**概要设计说明书**

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [ ] 正式发布  [ ] 正在修改 | 文件标识： |  |
| 当前版本： | 2.00 |
| 作 者： | 水质检测小组 |
| 完成日期： | 2024/07/17 |

昊天云智

2024年 07月 17日

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 一、文档介绍

## 文档目的

项目概要设计说明

## 文档范围

本文档包括但不限于项目的架构设计、数据存储设计、关键业务流程实现设计、软件功能设计、重要接口设计、系统部署设计。

## 读者对象

参与项目开发的学生，参与项目评测的教师团队，其他想要了解本项目的读者。

## 术语与缩写解释

|  |  |
| --- | --- |
| **缩写、术语** | **解 释** |
|  |  |
|  |  |
|  |  |
|  |  |

# 二、项目架构设计

## 总体系统架构

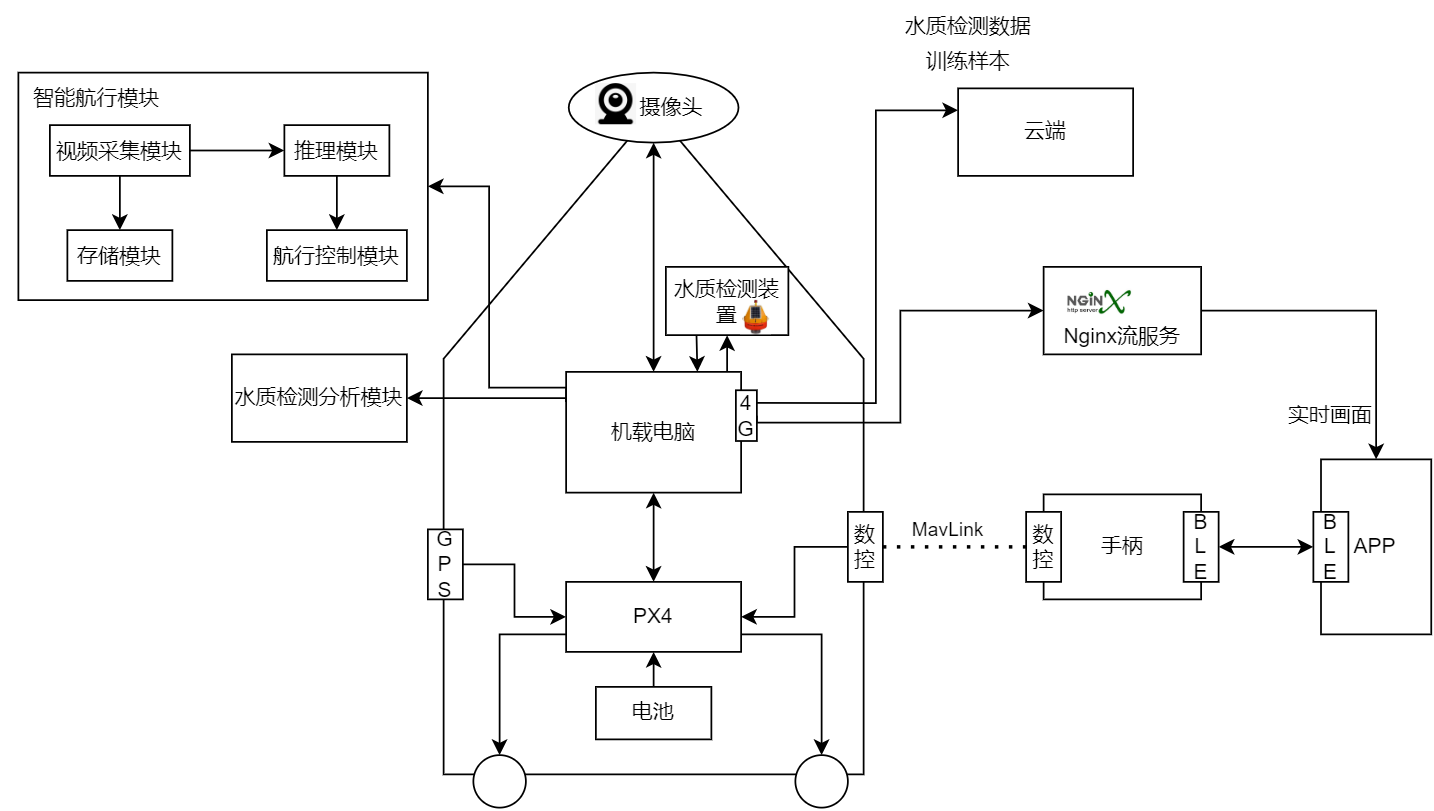


图1 总体系统架构图

1. APP控制无人艇启动模块：APP通过手柄蓝牙连接，手柄将蓝牙信号转换为数控信号，通过MavLink将信号传输到PX4中，通过飞控协议驱动PX4。
2. 飞手\操手航行指令执行：APP通过蓝牙连接机载电脑，飞手在APP上通过GPS地图服务设定任务点及其坐标，APP根据河海情况设计一条路线避免上岸搁浅，通过蓝牙信号将路线各参数发送给机载电脑，机载电脑接收到任务后按顺序选择任务提取目标点，根据自带GPS定位目标点坐标参数以及自身坐标计算出转向角，将参数发送至PX4，驱动PX4航行。
3. 智能航行模块：机载电脑开始执行任务时开始根据预设路线行进，同时摄像头采集的图像转换为数据流传输到机载电脑中的推理模块， 使用模型预测识别障碍物，识别到障碍物根据船的方向和障碍物的尺寸比例计算两边转向角，根据转向角最小原则将参数发给PX4，无人艇改变航行方向。
4. 手动航行模块：手柄通过遥感将航行信息参数通过数传传到PX4，PX4根据航行参数调整方向速度等。
5. 水质检测模块：APP通过蓝牙信号发送水质检测指令给机载电脑，机载电脑直接调用水质检测装置，水质检测装置进行取水样检测操作，并将传感器等检测配件信号转换为数据流传输到机载电脑，机载电脑对数据进行整合打包，通过蓝牙信号传输数据到APP端或通过4g信号直接传输云端。
6. 实时画面传输：摄像头采集的图像转换为数据流传输到机载电脑，机载电脑在有4g环境的前提下通过Nginx服务和流媒体服务推送到APP，APP将数据流转换为画面。

## 总体系统模块关系

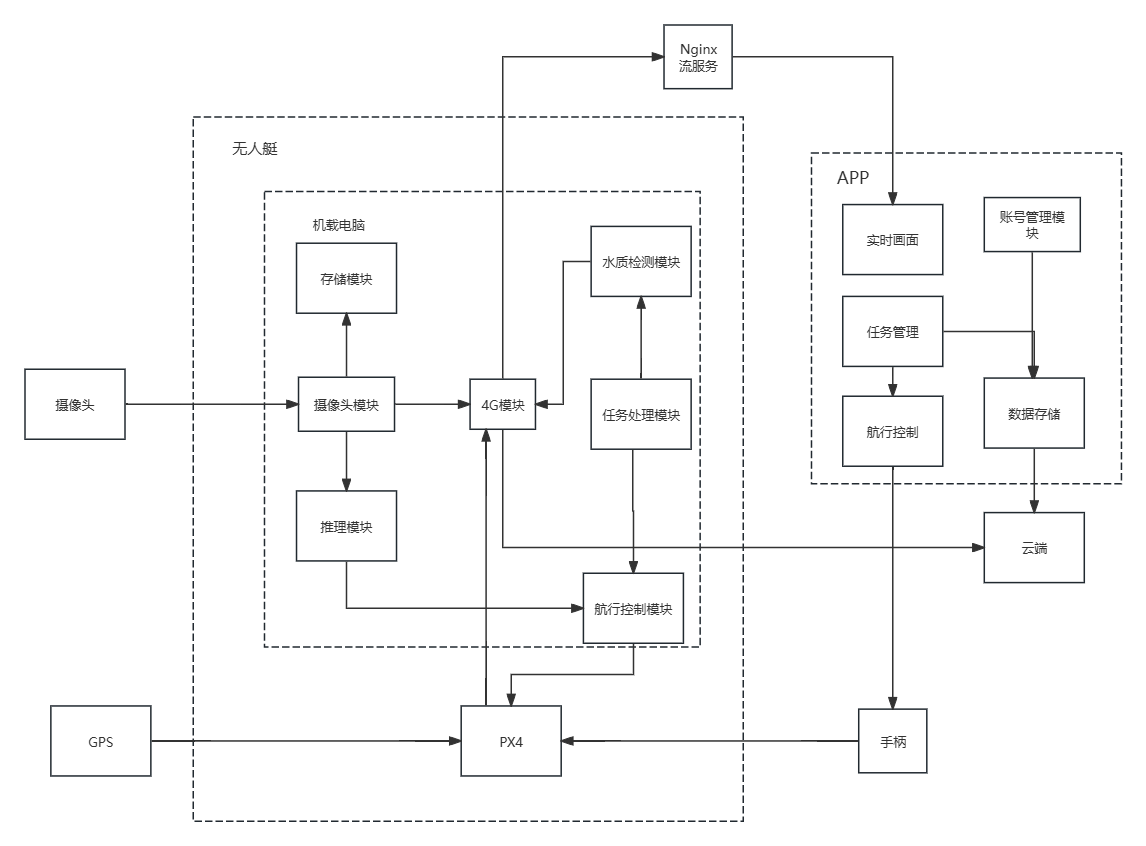


图2.1 系统设计模块关系图

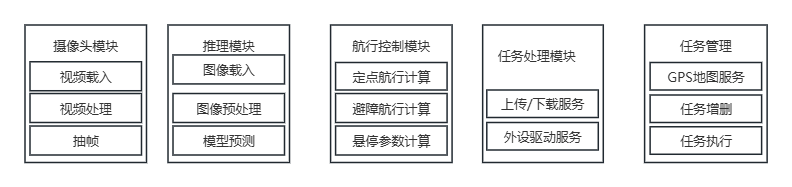
****

图2.2 模块详细服务

## 系统数据流

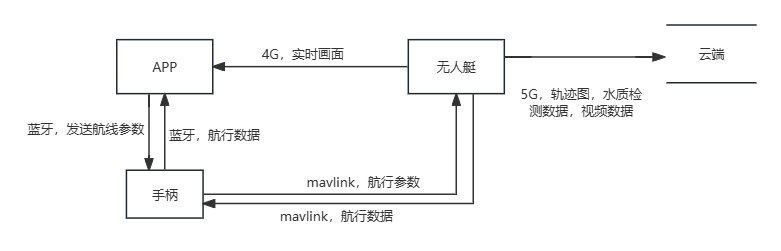


图3.1 总体数据流

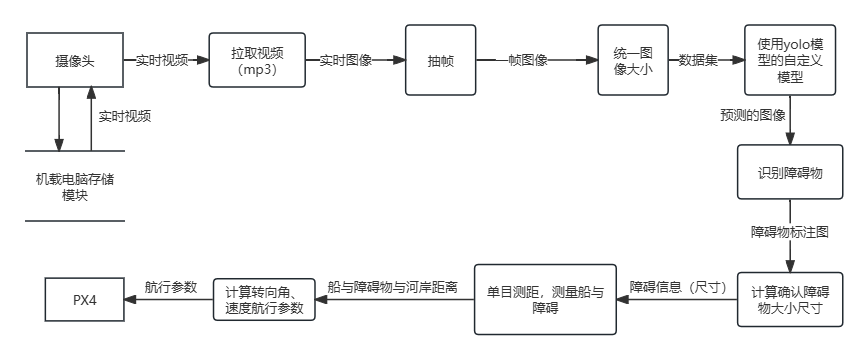


图3.2图像识别避障数据流



图3.3 水质检测数据流

****

图3.4 智能航行

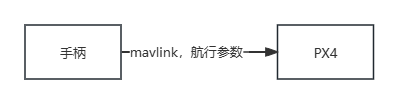


图3.5 手动航行

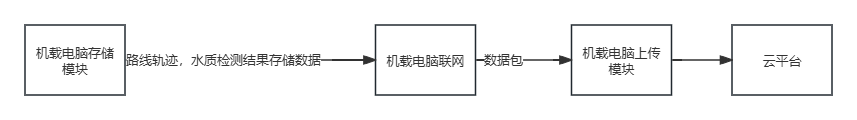


图3.6 数据上传

## 系统功能

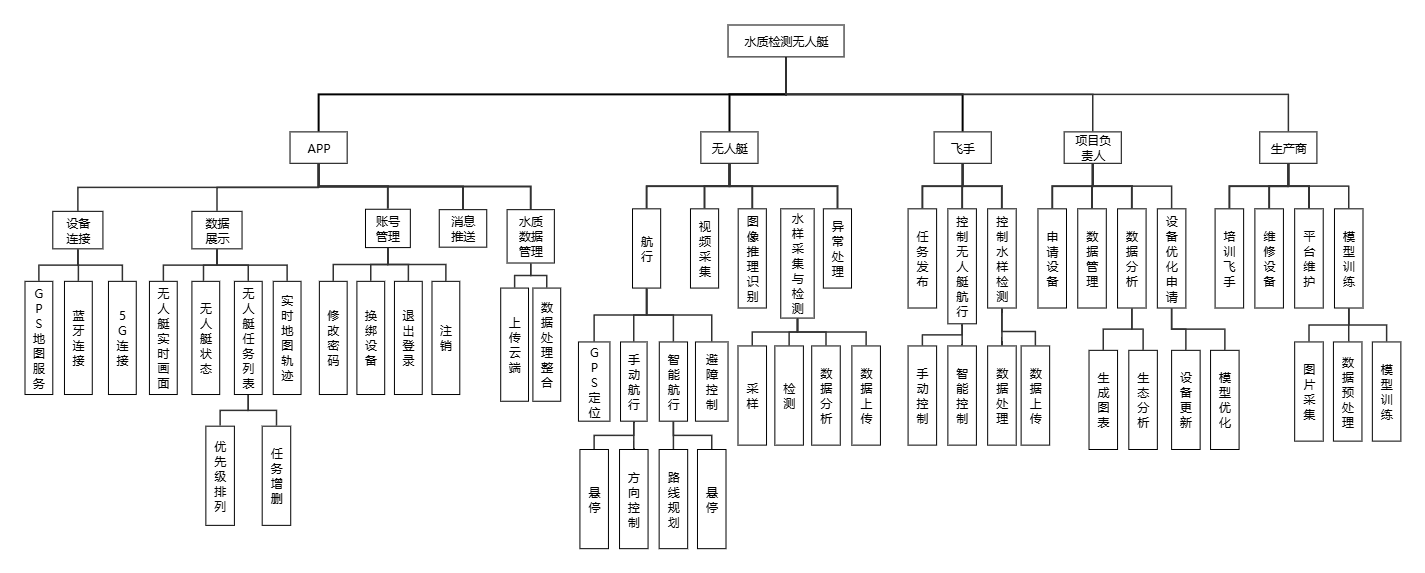
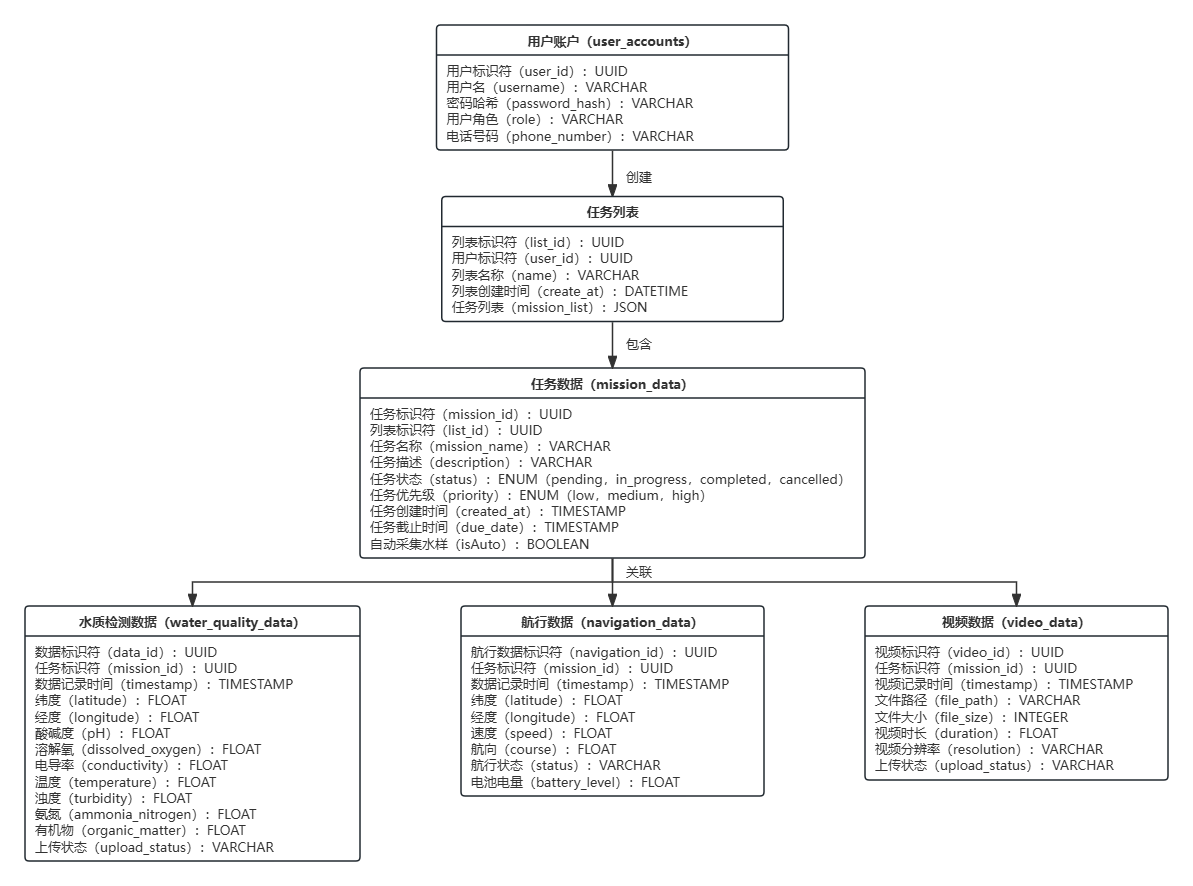


图4 总体功能框图

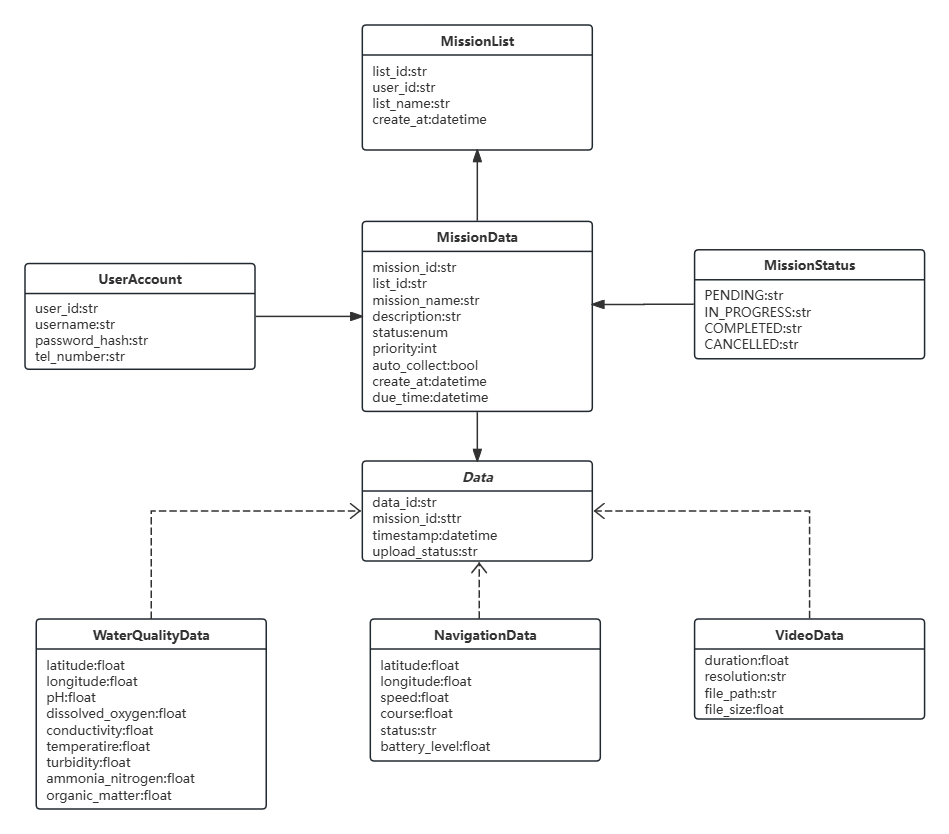
# 三、数据存储设计

## 1、数据库设计



## 数据结构设计

**2.1 类图**

****

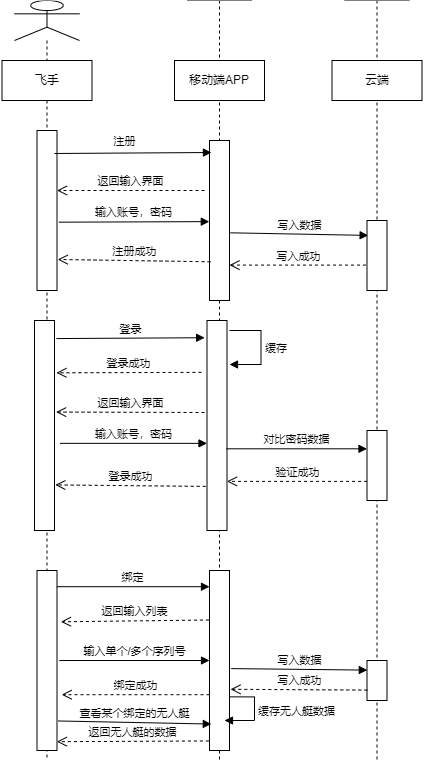
**2.2数据结构**

|  |  |
| --- | --- |
| 定义任务状态枚举类 | class MissionStatus(Enum):  PENDING = 'pending' # 等待  IN\_PROGRESS = 'in\_progress' # 进行中  COMPLETED = 'completed' # 已完成  CANCELLED = 'cancelled' # 取消 |
| 用户账户类(APP端) | class UserAccount:  def \_\_init\_\_(self, username, password\_hash, email, tel\_number):  self.user\_id = str(uuid.uuid4()) # 用户标识符  self.username = username # 用户名(str)  self.password\_hash = password\_hash # 密码(str)  self.tel\_number = tel\_number # 电话号码(str) |
| 任务数据类(无人艇、APP端) | class MissionData:  def \_\_init\_\_(self, list\_id, mission\_name, description, status, priority, auto\_collect, create\_at, due\_time):  self.mission\_id = str(uuid.uuid4()) # 任务标识符  self.list\_id = list\_id # 列表标识符(str)  self.mission\_name = mission\_name # 任务名称(str)  self.description = description # 任务描述(str)  self.status = status # 任务状态(enum)  self.priority = priority # 优先级(int,数字越小优先级越高)  self.auto\_collect = auto\_collect # 自动采集水样(bool)  self.create\_at = create\_at # 任务创建时间(datetime)  self.due\_time = due\_time # 任务截止时间(datetime) |
| 数据类基类(无人艇) | class Data:  def \_\_init\_\_(self, mission\_id, timestamp, upload\_status):  self.data\_id = str(uuid.uuid4()) # 数据标识符  self.mission\_id = mission\_id # 任务标识符(str)  self.timestamp = timestamp #数据记录时间(datatime)  self.upload\_status = upload\_status # 数据上传状态(str) |
| 水质检测数据类  (无人艇) | class WaterQualityData(Data):  def \_\_init\_\_(self, mission\_id, timestamp, upload\_status, latitude, longitude,pH, dissolved\_oxygen, conductivity, temperature, turbidity, ammonia\_nitrogen, organic\_matter):  super().\_\_init\_\_(mission\_id, timestamp, upload\_status)  self.latitude = latitude # 纬度(float)  self.longitude = longitude # 经度(float)  self.pH = pH # 酸碱度(float)  self.dissolved\_oxygen = dissolved\_oxygen # 溶氧量(float)  self.conductivity = conductivity # 电导率(float)  self.temperature = temperature # 温度(float)  self.turbidity = turbidity # 浊度(float)  self.ammonia\_nitrogen = ammonia\_nitrogen #氨氮量(float)  self.organic\_matter = organic\_matter # 有机物含量(float) |
| 航行数据类(无人艇) | class NavigationData(Data):  def \_\_init\_\_(self, mission\_id, timestamp, upload\_status, latitude, longitude, speed, course, status, battery\_level):  super().\_\_init\_\_(mission\_id, timestamp, upload\_status)  self.latitude = latitude # 纬度(float)  self.longitude = longitude # 经度(float)  self.speed = speed # 航行速度(float)  self.course = course # 航向(float)  self.status = status # 航行状态(str)  self.battery\_level = battery\_level # 电池电量(float) |
| 视频数据类(无人艇) | class VideoData(Data):  def \_\_init\_\_(self, mission\_id, timestamp, upload\_status, latitude, longitude, duration, resolution, file\_path, file\_size):  super().\_\_init\_\_(mission\_id, timestamp, upload\_status)  self.duration = duration # 视频时长(float)  self.resolution = resolution # 分辨率(str)  self.file\_path = file\_path # 文件路径(str)  self.file\_size = file\_size # 文件大小(float) |
| 任务列表类（APP端） | class MissionList:  def \_\_init\_\_(self, list\_id, user\_id, list\_name, create\_at):  self.list\_id = list\_id # 队列标识符(str)  self.user\_id = user\_id # 用户标识符(str)  self.list\_name = list\_name # 队列名称(str)  self.create\_at = create\_at # 列表创建时间(datetime)  self.mission\_list = [] # 创建列表(str) |

# 四、关键业务流程实现设计

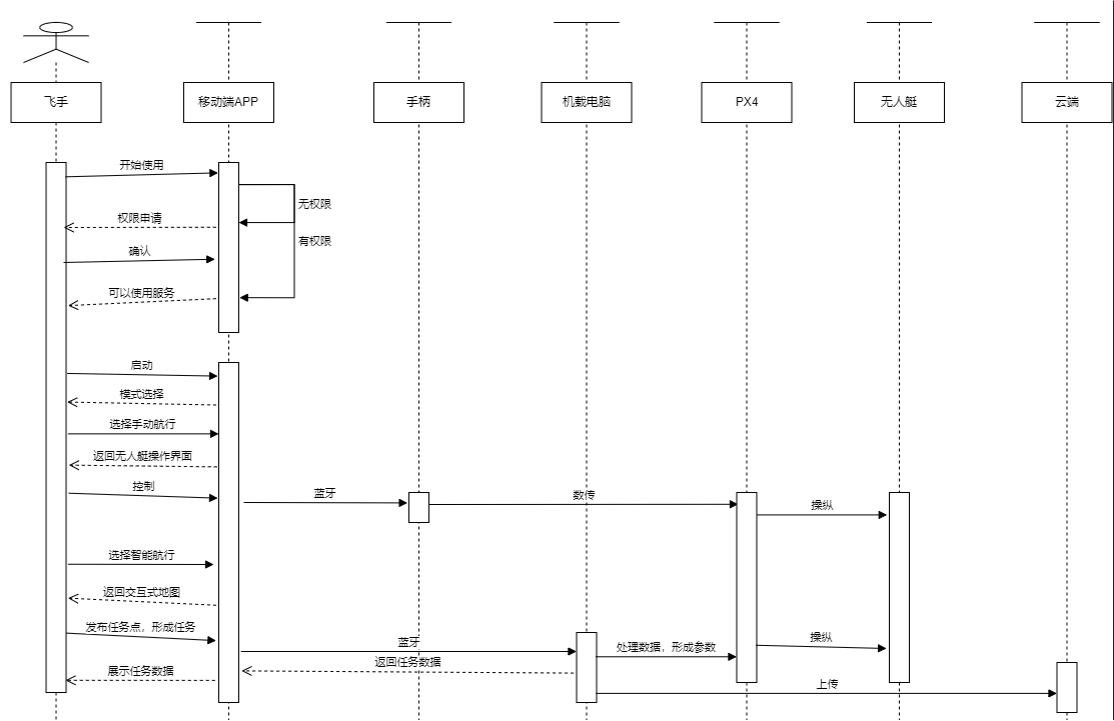
## 1、申请使用app控制无人艇设计

### 1.1 申请使用APP



1. 对于新的用户，APP端给出输入界面，用户应该输入账号，密码以及无人艇的相关数据进行注册，APP会把数据上传到云端，如果没啥冲突，就云端给APP返回注册成功，然后由APP通知用户注册成功
2. 登录时，假如APP端有缓存数据，用户可以直接登录，假如缓存不存在或者过期，用户需要在给定的界面中输入账号和密码，APP进行上传，云端进行比对，验证成功即可登录使用APP
3. 用户可以进入绑定界面，输入无人艇的序列号，并由APP端将对应关系数据写入云端，将无人艇的数据缓存至本地，进行备份，绑定成功后，用户可以点击某个具体的无人艇，可以查看其参数

### 1.2 APP控制无人艇

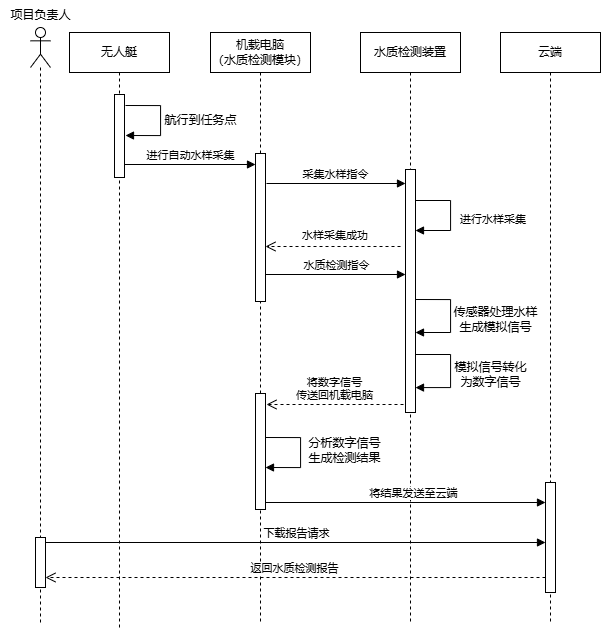


①飞手登录成功后,开始使用移动端app,首先判断app是否已经获取到所用的权限,若应用未获得所有需要的权限则提示飞手去为应用申请权限,若以获得所有需要的权限则可以正常使用APP提供的服务

②飞手启动后，可选择手动航行和智能航行

1. 选择手动航行，APP返回无人艇操作界面，是可以通过这个界面远端来操控手柄，是提供了一个用户友好型操作界面，APP通过蓝牙操作手柄，手柄通过数传发送给PX4，由PX4来操作无人艇
2. 选择智能航行，APP返回一个交互式地图，可以展示无人艇周边的地图，用户可以通过点击地图来发布任务点，这些任务点会形成一个任务，并且这些任务点可选择成为采样点还是中间节点，如果是采样点会进行采样，APP通过蓝牙把任务交互给机载电脑，由机载电脑来规划以及后续的行动，并发送参数给PX4，由PX4来操纵无人艇，这些产生的数据，由机载电脑返回给APP端以及上传给云端

## 无人艇水质检测



①飞手设置任务时选择手动采集水样或自动采集水样。

②手动采集水样：无人艇巡航到任务点后悬停，直到飞手在APP发送水质检测指令才会进行水样采集。

自动采集水样：无人艇巡航到任务点悬停后，自动进行水样采集。若任务点存在障碍物，机载电脑根据任务点坐标计算范围3m的水质检测点，根据GPS坐标和任务点坐标计算无人艇传向角，将传向角发送给PX4，控制无人艇驶向下一个采样点。

③到达采样点后，机载电脑向水质检测装置发送水样采集指令。

④水样检测装置进行水样采集。

⑤当水样采集成功后，机载电脑向水质检测装置发送水质检测指令。

⑥水质检测装置中的传感器将化学或物理信号转化为模拟信号。

⑦将模拟信号进行放大、滤波器去除噪声。

⑧模拟信号转化为数字信号，便于系统处理。

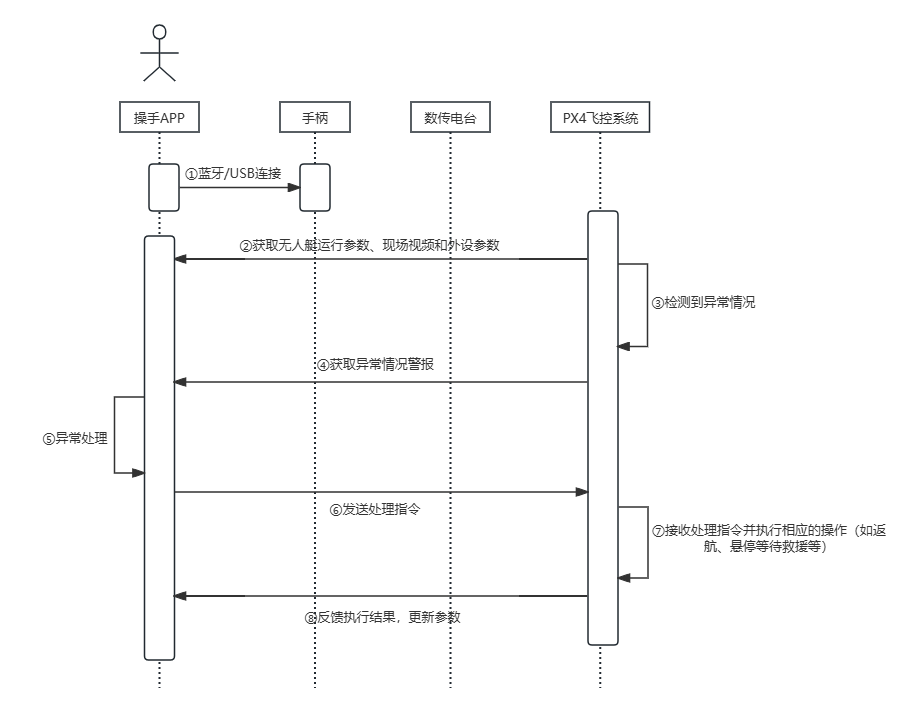
⑨水质检测装置将数字信号传回机载电脑。

⑩机载电脑对数字信号进行进一步的预处理，如校准、补偿、归一化，生成检测结果。

⑪机载电脑向APP端返回水质信测成功信号与水质检测结果，并将结果发送至云端。

⑫飞手可通过APP看到水质检测结果，项目负责人可从云端下载水质检测报告。

## 无人艇异常情况处理



①操手通过蓝牙或USB接口让APP与手柄建立连接。

②通过运行MavLink协议的数传电台，手柄获取无人艇上PX4飞控系统的运行参数（如GPS定位、电量信息、航行姿态等）、现场视频和外设仪器数据，并通过蓝牙或USB接口传回APP。

③无人艇上PX4飞控系统检测到异常情况，如没电或搁浅。

④无人艇使用数传电台将异常情况警报传到手柄，手柄再通过蓝牙或USB接口传回APP。

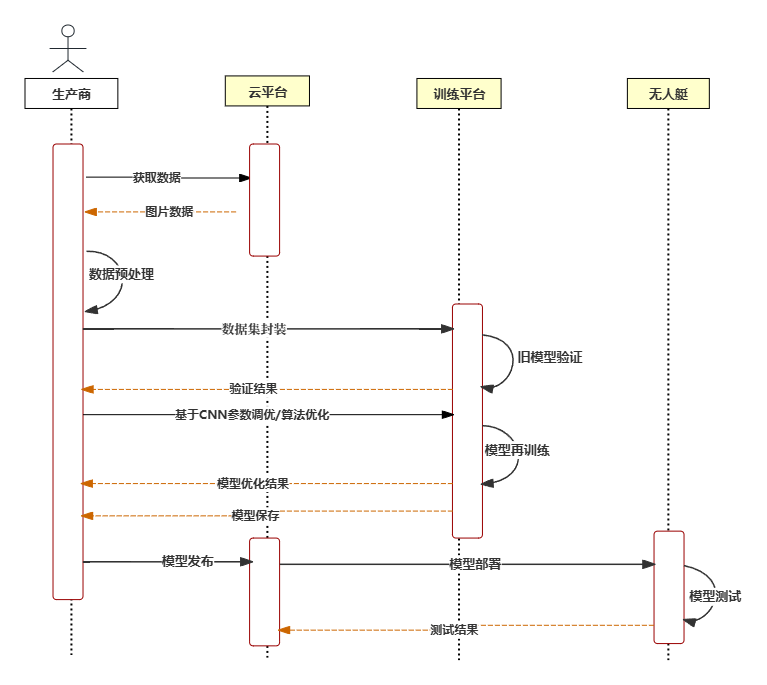
⑤操手在APP上根据警报类型进行相应的异常情况处理。

⑥APP通过蓝牙或USB接口将处理指令发送到手柄，手柄再通过数传电台将处理指令发送到PX4飞控系统。

⑦PX4飞控系统接收到处理指令并执行相应的操作，如返航或悬停等待救援等。

⑧无人艇通过数传电台将执行结果和PX4飞控系统参数（如GPS定位、电量信息等）更新反馈到手柄，并传回APP。

## **模型再训练功能**



①生产商从项目负责人处获取图片数据，该图片数据为无人艇通过其搭载的传感器和摄像头收集的图片数据。

②数据预处理和封装：对图片数据格式进行统一，包括调整大小、归一化、去噪等操作，以便于模型训练，为模型训练提供可靠的数据基础。将预处理后的图片数据封装成数据集，包括训练集、验证集和测试集。

③生产商将封装好的数据集放入旧模型里进行预测识别，根据其验证结果反馈出模型的缺陷，评估其性能。根据旧模型的验证结果，基于深度学习的卷积神经网络对模型进行参数调优或算法优化，如进行正则化、数据增强等方法，以提高模型性能。

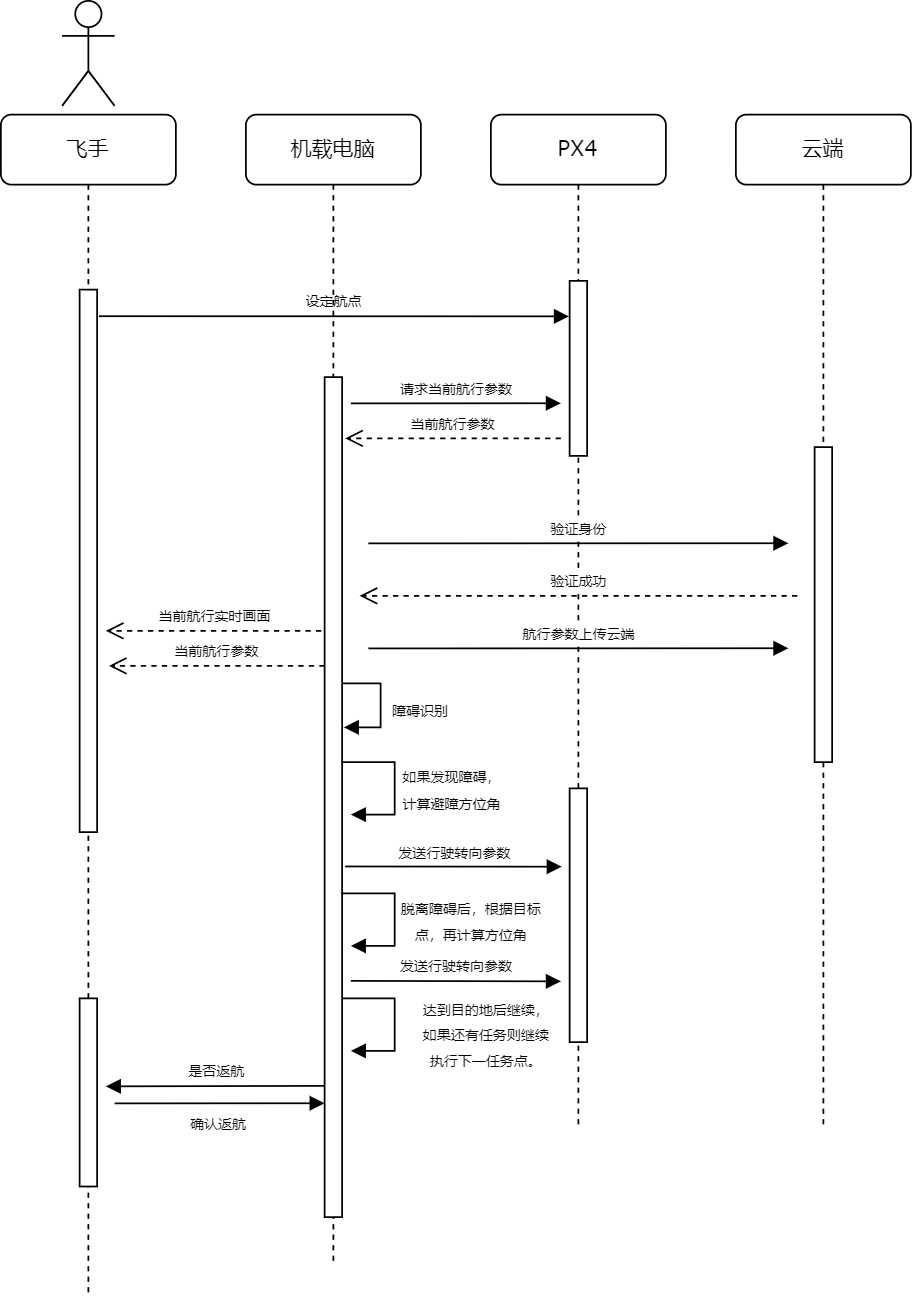
④根据数据集对新模型进行再训练，生产商分析模型优化结果并对模型进行保存。

⑤生产商发布新模型到云平台。

⑥app登录，无人艇联网启动时，无人艇会自动执行更新脚本，去最新模型发布云平台寻找最新模型，如果有新版本模型则拉去更新，新模型从云平台更新部署到无人艇上，并删除旧模型。

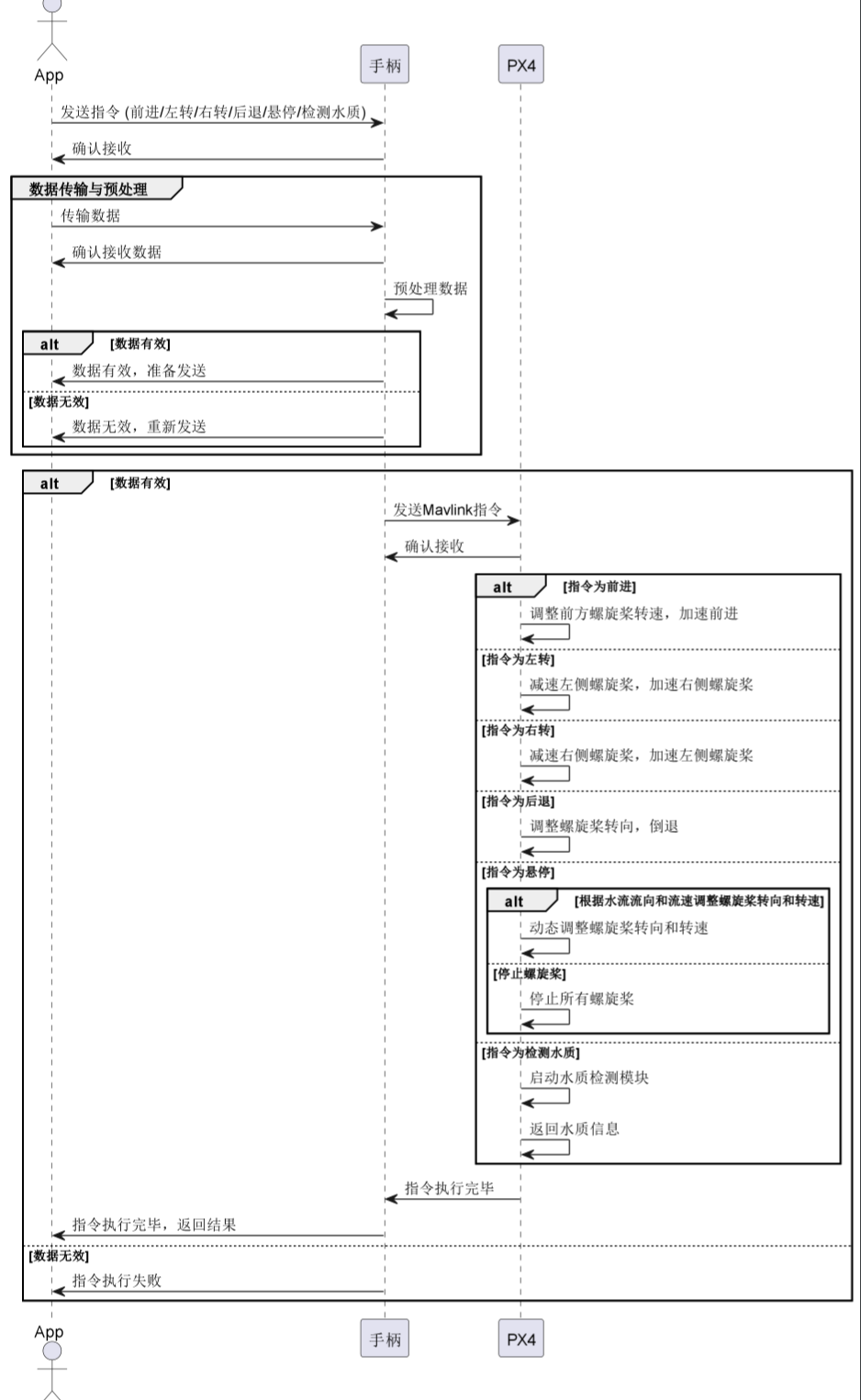
⑦新模型部署到无人艇，无人艇航行时测试新模型，并反馈结果给云平台。

## **定点智能航行功能**



1. 飞手使用APP设定任务点，通过手柄发送给PX4。
2. 机载电脑调用PX4的串口API获取当前无人艇的GPS坐标，速度等行驶参数。
3. 机载电脑在任务期间会通过4G模块发送实时画面给操手
4. 机载电脑在任务期间会实时传输当前运行参数给操手
5. 机载电脑在向云端上传数据前，需要发送自身机器码验证身份。
6. 云端验证成功后自动发送验证成功码
7. 机载电脑上传航行参数给云端
8. 机载电脑在任务期间会实时进行障碍识别
9. 如果机载电脑发现到障碍，将会根据障碍大小计算避障方位角。
10. 计算完成避障方位角后，机载电脑将再根据目标点计算方位角，向目标点进行行驶。
11. 向PX4发送新的行驶转向数据。
12. 达到目的地后机载电脑将再选取任务进行行驶，直至完成所有任务。
13. 完成所有任务后，向飞手发送是否返航的消息。
14. 飞手在APP上确定返航，无人艇则执行返航任务

## **手动航行功能**



①飞手通过 App发送指令 (前进/左转/右转/后退/悬停/检测水质):

App向手柄发送一个特定的指令，指令可以是前进、左转、右转、后退、悬停或检测水质。

②手柄确认接收:

手柄通过蓝牙协议接收到来自App的指令，并向App发送确认接收的响应。

③数据传输与预处理:

App传输数据:App向手柄发送详细的指令数据。

手柄确认接收数据:手柄接收到数据，并向App发送确认接收的数据响应。

手柄预处理数据:手柄对接收到的数据进行预处理，以确保数据的有效性和完整性。

数据有效:如果数据有效，手柄通知App数据有效，准备发送给PX4。

数据无效:如果数据无效，手柄通知App数据无效，并请求重新发送数据。

④手柄发送Mavlink指令:

手柄通过Mavlink协议向PX4发送预处理过的指令。

⑤PX4确认接收:

PX4接收到来自手柄的指令，并向手柄发送确认接收的响应。

⑥执行指令:

指令为前进:

PX4调整前方螺旋桨的转速以加速前进。

指令为左转 :

PX4减速左侧螺旋桨并加速右侧螺旋桨，使无人艇左转。

指令为右转 :

PX4减速右侧螺旋桨并加速左侧螺旋桨，使无人艇右转。

指令为后退 :

PX4调整螺旋桨的转向以使无人艇倒退。

指令为悬停 :

根据水流流向和流速调整螺旋桨转向和转速 :

PX4动态调整螺旋桨的转向和转速，以应对水流流向和流速，保持悬停。

停止螺旋桨 :

PX4停止所有螺旋桨，使无人艇悬停在原地。

指令为检测水质 :

PX4启动水质检测模块进行水质检测，并返回水质信息。

⑦ PX4指令执行完毕 :

PX4完成指令执行后，向手柄发送指令执行完毕的通知。

⑧ 手柄返回结果 :

手柄接收到PX4的通知后，向App报告指令执行完毕，并返回相关的结果。

⑨ 数据无效处理 :

如果在预处理过程中数据被认为是无效的，手柄会通知App指令执行失败，并请求重新发送有效数据。

# 五、重要接口设计

### 5.1 app与平台

**登录接口**

|  |  |
| --- | --- |
| **名称** | 登录 |
| **描述** | 登录的请求 |
| **URL** | userLogin |
| **调用方法** | POST |
| **传入参数** | userAcc # 账号  userPwd # 密码 |
| **返回值** | {  "state":200,  "message": "登录成功",  "map":map<String,Object>{{“token”,JWT},{“user”,”userName”}}”,  } |
| **状态码** | 200 : 登录成功  400 : 登录失败 |
| **说明** |  |

**注册接口**

|  |  |
| --- | --- |
| **名称** | 注册 |
| **描述** | 注册的请求 |
| **URL** | userSignup |
| **调用方法** | POST |
| **传入参数** | userAcc # 账号  userPwd # 密码 |
| **返回值** | {  "state":200,  "message": "注册成功",  "map":map<String,Object>{{“token”,JWT},{“user”,”userName”}}”,  } |
| **状态码** | 200 : 注册成功  402 : 用户已注册  403 : 请检查密码 |
| **说明** |  |

**水质数据下载接口**

|  |  |
| --- | --- |
| **名称** | 下载检测记录 |
| **描述** | 下载水质检测的所有数据，以excel方式下载 |
| **URL** | downloadData |
| **调用方法** | GET |
| **传入参数** | userRole |
| **返回值** | {  "state":200,  "message": "下载成功",  “excel” : {}  } |
| **状态码** | 400 : 参数错误  403 : 用户状态异常  404 : 未找到 |
| **说明** |  |

**实时视频信息接口**

|  |  |
| --- | --- |
| **名称** | 获取视频信息 |
| **描述** | 根据ID获取视频的详细信息 |
| **URL** | Videos\_Get |
| **调用方法** | POST |
| **传入参数** | Video\_id #视频唯一标识符 |
| **返回值** | {  “state”:，  “video\_id”:,  “mission\_id”,  “timestamp”:,  “file\_path”:,  “file\_size”:,  “duration”:,  “resolution”:,  “upload\_status”:  } |
| **状态码** | 200: 请求成功  404：未找到对应ID视频 |
| **说明** | 根据ID获取视频的详细信息 |

**水质数据展示接口**

|  |  |
| --- | --- |
| **名称** | 检测记录 |
| **描述** | 展示水质检测的所有数据 |
| **URL** | showData |
| **调用方法** | GET |
| **传入参数** | 无 |
| **返回值** | {  "state":200,  "message": "展示数据获取成功",  “obj” : {}  } |
| **状态码** | 400 : 参数错误  404 : 未找到 |
| **说明** |  |

**添加任务接口**

|  |  |
| --- | --- |
| **名称** | 添加任务 |
| **描述** | 用于添加新的任务到任务列表 |
| **URL** | Add\_task |
| **调用方法** | POST |
| **传入参数** | {  Mission\_id #任务标识符  User\_id #用户标识符  Mission\_name #任务名称  Description #任务描述  Status #任务状态  Priority #任务优先级  Created\_at #任务创建时间  Due\_date #任务截止时间  } |
| **返回值** | {  “state'”,  “message”:”添加任务成功”  } |
| **状态码** | 200: 添加成功  400：参数错误  404：请求无效 |
| **说明** | 用于添加新的任务到任务列表 |

**更新任务接口**

|  |  |
| --- | --- |
| **名称** | 更新任务 |
| **描述** | 用于更新任务列表已有的任务 |
| **URL** | update\_task |
| **调用方法** | POST |
| **传入参数** | {  Mission\_id #任务标识符  User\_id #用户标识符  Mission\_name #任务名称  Description #任务描述  Status #任务状态  Priority #任务优先级  Created\_at #任务创建时间  Due\_date #任务截止时间  } |
| **返回值** | {  “state'”,  “message”:”任务更新成功”  } |
| **状态码** | 200: 更新成功  400：参数错误  404：请求无效 |
| **说明** | 用于更新任务列表已有的任务 |

**删除任务接口**

|  |  |
| --- | --- |
| **名称** | 删除任务 |
| **描述** | 用于删除任务列表已有的任务 |
| **URL** | Delete\_task |
| **调用方法** | POST |
| **传入参数** | Mission\_id #任务标识符 |
| **返回值** | {  “state'”,  “message”:”任务删除成功”  } |
| **状态码** | 200: 删除成功  400：参数错误  404：请求无效 |
| **说明** | 用于删除任务列表已有的任务 |

### 5.2 app与手柄

**添加任务接口**

|  |  |
| --- | --- |
| **名称** | 添加任务 |
| **描述** | 用于添加新的任务到任务列表 |
| **URL** | Add\_task |
| **调用方法** | POST |
| **传入参数** | {  Mission\_id #任务标识符  User\_id #用户标识符  Mission\_name #任务名称  Description #任务描述  Status #任务状态  Priority #任务优先级  Created\_at #任务创建时间  Due\_date #任务截止时间  } |
| **返回值** | {  “state'”,  “message”:”添加任务成功”  } |
| **状态码** | 200: 添加成功  400：参数错误  404：请求无效 |
| **说明** | 用于添加新的任务到任务列表 |

**更新任务接口**

|  |  |
| --- | --- |
| **名称** | 更新任务 |
| **描述** | 用于更新任务列表已有的任务 |
| **URL** | update\_task |
| **调用方法** | POST |
| **传入参数** | {  Mission\_id #任务标识符  User\_id #用户标识符  Mission\_name #任务名称  Description #任务描述  Status #任务状态  Priority #任务优先级  Created\_at #任务创建时间  Due\_date #任务截止时间  } |
| **返回值** | {  “state'”,  “message”:”任务更新成功”  } |
| **状态码** | 200: 更新成功  400：参数错误  404：请求无效 |
| **说明** | 用于更新任务列表已有的任务 |

**删除任务接口**

|  |  |
| --- | --- |
| **名称** | 删除任务 |
| **描述** | 用于删除任务列表已有的任务 |
| **URL** | Delete\_task |
| **调用方法** | POST |
| **传入参数** | Mission\_id #任务标识符 |
| **返回值** | {  “state'”,  “message”:”任务删除成功”  } |
| **状态码** | 200: 删除成功  400：参数错误  404：请求无效 |
| **说明** | 用于删除任务列表已有的任务 |

**添加航行数据接口**

|  |  |
| --- | --- |
| **名称** | 添加航行数据 |
| **描述** | 添加新的航行数据记录 |
| **URL** | navigation |
| **调用方法** | POST |
| **传入参数** | {  "navigation\_id": , #航行数据唯一标识符  "mission\_id": , #任务唯一标识符  "timestamp":, #数据记录时间  "latitude": , #纬度  "longitude": , #经度  "speed": , #速度  "course": , #航向  "status": , #航行状态  "battery\_level": #电量  } |
| **返回值** | {  “State”：，  “message”:”航行数据创建成功”  } |
| **状态码** | |  |  | | --- | --- | | 200 | 添加成功 | | 400 | 请求无效 | | 404 | 请求资源未找到 | |
| **说明** | 此接口用于添加新的航行数据记录，包括数据记录时间，经纬度，速度、电量，航行状态，航向等信息 |

**获取航行数据接口**

|  |  |
| --- | --- |
| **名称** | 获取航行数据 |
| **描述** | 根据航行ID检索特定的航行数据记录 |
| **URL** | Get\_navigation |
| **调用方法** | POST |
| **传入参数** | navigation\_id #航行数据唯一标识符 |
| **返回值** | {  “State”：，  “message”:”航行数据获取成功”,  “Navigation\_id”:,  “mission\_id”:,  “timestamp”:,  “latitude”:,  “longitude”:,  “speed”:,  “course”:,  “status”:,  “battery\_level”:  } |
| **状态码** | |  |  | | --- | --- | | 200 | 获取成功 | | 400 | 请求无效 | | 404 | 请求资源未找到 | |
| **说明** | 此接口用于获取航行数据记录，包括数据记录时间，经纬度，速度、电量，航行状态，航向等信息 |

**更新航行数据接口**

|  |  |
| --- | --- |
| **名称** | 更新航行数据 |
| **描述** | 根据航行ID更新现有的航行数据记录 |
| **URL** | Update\_navigation |
| **调用方法** | POST |
| **传入参数** | {  "navigation\_id": , #航行数据唯一标识符  "mission\_id": , #任务唯一标识符  "timestamp":, #数据记录时间  "latitude": , #纬度  "longitude": , #经度  "speed": , #速度  "course": , #航向  "status": , #航行状态  "battery\_level": #电量  } |
| **返回值** | {  “State”：，  “message”："更新成功"  } |
| **状态码** | |  |  | | --- | --- | | 200 | 更新成功 | | 400 | 请求无效 | | 404 | 请求资源未找到 | |
| **说明** | 此接口用于更新航行数据记录，包括数据记录时间，经纬度，速度、电量，航行状态，航向等信息 |

**删除航行数据接口**

|  |  |
| --- | --- |
| **名称** | 删除航行数据 |
| **描述** | 根据航行ID删除特定的航行数据记录 |
| **URL** | Delete\_navigation |
| **调用方法** | POST |
| **传入参数** | navigation\_id #航行数据唯一标识符 |
| **返回值** | {  “State”：，  “message”:”航行数据删除成功”,  } |
| **状态码** | |  |  | | --- | --- | | 200 | 删除成功 | | 400 | 请求无效 | | 404 | 请求资源未找到 | |
| **说明** | 此接口用于删除航行数据记录，包括数据记录时间，经纬度，速度、电量，航行状态，航向等信息 |

### 5.3 手柄与px4

**添加任务接口**

|  |  |
| --- | --- |
| **名称** | 添加任务 |
| **描述** | 用于添加新的任务到任务列表 |
| **URL** | Add\_task |
| **调用方法** | POST |
| **传入参数** | {  Mission\_id #任务标识符  User\_id #用户标识符  Mission\_name #任务名称  Description #任务描述  Status #任务状态  Priority #任务优先级  Created\_at #任务创建时间  Due\_date #任务截止时间  } |
| **返回值** | {  “state'”,  “message”:”添加任务成功”  } |
| **状态码** | 200: 添加成功  400：参数错误  404：请求无效 |
| **说明** | 用于添加新的任务到任务列表 |

**更新任务接口**

|  |  |
| --- | --- |
| **名称** | 更新任务 |
| **描述** | 用于更新任务列表已有的任务 |
| **URL** | update\_task |
| **调用方法** | POST |
| **传入参数** | {  Mission\_id #任务标识符  User\_id #用户标识符  Mission\_name #任务名称  Description #任务描述  Status #任务状态  Priority #任务优先级  Created\_at #任务创建时间  Due\_date #任务截止时间  } |
| **返回值** | {  “state'”,  “message”:”任务更新成功”  } |
| **状态码** | 200: 更新成功  400：参数错误  404：请求无效 |
| **说明** | 用于更新任务列表已有的任务 |

**删除任务接口**

|  |  |
| --- | --- |
| **名称** | 删除任务 |
| **描述** | 用于删除任务列表已有的任务 |
| **URL** | Delete\_task |
| **调用方法** | POST |
| **传入参数** | Mission\_id #任务标识符 |
| **返回值** | {  “state'”,  “message”:”任务删除成功”  } |
| **状态码** | 200: 删除成功  400：参数错误  404：请求无效 |
| **说明** | 用于删除任务列表已有的任务 |

**添加航行数据接口**

|  |  |
| --- | --- |
| **名称** | 添加航行数据 |
| **描述** | 添加新的航行数据记录 |
| **URL** | navigation |
| **调用方法** | POST |
| **传入参数** | {  "navigation\_id": , #航行数据唯一标识符  "mission\_id": , #任务唯一标识符  "timestamp":, #数据记录时间  "latitude": , #纬度  "longitude": , #经度  "speed": , #速度  "course": , #航向  "status": , #航行状态  "battery\_level": #电量  } |
| **返回值** | {  “State”：，  “message”:”航行数据创建成功”  } |
| **状态码** | |  |  | | --- | --- | | 200 | 添加成功 | | 400 | 请求无效 | | 404 | 请求资源未找到 | |
| **说明** | 此接口用于添加新的航行数据记录，包括数据记录时间，经纬度，速度、电量，航行状态，航向等信息 |

**获取航行数据接口**

|  |  |
| --- | --- |
| **名称** | 获取航行数据 |
| **描述** | 根据航行ID检索特定的航行数据记录 |
| **URL** | Get\_navigation |
| **调用方法** | POST |
| **传入参数** | navigation\_id #航行数据唯一标识符 |
| **返回值** | {  “State”：，  “message”:”航行数据获取成功”,  “Navigation\_id”:,  “mission\_id”:,  “timestamp”:,  “latitude”:,  “longitude”:,  “speed”:,  “course”:,  “status”:,  “battery\_level”:  } |
| **状态码** | |  |  | | --- | --- | | 200 | 获取成功 | | 400 | 请求无效 | | 404 | 请求资源未找到 | |
| **说明** | 此接口用于获取航行数据记录，包括数据记录时间，经纬度，速度、电量，航行状态，航向等信息 |

**更新航行数据接口**

|  |  |
| --- | --- |
| **名称** | 更新航行数据 |
| **描述** | 根据航行ID更新现有的航行数据记录 |
| **URL** | Update\_navigation |
| **调用方法** | POST |
| **传入参数** | {  "navigation\_id": , #航行数据唯一标识符  "mission\_id": , #任务唯一标识符  "timestamp":, #数据记录时间  "latitude": , #纬度  "longitude": , #经度  "speed": , #速度  "course": , #航向  "status": , #航行状态  "battery\_level": #电量  } |
| **返回值** | {  “State”：，  “message”："更新成功"  } |
| **状态码** | |  |  | | --- | --- | | 200 | 更新成功 | | 400 | 请求无效 | | 404 | 请求资源未找到 | |
| **说明** | 此接口用于更新航行数据记录，包括数据记录时间，经纬度，速度、电量，航行状态，航向等信息 |

**删除航行数据接口**

|  |  |
| --- | --- |
| **名称** | 删除航行数据 |
| **描述** | 根据航行ID删除特定的航行数据记录 |
| **URL** | Delete\_navigation |
| **调用方法** | POST |
| **传入参数** | navigation\_id #航行数据唯一标识符 |
| **返回值** | {  “State”：，  “message”:”航行数据删除成功”,  } |
| **状态码** | |  |  | | --- | --- | | 200 | 删除成功 | | 400 | 请求无效 | | 404 | 请求资源未找到 | |
| **说明** | 此接口用于删除航行数据记录，包括数据记录时间，经纬度，速度、电量，航行状态，航向等信息 |

### 5.4 px4与机载电脑

**避障指令数据接口**

|  |  |
| --- | --- |
| **名称** | 无人艇指令接口 |
| **描述** | 控制无人艇避障 |
| **URL** | obstacleAvoidance |
| **调用方法** | POST |
| **传入参数** | {"转动角度"} |
| **返回值** | {  "state":200,  "message": "执行成功",  } |
| **状态码** | 200 : 成功  400 : 参数错误  404 : 未找到 |
| **说明** |  |

**悬停指令数据接口**

|  |  |
| --- | --- |
| **名称** | 无人艇指令接口 |
| **描述** | 控制无人艇悬停 |
| **URL** | hover |
| **调用方法** | POST |
| **传入参数** | {"无人艇指令"} |
| **返回值** | {  "state":200,  "message": "执行成功",  } |
| **状态码** | 200 : 成功  400 : 参数错误  404 : 未找到 |
| **说明** |  |

### 5.5 机载电脑与平台

**水质数据上传接口**

|  |  |
| --- | --- |
| **名称** | 数据上传 |
| **描述** | 上传水质参数的请求 |
| **URL** | waterQualityData |
| **调用方法** | POST |
| **传入参数** | { “DataId”, 数据条目id  “MissionId”, 任务条目id  “timestamp”:””, 时间戳  “latitude”:””, 经度  “longitude”:””, 纬度  “pH”:””, 酸碱度  “dissolved\_oxygen”:””, 溶解氧  “conductivity”:””, 电导率  “temperature”:””, 温度  “turbidity”:””, 浊度  “ammonia\_nitrogen”:””, 氨氮  “organic\_matter”:””, 有机物  “sensor\_status”:””, 传感器状态  “upload\_status”:”” 上传状态  } |
| **返回值** | {  "state":200,  "message": "水质数据上传成功",  } |
| **状态码** | 200 : 成功  400 : 参数错误  404 : 未找到 |
| **说明** |  |

**上传航行数据接口**

|  |  |
| --- | --- |
| **名称** | 添加航行数据 |
| **描述** | 添加新的航行数据记录 |
| **URL** | navigation |
| **调用方法** | POST |
| **传入参数** | {  "navigation\_id": , #航行数据唯一标识符  "mission\_id": , #任务唯一标识符  "timestamp":, #数据记录时间  "latitude": , #纬度  "longitude": , #经度  "speed": , #速度  "course": , #航向  "status": , #航行状态  "battery\_level": #电量  } |
| **返回值** | {  “State”：，  “message”:”航行数据创建成功”  } |
| **状态码** | |  |  | | --- | --- | | 200 | 添加成功 | | 400 | 请求无效 | | 404 | 请求资源未找到 | |
| **说明** | 此接口用于添加新的航行数据记录，包括数据记录时间，经纬度，速度、电量，航行状态，航向等信息 |

**更新航行数据接口**

|  |  |
| --- | --- |
| **名称** | 更新航行数据 |
| **描述** | 根据航行ID更新现有的航行数据记录 |
| **URL** | Update\_navigation |
| **调用方法** | POST |
| **传入参数** | {  "navigation\_id": , #航行数据唯一标识符  "mission\_id": , #任务唯一标识符  "timestamp":, #数据记录时间  "latitude": , #纬度  "longitude": , #经度  "speed": , #速度  "course": , #航向  "status": , #航行状态  "battery\_level": #电量  } |
| **返回值** | {  “State”：，  “message”："更新成功"  } |
| **状态码** | |  |  | | --- | --- | | 200 | 更新成功 | | 400 | 请求无效 | | 404 | 请求资源未找到 | |
| **说明** | 此接口用于更新航行数据记录，包括数据记录时间，经纬度，速度、电量，航行状态，航向等信息 |

# 六、系统部署设计

## 项目工程分类

project\_root/

├── docs/

│ ├── Design/ ##设计文档

│ │ ├── 概要设计说明书.md

│ │ └── 需求文档.md

│ ├── Manuals/ ##用户手册和测试文档

│ │ ├── 用户手册.pdf

│ │ └── 测试文档.md

│ └── Templates/ ##模板文件

│ ├── 文档模板.docx

│ └── 演示视频.mp4

├── src/ ##源代码目录

│ ├── app/

│ │ ├── android/ Android应用代码

│ │ │ ├── app/

│ │ │ │ ├── java/com/example/app/

│ │ │ │ │ ├── MainActivity.java

│ │ │ │ │ └── ...

│ │ │ ├── build.gradle

│ │ │ └── ...

│ │ ├── ios/  iOS应用代码

│ │ │ ├── App/

│ │ │ │ ├── ViewController.swift

│ │ │ │ └── ...

│ │ │ ├── Podfile

│ │ │ └── ...

│ │ ├── harmonyos/ 鸿蒙系统应用代码

│ │ │ ├── entry/ 应用入口，包含AbilitySlice和MainAbilitySlice

│ │ │ │ ├── AbilitySlice.java

│ │ │ │ ├── MainAbilitySlice.java

│ │ │ │ └── ...

│ │ │ ├── resources/ 资源文件，如基础资源、原始文件等。

│ │ │ │ ├── base/

│ │ │ │ ├── rawfile/

│ │ │ │ ├── resfile/

│ │ │ │ └── ...

│ │ │ ├── config.json 应用配置文件

│ │ │ └── ...

│ │ └── common/ 跨平台共享的资源和逻辑。

│ │ ├── shared\_resources/

│ │ └── shared\_logic/

│ ├── backend/ 后端服务代码

│ │ ├── user\_management/ 用户管理

│ │ │ ├── login.py

│ │ │ ├── signup.py

│ │ │ └── ...

│ │ ├── task\_management/ 任务管理

│ │ │ ├── add\_task.py

│ │ │ ├── update\_task.py

│ │ │ ├── delete\_task.py

│ │ │ └── ...

│ │ ├── data\_processing/ 数据处理

│ │ │ ├── water\_quality\_data.py

│ │ │ └── ...

│ │ └── interfaces/ 接口服务

│ │ ├── video.py

│ │ ├── navigation.py

│ │ └── ...

│ ├── firmware/ 固件代码

│ │ ├── px4/ PX4飞控系统代码

│ │ │ ├── flight\_control.c

│ │ │ └── ...

│ │ └── onboard\_computer/ 机载电脑控制逻辑

│ │ ├── navigation\_logic.py

│ │ ├── image\_recognition.py

│ │ └── ...

│ ├── libraries/ 库代码

│ │ ├── common\_lib/ 通用库

│ │ │ ├── utils.py

│ │ │ └── ...

│ │ └── third\_party\_libs/ 第三方库

│ │ ├── map\_service/

│ │ ├── streaming\_service/

│ │ └── cloud\_service/

│ └── model\_training/ 模型训练相关代码

│ ├── datasets/ 训练数据集

│ ├── training\_scripts/ 训练脚本

│ │ ├── train\_model.py

│ │ └── ...

│ └── models/ 训练好的模型

│ ├── trained\_model.

│ └── ...

├── db/ 数据库目录

│ ├── design/ 数据库设计文档

│ │ ├── database\_design.docx

│ │ └── data\_structures.py

│ ├── scripts/ 数据库脚本，包括初始化和迁移脚本

│ │ ├── init.sql

│ │ └── migrations/

│ └── migrations/

│ ├── 001\_initial\_migration.py

│ └── ...

├── tests/ 测试目录

│ ├── unit\_tests/ 单元测试

│ │ ├── app\_tests/

│ │ ├── backend\_tests/

│ │ └── firmware\_tests/

│ ├── integration\_tests/ 集成测试

│ │ ├── system\_tests/

│ │ └── ...

│ └── system\_tests/ 系统测试

│ ├── acceptance\_tests/

│ └── ...

├── deployment/ 部署目录

│ ├── docker/ Docker相关文件

│ │ ├── Dockerfile

│ │ └── docker-compose.yml

│ ├── kubernetes/ Kubernetes相关文件

│ │ ├── deployment.yaml

│ │ └── service.yaml

│ └── scripts/ 部署脚本

│ ├── deploy.sh

│ └── ...

├── resources/ 资源目录

│ ├── icons/ 图标资源

│ │ ├── app\_icon.png

│ │ └── ...

│ ├── templates/ 文档模板

│ │ ├── document\_template.docx

│ │ └── ...

│ └── videos/ 演示视频

│ ├── demo\_video.mp4

│ └── ...

└── third\_party\_services/ 第三方服务目录

├── map\_service/ 地图服务相关代码和配置

├── streaming\_service/ 流媒体服务相关代码和配置

└── cloud\_service/ 云服务相关代码和配置

## 项目工程包名、配置文件定义

## 代码库结构定义

## 项目部署图