

集置核心 GESTELL Core 无人机飞行控制系统 V1.0

软件设计说明书

申请单位：嘉佳科技有限公司

日期：2025 年 11 月 26 日

目录

0.1	适用范围	4
0.2	术语定义	4
1	总体设计	4
1.1	设计原则	4
1.2	系统逻辑架构	5
1.2.1	设备层 (Device Layer)	5
1.2.2	边缘层 (Edge Layer)	5
1.2.3	服务层 (Service Layer)	5
1.2.4	应用层 (Application Layer)	5
1.3	技术栈选型	5
2	详细模块设计 (Backend)	6
2.1	核心类图设计	6
2.1.1	DroneManager	6
2.1.2	MissionPlanner	6
2.2	飞行任务管理模块	7
2.2.1	功能详述	7
2.2.2	任务上传时序	7
2.3	实时遥测模块	7
2.3.1	数据流架构	7
2.3.2	坐标系转换	7
2.4	视觉感知服务	8
2.4.1	视频流处理	8
2.4.2	目标检测算法	8
3	数据库设计	8
3.1	数据库概览	8
3.2	详细表结构设计	8
3.2.1	drones (无人机设备表)	8
3.2.2	missions (飞行任务表)	9
3.2.3	telemetry_history (遥测历史表)	9
4	接口设计	10
4.1	接口规范	10
4.2	核心 API 定义	10
4.2.1	上传任务	10

4.2.2 获取无人机状态	11
4.3 WebSocket 实时推送	11
5 前端设计	11
5.1 界面布局	11
5.2 状态管理	12
6 安全与可靠性设计	12
6.1 安全机制	12
6.2 异常处理与容错	12
7 部署与维护	12
7.1 部署架构	12
7.2 监控运维	12
8 系统测试 (System Testing)	13
8.1 测试策略	13
8.2 功能测试用例	13
9 附录 A: 核心配置文件 (Configuration)	14
9.1 config.yaml (后端配置)	14
9.2 nginx.conf (反向代理配置)	15
10 附录 B: 数据字典 (Data Dictionary)	16
10.1 表: users (用户信息表)	16
10.2 表: flight_plans (飞行计划表)	16
10.3 表: maintenance_logs (维护记录表)	17
11 附录 C: 部署指南 (Deployment Guide)	17
11.1 环境准备	17
11.2 数据库初始化	18
11.3 应用部署	18
11.4 常见问题排查	18
12 附录 D: 错误码字典	19
13 附录 E: 用户操作手册 (User Operation Manual)	19
13.1 系统登录	19
13.2 仪表盘概览	19
13.3 任务规划流程	19

13.3.1 创建新任务	19
13.3.2 任务上传与执行	20
13.4 实时监控与干预	20
13.5 视频流与 AI 识别	20
14 附录 F: 安全与合规 (Security & Compliance)	20
14.1 数据隐私保护	20
14.2 访问控制策略	20
15 附录 G: 未来规划 (Roadmap)	21
15.1 V1.5 版本规划 (2026 Q1)	21
15.2 V2.0 版本规划 (2026 Q3)	21
16 附录 H: 硬件规格书 (Hardware Specifications)	21
16.1 机载计算机 (Companion Computer)	21
16.2 飞控模块 (Flight Controller)	21
17 附录 I: 开源许可声明 (Open Source Licenses)	22
17.1 后端组件	22
17.2 前端组件	22
18 附录 J: 数据库建表脚本 (Database Schema SQL)	23

当前市场上的飞控系统普遍存在以下痛点：首先，**通信延迟高**，在 4G/5G 网络环境下，传统的轮询式架构难以保证毫秒级的实时控制。其次，**算力瓶颈**，机载端缺乏高效的边缘计算能力，难以处理高清视频流的实时 AI 推理。最后，**扩展性差**，紧耦合的单体架构导致新功能（如异构无人机接入）开发周期长、风险高。基于此，嘉佳科技有限公司研发了 GESTELL Core 系统，采用云边端协同架构，实现了毫秒级低时延控制与高精度视觉感知。

0.1 适用范围

本系统适用于以下场景：本系统适用于以下场景：**多机集群调度**，支持同时管理超过 100 架异构无人机的飞行任务；**超视距自主飞行**，基于高精度地图的航点规划与自动避障；**实时安防监控**，基于 YOLOv8 的人员、车辆实时检测与追踪；以及**全生命周期管理**，涵盖无人机健康状态监测、固件升级与维护记录。

0.2 术语定义

术语	定义
MAVLink	Micro Air Vehicle Link，一种轻量级的无人机通信协议，用于传输遥测数据与控制指令。
BVLOS	Beyond Visual Line of Sight，超视距飞行，指无人机在操作员视距范围之外的自主飞行。
RTSP	Real Time Streaming Protocol，实时流传输协议，用于视频流的推拉流控制。
QGC	QGroundControl，一种开源的地面站软件，本系统兼容其标准协议。
Edge Computing	边缘计算，指在无人机机载计算机（如 Jetson 系列）上进行的本地数据处理。
Failsafe	失效保护机制，当系统检测到异常（如链路中断、电量过低）时自动触发的安全策略。

1 总体设计

1.1 设计原则

本系统遵循以下核心设计原则：本系统遵循以下核心设计原则：一是**高内聚低耦合**，采用微服务架构，各功能模块独立部署，通过标准 API 交互；二是**实时优先**，在遥测与控制链路中，优先保证数据的实时性，采用 UDP 与 WebSocket 协议；三是**安全可靠**，

全链路加密传输，具备完善的权限控制与异常熔断机制；四是**可扩展性**，支持通过插件机制接入新型号的传感器与载荷。

1.2 系统逻辑架构

系统自下而上分为四层：设备层、边缘层、服务层与应用层。

1.2.1 设备层 (Device Layer)

包含无人机飞行平台、飞控模块 (PX4/ArduPilot)、各类传感器 (GPS、IMU、激光雷达) 及挂载载荷 (云台相机、投放器)。

1.2.2 边缘层 (Edge Layer)

运行在机载伴侣计算机 (Companion Computer) 上。运行在机载伴侣计算机 (Companion Computer) 上，主要包含以下组件：**MAVSDK Proxy**，负责与飞控进行串口通信，将 MAVLink 消息转换为 gRPC/HTTP 请求；**Video Streamer**，采集相机数据，进行 H.264/H.265 编码并推流；以及 **AI Inference Engine**，部署 YOLOv8n 模型，执行本地目标检测。

1.2.3 服务层 (Service Layer)

部署在云端服务器，基于 Python FastAPI 构建。部署在云端服务器，基于 Python FastAPI 构建，核心服务包括：**Mission Service**，负责航点规划、任务校验与下发；**Telemetry Service**，基于 Redis Pub/Sub 的高并发遥测数据分发；**Auth Service**，处理用户认证、鉴权与会话管理；以及 **Media Service**，负责视频流转发与录像存储管理。

1.2.4 应用层 (Application Layer)

基于 Vue 3 + Vite 构建的 Web 端综合指挥平台，提供电子地图、实时视频墙、仪表盘与任务编辑器。

1.3 技术栈选型

类别	技术/工具	选型理由
后端框架	Python (FastAPI)	高性能异步 IO，原生支持 OpenAPI 文档，适合 IO 密集型任务。
前端框架	Vue 3 + TypeScript	响应式性能优异，类型安全，生态丰富。
数据库	PostgreSQL + PostGIS	强大的关系型数据存储，原生支持地理空间数据查询。

类别	技术/工具	选型理由
缓存/消息队列	Redis	极高的读写性能，支持 Pub/Sub 模式，适合实时遥测数据。
AI 推理	PyTorch + ONNX	广泛的模型支持，ONNX Runtime 提供跨平台加速。
容器化	Docker	保证环境一致性，简化部署流程。

2 详细模块设计 (Backend)

2.1 核心类图设计

系统核心业务逻辑围绕“无人机 (Drone)”与“任务 (Mission)”展开。

2.1.1 DroneManager

负责维护所有在线无人机的状态机。

```
1 class DroneManager:
2     - drones: Dict[str, DroneInstance]
3     - redis_client: Redis
4
5     + register_drone(drone_id: str, connection_info: Dict) -> bool
6     + update_telemetry(drone_id: str, telemetry: TelemetryData) ->
void
7     + send_command(drone_id: str, command: Command) -> CommandResult
8     + get_drone_status(drone_id: str) -> DroneStatus
9     + check_heartbeat() -> void
```

2.1.2 MissionPlanner

负责任务的解析、校验与优化。

```
1 class MissionPlanner:
2     - geo_fences: List[Polygon]
3
4     + validate_mission(mission: Mission) -> ValidationResult
5     + optimize_path(waypoints: List[Waypoint]) -> List[Waypoint]
6     + estimate_flight_time(mission: Mission, speed: float) -> float
7     + check_terrain_collision(waypoints: List[Waypoint]) -> bool
```

2.2 飞行任务管理模块

2.2.1 功能详述

任务管理模块是系统的核心指挥中枢,支持航点飞行(Waypoint)、环绕飞行(Orbit)、跟随飞行(Follow Me)等多种模式。

2.2.2 任务上传时序

任务上传与执行流程如下:首先,前端提交包含航点列表的 JSON 数据至 `/api/missions/upload`。随后,后端 `MissionService` 接收请求,调用 `MissionPlanner.validate_mission` 进行地理围栏与参数校验。校验通过后,将任务数据持久化至 PostgreSQL `missions` 表,状态置为 `PENDING`。当用户发送“执行”指令时,后端通过 MAVSDK 将航点上传至指定无人机。无人机确认接收(ACK)后,后端更新任务状态为 `UPLOADED`。最后,后端发送 `MISSION_START` 指令,无人机解锁并起飞,状态更新为 `EXECUTING`。

2.3 实时遥测模块

2.3.1 数据流架构

遥测数据(位置、姿态、电量)由无人机以 10Hz-50Hz 的频率发送。遥测数据流转分为三条链路:上行链路(Drone -> MQTT/gRPC -> Telemetry Service),负责数据的实时采集;处理链路,由 Telemetry Service 解析数据包,存入 Redis 缓存(设置 5 秒 TTL),并异步写入 PostgreSQL(降采样存储,用于历史回放);以及下行链路(Telemetry Service -> WebSocket -> Frontend Client),实现前端的实时展示。

2.3.2 坐标系转换

系统内部统一使用 WGS84 坐标系(经纬度),但在前端地图展示与距离计算时,需转换为 Web 墨卡托投影或 ENU 局部坐标系。核心转换公式(Haversine):

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_1 \cdot \cos\phi_2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right) \quad (1)$$

$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a}) \quad (2)$$

$$d = R \cdot c \quad (3)$$

其中 ϕ 为纬度, λ 为经度, R 为地球半径。

2.4 视觉感知服务

2.4.1 视频流处理

采用 ‘OpenCV’ 读取 RTSP 流，通过 ‘multiprocessing’ 实现帧解码与推理的并行处理，确保视频延迟低于 200ms。

2.4.2 目标检测算法

集成 YOLOv8 模型，针对俯视视角下的车辆与行人进行微调训练。推理流程：推理流程包含四个步骤：首先进行**预处理**，将图像缩放至 640x640，归一化并进行通道变换 (HWC -> CHW)；接着进行**推理**，使用 ONNX Runtime 执行前向传播；然后是**后处理**，采用 NMS（非极大值抑制）过滤冗余框，置信度阈值设为 0.5；最后进行**地理定位**，结合无人机当前的 GPS 与姿态角（Pitch, Roll, Yaw），利用单目测距算法估算目标的地面坐标。

3 数据库设计

3.1 数据库概览

采用 PostgreSQL 14 作为主数据库，启用 PostGIS 插件以支持空间数据类型。数据库命名规范：

- 表名：小写下划线复数（如 ‘flight_logs’
- 主键：‘id’ (BigSerial) 或 ‘uuid’。
- 字段：小写下划线（如 ‘created_at’

3.2 详细表结构设计

3.2.1 drones (无人机设备表)

存储无人机的静态配置信息。

字段名	类型	约束	说明
id	INTEGER	PK	自增主键
serial_number	VARCHAR(50)	UNIQUE	设备序列号
name	VARCHAR(100)	NOT NULL	设备自定义名称
model_type	VARCHAR(50)		机型（如 Quadcopter, VTOL）

字段名	类型	约束	说明
firmware_ver	VARCHAR(20)		固件版本
registered_at	TIMESTAMP		注册时间
last_online	TIMESTAMP		最后在线时间
is_active	BOOLEAN	DEFAULT T	激活状态

3.2.2 missions (飞行任务表)

存储任务的规划数据与执行状态。

字段名	类型	约束	说明
id	INTEGER	PK	任务 ID
drone_id	INTEGER	FK	关联无人机
creator_id	INTEGER	FK	创建用户
name	VARCHAR(100)		任务名称
waypoints	JSONB	NOT NULL	航点数据（GeoJSON 格式）
parameters	JSONB		任务参数（速度、高度、动作）
status	VARCHAR(20)		PENDING, EXECUTING, DONE, FAILED
start_time	TIMESTAMP		实际开始时间
end_time	TIMESTAMP		实际结束时间

3.2.3 telemetry_history (遥测历史表)

用于存储飞行轨迹回放，采用 TimescaleDB 超表技术优化时序写入。

字段名	类型	约束	说明
time	TIMESTAMP	NOT NULL	时间戳（分区键）
drone_id	INTEGER	FK	无人机 ID
latitude	DOUBLE		纬度
longitude	DOUBLE		经度
altitude	FLOAT		相对高度
speed	FLOAT		地速
battery	INTEGER		电量百分比
flight_mode	VARCHAR(20)		飞行模式

4 接口设计

4.1 接口规范

接口遵循以下规范：采用 HTTP/1.1 (RESTful) 协议，数据格式为 JSON，字符编码统一使用 UTF-8，并采用 Bearer Token (JWT) 进行鉴权。

4.2 核心 API 定义

4.2.1 上传任务

URL: /api/v1/missions

Method: POST

Request Body:

```
1 {  
2   "drone_id": 101,  
3   "name": "PowerLine_Inspection_05",  
4   "waypoints": [  
5     {  
6       "lat": 34.234567,  
7       "lon": 108.987654,  
8       "alt": 30.0,  
9       "speed": 5.0,  
10      "action": "HOVER",  
11      "param": 5  
12    },  
13    ...  
14  ],  
15  "rtl_on_finish": true  
16 }
```

Response (201 Created):

```
1 {  
2   "code": 20000,  
3   "message": "Mission created successfully",  
4   "data": {  
5     "mission_id": 5023,  
6     "estimated_duration": 1240,  
7     "total_distance": 3500.5  
8   }  
}
```

9

}

4.2.2 获取无人机状态

URL: /api/v1/drones/id/status

Method: GET

Response:

1 {

2 "code": 20000,

3 "data": {

4 "online": true,

5 "mode": "MISSION",

6 "armed": true,

7 "position": { "lat": 34.23, "lon": 108.98, "alt": 50.2 },

8 "battery": { "voltage": 22.4, "level": 78 },

9 "gps": { "satellites": 14, "hdop": 0.8 }

10 }

11 }

4.3 WebSocket 实时推送

Endpoint: /ws/stream

Events:

Event Type	Payload Description
TELEMETRY_UPDATE	包含无人机实时位置、姿态、速度的高频数据包。
MISSION_PROGRESS	当前航点索引、剩余距离、预计剩余时间。
ALERT	系统告警（低电量、链路信号弱、GPS 丢失）。
AI_DETECTION	视觉识别结果（目标类型、置信度、图像坐标）。

5 前端设计

5.1 界面布局

前端采用现代化的 Dashboard 布局，分为顶部导航栏、左侧资源树、中间地图主视区、右侧信息面板与底部状态栏。核心组件包括：**地图组件**，基于 Mapbox GL JS 或 Leaflet，支持卫星图/矢量图切换及加载 GeoJSON 禁飞区图层；以及**视频组件**，使用 WebRTC 或 FLV.js 播放低延迟视频流，支持在视频层上绘制 AI 识别框（Canvas Overlay）。

5.2 状态管理

利用 Pinia 进行模块化状态管理：利用 Pinia 进行模块化状态管理，主要包括：**Dron-eStore**，维护所有无人机的即时状态字典，通过 WebSocket 消息驱动更新；**Mission-Store**，管理当前编辑的任务数据，支持撤销/重做（Undo/Redo）；以及 **UserStore**，管理用户信息与权限列表。

6 安全与可靠性设计

6.1 安全机制

为确保系统安全，采取了多重防护措施：**通信加密**方面，所有 HTTP 流量强制使用 HTTPS (TLS 1.2+)，MAVLink 链路启用签名校验；**身份认证**采用 JWT (JSON Web Token) 进行无状态认证，Token 有效期 2 小时，并支持 Refresh Token 刷新；同时实施**操作审计**，所有关键指令（如解锁、起飞、修改参数）均记录审计日志，包含操作人 IP、时间与指令内容。

6.2 异常处理与容错

系统具备完善的异常处理机制：**断连保护**功能在 3 秒未收到心跳时，自动标记无人机为“失联”，并触发地面站声光报警；**地理围栏**机制在机载端与服务端进行双重校验，严防飞入禁飞区；此外，引入**服务熔断**策略，当后端服务负载过高时，优先保证遥测数据的接收，暂时拒绝非紧急的历史数据查询请求。

7 部署与维护

7.1 部署架构

推荐采用 Docker Compose 或 Kubernetes 进行容器化部署。推荐采用 Docker Compose 或 Kubernetes 进行容器化部署，架构组件包括：**Nginx**，负责反向代理与负载均衡，处理 SSL 卸载；**App Server**，进行多副本部署的 FastAPI 服务；**Worker**，由 Celery Worker 处理视频转码与日志归档等耗时任务；以及 **Database**，采用 PostgreSQL 主从复制与 Redis 集群模式。

7.2 监控运维

集成 Prometheus + Grafana 监控体系。集成 Prometheus + Grafana 监控体系，涵盖以下维度：**系统监控**（CPU、内存、磁盘 I/O、网络带宽）；**业务监控**（在线无人机

数量、任务成功率、API 响应时间、WebSocket 连接数)；并利用日志聚合，使用 ELK (Elasticsearch, Logstash, Kibana) 栈收集与分析分布式日志。

8 系统测试 (System Testing)

8.1 测试策略

系统测试分为单元测试、集成测试与验收测试三个阶段。系统测试分为三个阶段：**单元测试**，针对 MissionService 与 GeoMath 核心算法进行覆盖率测试，要求覆盖率 > 90%；**集成测试**，模拟 MAVLink 数据流，验证 Telemetry Service 的高并发处理能力；以及**验收测试**，在仿真环境（SITL）中进行全流程飞行演练。

8.2 功能测试用例

ID	测试项目	前置条件	预期结果	结果
TC-001	用户登录	数据库存在有效用户	返回 Token	通过
TC-002	密码错误登录	数据库存在有效用户	返回 401 错误	通过
TC-003	上传空任务	无人机在线	返回"Empty Mission"	通过
TC-004	越界航点上传	设置禁飞区	返回"No-Fly Zone"	通过
TC-005	任务正常执行	无人机解锁	状态变更为 EX-ECUTING	通过
TC-006	暂停任务	任务执行中	无人机悬停 (HOLD)	通过
TC-007	恢复任务	任务暂停中	继续飞向下一航点	通过
TC-008	返航触发	任务执行中	爬升至 RTL 高度并返航	通过
TC-009	视频流加载	推流服务正常	延迟 < 200ms	通过
TC-010	AI 车辆识别	视频中出现车辆	绘制 Bounding Box	通过
TC-011	AI 行人识别	视频中出现行人	绘制 Bounding Box	通过
TC-012	遥测数据中断	拔掉数传	3 秒后触发断连告警	通过

ID	测试项目	前置条件	预期结果	结果
TC-013	低电量返航	电量 < 20%	自动触发 RTL	通过
TC-014	多机并发控制	5 架无人机在线	所有指令无阻塞 下发	通过
TC-015	历史轨迹查询	存在历史飞行记录	返回 GeoJSON 路径	通过
TC-016	固件升级	上传.px4 文件	升级成功并重启	通过
TC-017	参数修改	修改最大速度	读取参数确认生效	通过
TC-018	账户权限控制	普通用户登录	无法删除任务	通过
TC-019	数据库备份	数据库运行中	生成.sql 备份文件	通过
TC-020	服务重启恢复	重启后端服务	无人机自动重连	通过

9 附录 A: 核心配置文件 (Configuration)

9.1 config.yaml (后端配置)

```
1 server:
2   host: "0.0.0.0"
3   port: 8000
4   workers: 4
5   debug: false
6
7 database:
8   url: "postgresql://user:pass@localhost:5432/drone_db"
9   pool_size: 20
10  max_overflow: 10
11
12 redis:
13   url: "redis://localhost:6379/0"
14   telemetry_channel: "drone/telemetry"
15
16 mavlink:
17   connection_string: "udp://:14550"
18   system_id: 255
19   component_id: 1
20   timeout: 5.0
```

```
21
22 security:
23   jwt_secret: "super_secret_key_change_me"
24   algorithm: "HS256"
25   access_token_expire_minutes: 120
```

9.2 nginx.conf (反向代理配置)

```
1 user  nginx;
2 worker_processes  auto;
3
4 events {
5     worker_connections  1024;
6 }
7
8 http {
9     include      /etc/nginx/mime.types;
10    default_type  application/octet-stream;
11
12    upstream backend_api {
13        server app:8000;
14    }
15
16    server {
17        listen 80;
18        server_name drone-control.futurewing.com;
19
20        location / {
21            root    /usr/share/nginx/html;
22            index  index.html index.htm;
23            try_files $uri $uri/ /index.html;
24        }
25
26        location /api/ {
27            proxy_pass http://backend_api;
28            proxy_set_header Host $host;
29            proxy_set_header X-Real-IP $remote_addr;
30        }
31
```

```
32     location /ws/ {
33         proxy_pass http://backend_api;
34         proxy_http_version 1.1;
35         proxy_set_header Upgrade $http_upgrade;
36         proxy_set_header Connection "upgrade";
37     }
38 }
39 }
```

10 附录 B: 数据字典 (Data Dictionary)

10.1 表: users (用户信息表)

字段名	数据类型	必填	说明
user_id	SERIAL	Y	用户唯一标识
username	VARCHAR(50)	Y	登录用户名, 唯一索引
password_hash	VARCHAR(255)	Y	PBKDF2 加密后的密码散列
email	VARCHAR(100)	N	用户邮箱
phone	VARCHAR(20)	N	联系电话
role	VARCHAR(20)	Y	角色: ADMIN, PILOT, VIEWER
created_at	TIMESTAMP	Y	账户创建时间
last_login	TIMESTAMP	N	最后登录时间
is_active	BOOLEAN	Y	账户是否启用
avatar_url	VARCHAR(255)	N	头像图片地址
department	VARCHAR(100)	N	所属部门

10.2 表: flight_plans (飞行计划表)

字段名	数据类型	必填	说明
plan_id	SERIAL	Y	计划唯一标识
name	VARCHAR(100)	Y	计划名称
description	TEXT	N	任务描述
area_polygon	GEOMETRY	Y	任务区域多边形 (PostGIS)
min_altitude	FLOAT	Y	最低飞行高度 (米)
max_altitude	FLOAT	Y	最高飞行高度 (米)

字段名	数据类型	必填	说明
start_time	TIMESTAMP	Y	计划开始时间
end_time	TIMESTAMP	Y	计划结束时间
approval_status	VARCHAR(20)	Y	审 批 状 态: DRAFT, AP- PROVED, REJECTED
approver_id	INTEGER	N	审批人 ID

10.3 表: maintenance_logs (维护记录表)

字段名	数据类型	必填	说明
log_id	SERIAL	Y	记录 ID
drone_id	INTEGER	Y	关联无人机 ID
maintainer_id	INTEGER	Y	维护人员 ID
log_date	DATE	Y	维护日期
type	VARCHAR(50)	Y	维护类型: ROUTINE, REPAIR, REPLACE
description	TEXT	Y	维护内容详细描述
parts_replaced	JSONB	N	更换零件列表
cost	DECIMAL(10,2)	N	维护费用
next_due_date	DATE	N	下次建议维护日期

11 附录 C: 部署指南 (Deployment Guide)

11.1 环境准备

本系统推荐运行在 Ubuntu 22.04 LTS 操作系统上。

```
1 # 更新系统包
2 sudo apt update && sudo apt upgrade -y
3
4 # 安装基础依赖
5 sudo apt install -y curl git build-essential
6
7 # 安装 Docker
8 curl -fsSL https://get.docker.com -o get-docker.sh
9 sudo sh get-docker.sh
10 sudo usermod -aG docker $USER
11
```

```
12 # 安装 Docker Compose
13 sudo apt install -y docker-compose-plugin
```

11.2 数据库初始化

```
1 -- 创建数据库用户
2 CREATE USER drone_user WITH PASSWORD 'secure_password';
3
4 -- 创建数据库
5 CREATE DATABASE drone_db OWNER drone_user;
6
7 -- 启用 PostGIS 扩展
8 \c drone_db
9 CREATE EXTENSION postgis;
10 CREATE EXTENSION timescaledb;
```

11.3 应用部署

```
1 # 克隆代码仓库
2 git clone https://github.com/futurewing/drone-control-core.git
3 cd drone-control-core
4
5 # 构建镜像
6 docker compose build
7
8 # 启动服务
9 docker compose up -d
10
11 # 查看日志
12 docker compose logs -f
```

11.4 常见问题排查

常见问题及排查方法如下：若出现 **MAVLink 连接超时**，请检查防火墙是否放行 UDP 14550 端口，并确认无人机 IP 地址配置正确；若**视频流无法播放**，请确认 RTSP 地址在 VLC 中可播放，并检查 Nginx 是否开启了 WebSocket 支持。

12 附录 D: 错误码字典

错误码	错误信息	解决方案
10001	System Error	联系管理员查看后台日志
20001	Auth Failed	检查 Token 是否过期
20002	Permission Denied	当前用户无权执行此操作
30001	Drone Offline	检查无人机电源与数传链路
30002	Mission Invalid	检查航点是否在禁飞区内
30003	Upload Timeout	重试上传或检查网络状况

13 附录 E: 用户操作手册 (User Operation Manual)

13.1 系统登录

系统登录步骤如下:首先,打开浏览器,访问系统地址 <http://drone-control.futurewing.com>;接着,在登录页面输入用户名与密码(默认管理员账号: admin / admin123);然后,点击“登录”按钮,系统将校验凭证并跳转至仪表盘首页;若忘记密码,请联系系统管理员重置。

13.2 仪表盘概览

登录成功后,用户将看到主仪表盘,包含以下区域:登录成功后,用户将看到主仪表盘,包含以下区域:顶部状态栏,显示当前在线无人机数量、告警信息及当前用户头像;左侧导航栏,提供“实时监控”、“任务规划”、“历史回放”、“系统设置”等功能入口;中央地图区,实时显示所有在线无人机的位置、航向及飞行轨迹;以及右侧详情板,选中某架无人机后,显示其详细遥测数据(电压、卫星数、高度、速度)。

13.3 任务规划流程

13.3.1 创建新任务

创建新任务的步骤为:首先,点击左侧导航栏的“任务规划”图标;其次,在地图上点击鼠标左键添加航点(Waypoint);接着,在右侧属性面板中设置每个航点的高度(默认 30 米)、悬停时间及动作(如拍照、投掷);最后,点击“保存任务”,输入任务名称(如“A 区巡检₂₀₂₅₁₁₂₆”

13.3.2 任务上传与执行

任务上传与执行的操作流程是：在任务列表中选择已保存的任务，点击“上传”按钮，系统将自动校验航点合法性（是否穿越禁飞区）；校验通过后，点击“执行”按钮；确认弹出的安全提示框，无人机将自动解锁并起飞。

13.4 实时监控与干预

在任务执行过程中，操作员可随时进行人工干预：在任务执行过程中，操作员可随时进行人工干预：点击“暂停”按钮，无人机将立即悬停在当前位置；点击“继续”按钮，无人机将飞向下一航点；点击“RTL”按钮，无人机将升高至安全返航高度（默认 50 米）并直线返回起飞点；遇到严重故障时，点击“紧急降落”，无人机将原地强制降落（慎用）。

13.5 视频流与 AI 识别

视频流与 AI 识别功能的使用方法如下：在右下角视频窗口中，可查看当前选中无人机的实时第一人称视角（FPV）；开启“AI 识别”开关，系统将自动在视频画面上框选识别到的车辆（蓝色框）与行人（红色框）；点击识别框，可查看目标的估算经纬度坐标。

14 附录 F：安全与合规 (Security & Compliance)

14.1 数据隐私保护

本系统严格遵循《个人信息保护法》及 GDPR 相关规定：本系统严格遵循《个人信息保护法》及 GDPR 相关规定：实行**数据脱敏**，在日志导出时，自动对用户手机号、邮箱进行掩码处理；启用**存储加密**，数据库磁盘采用 AES-256 透明加密；并强制**传输加密**，全链路使用 HTTPS，禁止明文 HTTP 访问。

14.2 访问控制策略

系统实施基于角色的访问控制 (RBAC)：

角色	权限描述
超级管理员	拥有所有权限，包括用户管理、系统配置修改。
飞行员	仅可创建任务、执行飞行、查看遥测数据。

角色	权限描述
观察员	仅可查看实时监控画面与历史记录，不可控制无人机。

15 附录 G: 未来规划 (Roadmap)

15.1 V1.5 版本规划 (2026 Q1)

V1.5 版本规划 (2026 Q1) 包括: 支持 5G 切片网络, 进一步降低端到端延迟至 20ms 以内; 集成红外热成像, 支持双光吊舱, 应用于夜间搜救与电力测温; 以及三维路径规划, 基于 3D 城市模型的避障算法。

15.2 V2.0 版本规划 (2026 Q3)

V2.0 版本规划 (2026 Q3) 将实现: 全自主集群编队, 支持 50 架以上无人机的动态编队飞行与灯光秀控制; 数字孪生, 在 Web 端构建 1:1 的高保真三维仿真环境; 以及区块链存证, 将关键飞行日志上链, 确保数据不可篡改, 用于事故定责。

16 附录 H: 硬件规格书 (Hardware Specifications)

16.1 机载计算机 (Companion Computer)

参数项	规格指标
处理器	NVIDIA Jetson Orin Nano (6-core ARM Cortex-A78AE)
GPU	1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores
内存	8GB 128-bit LPDDR5
存储	512GB NVMe SSD
接口	2x USB 3.2 Gen 2, 1x Gigabit Ethernet, 1x HDMI 2.1
功耗	7W - 15W
操作系统	Ubuntu 20.04 LTS (JetPack 5.1)

16.2 飞控模块 (Flight Controller)

参数项	规格指标
主控芯片	STM32H753 (480MHz)
协处理器	STM32F103 (IO Processor)
IMU	3x ICM-42688-P (Vibration Isolated)
气压计	2x BMP388
磁罗盘	RM3100
PWM 输出	16 channels
通信接口	5x UART, 3x I2C, 2x CAN FD

17 附录 I: 开源许可声明 (Open Source Licenses)

本软件使用了以下开源组件，特此声明：

17.1 后端组件

组件名称	版本	许可证
FastAPI	0.95.0	MIT License
Uvicorn	0.21.1	BSD-3-Clause
SQLAlchemy	2.0.7	MIT License
Pydantic	1.10.7	MIT License
Celery	5.2.7	BSD-3-Clause
Redis-py	4.5.4	MIT License
PyJWT	2.6.0	MIT License
MAVSDK-Python	1.4.1	BSD-3-Clause
NumPy	1.24.2	BSD-3-Clause
OpenCV-Python	4.7.0	MIT License
Ultralytics YOLO	8.0.50	AGPL-3.0

17.2 前端组件

组件名称	版本	许可证
Vue.js	3.2.47	MIT License
Vite	4.2.1	MIT License
Pinia	2.0.33	MIT License
Vue Router	4.1.6	MIT License

组件名称	版本	许可证
Axios	1.3.4	MIT License
Element Plus	2.3.1	MIT License
ECharts	5.4.1	Apache-2.0
Mapbox GL JS	2.13.0	BSD-3-Clause
Socket.io-client	4.6.1	MIT License
Lodash	4.17.21	MIT License

18 附录 J: 数据库建表脚本 (Database Schema SQL)

```
1  -- Enable PostGIS
2  CREATE EXTENSION IF NOT EXISTS postgis;
3  CREATE EXTENSION IF NOT EXISTS timescaledb;
4
5  -- Users Table
6  CREATE TABLE users (
7      user_id SERIAL PRIMARY KEY,
8      username VARCHAR(50) UNIQUE NOT NULL,
9      password_hash VARCHAR(255) NOT NULL,
10     email VARCHAR(100),
11     phone VARCHAR(20),
12     role VARCHAR(20) DEFAULT 'VIEWER',
13     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
14     last_login TIMESTAMP,
15     is_active BOOLEAN DEFAULT TRUE,
16     avatar_url VARCHAR(255),
17     department VARCHAR(100)
18 );
19
20 CREATE INDEX idx_users_username ON users(username);
21 CREATE INDEX idx_users_email ON users(email);
22
23 -- Drones Table
24 CREATE TABLE drones (
25     id SERIAL PRIMARY KEY,
26     serial_number VARCHAR(50) UNIQUE NOT NULL,
27     name VARCHAR(100) NOT NULL,
28     model_type VARCHAR(50),
```

```
29     firmware_ver VARCHAR(20),
30     registered_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
31     last_online TIMESTAMP,
32     is_active BOOLEAN DEFAULT TRUE
33 );
34
35 CREATE INDEX idx_drones_serial ON drones(serial_number);
36
37 -- Missions Table
38 CREATE TABLE missions (
39     id SERIAL PRIMARY KEY,
40     drone_id INTEGER REFERENCES drones(id),
41     creator_id INTEGER REFERENCES users(user_id),
42     name VARCHAR(100),
43     waypoints JSONB NOT NULL,
44     parameters JSONB,
45     status VARCHAR(20) DEFAULT 'PENDING',
46     start_time TIMESTAMP,
47     end_time TIMESTAMP,
48     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
49 );
50
51 CREATE INDEX idx_missions_drone ON missions(drone_id);
52 CREATE INDEX idx_missions_status ON missions(status);
53 CREATE INDEX idx_missions_created ON missions(created_at);
54
55 -- Flight Logs (Hypertable)
56 CREATE TABLE flight_logs (
57     time TIMESTAMP NOT NULL,
58     drone_id INTEGER REFERENCES drones(id),
59     latitude DOUBLE PRECISION,
60     longitude DOUBLE PRECISION,
61     altitude REAL,
62     speed REAL,
63     battery INTEGER,
64     flight_mode VARCHAR(20),
65     cpu_usage INTEGER,
66     ram_usage INTEGER
67 );
68
```

```
69 SELECT create_hypertable('flight_logs', 'time');
70
71 CREATE INDEX idx_flight_logs_drone_time ON flight_logs(drone_id, time
    DESC);
72
73 -- Maintenance Logs
74 CREATE TABLE maintenance_logs (
75     log_id SERIAL PRIMARY KEY,
76     drone_id INTEGER REFERENCES drones(id),
77     maintainer_id INTEGER REFERENCES users(user_id),
78     log_date DATE NOT NULL,
79     type VARCHAR(50) NOT NULL,
80     description TEXT,
81     parts_replaced JSONB,
82     cost DECIMAL(10,2),
83     next_due_date DATE
84 );
85
86 CREATE INDEX idx_maint_drone ON maintenance_logs(drone_id);
87
88 -- No Fly Zones
89 CREATE TABLE no_fly_zones (
90     zone_id SERIAL PRIMARY KEY,
91     name VARCHAR(100),
92     polygon GEOMETRY(POLYGON, 4326),
93     min_altitude REAL,
94     max_altitude REAL,
95     is_active BOOLEAN DEFAULT TRUE
96 );
97
98 CREATE INDEX idx_nfz_geom ON no_fly_zones USING GIST(polygon);
```