本章目录

平台框架与通信接口

第六章 外部控制与轨迹规划

6.1 外部控制基础与控制分层
6.2 外部控制原理与控制模式
6.3 轨迹规划建模与生成
6.4 轨迹规划算法与避障
6.5 轨迹跟踪原理与高机动控制
6.6 前沿技术与控制方法
6.7 平台框架与通信接口
6.8 外部控制例程与实验验证
6.9 习题与课堂讨论

本节 5 个教学单元

6.7.1 平台框架与职责
教材对应：第6.2.1节

6.7.2 通信链路与协议
教材对应：第6.1.1.5、6.2.2.1节

6.7.3 最小控制与状态获取
教材对应：第6.2.2.2、6.2.2.4节（控制流程）

6.7.4 扩展接口与算法工具
教材：6.2.2.2–6.2.2.3、6.2.2.5、6.2.3节

6.7.5 通信优化与验收
教材对应：第6.2.2.4节（发送频率）、6.2.3节

6.7.1 平台框架与职责：核心组件

先分清谁算目标、谁算响应、谁负责显示

组件	输入与职责	主要输出
外部程序	任务、状态、环境 → 计算参考	位置、速度等目标
PX4飞控	估计状态 → 底层控制	执行器控制量、遥测
CopterSim	运动模型计算；状态与指令中转	传感器数据、运动状态
RflySim3D	位姿 → 场景显示与成像	视觉图像、可见运动
QGC地面站	配置、监视、任务与人工操作	参数、模式、任务请求

6.7 平台框架与通信接口

6.7.1 平台框架与职责：通信架构总览

沿箭头区分控制量、估计状态和视觉图像

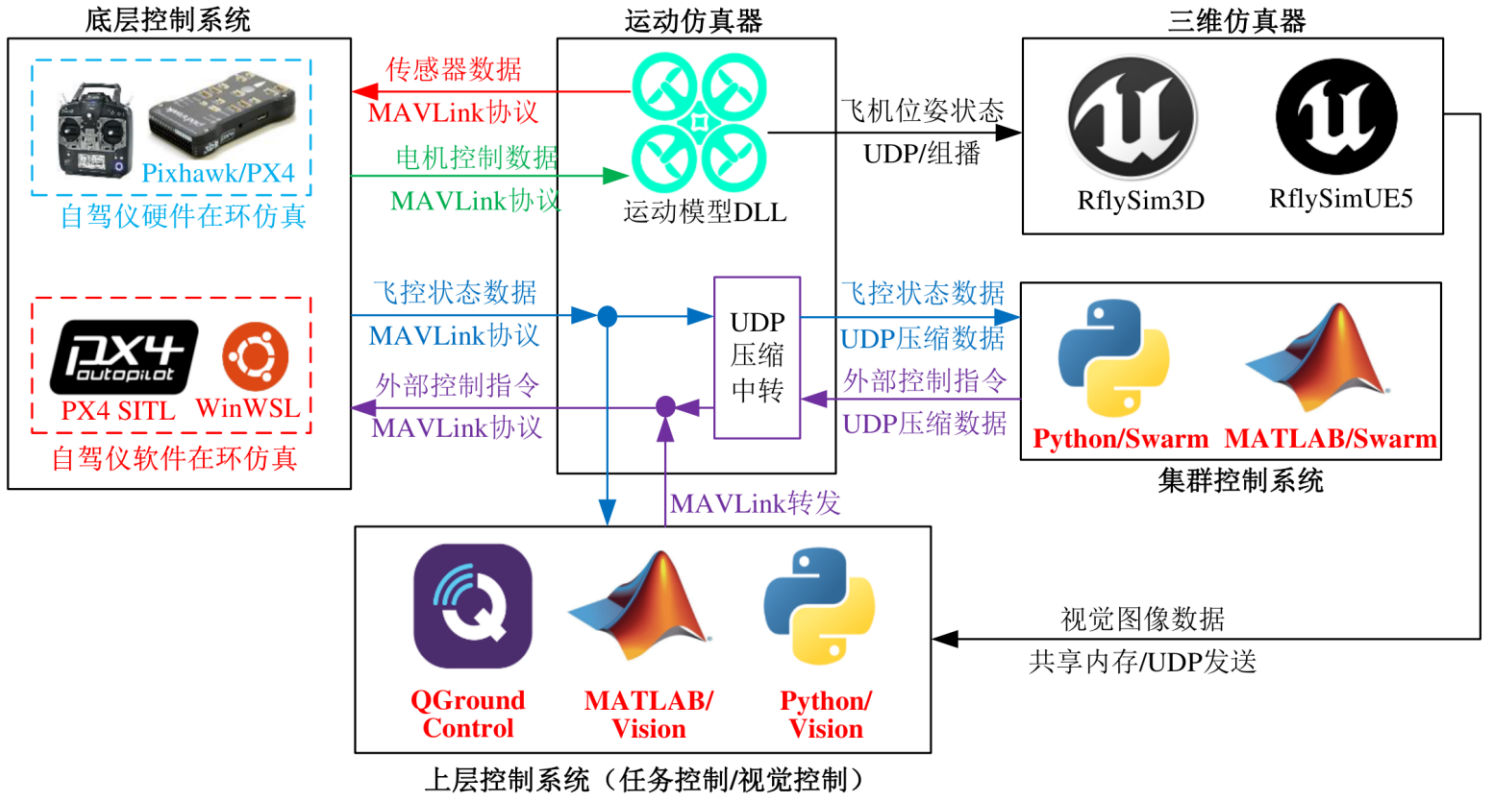


图6.8 外部控制程序的通信架构

控制量：改变模型或载具的输入。

状态：估计值与模型真值要分开。

图像：来自场景成像，供视觉程序处理。

6.7 平台框架与通信接口

6.7.1 平台框架与职责：传感器与执行器

运动模型和飞控构成底层仿真闭环

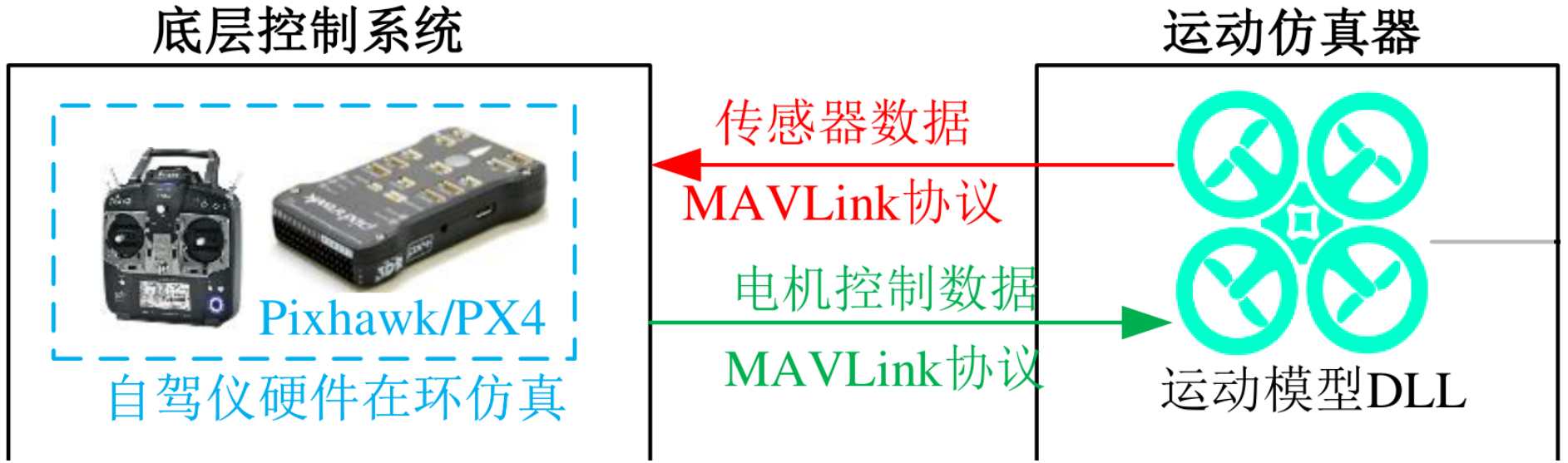


图6.8（左上）传感器输入与电机控制输出

DLL： 动态链接库；承载可运行的运动模型。

闭环： 模型生成传感器数据，飞控输出控制量，再由模型计算新的运动。

6.7 平台框架与通信接口

6.7.1 平台框架与职责：状态与指令中转

位置目标向内发送，估计状态向外回传

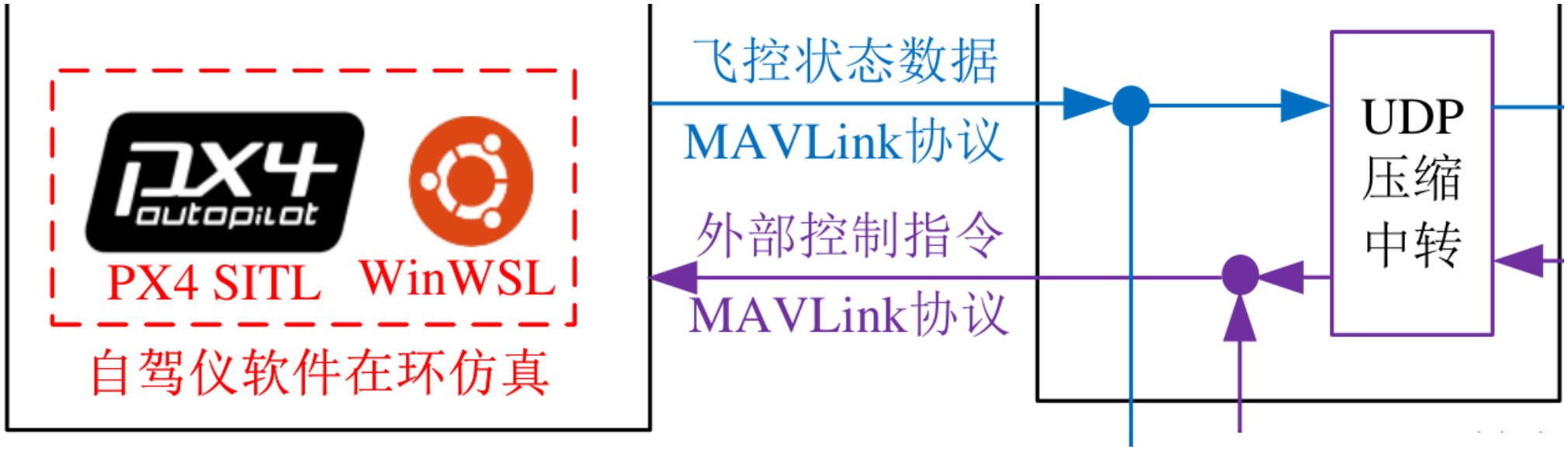


图6.8（左中）状态回传与外部控制指令

MAVLink：定义消息字段、编号和编码方式，可承载状态与指令。

UDP中转：选择完整或精简格式时，发送端、接收端必须采用同一数据定义。

6.7 平台框架与通信接口

6.7.1 平台框架与职责：显示与感知分支

三维显示需要位姿，视觉控制还需要图像



图6.8（右上）位姿驱动三维显示

位姿通道：传递位置与姿态，决定载具和相机在场景中的状态。

图像通道：传递相机图像，供视觉程序计算；数据量通常远大于一条位置消息。

6.7.1 平台框架与职责：集群分支

集群接口按每架载具的身份组织状态和目标

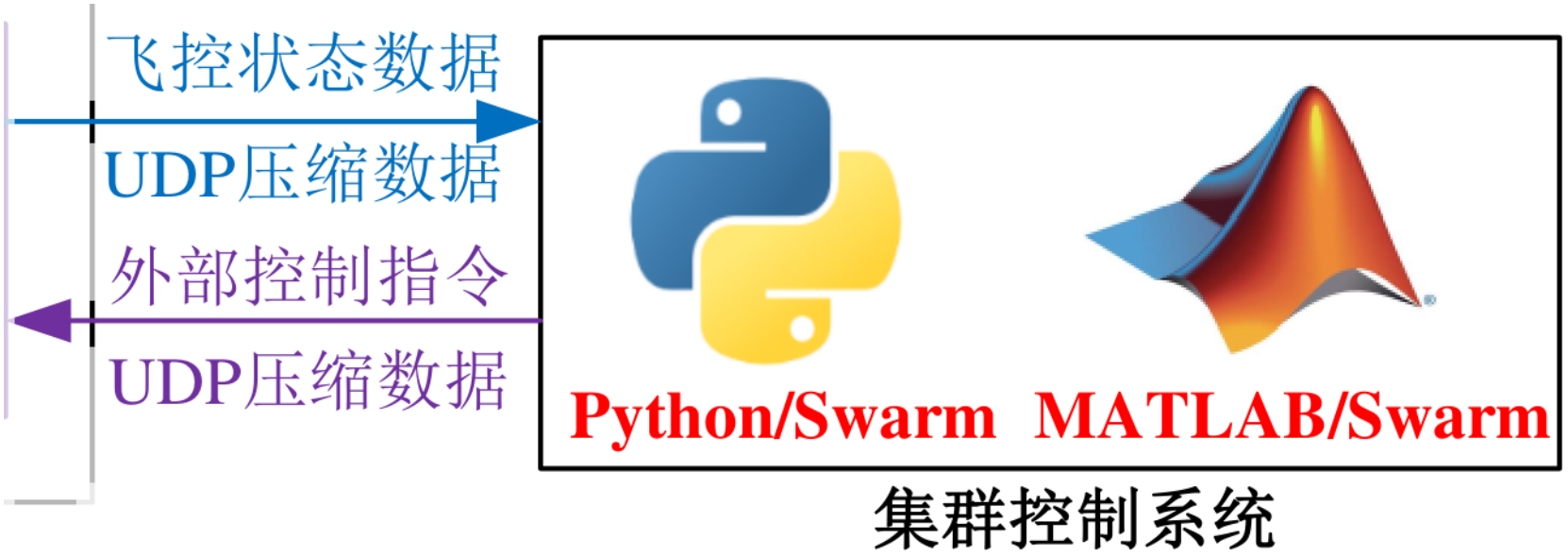


图6.8（右中）集群状态与指令通道

每机身份： ID、目标端口与接收状态必须一一对应。

精简数据： 保留任务需要的字段，并记录各字段的时间、坐标系与单位。

6.7.1 平台框架与职责：任务与视觉控制

上层程序把状态、图像和任务合成运动目标

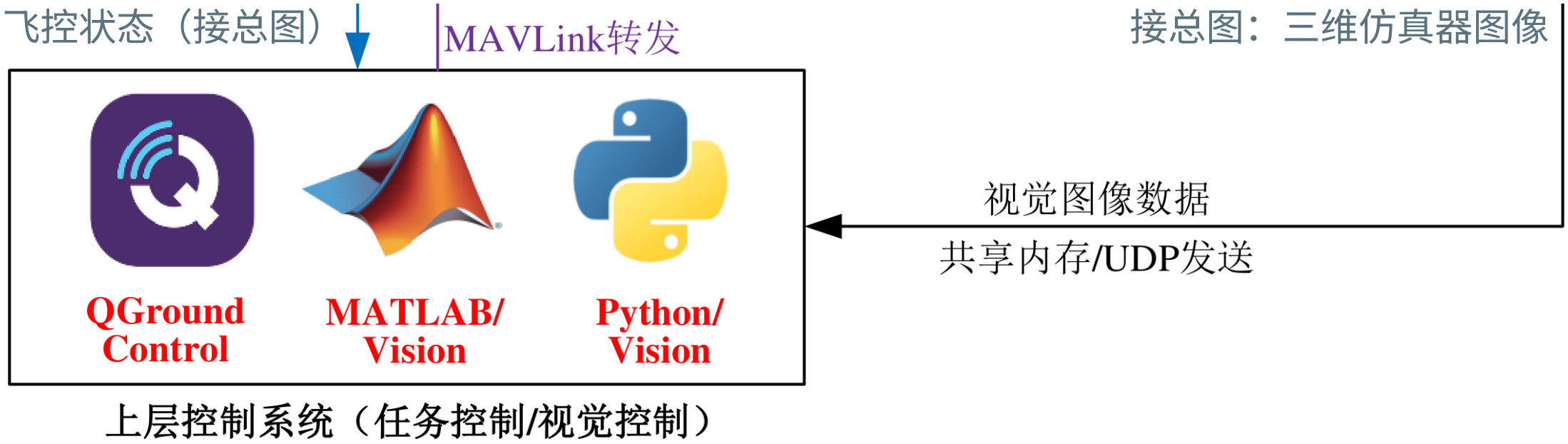


图6.8（下部）任务、视觉与飞控通信

同一目标：任务或视觉程序计算，再经接口交给飞控。

仿真迁到实机：复用消息语义，重新核对端口、坐标原点、状态估计和时序。

本章目录

平台框架与通信接口

第六章 外部控制与轨迹规划

6.1 外部控制基础与控制分层
6.2 外部控制原理与控制模式
6.3 轨迹规划建模与生成
6.4 轨迹规划算法与避障
6.5 轨迹跟踪原理与高机动控制
6.6 前沿技术与控制方法
6.7 平台框架与通信接口
6.8 外部控制例程与实验验证
6.9 习题与课堂讨论

本节 5 个教学单元

6.7.1 平台框架与职责
教材对应：第6.2.1节

6.7.2 通信链路与协议
教材对应：第6.1.1.5、6.2.2.1节

6.7.3 最小控制与状态获取
教材对应：第6.2.2.2、6.2.2.4节（控制流程）

6.7.4 扩展接口与算法工具
教材：6.2.2.2–6.2.2.3、6.2.2.5、6.2.3节

6.7.5 通信优化与验收
教材对应：第6.2.2.4节（发送频率）、6.2.3节

6.7.2 通信链路与协议：链路与软件分层

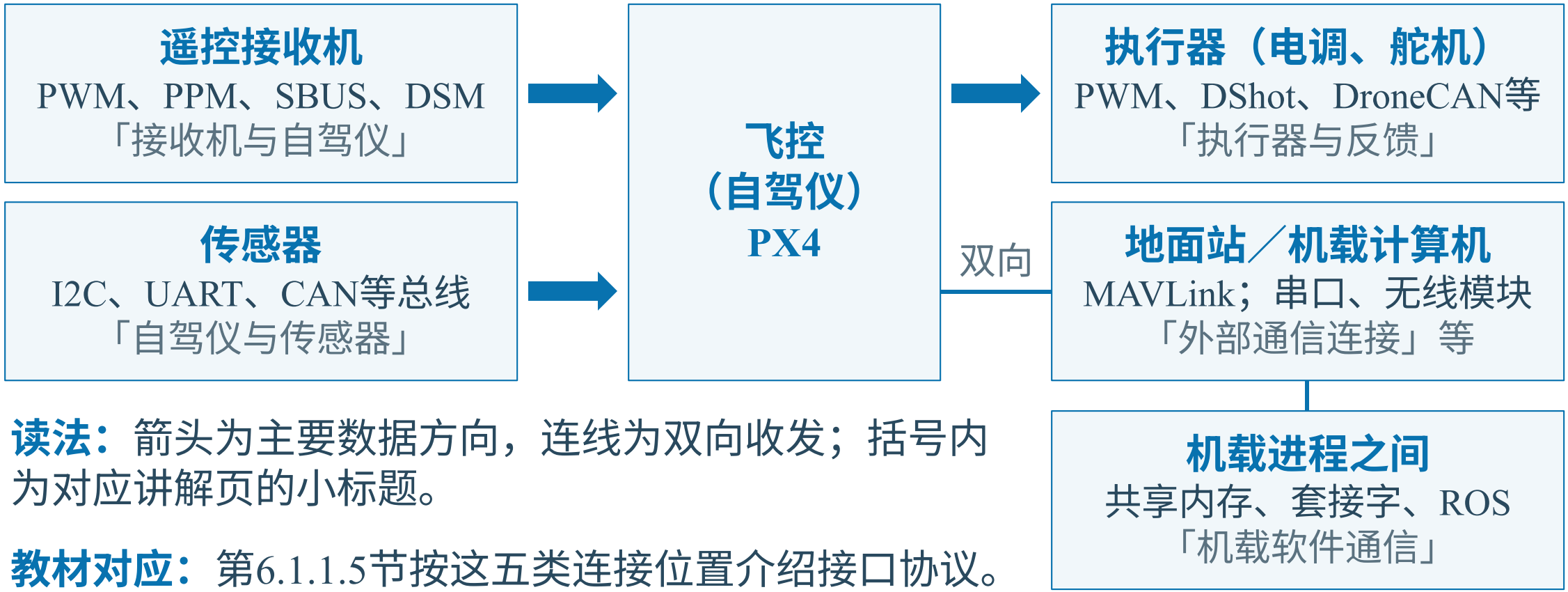
同一条链路可以承载多种消息协议

层次	解决的问题	本节例子
应用与SDK	怎样把任务写成函数调用	SendPosNED；MAVSDK
消息协议	字段、单位、身份如何解释	MAVLink；平台定制数据
传输方式	消息字节怎样送到进程	UDP数据报；TCP字节流
连接链路	使用什么设备和通道	串口、数传、Wi-Fi、蜂窝网

串口电气形式：TTL、RS232、RS485；电平、接线和转换模块必须匹配。

6.7.2 通信链路与协议：连接位置总览

飞控与外部的五类连接，各用不同的接口协议



读法：箭头为主要数据方向，连线为双向收发；括号内为对应讲解页的小标题。

教材对应：第6.1.1.5节按这五类连接位置介绍接口协议。

6.7.2 通信链路与协议：外部通信连接

TELEM1和TELEM2是可配置的串口

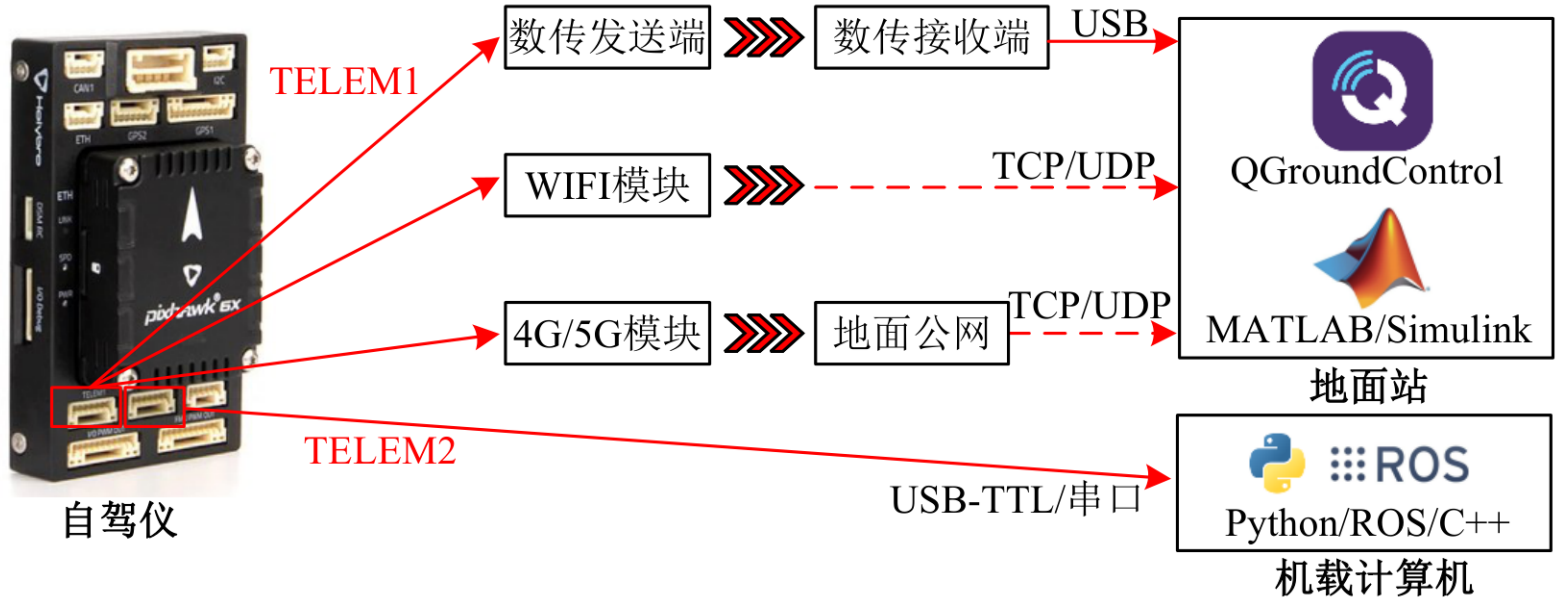


图6.9 自驾仪外部通信框架

TELEM1：常接数传、Wi-Fi或蜂窝模块，用于遥测与地面站。

TELEM2：常经USB-TTL或串口接机载计算机。

配置：用途由飞控参数设定；无线能力来自外接模块。

6.7.2 通信链路与协议：模块选择

按距离、负载和抖动选择通信模块

模块	适用任务	选择时核对
USB-TTL／串口	机载直连、硬件调试	电平、接线、波特率
无线数传	远程遥测和运动指令	频段、吞吐、丢包、覆盖
Wi-Fi	局域网控制、较大数据量	拥塞、干扰、断连恢复
4G／5G	跨区域接入地面服务	公网地址、时延抖动、覆盖

入口：[安装目录]/RflySimAPIs/6.RflySimExtCtrl/0.ApiExps/e0_ComConnDemo/Readme.pdf

6.7.2 通信链路与协议：接收机与自驾仪

遥控输入协议传递通道值，不等同于电机输出

形式	通道怎样表达	特点与使用要点
PWM	每路用脉冲宽度表示通道值	操纵输入；依模式解释
PPM	一串脉冲复用多个通道	接线少；抗干扰弱于SBUS
SBUS	串行数字帧承载多个通道	抗干扰强；注意信号反相
DSM／DSMX	Spektrum体系的控制链路	核对接收机及协议代际

SBUS用于接收机—飞控有线通信；无线抗干扰还取决于收发系统。

相同的信号形式，在遥控输入端与执行器输出端承载不同含义。

6.7.2 通信链路与协议：自驾仪与传感器

先辨认总线，再辨认总线中的数据格式

接口／格式	所属层次	典型用途与核对项
IIC（I^2C）	板级总线	引脚少、适合低速；地址与上拉
UART	异步串口接口	定位、测距；波特率与帧格式
CAN	多节点通信总线	高可靠、可组冗余网络；终端电阻
UAVCAN/DroneCAN	CAN消息与节点协议	版本、节点ID、消息类型
NMEA／UBX	卫星导航（GNSS）格式	NMEA可读；UBX省带宽、效率高

6.7.2 通信链路与协议：执行器与反馈

执行器协议把控制量交给舵机或电调

形式	命令表示	反馈与条件
PWM	脉冲宽度	常规PWM信号不携带状态回传
OneShot／MultiShot	更短的脉冲时序	仍属脉冲形式；核对电调支持
DShot／双向DShot	带校验的数字命令	回传eRPM等，依实现而定
DroneCAN	总线控制消息	转速、温度等字段按设备核对

串口遥测用于状态回传；BLHeli32是固件体系，非通用串口协议。

6.7.2 通信链路与协议：机载软件通信

进程通信负责交换数据，算法负责解释数据

机制	交换方式	要明确的规则
共享内存	多个进程访问同一块数据	同步、所有权、数据更新标志
管道／消息队列	按字节流或消息边界传递	积压、阻塞、消息顺序
套接字	本机或网络进程收发	地址、端口、分帧与超时
ROS话题／服务	发布订阅／请求应答	类型、时间戳、坐标与QoS

DDS：数据分发服务；QoS：服务质量策略，用于约定可靠性、队列等通信行为。

6.7.2 通信链路与协议：仿真端口

端口号必须带上发送者和接收者

$$p_{\mathrm{cmd}} = 20100 + 2(\mathrm{ID} - 1), \quad p_{\mathrm{state}} = p_{\mathrm{cmd}} + 1$$



例：ID=1用20100／20101，ID=2用20102／20103，ID=3用20104／20105。

30100系列：模型真值与综合模型接口；格式和用途按选定模式核对。

连接配置：UDP使用IP和端口；串口还要指定设备名与波特率。

本章目录

平台框架与通信接口

第六章 外部控制与轨迹规划

6.1 外部控制基础与控制分层
6.2 外部控制原理与控制模式
6.3 轨迹规划建模与生成
6.4 轨迹规划算法与避障
6.5 轨迹跟踪原理与高机动控制
6.6 前沿技术与控制方法
6.7 平台框架与通信接口
6.8 外部控制例程与实验验证
6.9 习题与课堂讨论

本节 5 个教学单元

6.7.1 平台框架与职责
教材对应：第6.2.1节

6.7.2 通信链路与协议
教材对应：第6.1.1.5、6.2.2.1节

6.7.3 最小控制与状态获取
教材对应：第6.2.2.2、6.2.2.4节（控制流程）

6.7.4 扩展接口与算法工具
教材：6.2.2.2–6.2.2.3、6.2.2.5、6.2.3节

6.7.5 通信优化与验收
教材对应：第6.2.2.4节（发送频率）、6.2.3节

6.7.3 最小控制与状态获取：外部控制开发流程

图中的“结束”是初始化结束，还不是任务完成

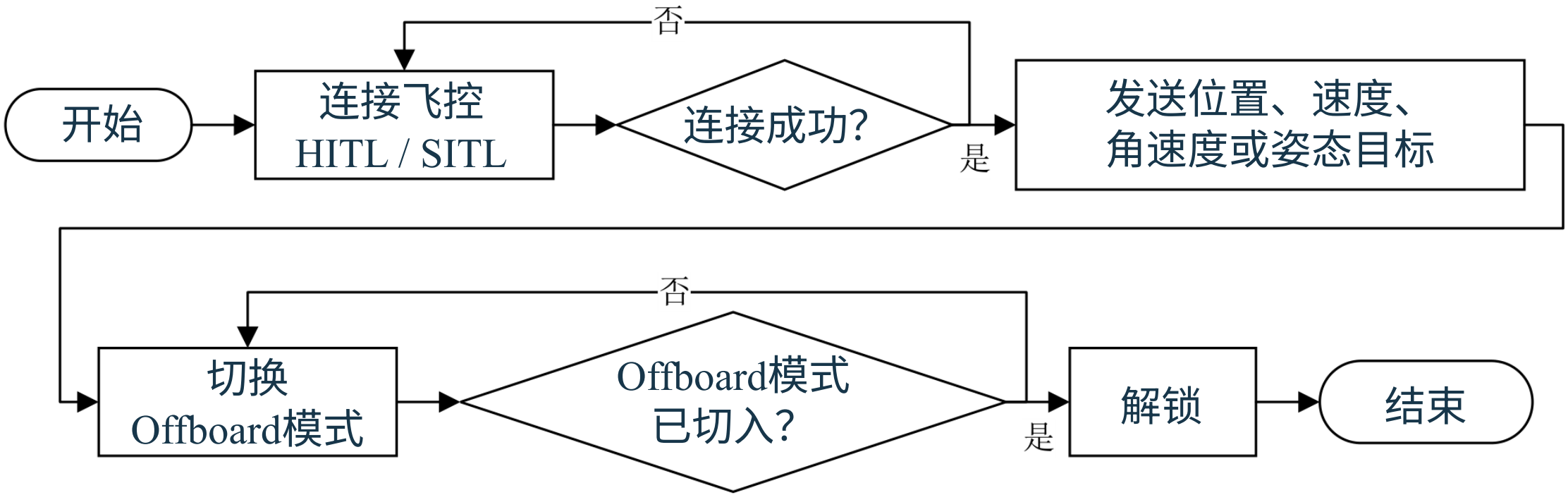


图6.10 外部控制程序开发流程（HITL：硬件在环；SITL：软件在环）

解锁（arm）后：执行监测 → 降落／接管 → 着陆并上锁（disarm）→ 清理连接。

6.7.3 最小控制与状态获取：完整状态流程

每个阶段都由收到的状态决定能否推进

1 准备	2 进入	3 执行	4 结束
连接、心跳 位置与健康状态	有效目标流 模式确认、解锁	更新参考 反馈、误差、超时	降落或接管 着陆、上锁

推进条件：心跳（HEARTBEAT）、模式、位置和终态都以回传状态为准。

异常分支：状态过期、模式丢失或误差超限，转入已约定的降落／人工接管流程。

6.7.3 最小控制与状态获取：例程入口

从已发布例程进入，先限定仿真起始状态

[安装目录]/RflySimAPIs/6.RflySimExtCtrl/0.ApiExps/e1_PX4MavCtrlAPITest/

文件	用途	操作前确认
Readme.pdf	实验目的、配置与步骤	仿真条件与当前安装版一致
PX4MavCtrlAPITest.bat	启动仿真、QGC与场景	需管理员权限；关闭同名进程
Python38Run.bat	打开平台Python环境	SDK可直接导入
PX4MavCtrlAPITest.py	控制与场景API阅读入口	分清控制、显示与清理函数

e1用initOffboard自动进入；本节改用显式发送与状态确认，仅限仿真。

6.7.3 最小控制与状态获取：例程运行与判据

e1例程按启动、就绪、运行、退出四步操作

1 启动：以管理员身份运行.bat，打开QGC、CopterSim和两个RflySim3D窗口。

2 就绪：CopterSim日志显示GPS 3D fixed & EKF initialization finished。

3 运行：双击Python38Run.bat，在该窗口输入python PX4MavCtrlAPITest.py。

4 退出：在批处理窗口按任意键，结束全部同名CopterSim、QGC和RflySim3D；启动时也会先结束已运行的CopterSim。

现象：两个三维窗口分别为前置相机和全局视角，出现人物模型，飞机起飞；源码目标为上方1.7 m，10 s后改为0.2 m/s下降。

排查：缺模块→改用Python38Run.bat环境；不起飞或解锁失败→等初始化完成并确认QGC已连接；三维显示异常→检查显卡驱动。

6.7.3 最小控制与状态获取：SDK副作用

初始化、退出和清理函数都有具体控制含义

函数	安装版实现中的动作	本节采用的边界
InitMavLoop	启动接收；并发送上锁（disarm）命令	地面、未解锁的仿真起点
initOffboard	启发送、自动解锁；改参数（见下）	关闭失联保护，仅限仿真； 本节改用显式流程
endOffboard	停发送；不发LAND，不恢复被改参数	不能作为降落完成判据
stopRun	若已解锁，强制上锁（disarm）	确认着陆并上锁后再调用

initOffboard改写：NAV_RCL_ACT、NAV_DLL_ACT置0，COM_RCL_EXCEPT置4。

6.7.3 最小控制与状态获取：SDK参数与恢复

initOffboard 改写失联参数，真机前要恢复

[安装目录]/RflySimAPIs/RflySimSDK/ctrl/PX4MavCtrlV4.py（PX4 MAVLink 分支）

initOffboard() 写入下表三个参数；endOffboard() 与 stopRun() 都不恢复。

参数	写入值与作用	真机使用前
NAV_RCL_ACT	置0：遥控失联不触发动作	恢复遥控失联动作
NAV_DLL_ACT	置0：数传失联不触发动作	恢复数传失联动作
COM_RCL_EXCEPT	置4：Offboard 忽略遥控丢失	恢复遥控丢失处理

恢复要求：先记录原值，真机使用前逐项改回或按机型重新设定。

函数行为与结束顺序：见课件6.7「SDK副作用」「结束与异常处理」页。

6.7.3 最小控制与状态获取：初始化与消息钩子

连接后由一个接收循环收消息，再分发给观察者

```
import time
import PX4MavCtrlV4 as PX4MavCtrl
mav = PX4MavCtrl.PX4MavCtrler(1)
mav.InitMavLoop(2)
conn = mav.the_connection
conn.message_hooks.append(on_message)
```

连接验收：收到目标飞控的HEARTBEAT；本地位置持续更新，模式与解锁状态可读取。

ID = 1：
对应第一架载具的端口。

模式 = 2：
MAVLink_Full（完整）；
与CopterSim模式一致。

消息钩子：
先定义on_message回调；
再注册，接收者不变。

一个接收者负责解析，多种观察者使用同一批收到的数据。

6.7.3 最小控制与状态获取：状态获取与新鲜度

位置、时间戳和接收时刻必须来自同一条消息

```
def on_message(conn, msg):
    src = (msg.get_srcSystem(), msg.get_srcComponent())
    if src != (1, 1): return
    if msg.get_type() == "LOCAL_POSITION_NED":
        sample = (msg.time_boot_ms,
                  msg.x, msg.y, msg.z,
                  time.monotonic())
        save_sample(sample)
```

time_boot_ms：飞控启动后ms；monotonic()：本机s。
两时基不能直接相减；NED为北／东／地（m）。
uavPosNED为估计；truePosNED为真值，需核对原点。
uavTimeStmp不能证明位置数据新鲜。

来源（本例）：
系统1、飞控部件1；
按实际心跳核对。

原子快照：锁内读写；
重启/回绕先退出，
再重建时间基准。

两项年龄均≤0.2 s：
距最近接收；
距最近样本推进。

6.7.3 最小控制与状态获取：位置指令与发送

写入目标变量后，还要把消息真正发出去

```
mav.SendPosNED(0, 0, -1.7, 0)

def publish():
    mav.sendMavOffboardAPI(
        mav.type_mask, mav.coordinate_frame,
        mav.pos, mav.vel, mav.acc,
        mav.yaw, mav.yawrate)
```

位置：
北0 m、东0 m，
下−1.7 m。

偏航：
0 rad；
不是完整三轴姿态。

掩码：
使用位置与偏航；
忽略其余字段。

设置参考、编码发送和飞控实际执行，是三个可分别检查的环节。

6.7.3 最小控制与状态获取：进入板外控制

先形成目标流，再请求模式和解锁

```
mav.SendPosNED(*position_now(), yaw_now())
pump(mav, 1.5)  # 1.5 s, 30 Hz
mav.SendSetMode(6)
wait_for(mav, watch.offboard, 5)  # 5 s
mav.SendMavArm(True)
wait_for(mav, watch.armed, 5)  # 5 s
mav.SendPosNED(0, 0, -1.7, 0)
```

v1.12.3：断流超1 s（COM_OF_LOSS_T）判丢失；
解锁后重设起飞目标，再按误差判定到达。
其他版本见课件5.5「控制输出的版本边界」页。

辅助函数（非SDK）：
等待时按30 Hz续发；
定义见「参考实现」。

模式值6：
PX4的Offboard主模式；
读取心跳确认。

SDK标志：
isInOffboard是发送标志；
不等于飞控当前模式。

6.7.3 最小控制与状态获取：最小闭环验收

目标到达由实际误差与保持时间共同判断

$$e = \|\mathbf{p}_{\text{ref}} - \mathbf{p}_{\text{meas}}\|_2$$

参考 $\mathbf{p}_{\text{ref}} = (0,0,-1.7)$ m；反馈 $\mathbf{p}_{\text{meas}}$ 见表；同一局部NED原点。
e 为两点的三维距离，单位m。

实际反馈位置 / m	位置误差 / m	能否判定到达
(0, 0, 0)	1.70	否，仍在起点
(0.15, −0.20, −1.70)	0.25	否，超出0.20 m
(0.06, −0.10, −1.68)	0.12	瞬时达标，还需保持2 s

教学判据：误差≤0.20 m连续保持2 s，且样本新鲜、模式仍为Offboard。

6.7.3 最小控制与状态获取：结束与异常处理

先确认降落和上锁，再清理通信

```
mav.SendSetMode(4, 6)
wait_for(mav, watch.auto_land, 5)
wait_for(mav, watch.on_ground, 60, False)
wait_for(mav, watch.disarmed, 15, False)
mav.stopRun()
```

endOffboard：停止发送，不发LAND。
stopRun：已解锁时会强制上锁。
终态：回传着陆状态＋心跳解锁标志清除。

模式枚举值：
PX4主模式AUTO=4；
子模式LAND=6。

False参数：
降落模式确认后停目标；
继续接收状态。

失败分支：
超时转接管；不把stopRun
放入无条件finally。

6.7.3 最小控制与状态获取：参考实现（状态存储）

同一锁内保存接收时刻与最近样本推进时刻

```
import threading, time
from pymavlink import mavutil
lock, latest = threading.Lock(), {}
FRESH = dict(POS=.2,ATT=.2,HB=3,EXT=3)
def put(key, item, t, src=None):
    with lock:
        old = latest.get(key)
        new = (src is None or old is None
               or src > old[3])
        if new:
            latest[key] = (item, t, t, src)
        else:
            latest[key] = (old[0], t,
                           old[2], old[3])
def save_sample(s):
    put("POS", s, s[4], s[0])
NAME = {"ATTITUDE":"ATT",
        "HEARTBEAT":"HB",
        "EXTENDED_SYS_STATE":"EXT"}
def on_message(conn, m):
    ids = (m.get_srcSystem(),
           m.get_srcComponent())
    if ids != (1, 1): return
    t, k = time.monotonic(), m.get_type()
    if k == "LOCAL_POSITION_NED":
        save_sample((m.time_boot_ms,
                     m.x, m.y, m.z, t))
    elif k in NAME: put(NAME[k], m, t)
```

6.7.3 最小控制与状态获取：参考实现（读取、续发与等待）

按门限读取新鲜状态，等待时继续发送目标

```
def get(key, must=False):  # 新鲜内容；过期返回 None，must=True 时抛错
    with lock: item, now = latest.get(key), time.monotonic()
    if item and max(now - item[1], now - item[2]) <= FRESH[key]: return item[0]
    if must: raise TimeoutError(key + " 过期")
# publish()：见「位置指令与发送」页，按当前目标调用 mav.sendMavOffboardAPI
def pump(mav, secs):  # secs 秒内按约30 Hz调用 publish()
    end = time.monotonic() + secs
    while time.monotonic() < end: publish(); time.sleep(1 / 30)
def wait_for(mav, pred, timeout, keep_sending=True):  # pred() 为真才返回
    end = time.monotonic() + timeout
    while not pred():
        if time.monotonic() > end: raise TimeoutError(pred.__name__ + " 超时")
        if keep_sending: publish()
        time.sleep(1 / 30)
```

6.7.3 最小控制与状态获取：参考实现（位姿读取与状态谓词）

谓词只读回传的模式、解锁标志和着陆状态

```
def position_now(): return get("POS", True)[1:4]  # 新鲜位置 (x, y, z) / m
def yaw_now(): return get("ATT", True).yaw  # 新鲜偏航角 / rad
ARM = mavutil.mavlink.MAV_MODE_FLAG_SAFETY_ARMED  # base_mode 中的解锁标志位
def field(key, name):  # 新鲜消息的字段；未收到或已过期为 None
    v = get(key); return None if v is None else getattr(v, name)
def px4_mode():  # custom_mode：第16–23位主模式，第24–31位子模式（PX4 v1.12.3）
    cm = field("HB", "custom_mode") or 0; return (cm >> 16) & 0xFF, (cm >> 24) & 0xFF
class watch:  # 各谓词只读飞控回传；消息过期时为 False
    def offboard(): return px4_mode()[0] == 6  # 主模式 OFFBOARD
    def auto_land(): return px4_mode() == (4, 6)  # 主模式 AUTO，子模式 LAND
    def armed(): bm = field("HB", "base_mode"); return bm is not None and bool(bm & ARM)
    def disarmed(): bm = field("HB", "base_mode"); return bm is not None and not bm & ARM
    def on_ground(): return field("EXT", "landed_state") == 1  # 1：在地面（ON_GROUND）
```

拼接顺序： 导入→函数定义→连接注册→保持目标→进入→验收→降落→清理

本章目录

平台框架与通信接口

第六章 外部控制与轨迹规划

6.1 外部控制基础与控制分层
6.2 外部控制原理与控制模式
6.3 轨迹规划建模与生成
6.4 轨迹规划算法与避障
6.5 轨迹跟踪原理与高机动控制
6.6 前沿技术与控制方法
6.7 平台框架与通信接口
6.8 外部控制例程与实验验证
6.9 习题与课堂讨论

本节 5 个教学单元

6.7.1 平台框架与职责
教材对应：第6.2.1节

6.7.2 通信链路与协议
教材对应：第6.1.1.5、6.2.2.1节

6.7.3 最小控制与状态获取
教材对应：第6.2.2.2、6.2.2.4节（控制流程）

6.7.4 扩展接口与算法工具
教材：6.2.2.2–6.2.2.3、6.2.2.5、6.2.3节

6.7.5 通信优化与验收
教材对应：第6.2.2.4节（发送频率）、6.2.3节

6.7.4 扩展接口与算法工具：开发接口比较

接口库改变编程方式，闭环验收要求保持一致

接口路线	面向程序的形式	适合的开发需求
Pymavlink／RflySim SDK	消息字段／常用控制函数	Python教学、协议调试
MAVSDK	动作Action／遥测Telemetry 板外控制Offboard	任务应用、多语言客户端
MAVROS	ROS话题与服务 ↔ MAVLink	已有ROS节点与感知系统
uXRCE-DDS／ROS 2（v1.14及以上）	PX4内部消息总线uORB 选定话题 ↔ ROS 2	直接使用PX4内部消息类型

主线：Python／RflySim SDK；MAVSDK、MAVROS、ROS 2按已有环境选学。

共同验收：状态有效、目标正确、模式确认、运动误差可读、终态明确。

6.7.4 扩展接口与算法工具：MAVSDK示例

MAVSDK用异步API组织目标和遥测

```
from mavsdk.offboard import PositionNedYaw
await drone.offboard.set_position_ned(
    PositionNedYaw(0, 0, 0, 0))
await drone.offboard.start()
await drone.offboard.set_position_ned(
    PositionNedYaw(5, 0, -5, 90))
```

片段前提：已连接、定位健康且已action.arm()；
在async函数内调用；原例末尾仅offboard.stop()。

目录：6.RflySimExtCtrl/0.ApiExps/e17_MAVSDKExps/
入口：2.OffboardPosNEDExp/offboard_position_ned.py

目标语义：
北5 m、东0 m，
上方5 m，朝东90°。

反馈入口：
telemetry位置与模式；
检查in_air和armed。

补充收尾：
另加action.land()；
确认已着陆并上锁。

6.7.4 扩展接口与算法工具：ROS集成与坐标

MAVROS话题中的位置通常使用东、北、上

操作	ROS 1的MAVROS接口	检查内容
持续目标	/mavros/setpoint_position/local	PoseStamped；ENU坐标
位置反馈	/mavros/local_position/pose	时间、原点与实际位置
模式与解锁	/mavros/state；服务请求	服务响应后再看回传状态

$$(x, y, z)_{\mathrm{ENU}} = (3,2,4)\ \mathrm{m} \Rightarrow (x, y, z)_{\mathrm{NED}} = (2,3,-4)\ \mathrm{m}$$

目录：6.RflySimExtCtrl/0.ApiExps/e18_MavrosExps/
入口：mavros_offboard_posctl_test.py；先核对ROS环境和Readme.pdf。

6.7.4 扩展接口与算法工具：uXRCE-DDS路线

直通路线省去MAVLink与MAVROS，但仍有客户端和代理



匹配：固件与px4_msgs消息定义、dds_topics配置、命名空间及QoS。

二次开发：uORB导出定义要同步；MAVLink也支持两端一致的自定义消息。

uXRCE-DDS需PX4 v1.14及以上；版本确认见课件5.5「控制输出的版本边界」页。接口文档：https://docs.px4.io/v1.14/en/middleware/uxrce_dds

目录：6.RflySimExtCtrl/0.ApiExps/e19_uXRCE-DDS_ROS2CtrlExps/

6.7.4 扩展接口与算法工具：常用接口及二次开发

按算法所在位置选择Python、ROS、模型或原生程序

实现位置	开发方式	对应接口与验收
外部脚本	Python：PX4MavCtrlV4.py	UDP或串口收发；接口签名与时序
ROS节点	ROS：PX4MavCtrlV4ROS.py	经MAVROS收发；需Linux与ROS环境
外部模型	Simulink：OffboardAPI.slx	UAV Toolbox模块；输入输出单位
原生应用	C++／MAVLink或MAVSDK	消息定义、线程、超时与记录
综合模型DLL	控制器与运动模型一起运行	inSIL外部输入口；模型边界与真值

入口：6.RflySimExtCtrl/1.BasicExps/e4_OffboardCtrl/OffboardAPI.slx
接口库：RflySimSDK/ctrl/；字段查阅：本章API.pdf与RflySimSDK/html/index.html。

6.7.4 扩展接口与算法工具：工具箱概览

算法库分别承担模型、制导与轨迹生成

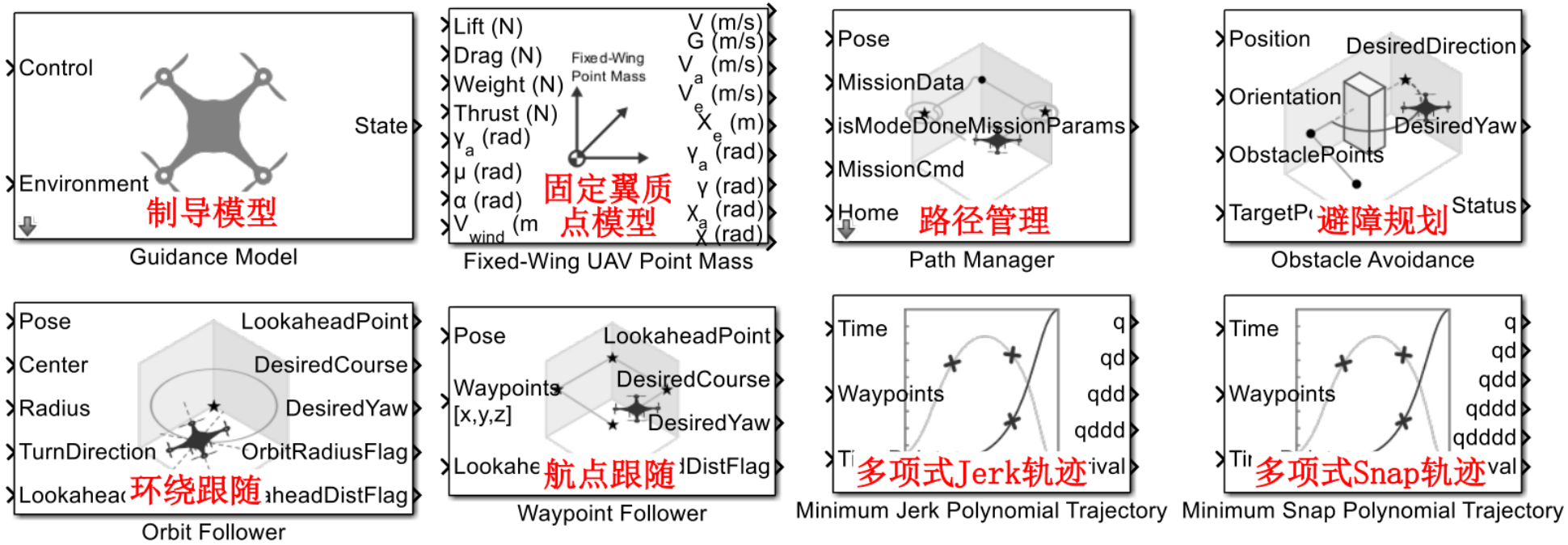


图6.11 Simulink无人机算法库

库中模块分属模型、制导参考与轨迹参考；控制器需另行设计。

6.7.4 扩展接口与算法工具：模型与制导

制导模型与路径跟随器输出不同层次的量

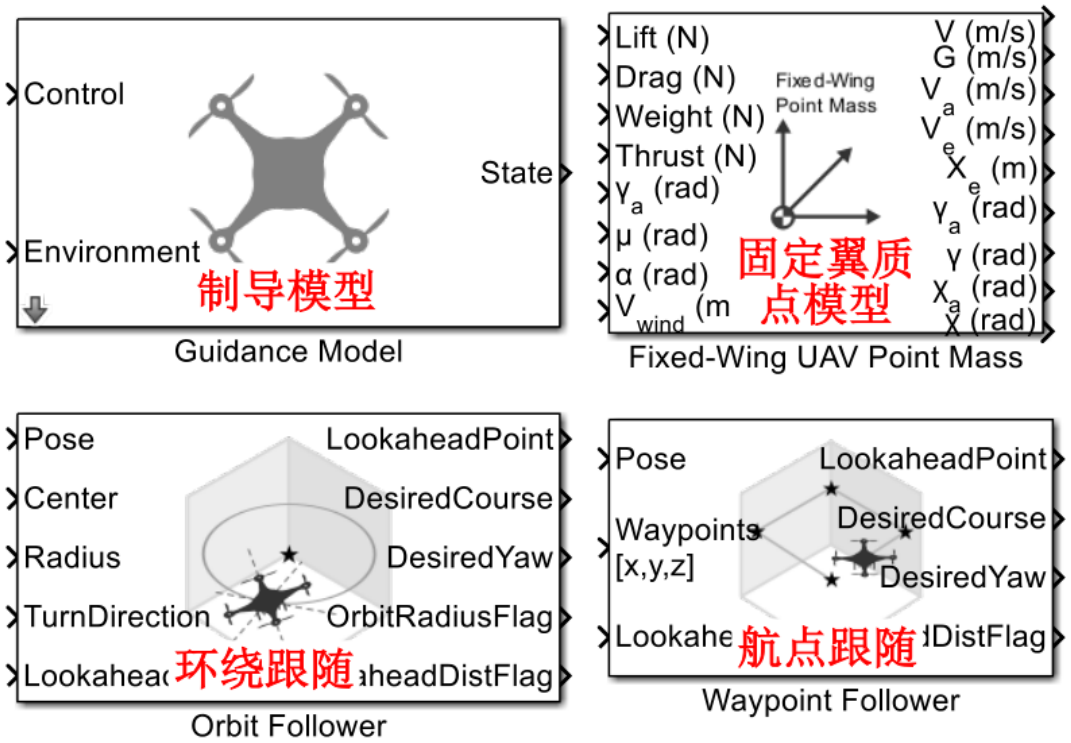


图6.11（左半）模型与路径跟随模块

制导模型：降阶运动学与自驾仪响应，输出状态。

固定翼质点：由力及运动参数计算近似运动。

航点 / 环绕：由位姿和路径生成前视点及航向参考。

6.7.4 扩展接口与算法工具：任务与参考

任务管理与轨迹模块仍需接入下游控制

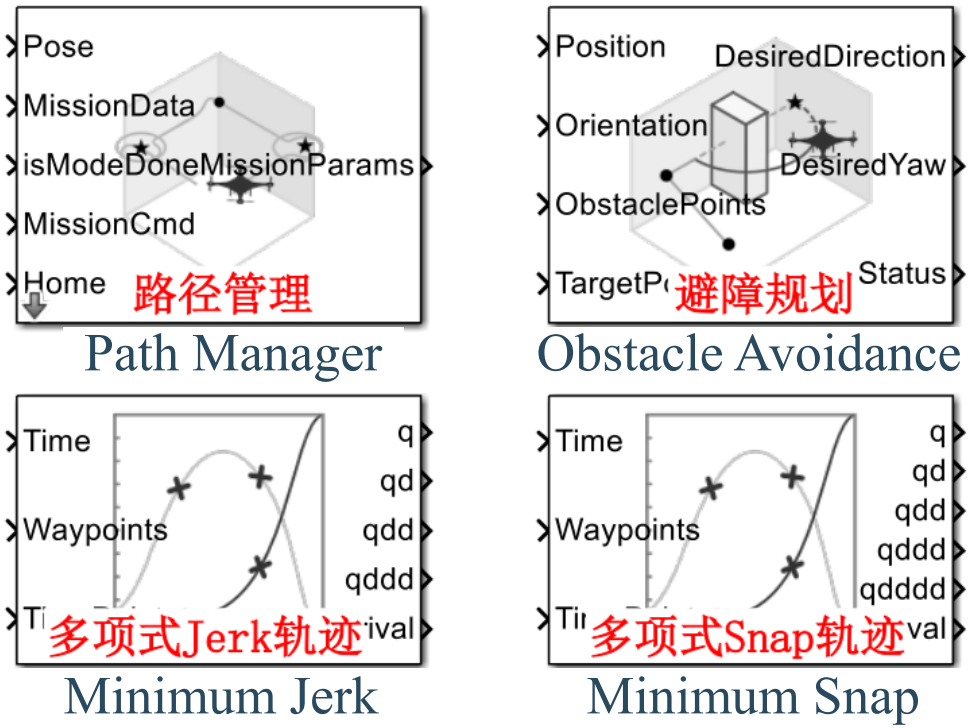


图6.11（右半）任务、避障与轨迹模块

路径管理：任务序列与位姿输入，输出任务参数。

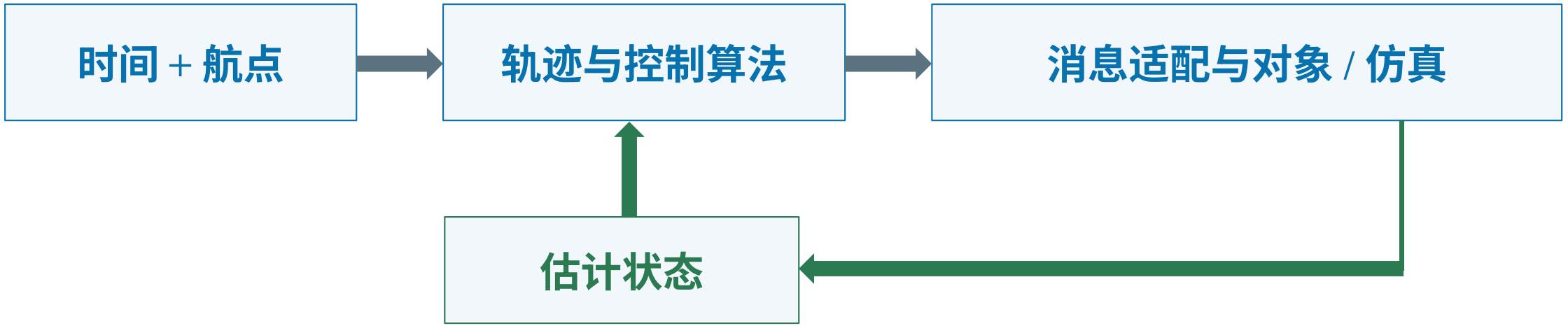
避障：3D VFH+（三维向量场直方图）给方向，需核对运动约束。

多项式：航点与时间生成PVA；q、qd、qdd依次为位置、速度、加速度，q不是四元数。

高阶输出：Jerk模块另输出qddd，Snap模块另输出qddddd。

6.7.4 扩展接口与算法工具：算法搭建

Simulink闭环要明确参考、反馈和输出接口



接口适配：明确坐标系、单位、输入掩码、采样周期与超时处理。

记录：保存参考PVA、反馈状态、实际指令、时间戳和飞行模式。

工具与部署：UAV、自动驾驶、强化学习、MPC工具箱经ROS/UDP接入，可生成代码。

本章目录

平台框架与通信接口

第六章 外部控制与轨迹规划

6.1 外部控制基础与控制分层
6.2 外部控制原理与控制模式
6.3 轨迹规划建模与生成
6.4 轨迹规划算法与避障
6.5 轨迹跟踪原理与高机动控制
6.6 前沿技术与控制方法
6.7 平台框架与通信接口
6.8 外部控制例程与实验验证
6.9 习题与课堂讨论

本节 5 个教学单元

6.7.1 平台框架与职责
教材对应：第6.2.1节

6.7.2 通信链路与协议
教材对应：第6.1.1.5、6.2.2.1节

6.7.3 最小控制与状态获取
教材对应：第6.2.2.2、6.2.2.4节（控制流程）

6.7.4 扩展接口与算法工具
教材：6.2.2.2–6.2.2.3、6.2.2.5、6.2.3节

6.7.5 通信优化与验收
教材对应：第6.2.2.4节（发送频率）、6.2.3节

6.7.5 通信优化与验收：通信频率

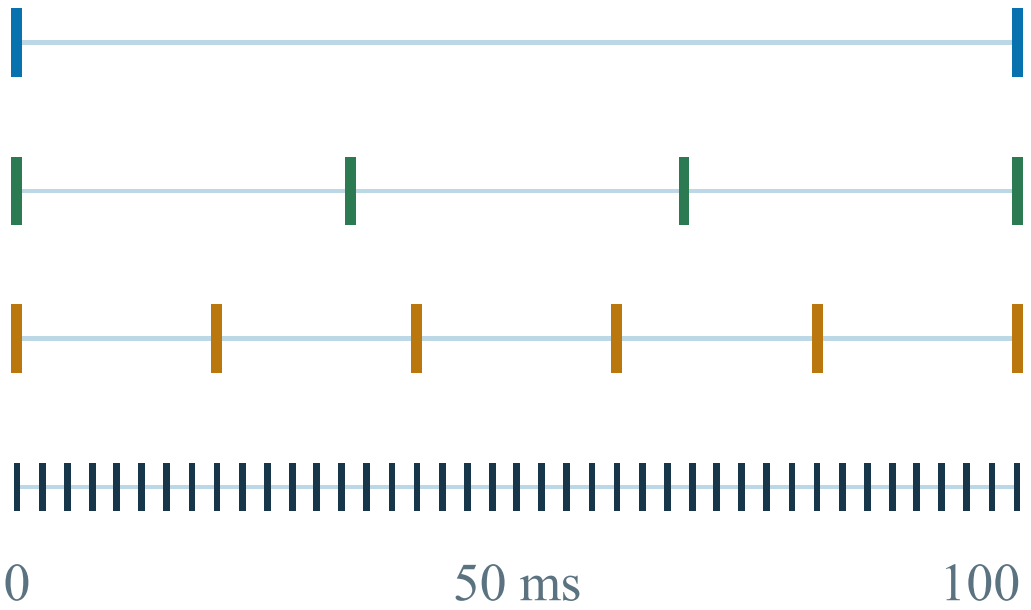
更新率、发送率、接收率和控制内环不是一个量

参考更新率
示意10 Hz

目标发送率
本节取30 Hz

状态接收率
示意50 Hz

飞控内环频率
示意400 Hz



决定：算法何时给新目标
记录：参考版本号与时间

决定：指令流是否持续
记录：发送时刻与最长间隔

决定：反馈是否更新
记录：各类消息接收时刻

决定：底层控制计算节奏
记录：飞控配置与内部日志

图：0–100 ms内四个过程的发生时刻（示意值，非平台实测）

当前SDK源码：Full（完整）发送间隔0.01 s；Simple（精简）为1/30 s。
本节教学显式发送采用30 Hz；实际间隔与抖动仍需记录。

6.7.5 通信优化与验收：数据年龄算例

收到很多帧，也可能只得到重复的旧样本

样本时间 / ms	本机接收时间 / s	状态含义
10000	10.000	首个位置样本
10050	10.050	出现新位置样本
10050	10.100	重复样本；接收率≠更新率

评估时刻取本机10.350 s；接收年龄＝评估时刻−最近一帧的本机接收时刻。

$$A_{\mathrm{recv}} = 10.350 - 10.100 = 0.250\ \mathrm{s}$$

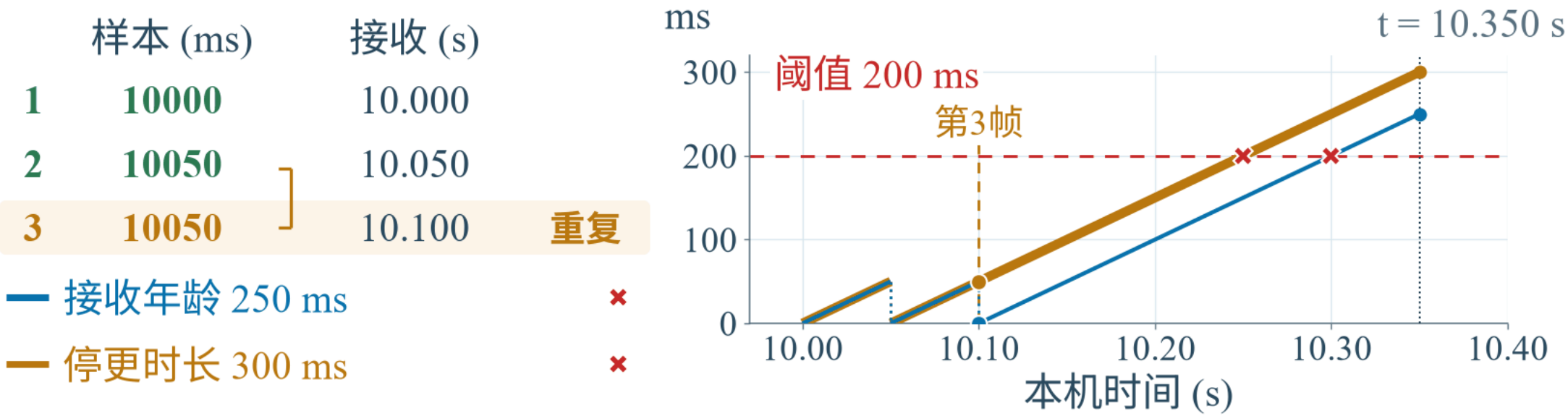
停更时长＝评估时刻−最近新样本的本机接收时刻＝10.350−10.050＝0.300 s。
教学阈值取200 ms：两项都已超限，转入异常处理。

6.7.5 通信优化与验收：重复样本与新鲜度检查

重复帧让接收年龄归零，停更时长却继续增长

第3帧重复；停更比接收年龄早 50 ms 超限

	样本 (ms)	接收 (s)	
1	10000	10.000	
2	10050	10.050	
3	10050	10.100	重复



图：位置帧的样本时间、本机接收时刻与两项检查（离线教学算例，非平台实测）

接收年龄：从最近一帧的本机接收时刻算起，重复帧也会让它归零。

停更时长：从最近新样本的本机接收时刻算起；飞控样本时间只比大小。

6.7.5 通信优化与验收：带宽预算算例

提高发送频率之前，先算链路占用

教学假设：串口115200 baud，8N1；每帧含协议开销共80 B。
8N1：8位数据、无校验、1位停止；加1位起始，每字节占10 bit。

$$\text{单向容量 } C = \frac{115200}{10} = 11520 \text{ B/s}, \quad R = 80f$$

发送频率 f	单方向数据率 R	容量占用
30 Hz	2400 B/s	20.8%
100 Hz	8000 B/s	69.4%

余量：心跳、遥测、参数和突发流量都要计入预算。

优化：下调无关消息或用兼容精简格式；保留目标流，复查误差与空档。

6.7.5 通信优化与验收：模式与数据粒度

仿真模式与数据格式是两个独立选择

仿真模式	闭环中运行什么	主要核对点
PX4_HITL	真实飞控硬件＋运动模型	硬件连接、固件与传感器仿真
PX4_SITL	飞控软件＋运动模型	进程、端口及消息方向
PX4_SITL_RFLY	平台组织的SITL连接	专用端口映射与多实例配置
Simulink／DLL_SIL	控制器＋运动模型DLL	模型接口、参数与真值输出

再选通信格式：MAVLink／平台UDP，Full／Simple，以及传感器数据配置。
模式是否可用与安装版本有关；按当前软件和接口定义核对。

6.7.5 通信优化与验收：连接与控制排错

用最靠近故障的证据逐层排查

现象	先看哪条证据	对应处理
没有心跳	IP、端口、消息格式、来源ID	修复连接配置
有心跳、位置不更新	位置消息时间与接收年龄	检查消息流和估计器
模式切换被拒绝	COMMAND_ACK（命令应答）、状态文本	修复目标流和进入条件
能运动、误差异常	参考与反馈坐标、单位、时序	核对原点、符号与运动

诊断记录：SDK与固件版本、连接配置、样本时间、命令应答、误差和终态。

课堂练习

连接、坐标与时间各检查一个具体条件

问题	参考要点	验收证据
ID=5用哪对端口?	指令20108；状态20109	20100+2×(5−1)；注明收发方
ENU (−1, 4, 2.5)转NED?	(4, −1, −2.5)，单位m	只改符号会得(−1, 4, −2.5)
10.2 s新样本，10.3 s重复； 10.45 s时哪项超0.2 s?	接收年龄0.15 s，未超； 停更0.25 s，已超限	两项分开算， 都用本机时间相减

三个共同要求：对象身份明确、坐标含义明确、证据来自实际状态消息。

课堂练习

程序报“结束”，还缺哪些终态证据

```
send_target()
time.sleep(10)
mav.endOffboard()
# finally: mav.stopRun()
```

左侧流程不完整。
endOffboard不发LAND；
已解锁时stopRun会强制上锁。

到达证据：参考与实际位置的误差在阈值内保持一段时间。

回收证据：明确的降落／接管请求，以及回传模式和着陆状态。

清理条件：已着陆并上锁；状态不明时转入接管流程。

本节小结

接口接通之后，持续目标还要由轨迹生成器给出

职责：外部程序算目标，飞控算底层控制，模型算响应，三维程序负责显示与成像。

分层：链路承载字节，消息协议规定含义，SDK提供调用；端口与坐标都要明确。

闭环：先建立有效指令流，再确认模式与解锁；用真实状态检验误差和终态。

接口：MAVSDK、MAVROS、ROS 2写法各异，验收都看状态与终态。

优化：分别观察生成、发送和接收的时间；减少负载以后仍要复查控制效果。

易错：把目标变量当反馈；把endOffboard当降落；用uavTimeStmp判断位置新鲜；ENU转NED只改符号不交换轴；端口收发方向写反。

边界：本节流程仅限仿真；initOffboard关闭的失联保护，上真机前要恢复。

接口定义怎样传递目标，轨迹规划决定每个时刻应给出什么目标。