北方工业大学

2025-2026 学年第二学期课设报告

题目：开源鸿蒙赋能智慧农业

——基于 Hi3861 与 HarmonyOS 的智慧大棚控制终端

课程名称：鸿蒙项目制综合实践

组别：清霁

组长：陈梓睿

软件名称：Smart Shed

指导教师：王岚

2026 年 6 月 6 日

North China University of Technology

2025-2026 Second Semester
Course Design Report

Title:

OpenHarmony Empowers Smart Agriculture
—— Smart Greenhouse Control Terminal Based on Hi3861 and HarmonyOS

Course Name: HarmonyOS Full-Stack Project-Based Practice

Group: Qingji

Leader: Ray Chen (Rayawa)

Software Name: Smart Shed

Teacher: Wang Lan

2026/6/6

开源鸿蒙赋能智慧农业
——基于 Hi3861 与 HarmonyOS 的智慧大棚控制终端

摘　要：本课程设计面向现代农业温室环境监测与智能控制需求，设计并实现了一套基于开源鸿蒙生态的智慧大棚物联网控制系统 Smart Shed。系统采用 HiSilicon Hi3861 开发板作为底层感知与执行核心，基于 OpenHarmony LiteOS-M 实现温湿度、光照强度、土壤湿度等环境参数采集，以及补光灯、风扇、水泵等执行设备控制；同时结合 HarmonyOS 移动终端，基于 ArkTS、ArkUI 与 HDS Design System 构建可视化控制应用，实现数据展示、远程控制、智能调节与通信日志监测。系统整体采用“嵌入式端—通信服务端—移动应用端”的三层架构，通过 MQTT 发布/订阅机制实现设备与应用之间的双向通信。嵌入式端采用双 Hi3861 开发板设计，分别承担环境感知与设备控制任务；通信服务端利用 EMQX Broker 完成消息转发与主题管理；移动应用端通过自研 MQTT 客户端、状态管理机制、心跳检测、自动重连以及设备在线监测功能，提高系统通信稳定性与可维护性。针对智慧农业场景中的多设备协同、人机交互和系统展示需求，本项目进一步设计了响应式布局、日志可视化控制台以及基于 HDS Design 的沉浸式交互界面，使系统同时具备功能验证能力与良好的交互体验。测试结果表明，该系统能够实现温室环境实时监测、远程设备控制以及基于阈值的自动调节，验证了开源鸿蒙生态在物联网智能控制场景中的应用价值。

关键词：智慧农业；OpenHarmony；Hi3861；HarmonyOS；MQTT；ArkUI；物联网控制

OpenHarmony Empowers Smart Agriculture
—— Smart Greenhouse Control Terminal Based on Hi3861 and HarmonyOS

Abstract: This course project focuses on environmental monitoring and intelligent control requirements in modern agricultural greenhouses. A smart greenhouse IoT control system named Smart Shed is designed and implemented based on the OpenHarmony ecosystem. The system uses HiSilicon Hi3861 development boards as the core devices for environmental sensing and actuator control. Running on OpenHarmony LiteOS-M, the embedded system collects temperature, humidity, light intensity, and soil moisture data, while controlling actuators such as supplementary lighting, fans, and water pumps. Meanwhile, a HarmonyOS mobile application is developed with ArkTS, ArkUI, and HDS Design System to provide data visualization, remote control, intelligent adjustment, and communication monitoring.

The system adopts a three-layer architecture consisting of the embedded device layer, communication service layer, and mobile application layer. MQTT publish/subscribe communication is introduced to achieve bidirectional interaction between devices and applications. Two Hi3861 boards are used to separately perform sensing and actuator control tasks. The communication layer uses an EMQX Broker for message routing and topic management. The HarmonyOS application integrates a self-developed MQTT client, global state management, heartbeat detection, automatic reconnection, and device status monitoring mechanisms to improve communication reliability and maintainability.

To better satisfy the requirements of smart agriculture scenarios, including multi-device coordination, human-computer interaction, and system demonstration, the project further implements responsive layouts, a visual communication console, and immersive interfaces based on HDS Design System. Experimental results show that the system can achieve real-time greenhouse monitoring, remote equipment control, and threshold-based automatic regulation, demonstrating the application potential of the OpenHarmony ecosystem in IoT intelligent control systems.

Keywords: smart agriculture; OpenHarmony; Hi3861; HarmonyOS; MQTT; ArkUI; IoT control

目录

一、绪论

1．项目背景

传统农业大棚的温湿度控制、光照补偿、土壤浇灌等工作高度依赖人工巡检和手动操作，容易出现响应滞后、精度不足、管理成本较高等问题。当温度、湿度、光照和土壤水分等微环境参数发生快速变化时，人工管理模式难以及时捕捉并形成连续、稳定的调控闭环。随着物联网、国产芯片与国产操作系统生态的发展，将嵌入式感知控制与移动端智能决策结合起来，已经成为智慧农业场景的重要技术路径。

本项目以“开源鸿蒙赋能智慧农业”为主题，围绕鸿蒙智能硬件开发综合实践要求，设计并实现一套面向大棚场景的智慧农业物联网控制系统。系统以 Hi3861 作为底层边缘采集与执行控制平台，以 HarmonyOS 移动终端作为上位机交互入口，通过 MQTT 协议实现端云协同通信，最终形成从传感器采集、云端消息转发、移动端展示与决策到执行器动作反馈的完整控制闭环。

2．设计目标

本项目的设计目标主要有以下几个方面：

（1）实时环境监测：采集空气温度、空气湿度、光照强度和土壤湿度等关键环境参数，为后续手动控制与自动策略提供数据基础。

（2）远程手动控制：用户可在 HarmonyOS 移动端通过滑块或按钮对补光灯、风扇、水泵等执行器进行远程控制。

（3）智能自动调节：用户设定阈值后，系统依据实时数据执行自动控制逻辑，实现温室环境的无人值守调节。

（4）稳定可靠通信：基于 MQTT 发布/订阅机制构建双向通信链路，并引入心跳保活、看门狗离线检测与自动重连机制。

（5）良好用户体验：应用端遵循 HarmonyOS 设计语言，支持响应式布局、日志可视化与沉浸式动效，使系统既可演示、也便于调试。

3．系统架构

系统采用“嵌入式端—通信服务—应用端”的三层架构。嵌入式端负责环境感知与执行控制，通信服务负责消息中转与发布/订阅路由，应用端负责数据展示、用户交互和智能决策。三层之间通过 MQTT 主题实现松耦合协同，避免了硬件端与移动端之间的 IP 直连依赖。

层级	核心组成	主要职责
嵌入式端	Rayawa/rgb、Rayawa/soi；传感器与执行器	采集温湿度、光照、土壤湿度；控制补光灯、风扇、水泵；OLED 显示状态
通信服务	EMQX 公网 Broker；MQTTX 调试工具；WebSocket/TCP 接入	完成主题路由、消息转发、连接调试与通信验证
应用端	HarmonyOS 6.1.0；ArkTS；ArkUI；HDS Design	实时数据显示、手动控制、智能策略、日志监控与跨设备适配

表 1 系统三层架构组成与职责说明

4. 开发环境与技术栈

本系统技术栈分为通信服务、嵌入式端与应用端三部分，主要软硬件平台、操作系统、工具链及通信组件配置如下：

类别	技术选型	说明
通信服务	EMQX Broker / MQTTX / broker.emqx.io:8084	用于发布/订阅中转、调试与消息路由验证
嵌入式硬件	HiSilicon Hi3861，RISC-V，160KB SRAM	作为底层感知和执行控制核心
嵌入式系统	OpenHarmony 1.0 Release / LiteOS-M	提供轻量级实时运行环境
嵌入式开发	C、Paho MQTTPacket、lwIP、GN、hb、HiBurn	完成外设驱动、MQTT 报文处理与编译烧录
应用端系统	HarmonyOS 6.1.0 Release（API 23）	运行于手机和平板等鸿蒙终端
应用端开发	DevEco Studio 6.1.0、ArkTS、ArkUI、Hvigor	完成页面、状态管理、路由与网络通信
UI 与交互	HDS Design System、UIDesignKit、Sensor Service Kit	完成沉浸式材质、点光源动效与触感反馈

表 2 开发环境与技术栈配置说明

二、MQTTX 通信服务设计

1. 发布/订阅架构

本系统采用经典的发布/订阅架构，发布者与订阅者不直接感知彼此的 IP 地址和连接状态，而是通过 Broker 完成消息路由与转发。该机制适合智慧大棚这种多端协同场景：硬件端需要持续上报传感器数据，应用端需要随时下发控制指令，二者在时序与网络环境上均可能不同步。

角色	本项目中的对应对象	功能说明
Publisher	Hi3861 嵌入式端 / HarmonyOS 客户端	Hi3861 发布传感器 JSON 数据；客户端发布风扇、补光灯、水泵控制指令
Broker	EMQX 公网服务器	负责主题匹配、消息过滤与转发，解耦设备端与应用端
Subscriber	HarmonyOS 客户端 / Hi3861 嵌入式端	客户端订阅环境参数；Hi3861 订阅控制主题并执行外设动作

表 3 MQTTX 通信架构角色说明

在课程演示与调试中，MQTTX 承担了通信服务连接验证与消息观测功能；在系统运行

过程中，EMQX Broker 负责实际的主题路由与数据转发。由于环境监测数据具有较强时效性，系统更关注“最新状态是否能及时到达”，因此整体设计倾向于轻量、低延迟和可快速恢复的通信模式。

2. 数据流展示

本系统采用 MQTT 协议作为轻量级数据传输载体，通过 MQTTX 代理（Broker）实现了 Hi3861 硬件设备端与 HarmonyOS 移动应用端的高效双向异步通信。具体通信流转过程如下：

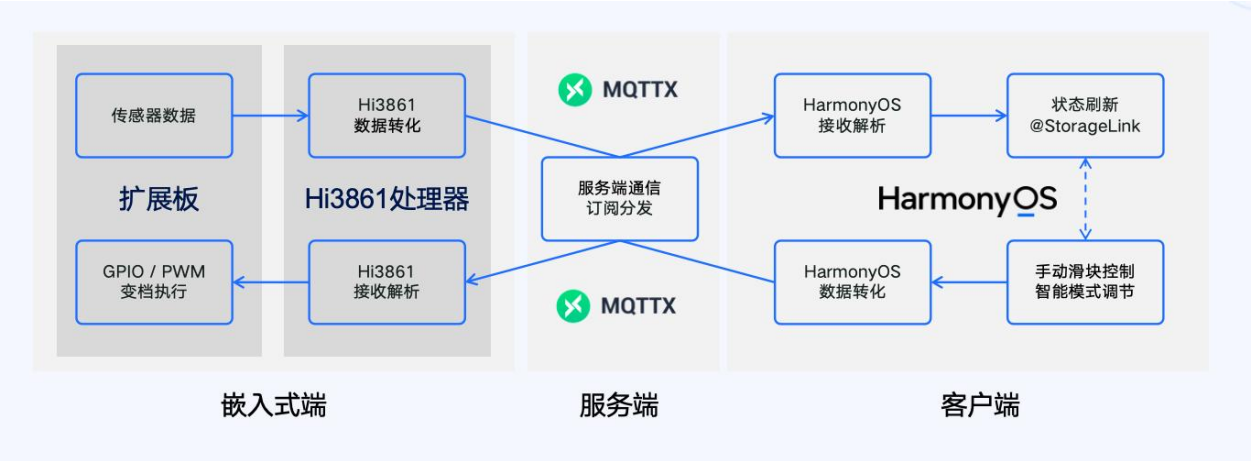

图 1 MQTTX 通信上下行数据流程示意图

（1）连接建立阶段：系统初始化后，Hi3861 设备端与 HarmonyOS App 分别配置对应的 Host 地址、端口号、客户端 ID（ClientID）及鉴权信息，向 MQTTX Broker 发起连接请求，建立稳定的 TCP 长连接。

（2）上行数据流（设备监测）：Hi3861 设备端通过“MQTT 客户端”模块，将采集到的环境或传感器数据封装为 JSON 格式的 Payload（例如 {"temp": 25.6}），并指定主题 hi3861/data 向上发布（Publish）。同时，HarmonyOS App 提前订阅（Subscribe）了该主题。MQTTX Broker 触发主题路由与消息转发机制，将该数据实时推送至 App 端，应用端接收解析后更新用户界面。

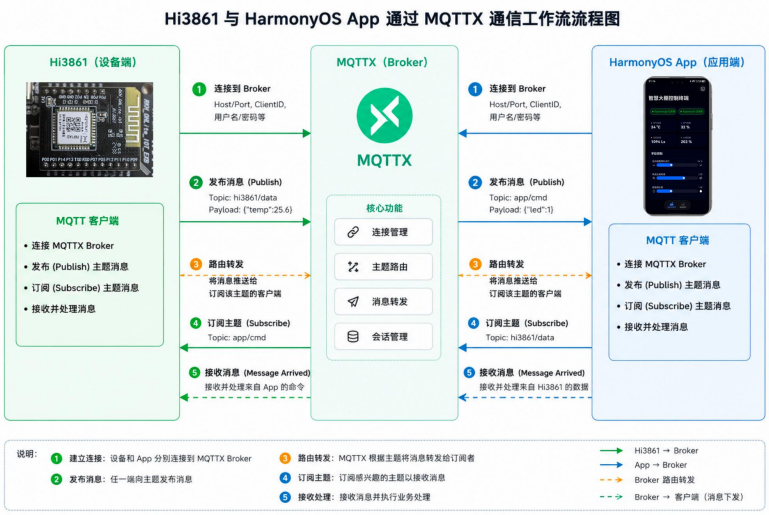

图 2 MQTTX 通信详细流程图

（3）下行控制流（应用反控）：当用户在 HarmonyOS App 上进行控制操作时，App 端向主题 app/cmd 发布控制指令（例如 {"led": 1}）。由于 Hi3861 设备端已处于该主题的订阅状态 MQTTX Broker 将指令准确转发至硬件端，硬件端接收解析 (Message Arrived) 后立即执行相应的硬件外设驱动逻辑。

上行采集流与下行控制流共同构成智慧大棚的闭环。上行链路解决“环境状态能否被移动端及时观察”的问题；下行链路解决“用户操作或自动策略能否驱动硬件执行”的问题。两条链路均通过统一的 Topic 命名空间进行区分，从而保证系统可调试、可扩展。

三、Hi3861 嵌入式端设计

1. 总览

Hi3861 嵌入式端承担系统底层环境感知、设备控制以及数据通信任务，是连接物理环境与上层应用的核心节点。由于单块扩展板的引脚资源有限，且多个传感器与执行器存在接口冲突，本项目采用双开发板并行方案：Rayawa/rgb 负责空气温湿度、光照与补光灯；Rayawa/soi 负责土壤湿度、水泵和风扇。两块板通过同一 Broker 与 HarmonyOS 应用端通信，不需要直接进行板间通信。

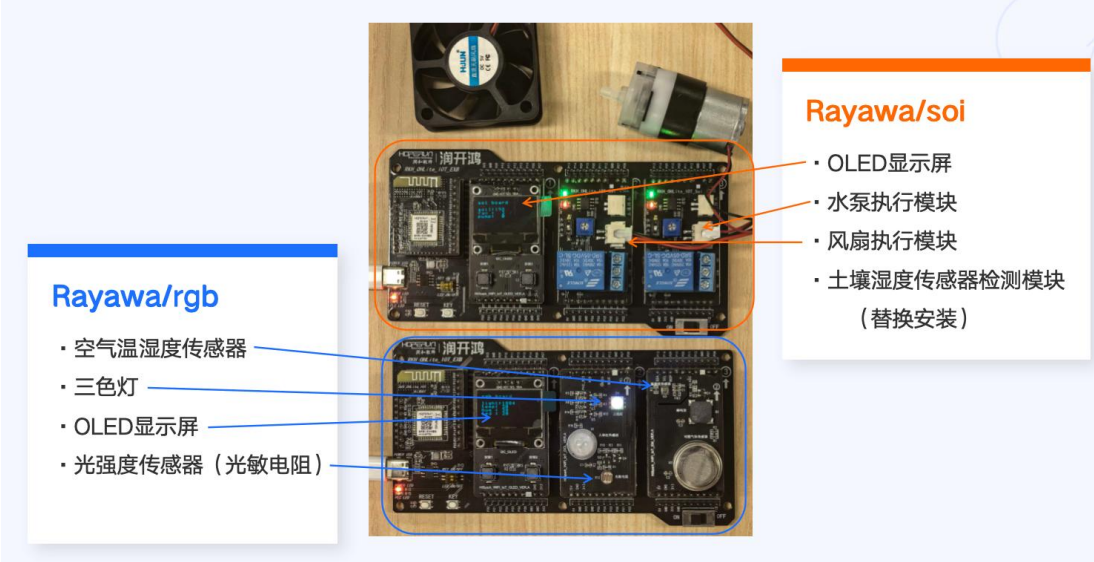

图 3 嵌入式端系统结构总览

各元件功能、对应接口与控制程序如下表所示：

元件	功能	硬件接口	控制程序
OLED	状态打印	I2C0-0x78 GPIO13&14	oled_ssd1306.c
空气温湿度传感器	检测空气参数	I2C0-0x44	temp_and_hum_task.c
光敏电阻	检测光照强度	ADC_CHANNEL_4	light_intensity_task.c
三色灯(RGB)	大棚光照控制	GPIO10/11/12 => PWM1/2/3	led_task.c

Rayawa/rgb

元件	功能	硬件接口	控制程序
OLED	状态打印	I2C0-0x78 GPIO13&14	oled_ssd1306.c
土壤湿度传感器	外接土壤探头	ADC_CHANNEL_4	soil_moisture_task.c
水泵	外接水泵	(P06)	water_pump_task.c
风扇	外接风扇	(P08)	fan_task.c

Rayawa/soi

表 4 Hi3861 外设接口与功能模块对应关系

2．项目结构

嵌入式端软件采用“大分功能、小分模块”的文件组织方式，自底向上划分为公共通用层、业务模块层和开发板构建层。该结构将 Wi-Fi、MQTT、OLED 等通用能力与具体传感器/执行器业务分离，使不同板卡只需切换 GN 构建配置即可完成编译适配：

```
applications/sample/wifi-iot/app/workspaces/smart_shed_2.0/
├── common/
│   ├── build/                # GN 构建共享片段
│   │   └── smart_shed_common.gni
│   ├── runtime/              # 运行时核心
│   │   ├── main.c            # 入口，启动所有线程
│   │   ├── smart_shed_mqtt.c  # MQTT 连接、发布、订阅
│   │   ├── smart_shed_wifi.c  # Wi-Fi 联网
│   │   ├── smart_shed_oled.c  # OLED 显示
│   │   └── smart_shed_shared.c # 共享资源（I2C 锁等）
│   └── display/
│       ├── oled_ssd1306.c    # SSD1306 驱动
│       └── oled_fonts.h      # 字库
├── modules/
│   ├── sensors/
│   │   ├── temp_and_hum/     # AHT20 温湿度传感器
│   │   ├── light_intensity/  # 光强传感器（ADC）
│   │   └── soil_moisture/    # 土壤湿度传感器（ADC）
│   └── actuators/
│       ├── fan/              # 散热风扇控制
│       ├── led/              # 补光灯控制
│       └── water_pump/       # 潜水泵控制
└── boards/
    ├── env_rgb_board/        # Board 1 构建目标
    └── soil_actuator_board/  # Board 2 构建目标
```

图 4 嵌入式端软件模块结构图

（1）公共通用层负责统一管理 Wi-Fi 连接、MQTT 通信及 OLED 驱动，并通过共享资源 I2C 锁机制确保多模块并发访问时的系统运行效率与稳定性。

（2）业务模块层分为传感器与执行器两个子模块。传感器目录存放温湿度传感器、光强传感器及土壤湿度传感器的头文件与程序源文件；执行器目录则控制风扇、补光灯和潜水泵等设备。

（3）开发板构建层包含两个开发板的 GN 构建配置的统一管理。该层设计尤为关键——它决定了系统在面对硬件平台迭代或扩展时的适配成本。统一的 GN 管理使得新增开发板仅需补充对应构建配置，无需改动上层业务逻辑，从而显著提升系统的可扩展性与维护效率。

3．程序架构设计

嵌入式端采用基于 LiteOS 实时内核的多线程结构。系统启动后由 SYS_RUN 入口函数创建主线程 smart_shed_all.c，主线程栈大小为 8KB，随后按板卡条件编译配置拉起传感器采集、执行器控制、MQTT 通信和 OLED 显示等子线程。各线程拥有相对独立的栈空间和优先级，从而降低单一任务阻塞对系统整体运行的影响。

各功能模块通过全局变量进行轻量级通信。MQTT 线程收到控制主题的 PUBLISH 报文后，调用解析函数取得 Topic 与 Payload，再通过 mqtt_apply_command()修改 fan_level、led_level、water_pump_level 等全局变量；执行器线程以固定周期读取这些变量，并据此控制 GPIO 或 PWM 输出。该设计属于简单有效的生产者—消费者模型，将网络通信时序与硬件执行时序解耦。

由于 AHT20 温湿度传感器与 OLED 显示屏共用 I2C0 总线，系统引入 i2c0_lock/unlock 互斥锁机制。所有 I2C 读写操作在访问前加锁、访问后释放，保证同一时刻只有一个设备占用总线，避免并发读写造成的数据冲突、显示异常或传感器通信失败。

4．WiFi 与 MQTT 通信

设备连接 SSID 为“Rayawa”的无线网络，通过 DHCP 动态获取 IP 地址后，进入 lwIP 协议栈的 TCP 通信阶段。本方案中，Paho 库仅负责 MQTT 报文的编解码工作，TCP Socket 的完整生命周期（包括创建、连接、收发、关闭及断线重连）由开发者自行管理。

```c
int packet_type = MQTTPacket_read(buf, sizeof(buf), transport_getdata);
if (packet_type == PUBLISH) {
    unsigned char dup = 0;
    int qos = 0;
    unsigned char retained = 0;
    unsigned short msgid = 0;
    int payload_len = 0;
    unsigned char *payload_in = NULL;
    MQTTString received_topic = MQTTString_initializer;
    char topic[32];

    if (MQTTDeserialize_publish(&dup, &qos, &retained, &msgid, &received_topic,
        &payload_in, &payload_len, buf, sizeof(buf)) == 1 &&
        payload_len > 0 && payload_len < (int)sizeof(payload)) {
        int topic_len = received_topic.lenstring.len;
        if (topic_len >= (int)sizeof(topic)) {
            topic_len = sizeof(topic) - 1;
        }
        memcpy(topic, received_topic.lenstring.data, topic_len);
        topic[topic_len] = '\0';
        memcpy(payload, payload_in, payload_len);
        payload[payload_len] = '\0';
        printf("MQTT cmd topic=%s payload=%s\n", topic, payload);
        mqtt_apply_command(topic, payload);
    }
}
```

图 5　Hi3861 MQTT 通信处理流程

传感器数据以 JSON 格式封装并发布至指定主题；应用端下发的控制指令由 Hi3861 订阅后解析执行。

收到 PUBLISH 报文时，系统调用 MQTTDeserialize_publish() 解析主题与负载，再通过主题匹配规则更新全局控制变量。独立执行器线程持续轮询这些变量，并根据当前数值驱动风扇、补光灯、水泵等设备，完成从通信报文到实际硬件动作的转换。

四、HarmonyOS 客户端设计

1. 总览

Smart Shed App 是系统的上位机控制终端，面向 HarmonyOS 6.1.0 Release（API 23）开发，使用 DevEco Studio 6.1.0 Release 作为主要开发环境。应用完全采用 ArkTS 与 ArkUI 声明式框架实现，并结合 HDS Design System 构建符合鸿蒙生态视觉语言的交互界面。

图 6 Smart Shed 移动应用图标

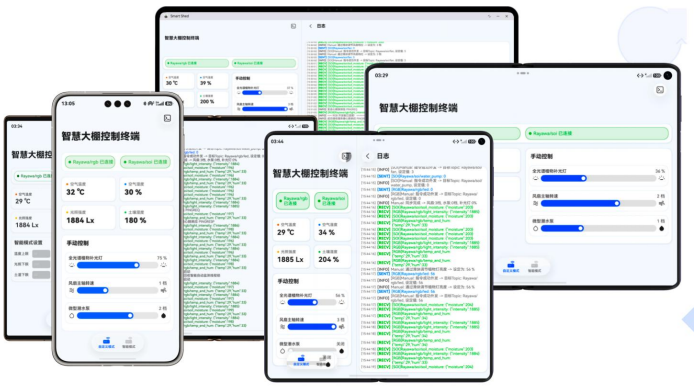

图 7 HarmonyOS 多设备响应式布局示意图

模块	说明
自定义模式	通过滑块手动调节补光灯亮度、风扇档位和水泵档位，适合演示、调试和精细控制场景
智能模式	用户设置温湿度、光照和土壤湿度阈值，系统按定时策略自动下发控制指令
日志控制台	展示 MQTT 发送、接收、连接、断连、错误等日志，并在平板端支持分栏显示
响应式布局	根据设备宽度自动切换手机单栏和平板分栏布局，提升多设备适配能力

表 5 HarmonyOS 应用主要功能模块说明

2．项目结构

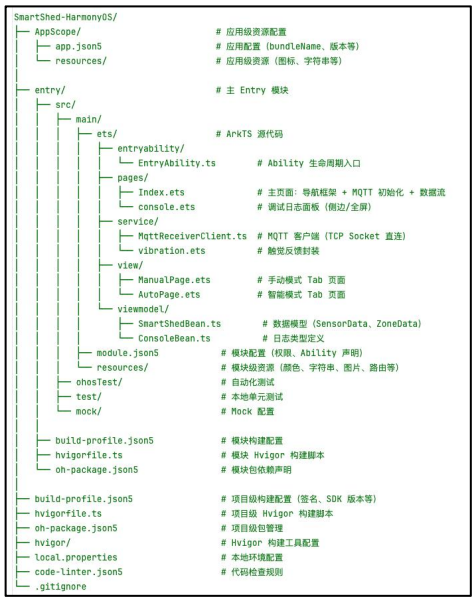

图 8 应用端项目结构示意图

鸿蒙应用端采用清晰的分层架构，主要包含以下四个核心目录。

AppScope 目录负责统一管理应用级全局资源与 Hvigor 构建脚本，为整个应用提供基础的配置与环境支持。

view 与 pages 目录基于 ArkTS 语言，构建了包含手动控制与智能自动双模式、调试面板在内的流畅用户界面。

viewmodel 目录用于定义核心数据结构，驱动应用状态的安全传递与界面间的数据同步。

service 目录封装了 MQTT 客户端，通过 TCP Socket 直连方式实现与云端的数据实时双向通信。

3．MQTTX 客户端逻辑

（1）申报机制

申报机制包括订阅与发布两个环节。订阅时发送 0x20 报文，调用 subscribeToTopic 函数并拼装 0x82 响应；发布时通过控制报文 0x30 调用 publish()函数，底层经由 TCP Socket 传输。在数据的接收与路由识别方面，系统通过 SocketInstance 监听 message 事件，调用 handleMqttMessage 函数，提取控制字节并与 0xF0 进行按位与操作，经由流式解码器解析出主题与负载，最后通过 this.callback(topic, payload)完成回调分发。

（2）看门狗机制

看门狗机制的核心在于喂狗操作。每次接收到有效数据时，分别更新 lastrgbDataTime 与 lastsoiDataTime 为当前时间戳。系统设置 BOARD_TIMEOUT_MS 为 5000 毫秒，并通过 this.boardOfflineCheckTimer 定时器（间隔 1000 毫秒）定时检查设备在线状态。当超时触发时，将 isBoardRgbConnected 标志位置为 false，并记录 LogType.ERROR 级别日志，实现状态级联响应。

（3）心跳保护机制

心跳保护机制首先涉及心跳周期的设计。系统将心跳周期设置为保活时间（keepAliveSeconds）的 0.75 倍，以容忍网络传输延迟，确保服务器不会因网络抖动而误判客户端离线。在双向握手验证方面，系统维护 PingOutstanding 旗标，客户端发送 0xC0 心跳请求，服务器收到后立即回复 0xD0 响应。若超时未收到响应，触发超时惩罚，执行状态清理后启动异步退避重连，再次完成 TCP 握手。

（4）报错与自动重连机制

报错与自动重连机制实现三层异常捕获，包括 Socket 链路异常、Socket 关闭通知及 Promise 异常。状态清理阶段调用 disconnectInternal()函数，释放 Socket 连接与心跳定时器。随后启动异步退避重连，通过 setInterval 结合 setTimeout 方式，每隔 5000 毫秒尝试一次 TCP 握手，直至连接恢复。

4. 状态管理与数据模型

（1）总览

Smart Shed 客户端的状态管理围绕三个层次展开：AppStorage 全局状态容器作为跨组件数据总线；ZoneData 与 SmartShedBean 等模型类定义数据结构约束；GlobalLogBus 单例实现日志事件的发布-订阅解耦。

（2）全局状态与数据模型

应用涉及传感器数值、执行器档位、设备在线状态、自动模式开关、导航栈状态及日志数据等多类状态。若状态分散在各页面中独立维护，页面切换将导致数据丢失或控制值不同步，且多个页面需重复订阅同一 MQTT 主题。因此本项目以 AppStorage 作为全局状态容器，在 Index 主页面 aboutToAppear 中统一初始化 isBoardRgbConnected、isBoardSoiConnected、autoEnabled、temp、hum、intensity、moisture、fanValue、waterPumpValue、ledValue 及 zone1 等键值。各子页面通过@StorageLink 装饰器建立与全局状态的双向绑定——任意一处修改，所有绑定组件同步更新，无需手动传递回调或事件。

在数据模型层，项目定义了三个核心类型。SmartShedBean 承载单区传感器读数（温度、湿度、光照、土壤湿度）与对应的阈值上下限，以及执行器当前档位；ZoneData 在 SmartShedBean 基础上聚合手动控制值（fanValue、waterPumpValue、ledValue）、自动模式阈值字符串（autoTempMin/Max 等六项）及自动控制定时器句柄 autoManagerThread，作为 AppStorage 中 zone1 的实际类型；SensorData 接口约束 MQTT 上行 JSON 的字段结构。ConsoleBean 定义 LogType 枚举（RECEIVE、SEND、INFO、ERROR）与 LogMessage 接口（id、time、type、content），统一日志数据结构。通过将业务数据抽象为模型，UI 层仅关注展示与交互，服务层处理 MQTT 通信，模型层约束数据格式，各层职责清晰。

（3）参数传递

组件间参数传递主要包括页面可见性传递、控制值传递和日志事件传递三类。

ManualPage 与 AutoPage 通过@Prop 接收 Index 主页面传入的 isVisible 状态，并以@Watch('onVisibleChange')监听其变化。当页面可见时：ManualPage 将 autoEnabled 置为 false 以关闭自动模式，同步 AppStorage 中的执行器值并通过 safePublish()下发至设备，同时停止自动控制定时器并启动分层入场动画；AutoPage 从 ZoneData.bean 同步最新传感器读数，若 autoEnabled 已开启则重启自动控制循环，同时触发入场动画。当页面不可见时，两页面均重置动画状态变量（animStep1/2/3 = 0）并停止动画定时器，AutoPage 额外调用 stopAutoLogic()终止自动控制循环，避免后台定时器空转。

控制值传递依赖 AppStorage 的共享状态机制。手动模式中，Slider 组件仅在 onChangeMode 为 SliderChangeMode.End 或 Click 时触发更新，而非拖拽过程中持续发布，从而降低 MQTT 通信频率。滑块变更同时更新 AppStorage 中的 fanValue、waterPumpValue 与 ledValue 并调用 safePublish()下发指令。智能模式中，自动控制循环直接修改同一组 AppStorage 字段并发布控制报文。因此，无论控制来源是手动拖拽还是自动策略，最终均落到同一套状态字段和发布接口，保证控制链路一致。

日志传递通过 GlobalLogBus 单例实现解耦。该总线提供 register()、unregister()、emit()三个静态方法，Index 在 aboutToAppear 中注册回调将日志推入 logList（上限 100 条）MqttReceiverClient 通过 setLogCallback()接入总线，ManualPage 与 AutoPage 中的 safePublish()也直接调用 GlobalLogBus.emit()写入发送日志。ConsolePage 无需直接依赖 MQTT 客户端，仅消费 logList 即可展示完整通信记录。

(4) 全局日志同步

ConsolePage 中的 LogDataSource 实现 IDataSource 接口，内部维护 LogMessage 数组并以 notifyDataReload()通知 LazyForEach 刷新。页面通过@Watch('onLogListChange') 监听 logList 变化，在数据源重载后若用户未手动上滑（isAtBottom 为 true），则通过 Scroller.scrollEdge(Edge.Bottom) 自动滚动至最新日志。用户上滑浏览历史日志时 isAtBottom 标记为 false，暂停自动滚动；回到底部后恢复。

每条日志以 Row 布局展示时间戳（10px）、类型标签[SENT]/[RECV]/[INFO]/[ERR]（12px 加粗）和正文内容。正文按设备来源自动携带 [RGB] 或 [SOI] 前缀，类型标签颜色按 LogType 区分——SEND 为蓝色、RECEIVE 为绿色、INFO 为灰色、ERROR 为红色，且颜色随自动模式开关在亮/暗两套色值间切换。List 组件设置 cachedCount(5) 以优化 LazyForEach 的滑动性能。

(5) 生命周期管理

生命周期管理保证页面切换、应用退出与通信资源释放之间的一致性。

Index 主页面在 aboutToAppear 中依次执行：注册 GlobalLogBus 日志回调；创建 MqttReceiverClient 实例并传入 MQTT 消息回调 onMqttMessage()；通过 setLogCallback() 将 MQTT 客户端日志接入全局总线；调用 onStart()启动 TCP 连接与 MQTT CONNECT 流程；根据屏幕宽度判断是否以分栏模式打开 ConsolePage。在 aboutToDisappear 中，先将 zone1.isExit 置为 true 通知自动控制循环退出，清除 autoManagerThread 定时器，调用 GlobalLogBus.unregister()移除日志回调，最后执行 mqttClient.disconnect()发送 MQTT DISCONNECT 报文并清理心跳定时器、板载看门狗定时器及重连定时器。

ManualPage 与 AutoPage 依据 isVisible 变化控制页面动画与自动逻辑的启停 ConsolePage 通过 HdsNavDestination 接入 NavPathStack 导航栈，在显示时自动滚动至日志底部，在返回时调用 vibration.ets 中的 triggerVibration()触发触感反馈并执行 pageStack.pop()回退导航栈。

5. 智能控制策略

智能模式通过周期性阈值比较实现自动调节。AutoPage 在 autoEnabled 开启后启动 setInterval 定时器，每 2 秒读取最新环境数据，与用户设定的上下限进行比较，并根据触发条件控制对应执行器。该策略属于轻量级边缘决策：决策运行在移动端，执行动作下发到 Hi3861，适合课程设计中展示“感知—判断—执行”的完整闭环。

触发条件	控制动作	设计意图
温度高于上限	逐级开启风扇	降低棚内温度，避免过热
空气湿度高于上限	逐级开启风扇	促进空气流动，降低过湿风险
土壤湿度低于下限	逐级开启水泵	补充土壤水分，维持植物生长环境
土壤湿度高于上限	关闭水泵并可开启通风	避免过度浇灌与土壤积水
光照低于下限	按比例开启补光灯	补足光照强度，提升光环境稳定性
参数恢复正常	关闭对应执行器或保持低档	减少不必要能耗并避免设备频繁工作

表 6 智能控制策略设计说明

当所有传感器参数回落至正常范围时，对应执行器目标值保持默认值 0，不发送控制报文。该策略属于轻量级边缘决策：决策逻辑运行在移动端，无需云端参与；执行指令通过 MQTT 下发至 Hi3861 开发板执行，形成"感知—判断—执行"的完整闭环，适合在课程设计中展示端-边协同的物联网控制范式。

安全防护方面，应用端通过 Slider 组件的 min/max 属性约束手动控制值范围，通过 parseInt 与默认值兜底防止非法阈值输入；设备端对收到的控制报文进行简单校验 MqttReceiverClient 内置板载看门狗机制——以 1000ms 间隔检查各板卡最近数据到达时间，若超过 BOARD_TIMEOUT_MS（5000ms）未收到上报，则将对应 AppStorage 连接状态标记为 false 并通过日志输出错误信息，提示用户当前自动决策可能缺少关键数据来源。此外，MQTT 协议层的心跳看门狗以 keepAlive × 0.75 的间隔（15s）发送 PINGREQ，若连续两次未收到 PINGRESP 则判定连接断开并触发 5s 延迟重连。

6. UI 界面设计

Smart Shed App 的 UI 设计以“数据清晰、控制直观、调试可见”为核心原则。页面整体基于 HDS 语义化色彩与 ArkUI 声明式组件构建，既保留鸿蒙系统级视觉一致性，也针对智慧大棚场景加入生态绿、科技灰蓝和警告红等语义色。

设计维度	实现方式	效果
色彩方案	color.json 与页面自定义颜色；生态绿表示在线与正常；警告红表示断连或错误	状态识别直观，符合用户对设备运行状态的判断习惯
字体层级	核心数值 22px 粗体；模块标题 17px 强调体；辅助信息 12-14px 常规体	保障温度、湿度、光照等关键数据一目了然
图标系统	app.media 业务图标 + sys.symbol 系统符号	兼顾业务辨识度与鸿蒙原生美学
日志布局	LazyForEach + Scroller；自动滚动到底部；颜色区分发送、接收、错误	提升调试效率，减少通信问题排查成本
响应式布局	onAreaChange 监听宽度；宽屏切换控制面+日志面板分栏	适配手机和平板，提升大屏展示效率

表 7 Smart Shed 用户界面设计方案

响应式布局是本项目 UI 设计的关键点之一。窄屏设备保持单页控制逻辑，避免界面拥挤；当容器宽度达到平板阈值后，界面自动切换为左侧控制面板、右侧日志控制台的分栏布局。分栏切换使用 springMotion 曲线和 navBarWidth 宽度变化实现，使布局变化更平滑。

动效方面，系统主要包括三类：第一，引力动效，即页面进入时采用分层弹簧动画淡入上移，点击控件时配合 scale 缩放和触感反馈；第二，光场动效，即 HDS 材质与 PointLight 点光源在触摸时提供局部照亮效果；第三，转场动效，即智能模式和控制台页面激活时启用背景流光或组件淡入淡出，增强场景切换的层次感。

7. Smart Shed Glass

Smart Shed Glass 是本项目在 UI 交互设计方向上的扩展版本，主要针对智慧大棚控制场景中的信息展示与人机交互体验进行优化。与普通版本侧重功能完整性不同，Glass 版本在保持传感器数据显示、设备控制和通信监测等核心功能的基础上，引入 HDS Design System 提供的材质效果与交互能力，通过玻璃化视觉效果、动态反馈和组件化设计提升应用整体的交互表现。

图 9 Smart Shed Glass 应用图标

普通版本主要关注环境数据采集、设备控制和通信调试等基础功能，而 Glass 版本在保持上述功能完整性的基础上，结合 HDS Design System 提供的系统级材质效果与交互能力，对控制界面进行了视觉和交互优化。通过玻璃化材质、动态反馈以及组件化设计，使温湿度、光照、设备运行状态等信息能够以更加清晰的方式呈现，同时增强用户对设备操作结果的感知。

Smart Shed Glass 的设计核心是将功能组件、状态信息和交互反馈进行统一设计，使用户能够在查看环境参数和控制设备时获得更加明确的操作反馈。项目通过自定义 HdsButton 组件构建统一的玻璃材质交互组件，将按钮、控制卡片以及状态展示区域进行抽象封装。相比传统 UI 中每个页面独立编写样式和动画逻辑的方式，该设计能够降低界面开发复杂度，提高后续功能扩展时的复用能力。

HdsButton 组件采用三层 Stack 结构实现。其中，底层为系统材质容器，通过 barFloatingStyle 结合 systemMaterialEffect 实现基础玻璃材质效果，并通过 materialType 与 materialLevel 参数控制不同材质等级和透明效果。相关参数通过@StorageLink 与 AppStorage 中的全局状态绑定，使 ManualPage、AutoPage 和 ConsolePage 等不同页面能够共享统一的材质配置，避免多个页面之间出现视觉风格不一致的问题。

中层为业务内容区域，通过@BuilderParam buttonContent 接收外部传入的自定义内容，使同一组件能够根据不同业务需求显示控制按钮、传感器数据卡片、导航入口等不同界面元素。上层为交互响应层，通过 TouchEvent 与 PanGesture 监听用户输入，并根据组件配置启用不同类型的动态反馈。

在交互设计方面，HdsButton 主要实现三类反馈机制：

（1）拖拽形变效果。组件通过 PanGesture 获取用户操作过程中的位移变化，并将位移映射至 offset 和 scale 参数，使组件在拖动过程中产生轻微视觉形变；用户释放后，通过 responsiveSpringMotion 弹簧曲线恢复初始状态，使交互过程更加自然。

（2）点击缩放效果。用户触摸组件时，组件 scale 参数短暂增加至 1.05，释放后通过弹簧动画恢复原状态，使按钮操作具有明确的触控反馈。

（3）点光源效果。组件通过 HdsEffectBuilder().pointLight()调用系统光效能力，在触摸位置产生局部光照变化。其中光源类型设置为 SOFT，照明模式设置为 BORDER_CONTENT，高度为 150px，强度为 1.0，用于增强玻璃材质的动态表现，使用户操作过程能够获得即时视觉反馈。

基于上述组件化设计，ManualPage 中的设备控制区域、AutoPage 中的智能策略卡片以及 ConsolePage 中的返回入口均复用了 HdsButton 组件模板。该设计使新增传感器模块、执行器控制区域或多区域管理界面时，只需要传入对应业务内容即可继承统一视觉效果和交互逻辑，降低后续功能扩展和界面维护成本。

除基础组件效果外，Smart Shed Glass 还针对不同功能场景设计了页面级动态效果。AutoPage 在智能模式开启时，通过 hdsEffect.EffectType.DUAL_EDGE_FLOW_LIGHT 实现双色边缘流光效果，用于强化自动控制状态识别；ConsolePage 启用 UV_BACKGROUND_FLOW_LIGHT 背景效果，并为日志列表项增加 pointLight 反馈，提高通信调试界面的层次感和可读性；Index 主页面标签栏结合 BlurStrategy.ENABLE 与 IMMERSIVE+EXQUISITE 材质效果，实现导航区域与内容区域之间的视觉统一。

此外，Smart Shed Glass 还体现了 HarmonyOS 声明式 UI 框架在物联网场景中的适应性。由于智慧大棚控制终端可能运行于手机、平板等不同尺寸设备，本项目在界面设计过程中保留了响应式布局思想，使控制区域能够根据设备屏幕尺寸自动调整展示方式。在较小屏幕设备上，系统采用单页面布局保证操作集中；在较大屏幕设备上，则可以结合分栏方式同时展示控制区域和日志信息，提高调试和演示效率。

从工程实现角度来看，Smart Shed Glass 并非单纯增加视觉效果，而是在保持功能逻辑不变的情况下，对 UI 组件、状态管理和交互反馈进行了进一步抽象。通过统一组件封装和效果复用，使应用具备更好的维护性和扩展能力。当未来增加新的传感器类型、执行器设备或多个温室区域管理功能时，只需要扩展业务数据模型和页面内容，即可快速复用已有交互框架。

通过 Smart Shed Glass 设计，本项目不仅完成了智慧大棚控制终端的功能实现，同时探索了 HarmonyOS 在物联网设备交互场景中的设计方式。该方案在保证系统功能可靠性的同时，提高了应用的可维护性、可扩展性和展示效果，为后续多设备、多区域智慧农业场景扩展提供了界面设计基础。

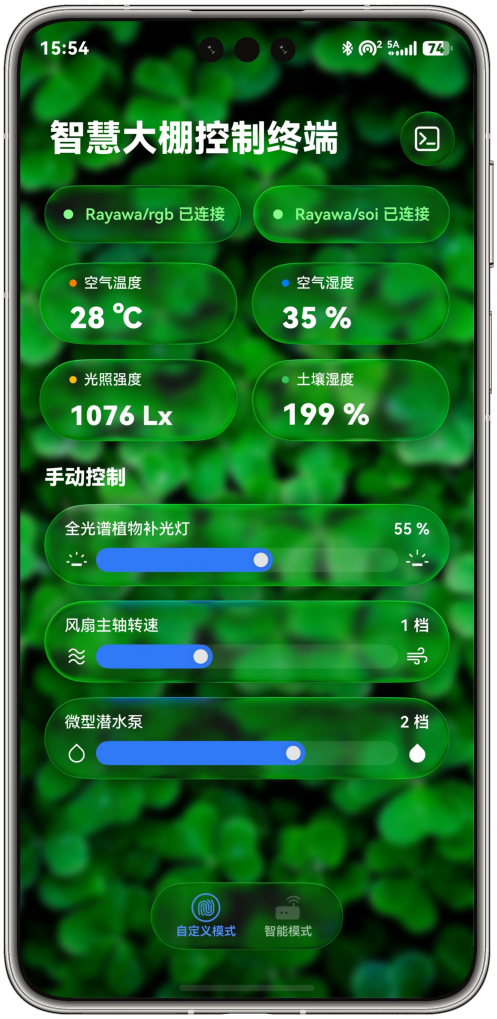

图 10 主页面玻璃效果

五、项目创新点总结

1. 嵌入式端创新

本项目在嵌入式端采用双开发板并行设计，解决了单块扩展板引脚冲突和负载能力不足的问题。Rayawa/rgb 与 Rayawa/soi 分别承担感知和执行任务，通过同一 Broker 协同工作。项目结构采用 common、modules、boards 三层组织，新增或替换板卡时只需调整 GN 构建配置，具有较好的可维护性和扩展性。

2. 通信架构创新

系统摒弃传统热点直连方式，采用 Broker 中转的发布/订阅架构。该方案不要求设备端与应用端处于固定局域网或直接互相发现，调试时也可以通过 MQTTX 快速观察主题数据。应用端进一步加入看门狗、心跳、异常捕获和自动重连逻辑，使通信链路从“能跑通”提升到“可观察、可恢复、可定位问题”。

3. 应用端创新

应用端由传统 Java UI 开发方式迁移至 ArkTS 与 ArkUI 技术体系。系统引入 HDS 设计语言、响应式分栏、日志控制台、全局状态管理和 Smart Shed Glass 视觉体系，实现了功能性与展示性的统一。尤其是日志同步与分栏布局，使课程答辩和现场调试更直观。

4. 技术难点与解决思路

本项目涉及嵌入式硬件、系统应用、网络通信与跨设备适配四个技术层面，各层面存在不同的技术难点。硬件层面，单块 Hi3861 开发板的引脚资源与计算能力有限，难以同时驱动多类传感器与执行器；软件层面，需从已停维护的 API 7 Java UI 框架迁移至最新的 HarmonyOS 6.1.0 ArkTS 技术栈；通信层面，基于裸 TCP 实现的 MQTT 协议在公网环境下面临断连、丢包与调试信息不足的问题。以下按问题类别分别说明解决思路：

问题类别	具体问题	解决思路
嵌入式源码	示例代码模块独立且引脚定义冲突，单 Hi3861 板难以同时运行多个元器件	重构为双板架构与模块化目录，通过条件编译选择板卡功能
传感器与执行器	土壤湿度传感器、风扇、水泵等外设驱动和接口适配存在问题	按硬件接口重新划分 GPIO/ADC/PWM 控制程序，并分板运行
应用端	旧文档基于 API 7 和 Java UI，界面适配能力弱	升级为 HarmonyOS 6.1.0 + ArkTS + ArkUI + HDS Design
通信调试	MQTT 传输不稳定，报错信息不足	加入日志控制台、看门狗、心跳保活、异常捕获与自动重连机制
用户体验	移动端需要同时满足手机与平板展示	通过 onAreaChange 实现响应式分栏，并统一字体、色彩和交互动效

表 8 项目技术难点与解决方案

六、总结与展望

本课程设计围绕智慧农业大棚场景，完成了从 Hi3861 嵌入式硬件、MQTTX/EMQX 通信服务到 HarmonyOS 移动应用端的全链路设计与实现。系统能够实现环境参数实时采集、执行器远程控制、智能阈值调节、日志可视化和多设备界面适配，较完整地体现了鸿蒙项目制综合实践中的软硬协同、端云协同和移动端交互设计能力。

从工程完整性看，项目的主要价值在于将多个分散模块整合为一套可演示、可调试、可扩展的物联网系统。嵌入式端通过模块化和双板架构提升稳定性；通信端通过发布/订阅机制降低耦合；应用端通过全局状态、日志控制台和响应式布局提升用户体验。

未来仍可从三个方向继续改进：第一，在安全层面引入 MQTT over TLS、用户名密码认证、设备证书白名单和访问控制列表；第二，在自治能力层面，将部分兜底策略下沉到 Hi3861 本地，使断网情况下仍可根据传感器数据执行基本保护动作；第三，在农业应用层面，增加多区域 zone 模型、历史数据存储、趋势图表和更细粒度的作物生长策略，使系统从课程原型进一步走向实际应用。

——2026.6.6

致　谢

本项目的顺利完成，离不开多位师长、同学与家人的支持与帮助，在此谨致以诚挚的感谢。

特别感谢数字产业学院姜放放老师对本项目的鼎力支持。姜老师提供了开发板调试环境、SSD 运行虚拟机，以及博远 701 与 704 场地，助力我专注深耕鸿蒙全栈开发与学习；

感谢中软国际教育王岚老师进行为期一学期的三段式授课，并提供了丰富的课程资源与项目内容。王老师在项目制综合实践中的指导，为本项目奠定了重要基础；

感谢所有参与答辩的老师和同学。特别感谢数字产业学院姜放放老师、中软国际教育王岚老师、计算机科学与技术李源老师，以及计算机科学与技术 24 级李焱龙同学、北京信息科技大学软件工程 25 级黄继洲同学特别到场聆听答辩。感谢在场所有《鸿蒙项目制综合实践》课程同学的宝贵意见与支持；

感谢我的父母，以及 Tracy、Joe、Youca 在我学习与项目开发过程中给予的理解与支持；

感谢所有正常工作运行的生产力设备。感谢 OpenAI、Google、深度求索、智谱 AI 等公司提供的大语言模型技术，在代码调试、文档撰写与网页设计过程中提供了有力辅助；

再次感谢所有直接或间接帮助过本项目的老师与同学。

附　录

页面内容	URL
鸿蒙应用端	https://github.com/Rayawa/SmartShed-HarmonyOS
Hi3861 嵌入式端	https://github.com/Rayawa/SmartShed-Hi3861
项目主页	https://rayawa.top/projects/SmartShed.html