毕 业 设 计 （ 论 文 ）

无线火灾报警软件系统设计

专　　业	物联网应用技术
班　　级	2016 级物联网应用技术高职（1）班
学生姓名	赵晨瑄
指导老师	袁铭

无线火灾报警软件系统设计

目 录

无线火灾报警软件系统设计

摘要：本设计为一个无线火灾报警软件系统设计，其控制核心为 STM32 单片机。当烟雾传感器检测到一定浓度的烟雾值的时候，将采集的信号通过电压比较器转换成一个模拟量信号，从而对其进行读取。将模拟信号传输至 AD 模块，转换成数字信号后，发送给单片机进行分析和信号处理，然后通过单片机的串口对 Wi-Fi 模块进行控制，向云服务器发送传感器信息，小程序与云服务器交互，展示报警信息。本设计对基于云平台的家庭火灾检测系统的设计思路和系统组成方案，对 STM32 主控模块、Kalman Filter 算法、HTTPS 通信协议、I²C 协议、SPI 协议、RESTful 架构、小程序开发、Linux 操作系统、Nginx Web 服务器等进行了较深入的分析和研究。本设计实现的火灾报警系统具有性价高、硬件结构简单等亮点。程序基于 RT-Thread 开源嵌入式系统，使整个软件部分程序易于维护，方便移植。本设计是基于云平台的家庭火灾报警系统，实现了系统的远程报警、远程控制功能、远程更新固件等功能，达到了远程监控屋内环境的目标，具有非常好的应用前景。

关键词：烟雾报警 单片机 温度采集 无线通信

第 1 章 项目概述

1.1 项目背景

在当今社会，科学技术几乎占据了我们的日常生活，使我们的生活更加舒适，但它给我们带来了所有的便利，也给我们带来了许多潜在的危险。我们经常可以在网上和新闻上看到一些煤气泄漏引起的爆炸和意外火灾。使用火灾报警器为了在火灾初期监测和报告火灾情况，通过报警，及时预防和减轻火灾造成的灾害。为了减少事故的发生，我们必须采取实时监控烟雾浓度的方法。这样我们才能尽快发现潜在的事故，避免事故进一步恶化，要把这一切都扼杀在萌芽状态。

目前，我国的烟雾报警系统是安装在大型场所防火，但在其他国家，许多城镇的居民都安装了火灾报警系统，但我国对这个问题还没有给予足够的重视。近年来，中国人口逐渐增加，许多建筑也已经拔地而起。虽然方便了大家，但是相应的发生火灾的可能性也大大提高了，尤其是对于高楼层的住户。一旦发生火灾，极有可能造成疏散困难，极易造成人员伤亡和经济损失。而且如果不提前关注，甚至有可能引发范围更广、影响程度更大的火灾。因此，预防家庭火灾迫在眉睫。本设计是一种结构相对简单、成本低廉的烟雾报警器，以防患于未然，扩大市场的销售范围。

1.2 项目意义

今天，在我们的生活中，我们经常听到关于燃烧气体爆炸和有毒气体泄漏造成爆炸和有毒气体泄漏的新闻报道。因此，市场上出现了各种火灾报警装置，尤其是无线报警装置发展迅速。在这个科技与社会同步发展的时代，无线通信网络是最重要的。一些高科技报警器会利用无线通信来达到无线远程报警的目的。

因此设计出性能可靠且经济实惠的无线报警系统已经成为市场需求。基于网络的烟雾报警系统利用 Wi-Fi 模块进行发送报警信息，以 STM32 单片机为控制中心，进行远程控制。实现远程报警呼救、远程控制等功能。

第 2 章 系统搭建

2.1 项目工作原理

本系统由单片机、服务器、小程序等组成。 MQ-2 烟雾传感器采集模拟信号，将信号发送到 AD 转换模块，单片机通过 SPI 协议和 AD 模块进行通讯，读取烟雾浓度。单片机通过 I²C 协议和 LM75 温度传感器模块进行通信，读取当前温度。单片机使用卡尔曼滤波，对传感器数据进行处理，将处理结果，通过 Wi-Fi 模块上传至云服务器，最后用户终端的小程序与云服务器数据交互，从而实现火灾报警系统的整体功能。拓扑结构，如图 2.1 所示。

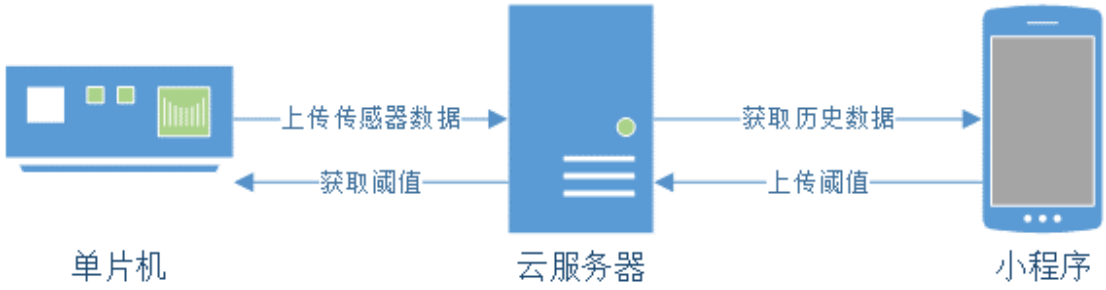

图 2.1 无线火灾报警系统拓扑结构

2.2 系统组成

2.2.1 项目设备清单

该系统有以下几个模块组成，分别为:

MQ-2 烟雾传感器、LM75 温度传感器、ADS1118 AD 转换芯片、CH340 串口转 TTL 芯片、STM32F411CEU6 单片机控制芯片、ESP8266 串口 Wi-Fi 模块、Virmach 虚拟服务器、小程序用户终端。

表 2.1 项目设备清单

名称	个数
MQ-2 烟雾传感器模块	1 块
LM75 温度传感器模块	1 块
ADS1118 AD 转换芯片	1 块
CH340 串口转 TTL 芯片	1 块
STM32F411CEU6 单片机控制芯片	1 块
ESP8266 串口 Wi-Fi 模块	1 块
Virmach 虚拟服务器	1 台

2.2.2 STM32F411CEU6 开发板介绍

STM32F411CEU6-MiniF4 是 WeActTC 推出的一款基于 ARM Cortex-M4 内核的开源硬件开发板，其最高主频为 100Mhz，该款开发板具有非常丰富的板载资源，能够充分施展 STM32F411CEU6 这款芯片性能。

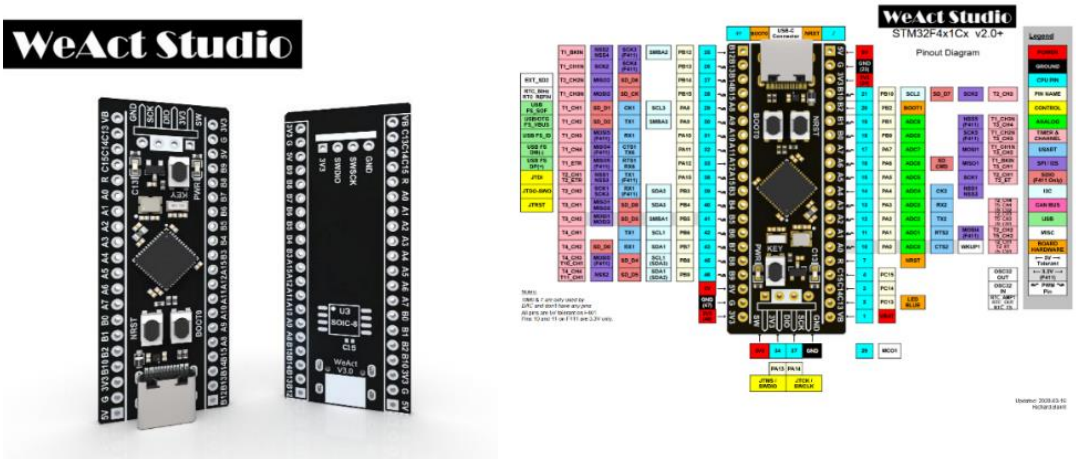

图 2.2 STM32 开发板的外观与引脚功能图

2.2.3 虚拟服务器介绍

云服务器端使用的是 Virmach 的虚拟服务器，使用的是 KVM 架构。搭载 Debian 10 buster 的 Linux 发行版，内核为 5.7.7，内存为 256 MB。

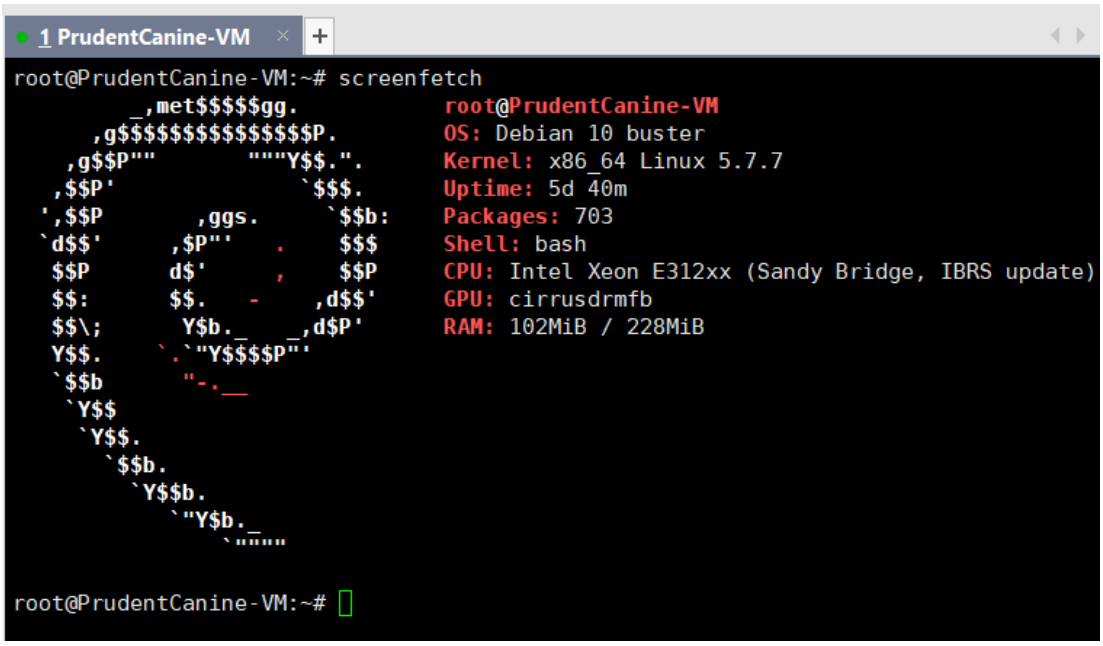

图 2.3 云服务器硬件配置

第 3 章 软件设计

3.1 单片机控制器软件设计

3.1.1 单片机资源分配

通过对 STM32 单片机片上 ROM 和片外 Flash 的合理规划，可以充分压榨有限的存储空间。该设计将片上前 128 KB 内容划分给 bootloader 分区，作为单片机系统的引导，担负着 OTA 远程下载固件，保证 app 分区固件完整的重要作用。片上剩余的 384 KB 内容划分给 app 分区，作为最主要的核心，采集传感器数据，滤波，上传和同步云服务器数据。片外前 512KB 内容划分给 factory 分区用作存放默认固件，当 app 分区的固件出现错误，可以及时恢复。之后的 512KB 内容划分给 download 分区，用于临时存放下载固件，更具情况自动升级固件。最后剩余的 15MB 用于挂载文件系统，供存放在 app 分区中的主程序进行挂载使用。

表 3.1 单片机 Flash 资源分配

分区名	Flash 设备名	偏移地址(KByte)	大小(KByte)
bootloader	onchip_flash_16k	0	64
param	onchip_flash_64k	0	64
app	onchip_flash_128k	0	384
factory	NOR_FLASH_DEV_NAME	0	512
download	NOR_FLASH_DEV_NAME	512	1024
filesys	NOR_FLASH_DEV_NAME	1024	15 * 1024

3.1.2 Bootloader 程序设计

制作 Bootloader 主要是为了实现 OTA 功能， OTA 升级其实就是 IAP 在线编程。在本设计中，OTA 通常通过串口等方式，将升级数据包下载到 Flash，然后将下载得到的数据包搬运到 MCU 的代码执行区域进行覆盖，以完成设备固件升级更新的功能。

在本设计的系统方案中，要完成一次 OTA 固件远端升级，需要进行以下两个核心阶段：

1. 单片机从远端下载最新的 OTA 固件。

2. Bootloader 对 OTA 固件进行 CRC 校验、AES256 解密和搬运到 app 分区。

为了在单片机有限的 ROM 空间上，实现 ROM 的最大利用率，在打包制作固件时，该设计可以使用以下 3 种压缩算法，见表 3.2。

表 3.2 Bootloader 支持的压缩算法

算法	压缩率
gzip	57.06%
quicklz	72.61%
fastlz	75.81%

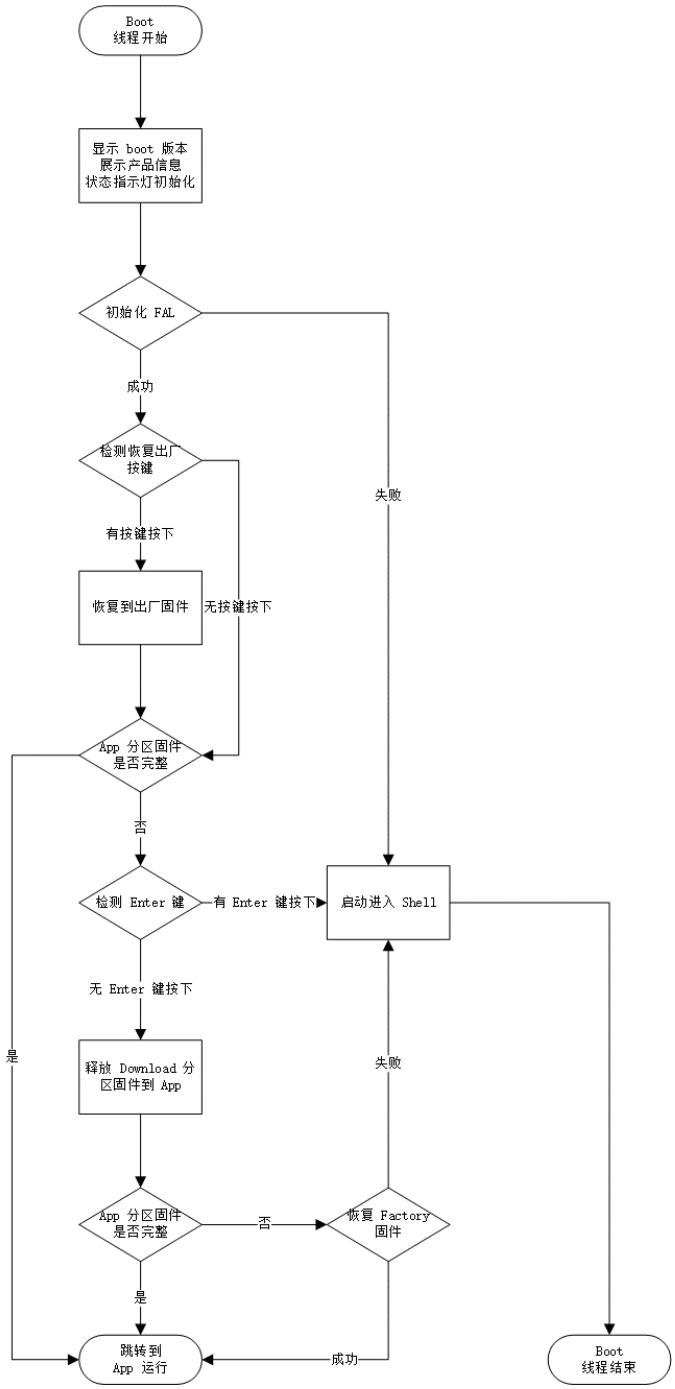

图 3.1 Bootloader 流程图

防止非法固件升级，该设计还使用产品识别码来辅助验证固件的可靠性。

3.1.3 传感器数据采集与处理程序设计

该设计主要通过检测当前温度和空气中烟雾浓度来判断是否报警。

采集部分程序主要使用 C/C++ 编写，使用开源嵌入式操作系统 RT-Thread。

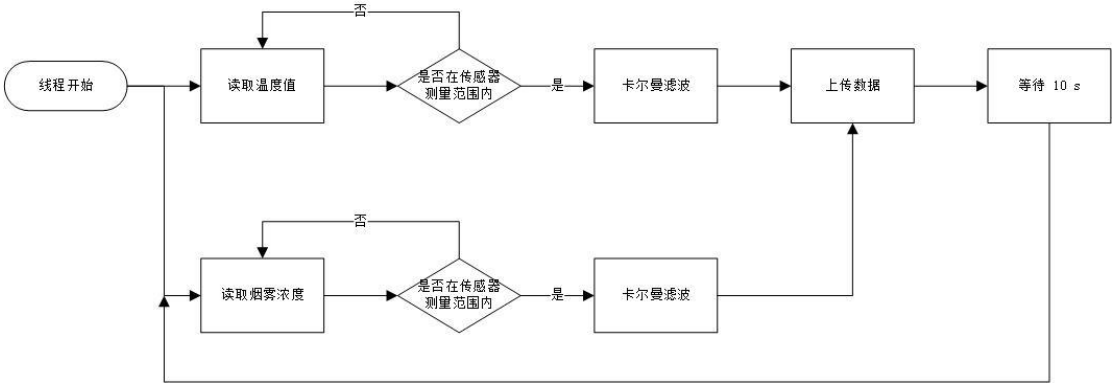

图 3.2 传感器数据采集处理流程图。

1.采集

MQ-2 烟雾传感器 AO 引脚输出的模拟量信号，通过 ADS1118 AD 芯片转换成数字量信号。

ADS1118 是使用 SPI 协议与单片机通信，通信时序图如下：

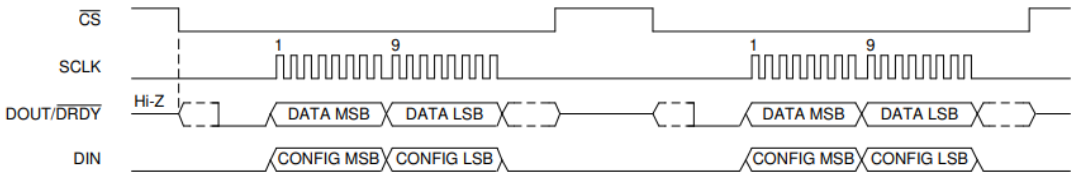

图 3.3 ads1118 SPI 时序。

该设计的 ADS1118 AD 模块和 W25Q128 SPI Flash 共用单片机 SPI1 总线，此处将 ADS1118 注册到系统中，名称为 spi11。单片机通过控制 CS 引脚对 ADS1118 进行片选,CS 引脚为低电平时,此时 ADS1118 可以与单片机通信。

SPI 的工作时序模式是由 CPOL（时钟极性）和 CPHA（时钟相位）之间的相位关系决定，CPOL 表示的是时钟信号的初始电平的状态，CPOL 为 0 时，表示的是时钟信号初始状态为低电平，为 1 时，表示的是时钟信号的初始电平是高电平。本设计中使用的工作时序模式为 CPOL = 0, CPHA = 1。

LM75 温度传感器是使用 I²C 协议与单片机通信，通信时序图如下：

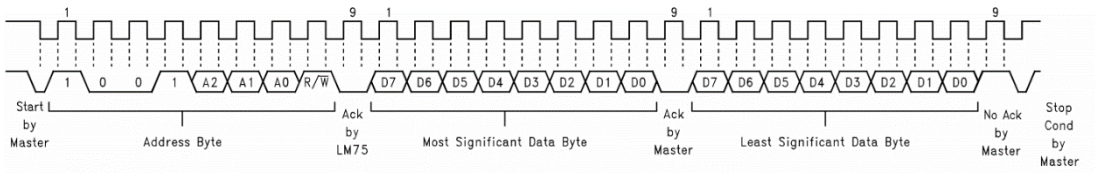

图 3.4 LM75 I²C 时序

对于温度传感器这类常见传感器，对接 RT-Thread 的 Sensor 框架，主要

是为主程序方便控制，提高代码的可移植性、可维护性。

通过阅读芯片手册可知 LM75 芯片 I²C 从机地址为 1001$A_2A_1A_0$，将芯片 A_2、A_1、A_0引脚接地。此时 I²C 从机地址为$(1001000)_2=(0x48)_{16}$。

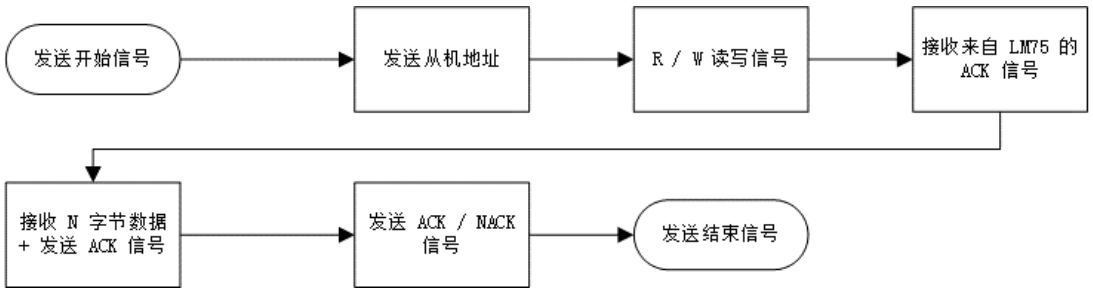

图 3.5 I²C 时序流程

开始条件： SCL 引脚为高电平时，单片机将 SDA 引脚拉低成低电平，表示数据传输即将开始。

从机地址： 单片机发送的第一个字节为 LM75 温度传感模块的从机地址，高 7 位为地址，最低位为 R/W 读写控制位，0 表示写操作，1 表示读操作。

应答信号： 每传输完成一个字节的数据，就需要向 LM75 温度传感模块回复一个 ACK 信号。

数据： 从机地址发送完后开始传输数据，由 LM75 温度传感模块发送，每个数据为 8 位，数据的字节数共 16 位。

停止条件： 在 SDA 引脚为低电平时，单片机将 SCL 引脚拉高并保持高电平，然后在将 SDA 引脚拉成高电平，表示传输结束。

2.滤波

为了减少误报，传感器采集部分使用了卡尔曼滤波算法来估计当前最优的温度值与烟雾浓度值，以此减少误报概率。

卡尔曼滤波器是一种优化估算的算法，该设计主要使用了以下公式：

时间更新：

$$\hat{x}_{\bar{k}} = A\hat{x}_{k-1} + Bu_{k-1} \quad (式\ 3.1)$$

$$P_{\bar{k}} = AP_{k-1} + A^T + Q \quad (式\ 3.2)$$

状态更新：

$$K_k = P_{\bar{k}}H^T(HP_{\bar{k}}H^T+R)^{-1} \quad (式\ 3.3)$$

$$\hat{x}_k = \hat{x}_{\bar{k}} + K_k(z_k - H\hat{x}_{\bar{k}}) \quad (式\ 3.4)$$

$$P_k = (I - K_kH)P_{\bar{k}} \quad (式\ 3.5)$$

$\hat{x}_k$和$\hat{x}_{k-1}$：是 Kalman Filter 滤波结果之一。它们分别表示时间 k 和时间 $k-1$ 的状态最佳估计值。

$\hat{x}_{\bar{k}}$：是滤波计算的中间结果，k 时刻的先验状态估计值，这是根据 $k-1$ 时刻的最佳估计值，预测出的 k 时刻的结果。

P_k和P_{k-1}：是滤波的结果之一，它们分别表示 k 时刻和 $k-1$ 时刻的后验估计协方差（就是 $\hat{x}_k$ 和$\hat{x}_{k-1}$ 的协方差，这里用来表示状态的不确定度）。

$P_{\bar{k}}$：k 时刻的先验估计协方差（就是$\hat{x}_{\bar{k}}$ 的协方差），这也是滤波计算的中间结果。

H：是滤波计算的前提条件之一，这是状态变量到测量的变换矩阵，表示将当前状态和观测值连接起来的关系。从数据来看这里呈现为线性关系，负责将 m 维的测量值转化为 n 维，并使其符合状态变量的数学形式。

z_k：是滤波计算的输入，这里代表测量值。

K_k：是滤波计算的中间结果，这里表示滤波增益矩阵。

A：状态转移矩阵，实际上是对目标状态转移的一个猜想模型。当状态转移矩阵不符合目标状态转移模型时，滤波会迅速发散。

Q：过程激励噪声的协方差。该参数用于表示状态转换矩阵和实际过程之间的误差。由于我们不能直接观测到过程信号，很难确定卡尔曼滤波器用来估计离散时间过程状态变量的 Q 值。

R：这是卡尔曼滤波器的已知条件之一，用来记录测量噪声协方差。当滤波器实际实现时，通常可以观察到测量噪声协方差 R。

B：是把输入转化为状态的矩阵。

$(z_k - H\hat{x}_{\bar{k}})$：实际观测值和预测观测值的残差，与卡尔曼增益一起修正先验状态估计值，然后计算得到后验状态估计值。

3.1.4 Wi-Fi 通信程序设计

1.配置

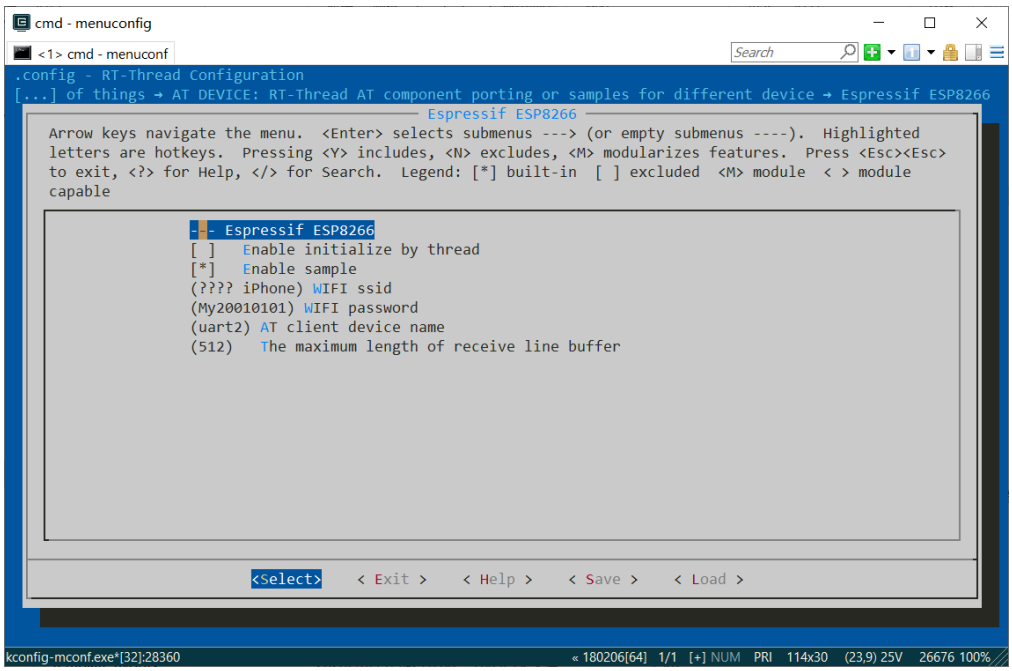

图 3.6 RT-Thread menuconfig 配置页面

通过使用 Kconfig 配置、 Scons 裁剪 RT-Thread 组件，将 ESP8266 作为网卡注册到系统中。开启 AT Client。此时单片机(AT 客户端)和 ESP8266(AT 服务器）间的设备连接与数据通信的方式。 其基本结构如下图所示：

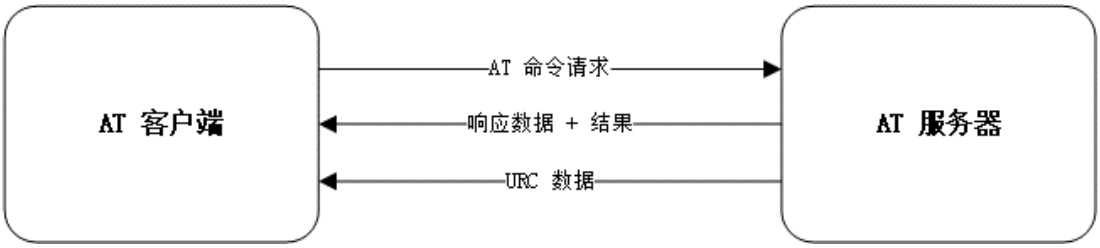

图 3.7 RT-Thread AT 框架流程

通过 Web 服务使用 RESTful 架构，单片机端使用 GET， POST， PUT 和 DELETE 四种请求方式分别对指定的 URL 资源进行增删改查操作。使用 WebClient 和 cJSON 开源项目，并对 GET 和 POST 请求程序进行封装，使用 JSON 格式发送传感器数据和获取设置阈值。

2.加密

通过之前的收集材料，发现之前类似的产品，很少在数据传输时，使用加密或者混淆，而如今网络安全问题已经成为一个全球问题，本设计在数据传输时，全部使用了 HTTPS 协议，TLS1.2 加密， 使用了 Baltimore CyberTrust Root 的 CA 证书。

本设计使用 ARM 公司开源的 mbedTLS 项目，来实现 TLS 加密。

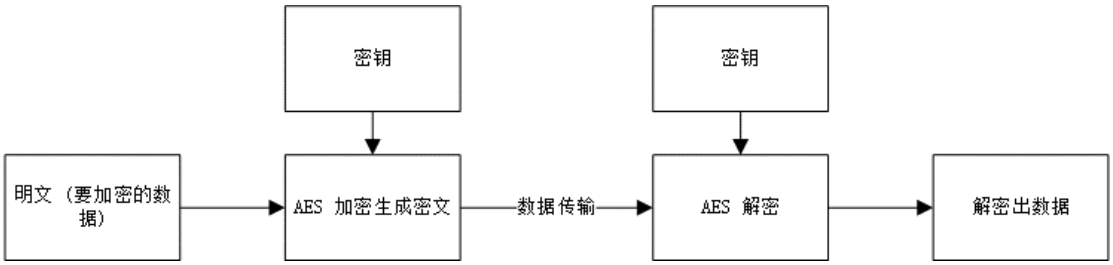

图 3.8 加密传输过程

3.2 云服务器端

3.2.1 CDN 加速

由于虚拟服务器在美国纽约州布法罗，距离遥远，延迟高，丢包严重。

为此，本设计中使用 CDN 来解决以下 2 个问题：

1. 可以减轻服务器的访问压力，将请求发送到不同的缓存服务器中，这样不仅可以加快了访问速度，而且也可以有效的抵抗住 DDOS 攻击。

2. 由于不同的访客访问的可能是不同的缓存服务器中的内容，所以这样可以隐藏服务器真实的 IP 地址，使服务器不容易受到攻击。

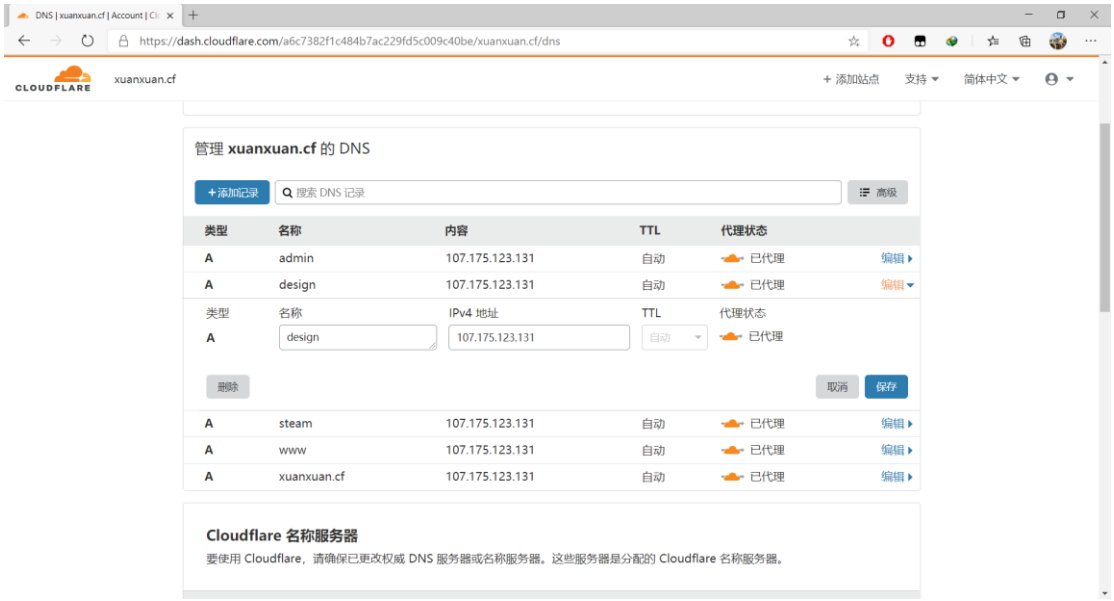

图 3.9 Cloudflare CDN 代理设置页面

3.2.2 BBR 加速

当数据路由拥挤时，开启 BBR 能够更加有效地处理流量。

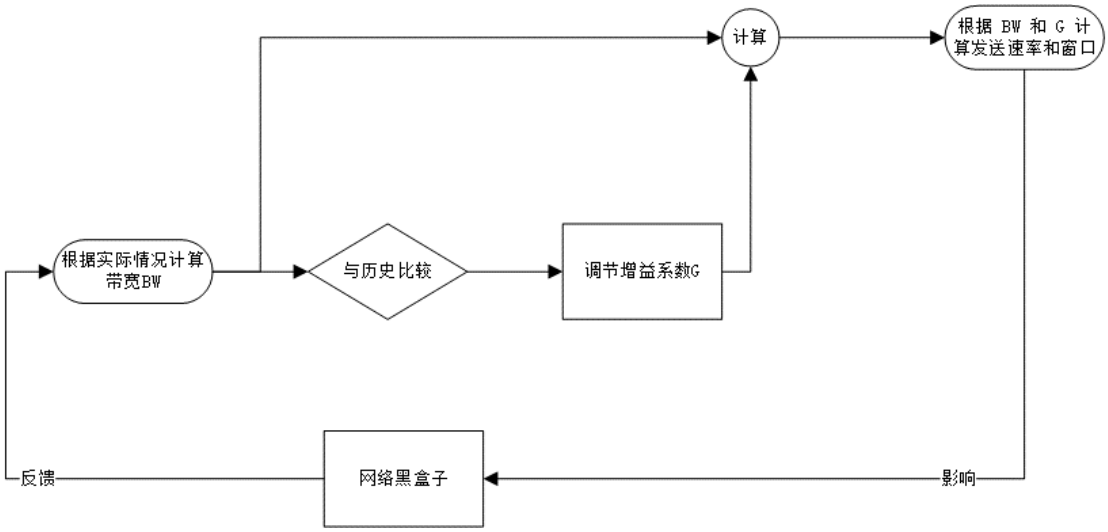

图 3.10 Google TCP BBR 拥塞控制算法流程

3.2.3 Web 服务

主要使用 Python 编写，使用 Django REST Framework 框架搭建 RESTful 服务，对单片机和小程序发送的实时数据进行接收和处理。

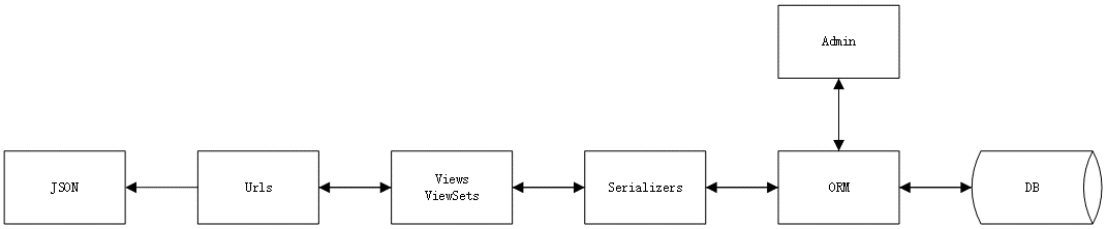

图 3.11 Django REST Framework 框架

其中 Serializer 模块是该模块的核心，发送数据时通过序列化将 Django 中的模型类对象转换为 JSON 字符串。在接收时，通过反序列化将 JSON 字符串转换为 Django 中的模型类对象。通过编写程序注释，使用 coreapi，生成规范的接口文档。

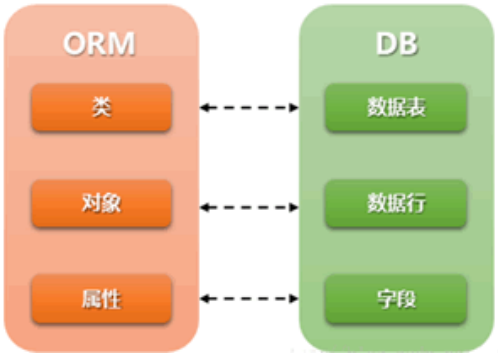

图 3.12 ORM 与数据库的映射关系图

ORM 模块也是一个重要模块，用于存储 Django 模型类对象。ORM 把类映射

成数据库中的表，把 Model 类的一个实例对象映射成数据库中的数据行，然后把 Model 类的属性映射成表中的字段，再通过 Model 对象的操作对应到 SQLite 表的操作，最后实现了 Django 对象到 SQL、SQL 到 Django 对象转换过程。

3.2.4 服务器通信

该设计使用 Let's Encrypt 证书，用于和 CDN 服务器加密通信。

CDN 上，启用边缘证书并开启 HTTP 严格传输安全协议，这样单片机端和小程序端进行加密通信时，使用的是 Cloudflare 签发的证书，安全性更高。

单片机和小程序先是通过 HTTPS 握手开启一条客户端和 Web 服务器之间的 SSL 安全通道。

其中数字证书包括一个公共密钥和一个私用密钥。服务器生成的公共密钥用来加密信息，单片机和小程序生成的私用密钥用来解译公共密钥加密的信息。

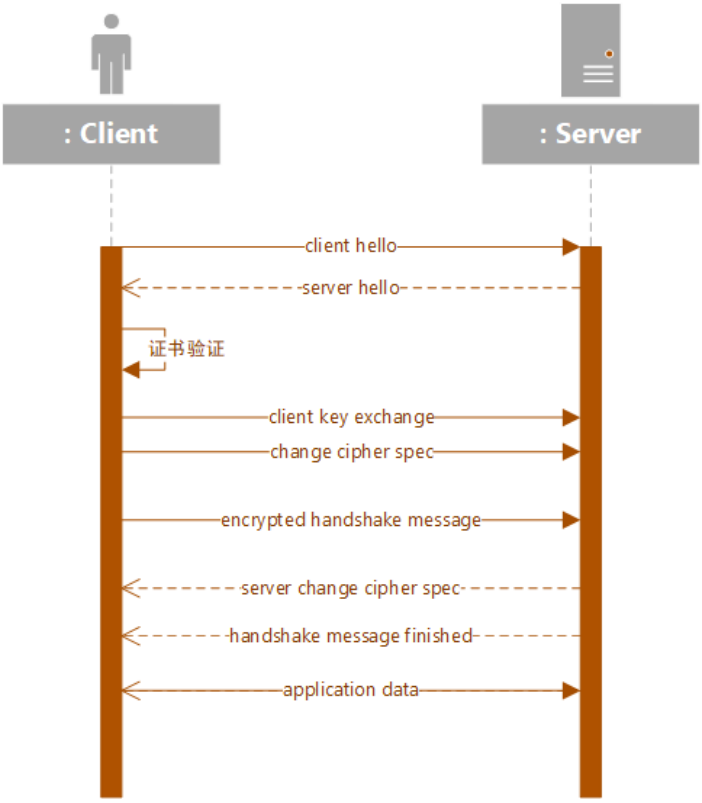

图 3.13 HTTPS 握手时序图

之后，单片机和小程序的 HTTPS 请求先是到 Nginx Web 服务器，解密 TLS 数据包，然后通过反向代理到 uWSGI Web 服务器，然后 uWSGI 通过 WSGI 协议与用 Django 搭建的 RESTful 服务进行通信。

图 3.14 服务器内部服务通信

3.3 小程序客户端

3.3.1 传感器数据可视化

该设计的小程序端基于开源的 uni-app 前端框架开发，整体以 Vue 为主体语言。

通过向云服务器发送 GET 请求，获得之前单片机端上传的数据，通过序列化将 JSON 数据转为 uCharts 图表所要求的数据格式，更新图表，实现数据可视化。

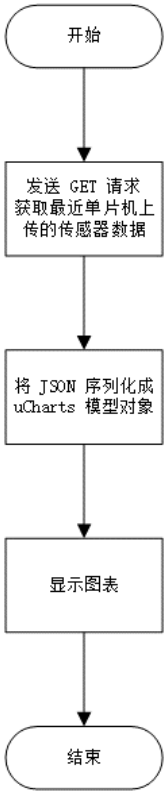

图 3.15 图表展示流程图

3.3.2 设置阈值

通过使用 Slider 组件，通过滑动设置阈值，点击提交按键生成 Form 表单，通过 PUT 请求，将 JSON 数据上传至服务器。等待单片机同步设置。

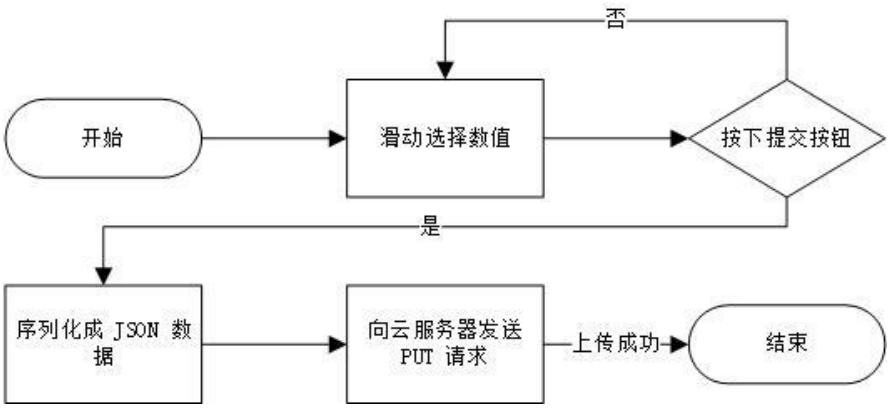

图 3.16 阈值设置流程图

第 4 章 系统测试与运行

4.1 系统测试

安装的时候看清楚元器件的正反。先通过稳压电源供电，检查各个模块工作指示灯是否正常亮起，是否存在短路情况。

通过 CH340 串口转 TTL 模块，将板子与电脑相连，打开 XShell，设置串口波特率。如图 4.1。

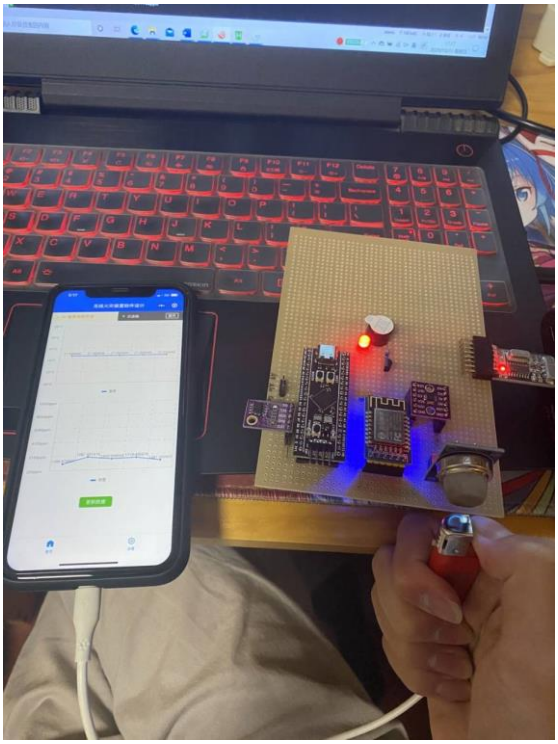

图 4.1 无线火灾报警装置

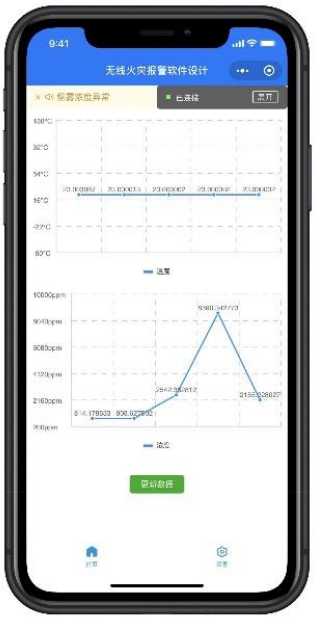

图 4.2 小程序报警信息展示

4.2 系统运行

系统上电，单片机进入 Bootloader。将打包好的固件使用 YModem 协议发送到单片机 download 分区。

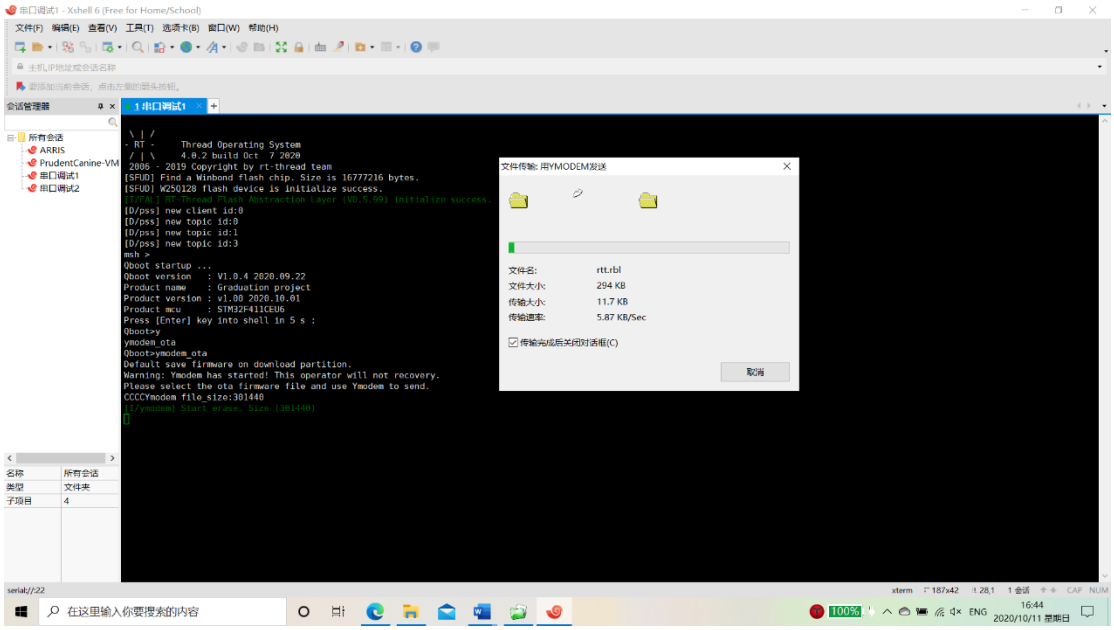

图 4.3 YModem OTA

下载完毕，系统重启。将 download 固件解密后搬运至 app 分区，然后跳转至 app 分区运行。

STM32 板载的小蓝灯开始以 1Hz 的频率开始闪烁。串口没有报错，证明系统开始正常运行。

将打火机对准烟雾传感器喷气，数值超过设定阈值开始报警，小程序端显示报警信息。

温度和烟雾浓度的历史数据，以折线图方式呈现。

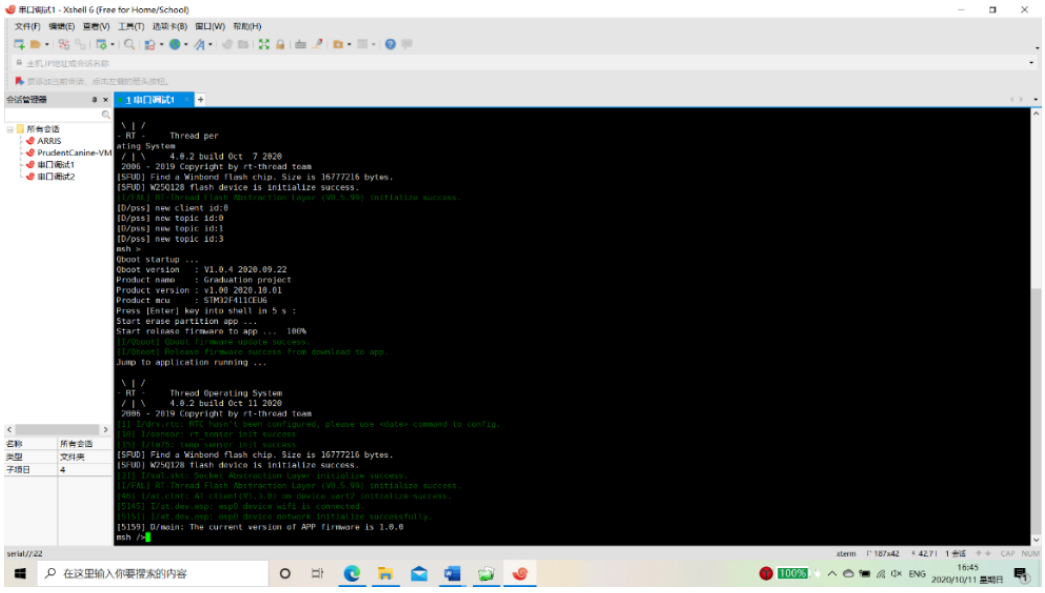

图 4.4 串口日志

第 5 章 总结与展望

5.1 总结

从 10 月初开始毕业设计，到现在论文也基本完工。从最初的茫然，到逐渐进入状态，整个写作过程难以用语言来表达。为了解决各种奇怪问题，我上网查阅资料，翻阅书籍，请教老师，也从中训练了自己发现问题与解决问题的能力。通过这几周的努力，也让我意识到自身还存在一些不足之处，还需通过刻苦学习来弥补。在利用网络来查询硬件方面相关资料的时候，可能由于这款芯片在国内使用率比较低，缺少正规的中文翻译，这让我意识到在注重专业技能训练的同时，更应当加强英语学习。

5.2 展望

本设计只实现了一台设备一部手机的远程控制设定，如果还需要多设备同时运行，还需要对服务器 Web 服务部分架构进行相应的优化与重构。单片机端，如果想要更改其他 Wi-Fi，还需要在程序中重新设定。若是能简单方便的更改使用 Wi-Fi，例如微信 AirKiss 或者用 Bluetooth 进行一键配网，就必须购买相应模块并编写相应的程序，此功能还需进一步完善。其次是由于环境引起的延误报警的发生，从原理上来说这是云服务器和单片机报警系统的物理距离过于遥远，网络延迟太高的情况下产生的，这也是不可避免的。

参考文献

[1]Qiu Yi,Xiong Puxiang,Tianlong Zhu. The Design and Implementation of the RT-Thread Operating System[M].CRC Press:2020-09-25.

[2]梁玉,谭兴望,胡翔,龚缘,陈燚,张修军.基于 Vuejs 和 Rest-Framework 的手续流程网站设计与实现[J].电脑编程技巧与维护,2020(09):51-54.

[3]贾伟伟,王朋伟.基于 RT-Thread 的空气质量检测系统[J].电子测试,2020(17):18-20.

[4]许镭韬,吕志刚,黄义国,李晓艳.基于 STM32 单片机的 AES 算法优化与实现[J].自动化仪表,2020,41(07):56-60.

[5]王莉莉,包红军,李致家.基于 Kalman 滤波的实时校正模型研究[J].水力发电,2020,46(08):24-26+47.

[6]严娟,张玉川,杨鹏翔,李欣,卫明慧.基于以太网 OTA 远程升级的研究[J].上海汽车,2020(03):15-18+27.

[7]梁剑.TCP BBR 拥塞控制算法的研究[J].山西能源学院学报,2019,32(03):97-99.

[8]邢进. 基于 RT-Thread 的嵌入式网络控制器软件设计[D].海南大学,2019.

[9]陈大莉. 基于 FPGA 的 GZIP 解压缩算法的设计和实现[D].西安电子科技大学,2016.

致谢

在本次毕业设计完成之际，首先我要感谢我此次毕业设计的指导老师袁老师。从开题报告到毕业设计的完成，我始终感受着老师的精心教导和无私的关怀，使我受益匪浅，再次向老师表示深深的感谢与最崇高的敬意。

本次毕业设计的完成，不仅离不开袁铭老师的耐心教导，也离不开同学的关心与帮助。感谢同学们在此期间给予我无言的帮助，感谢他们为我的毕业设计提出宝贵的意见和建议。

距离离校的日子也已日趋渐进，随着毕业论文的完成也随之进入了尾声。在此，我想衷心的感谢所有同学和老师们。感谢之情难以用言语来度量，请允许我以最朴实的语言，致以最崇高的敬意。

附录 源程序清单

ADS1118 驱动程序：

```
/*
 * Copyright (c) 2006-2018, RT-Thread Development Team
 *
 * SPDX-License-Identifier: Apache-2.0
 *
 * Change Logs:
 * Date           Author       Notes
 * 2020-10-01     Chenxuan     first version
 */

#include "ads1118.h"

#define DRV_DEBUG
#define LOG_TAG              "ads1118"

#include <drv_log.h>

#include <rtthread.h>
#include <rtdevice.h>
#include <board.h>
#include <drv_spi.h>

#if defined(BSP_USING_ADS1118)
float ads1118_get_temperature(const char *name)
{
    union ADS1118_CONFIG cfg = { 0 };
    cfg.field.SS = ADS1118_REG_SS_START;
    cfg.field.MUX = ADS1118_REG_MUX_AIN0_GND;
    cfg.field.PGA = ADS1118_REG_PGA_4096MV;
    cfg.field.MODE = ADS1118_REG_MODE_DEFAULT;
    cfg.field.DR = ADS1118_REG_DR_860SPS;
    cfg.field.TS_MODE = ADS1118_REG_TS_MODE_TEMP;
    cfg.field.PULL_UP = ADS1118_REG_PULL_UP_ENABLE;
    cfg.field.NOP = ADS1118_REG_NOP_VALID;
    cfg.field.RESERVED = ADS1118_REG_RESERVED;
    ads1118_read_data(name, cfg.reg);
    ads1118_read_data(name, cfg.reg);
    rt_thread_mdelay(5);
    return ads1118_read_data(name, cfg.reg);
}
```

```
float ads1118_read_data(const char *name, const uint16_t cmd)
{
    uint16_t recv_buf = 0;
    union ADS1118_CONFIG cfg = { 0 };
    struct rt_spi_device *dev = (struct rt_spi_device *)rt_device_find(name);
    if (dev == RT_NULL)
    {
        LOG_D("Can't find %s device!", ADS1118_SPI_DEVICE_NAME);
        return 0;
    }
    else
    {
        rt_spi_transfer(dev, &cmd, &recv_buf, 1);
        cfg.reg = cmd;
        if (cfg.field.TS_MODE == ADS1118_REG_TS_MODE_TEMP)
        {
            float temperature = 0;
            recv_buf >>= 2;
            if (recv_buf & 0x2000)
            {
                recv_buf = ~recv_buf + 1;
                recv_buf &= 0x7FF;
                temperature = -recv_buf * 0.03125;
            }
            else
            {
                temperature = recv_buf * 0.03125;
            }
            return temperature;
        }
        else
        {
            float voltage = recv_buf * 1.0 / 32768 * 4.096;
            if (voltage < 0)
            {
                voltage = 0;
            }
            else if (voltage > 3.3)
            {
                voltage = 3.3;
            }
            return voltage;
        }
    }
```

```
}

static int ads1118_hw_init(void)
{
    rt_hw_spi_device_attach(ADS1118_SPI_BUS, ADS1118_SPI_DEVICE_NAME, GPIOB, GPIO_PIN_1);
    struct         rt_spi_device         *dev         =         (struct         rt_spi_device
*)rt_device_find(ADS1118_SPI_DEVICE_NAME);
    if (dev == RT_NULL)
    {
        LOG_D("Can't find %s device!", ADS1118_SPI_DEVICE_NAME);
        return -RT_ERROR;
    }
    else
    {
        // Configure
        struct rt_spi_configuration cfg = { 0 };

        cfg.data_width = 16;
        cfg.max_hz = 30 * 1000 * 1000;      // 30M
        cfg.mode = RT_SPI_MASTER | RT_SPI_MODE_1 | RT_SPI_MSB;

        rt_spi_configure(dev, &cfg);
    }
    return RT_EOK;
}

INIT_DEVICE_EXPORT(ads1118_hw_init);

#endif
```

LM75 温度传感驱动程序：

```
/*
 * Copyright (c) 2006-2020, RT-Thread Development Team
 *
 * SPDX-License-Identifier: Apache-2.0
 *
 * Change Logs:
 * Date           Author       Notes
 * 2020-10-01     Chenxuan    first version
 */

#include <rtthread.h>
#include <rtdevice.h>
```

```
#define DBG_TAG "lm75"
#define DBG_LVL DBG_LOG
#include <rtdbg.h>

#define SENSOR_TEMP_RANGE_MAX (125)
#define SENSOR_TEMP_RANGE_MIN (-55)

#define LM75_I2C_BUS_NAME          "i2c1"  /* 传感器连接的 I2C 总线设备名称 */
#define LM75_ADDR                  0x48    /* 从机地址 */

#if defined(BSP_USING_LM75)

#include <sensor.h>
static float lm75_get_temperature(const char *name, uint16_t addr)
{
    uint8_t buf[2] = { 0 };
    uint16_t tmp = 0;
    float temperature = 0;
    struct rt_i2c_bus_device *i2c_bus = (struct rt_i2c_bus_device *)rt_device_find(name);

    if (i2c_bus == RT_NULL)
    {
        LOG_D("Can't find %s device!", name);
        return 0;
    }
    else
    {
        rt_i2c_master_recv(i2c_bus, addr, 0, buf, 2);
    }

    tmp = (buf[0] * 0x100 + buf[1]) >> 5;
    if (tmp & 0x400)
    {
        tmp = ~tmp + 1;
        tmp &= 0x7FF;
        temperature = -tmp * 0.125;
    }
    else
    {
        temperature = tmp * 0.125;
    }

    return temperature;
}
```

```
static rt_size_t _lm75_polling_get_data(rt_sensor_t sensor, struct rt_sensor_data *data)
{
    if (sensor->info.type == RT_SENSOR_CLASS_TEMP)
    {
        float temperature_x10 =
            10              *              lm75_get_temperature(sensor->config.intf.dev_name,
(rt_uint32_t)sensor->config.intf.user_data);
        data->data.temp = (rt_int32_t)temperature_x10;
        data->timestamp = rt_sensor_get_ts();
    }
    return 1;
}

static rt_size_t lm75_fetch_data(struct rt_sensor_device *sensor, void *buf, rt_size_t len)
{
    RT_ASSERT(buf);

    if (sensor->config.mode == RT_SENSOR_MODE_POLLING)
    {
        return _lm75_polling_get_data(sensor, buf);
    }
    else
    {
        return 0;
    }
}

static rt_err_t lm75_control(struct rt_sensor_device *sensor, int cmd, void *args)
{
    rt_err_t result = RT_EOK;

    return result;
}

static struct rt_sensor_ops sensor_ops =
    {
        lm75_fetch_data,
        lm75_control
    };

static int rt_hw_lm75_init(const char *name, struct rt_sensor_config *cfg)
{
    rt_int8_t result;
```

```
    rt_sensor_t sensor = RT_NULL;

    sensor = rt_calloc(1, sizeof(struct rt_sensor_device));
    if (sensor == RT_NULL)
    {
        return -RT_ENOMEM;
    }

    sensor->info.type = RT_SENSOR_CLASS_TEMP;
    sensor->info.vendor = RT_SENSOR_VENDOR_UNKNOWN;
    sensor->info.model = "lm75_temp";
    sensor->info.unit = RT_SENSOR_UNIT_DCELSIUS;
    sensor->info.intf_type = RT_SENSOR_INTF_I2C;
    sensor->info.range_max = SENSOR_TEMP_RANGE_MAX;
    sensor->info.range_min = SENSOR_TEMP_RANGE_MIN;
    sensor->info.period_min = 100;

    rt_memcpy(&sensor->config, cfg, sizeof(struct rt_sensor_config));
    sensor->ops = &sensor_ops;

    result = rt_hw_sensor_register(sensor, name, RT_DEVICE_FLAG_RDONLY |
RT_DEVICE_FLAG_FIFO_RX, RT_NULL);
    if (result != RT_EOK)
    {
        LOG_E("device register err code: %d", result);
        rt_free(sensor);
        return -RT_ERROR;
    }

    LOG_I("temp sensor init success");
    return RT_EOK;
}

static int rt_hw_lm75_port(void)
{
    struct rt_sensor_config cfg;
    cfg.intf.dev_name = LM75_I2C_BUS_NAME;
    cfg.intf.user_data = (void *)LM75_ADDR;
    rt_hw_lm75_init("lm75", &cfg);

    return RT_EOK;
}
INIT_COMPONENT_EXPORT(rt_hw_lm75_port);
```

```
#endif
```

卡尔曼滤波算法程序：

```
/**
 * @file lib_kalman.c
 * @author 赵晨瑄 (chenxuan.zhao@icloud.com)
 * @brief 卡尔曼滤波
 * @version 0.1
 * @date 2020-05-27
 *
 * @copyright Copyright (c) 2020
 *
 */

#include "lib_kalman.h"

#include <math.h>
#include <stdlib.h>

void KalmanFilter_Init(KalmanTypeDef *cfg) {
    cfg->x = 0;
    cfg->p = 5;
    cfg->A = 1;
    cfg->H = 1;
    cfg->q = 0.25;
    cfg->r = 1;
}

float KalmanFilter_Update(KalmanTypeDef *cfg, float measure) {
    cfg->x = cfg->A * cfg->x;
    cfg->p = cfg->A * cfg->A * cfg->p + cfg->q;

    cfg->gain = cfg->p * cfg->H / (cfg->p * cfg->H * cfg->H + cfg->r);
    cfg->x = cfg->x + cfg->gain * (measure - cfg->H * cfg->x);
    cfg->p = (1 - cfg->gain * cfg->H) * cfg->p;

    return cfg->x;
}
```

GET 和 POST 封装程序：

```
/*
 * Copyright (c) 2006-2020, RT-Thread Development Team
 *
 * SPDX-License-Identifier: Apache-2.0
```

```
 *
 * Change Logs:
 * Date           Author       Notes
 * 2020-10-01     Chenxuan     first version
 */

#include "http_webclient.h"

#define DBG_TAG "http_webclient"
#define DBG_LVL DBG_LOG
#include <rtdbg.h>

#define GET_HEADER_BUFSZ               2048
#define GET_RESP_BUFSZ                 2048
#define POST_HEADER_BUFSZ              2048
#define POST_RESP_BUFSZ                2048

int webclient_get_comm(const char *uri, std::string &response)
{
    struct webclient_session *session = nullptr;
    unsigned char *buffer = nullptr;
    int ret = 0;
    int bytes_read, resp_status;
    int content_length = -1;

    response.clear();

    buffer = (unsigned char *)web_calloc(GET_RESP_BUFSZ, sizeof(unsigned char));
    if (buffer == nullptr)
    {
        LOG_D("No memory for receive buffer.");
        ret = -RT_ENOMEM;
        goto __exit;

    }

    /* Create webclient session and set header response size. */
    session = webclient_session_create(GET_HEADER_BUFSZ);
    if (session == nullptr)
    {
        ret = -RT_ENOMEM;
        goto __exit;
    }
```

```
/* Send GET request by default header. */
if ((resp_status = webclient_get(session, uri)) != 200)
{
    LOG_D("Webclient GET request failed, response(%d) error.", resp_status);
    ret = -RT_ERROR;
    goto __exit;
}

content_length = webclient_content_length_get(session);
if (content_length < 0)
{
    do
    {
        bytes_read = webclient_read(session, (void *)buffer, GET_RESP_BUFSZ);
        if (bytes_read <= 0)
        {
            break;
        }

        for (int index = 0; index < bytes_read; index++)
        {
            response += buffer[index];
        }
    } while (true);
}
else
{
    int content_pos = 0;

    do
    {
        bytes_read = webclient_read(session, (void *)buffer,
            content_length - content_pos > GET_RESP_BUFSZ ?
            GET_RESP_BUFSZ : content_length - content_pos);
        if (bytes_read <= 0)
        {
            break;
        }

        for (int index = 0; index < bytes_read; index++)
        {
            response += buffer[index];
        }
```

```
            content_pos += bytes_read;
        } while (content_pos < content_length);
    }

__exit:
    if (session)
    {
        webclient_close(session);
    }

    if (buffer)
    {
        web_free(buffer);
    }

    return ret;
}

int webclient_post_comm(const char *uri, const char *post_data, size_t data_len, std::string
&response)
{
    struct webclient_session *session = nullptr;
    unsigned char *buffer = nullptr;
    int ret = 0;
    int bytes_read, resp_status;

    buffer = (unsigned char *)web_calloc(POST_RESP_BUFSZ, sizeof(unsigned char));
    if (buffer == nullptr)
    {
        LOG_D("No memory for receive response buffer.");
        ret = -RT_ENOMEM;
        goto __exit;
    }

    /* Create webclient session and set header response size. */
    session = webclient_session_create(POST_HEADER_BUFSZ);
    if (session == nullptr)
    {
        ret = -RT_ENOMEM;
        goto __exit;
    }

    /* Build header for upload. */
    webclient_header_fields_add(session, "Content-Length: %d\r\n", data_len);
```

```
    webclient_header_fields_add(session, "Content-Type: application/json\r\n");

    /* Send POST request by default header. */
    if ((resp_status = webclient_post(session, uri, post_data, data_len)) != 201)
    {
        LOG_D("Webclient POST request failed, response(%d) error.", resp_status);
        ret = -RT_ERROR;
        goto __exit;
    }
    do
    {
        bytes_read = webclient_read(session, buffer, POST_RESP_BUFSZ);
        if (bytes_read <= 0)
        {
            break;
        }

        for (int index = 0; index < bytes_read; index++)
        {
            response += buffer[index];
        }
    } while (true);
__exit:
    if (session)
    {
        webclient_close(session);
    }

    if (buffer)
    {
        web_free(buffer);
    }

    return ret;
}
```

云服务器端

序列化模块：

```
from rest_framework import serializers
from webApp import models


class UserSensorDataSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = models.UserSensorDataModel
        fields = '__all__'


class UserSensorConfigSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = models.UserSensorConfigModel
        fields = '__all__'
```

显示模块：

```
from django.shortcuts import render

# Create your views here.

from rest_framework import viewsets, filters, pagination
from django_filters.rest_framework import DjangoFilterBackend
from webApp import serializers, models


class UserSensorDataPagination(pagination.PageNumberPagination):
    page_size = 5
    max_page_size = 100


class UserSensorDataViewSet(viewsets.ModelViewSet):
    """
    无线火灾报警系统 获取装置当前状态 API。
    """
    queryset = models.UserSensorDataModel.objects.all()
    pagination_class = UserSensorDataPagination
    serializer_class = serializers.UserSensorDataSerializer
    filter_backends = [DjangoFilterBackend, filters.SearchFilter, filters.OrderingFilter]
    filter_fields = '__all__'
```

```
class UserSensorConfigViewSet(viewsets.ModelViewSet):
    """
    无线火灾报警系统 获取装置当前配置 API。
    """
    queryset = models.UserSensorConfigModel.objects.all()
    pagination_class = UserSensorDataPagination
    serializer_class = serializers.UserSensorConfigSerializer
    filter_backends = [DjangoFilterBackend, filters.SearchFilter, filters.OrderingFilter]
    filter_fields = '__all__'
```

数据库模块：

```
from django.db import models


# Create your models here.

class UserSensorDataModel(models.Model):
    temperature = models.FloatField('温度')
    smokescope = models.FloatField('烟雾浓度')
    updateTime = models.DateTimeField(auto_now=True)



class UserSensorConfigModel(models.Model):
    temperatureThreshold = models.FloatField('温度阈值')
    concentrationThreshold = models.FloatField('浓度阈值')
     updateTime = models.DateTimeField(auto_now=True)
```

路由模块：

```
"""djangoProject URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
```

```
from django.urls import path

from django.urls import path, include
from rest_framework import routers
from rest_framework.documentation import include_docs_urls
from webApp import views

router = routers.DefaultRouter()
router.register(r'data', views.UserSensorDataViewSet)
router.register(r'config', views.UserSensorConfigViewSet)

urlpatterns = [
    path('', include_docs_urls(title='火灾报警装置软件设计-接口文档')),
    path('api/v1/', include(router.urls)),
    path('api-auth/', include('rest_framework.urls', namespace='rest_framework')),
]
```

小程序端：

首页：

```
<template>
    <view>
        <uni-notice-bar             v-if="isDisplayNoteMessage"             showClose="true"
showIcon="true" :text="noteMessage"></uni-notice-bar>
        <view class="qiun-columns">
            <view class="qiun-charts">
                <!--#ifdef MP-ALIPAY -->
                <canvas               canvas-id="canvasLineA"               id="canvasLineA"
class="charts" :width="cWidth*pixelRatio" :height="cHeight*pixelRatio"
                 :style="{'width':cWidth+'px','height':cHeight+'px'}"
@touchstart="touchLineA" @touchmove="moveLineA" @touchend="touchEndLineA"></canvas>
                <!--#endif-->
                <!--#ifndef MP-ALIPAY -->
                <canvas     canvas-id="canvasLineA"     id="canvasLineA"     class="charts"
@touchstart="touchLineA" @touchmove="moveLineA"
                 @touchend="touchEndLineA"></canvas>
                <!--#endif-->
            </view>
        </view>
        <view class="qiun-columns">
            <view class="qiun-charts">
                <!--#ifdef MP-ALIPAY -->
                <canvas               canvas-id="canvasLineB"               id="canvasLineB"
class="charts" :width="cWidth*pixelRatio" :height="cHeight*pixelRatio"
                 :style="{'width':cWidth+'px','height':cHeight+'px'}"
@touchstart="touchLineB" @touchmove="moveLineB" @touchend="touchEndLineB"></canvas>
                <!--#endif-->
                <!--#ifndef MP-ALIPAY -->
                <canvas     canvas-id="canvasLineB"     id="canvasLineB"     class="charts"
@touchstart="touchLineB" @touchmove="moveLineB"
                 @touchend="touchEndLineB"></canvas>
                <!--#endif-->
            </view>
        </view>
        <view class="btn_view">
            <button :loading="isLoading" @click="getServerData" type="primary" size="mini">
更新数据</button>
        </view>
    </view>
</template>

<script>
```

```
import uCharts from '@/components/u-charts/u-charts.js'
import uniNoticeBar from '@/components/uni-notice-bar/uni-notice-bar.vue'

var _self
var canvaLineA = null
var canvaLineB = null
var lastMoveTime = null //最后执行移动的时间戳
var timerUpdate = null

export default {
    components: {
        uniNoticeBar
    },
    data() {
        return {
            cWidth: '',
            cHeight: '',
            pixelRatio: 1,
            InteractiveA: '', //交互显示的数据
            InteractiveB: '', //交互显示的数据
            temperature: 60,
            concentration: 2000,
            temperatureThreshold: 60,
            concentrationThreshold: 2000,
            noteMessage: '',
            isDisplayNoteMessage: false,
            isLoading: false,
            isInit: false,
        }
    },
    onLoad() {
        _self = this;
        //#ifdef MP-ALIPAY
        uni.getSystemInfo({
            success: function(res) {
                if (res.pixelRatio > 1) {
                    _self.pixelRatio = 2
                }
            }
        });
        //#endif
        this.cWidth = uni.upx2px(750)
        this.cHeight = uni.upx2px(500)
        timerUpdate = setInterval(callback => {
```

```
        this.getServerData()
    }, 1000 * 10)
    this.getServerData()
},
onShow() {
    this.getServerData()
},
onUnload() {
    clearInterval(timerUpdate)
    clearInterval(timerUpdate1)
},
methods: {
    toJSON() {},
    getServerData() {
        let that = this
        this.isLoading = true
        uni.request({
            // url: 'http://172.20.10.2:8000/api/v1/config/',
            url: 'https://design.xuanxuan.cf/api/v1/config/',
            data: {},
            success: function(res) {
                console.log(res)
                let results = res.data.results[0]
                that.temperatureThreshold = results.temperatureThreshold;
                that.concentrationThreshold = results.concentrationThreshold;
            },
            fail: () => {
                _self.tips = "网络错误，小程序端请检查合法域名"
            },
        })
        uni.request({
            // url: 'http://172.20.10.2:8000/api/v1/data/?ordering=-updateTime',
            url: 'https://design.xuanxuan.cf/api/v1/data/?ordering=-updateTime',
            data: {},
            method: 'GET',
            success: function(res) {
                let results = res.data.results
                let LineA = {
                    categories: ["", "", "", "", ""],
                    series: [{
                        name: '温度',
                        data: [0, 0, 0, 0, 0],
                    }]
                }
```

```
                let LineB = {
                    categories: ["", "", "", "", ""],
                    series: [{
                        name: '浓度',
                        data: [0, 0, 0, 0, 0],
                    }]
                }
                that.temperature = results[0].temperature
                that.concentration = results[0].smokescope
                for (let i = 0; i < 5; i++) {
                    LineA.categories[4 - i] += results[i].updateTime
                    LineA.series[0].data[4 - i] += results[i].temperature
                    LineB.categories[4 - i] += results[i].updateTime
                    LineB.series[0].data[4 - i] += results[i].smokescope
                }
                if (that.isInit) {
                    canvaLineA.updateData(LineA);
                    canvaLineB.updateData(LineB);
                } else {
                    _self.showLineA("canvasLineA", LineA)
                    _self.showLineB("canvasLineB", LineB)
                }
                that.isInit = true

                that.isLoading = false

                if (that.temperature > that.temperatureThreshold ||
that.concentration > that.concentrationThreshold) {
                    if (that.temperature > that.temperatureThreshold) {
                        that.noteMessage = '温度异常'
                    }
                    if (that.concentration > that.concentrationThreshold) {
                        that.noteMessage = '烟雾浓度异常'
                    }
                    that.isDisplayNoteMessage = true
                } else {
                    that.noteMessage = ''
                    that.isDisplayNoteMessage = false
                }
            },
            fail: () => {
                _self.tips = "网络错误，小程序端请检查合法域名"
                that.isLoading = false
            },
```

```
    })
},
showLineA(canvasId, chartData) {
    canvaLineA = new uCharts({
        $this: _self,
        canvasId: canvasId,
        type: 'line',
        fontSize: 11,
        legend: {
            show: true
        },
        dataLabel: true,
        dataPointShape: true,
        background: '#FFFFFF',
        pixelRatio: _self.pixelRatio,
        categories: chartData.categories,
        series: chartData.series,
        animation: false,
        xAxis: {
            type: 'grid',
            gridColor: '#CCCCCC',
            gridType: 'dash',
            dashLength: 8,
            disabled: true
        },
        yAxis: {
            gridType: 'dash',
            gridColor: '#CCCCCC',
            dashLength: 8,
            splitNumber: 5,
            min: -55,
            max: 125,
            format: (val) => {
                return val.toFixed(0) + '℃'
            }
        },
        width: _self.cWidth * _self.pixelRatio,
        height: _self.cHeight * _self.pixelRatio,
        extra: {
            line: {
                type: 'straight'
            }
        }
    })
```

```
},
touchLineA(e) {
    lastMoveTime = Date.now()
},
moveLineA(e) {
    let currMoveTime = Date.now()
    let duration = currMoveTime - lastMoveTime
    if (duration < Math.floor(1000 / 60)) {
        return
    }
    lastMoveTime = currMoveTime

    let currIndex = canvaLineA.getCurrentDataIndex(e)
    if (currIndex > -1 && currIndex < canvaLineA.opts.categories.length) {
        let riqi = canvaLineA.opts.categories[currIndex]
        let leibie = canvaLineA.opts.series[0].name
        let shuju = canvaLineA.opts.series[0].data[currIndex]
        this.InteractiveA = leibie + ':' + riqi + ' ' + shuju + '℃'
    }
    canvaLineA.showToolTip(e, {
        format: function(item, category) {
            return category + ' ' + item.name + ':' + item.data + '℃'
        }
    });
},
touchEndLineA(e) {
    let currIndex = canvaLineA.getCurrentDataIndex(e)
    if (currIndex > -1 && currIndex < canvaLineA.opts.categories.length) {
        let riqi = canvaLineA.opts.categories[currIndex]
        let leibie = canvaLineA.opts.series[0].name
        let shuju = canvaLineA.opts.series[0].data[currIndex]
        this.InteractiveA = leibie + ' ' + ':' + riqi + ' ' + shuju + '℃'
    }
    canvaLineA.touchLegend(e)
    canvaLineA.showToolTip(e, {
        format: function(item, category) {
            return category + ' ' + item.name + ':' + item.data + '℃'
        }
    })
},

showLineB(canvasId, chartData) {
    canvaLineB = new uCharts({
        $this: _self,
```

```
        canvasId: canvasId,
        type: 'line',
        fontSize: 11,
        legend: {
            show: true
        },
        dataLabel: true,
        dataPointShape: true,
        background: '#FFFFFF',
        pixelRatio: _self.pixelRatio,
        categories: chartData.categories,
        series: chartData.series,
        animation: false,
        xAxis: {
            type: 'grid',
            gridColor: '#CCCCCC',
            gridType: 'dash',
            dashLength: 8,
            disabled: true
        },
        yAxis: {
            gridType: 'dash',
            gridColor: '#CCCCCC',
            dashLength: 8,
            splitNumber: 5,
            min: 200,
            max: 10000,
            format: (val) => {
                return val.toFixed(0) + 'ppm'
            }
        },
        width: _self.cWidth * _self.pixelRatio,
        height: _self.cHeight * _self.pixelRatio,
        extra: {
            line: {
                type: 'straight'
            }
        }
    });
},
touchLineB(e) {
    lastMoveTime = Date.now()
},
moveLineB(e) {
```

```
                let currMoveTime = Date.now()
                let duration = currMoveTime - lastMoveTime
                if (duration < Math.floor(1000 / 60)) return; //每秒60帧
                lastMoveTime = currMoveTime

                let currIndex = canvaLineB.getCurrentDataIndex(e)
                if (currIndex > -1 && currIndex < canvaLineB.opts.categories.length) {
                    let riqi = canvaLineB.opts.categories[currIndex]
                    let leibie = canvaLineB.opts.series[0].name
                    let shuju = canvaLineB.opts.series[0].data[currIndex]
                    this.InteractiveB = leibie + ':' + riqi + ' ' + shuju + 'ppm'
                }
                canvaLineB.showToolTip(e, {
                    format: function(item, category) {
                        return category + ' ' + item.name + ':' + item.data + 'ppm'
                    }
                })
            },
            touchEndLineB(e) {
                let currIndex = canvaLineB.getCurrentDataIndex(e)
                if (currIndex > -1 && currIndex < canvaLineB.opts.categories.length) {
                    let riqi = canvaLineB.opts.categories[currIndex]
                    let leibie = canvaLineB.opts.series[0].name
                    let shuju = canvaLineB.opts.series[0].data[currIndex]
                    this.InteractiveB = leibie + ' ' + ':' + riqi + ' ' + shuju + 'ppm'
                }
                canvaLineB.touchLegend(e)
                canvaLineB.showToolTip(e, {
                    format: function(item, category) {
                        return category + ' ' + item.name + ':' + item.data + 'ppm'
                    }
                })
            },
        }
    }
</script>

<style>
    /*样式的width和height一定要与定义的cWidth和cHeight相对应*/
    .qiun-charts {
        width: 750upx;
        height: 500upx;
        background-color: #FFFFFF;
    }
```

```
    .charts {
        width: 750upx;
        height: 500upx;
        background-color: #FFFFFF;
    }

    .btn_view {
        margin: 15px auto;
        width: 250px;
        height: 40px;
        line-height: 40px;
        text-align: center;
        border-radius: 5px;
    }
</style>
```

阈值设置界面：

```
<template>
    <view>
        <view class="uni-padding-wrap uni-common-mt">
            <form @submit="formSubmit" @reset="formReset">
                <view class="uni-form-item uni-column">
                    <view class="title">温度阈值 (单位: ℃)</view>
                    <slider value="60" min="-55" max="125" name="temperatureThreshold" show-
value></slider>
                </view>
                <view class="uni-form-item uni-column">
                    <view class="title">浓度阈值 (单位: ppm)</view>
                    <slider value="300" min="200" max="10000" name="concentrationThreshold"
show-value></slider>
                </view>
                <view class="btn_view">
                    <button  :loading="isLoading"  type="primary"  form-type="submit"> 提 交
</button>
                </view>
                <view class="btn_view">
                    <button type="warn" form-type="reset">重置</button>
                </view>
            </form>
        </view>
    </view>
</template>
<script>
    export default {
```

```
        data() {
            return {
                tips: "",
                isLoading: false
            }
        },
        methods: {
            postServerData(e) {
                let that = this
                this.isLoading = true
                uni.request({
                    // url: 'http://172.20.10.2:8000/api/v1/config/1/',
                    url: 'https://design.xuanxuan.cf/api/v1/config/1/',
                    data: e,
                    method: 'PUT',
                    success: function(res) {
                        console.log("OK")
                        that.isLoading = false
                    },
                    fail: () => {
                        tips = "网络错误，小程序端请检查合法域名"
                        that.isLoading = false
                    }
                })
            },
            formSubmit: function(e) {
                var formData = JSON.stringify(e.detail.value)
                console.log('form 发生了 submit 事件，携带数据为：' + formData)
                this.postServerData(formData)
            },
            formReset: function(e) {
                console.log('清空数据')
            }
        }
    }
</script>

<style>
    .uni-form-item .title {
        padding: 20rpx 0;
    }

    .btn_view {
        margin: 15px auto;
```

```
        width: 250px;
        height: 40px;
        line-height: 40px;
        text-align: center;
        border-radius: 5px;
    }
</style>
```