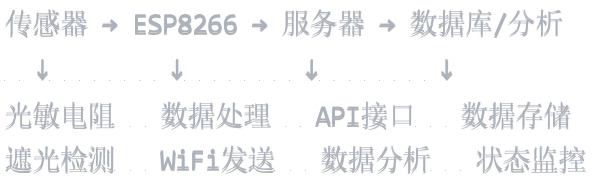
光传感器监测系统框架

1. 系统架构



2. ESP8266端 (C++)

2.1 主要功能

- 光传感器数据采集
- 遮光时间计算
- 数据预处理和过滤
- WiFi通信
- 数据缓存和重发机制

2.2 文件结构

```
ESP8266_Project/
├── main.cpp                    # 主程序
├── config.h                    # 配置文件
├── sensor.h/cpp                # 传感器处理
├── network.h/cpp               # 网络通信
└── data_manager.h/cpp          # 数据管理
```

2.3 核心模块

- **传感器模块**: 光照检测、遮光时间统计
- **网络模块**: WiFi连接、HTTP请求、错误重试
- **数据模块**: 数据缓存、批量发送、数据验证

3. 服务器端 (Python)

3.1 主要功能

- 接收ESP8266数据
- 数据验证和清洗
- 复杂数据分析
- 数据持久化存储

- 状态监控和告警

3.2 文件结构

```
Server_Project/
├── app.py                    # Flask主应用
├── config.py                 # 配置管理
├── models/
│   ├── __init__.py
│   ├── sensor_data.py        # 数据模型
│   └── device.py             # 设备模型
├── api/
│   ├── __init__.py
│   ├── sensor.py             # 传感器API
│   └── webhook.py            # Webhook处理
├── services/
│   ├── __init__.py
│   ├── data_processor.py     # 数据处理服务
│   ├── analyzer.py           # 数据分析服务
│   └── alert.py              # 告警服务
├── utils/
│   ├── __init__.py
│   ├── database.py           # 数据库工具
│   └── validator.py          # 数据验证
└── requirements.txt          # 依赖包
```

3.3 核心模块

- **API模块**: RESTful接口、数据接收、响应处理
- **数据处理模块**: 数据清洗、统计分析、异常检测
- **存储模块**: 数据库操作、数据备份
- **监控模块**: 设备状态、数据质量、系统健康

4. 数据流设计

4.1 数据格式

```json
{
  "device_id": "ESP8266_001",
  "timestamp": 1703123456789,
  "duration": 1500,
  "light_level": 245,
  "wifi_rssi": -45,
  "battery_level": 85
}
```

4.2 通信协议

- **单次数据**: POST /api/sensor-data
- **批量数据**: POST /api/sensor-data/batch
- **设备状态**: POST /api/device/heartbeat
- **配置更新**: GET /api/device/config

5. 技术栈

5.1 ESP8266端

- **开发环境**: Arduino IDE / PlatformIO
- **核心库**: ESP8266WiFi, ArduinoJson, ESP8266HTTPClient
- **传感器**: 光敏电阻 + ADC

5.2 服务器端

- **Web框架**: Flask
- **数据库**: SQLite (开发) / PostgreSQL (生产)
- **数据处理**: Pandas, NumPy
- **任务队列**: Celery (可选)
- **监控**: Logging, 健康检查接口

6. 关键特性

6.1 可靠性

- 数据缓存和重发
- 网络断线重连
- 异常处理和恢复

6.2 可扩展性

- 支持多设备接入
- 模块化设计
- 配置化管理

6.3 监控能力

- 设备在线状态
- 数据质量监控
- 系统性能监控

7. 部署建议

7.1 开发环境

- ESP8266: 本地开发板调试
- 服务器: 本地Flask开发服务器
- 数据库: SQLite文件数据库

7.2 生产环境

- ESP8266: 部署到实际硬件
- 服务器: Docker容器 + Nginx反向代理
- 数据库: PostgreSQL + Redis缓存

8. 下一步开发计划

1. **阶段1**: 基础功能实现
   - ESP8266传感器数据采集
   - 基础服务器API
   - 简单数据存储
2. **阶段2**: 功能完善
   - 数据分析和统计
   - 异常检测和告警
   - 系统监控
3. **阶段3**: 优化和扩展
   - 性能优化
   - 多设备支持
   - 高级分析功能