Validation, and Restoration

Keep the Same Task

Keep targets, parameters, initial conditions, and constraints comparable. Save evidence at each stage: template→build→SITL→HITL.

Attitude → Position Extension

Retain the inner loop. The outer loop adds states, limits, and target initialization at switching.

Normal Shutdown and Failure Recovery

In simulation, stop the loop first. For real flight, follow takeover procedures. Land and disarm before exit and restoration.

Common Errors

Model simulation≠firmware SITL; ground truth≠estimates; compilation≠closed-loop validation; HITL airframe≠real airframe. Exp1: 8-dimensional PWM; Deck 5.5 Exp1_HIL16CtrlsPWM: 16-dimensional PWM. At switching, mode=0 selects the official controller; mode=2 masks only torque/thrust. Use each example's Readme for its version.
Deck 5.7: revalidate affected parts the same way after vehicle-type or algorithm changes.