近代无线电实验报告

实验选题：　基于 Android 平台的无线 WiFi 控制

姓名：　江一正

学号：　20300726001

专业：　电子信息科学与技术

指导老师：　陆起涌

一、实验目的

1. 了解 Android 开发相关基础问题，掌握 Android 开发流程。
2. 了解 Android 系统与无线 WIFI 模块的基本通信问题。
3. 使用 Android 开发组件进行基础的 Android APP 开发。
4. 设计并实现 Android APP 与无线 WIFI 模块的通信。

二、问题提出与解决意义

1. 背景

近年来物联网技术不断提升，智能家居领域正在迅速发展，促使人们追求更舒适的生活。将智能手机、智能控制与家居控制系统相结合，为新一代智能家居系统提供一个方向。

中国智能家居的发展，深受外国智能家居的影响，其为国内的智能家居开拓了道路。二十年来，中国智能家居行业，也是从基础开始逐渐发展起来的。最开始是从引入智能家居的概念起步的，紧接着通过各大厂家的推广和宣传，缓慢发展，逐渐在智能家居市场中形成产业链。

后来，随着 5G、人工智能和物联网（artificial intelligence & internet of things，AIoT）技术的引入，为智能家居产业的发展注入了新的力量。得益于 5G、大数据、人工智能、物联网等新兴技术，智能家居产品迎来了新的发展趋势，智能产品和系统不断升级改造，更大程度的满足用户多样化需求，为用户提供真正的万物互联的居家生活体验。

2. 问题陈述

随着人们需求提高，市场上智能家居产品趋向多元化。然而，目前市面上产品依然存在问题，一是各厂商间兼容性差，协议、平台均无统一行业标准。二是成本高，一套常规家庭使用成套产品购买安装上万元，这还不含后期维护成本。三是对网络要求高，一般都需连网，比如智能门锁因断电而无法进屋。

3. 解决意义

解决上述问题对于推动智能系统的进步具有重要的实际应用价值。一个功能完善、性能卓越的系统将提高用户满意度，拓展系统的应用领域，为相关行业带来更多创新。通过本研究的成果，可以推动智能系统在物联网、人工智能等领域的更广泛应用。

三、系统功能和性能要求

1. 实验要求

（1）设计一个基于 Android 与无线 WIFI 开关的控制系统，通过 Android 手机界面的操作控制无线 WIFI 开关的开启与闭合。

（2）实现 Android APP 的设计与实现，要求有简单的界面进行操作。

（3）探讨服务器端通信的方法与实现。

2. 系统设计思路

系统基于 ESP8266 单片机为核心控制各模块，并与手机端通信。手机作为控制设备的管理器，提供交互界面。将上述模块结合，可将家电智能控制、无线遥控控制、空气质量检测等功能一体化，实现居家应用场景下的智能系统。

以下为系统创新点的设计思路：

（1）交互性强：系统集成多媒体技术，通过触屏、图片、动画、图表等形式，提升用户对家居状态的感知和控制。

（2）万用空调控制：基于空调红外遥控编码，根据厂商型号，自行匹配编码模式实现红外控制。

（3）舒适度评估：引入舒适度模型，通过温/湿度传感器数据，实现空调的自动调整。

四、 系统选型和技术

1. 单片机

单片机作为系统控制中心，在选型对系统开发至关重要，对软硬件架构的底层开发起中，起到直接的影响。市场上单片机开发板种类众多，常见的有 51 单片机、STM32、ESP8266、树莓派等。将从计算与存储资源、成本、网络能力等方面进行分析对比（见下表），选取合适单片机作为系统的主控板。

	51 单片机	**STM32**	**ESP8266**	**树莓派**
计算能力	弱	强	中等	强
成本	低	中等	低	高
功耗	低	中等	低	高
网络应用	需外接 WiFi 模块	需外接 WiFi 模块	内置 WiFi 模块，开发资料健全	内置 Wi-Fi 模块
编程语言	汇编、C	C/C++	C/C++	Python
应用领域	简单 GPIO 控制	各种复杂应用	网络应用场景	各种复杂应用
优势	成本低、体积小	GPIO 引脚数多，支持多种通信协议	成本低、网络能力强、资料丰富	支持完整操作系统，计算能力强
劣势	计算能力弱，不适合网络通信应用	成本高，并行处理能力弱	GPIO 引脚数有限，计算能力有限	成本高，功耗高

经对比上表得出，可看出 ESP8266 单片机的优点突出，提供了低成本、低功耗、适中的计算性能，其内置 Wi-Fi 模块提供了网络连接能力，且资料丰富更易于系统开发，因此选取 ESP8266 单片机作为系统主控板。

ESP8266 单片机，具有低功耗 32 位双核处理器，内置蓝牙、WiFi、天线等模块。同时单片机支持 802.11 WiFi 标准、UART/GPIO 接口，且可用 Arduino C/C++编程平台进行开发，适用于物联网项目，具体技术应用如下：

（1）物联网通信：作为系统通信模块，可通过 Wi-Fi 连接上网，对 Web 服务器进行访问和更新数据

（2）输入/输出控制：可同时控制多个输入/输出引脚，如传感器、继电器、显示屏、红外发射器等。

（3）状态机控制：通过 Arduino 平台上 C/C++编程，实现状态机控制，如定时器、计数器，进而推进显示屏更新、传感器数据上传、输出状态更新等功能。

2. Android 系统开发

Android 平台作为当今最常见的智能手机软件平台之一，自其 2007 年发展以来，移动应用开发工具不断发展，已面世多个工具，常见的有 MIT App Inventor（麻省理工 2010 年开发）, Android Studio （官方 2013 年开发）, Flutter（谷歌 2018 年开发）等。

开发工具发展脉络展示了移动应用开发领域的不断创新和演进，从简化开发过程的 MIT App Inventor，到 Android Studio 的专业化支持，再到 Flutter 的跨平台革新，各个阶段都为开发者提供了更多的选择和灵活性。下表将从各工具最新版（截至 2023-11-30）进行对比，分析合适的 Android 系统开发工具。

	MIT App Inventor	**Android Studio**	**Flutter**
编程语言	图形化编程	Java/Kotlin	Dart
功能性、灵活性	有限	高	高
学习门槛	低	高	中等
开发周期	短	较长，需一定的编程基础才可上手	中等，有丰富 UI 组件和 HotReload 功能
性能	一般，受编译方式局限	高	较高，解决原生应用
平台	Android、iOS	Android	Android、iOS、Web 等
优势	无需编程，可快速开发	功能丰富，调试工具强大，灵活性高	跨平台，丰富 UI 组件，HotReload 快速开发
劣势	高级功能有限，定制性较差	学习门槛高，开发周期较长	开发模式特殊，需时间上手

经对比上表得出，可看出 Flutter 的优点突出，提供了更灵活、更丰富且高度可定制 UI 组件，且热重载功能允许实时查看修改效果进而提示效率，其跨平台开发和性能优越也是优势之一。因此，最终选择 Flutter 作为 Android 系统开发平台。

Flutter 通过 Dart 语言在 Visual Studio 上进行开发。前端涵盖用户界面、交互、控制和监控。后端涵盖手机端与 Web 服务器的数据联通，是支撑系统运行数据流的底层框架，具体技术应用如下：

（1）交互性界面：可用直观图像列表展示家居开关、遥控器界面，可创建按钮点击即可切换状态。

（2）设备控制：在按钮点击后，可实时访问 Web 服务器更新状态。

（3）数据可视化：将传感器以图表形式展示，让用户更直观了解环境变化趋势，如温度、湿度、空气质量。

3. Web 服务器

Web 服务器作为系统重要的一环，是连接单片机、的桥梁，选择市场上较普及的国内外服务器供应商，包括阿里云（阿里巴巴推出）、华为云（华为推出）、Firebase（谷歌推出）、Supabase（Github 开源）供选择。

	阿里云	**华为云**	**Firebase**	**Supabase**
开发语言	Java, Python, NodeJS 等	C, C#, Java, NodeJS 等	JavaScript 为主	JavaScript, NodeJS 等
文档资料	丰富	一般	丰富	一般
学习门槛	中等	高	低	低
优势	服务器稳定，文档资料丰富	服务器稳定，安全性高	实时数据库，身份认证功能，文档资料丰富	开源可本地运行，自定义安全密钥，实时数据库，免费额度高
劣势	多数功能需付费	学习成本高，配置方式复杂，服务器较少	需科学上网，国内不适用	现有文档资料较少

经对比上表得出，可看出 Supabase 平台的优点提出，其提供了完整、安全性高（自定义安全密钥）的实时数据库，且作为支持多种开发语言（JavaScript, NodeJS 为主）的开源平台提供了可定制性和可控性，降低了开发成本。因此，最终选用 Supabase 平台作为 Web 服务器。

手机端和 ESP8266 单片机通过 Wi-Fi 上网与 Web 服务器连接，实现远程通信和数据传输。选用 Supabase 开源后端平台，其使用 PostgreSQL 作为数据库引擎，并提供 RESTful API 接口，能便利地访问和操作数据库资料，具体技术应用如下：

（1）数据交互：充当设备之间数据交互桥梁，如接受单片机传感器数据，并传输数据到手机端。

（2）安全机制：只有用户身份认证后，授权用户才可访问或更改数据，且透过 RLS 安全机制，可保证通信是加密的，防止资料被窃取。

（3）实时更新：支持低延时的实时数据同步，只要简短代码可获取数据更新信息。

4. 印刷电路板

系统硬件靠印刷电路板（PCB）实现，系统电路连接单片机与各个电子元件，包括传感器、输出设备等。PCB 设计由嘉立创（JLCPCB）的 EasyEDA 平台上实现，通过合理的电路图连线，并配置好原件封装、设定布线规则、组件布局布线、设计规则验证等步骤，可将生成一个 PCB 图。

经成本、应用选取、性能考量，PCB 工艺信息如下：

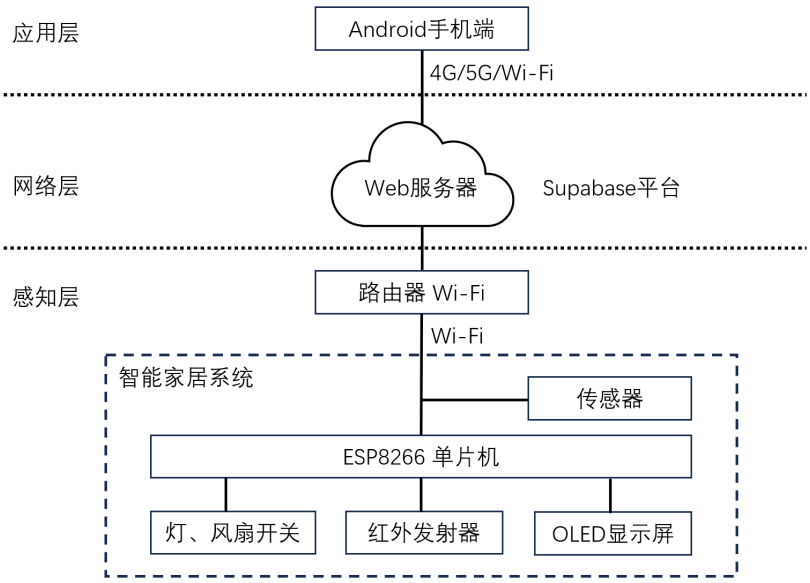
板子层数：	2	板子类型：	FR4
长度 x 宽度：	10cm x 10cm	板子厚度：	1.6mm
成本：	20 元（5 片）	焊盘喷镀：	有铅喷锡

确认设计无误后，即可进行 PCB 下单和生产。收到 PCB 后，需通过万用表测试确认连线正确，再焊接已备好的电子元件上板，最后将单片机载入代码，确认系统正常运行。

五、系统设计与实现

1. 系统框架

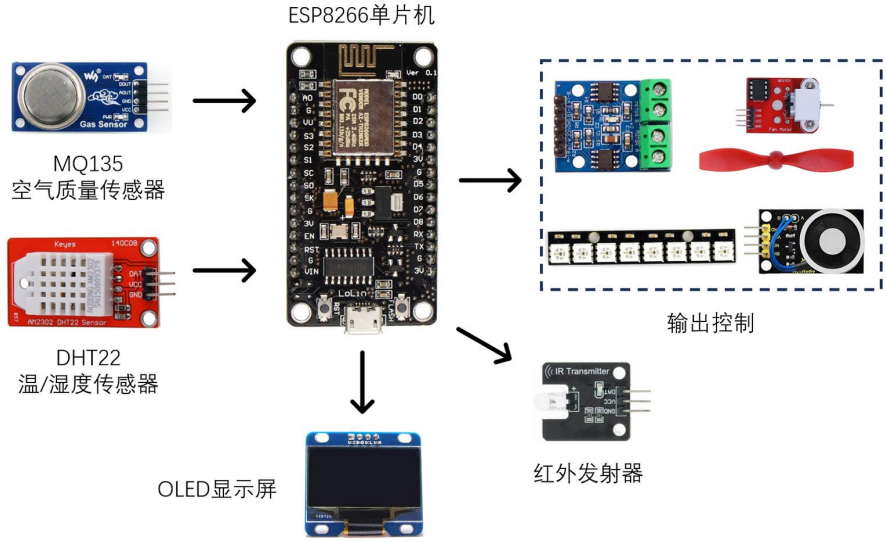
下图展示智能家居系统、Web 服务器和 Android 手机端之间的连接。在应用层，Android 手机端可通过 Supabase 服务器实现控制和数据获取。在网络层，通过 Supabase 平台的服务器，实现了手机端与智能家居系统之间的连接。在感知层，采用 ESP8266 单片机对各个传感器进行数据采集和处理，与此同时实时控制各个输出端的状态。



2. 硬件组成

（1）智能家居系统：系统框架图

智能家居系统（见下图）采用温/湿度和空气质量传感器，数据将传输到 ESP8266 单片机的输入引脚中。输出引脚方面，ESP8266 按手机端指示操控输出设备（灯、风扇、门等）和红外发射器。此外，OLED 显示屏可显示传感器和输出设备状态，作为用户提示用途。

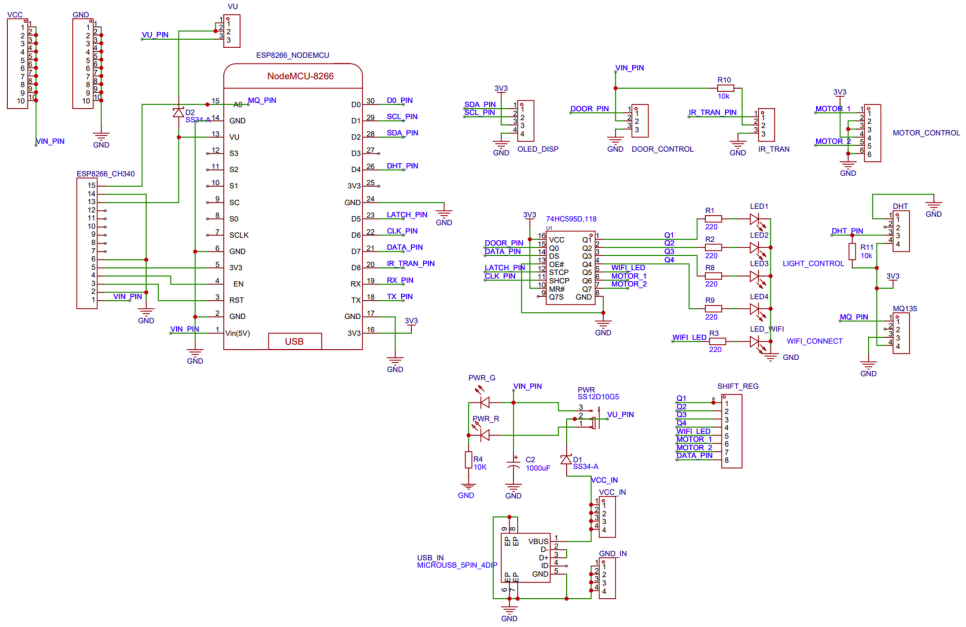


（2）智能家居系统：电路图

通过嘉立创 EDA 平台，按上述系统框架图的思路拓展，绘制系统整体电路图（见下图） 。其中电

源控制采用 MicroUSB 为 ESP8266 单片机 5V 供电，并加入二极管作为保护作用。另外，通过 74HC595 移位寄存器芯片，在单片机输出引脚有限情况下，实现输出引脚扩充的作用。

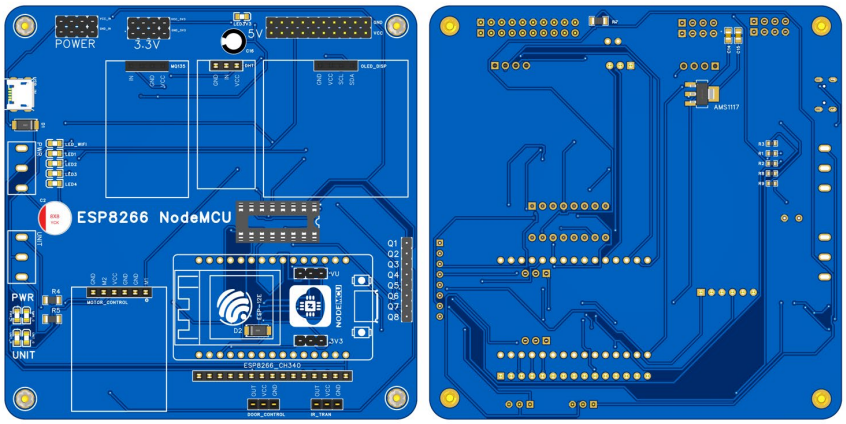
进入 PCB（印刷电路板）设计之前，将各元件配置合适 PCB 封装（即元器件大小、长宽、间距等的表达形式） 即可进入 PCB 部分。



（3）智能家居系统：PCB（印刷电路板）

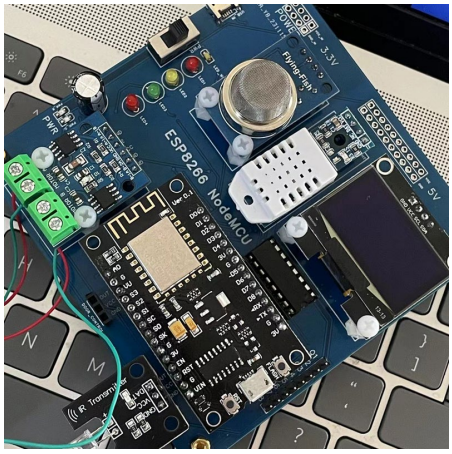
将上述电路图转为 PCB，并设定 PCB 布线规则，供安全性、检查与纠错用途。之后，再将元件进行布局，并放置于顶层、底层的合适位置。使用软件自动布线功能，可将大部分线路自动连接起来，其余线路手动完成即可。

完成连线后，启用 DRC（设计规则检查）进行检查，最后在顶层、底层进行覆铜，有降低地线阻抗作用、提高抗干扰能力、提高电源效率等功能。最终 PCB 设计完成后，PCB 展示图如下。



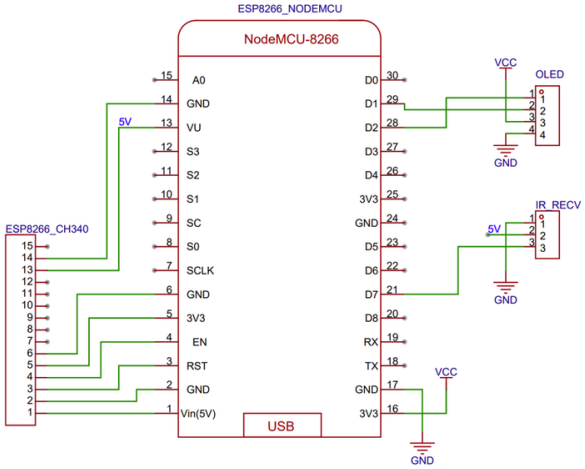
（4）智能家居系统：焊接与组装

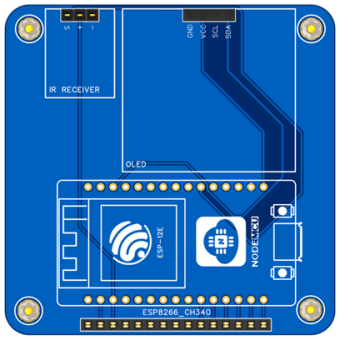
将提前预备好的模块和元器件准备好，按元器件高度决定焊接顺序，依次为 0805 贴片电阻和 LED（发光二极管） 、USB 接口、各类模块等。完成焊接后，用万用表测试各引脚连线状态，才上电导入测试脚本对各模块功能进行分开测试，确保所有传感器、输出模块、显示屏正常操作。完成上述测试后，进一步导入完整程序，并确保手机端各个功能可正常使用。

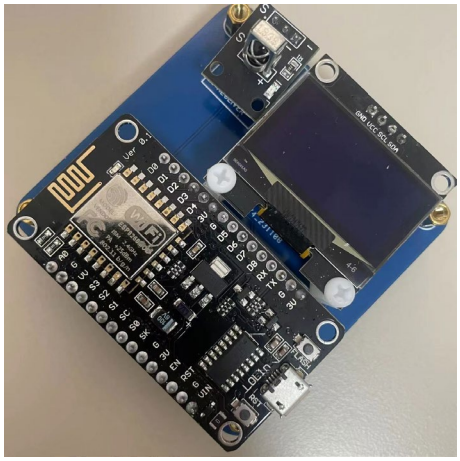


（5）测试模块：空调模拟器

为了整体测试红外遥控器功能，额外设计一个空调模拟器，内含 ESP8266 单片机红外接收器、显示屏。红外接收器作为主要信号输入，显示屏用于空调状态显示，如温度、风速、模式等设定。利用嘉立创 EDA 平台，将电路图和 PCB 图绘制。将各元件焊接上 PCB，通过连线短路测试后，即完成硬件部分，展示图如下。





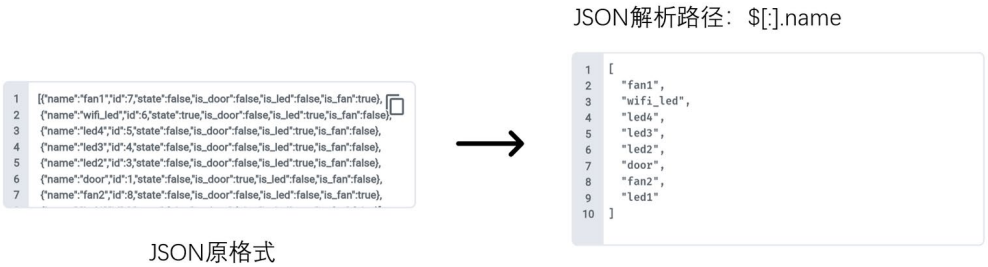


3. 软件组成

（1）Supabase 配置

Android 手机端、ESP8266 单片机通过 RESTful API 连接到 Supabase，以实现双向通信。设备身份通过三个组件：TableName、ObjectID、AnonKey 进行确认，其中 TableName 和 ObjectID 为设备组别和编号，AnonKey 则充当密钥的角色。在三个确认正确无误后，Supabase 会颁发 Token（许可证），数据才可进行传输或更新。

在数据传输过程中，上传（更新）和下载（获取）数据均为 JSON 格式，因此需解析和生产 JSON（JavaScript 对象表示法）数据。对于数据获取，需使用 JSON 解析格式（见下图），即可从中得出关键信息。



JSON原格式　　JSON解析结果

对于数据更新，则根据需更新 ObjectID，生成对应部分 JSON 原格式数据（见下图），对 Supabase 申请数据更新请求，方可更新对应 TableName 和 ObjectID（数据组别和编号）的数据。

C++代码　　JSON原格式

备注：以上 JSON 转换在手机端、ESP8266 单片机上代码实现

（2）ESP8266 单片机

利用基于 C/C++的 Arduino 编程（见下图），定义了智能家居系统运行的各个，层面。Arduino 程序由以下部分组成：

i）功能支持库：包括连线上网（“ESP8266WiFi.h”） 、Supabase 支持（“ESP32_Supabase.h”） 、传感器驱动（“DHT.h”） 、JSON 支持（“ArduinoJSON.h”）等。

ii）引脚映射：定义全部输入/输出的引脚在 ESP8266 单片机的连接位置。

iii）初始化：一次性运行项，包括使用 ssid 和密码连上 WiFi、Supabase 连接密钥配置、传感器初始化、引脚类型定义等。

iv）循环：按功能设置三个定时器，对应 Supabase 输出控制（灯、风扇等家电的开关控制） 、Supabase 传感器数据上传（温/湿度、空气质量传感器） 、OLED 显示屏，这三项分别设定为三 种状态机，设定为在不同更新速度下同时跑动。

oledDisplaySupabase_5.ino | PinDefinitionsAndMore.h

```
#include <Arduino.h>
#include <ESP32_Supabase.h>
#include <ArduinoJson.h>
#include <U8g2lib.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include "PinDefinitionsAndMore.h" // Define IR pinouts,macro,etc
#include <IRremote.hpp>
#include <MQ135.h>
#include "DHT.h"
//#include <Adafruit_Sensor.h>

// ESP8266WiFi Init()
#if defined(ESP8266)
#include <ESP8266WiFi.h>
#else
#include <WiFi.h>
#endif

// U8g2 Init()
#ifdef U8X8_HAVE_HW_SPI
#include <SPI.h>
#endif
#ifdef U8X8_HAVE_HW_I2C
#include <Wire.h>
#endif

// select only NEC and the universal decoder for pulse distance protocols
#define DECODE_NEC          // Includes Apple and Onkyo
#define DECODE_DISTANCE_WIDTH // In case NEC is not received correctly. Universal decoder for
```

备注：详细代码解析见 Arduino 源代码

（3）Android 系统

利用基于 Dart 语言的 Flutter 编程（见下图），项目结构和文件布局项目简介如下：

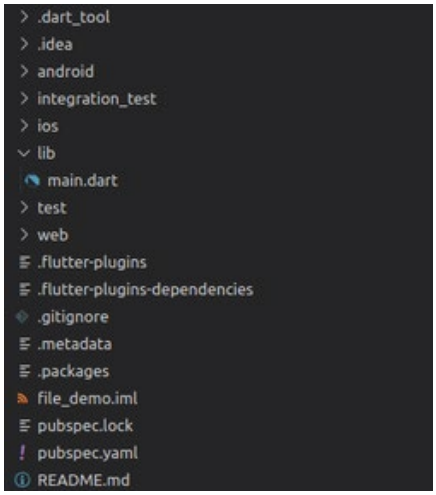
（i）pubspec.yaml： Flutter 项目的清单文件，类似于 AndroidManifest.xml。在这个文件中，你可以 定义项目的名称、描述、依赖关系（Flutter 包、第三方库等）等信息。

（ii）lib 文件夹： 包含 Dart 代码文件，通常包括 main.dart，即应用程序主要入口点，类似于 Android 中的 MainActivity，包含应用程序的主要逻辑，如生命周期、用户界面和用户交互等。

（iii）assets 文件夹： 包含应用程序使用的静态资源文件，如图像、字体等。这可以用于存储和访问与用户界面相关的资源。

（iv）pubspec.lock 和 pubspec.yaml： 定义了 Flutter 项目的依赖项，类似于 Android 中的 build.gradle 文件。pubspec.lock 记录了实际使用的依赖项的版本信息。

（v）android 和 ios 文件夹： 分别包含 Android 和 iOS 平台的原生代码。通常情况下，你在 Flutter 中主要编写 Dart 代码，而这些文件夹中的内容由 Flutter 框架自动生成和管理。



前端开发主要为用户界面和交互效果，Android 应用程序由三个界面组成（见下图），分别为远程开关控制、传感器数据图表、空调遥控器。其中，用户界面利用 XML 语言进行定义，利用图层架构的概念实现：

（i）图层：界面可含层级结构，每个图层可以包含不同的元素。这些元素可以叠加在一起，形成用户最终看到的界面。图层的概念常用于图形设计和动画制作中，用于管理不同元素的显示顺序。

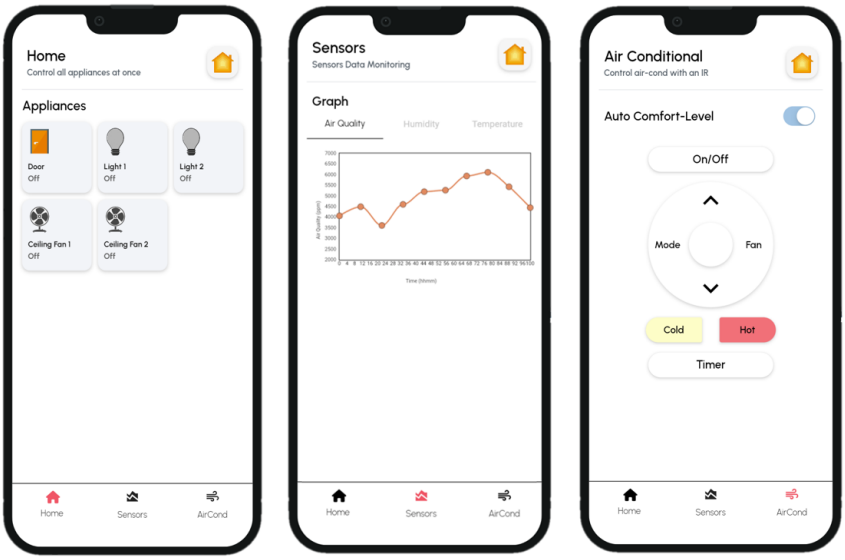
（ii）堆叠（stack）：元素按照堆叠的顺序依次显示，后面的元素可能会覆盖前面的元素。在移动应 用开发中，常见于堆叠式导航或层叠式卡片布局。

（iii）行（row）：行是一种布局方式，元素水平排列，从左到右依次显示。行常用于横向展示一系 列元素，如导航栏中的链接或按钮。

（iv）列（column）：列是一种布局方式，元素垂直排列，从上到下依次显示。列常用于纵向展示 一系列元素，如侧边栏中的菜单选项。

（v）网格视图（GridView）：网格视图 是一种用于显示数据的网格布局，通常用于展示二维数据集。它将数据按行和列的形式排列，每个网格单元格可以容纳一个视图元素。

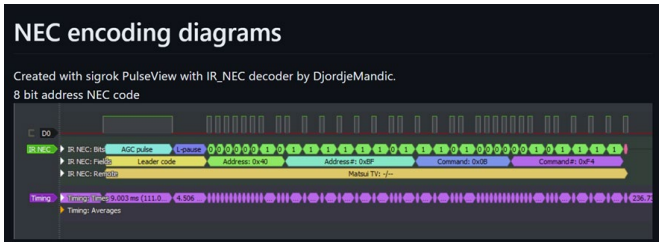
备注：以上只含基本模块，具体实现过程更复杂



界面设计含多种类型元素的结合，结合以上基本框架，可建立一个可用的用户界面。Kotlin 文件用于定义各模块逻辑和后端交互，涵盖按钮功能、后端 Supabase API 调用、界面更新等。后端开发主要透过 RESTful API 与 Supabase 进行访问和更新数据，均以 JSON 数据表示（JSON 生成和解析） 。

（4）测试模块：空调模拟器

红外遥控器按 NEC 编码形式发射信息，通过在 Arduino 编程实现 NEC 解码器，再根据对应功能设定空调模拟器状态，如开关、温度上升/下降、暖气/冷气模式、风速上升/下降等。ESP8266 内设定状态机，对红外接收信息进行实时状态更新，并及时反映到 OLED 显示屏上。



五、系统测试与分析

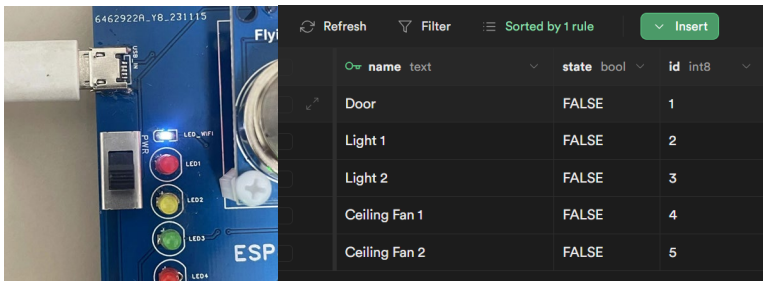
1. 测试准备工作

测试前确保所有系统硬件组成已完成安装，主要包括 PCB 板、温/湿度传感器、空气质量传感器、显示屏、IR 发射器、其余输出模块等。按电路图将所有相关模块按对应引脚连上，确保每个连线正确后，方可上电测试。

将 PCB 板连上电脑，在 Arduino 上确保 ESP8266 单片机对应的 COM 编号有显示出来，此时导入程序上传，若 Arduino 界面输出框中显示上传完成代表程序成功传入设备中。

2. Supabase 设备接入测试

将 PCB 板上电后，通过 Wi-Fi 和 Supabase 平台连接，当连接成功后，设备状态 LED 会亮起显示如下图一。另外，也可通过 Supabase 平台上确认连接，只要修改其一状态为 TRUE，若对应输出点亮，则连接成功。



3. 模块功能测试

（1）远程开关控制

针对远程开关控制进行测试，用手机界面上对灯、风扇分别进行 100 次开和关的操作，其中每次操作之间间隔 5 秒。在完成各项测试后，实验数据如下:

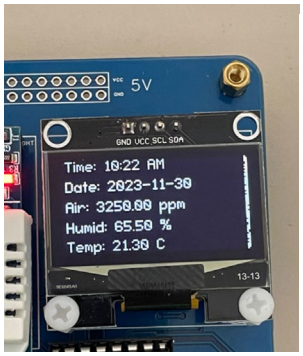
测试模块	状态	次数	成功率（%）	平均延时（s）
灯	开	100	100%	2.08
	关	100	100%	2.13
风扇	开	100	100%	1.90
	关	100	100%	1.84

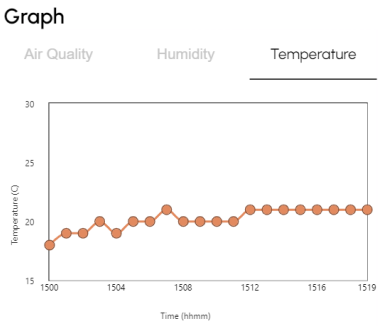
经总数 400 次的测试，灯、风扇的开关控制功能 100%正常，无失效情况。在平均延时方面，灯平均延时在 2.105s，风扇平均延时在 1.87s。

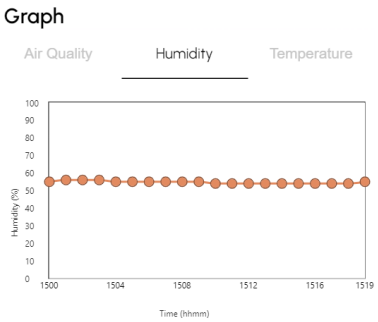
（2）传感器数据采集

系统运用温/湿度传感器，ESP8266 单片机将采集空气中的温、湿度进行处理，结合当前时间信息，将数据连网上传到 Supabase 平台，在手机端上实现数据可视化。

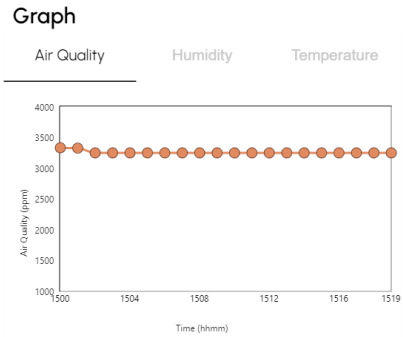
如下图所示，当前系统设定为每分钟采集一次数据（应用程序上可更改），因此图表上显示近 20 分钟的温、湿度趋势变化图。显示屏上显示当前温、湿度信息，而手机端上选用折线图表示采集到的温、湿度情况，纵坐标代表温/湿度，横坐标表示当前时间。





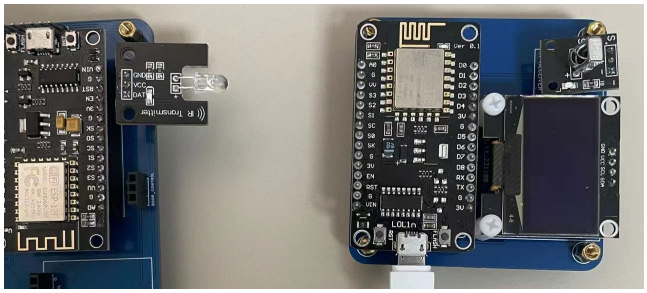


空气质量传感器沿用相同模式，在手机端上实现可视化。显示屏上显示当前的空气质量指数，手机端上则呈现 20 分钟的折线图（见下图） 。



（3）空调智能控制

针对空调智能控制的测试，用户在手机端上操作后，观察主控板的红外发射器和空调模拟器的红外接收器之间的通信。在测试前，确保两端均上电，并将发射器、接收器两端对准在一直线上如下图。



在测试中，关注不同距离下的各 100 次接收准确率，5 组距离分别为 0.25、0.75、1.5、3、6 米，其中每次操作之间间隔 5 秒。完成各项测试后，实验数据如下：

距离（m）	次数	成功率（%）	平均延时（s）
0.25	100	100%	1.85
0.75	100	100%	1.94
1.5	100	99%	2.00
3.0	100	97%	2.12
6.0	100	95%	2.20

经总数 500 次的测试，0.75m 以内控制功能 100%正常，无失效情况；在 1.5m 至 6m 之间则成功率

为 95%-99%。在平均延时方面，不同距离下的整理平均在 2.022s。

六、总结与展望

1. 实验总结

本文研究基于 Android 平台的无线 WiFi 控制的方法。主要研究工作如下:

（1）背景研究，智能家居技术在物联网、人工智能和 5G 推动下迅速发展。中国智能家居市场兴起并逐渐形成产业链。新技术引入为产品提供升级机会，满足用户多样需求。

（2）系统需求分析总结，系统旨在解决智能家居产品存在的兼容性、高成本等问题。选用 ESP8266、Flutter 和 Supabase 等技术，设计灵活、成本低、用户体验良好的智能控制系统。

（3）硬件设计与实现，采用温/湿度、空气质量传感器，ESP8266 单片机，红外发射器等多种硬件组件。通过嘉立创 EDA 平台成功完成 PCB 设计，包括电路图、PCB 图以及焊接与组装步骤。空调模拟器的设计增加了对红外遥控器功能的全面测试。

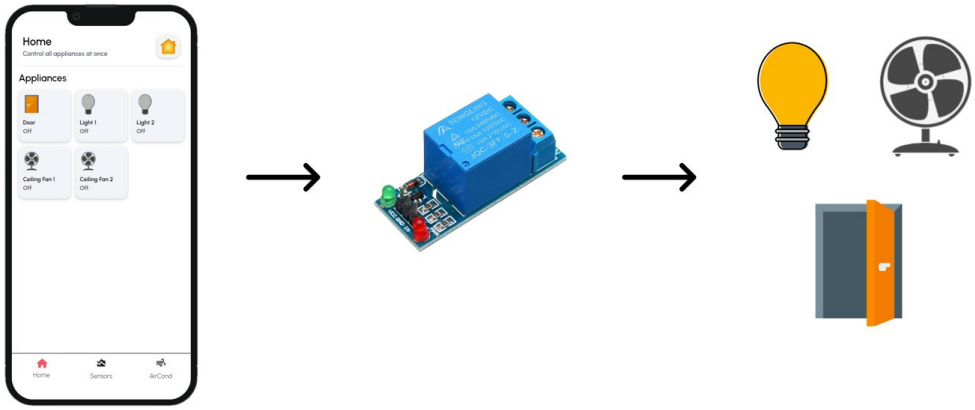
（4）软件设计与实现，Supabase 平台成功配置，实现了 Android 手机端和 ESP8266 单片机通过 RESTful API 连接到 Supabase 的双向通信。ESP8266 单片机采用 Arduino 编程，包括初始化、循环等功能，实现了对传感器数据采集和输出控制的综合管理。Android 系统利用 Flutter 框架进行前端开发，实现了用户界面、交互效果和三个主要功能模块。

（5）系统测试与分析，通过 Supabase 设备接入测试，确保 Wi-Fi 和 Supabase 平台的连接正常。在模块功能测试中，对远程开关控制、传感器数据采集和空调智能控制进行了详细测试，结果表明系统在各方面表现良好。

2. 应用前景

（1）远程开关控制

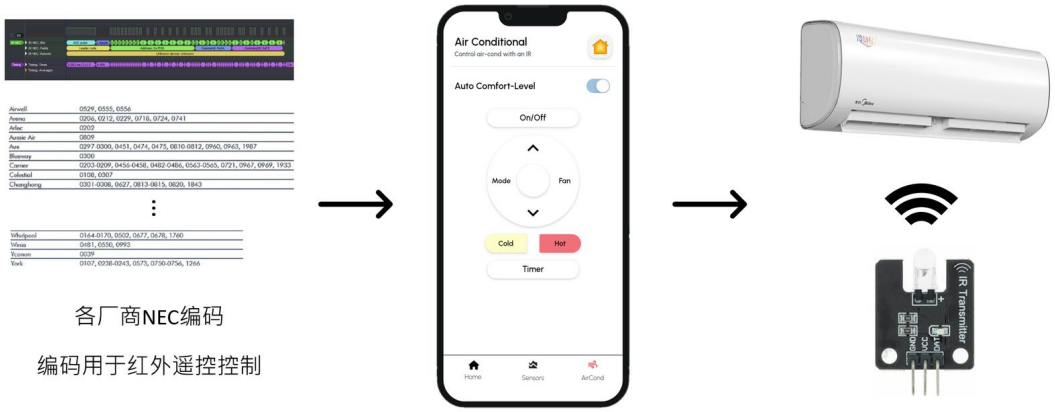
实际家居开关控制中，为了实现直流弱电控制交流强电，则采用继电器作为隔绝和控制的效果。为了防止断网而使系统无法运作，可采用二路开关装置，作为断网的后备方案。具体使用过程如下图，点击图像控制家居开关，一次打开，一次关闭。



（2）空调智能控制

为了匹配各厂商型号的空调，通过拆解万用遥控器，可得到各厂商的红外遥控 NEC 格式编码，再根据对应型号得确切的完整控制编码。

具体使用过程如下图，位于空调控制界面，选取空调厂商、型号，即可打开对应界面，点击相关控制按钮，红外发射器将实时控制空调。

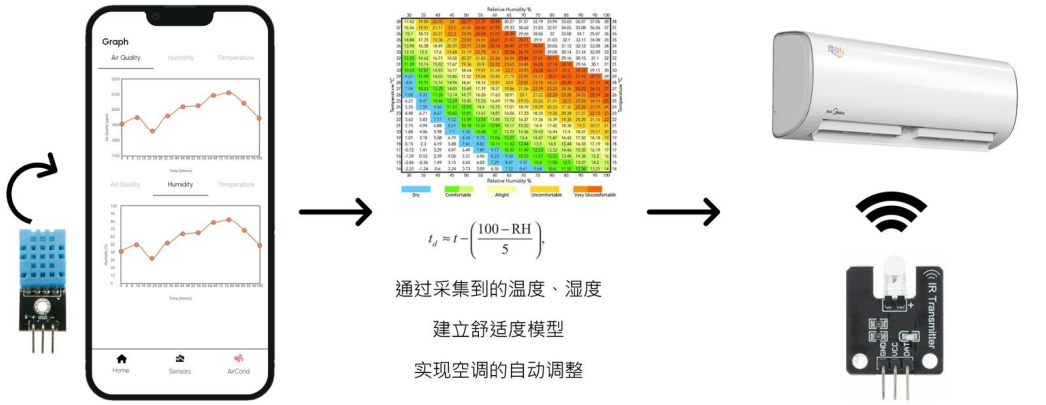


各厂商NEC编码

编码用于红外遥控控制

（3）舒适度模型

温/湿度、空气质量传感器采集数据后，以图表形式显示在传感器界面，并附上时间数据。截取温度、湿度信息，可用于建立舒适度模型，通过计算露点温度得出当前舒适度状态。在空调控制界面，加入空调自动控制开关，在保持舒适状态的前提下，系统可实现自动调控制，在必要时用红外发射器增减空调温度、模式等设置。



通过采集到的温度、湿度

建立舒适度模型

实现空调的自动调整

3. 未来展望

（1）ESP32 单片机

对于系统单片机选型，计划将 ESP8266 升级到 ESP32，考量点如下：

（i）性能提升： ESP32 相对于 ESP8266 来说在性能上有显著提升。它具有更快的处理速度、更多的内存和更丰富的外设。这种性能提升可以带来更好的响应速度和更复杂的应用能力。

（ii）更多的通信接口： ESP32 集成了更多的通信接口，包括蓝牙、Wi-Fi、低功耗蓝牙（BLE）等，这为智能家居系统提供了更多的连接选项。这使得 ESP32 可以更灵活地与其他设备和传感器 进行通信。

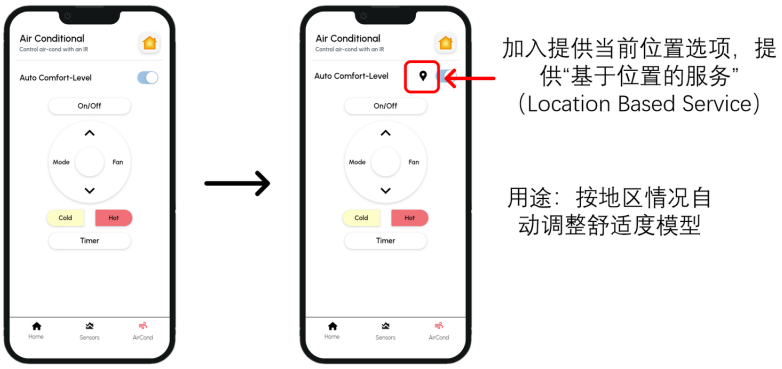
（iii）多核处理： ESP32 具有双核处理器，这可以允许系统同时执行多个任务。在智能家居系统中，可能需要同时处理多个传感器数据、用户输入等，多核处理器可以提高系统的并发性能。

（iv）低功耗选项： ESP32 支持不同功耗模式，包括深度睡眠模式，这对于需要长时间运行、但又需要保持低功耗的智能家居设备是一个重要的考虑因素。

（v）更丰富的硬件接口： ESP32 提供了更多的通用输入输出引脚（GPIO），这使得连接各种传感器、执行器和其他硬件组件更加容易。

（2）舒适度模型优化

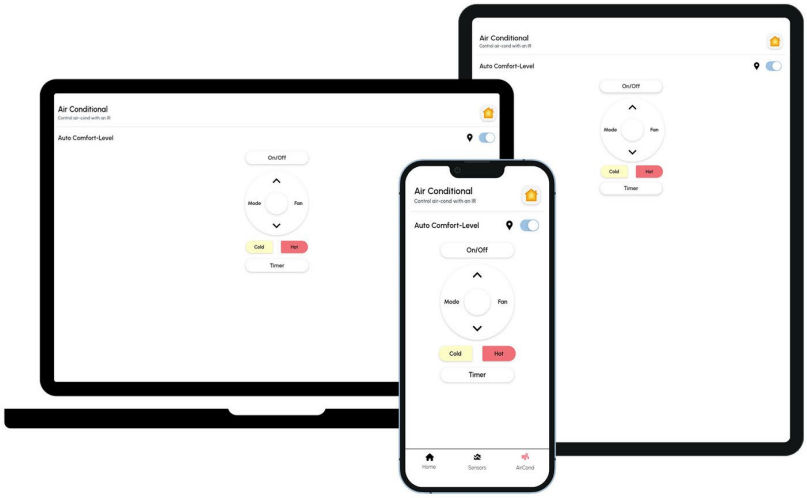
对舒适度模型进行深度优化，引入更多传感器数据和智能算法。通过更精准的温度、湿度、空气质量等数据采集，结合先进的机器学习技术，实现对家居环境的更智能、个性化的控制。优化后的舒适度模型将更加贴近用户的需求，提供更智能、人性化的居家体验。



原版　　修改后

（3）多平台兼容

未来展望着眼于实现系统的多平台兼容性，使其不仅仅局限于 Android，还能支持 iOS（苹果手机系统）、平板、Web 平台以及微信小程序等多种终端，以提供更广泛的用户覆盖和更便捷的使用体验。



七、个人收获

透过智能家居系统项目，让我对嵌入式系统、Web 服务器、移动端开发等领域有更深的认识，更是拓展了我的专业知识与应用，特别是 ESP8266 单片机在物联网的应用，需对数据传输协定有足够的了解，如 I2C、串行、JSON、API 等。

在系统设计中，培养了我对解决实际问题的能力，特别是考虑家居场景中人们的需求与痛点，在考虑功能性、性价比、平台兼容性等多种考量后，应该怎么选取合适的技术，以提出一个有价值的智能家居系统解决方案。

此外，在手机应用开发过程中，增强交互性是最具挑战性的，并非增加一个按钮或图片就办得到，而是需在每个细节下功夫，比如开关控制界面设计中，风扇从关闭到开启过程中，要呈现一个风扇转动的动画，就需用 Photoshop 设计 gif 风扇转动的动画，再导入应用程序中。

经历 3 个月的开发周期，最终完成了系统的构思、设计、实现全过程，更让我深刻了解从“梦想家”

到“实践者”的转变，从一张图纸到最终成品展现在眼前，是多个夜晚努力拼搏的成果。

通过这个项目，不仅在技术上有所提升，还培养了解决问题的能力，为我的未来学习和职业发展奠定了良好的基础。此次经历将成为我往后实习就业宝贵的一段经历，让我更加自信地迎接未来的挑战。

参考文献

[1] 金纯用. 基于 iOS 的智能家居系统设计与实现[D]. 桂林: 广西师范大学, 2022.

[2] Li S, Li D, Zhao S. The internet of things: a survey[J]. Information Systems Frontiers, 2015, 17(2).

[3] AI-Mutawa R, Albouraey F. A smart home system based on internet of things[J]. International Journal of Advanced Computer Science and Applications, 2020, 11(2).

[4] 陆起涌. 近代无线电实验电子资料 [EB/OL]. 复旦大学 (2023-05-26) [2023-11-30] https://elearning.fudan.edu.cn/courses/67423/files

[5] Marco, S. Internet of Things with ESP8266 [M]. Birmingham, Packt Publishing, 2016: 151-173.

[6] Benoit, B. Mastering ArduinoJson 6 [M]. France, Benoît Blanchon, 2020: 75-144.

[7] 欧阳燊. Android Studio 开发实战：从零基础到 App 上线 [M]. 北京，清华大学出版社, 2022.

[8] SNx4HC595 8-Bit Shift Registers With 3-State Output Registers [EB/OL]. Texas Instruments. (2021-10-01) [2023-11-30]. https://www.ti.com/lit/ds/symlink/sn74hc595.pdf

[9] 立创 EDA 使用手册 [EB/OL]. 嘉立创. (2019-07-30) [2023-11-30] https://club.szlcsc.com/article/details_40947_1.html

[10] Mark, G. The Relationship between Relative Humidity and the Dewpoint Temperature in Moist Air: A Simple Conversion and Applications [J]. Bulletin of the American Meteorological Society, 2005, 86(2): 225-234.

[11] 王爱利. 基于 ESP32 的智能家居系统的设计与实现 [D]. 沈阳: 沈阳师范大学, 2023.

[12] 基于 STM32 的物联网智能家居系统设计 [J]. 工业仪表与自动化装置, 2022 (2): 27-31.

[13] 趙英傑. 超图解 ESP32 深度实作 [M]. 台湾，旗标科技股份有限公司, 2021.

[14] ESP8266 Technical Reference [EB/OL]. Expressif Systems. (2020-10-01) [2023-11-30] https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf

[15] 陈敏，关欣，马宝罗等. 物联网白皮书 [R] 北京. 中国信息通信研究院, 2020.

[16] 2022 年中国物联网市场研究报告：以宏观视角探析产业生态 [R] 南京. 头豹研究院, 2022.