Chapter Outline

From Finding Examples to Experimental Results

Chapter 2
Experimental Platform Configuration

2.1 Key Tools

2.2 Installation and First Simulation

2.3 Architecture and Workflows

2.4 API and SDK Interfaces

2.5 Examples and Experimental Routes

2.6 Exercises and Discussion

3 Units in This Section

2.5.1 Finding Examples and Choosing Tasks

Textbook: Sec. 2.2.4

2.5.2 Reproducing and Interpreting Representative Tasks

Textbook: Sec. 2.4

2.5.3 Example Index and Prerequisites for Advanced Work

Textbook: Sec. 2.4

2.5.1 Finding Examples and Choosing Tasks: Lesson Task

Complete an Experiment with Inputs and Outputs

- **Find the files:** instructions, programs or models, configurations, and data; the launcher groups them by task.
- **Match prerequisites:** offline analysis needs an existing log; simulation, builds, and deployment need task-specific software or equipment.
- **Keep a baseline:** run the original setup; save inputs and results; change one item.
- **Explain the outputs:** recorded object, time, physical quantities; explain change effects.

PX4 flight-controller logs (ULog) → topic CSV (comma-separated values) files (2.5.2)
https://github.com/PX4/pyulog

Local file:
[InstallationDirectory]\RflySimAPIs\2.RflySimUsage\[GroupFolder]\[ExampleFolder]\Readme.pdf

Reproducing an example yields an explainable, repeatable input-output relationship.

2.5 Examples and Experimental Routes

2.5.1 Finding Examples and Choosing Tasks: Finding Tasks

Find Chapter 2 Tasks in the Example Gallery

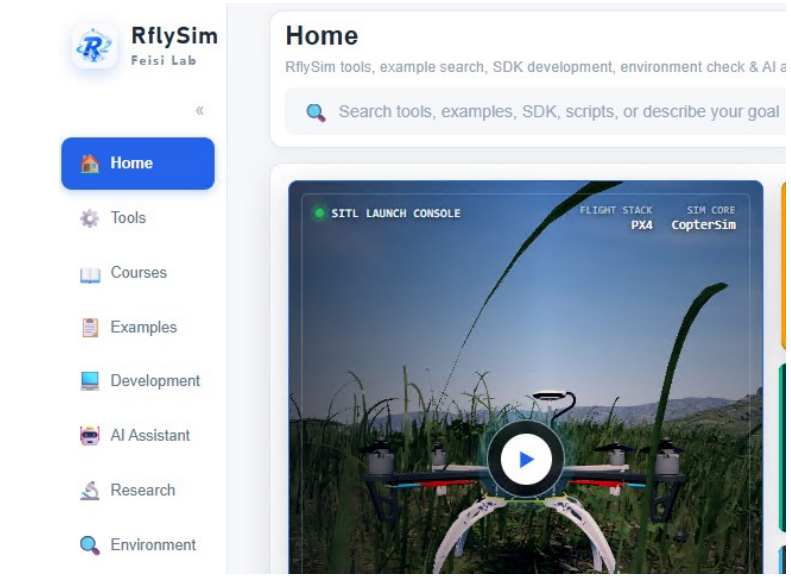


Figure: Example Gallery (left menu)

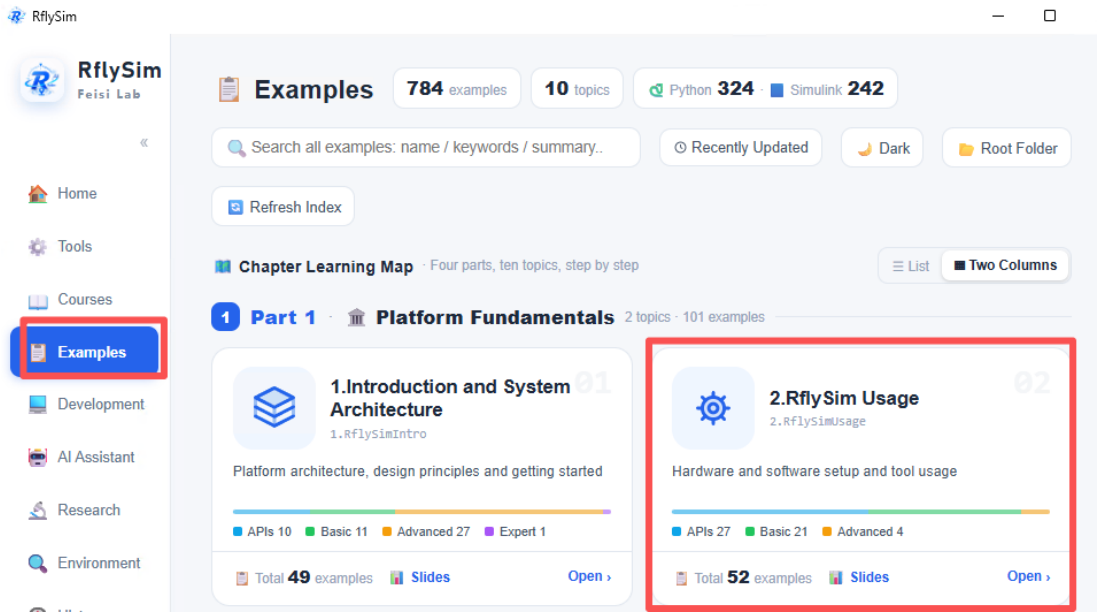


Figure: Chapter 2 entry and folder name

- **Open this chapter:** Example Gallery → "2. Experimental Platform Configuration" (folder: 2.RflySimUsage).
- **Check requirements:** open a specific example's Readme.pdf and check its software, hardware, and expected outputs.

2.5.1 Finding Examples and Choosing Tasks: Using the Documentation

Use Chapter Documents with Each Example's Instructions

Document	Contents and Purpose
PPT.pdf / Intro.pdf	PPT.pdf: chapter slides; Intro.pdf: components, interfaces, and development workflows
Index.pdf	All chapter examples, with brief descriptions and links
API.pdf	Interface entry points, parameters, and data formats; no operating steps
Each Example's Readme.pdf	Objectives, requirements, key concepts, detailed steps, and expected results

Textbook: Sec. 2.4: PPT.pdf → Intro.pdf → Index.pdf + API.pdf → the example's Readme.pdf

Chapter documents provide context; each Readme connects the task to its files.

2.5.1 Finding Examples and Choosing Tasks: Choosing by Prerequisites

Three Groups by Purpose; Advanced Work Builds on the First Two

Group	Typical Tasks	Prerequisites and Results
Interface Learning 0.ApiExps	Read docs, set up environments, receive messages	Install task dependencies; obtain a working interface or environment
Basic Usage 1.BasicExps	Configurations, logs, models, and piloting	Run a specific function; obtain configurations, data, or responses
Advanced Development 2.AdvExps	Image access, SDK use, and tool development	Meet development prerequisites; create an interface or application

Install dependencies for the task; you need not install every development environment first.

2.5.1 Finding Examples and Choosing Tasks: Reproduction Steps

Save a Baseline, Then Change One Condition

- **Reproduce unchanged:** read requirements; keep programs, models, and parameters fixed for the first output.
- **Save the baseline:** record example, version, object ID, settings; keep original result files.
- **Change one input:** predict its effect, run, and check the corresponding output.
- **Explain the results:** use time, physical quantities, or image changes to explain differences; submit working settings and results.

Camera exercise (see 2.5.2)

Fix resolution, pose, and target → **Change only the field of view** → Compare image coverage

Changing one item with the rest fixed helps explain output changes.

Chapter Outline

Reproducing and Interpreting Representative Tasks

Chapter 2
Experimental Platform Configuration

2.1 Key Tools

2.2 Installation and First Simulation

2.3 Architecture and Workflows

2.4 API and SDK Interfaces

2.5 Examples and Experimental Routes

2.6 Exercises and Discussion

3 Units in This Section

2.5.1 Finding Examples and Choosing Tasks

Textbook: Sec. 2.2.4

2.5.2 Reproducing and Interpreting Representative Tasks

Textbook: Sec. 2.4

2.5.3 Example Index and Prerequisites for Advanced Work

Textbook: Sec. 2.4

2.5.2 Reproducing and Interpreting Representative Tasks: Task Overview

Five Representative Tasks: Inputs, Outputs, and Criteria

Task	Input	Output	Interpretation Criteria
Offline Logs	Supplied ULog	CSV Files by Topic	Align position and mode by time
Configuration Reuse	Vehicle and experiment settings	Saved experiment	Same conditions after restart
Motion and Startup	Startup scripts and actions	Trajectory and state records	Match actions to records
Propulsion and Endurance	Mass and propulsion parameters	Hover and endurance predictions	Hover time meets the target
Scenes and Cameras	Scene, vehicle, and camera	Scene views and camera images	IDs and images match setting changes

Choose among five tasks; preserve conditions, reproduce, record, and explain differences in each.

2.5.2 Reproducing and Interpreting Representative Tasks: Offline Logs

Convert the Supplied Offline ULog to Topic CSV Files

e4_Log-Reads-Python38Env: open the Python38Env terminal in RflyTools, then convert the log.

Supplied log.ulg → **ulog2csv log.ulg** → **log_topic_instance.csv ×32**

Sample Output Files (log_topic_instance)	Contents
log_vehicle_local_position_0.csv	North-east-down position x, y, z and velocity vx, vy, vz
log_battery_status_0.csv	Battery voltage, current, and other records
log_actuator_outputs_0.csv	Actuator output records

2.5.2 Reproducing and Interpreting Representative Tasks: Offline Logs

Read One Topic: Altitude During Takeoff, Hold, and Return

- Time: $t = \text{timestamp}/10^6$, in seconds; samples are about 0.1 s apart here, with variable intervals.
- Position: x, y, z use north-east-down (NED) coordinates in m; z is positive downward, so negative z is above the starting point.
- Mode: nav_state in vehicle_status; in this log, 17 = Auto Takeoff, 4 = Hold, and 5 = Return.

nav_state and Phase (Start Time)	Changes in z and Key Times
17 Auto Takeoff (from 271.6 s)	Liftoff at 273.2 s; z falls from 0 to about −2.5 m
4 Hold (from 278.1 s)	z remains near −2.5 m for about 76 s
5 Return (from 354.6 s)	z reaches about −9.3 m at 362.9 s; landing at 377.4 s

Align position and flight mode by time to explain a flight from its logs.

2.5.2 Reproducing and Interpreting Representative Tasks: Offline Logs

Supplied Log: Align Vertical Position z with nav_state

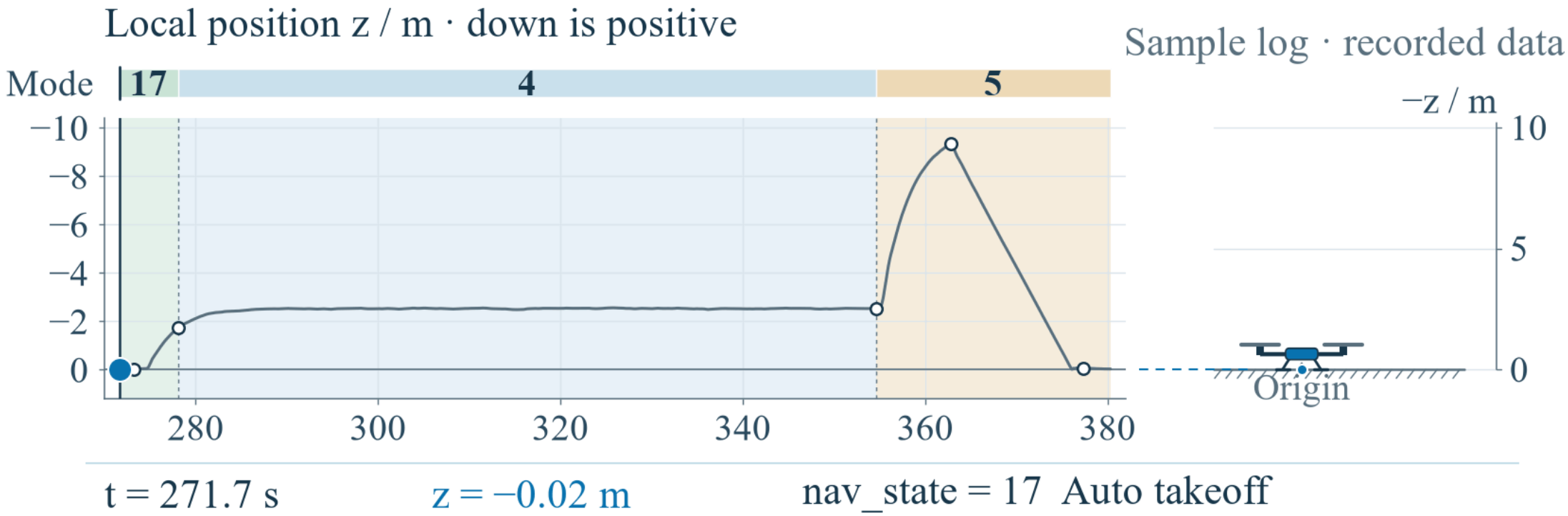


Figure: Supplied log (collection environment unverified); z and nav_state aligned by time

Mode identifies the task phase; position shows the motion response. Interpret both on the same timeline.

2.5.2 Reproducing and Interpreting Representative Tasks: Data Sources

A Log's Source Determines Which Questions It Can Answer

Comparison	Autopilot ULog	CopterSim CSV (e13_Log-Get)
Recorded Data	Estimated states, sensor readings, and control outputs	Ground truth: position, velocity, attitude, and motor speeds
Check First	Topics, fields, and units	Vehicle ID, headers, and units
Time Column	$\text{timestamp}/10^6$ (μs → s)	runnedTime(s) (seconds)

1	0:runnedTi	22:MotorR	23:MotorR	24:MotorR	25:MotorR	26:MotorR	27:MotorR	28:MotorR	29:MotorR	3
6313	126.95	5232.76	5230.93	5232.54	5236.24	0	0	0	0	
6314	126.965	5232.45	5227.9	5234.57	5236.25	0	0	0	0	
6315	126.99	5227.91	5228.46	5239.45	5235.15	0	0	0	0	
6316	127.01	5226.55	5231.23	5240.88	5233.5	0	0	0	0	
6317	127.025	5226.3	5234.42	5239.85	5233.43	0	0	0	0	

Figure: CopterSim CSV time (column 0) and motor speeds (22–25); align both log types by a shared event

ULog records flight-controller internals; CopterSim CSV records simulation ground truth.

2.5.2 Reproducing and Interpreting Representative Tasks: Saving Configurations

Save an Experiment in SimCreate for Reuse

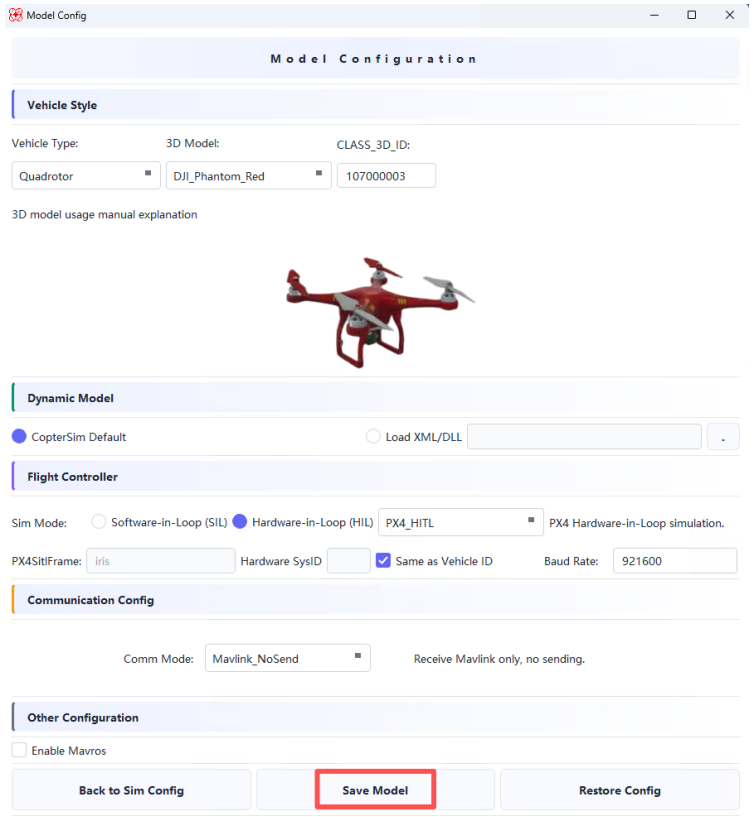


Figure: Vehicle settings

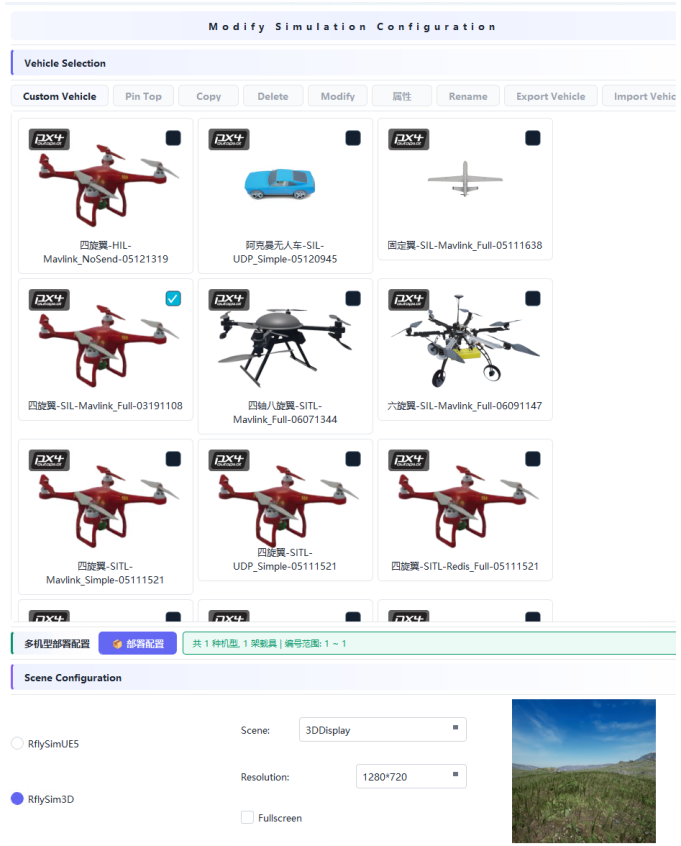


Figure: Experiment settings

- **Vehicle:** select appearance, motion model, and simulation mode; click Save Vehicle.
- **Experiment:** select vehicles; set count, IDs, initial states, and scene; click Save Experiment and name it.
- **Reuse:** click Start Experiment, then check that the conditions are restored.

2.5.2 Reproducing and Interpreting Representative Tasks: Reusing Configurations

Change Only Initial Heading to Check Configuration Reuse

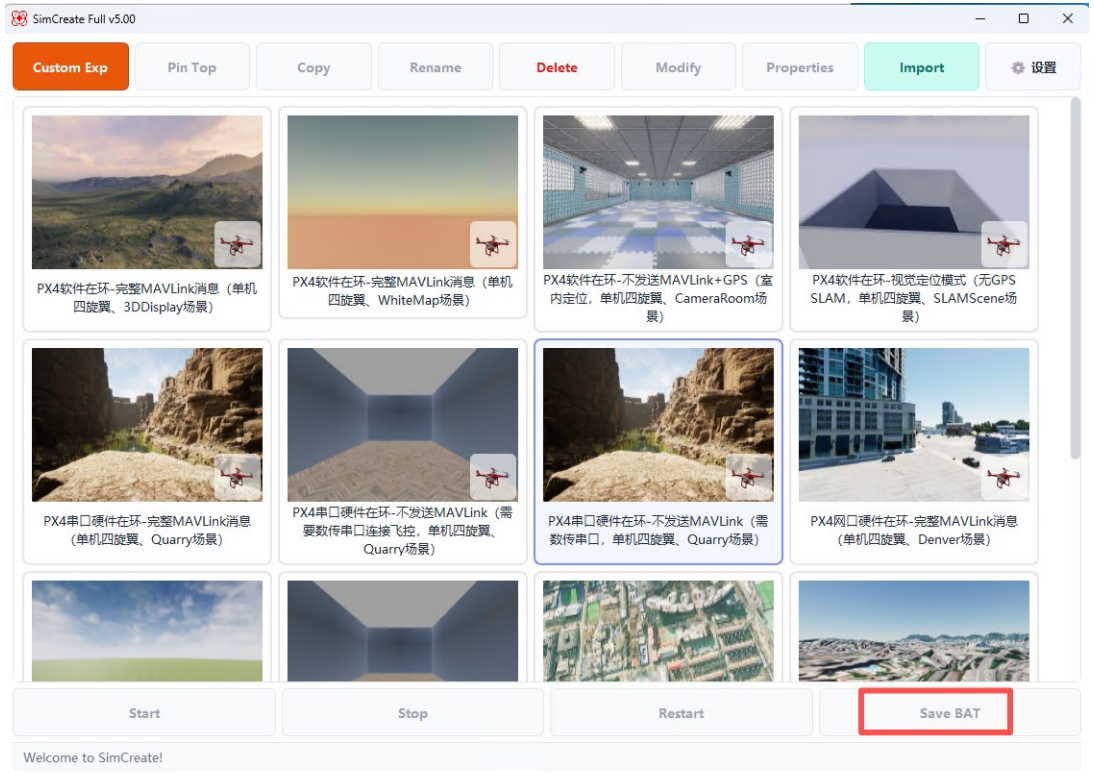


Figure: SimCreate main screen; red box: Save as BAT

- **Configuration A:** save a single-vehicle experiment and use Save as BAT; run it and record the baseline heading.
- **Configuration B:** copy A, change only initial heading, and save under another name.
- **Reuse criterion:** heading changes as set; ID, vehicle type, and scene stay the same.

2.5.2 Reproducing and Interpreting Representative Tasks: Preparing to Record

Match Filenames to Vehicle IDs First

Step	Action	Check
Prepare	Copy CopterSim1–3.csv to [InstallationDirectory]\CopterSim	The filename number is the vehicle ID; the extension must be .csv
Start	Software-in-the-loop (SITL; SIL in the textbook): RflyTools → SITLRun; enter 3	IDs 1–3 appear in the CopterSim display
Record	After simulation starts, select an ID in QGC; take off, fly forward, and land	Record the time of each action

Read e13_Log-Get: stop simulation in CMD and close programs, then open the CSV in Excel.
Check: file size grows; position, velocity, and motor speeds change with actions.

The ID in a CopterSim filename links the data to a specific simulation instance.

2.5.2 Reproducing and Interpreting Representative Tasks: Motion and Records

Explain a Motion Task Using Time and Response

Action	Columns and Expected Changes
Throughout	Column 0, runnedTime (s): subtract actual times in adjacent rows; column 1, copterID: identify the vehicle
Takeoff	Column 5, PosE[2]: NED z is positive downward; takeoff changes it from about 0 to negative values
Forward Flight	Columns 3–4, PosE[0/1], and 6–8, VelE, change continuously; which column changes depends on heading
Hover	Columns 22–25: motor speeds are nearly steady (about 5230 rpm in the data-source screenshot)

Motion records become meaningful when targets, actions, times, and states are linked.

2.5.2 Reproducing and Interpreting Representative Tasks: Organizing Startup

Start the Same Experiment Manually or with a Script

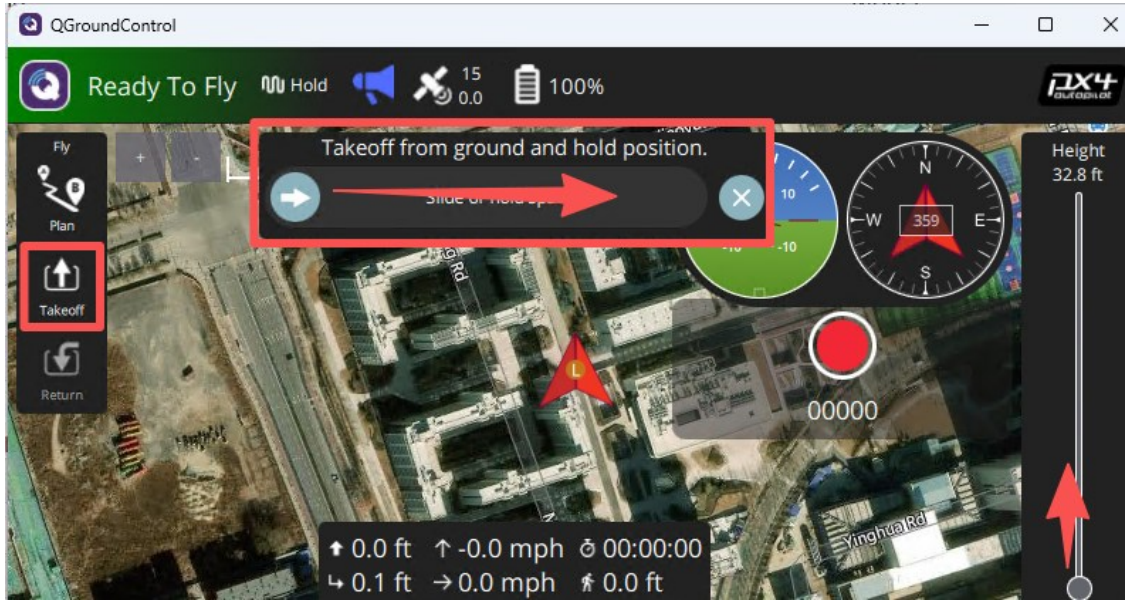


Figure: Enter make px4_sitl none in WinWSL

e5_Manual-SIL: open CopterSim, RflySim3D, then PX4; take off after 3D Fixed.

e6_BAT-Startup: OneStartup.bat opens programs and builds PX4; use SITLRun for a complete startup.

CopterSim sends simulated measurements to PX4; PX4 returns actuator commands; model updates motion.



QGC: Takeoff and slide confirmation (red boxes)

Startup scripts organize programs and links; measurement/actuator exchange closes the low-level loop.

2.5.2 Reproducing and Interpreting Representative Tasks: Organizing Startup

SITLRun.bat Sets IDs and Initial Positions with Variables

Variable	Default	Purpose
START_INDEX	1	Starting ID; with start = 1 and count = 3, IDs are 1–3
ORIGIN_POS_X/Y	0, 0	Grid origin (m); the first vehicle is here when the starting ID is 1
ORIGIN_YAW	0	Initial yaw angle in °
VEHICLE_INTERVAL	2	Spacing between adjacent vehicles in m; multiple vehicles form a square grid

Inspect RflySimAPIs\SITLRun.bat in Notepad; SimMode, UE4_MAP, etc.: see deck 2.4.
REM lines are comments; back up the original and edit only the relevant line.

Startup scripts express simulation conditions as variables; changing one variable changes one condition.

2.5 Examples and Experimental Routes

2.5.2 Reproducing and Interpreting Representative Tasks: Parameter Calculations

Design Inputs Yield Propulsion and Endurance Predictions

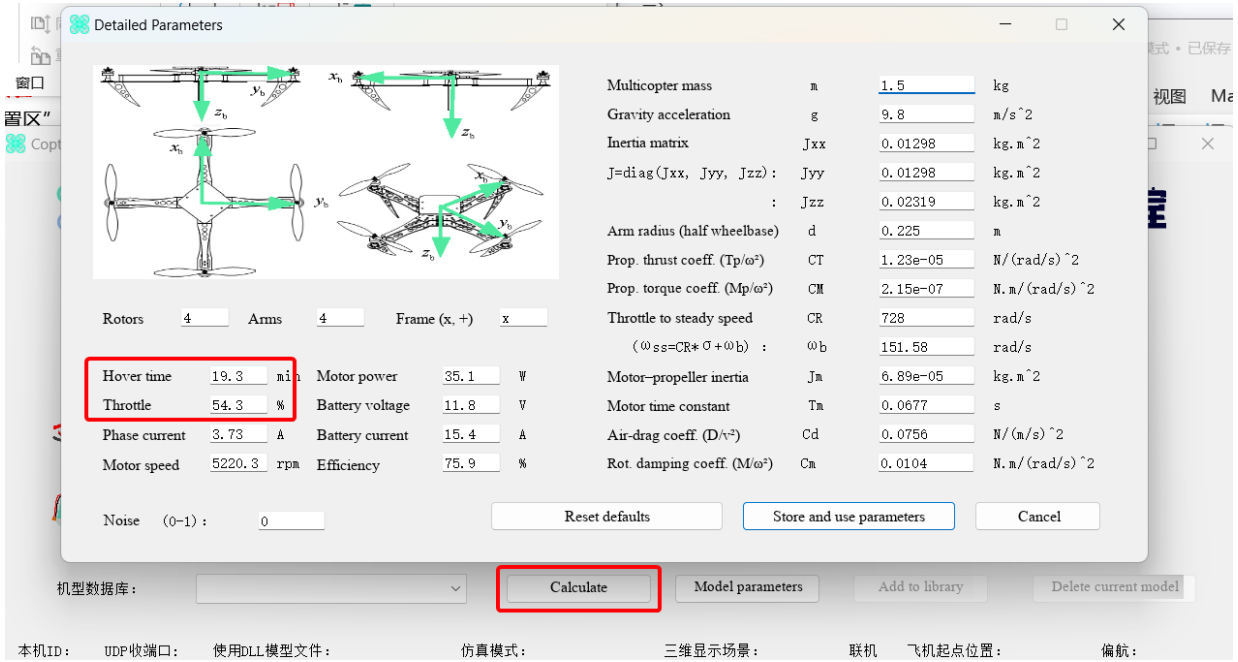


Figure: e1_CopterSim-Usage calculations; hover in red

- Design inputs: select a built-in model, set mass, dimensions, and propulsion parameters, then calculate.
- Hover prediction: $T = mg$; thrust per rotor $T/4 = C_T\omega^2$; endurance ≈ energy / average power.
- Apply settings: click Store and Use Parameters; retain the baseline for comparison.

Predictions for a given model and conditions guide configuration selection and comparison.

2.5.2 Reproducing and Interpreting Representative Tasks: Parameter Calculations

Check the Calculation: Mass to Hover Speed and Throttle

Step	Relationship	Previous Screenshot Values
① **Thrust balance**	Hover balance: $T = mg$ Four equal shares: $T/4$ per rotor	$m = 1.5\,\text{kg}$, $g = 9.8\ \text{m/s}^2$ $T = mg = 14.7\,\text{N}$, $T/4 = 3.675\,\text{N}$
② **Thrust → speed**	Per rotor: $T/4 = C_T\omega^2$ C_T: thrust coefficient; ω: rotor speed	$C_T = 1.23 \times 10^{-5}\,\text{N/(rad/s)}^2$; thus $\omega = \sqrt{3.675/C_T} \approx 547\,\text{rad/s}$
③ **Speed → throttle**	Steady-state: $\omega_{ss} = C_R\sigma + \omega_b$ slope C_R, intercept ω_b; σ is 0–1	$C_R = 728\,\text{rad/s}$, $\omega_b = 151.58\,\text{rad/s}$ $\sigma = (547 - 151.58)/728 \approx 54.3\%$

Check: 5220 rpm and 54.3% throttle match the display; 19.3 min hover exceeds the 15 min target.
Error: total thrust (14.7 N) in ② gives ≈10440 rpm and 129% throttle: a clear display mismatch.

Hover thrust balances weight; thrust coefficient gives rotor speed; speed–throttle line gives throttle.

2.5.2 Reproducing and Interpreting Representative Tasks: Experiment Conditions

Scenes and Vehicles Are Separate Configuration Objects

Figure: Different scenes and vehicles



- **Environment:** indoor spaces, roads, forests, and mountains provide different geometry and visible features.
- **Vehicles:** multirotors, ground vehicles, and fixed-wing aircraft each use their corresponding motion model.
- **Roles:** scenes describe the environment; models such as DLLs calculate motion.

Scenes change the environment and observations; vehicle models determine which vehicle is simulated.

2.5.2 Reproducing and Interpreting Representative Tasks: Camera Access

Connect a Specific Simulation Camera to QGC

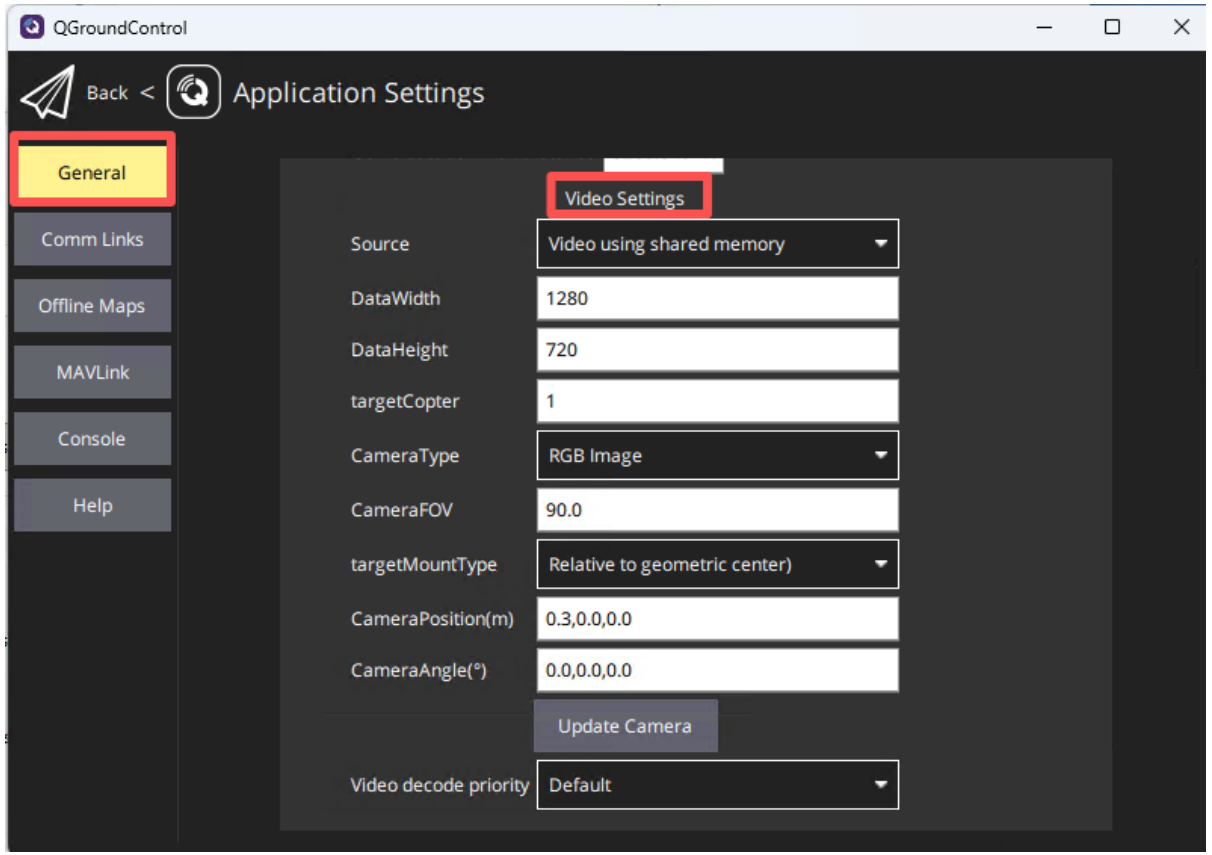


Figure: e1_QGCLoadimage video/camera settings

- **Open:** bundled QGC → Application Settings → General; set Source to Video using shared memory.
- **Camera:** targetCopter selects the vehicle; CameraNumber (28–31) selects the camera. Images use shared memory.
- **Update:** click Update Camera to send parameters via UDP; when it turns gray, click Back to see the new image.

2.5 Examples and Experimental Routes

2.5.2 Reproducing and Interpreting Representative Tasks: Interpreting Images

Assess Display Layout and Camera Parameters Separately



Figure: Main map; camera 30 picture-in-picture

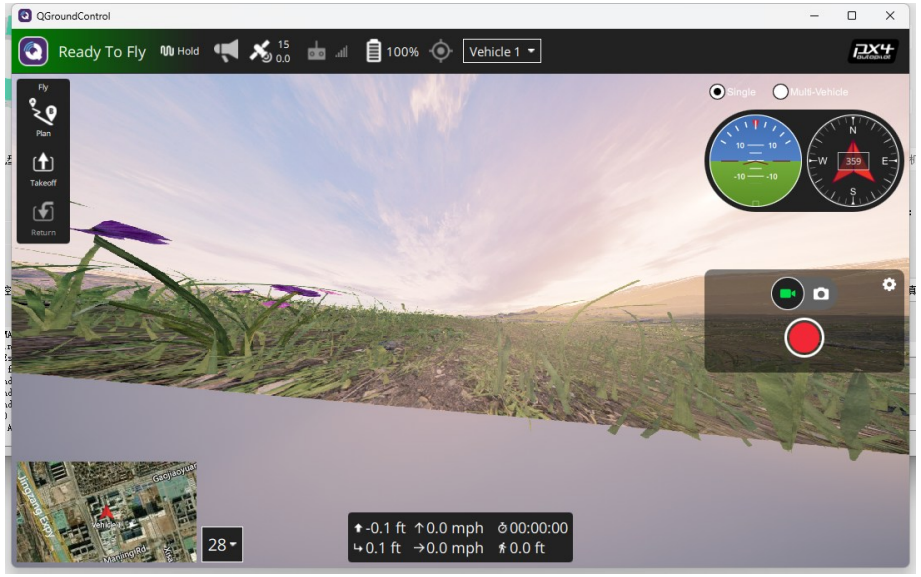


Figure: Camera 28 image as the main view

ID: camera; main view: layout; resolution: sampling; field of view: coverage; pose: viewpoint.

Resolution changes pixel sampling; field of view changes image coverage.

Chapter Outline

Example Index and Prerequisites for Advanced Work

Chapter 2
Experimental Platform Configuration

2.1 Key Tools

2.2 Installation and First Simulation

2.3 Architecture and Workflows

2.4 API and SDK Interfaces

2.5 Examples and Experimental Routes

2.6 Exercises and Discussion

3 Units in This Section

2.5.1 Finding Examples and Choosing Tasks

Textbook: Sec. 2.2.4

2.5.2 Reproducing and Interpreting Representative Tasks

Textbook: Sec. 2.4

2.5.3 Example Index and Prerequisites for Advanced Work

Textbook: Sec. 2.4

2.5.3 Example Index and Prerequisites for Advanced Work: Software Documentation Index

Tool Entry Points and Documentation Have Different Roles

Links: default PX4PSP folder on C: only; otherwise open the local example folder.

RflySimUI (Launcher)

Start from the desktop RflySim shortcut; browse tools, find examples, and read instructions.

PX4Switch

Use the GUI to switch PX4 versions and build commands, such as px4_fmu-v6x_default.

e1_RflySimSoftwareReadme

Find parameters, communication settings, and startup by software name; start tools via the launcher.

CopterSim (under e1_RflySimSoftwareReadme)

Read about models, communication, parameter calculations, and motion simulation.

2.5.3 Example Index and Prerequisites for Advanced Work: 0.ApiExps/e1_RflySimSoftwareReadme

Where to Look for Drivers, Firmware, and Runtime Help

Links: default PX4PSP folder on C: only; otherwise open the local example folder.

drivers

driver.msi installs common controller drivers; repair/reinstall if the PC cannot detect the controller.

Firmware

PX4 architecture, WSL build setup, source folders, and a multicopter control-code walkthrough.

Python38

Learn interpreter, library, and editor setup; check the actual version with python --version.

QGroundControl

Read ground-station instructions for status displays, interaction, and mission setup.

Drivers connect devices, firmware provides control programs, and the Python environment runs scripts.

2.5.3 Example Index and Prerequisites for Advanced Work: 0.ApiExps/e1_RflySimSoftwareReadme

Documentation for 3D Display and Experiment Setup

Links: default PX4PSP folder on C: only; otherwise open the local example folder.

RflySim3D

Read about 3D scenes, vehicle appearance, and interaction.

RflySimUE5

Read about UE5 scenes and display setup; the corresponding engine must be available locally.

SimCreate

Save vehicle and experiment settings for repeatable simulation startup.

viscreate

Configure virtual sensors and their image outputs.

WinWSL

Read the bundled instructions for Linux runtime and development environments.

Display, experiment organization, and sensor setup serve different roles; combine them as the task requires.

2.5.3 Example Index and Prerequisites for Advanced Work: 0.ApiExps

Configuration, Request Interfaces, and Optional Environments

Links: default PX4PSP folder on C: only; otherwise open the local example folder.

e2_FWConfig

Plan and fly fixed-wing takeoff, waypoints, and landing in QGC; includes SITL and hardware-in-the-loop (HITL; HIL in the textbook).

e3_ReqCopterSim

Discover nodes; reset communication, initial state, model, and scene; arm and command motion.

e4_VMware

Set up VMware to run an independent Linux environment on a Windows host.

e5_Ubuntu

Set up Ubuntu and development tools for the relevant Linux programs.

Requests can reset simulation conditions; virtual machines and Ubuntu run the relevant programs.

2.5.3 Example Index and Prerequisites for Advanced Work: e3_ReqCopterSim Call Sequence

Node Discovery, Configuration Resets, and Motion Control

Previous links: default PX4PSP folder on C: only; otherwise open the local example folder.

Stage	Calls Used in This Example
Node Discovery	getSimIpList lists discovered CopterSim instances; getSimIpID(1) gets ID 1's IP, probing if no heartbeat arrives; sendReSimIP directs replies to this computer.
Configuration Changes	Five interfaces, including sendReSimUdpMode, reset communication, pose, DLL model, and map; the quadrotor becomes a hexarotor and the scene becomes VisionRing.
Motion Control	Connect with InitMavLoop; wait until isPX4Ekf3DFixed is true, then call initOffboard, arm with SendMavArm, and use SendPosNED(0,0,−20,0) to fly 20 m above the origin.

Prerequisite: place HexarotorModelCTRL.dll in [InstallationDirectory]\CopterSim\external\model.

Discovery can probe actively; resets change simulation conditions; control targets drive subsequent motion.

2.5.3 Example Index and Prerequisites for Advanced Work: 0.ApiExps

Choose Firmware Builds and ROS Environments as Needed

Links: default PX4PSP folder on C: only; otherwise open the local example folder.

e6_PX4Build

Set up the PX4 toolchain; build a simulation program or onboard firmware for the selected target.

e7_ApmBuild

Set up the ArduPilot toolchain and build for the selected vehicle type.

e8_RosInstall

Set up ROS for the corresponding communication and development examples.

e9_Ros2Install

Set up ROS 2 to match the example's requirements and dependency versions.

PX4 and ArduPilot have separate build toolchains; match the ROS version to the application's interfaces.

2.5.3 Example Index and Prerequisites for Advanced Work: 0.ApiExps

Deployment Examples for Containers and Message Bridges

Links: default PX4PSP folder on C: only; otherwise open the local example folder.

e11_docker

Containers bundle apps/dependencies; cross-architecture builds must support the target architecture.

e12_PX4ROS2

Connect PX4 and ROS 2 through MicroXRCEAgent to send and receive state and control topics.

e14_GazeboSim

Set up the Gazebo simulation branch and its runtime dependencies.

e10_Mavros / 1.MavrosInstall

Set up MAVROS to translate between MAVLink messages and ROS topics and services.

Bridges translate between endpoints; connection settings route states and control targets.

2.5.3 Example Index and Prerequisites for Advanced Work: 0.ApiExps

Control and Vision Examples in Different Languages

Links: default PX4PSP folder on C: only; otherwise open the local example folder.

e10_Mavros / 2.Mavros1Offboard

Use ROS 1 with MAVROS to send position targets and observe motion in Offboard mode.

e10_Mavros / 3.Mavros2Offboard

Use ROS 2 with MAVROS to send position targets and receive mode and armed-state updates.

e13_VisAPIPyCpp / 1.rosCppExps

Subscribe to image, depth, or point-cloud messages in C++ to learn sensor-data reception.

e13_VisAPIPyCpp / 2.rosPythonExps

Subscribe to sensor topics in Python; learn ROS messages and image-format conversion.

Control sends targets; vision receives observations. Language changes preserve task inputs and outputs.

2.5.3 Example Index and Prerequisites for Advanced Work: 1.BasicExps

Basic Examples for Models, Views, and Offline Data

Links: default PX4PSP folder on C: only; otherwise open the local example folder.

e1_CopterSim-Usage

Calculate parameters and run simulations; the HITL branch also needs a physical autopilot.

e2_DLL-Load

Load fixed-wing or multirotor DLL models and observe motion; HITL requires a flight controller.

e3_RflySim3D-Shortcut-Instruct

Practice 3D viewing and keyboard shortcuts; identify objects and viewpoints.

e4_Log-Reads-Python38Env

Parse the supplied ULog and check topic CSV files; see the task in 2.5.2.

Motion models calculate states, viewpoints change observation, and logs preserve state data.

2.5.3 Example Index and Prerequisites for Advanced Work: 1.BasicExps

Startup, Code Generation, and External Motion Interfaces

Links: default PX4PSP folder on C: only; otherwise open the local example folder.

e5_Manual-SIL

Start components manually to identify startup dependencies.

e6_BAT-Startup

Read the full startup script to understand building, configuration, and process organization.

e7_Code-Generation

Generate Simulink control code and build firmware; code-generation components are required.

e8_SwarmAPI

The supplied example circles one vehicle; MATLAB/Simulink prerequisites apply.

2.5.3 Example Index and Prerequisites for Advanced Work: 1.BasicExps

Match Board Type, Build Target, and Firmware Deployment

Links: default PX4PSP folder on C: only; otherwise open the local example folder.

e15_Identify-Hardware-Command

Identify the connected device and confirm its board type.

e9_Build-Firmware

Select the build target by board type and generate matching firmware.

e10_Firmware-Upload

Flash matching hardware; USB data cable and drivers required. Check the version after reboot.

e12_PX4-App

Develop onboard apps and startup settings; test in SITL first; hardware requires Pixhawk 6X.

Board type determines the build target; deploy the resulting firmware to matching hardware.

2.5.3 Example Index and Prerequisites for Advanced Work: 1.BasicExps

Logs, Scenes, Routes, and External Piloting

Links: default PX4PSP folder on C: only; otherwise open the local example folder.

e13_Log-Get

Place empty CSV files beside CopterSim.exe; record after simulation starts and read after closing.

e14_Scene-Load

Switch indoor, forest, and other scenes; compare spatial constraints and visible features.

e16_RoutePlanning

Plan and execute a HITL route in QGC; requires Pixhawk 6C, a transmitter, and a receiver.

e17_RCusbSIL

Connect the supported USB transmitter; observe inputs and motion responses.

Recording, environment changes, missions, and piloting inputs yield different experimental results.

2.5.3 Example Index and Prerequisites for Advanced Work: 1.BasicExps

Reuse Configurations and Vehicle Model Data

Links: default PX4PSP folder on C: only; otherwise open the local example folder.

e18_PX4VersionSwitch

PX4Switch changes versions and build targets; PX4CMD changes only targets the source supports.

e19.VisCreateUsage

Save virtual-sensor settings and obtain the corresponding observations.

e20.CopterSimJsonUsage

Configure directed communication with JSON; requires three LAN hosts or virtual machines.

e21.SimcreateUsage

Save and reuse vehicle and experiment configurations; see 2.5.2.

e22.flyeval2CopterSim

Export FlyEval evaluation results as XML for CopterSim to load as a dynamics model.

JSON sets destinations; XML transfers models; experiment settings organize simulation startup.

2.5.3 Example Index and Prerequisites for Advanced Work: 2.AdvExps

What the Four Advanced Examples Add

Links: default PX4PSP folder on C: only; otherwise open the local example folder.

e1_QGCLoadimage

Receive images and configure cameras; requires the QGC bundled with the platform.

e2_RflySimAPIUsage

Learn SDK modules and calls; add state, image, or control interfaces to your program.

e3_QGCDevEnvDep

Obtain the specified QGC source; set up Qt and the compiler, then change the name or icon.

e4_MPDevEnvDep

Obtain Mission Planner source; set up Visual Studio and build a custom ground station.

The SDK links platform functions to apps; ground-station developers build custom programs from source.

2.5.3 Example Index and Prerequisites for Advanced Work: Optional Routes

Choose a Deliverable Task for Your Available Resources

Available Resources	Optional Tasks	Deliverables
Computer and bundled environment	Logs, configurations, images	Topic CSVs, configurations, image interpretation
MATLAB/Simulink	Code generation, single-vehicle external control	Generated code or trajectory records
Matching ROS environment	MAVROS, vision-data reception	Target messages or camera data
Matching flight controller USB data cable	Board identification, firmware, HITL	Firmware version, heartbeat, and state records

Choose a task you can complete, then add software or hardware as needed.

2.5.3 Example Index and Prerequisites for Advanced Work: Troubleshooting

Use Current Outputs to Narrow Down the Problem

- **Recording file stays empty:** check that CSV files are in [InstallationDirectory]\CopterSim, filename numbers match vehicle IDs, and simulation has started; read after closing CopterSim.

- **Telemetry but no response:** the return path works, so check the control side—target ID, armed state,
whether the mode accepts the target (e.g., Offboard), and 3D Fixed status.
- **Image does not change:** images use shared memory; parameters use UDP. Check targetCopter and CameraNumber; click Update Camera after changes, then Back to view the result.

- **Problem report:** identify the example, conditions, reproduction steps, and actual output; state how observation differs from expectation.

2.5.3 Example Index and Prerequisites for Advanced Work: Experiment Deliverables

Deliver a Record Someone Else Can Reproduce

- **Task and conditions:** identify the example, version, object, and required software or hardware.
- **Baseline and change:** retain original inputs; identify the one variable changed and its predicted effect.
- **Outputs and interpretation:** submit actual files or continuous observations; explain changes in time, object, physical quantities, or images.
- **Reuse and explanation:** retain reusable settings and essential steps; explain the actual results and their causes.

Example: in Python38Env, run ulog2csv log.ulg on the supplied log.ulg to obtain 32 CSV files.
Interpretation: vehicle_local_position z changes from 0 to about −2.5 m: a 2.5 m climb, held for about 76 s.
Cross-check: vehicle_global_position alt rises from about 101.7 m to 104.2 m, also about 2.5 m.
Explanation: vehicle_status nav_state follows 17 → 4 → 5: Auto Takeoff → Hold → Return.