



课 程 论 文



论文题目： 疫情背景下的智能家居系统

课 程： 嵌入式系统原理与应用

姓 名： 刘浩文

学 号： 517021911065

学院(系)： 电子信息与电气工程学院

专 业： 信息安全

指导教师： 周志洪

疫情背景下的智能家居系统

摘要

在新冠肺炎疫情席卷国内外的背景下，居家隔离作为关键防疫手段，成为了广大群众共同的一段特殊生活经历。在居家隔离和保持社交距离的过程中，如何维持个人良好健康的心态，维护家庭和社会的基本秩序，是一个日益重要的课题。我们在本项目中提出应用嵌入式设备构建智能家居系统，通过提供丰富的功能来满足居家时的各种需求，为居家隔离和抗击疫情作出贡献。

我们基于 STM32F103 开发板实现的疫情背景下的智能家居嵌入式系统，综合了环境监测和生物信息采集等传感器、显示屏、按键、风扇等输入输出及控制设备、有线及无线通讯设备等硬件模块，实现了指纹身份认证、疫情新闻浏览、温湿度检测响应、俄罗斯方块游戏、手机及电脑上位机通信、Huawei Lite OS 多任务系统等任务特性，提供了资讯、测量、控制、交互、娱乐各方面的功能。

关键词：STM32F103 开发板, 智能家居系统, 传感器模块, 显示控制设备, 主-从机通信, Huawei Lite OS 操作系统

ABSTRACT

As the COVID-19 pandemic sweeping into countries throughout the world, home isolation, which servers as a vital strategy against the spread of virus, has become a special common memory of many citizens. However, during home isolation and social distancing to prevent physical illness, it is of increasingly importance to maintain positive mental health and controllable social order. In this project, we propose applying embedded system devices to build a smart home system. By offering abundant features to fulfill various indoor demands, our project demonstrate how electronic devices would contribute to the fight of infectious diseases.

Base on STM32F103 development board, we implemented an embedded smart home system in the context of pandemic. The project integrates environmental monitoring and biological information collection sensors, IO and control equipment such as screen, buttons, and fans, wired and wireless communication equipment, assembling all modules into one terminal device. With a set of functions including fingerprint identity authentication, epidemic news browsing, indoor detection and response, Tetris games, mobile phone and host computer communication, and multitask Huawei Lite Operating System, the smart system provides rich user experience in respect of information, measurement, control, interaction, and entertainment.

KEY WORDS: STM32F103 Development Board, Smart Home System, Sensor Module, Display and Control Device, Master/Slave Communication, Huawei Lite Operating System

目 录

一 项目概述	1
1.1 项目意义	1
1.2 项目目的	2
1.3 项目成果	3
1.4 项目分工	3
二 总体设计	4
2.1 系统构成	4
2.2 功能需求	4
2.3 设计要求	5
三 硬件设计	6
3.1 基于 ATK-AS608 的指纹识别认证	6
3.1.1 ATK-AS608 指纹识别模块	6
3.1.2 模块系统资源	6
3.1.3 模块连接	7
3.2 疫情新闻	8
3.2.1 通过串口与上位机通讯	8
3.2.2 SPI 外设与 FLASH 芯片	9
3.2.3 LCD 显示屏与按键	9
3.3 基于 DS18B20 的室内温控	10
3.3.1 DS18B20 模块	10
3.3.2 通信过程	11
3.3.3 风扇、LED、有源蜂鸣器、按键	11
3.3.4 模块连接	11
3.4 无线物联网	11
3.4.1 DHT11 环境温湿度传感模块	11
3.4.2 ESP8266 无线通信模块	13
3.4.3 CJY5V 直流风扇模块	14
3.4.4 MH-FMD 无源蜂鸣器模块	15
3.4.5 模块连接	15
3.5 俄罗斯方块	16
3.5.1 LCD 液晶显示屏	17
3.5.2 ILI9341 液晶控制器	17

3.5.3	FSMC 外设	18
3.5.4	按键	18
3.5.5	模块连接	19
四 软件设计		20
4.1	基于 ATK-AS608 的指纹识别认证	20
4.1.1	指纹录入与识别流程	20
4.1.2	具体通信过程	20
4.1.3	关键代码	22
4.2	疫情新闻	26
4.2.1	与上位机通讯	26
4.2.2	重定向输入输出函数到串口	28
4.2.3	FatFs 文件系统	28
4.2.4	液晶显示	30
4.3	基于 DS18B20 的室内温控	30
4.3.1	模块概述	30
4.3.2	关键代码	31
4.4	无线物联网	34
4.4.1	DHT11 温湿度读取	34
4.4.2	ESP8266 通信	35
4.4.3	显示界面	36
4.4.4	手机 APP 通信	36
4.4.5	上位机通信	37
4.5	多任务并行操作系统	38
4.5.1	嵌入式操作系统的选择	38
4.5.2	Huawei Lite OS 简介	38
4.5.3	关键代码及操作	40
4.6	俄罗斯方块	42
4.6.1	游戏开始结束界面	42
4.6.2	游戏主界面	43
4.6.3	关键逻辑	45
五 成果展示		48
5.1	指纹识别认证	48
5.1.1	串口模式	48
5.1.2	运行上位机	49
5.1.3	录入指纹	49
5.1.4	目标指纹匹配	50
5.2	疫情新闻	50

5.3 室内温控	51
5.4 无线物联网	51
5.5 俄罗斯方块	52
六 项目总结	53
6.1 项目评价	53
6.1.1 项目特点	53
6.1.2 改进方向	53
6.2 心得体会	54
致 谢	56

一 项目概述

1.1 项目意义

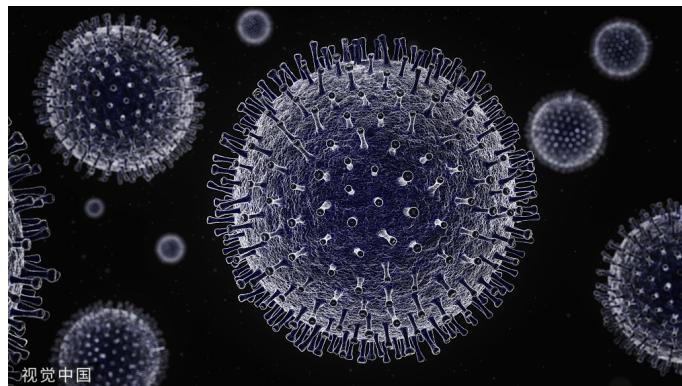


图 1-1 COVID-19

自 2019 年 12 月以来，一场由新型冠状病毒 **COVID-19** 导致的肺炎疫情蔓延全国，我们的祖国与人民经历了一场前所未有的疫情危机，对国家的经济，人民的生命财产安全造成了巨大的影响。目前，疫情仍在世界范围内肆虐。

由于 COVID-19 可怕的传染率与致死率，全国人民或多或少都进行了居家隔离的抗疫实践，这深刻地改变了人们的生活状态。如今，我国的疫情已经趋近尾声，在胜利的欢呼声中，我们也要反思：面对传染病疫情，我们应该做好什么准备？如果再次陷入疫情危机中，身为普通人的我们如何更好地应对？再者，我们如何在全球范围内有效地抗击疫情？本项目就是在这样的思考中诞生的。



图 1-2 居家隔离

国内外抗击疫情的科学的研究和实践经验告诉我们，在没有特效药与疫苗的情况下，防控

重大疫情蔓延的最有效做法就是**居家隔离**。很多专家指出，在居家隔离的过程中，不止需要关注人们的身体健康，如何维护良好的心理健康也是相当重要的课题。对于居家的人们来说，一是对疫情蔓延的惶惶不可终日，二是长时间处于封闭环境的孤独闭塞与枯燥无味；对于社会与经济来说，一是经济活动几乎停滞，二是增加了社会不稳定因素。因此，为了更好地应对疫情，我们决定聚焦于这样一个问题：如何结合嵌入式设备的所学知识，减小疫情时的居家隔离对个人及社会的负面影响？我们认为有效的解决方案有以下几点：

1. 及时甄选出疫情消息，让人们充分了解抗疫进展，缓和民众对疫情紧张恐惧的心理。
2. 提供健康自测方法，让人们居家时能简单方便地得知自己的大致健康状况。
3. 提供更舒适、方便的居家环境，缓解人们居家时的焦躁，使人们爱上居家。
4. 提供一些娱乐手段，缓解人们居家时的闭塞无聊。



图 1-3 物联网家居

经过我们小组寻找与讨论，我们确定了这样一种形式，能够囊括以上的所有要求：**物联网智能家居系统**。通过设计并实现**疫情背景下的物联网智能家居嵌入式系统**，我们希望为疫情时期居家隔离中的人们提供一个方便、舒适、健康、智能、趣味的居家环境，**使人们享受居家、爱上居家，使社会稳定、家庭幸福**。

1.2 项目目的

在设计实现我们的物联网智能家居嵌入式系统过程中，我们集中于解决实际问题，实现以下几个方面的目的：

1. 注重培养综合运用所学知识、独立分析和解决实际问题的能力，培养创新意识和创新能力，并获得科学基础训练。
2. 了解嵌入式系统的概念和理论，了解 STM32 开发板的各部分原理和实现，巩固并应用课程中的嵌入式相关内容知识。
3. 了解各个硬件模块的原理和应用，通过实践完成筛选采购、查阅文档、编程调试、硬件测试等全过程，掌握各种嵌入式设备硬件的使用方式，并实现设计要求的各项功能。

4. 了解系统项目的开发流程，通过分功能、分层次的设计开发，使各个硬件模块和软件功能之间有机联系，成为整个嵌入式系统中有意义的功能模块。

1.3 项目成果

我们完成的**疫情背景下的物联网智能家居嵌入式系统**，围绕 STM32F103 野火指南者开发板作为核心嵌入式设备，充分利用板载处理器和其他元件，成功实现了十余种硬件模块的调试和应用，通过各模块之间的交互合作，完成了资讯、测量、控制、交互、娱乐等各项设计功能。项目各模块及功能均实际测试通过，具有较强的可用性和可拓展性。

项目实现了各项设计目标要求，满足了项目开发的目的，在疫情背景的居家隔离生活中具有一定的实用价值。项目还完成了展示视频和项目报告。

1.4 项目分工

表 1-1 项目具体分工

小组成员	负责内容
刘浩文 517021911065	部分模块的选择与采购，ATK-AS608 指纹识别模块的资料收集，项目报告中项目概述、总体设计、项目总结部分的撰写。
江浩宇 517021910754	部分模块的选择与采购，DHT11 温湿度传感器模块和 ESO8266 无线模块的资料收集与调试，参与无线物联网功能的设计与调试，移植 Huawei lite OS 实现多任务并行，项目功能整合进操作系统（仅部分），项目无线物联网与 Huawei lite OS 部分软硬件的撰写。
郭子豪 517021910867	室内温控功能的软硬件设计与实现、疫情新闻功能的软硬件设计与实现、无线物联网功能（除 DHT11 温湿度读取）的软硬件设计与实现、俄罗斯方块功能的软硬件设计与实现。采购部分模块（MLX90614/ds18b20，无源蜂鸣器、风扇、面包板、按键等）。报告中负责了俄罗斯方块和疫情新闻的软硬件部分，同时参与了对其它部分的调整和修改。
郭建铭 517021910722	智能指纹锁模块的软硬件设计与实现，实现指纹的录入、匹配、搜索、删除等功能。完成项目的 VLOG 制作以及 BOM 清单整合。采购部分模块（ATK-AS608，面包板，按键等）。ATK-AS608 模块的资料收集与调试。报告中完成指纹识别认证模块内容的撰写。
廖宁 F 517021910844	部分模块的选择与采购，MLX90614 模块的资料收集，DY-SV8F 模块的资料收集与调试（未成功），项目摘要概述、DS18B20 模块及室内温控功能的撰写，报告的整体规划与统稿。

二 总体设计

2.1 系统构成

我们的疫情背景下的物联网智能家居嵌入式系统包含智能指纹锁、疫情新闻、室内智能温控、无线物联网（包括智能风扇电灯、音乐播放、室内温湿度读取、无线通信）、游戏娱乐、操作系统等功能模块。当然，作为一个系统，用户也被包含在内。各模块与元素间的组成与交互如图2-1所示。

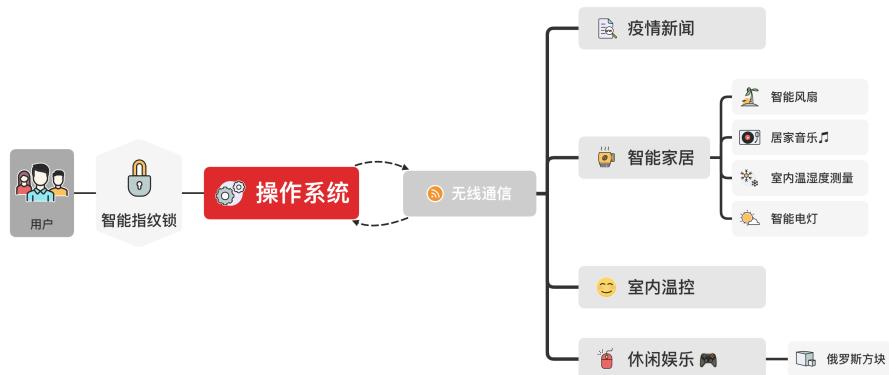


图 2-1 系统构成与交互

2.2 功能需求

具体到各个模块，其作为系统的组成部分，与其他模块交互，完成系统的部分设计要求：

1. **智能指纹锁模块**能够保存或验证使用者的指纹，作为家居安全便利而智能的保障者。
2. **疫情新闻模块**能够帮助人们了解最新疫情信息，完成资讯功能。
3. **室内智能温控**需要感知室内温度并进行智能控制。
4. **无线物联网**智能家居的需求，可通过手机 app 直接控制风扇、电灯、音乐播放、并且读取室内温湿度。
5. **俄罗斯方块游戏**疫情期间使用户进行一些小娱乐，以打发居家隔离的枯燥无聊。
6. **操作系统模块**能够通过通信模块接收各功能模块的数据，并进行处理；且能够向各功能下达指令，控制各模块的工作。

由于资金与时间所限，我们的各模块能完成的功能也有限。但我们相信，如果有更多资源对各模块进行更深入的分析与扩展，我们的**疫情背景下的物联网智能家居嵌入式系统**一定能成为一个完善、实用的系统，为抗击疫情做出贡献。

2.3 设计要求

1. **智能指纹锁模块：**实现指纹的录入、保存功能，实现指纹的识别、匹配和搜索，能够输出匹配成功与否的结果并通过指纹匹配当前用户。
2. **疫情新闻模块：**实现上位机疫情新闻的获取刷新，实现新闻在上位机与嵌入式设备之间的通信，实现嵌入式设备上新闻的存储和展示。
3. **室内智能温控**获取室内温度并显示在 led 屏幕，有两个阈值分别用于控制风扇自动开启降温和高温报警，可通过按键调节阈值。
4. **无线物联网**通过手机 app 利用 esp8266 与 stm32 通信，直接开关风扇、电灯、音乐播放、并且查看室内温湿度。各种设备的状态除了在手机 app 上查看，led 屏幕上也会相应显示。
5. **俄罗斯方块游戏**实现俄罗斯方块游戏的完整功能，包括界面显示、游戏操作、控制处理等。
6. **操作系统模块：**实现嵌入式设备的操作系统移植，实现操作系统的多任务功能。

三 硬件设计

3.1 基于 ATK-AS608 的指纹识别认证

3.1.1 ATK-AS608 指纹识别模块

ATK-AS608 指纹识别模块(以下简称 AS608 模块)是 ALIENTEK 推出的一款高性能的推出的一款高性能的光学指纹识别模块。AS608 模块采用了国内著名指纹识别芯片公司杭州晶元芯片技术有限公司的 AS608 指纹识别芯片。芯片内置 DSP 运算单元,集成了指纹识别算法,能高效快速采集图像并识别指纹特征。模块配备了串口、USB 通讯接口,用户无需研究复杂的图像处理及指纹识别算法,只需通过简单的串口、USB 按照通讯协议便可控制模块。本模块可应用于各种考勤机、保险箱柜、指纹门禁系统、指纹锁等场合。

模块实物图如3-1所示,结构尺寸图如3-2所示。AS608 各项技术指标列表如表3-1所示。

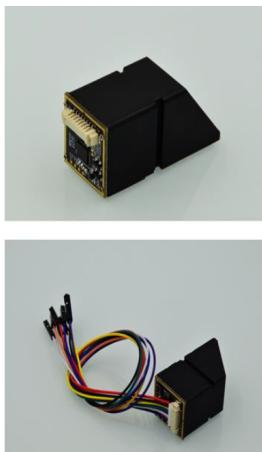


图 3-1 AS608 模块实物图

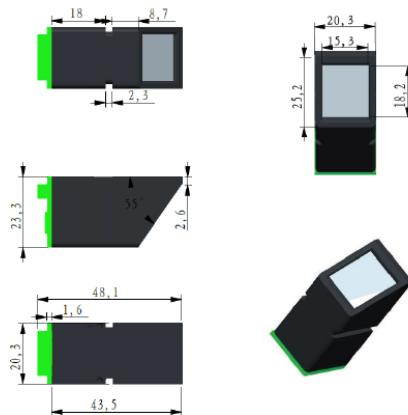


图 3-2 AS608 模块尺寸图

3.1.2 模块系统资源

- 缓冲区与指纹库:** 系统内设有一个 72K 字节的图像缓冲区与两个 512bytes 大小的特征文件缓冲区,名字分别称为:ImageBuffer,CharBuffer1 和 CharBuffer2。用户可以通过指令读写任意一个缓冲区。CharBuffer1 或 CharBuffer2 既可以用于存放普通特征文件也可以用于存放模板特征文件。通过 UART 口上传或下载图像时为了加快速度,只用到像素字节的高 4 位,即将两个像素合成一个字节传送。通过 USB 口则是整 8 位像素。
- 用户记事本:** 系统在 FLASH 中开辟了一个 512 字节的存储区域作为用户记事本,该记事本逻辑上被分成 16 页,每页 32 字节,上位机可以通过 PSWriteNotepad 指令和 PSReadNotepad 指令访问任意一页。写记事本某一页的时候,该页 32 字节的内容被整体写入,原来的内容被覆盖。

表 3-1 AS608 技术指标清单

项目	说明
工作电压 (V)	3.0-3.6V, 典型值:3.3V
工作电流 (mA)	30-60mA, 典型值:40mA
USART 通讯	波特率:9600CEN(N=1-12, 默认为 6),bps=57600 数据位:8 停止位:1 校验位:none TTL 电平
USB 通讯	2.0FS
传感器图像大小 (pixel)	256 x 288pixel
图像处理时间 (s)	<0.4(s)
上电延时 (s)	<0.1(s), 模块上电后初始化工作时间
搜索时间	<0.3(s)
拒真率 (FRR)	<1%
认假率 (FAR)	<0.001%
指纹存容量	300 枚 (ID: 299)
工作环境	温度 (°C):-20~60 湿度 < 无凝露)

3. 随机数生成器：系统内部集成了硬件 32 位随机数生成器，用户可以通过指令让模块产生一个随机数并上传给上位机。
4. 特征与模板：指纹特征文件大小为 256 字节，包含特征点信息与总体信息；模板大小为 512 字节，是两个相同指纹特征之和。
5. ROM 及传感器驱动：ROM 内嵌了完整的指纹识别算法，传感器驱动为 synochip 提供，用户可自行开发相关应用层程序。

3.1.3 模块连接

AS608 模块内部内置了手指探测电路，用户可读取状态引脚（判断有无手指按下。接口采用 8 芯 1.25 mm 间距单排插座，PCB 如3-3所示。连接实物图如图3-4 所示。

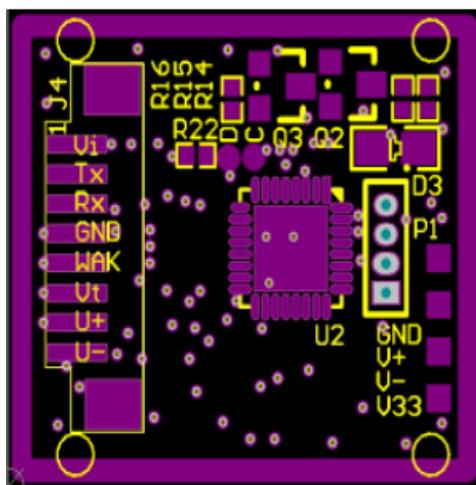


图 3-3 AS608 模块 PCB



图 3-4 指纹模块实物连接图

在本项目中，受限于 stm32f103 的资源，我们采用 USART 串口和上位机通信来实现本模块的应用。模块引脚及与 stm32f103 的连接列表如下：

表 3-2 AS608 引脚列表

序号	名称	开发板管脚	说明
1	Vi	3.3V	模块电源正输入端，接 3.3V 电源
2	Tx	Rx	串行数据输入 TTL 逻辑电平
3	Rx	Tx	串行数据输入 TTL 逻辑电平
4	GND	GND	信号地，内部与电源地连接
5	WAK	PA6	感应信号输出，默认高电平有效
6	Vt	3.3V	触摸感应电源输入端,3.3v 供电
7	U+	-	USB D+
8	U-	-	USB D-

3.2 疫情新闻

该模块使用串口 usart1 与上位机进行周期性通讯，将从上位机获得的疫情新闻发送给 stm32，同时利用文件系统将疫情新闻存储在 flash 中（若 stm32 已有新闻则相应更新新闻），并使用 LCD 显示屏 + 按键查看新闻。

3.2.1 通过串口与上位机通讯

为利用 USART 实现开发板与电脑通信，需要用到一个 USB 转 USART 的 IC，我们选择 CH340G 芯片来实现这个功能，CH340G 是一个 USB 总线的转接芯片，实现 USB 转 USART、USB 转 IrDA 红外或者 USB 转打印机接口，我们使用其 USB 转 USART 功能。具体电路设计见下图。我们将 CH340G 的 TXD 引脚与 USART1 的 RX 引脚连接，CH340G 的 RXD 引脚与 USART1 的 TX 引脚连接。CH340G 芯片集成在开发板上，其地线 (GND) 已与控制器的 GND 连通。

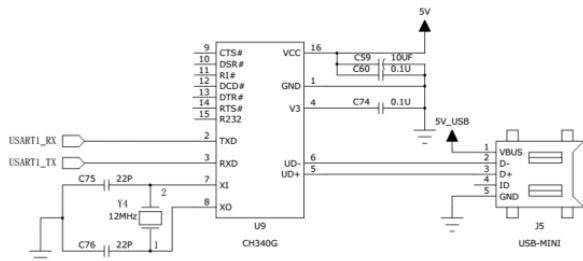


图 3-5 USB 转串口硬件设计

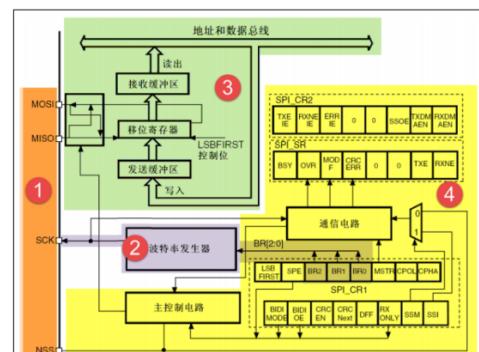


图 3-6 SPI 架构图

3.2.2 SPI 外设与 FLASH 芯片

SPI 的所有硬件架构都从上图图中左侧 MOSI、MISO、SCK 及 NSS 线展开的。STM32 芯片有多个 SPI 外设，它们的 SPI 通讯信号引出到不同的 GPIO 引脚上，使用时必须配置到这些指定的引脚。图 3-7 为这些引脚的详细说明。

引脚	SPI 编号		
SPI1	SPI2		
NSS	PA4	PB12	PA15 下载口的 TDI
CLK	PA5	PB13	PB3 下载口的 TDO
MISO	PA6	PB14	PB4 下载口的 NTRST
MOSI	PA7	PB15	PB5

图 3-7 STM32F10x 的 SPI 引脚

本部分需要使用 SPI 通讯的串行 FLASH 存储芯片的读写。其中 STM32 的 SPI 外设采用主模式，通过查询事件的方式来确保正常通讯。

本实验板中的 FLASH 芯片（型号：W25Q64）是一种使用 SPI 通讯协议的 NORFLASH 存储器，它的 CS/CLK/DIO/DO 引脚分别连接到了 STM32 对应的 SPI 引脚 NSS/SCK/MOSI/MISO 上，其中 STM32 的 NSS 引脚是一个普通的 GPIO，不是 SPI 的专用 NSS 引脚，所以程序中我们要使用软件控制的方式。

FLASH 芯片中还有 WP 和 HOLD 引脚。WP 引脚可控制写保护功能，当该引脚为低电平时，禁止写入数据。直接接电源，不使用写保护功能。HOLD 引脚可用于暂停通讯，该引脚为低电平时，通讯暂停，数据输出引脚输出高阻抗状态，时钟和数据输入引脚无效。本部分直接接电源，不使用通讯暂停功能。

spi 的硬件连接图如下图所示。

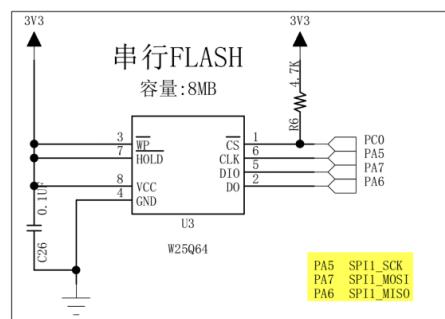


图 3-8 SPI 串行 FLASH 硬件连接图

3.2.3 LCD 显示屏与按键

液晶显示需要使用的硬件设备，在前面已经介绍过了。本部分的按键只使用了野火内置的 K1,K2,K1 用于切换新闻条目，K2 用于查看新闻内容和返回条目界面。

3.3 基于 DS18B20 的室内温控

3.3.1 DS18B20 模块

我们原计划使用 MLX90614 或类似模块进行近距离的人体测温，但受疫情影响，成套的测温模块在采购阶段发现已经脱销，所购入的单个该芯片因质量问题难以在 STM32 上使用。故此处我们使用了精度与准确性较前者稍弱 DS18B20 数字测温模块，并且重新搭建了一个室内温控模块。DS18B20 测温范围更大、应用普遍、成本较低，故也可以应用在通用的测温场景下。

DS18B20 是常用的数字温度传感器，其输出的是数字信号，具有体积小、硬件开销低、抗干扰能力强、精度高的特点。DS18B20 数字温度传感器接线方便，封装成后可应用于多种场合。

DS18B20 采用数字测温方式，主要器件为二不同温度系数的晶振。低温度系数晶振的振荡频率受温度影响很小，用于产生固定频率的脉冲信号，使温度寄存器的值随时间增加；高温度系数晶振随温度变化其振荡率明显改变，其振荡指定个数个周期后，使温度寄存器的值被读入，即为此时的温度测量值。DS18B20 单线通信功能是分时完成的，有严格的时隙概念，因此读写时序很重要，系统对 DS18B20 的各种操作必须按协议进行，操作协议为：初始化 DS18B20 → 发 ROM 功能命令 → 发存储器操作命令 → 处理数据。

我们使用的模块实物图如图 3-9 所示。其采用单总线半双工通信方式，仅需一个数据引脚，引脚对应关系如表 3-3 所示。

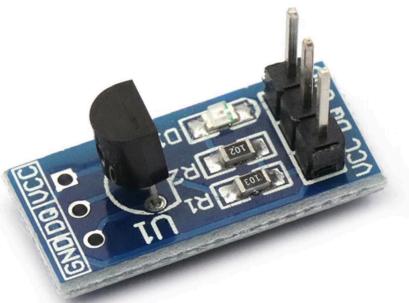


图 3-9 DS18B20 模块实物图

表 3-3 DS18B20 引脚对应关系

序号	名称	开发板管脚	说明
1	VDD	3V3	供电 3 - 5.5VDC
2	DQ	PA0	数据端口
4	GND	GND	接地，电源负极

3.3.2 通信过程

DS18B20 底层通信时序分为复位/应答时序、写字节时序、读字节时序三种，通过复位脉冲、应答脉冲、写 0、写 1、读 0 和读 1 六种信号进行实现，均按协议实现即可，详见下一章。具体到获取当前温度，我们还需要通过写时序写入跳过 ROM、温度转换、读 RAM 三种指令字节，以使其返回当前温度值。

3.3.3 风扇、LED、有源蜂鸣器、按键

风扇模块的硬件介绍会在无线物联网的硬件部分给出，此处省略。报警所用的 LED 灯和有源蜂鸣器，以及调节阈值的按键均使用板子内置的外设。

3.3.4 模块连接

室内温控模块使用的 GPIO 口如下表所示。

表 3-4 室内温控模块使用的 GPIO 管脚

模块名称	ds18b20	按键 1	按键 2	红灯	绿灯	蜂鸣器
GPIO 口	PA11	PA0	PC13	PB5	PB0	PA8

3.4 无线物联网

无线物联网部分需要使用到的硬件模块有：DHT11 环境温湿度传感模块、ESP8266 无线通信模块、CJY5V 直流风扇模块和 MH-FMD 无源蜂鸣器模块。其中运用 DHT11 采集温湿度数据，再通过 wifi 模块传递 wifi 信息到物联网终端，并通过物联网终端控制电风扇、无源蜂鸣器和 LED 灯的开关。

3.4.1 DHT11 环境温湿度传感模块

3.4.1.1 硬件概述

STM32 虽然内部自带了温度传感器，但是因为芯片温升较大等问题，与实际温度差别较大，难以作为环境温度的测量。DHT11 数字温湿度传感器是一款含有已校准数字信号输出的温湿度复合传感器。它应用专用的数字模块采集技术和温湿度传感技术，确保产品具有极高的可靠性与卓越的长期稳定性。传感器包括一个电阻式感湿元件和一个 NTC 测温元件，并与一个高性能 8 位单片机相连接。因此该产品具有品质卓越、超快响应、抗干扰能力强、性价比高等优点。每个 DHT11 传感器都在极为精确的湿度校验室中进行校准。校准系数以程序的形式储存在 OTP 内存中，传感器内部在检测信号的处理过程中要调用这些校准系数。单线制串行接口，使系统集成变得简易快捷。超小的体积、极低的功耗，信号传输距离可达 20 米以上，使其成为各类应用甚至最为苛刻的应用场合的最佳选则。产品为 4 针单排引脚封装。连接方便，特殊封装形式可根据用户需求而提供。模块的电路图、实物图各如图3-10和3-11所示。

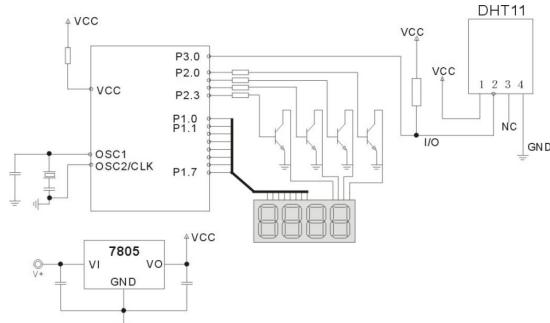


图 3-10 DHT11 模块电路图

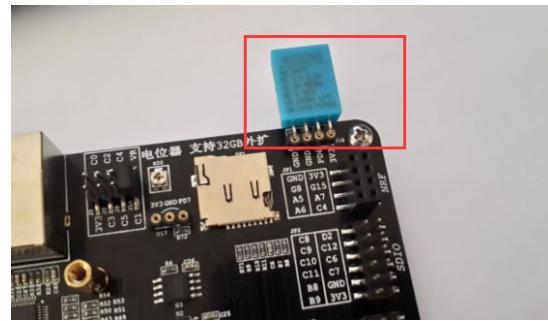


图 3-11 DHT11 模块实物图

DHT11 模块的引脚说明如下表：

表 3-5 AS608 引脚对应关系

序号	名称	开发板管脚	说明
1	VDD	3V3	供电 3 - 5.5VDC
2	DATA	PE6	串行数据，单总线
3	NC	-	空脚，请悬空
4	GND	GND	接地，电源负极

3.4.1.2 通信过程

本项目开发板 MCU 发送一次开始信号后,DHT11 从低功耗模式转换到高速模式, 等待主机开始信号结束后,DHT11 发送响应信号, 送出 40bit 的数据, 并触发一次信号采集, 用户可选择读取部分数据. 从模式下,DHT11 接收到开始信号触发一次温湿度采集, 如果没有接收到主机发送开始信号,DHT11 不会主动进行温湿度采集. 采集数据后转换到低速模式。读取 DHT11 的数据可分为以下步骤:

1. DHT11 上电后 (DHT11 上电后要等待 1s 以越过不稳定状态在此期间不能发送任何指令), 测试环境温湿度数据, 并记录数据, 同时 DHT11 的 DATA 数据线由上拉电阻拉高一直保持高电平;
2. 微处理器的 I/O 设置为输出同时输出低电平, 且低电平保持时间不能小于 18ms (最大不得超过 30ms), 然后微处理器的 I/O 设置为输入状态, 由于上拉电阻, 微处理器的 I/O 即 DHT11 的 DATA 数据线也随之变高, 等待 DHT11 作出回答信号;
3. DHT11 的 DATA 引脚检测到外部信号有低电平时, 等待外部信号低电平结束, 延迟后 DHT11 的 DATA 引脚处于输出状态, 输出 83 微秒的低电平作为应答信号, 紧接着输出 87 微秒的高电平通知外设准备接收数据, 微处理器的 I/O 此时处于输入状态, 检测到 I/O 有低电平 (DHT11 回应信号) 后, 等待 87 微秒的高电平后的数据接收此时 DHT11 的 DATA 引脚处于输入状态, 时刻检测外部信号。
4. 由 DHT11 的 DATA 引脚输出 40 位数据, 微处理器根据 I/O 电平的变化接收 40 位数据, 位数据 “0”的格式为：54 微秒的低电平和 23-27 微秒的高电平, 位数据 “1”的

格式为：54 微秒的低电平加 68-74 微秒的高电平；

5. DHT11 的 DATA 引脚输出 40 位数据后，继续输出低电平 54 微秒后转为输入状态，由于上拉电阻随之变为高电平。但 DHT11 内部重测环境温湿度数据，并记录数据，等待外部信号的到来。

本项目通过该模块收集温湿度信息，以便后边工作上传到手机 app 或直接显示在整合后的 Huawei lite OS 上。

3.4.2 ESP8266 无线通信模块

3.4.2.1 硬件概述

ESP8266 拥有高性能无线 SOC，给移动平台设计师带来福音，它以最低成本提供最大实用性，为 WiFi 功能嵌入其他系统提供无限可能。ESP8266 是一个完整且自成体系的 WiFi 网络解决方案，能够独立运行，也可以作为 slave 搭载于其他 Host 运行。ESP8266 在搭载应用并作为设备中唯一的应用处理器时，能够直接从外接闪存中启动。内置的高速缓冲存储器有利于提高系统性能，并减少内存需求。ESP8266 高度片内集成，包括天线开关 balun、电源管理转换器，因此仅需极少的外部电路，且包括前端模块在内的整个解决方案在设计时将所占 PCB 空间降到最低。

模块资源描述图如3-12所示：

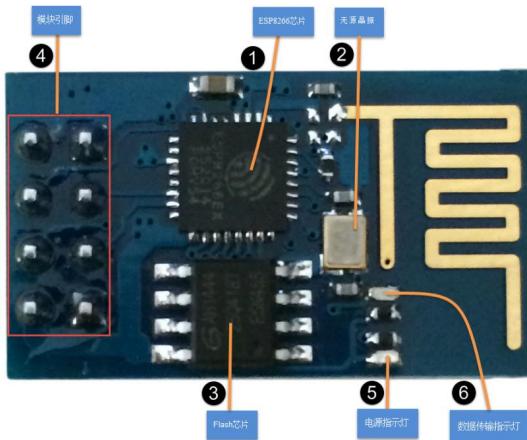


图 3-12 ESP9266 模块资源描述图

3.4.2.2 模块接口

模块引脚说明如下：本项目利用野火指南者 F103VET6 板载的 ESP8266 WIFI 模块，实现两方面的数据传输。一方面利用用 ESP8266 的 AP 模式，将 DHT11 温湿度数据上传到手机 APP，实现简易物联网终端；另一方面利用 ESP8266 的 Station 模式，通过串口转网络调试助手将温湿度数据传向上位机。如果将模块配置成 AP 模式，向外发出 WIFI 信号，ESP8266 作为 softAP，手机、电脑、用户设备、其他 ESP8266 station 接口等均可以作为 station 连入 ESP8266，组建成一个局域网，如果配置成 Station 模式，ESP8266 模块通过局域网不断地将

表 3–6 AS608 引脚对应关系

序号	名称	方向	说明
1	GND	GND	电源地
2	UTXD	O	USART Tx
3	GPIO2	I/O	GPIO2
4	CH_PD	I	模块使能端，高电平有效
5	GPIO0	I/O	GPIO0
6	RST	I	重启，低电平有效
7	URXD	I	USART Rx
8	VCC	I	模拟电源 3.0 3.3V

实时更新的温湿度数据发送给上位机，一次发送 1120 个字节，时间间隔为 100ms。

3.4.3 CJY5V 直流风扇模块

3.4.3.1 硬件概述

本项目为了多样化物联网 APP 控制开发板的功能，将开发板打造成物联网智能家居控制终端的雏形，尝试模拟日常的家居生活，为指南者开发板增添一个直流小风扇的模块，来模拟嵌入式开发板对家居家电的控制。由于板子能够提供支持的电流较小，我们在设计一个三极管放大电路的基础上，尽量选取一个功率最小的直流双滚珠风扇，来保证开发板提供的电流能正常启动风扇。

因此，我们采用 SNOWFAN 厂家生产的型号为 c jyYY5010H05S 直流风扇模块，风扇模块的描述图如3–13所示：



图 3–13 直流风扇模块描述图

直流风扇模块的参数如下表：

表 3-7 直流风扇模块参数

工作电压	工作电流	转速	噪音	接口类型
DC 5V	0.2A	4500±10%RPM	20DBA	2510-2P

3.4.3.2 三极管放大电路

由于开发板能直接提供的电流大小不足 0.2A，不能直接带动上述直流风扇正常运行，所以我们设计并焊接了一个三极管放大电路，来放大电流，达到直流风扇的工作电流。该电路是由一个型号为 S8550 的三极管、一个 1K 的电阻和一个 10k 的电阻构成，电路原理图、实物图各如3-14和3-15所示：

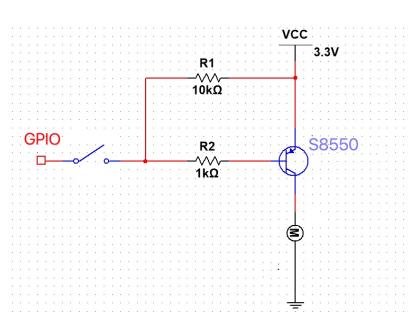


图 3-14 三极管放大电路原理图

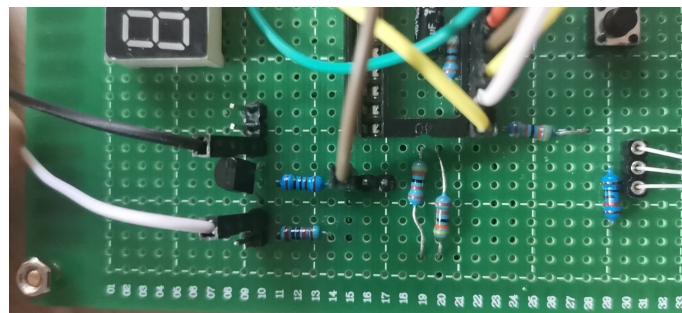


图 3-15 三极管放大电路实物图

3.4.4 MH-FMD 无源蜂鸣器模块

3.4.4.1 硬件概述

为了丰富物联网 app 的功能，我们引入无源蜂鸣器模块，我们在物联网手机 app 上便可控制无源蜂鸣器播放音乐。无源蜂鸣器相对于野火指南者板载的有源蜂鸣器，具有以下特点：

首先，无源蜂鸣器内部不带震荡源，直接用直流信号是无法让其鸣叫，必须使用 2K 5K 的方波去驱动它；正因为如此，我们可以控制声音的频率，实现播放音乐的功能；再者，在一些特殊用法中，无源蜂鸣器可以和 LED 复用一个控制口。

我们选取的这款无源蜂鸣器型号为 MH-FMD，采用三线制的接口，使用 S8550 三极管驱动，并且设有固定螺栓孔，方便安装。无源蜂鸣器模块的实物图如3-16所示：

3.4.4.2 模块接口

无源蜂鸣器模块接口如下表：

3.4.5 模块连接

下图是指南者开发板上的 DHT11 温湿度传感器和 ESP8266 模块原理图，它说明了 DHT11 传感器和配套的 ESP8266 接入到开发板上时的信号连接关系。其中 DHT11 的数据接口与开



无源蜂鸣器模块
低电平触发

图 3-16 无源蜂鸣器实物图

表 3-8 无源蜂鸣器接口关系

编号	名称	说明
1	VCC	外接 3.3V-5V 电压
2	GND	外接 GND
3	I/O	外接单片机 IO 口

发板 GPIO 管脚 PE6 相连，ESP8266 的 URXD 与 PB10、UTXD 与 PB11、CH-PD 与 PB8、RST 与 PB9 依次相连。

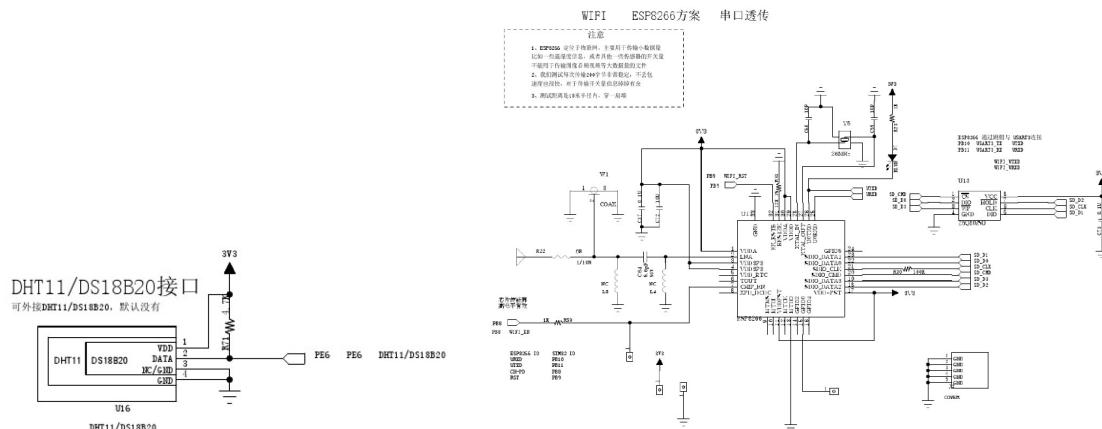


图 3-17 DHT11 原理图

图 3-18 ESP8266 原理图

3.5 俄罗斯方块

该游戏模块使用到的硬件有：LCD 液晶显示屏、ILI9341 液晶控制器、FSMC 外设，他们用于实现游戏的显示。此外除了利用板上自带的两个按键外，还使用了四个单独焊接的按键用于操作游戏（焊接的按键需要通过附加的上拉电阻来保证其稳定性）。

表 3-9 各个模块使用的 GPIO 管脚

模块名称	无源蜂鸣器	直流风扇	DHT11	ESP8266
GPIO 口	PA2	PC3	PE6	PB10、PB11、PB8、PB9
模块名称	LED1	LED2	LED3	有源蜂鸣器
GPIO 口	PB5	PB0	PB1	PA8

3.5.1 LCD 液晶显示屏

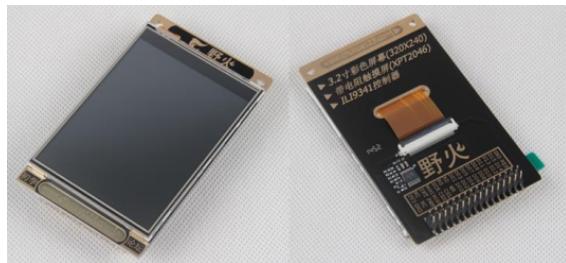


图 3-19 液晶屏实物图

LCD 液晶显示屏分为液晶触摸面板和 PCB 底板两部分。液晶触摸面板由液晶屏和触摸屏组成，触摸屏的下方即为液晶面板，在它的内部包含了一个型号为 ILI9341 的液晶控制器芯片，该液晶控制器使用 8080 接口与单片机通讯。

液晶面板引出的 FPC 信号线即 8080 接口 (RGB 接口已在内部直接与 ILI9341 相连)，且控制器中包含有显存，单片机把要显示的数据通过引出的 8080 接口发送到液晶控制器，这些数据会被存储到它内部的显存中，然后液晶控制器不断把显存的内容刷新到液晶面板，显示内容。

上图表示的是 PCB 底板引出的排针线序，屏幕整体通过这些引出的排针与开发板或其它控制器连接。

3.5.2 ILI9341 液晶控制器

本液晶屏内部包含有一个液晶控制芯片 ILI9341。该芯片最核心部分是位于中间的 GRAM(GraphicsRAM)，它就是显存。GRAM 中每个存储单元都对应着液晶面板的一个像素点。它右侧的各种模块共同作用把 GRAM 存储单元的数据转化成液晶面板的控制信号，使像素点呈现特定的颜色，而像素点组合起来则成为一幅完整的图像。

框图的左上角为 ILI9341 的主要控制信号线和配置引脚，根据其不同状态设置可以使芯片工作在不同的模式，如每个像素点的位数是 6、16 还是 18 位；可配置使用 SPI 接口、8080 接口还是 RGB 接口与 MCU 进行通讯。MCU 通过 SPI、8080 接口或 RGB 接口与 ILI9341 进行通讯，从而访问它的控制寄存器 (CR)、地址计数器 (AC)、及 GRAM。

在 GRAM 的左侧还有一个 LED 控制器 (LEDController)。LCD 为非发光性的显示装置，

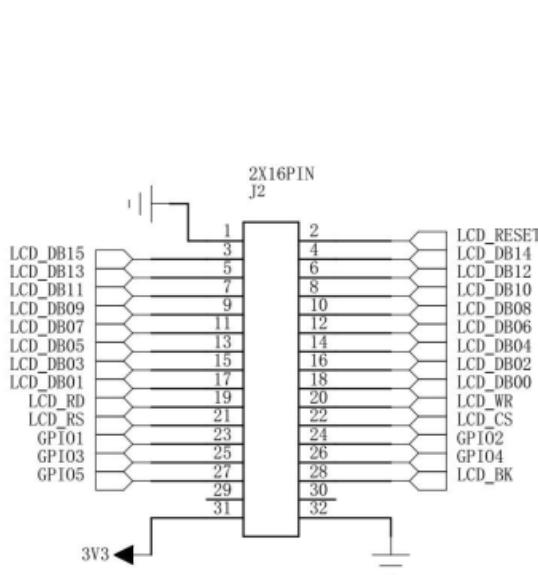


图 3-20 液晶屏接口

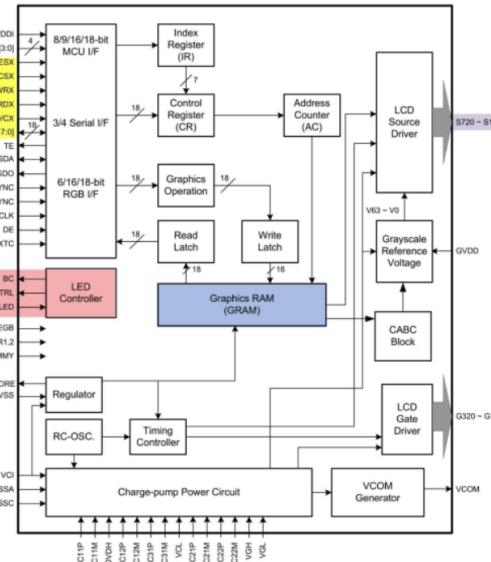


图 3-21 ILI9341 控制器内部框图

它需要借助背光源才能达到显示功能，LED 控制器就是用来控制液晶屏中的 LED 背光源。

3.5.3 FSMC 外设

FSMC 是 Flexible Static Memory Controller 的缩写，译为灵活的静态存储控制器。它可以用于驱动包括 SRAM、NOR FLASH 以及 NAND FLSAH 类型的存储器，由于 FSMC 外设可以用于控制扩展的外部存储器，而 MCU 对液晶屏的操作实际上就是把显示数据写入到显存中，与控制存储器非常类似，且 8080 接口的通讯时序完全可以使用 FSMC 外设产生，因而非常适合使用 FSMC 控制液晶屏。

3.5.4 按键

内置按键的原理图和焊接按键的原理图如下所示，内置按键包括 K1 和 K2，焊接按键一共 4 个，VCC 使用 3.3V，上拉电阻大小为 10k。最右为焊接的实物图。

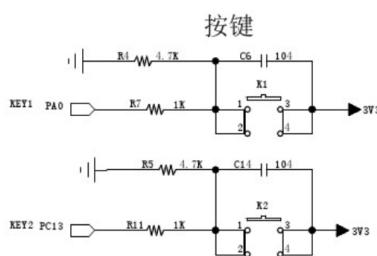


图 3-23 内置按键

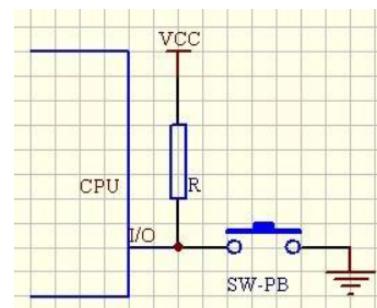


图 3-24 焊接按键

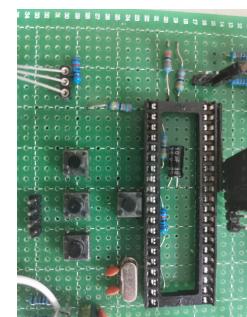


图 3-25 实物图

3.5.5 模块连接

下图是指南者开发板上的液晶排母接口原理图，它说明了配套的 3.2 寸屏幕接入到开发板上时的信号连接关系。其中液晶屏 LCD-CS 及 LCD-RS(即 DC 引脚)与 FSMC 存储区选择引脚 FSMC-NE 及地址信号 FSMC-A 的编号，会决定 STM32 要使用什么内存地址来控制与液晶屏的通讯。

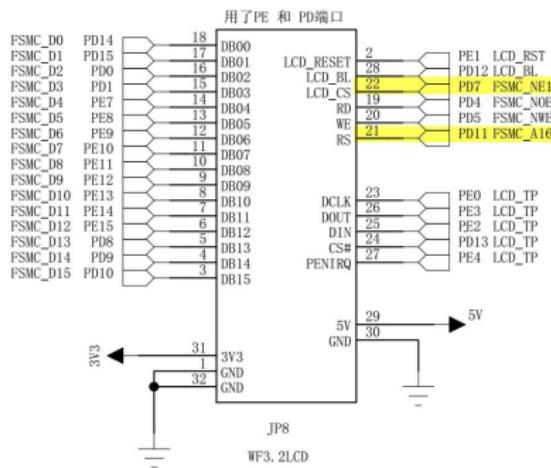


图 3-25 液晶屏与开发板连接图

下表为各个按键所使用的 GPIO 口。

表 3-10 按键使用的 GPIO 管脚

模块名称	左移按键	右移按键	下降按键	变形按键	难度切换	游戏开始
GPIO 口	PA0	PC13	PA3	PC2	PC4	PC5

四 软件设计

4.1 基于 ATK-AS608 的指纹识别认证

4.1.1 指纹录入与识别流程

受限于模块的独特性和 stm32 的有限资源，本部分借助 USB 转串口设备和上位机程序实现相应功能。相应功能实现过程以流程图的形式展示，如4-1所示

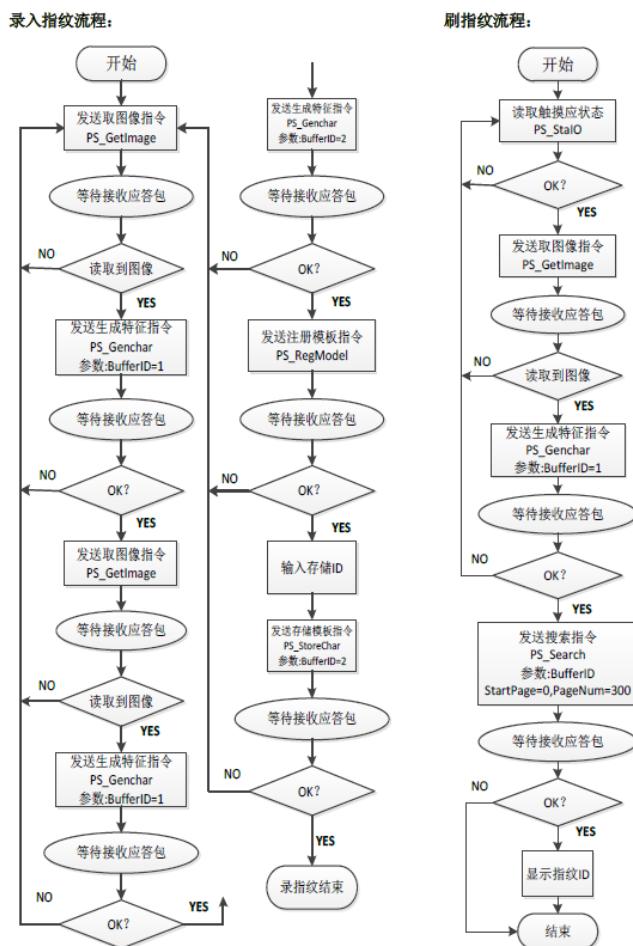


图 4-1 指纹录入与识别流程图

ATK-AS608 模块共有指令 31 条，列举本项目用到的指令集及功能列表所示。

4.1.2 具体通信过程

具体几项功能运行过程解读如下：

表 4-1 AS608 指令集

指令码	函数名	功能描述
01H	PS_GetImage	从传感器上读入图像存于图像缓冲区
02H	PS_GetChar	根据原始图像生成指纹特征存于 CharBuffer1 或 CharBuffer2
03H	PS_Match	精确比对 CharBuffer1 与 CharBuffer2 中的特征文件
04H	PS_Search	以 CharBuffer1 或 CharBuffer2 中的特征文件搜索整个或部分指纹库
05H	PS_RegModel	将 CharBuffer1 与 CharBuffer2 中的特征文件 合并生成模板存于 CharBuffer1 和 CharBuffer2
06H	PS_StoreChar	将特征缓冲区的文件储存到 flash 指纹库中
0CH	PS_DeleteChar	删除 flash 指纹库的一个特征文件
0DH	PS_Eempty	清空 flash 指纹库
0EH	PS_WriteReg	设置系统参数
0FH	PS_ReadSysPara	读系统基本参数
1BH	PS_HighSpeedSearch	高速搜索 FLASH
1DH	PS_ValidTemplateNum	读有效模板个数

4.1.2.1 按两次指纹登录一个模板存于 flash 指纹库

发送获取图像指令：PS_GetImage，等待接收应答包直至读取到图像，发送生成特征指令：PS_Genchar，参数为 BufferID=1，等待接收应答包，若成功，灯带执行指定操作并继续发送获取图像指令：PS_GetImage，等待接收应答包直至读取到图像，发送生成特征指令：PS_Genchar，参数为 BufferID=2，若成功，灯带执行指定操作并发送注册模板指令 Godo_RegModel，等待接收应答包，若成功则发送存储模板指令：Godo_StoreChar，参数为 BufferID=2，并返回登录成功提示

4.1.2.2 用上位机下载一个指纹特征并以该特征搜索指纹库

发送指令 PS_DownChar，参数为 BufferID，收到接收应答包后发送特征文件数据包，收到接受应答包后发送指令 PS_Search，参数为 BufferID,StarPage,PageNum，若比对通过，则执行输出

4.1.2.3 从传感器获取指纹并生成特征然后上传给上位机

发送获取图像指令：PS_GetImage，等待接收应答包直至读取到图像，发送生成特征指令：PS_Genchar，参数为 BufferID=1，等待接收应答包，若成功发送指令 PS_UpChar，接收后续数据包直至结束包后结束。

4.1.2.4 从 flash 指纹库中读取一个指定的模板上传

发送指令 PS_LoadChar，参数为 PageID,BufferID 并等待接收应答包，发送指令 PS_UpChar，成功后接收后续数据包直至结束包后结束。

4.1.2.5 从传感器读入现场指纹搜索从 10-100 的指纹库区间

发送获取图像指令：PS_GetImage，等待接收应答包直至读取到图像，发送生成特征指令：PS_Genchar，等待接收应答包，若成功发送指令 PS_Search，参数中的 startPage 和 PageNum 分别设置为 10 和 90，等待接受应答包，返回匹配到的 FingerID 或者返回无结果。

4.1.3 关键代码

指纹的录入与获取模块中用到的主要函数如下所示：

代码 4-1 function

```
1 void Add_FR(void); //录入指纹
2 void Del_FR(void); //删除指纹
3 void press_FR(void); //搜索并匹配指纹
4 void ShowErrMsg(u8 ensure); //显示确认码错误信息
5 u16 GET_NUM(void); //获取数值
```

4.1.3.1 模块初始化

此部分功能模块在使用前需要完成系统时钟设置并完成延时初始化，同时需要初始化串口以支持 USMART 通信。此外还需要初始化按键、液晶显示屏、串口 2、状态引脚读入、内存池以及为 fatfs 相关变量申请内存等等。

代码 4-2 initial

```
1 Stm32_Clock_Init(9); //系统时钟设置
2 delay_init(72); //延时初始化
3 uart_init(72, 115200); //按键初始化
4 LCD_Init(); //初始化液晶显示屏
5 KEY_Init(); //初始化按键
6 usmart_dev.init(72);
7 usart2_init(36, usart2_baud); //串口初始化
8 PS_StaGPIO_Init(); //读入状态引脚
9 tp_dev.init(); //初始化触摸屏
10 mem_init(); //内存池初始化
11 exfun_init(); //为fatfs相关变量申请内存
12 f_mount(fs[1], "1:", 1); //挂载FLASH
```

4.1.3.2 与模块的通信

首先加载指纹识别界面，与 AS608 模块握手，建立通信，通讯成功后返回信息，显示波特率以及库指纹个数等信息。读取模块参数后等待指令的输入以进行录入指纹、删除指纹、匹配与搜索指纹等操作。

代码 4-3 main.c

```
1 LCD_Clear(WHITE);
2 POINT_COLOR=RED;
3 Show_Str_Mid(0, 0, "AS608 Fingerprint module test", 16, 240);
4 Show_Str_Mid(0, 20, "Author: @ALIENTEK", 16, 240);
5 POINT_COLOR=BLUE;
6 Show_Str_Mid(0, 40, "Connect with AS608....", 16, 240);
```

```

7 while(PS_HandShake(&AS608Addr)) //与AS608模块握手
8 {
9     LCD_Fill(0,40,240,80,WHITE);
10    Show_Str_Mid(0,40,"Cannot connect with AS608!",16,240);
11    delay_ms(1000);
12    LCD_Fill(0,40,240,80,WHITE);
13    Show_Str_Mid(0,40,"Try to connect again....",16,240);
14    delay_ms(1000);
15 }
16 LCD_Fill(0,40,240,320,WHITE);
17 Show_Str_Mid(0,40,"Connect success!",16,240); //通讯成功
18 str=mymalloc(30);
19 sprintf(str,"Baudrate:%d Addr:%x",usart2_baund,AS608Addr); //显示波特率
20 Show_Str(0,60,240,16,(u8*)str,16,0);
21 delay_ms(100);
22 ensure=PS_ValidTempleteNum(&ValidN); //显示确认码错误信息
23 if(ensure!=0x00)
24 ShowErrMsg(ensure); //读库指纹个数
25 ensure=PS_ReadSysPara(&AS608Para); //读AS608模块参数
26 if(ensure==0x00)
27 {
28     mymemset(str,0,50);
29     sprintf(str,"RemainNum:%d Level:%d",AS608Para.PS_max-ValidN,AS608Para.PS_level); //剩余指纹数量和安全等级
30     Show_Str(0,80,240,16,(u8*)str,16,0);
31 }
32
33 while(1)
34 {
35     key_num=AS608_get_keynum(0,170);
36     if(key_num)
37     {
38         if(key_num==1) Del_FR(); //删除指纹
39         if(key_num==3) Add_FR(); //录入指纹
40     }
41     if(PS_Sta) //检测是否有手指按压
42     {
43         press_FR(); //匹配并搜索指纹
44     }
45 }

```

4.1.3.3 录入指纹

发送获取图像指令：PS_GetImage，等待接收应答包直至读取到图像，发送生成特征指令：PS_Genchar，参数为BufferID=1，等待接收应答包，若成功，灯带执行指定操作并继续发送获取图像指令：PS_GetImage，等待接收应答包直至读取到图像，发送生成特征指令：PS_Genchar，参数为BufferID=2，若成功，灯带执行指定操作并发送注册模板指令 Godo_RegModel，等待接收应答包，若成功则发送存储模板指令：Godo_StoreChar，参数为BufferID=2，并返回登录成功提示

代码 4-4 main.c

```

1 void Add_FR(void)
2 {
3     u8 i=0,ensure ,processnum=0;
4     u16 ID;
5     while(1)
6     {
7         switch (processnum)
8         {
9             case 0:

```

```
10     i++;
11     LCD_Fill(0,100,lcddev.width,160,WHITE);
12     Show_Str_Mid(0,100,"Pleas touch finger!",16,240); //按压手指
13     ensure=PS_GetImage();
14     if(ensure==0x00)
15     {
16         ensure=PS_GenChar(CharBuffer1); //生成特征
17         if(ensure==0x00)
18         {
19             LCD_Fill(0,120,lcddev.width,160,WHITE);
20             Show_Str_Mid(0,120,"Fingerprint correct",16,240); //指纹正确
21             i=0;
22             processnum=1; //跳到第二步
23         } else ShowErrMsg(ensure);
24     } else ShowErrMsg(ensure);
25     break;
26
27 case 1:
28     i++;
29     LCD_Fill(0,100,lcddev.width,160,WHITE);
30     Show_Str_Mid(0,100,"Pleas touch finger again!",16,240); //再按一次手指
31     ensure=PS_GetImage();
32     if(ensure==0x00)
33     {
34         ensure=PS_GenChar(CharBuffer2); //生成特征
35         if(ensure==0x00)
36         {
37             LCD_Fill(0,120,lcddev.width,160,WHITE);
38             Show_Str_Mid(0,120,"Fingerprint correct",16,240); //指纹正确
39             i=0;
40             processnum=2; //跳到第三步
41         } else ShowErrMsg(ensure);
42     } else ShowErrMsg(ensure);
43     break;
44
45 case 2:
46     LCD_Fill(0,100,lcddev.width,160,WHITE);
47     Show_Str_Mid(0,100,"Compare twice fingerprint",16,240); //对比两次指纹
48     ensure=PS_Match();
49     if(ensure==0x00)
50     {
51         LCD_Fill(0,120,lcddev.width,160,WHITE);
52         Show_Str_Mid(0,120,"Twice fingerprint are same",16,240); //两次指纹是一样的
53         processnum=3; //跳到第四步
54     }
55     else
56     {
57         LCD_Fill(0,100,lcddev.width,160,WHITE);
58         Show_Str_Mid(0,100,"Compare fail,pleas touch again!",16,240); //对比失败,重新按手指
59         ShowErrMsg(ensure);
60         i=0;
61         processnum=0; //跳回第一步
62     }
63     delay_ms(1000);
64     break;
65
66 case 3:
67     LCD_Fill(0,100,lcddev.width,160,WHITE);
68     Show_Str_Mid(0,100,"Generate fingerprint template",16,240); //产生一个指纹模板
69     ensure=PS_RegModel();
70     if(ensure==0x00)
71     {
72         LCD_Fill(0,120,lcddev.width,160,WHITE);
73         Show_Str_Mid(0,120,"Generate fingerprint success",16,240); //生成指纹模板
74         processnum=4; //
```

```

75         }else {processnum=0;ShowErrMsg(ensure);}
76         delay_ms(1000);
77         break;
78
79     case 4:
80         LCD_Fill(0,100,lcddev.width,160,WHITE);
81         Show_Str_Mid(0,100,"Input ID and save with ENTER!",16,240);//
82         //输入ID并按ENTER键保存
83         Show_Str_Mid(0,120,"0=< ID <=299",16,240);
84         do
85             ID=GET_NUM();
86             while(!(ID<300));//输入ID须小于300
87             ensure=PS_StoreChar(CharBuffer2,ID);//储存模板
88             if(ensure==0x00)
89             {
90                 LCD_Fill(0,100,lcddev.width,160,WHITE);
91                 Show_Str_Mid(0,120,"Add fingerprint success!!!",16,240);//
92                 //添加指纹成功
93                 PS_ValidTemplateNum(&ValidN);//
94                 LCD_ShowNum(80,80,AS608Para.PS_max-ValidN,3,16);//
95                 delay_ms(1500);//
96                 LCD_Fill(0,100,240,160,WHITE);
97                 return ;
98             }else {processnum=0;ShowErrMsg(ensure);}
99             break;
100        }
101        if(i==5)//超过5次没有按手指则退出
102        {
103            LCD_Fill(0,100,lcddev.width,160,WHITE);
104            break;
105        }
106    }

```

4.1.3.4 指纹匹配与搜索

此部分分为两个功能，指定指纹的匹配与录入指纹并在指纹库中搜索对应的指纹。函数源码如下。

代码 4-5 main.c

```

1 void press_FR(void)
2 {
3     SearchResult seach;
4     u8 ensure;
5     char *str;
6     ensure=PS_GetImage();
7     if(ensure==0x00)//获取图像成功
8     {
9         ensure=PS_GenChar(CharBuffer1);
10        if(ensure==0x00) //生成特征成功
11        {
12            ensure=PS_HighSpeedSearch(CharBuffer1,0,300,&seach);
13            if(ensure==0x00)//搜索成功
14            {
15                LCD_Fill(0,100,lcddev.width,160,WHITE);
16                Show_Str_Mid(0,100,"Search fingerprint success",16,240);//
17                str=malloc(50);
18                sprintf(str,"Match ID:%d Match score:%d",seach.pageID,seach.mathsScore);
19                //显示匹配指纹的ID和分数
20                Show_Str_Mid(0,140,(u8*)str,16,240);
21                myfree(str);

```

```

21         }
22     else
23         ShowErrMsg(ensure);
24     }
25     else
26         ShowErrMsg(ensure);
27     delay_ms(1000); //延时后清除显示
28     LCD_Fill(0,100,lcddev.width,160,WHITE);
29   }
30 }
```

4.2 疫情新闻

4.2.1 与上位机通讯

4.2.1.1 信息搜集

返回数据

变量名	注释
pubDate	新闻发布时间
title	新闻标题
summary	新闻内容概述
infoSource	数据来源
sourceUrl	来源链接
province	省份或直辖市名称
provinceld	省份或直辖市代码

示例

1. ./nCoV/api/news?page=1&num=10
返回所有地区范围内第1页的新闻，每页10则。
2. ./nCoV/api/news?page=2&num=10&province=湖北省
返回湖北省范围内第2页的新闻，每页10则。因此，返回的新闻即为湖北省第11至20则新闻。

图 4-2 api 接口

在上位机中使用 python 利用 <https://lab.isaaclin.cn/nCoV/> 提供的新闻 api 接口（上图为 api 接口的详细信息），查询丁香园中的最新疫情新闻（利用 requests.get 函数获取内容），并将获取的新闻写入本地的 txt。

```

1 url = "https://lab.isaaclin.cn/nCoV/api/news?num=16"
2 r = requests.get(url)
3 sds = r.json()
4 titles = []
5 news = []
6
7 for i in range(8, 16):
8     titles.append(sds["results"][i]["title"])
9     news.append(sds["results"][i]["summary"])
10
11 for i in range(4, 12):
12     with open("C:\\\\Users\\\\Administrator\\\\Desktop\\\\news\\\\" + str(i) + ".txt", 'w', errors
13         ='ignore') as obj:
14         obj.write(titles[i - 4]+"\n")
15     obj.close()
16
17 for i in range(12, 20):
18     with open("C:\\\\Users\\\\Administrator\\\\Desktop\\\\news\\\\" + str(i) + ".txt", 'w', errors
19         ='ignore') as obj:
```

```
18         obj.write(news[i - 12] + "\n")
19     obj.close()
20 print("collect done")
```

4.2.1.2 信息通讯

在上位机中使用 Python 实现串口通讯的函数主要有以下四个，包括打开串口，发送数据，接收数据，关闭串口。考虑到每次发送的容量有限，故把先前搜集的消息每一条存入一个 txt 文件，而后传送时传送多次，一次只传送一个 txt 中的内容。

```
1 # 串口打开函数
2 def open_ser():
3     port = 'com5'    # 串口号
4     baudrate = 115200  # 波特率
5     try:
6         global ser
7         ser = serial.Serial(port, baudrate, timeout=2)
8         if (ser.isOpen() == True):
9             print("串口打开成功")
10    except Exception as exc:
11        print("串口打开异常", exc)
12
13 # 数据发送
14 def send_msg(content):
15     try:
16         send_datas = ""
17         for i in range(0, 10):
18             try:
19                 send_datas += str(content[i])
20             except:
21                 send_datas = content
22                 break
23         send_datas += '\n'
24         ser.write(str(send_datas).encode("gbk"))
25         print("已发送信号:", send_datas)
26    except Exception as exc:
27        print("发送异常", exc)
28
29 # 接收数据
30 def read_msg():
31     try:
32         #print("等待接收数据")
33         while True:
34             data = ser.read(ser.in_waiting)
35             if data != '':
36                 break
37             #print("已接受到数据:", data)
38    except Exception as exc:
39        print("读取异常", exc)
40
41 # 关闭串口
42 def close_ser():
43     try:
44         ser.close()
45         if ser.isOpen():
46             print("串口未关闭")
47         else:
48             print("串口已关闭")
49    except Exception as exc:
50        print("串口关闭异常", exc)
```

4.2.2 重定向输入输出函数到串口

通过重定向，在stm32使用scanf函数可以直接从上位机接收信息，使用printf可以直接向上位机发送信息。详细使用串口进行通信的代码参见野火例程。

```

1 //重定向c库函数printf到串口，重定向后可使用printf函数
2 int fputc(int ch, FILE *f)
3 {
4     /* 发送一个字节数据到串口 */
5     USART_SendData(DEBUG_USARTx, (uint8_t) ch);
6     /* 等待发送完毕 */
7     while (USART_GetFlagStatus(DEBUG_USARTx, USART_FLAG_TXE) == RESET);
8     return (ch);
9 }
10
11 //重定向c库函数scanf到串口，重写后可使用scanf、getchar等函数
12 int fgetc(FILE *f)
13 {
14     /* 等待串口输入数据 */
15     while (USART_GetFlagStatus(DEBUG_USARTx, USART_FLAG_RXNE) == RESET);
16     return (int)USART_ReceiveData(DEBUG_USARTx);
17 }
```

4.2.3 FatFs 文件系统

FatFs文件系统的搭建基于SPI Flash芯片驱动程序，为移植FatFs方便，直接拷贝一份该工程，并在工程基础上添加FatFs组件，并修改main函数的用户程序。

4.2.3.1 关键变量

文件系统的主函数中，需要使用以下关键变量。其中

```

1 FATFS fs;                                /* FatFs文件系统对象 */
2 FIL fnew;                                 /* 文件对象 */
3 FRESULT res_flash;                         /* 文件操作结果 */
4 UINT fnum;                                /* 文件成功读写数量 */
5 BYTE ReadBuffer[2048]={0};                  /* 读缓冲区 */
6 BYTE WriteBuffer[] = "";                   /* 写缓冲区 */
```

FATFS是在ff.h文件定义的一个结构体类型，针对的对象是物理设备，包含了物理设备的物理编号、扇区大小等等信息，一般都需要为每个物理设备定义一个FATFS变量。

FIL也是在ff.h文件定义的一个结构体类型，针对的对象是文件系统内具体的文件，包含了文件很多基本属性，比如文件大小、路径、当前读写地址等等。如果需要在同一时间打开多个文件进行读写，才需要定义多个FIL变量，不然一般定义一个FIL变量即可。

FRESULT是也在ff.h文件定义的一个枚举类型，作为FatFs函数的返回值类型，主要管理FatFs运行中出现的错误。总共有19种错误类型，包括物理设备读写错误、找不到文件、没有挂载工作空间等等错误。这在实际编程中非常重要，当有错误出现时我们要停止文件读写，通过返回值我们可以快速定位到错误发生的可能地点。如果运行没有错误才返回FR_OK。

fnum是个32位无符号整形变量，用来记录实际读取或者写入数据的数组。buffer和

textFileBuffer 分别对应读取和写入数据缓存区，都是 8 位无符号整形数组。

4.2.3.2 文件读写函数

4.2.3.2.1 f_mount 函数 FatFs 的第一步工作就是使用挂载工作区。f_mount 函数有三个形参，第一个参数是指向 FATFS 变量指针，如果赋值为 NULL 可以取消物理设备挂载。第二个参数为逻辑设备编号，使用设备根路径表示，与物理设备编号挂钩。

f_mount 函数会返回一个 FRESULT 类型值，指示运行情况。如果 f_mount 函数返回值为 FR_NO_FILESYSTEM，说明没有 FAT 文件系统，比如新出厂的 SPIFlash 芯片就没有 FAT 文件系统。我们就必须对物理设备进行格式化处理。最后，不再使用文件系统时，使用 f_mount 函数取消挂载。

4.2.3.2.2 f_mkfs 函数 使用 f_mkfs 函数可以实现格式化操作。f_mkfs 函数有三个形参，第一个参数为逻辑设备编号；第二参数可选 0 或者 1，0 表示设备为一般硬盘，1 表示设备为软盘。第三个参数指定扇区大小，如果为 0，表示通过 disk_ioctl 函数获取。格式化成功后需要先取消挂载原来设备，再重新挂载设备。在设备正常挂载后，就可以进行文件读写操作了。

4.2.3.2.3 f_open/f_close 函数 使用文件之前，必须使用 f_open 函数打开文件，不再使用文件必须使用 f_close 函数关闭文件，这个跟电脑端操作文件步骤类似。f_open 函数有三个形参，第一个参数为文件对象指针。第二参数为目标文件，包含绝对路径的文件名称和后缀名。第三个参数为访问文件模式选择，可以是打开已经存在的文件模式、读模式、写模式、新建模式、总是新建模式等的或运行结果。

4.2.3.2.4 f_write/f_read 函数 成功打开文件之后就可以使用 f_write 函数和 f_read 函数对文件进行写操作和读操作。这两个函数用到的参数是一致的，只不过一个是数据写入，一个是数据读取。f_write 函数第一个形参为文件对象指针，使用与 f_open 函数一致即可。第二个参数为待写入数据的首地址，对于 f_read 函数就是用来存放读出数据的首地址。第三个参数为写入数据的字节数，对于 f_read 函数就是欲读取数据的字节数。第四个参数为 32 位无符号整形指针，这里使用 fnum 变量地址赋值给它，在运行读写操作函数后，fnum 变量指示成功读取或者写入的字节个数。

下面的代码为 main 函数中挂载磁盘和取消挂载的操作。

```
1 int main(void)
2 {
3     /* 其它配置的初始化略 */
4     res_flash = f_mount(&fs, "1:", 1);
5     while(1)
6     {
7     }
8     f_close(&fnew);
9     f_mount(NULL, "1:", 1);
10 }
```

下面的代码为stm32接收上位机发送的新闻，并将其写入文件系统的过程。

```
1 char tmp_file_name[100];
2 scanf("%s",WriteBuffer);
3
4 printf("%s",WriteBuffer);
5
6 printf("\r\n***** 即将进行文件写入... *****\r\n");
7
8 if (count<0) count=1;
9
10 sprintf(tmp_file_name, "1:%ld.txt",count);
11 res_flash = f_open(&fnew,tmp_file_name ,FA_CREATE_ALWAYS | FA_WRITE );
12
13 count++;
14 if (count==20)
15 {
16     count=1;
17     LCD_ini();
18 }
19 if ( res_flash == FR_OK )
20 {
21     printf("» 打开/创建 %s文件成功，向文件写入数据。 \r\n",tmp_file_name);
22     /* 将指定存储区内容写入到文件内 */
23     res_flash=f_write(&fnew,WriteBuffer,sizeof(WriteBuffer),&fnum);
24     if(res_flash==FF_OK)
25     {
26         printf("» 文件写入成功，写入字节数据: %d\r\n",fnum);
27         printf("» 向文件写入的数据为: \r\n%s\r\n",WriteBuffer);
28     }
29 else
30 {
31     printf("！！文件写入失败: (%d)\r\n",res_flash);
32 }
33 /* 不再读写，关闭文件 */
34 f_close(&fnew);
35 }
36 else
37 {
38     LED_RED;
39     printf("！！打开/创建文件失败。 \r\n");
40 }
```

4.2.4 液晶显示

液晶显示部分用到的函数主要有以下几个：

```
1 ILI9341_GramScan ( 6 ); /*确定扫描输出模式*/
2 ILI9341_Clear(0,0,LCD_X_LENGTH,LCD_Y_LENGTH); /*清屏，显示全黑*/
3 LCD_SetTextColor(RED); /*设置字体颜色*/
4 ILI9341_DispStringLine_EN_CH(LINE(4),first);/*设置某行内容字体颜色*/
```

4.3 基于DS18B20的室内温控

4.3.1 模块概述

室内温控部分由DS18B20温度模块、CJY5V直流风扇模块，以及开发板上的LCD显示屏、按键、有源蜂鸣器和LED组成，实现了温度阈值的设定与显示、温度测量与显示、超过

阈值时的报警与降温等功能，体现了通过嵌入式设备控制各种模块，实现智能家居系统的思想。

该部分的主要功能为使用 DS18B20 模块进行温度测量，并调用相关模块给出相应反馈，同时，温度参数可以通过按键进行设置调节。该部分完整的工作流程为：STM 开发板的 LCD 显示屏展示当前测量温度及设定的温度阈值，彩色 LED 正常情况下为绿灯，用户可以通过板上按键调节阈值参数；当 DS18B20 模块测量温度发生改变时，如用户身体接近模块测量区域时，若温度超过 **报警阈值 Alert**，有源蜂鸣器报警，LED 转为红灯；若温度超过 **降温阈值 Cool**，CJY5V 风扇启动进行降温。

4.3.2 关键代码

液晶显示、按键中断、LED 与按键配置的相关代码野火例程中已经给出并讲解，此处不再重复，仅给出 ds18b20 的初始化以及使用的代码。该模块中使用的主要函数如下所示：

代码 4-6 main.c

```

16 u8 DS18B20_Init(void);                                // 初始化DS18B20
17 short DS18B20_Get_Temp(void);                          // 获取温度
18 void DS18B20_Start(void);                             // 开始温度转换
19 void DS18B20_Write_Byte(u8 dat);                      // 写入一个字节
20 u8 DS18B20_Read_Byte(void);                           // 读出一个字节
21 u8 DS18B20_Read_Bit(void);                            // 读出一个位
22 u8 DS18B20_Check(void);                             // 检测是否存在DS18B20
23 void DS18B20_Rst(void);                            // 复位DS18B20

```

4.3.2.1 模块初始化

和其它外设一样，初始化时需要开启时钟，配置 GPIO 口的管脚，模式，速度。此外，查看 ds18b20 手册，为了使其能正常工作，在使用前还需将连接 DS18B20 的 GPIO 口置 1，并复位该模块（先拉低电平 750us 后拉高电平）。初始化完毕，调用 DS18B20_Check(void) 函数检测 ds18b20 是否初始化成功，若返回 0 说明其能正常工作。

代码 4-7 main.c

```

16 u8 DS18B20_Init(void)
17 {
18     GPIO_InitTypeDef GPIO_InitStructure;
19     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);      // 使能PORTB口时钟
20     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;                  // PORTB1 推挽输出
21     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
22     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
23     GPIO_Init(GPIOA, &GPIO_InitStructure);
24     GPIO_SetBits(GPIOA, GPIO_Pin_11);                         // 输出1
25     DS18B20_Rst();
26     return DS18B20_Check();
27 }
28
29 void DS18B20_Rst(void)
30 {
31     DS18B20_IO_OUT(); //SET PA0 OUTPUT
32     DS18B20_DQ_OUT = 0; //拉低DQ
33     Delay_us(750);    //拉低750us
34     DS18B20_DQ_OUT = 1; //DQ=1

```

```

35     Delay_us(15);      //15US
36 }
37
38 //等待DS18B20的回应
39 //返回1:未检测到DS18B20的存在
40 //返回0:存在
41 u8 DS18B20_Check(void)
42 {
43     u8 retry = 0;
44     DS18B20_IO_IN(); //SET PA0 INPUT
45     //printf("%ul", DS18B20_DQ_IN);
46     while (DS18B20_DQ_IN&&retry < 200)
47     {
48         retry++;
49         Delay_us(1);
50     }
51 ;
52 if (retry >= 200){printf("retry1:%u\n",retry);return 1;}
53 else retry = 0;
54 while (!DS18B20_DQ_IN&&retry < 240)
55 {
56     retry++;
57     Delay_us(1);
58 }
59 ;
60 if (retry >= 240){printf("retry2:%u\n",retry);return 1;}
61 return 0;
62 }
```

4.3.2.2 寄存器操作

ds18b20 测温模块的数据通信主要是通过 ROM 指令进行寄存器读写来进行。通过按时序要求逐位进行读/写操作，如下实现了在寄存器中读/写一字节的方法。

代码 4-8 main.c

```

44 //从DS18B20读取一个字节
45 //返回值: 读到的数据
46 u8 DS18B20_Read_BytE(void)      // read one byte
47 {
48     u8 i;
49     u8 dat = 0;
50     for(i = 0;i < 8;i++)
51     {
52         dat >= 1;
53         DS18B20_IO_OUT(); //SET PA0 OUTPUT
54         DS18B20_DQ_OUT=0;
55         Delay_us(2);
56         DS18B20_DQ_OUT=1;
57         DS18B20_IO_IN(); //SET PA0 INPUT
58         Delay_us(2);
59         if(DS18B20_DQ_IN)dat |= 0x80;
60         Delay_us(70);
61     }
62     return(dat);
63 }
64 //写一个字节到DS18B20
65 //dat: 要写入的字节
66 void DS18B20_Write_BytE(u8 dat)
67 {
68     u8 j;
69     u8 testb;
70     DS18B20_IO_OUT(); //SET PA0 OUTPUT;
71     for (j = 1; j <= 8; j++)
```

```

72     {
73         testb = dat & 0x01;
74         dat = dat >> 1;
75         if (testb)
76         {
77             DS18B20_DQ_OUT = 0;// Write 1
78             Delay_us(2);
79             DS18B20_DQ_OUT = 1;
80             Delay_us(60);
81         }
82     else
83     {
84         DS18B20_DQ_OUT = 0;// Write 0
85         Delay_us(60);
86         DS18B20_DQ_OUT = 1;
87         Delay_us(2);
88     }
89 }
90 }
```

4.3.2.3 温度读取与转换

关于 ds18b20 测温模块的关键操作为温度的读取与转换。按照手册，写入开始温度转换 0x44 和开始读 RAM 0xbe 两个指令码，并依次读出温度数据的 LSB 与 MSB 两个字节，经转换后即可得到精度为 0.1 摄氏度的温度浮点值。

代码 4-9 main.c

```

44 //从ds18b20得到温度值
45 //精度: 0.1C
46 //返回值: 温度值 (-550~1250)
47 short DS18B20_Get_Temp(void)
48 {
49     u8 temp;
50     u8 TL, TH;
51     short tem;
52
53     DS18B20_Start();           // ds1820 start convert
54     DS18B20_Rst();
55     DS18B20_Check();
56     DS18B20_Write_Byt(0xcc); // skip rom
57     DS18B20_Write_Byt(0xbe); // read RAM
58     TL = DS18B20_Read_Byt(); // LSB
59     TH = DS18B20_Read_Byt(); // MSB
60
61     if (TH > 7)
62     {
63         TH = ~TH;
64         TL = ~TL;
65         temp = 0;//温度为负
66     }
67     else temp = 1;//温度为正
68     tem = TH; //获得高八位
69     tem <= 8;
70     tem += TL;//获得底八位
71     tem = (float)tem * 0.625;//转换
72     if (temp) return tem; //返回温度值
73     else return -tem;
74 }
75
76 //开始温度转换
77 void DS18B20_Start(void)// ds1820 start convert
78 {
```

```

79     DS18B20_Rst();
80     DS18B20_Check();
81     DS18B20_Write_Byte(0xcc); // skip rom
82     DS18B20_Write_Byte(0x44); // convert
83 }
```

4.4 无线物联网

我们项目其中一部分模块的内容是利用 ESP8266 WIFI 模块实现指南者开发板与上位机或者手机 app 进行数据通信。而 ESP8266 是串口型 WIFI，速度比较低，不能用来传输图像或者视频这些大容量的数据，主要应用于数据量传输比较少的场合，比如温湿度信息，一些传感器的开关量等。所以我们利用手中现有的 DHT11 温湿度模块，读取温湿度信息，并且利用 ESP8266 进行传输。

该部分的关键代码包括 DHT11 模块的固件驱动程序代码和 ESP8266 模块程序的核心部分代码，即传输数据函数 `ESP8266_SendString()` 和接收数据函数 `ESP8266_ReceiveString()`。

4.4.1 DHT11 温湿度读取

其中 DHT11 模块的固件核心代码功能包初始化函数 `DHT11_Init()` 负责拉高 GPIOB10，并且使用配置 DHT1 所用 I/O 口函数 `DHT11_GPIO_Config()` 设置外设时钟、引脚速率和引脚模式通用推挽输出，测试并记录环境的温湿度数据，DATA 线由上拉电阻拉高一直保持高电平。在 DHT11 的 DATA 引脚要输出 40 位数据时，微处理器判断 I/O 电平是否变为低电平，并且根据 I/O 电平的变化接受 40 位数据，一次完整的数据传输为 40bit，高位先出，数据的格式为 8bit 湿度整数 + 8bit 湿度小数 + 8bit 温度整数 + 8bit 温度小数 + 8bit 校验和。其控制数据完整输出函数 `DHT11_Read_TempAndHumidity()` 代码如下所示：

```

1 uint8_t DHT11_Read_TempAndHumidity(DHT11_Data_TypeDef *DHT11_Data)
2 {
3     /* 输出模式 */
4     DHT11_Mode_Out_PP();
5     /* 主机拉低 */
6     DHT11_Dout_0;
7     /* 延时 18ms */
8     DHT11_DELAY_MS(18);
9     /* 总线拉高 主机延时 30us */
10    DHT11_Dout_1;
11    DHT11_DELAY_US(30); // 延时 30us
12    /* 主机设为输入 判断从机响应信号 */
13    DHT11_Mode_IPU();
14    /* 判断从机是否有低电平响应信号 如不响应则跳出，响应则向下运行 */
15    if(DHT11_Dout_IN() == Bit_RESET)
16    {
17        /* 轮询直到从机发出的 80us 低电平 响应信号结束 */
18        while(DHT11_Dout_IN() == Bit_RESET);
19        /* 轮询直到从机发出的 80us 高电平 标置信号结束 */
20        while(DHT11_Dout_IN() == Bit_SET);
21        /* 开始接收数据 */
22        DHT11_Data->humid_int = DHT11_ReadByte();
23        DHT11_Data->humid_deci = DHT11_ReadByte();
24        DHT11_Data->temp_int = DHT11_ReadByte();
```

```

25     DHT11_Data->temp_deci= DHT11_ReadByte();
26     DHT11_Data->check_sum= DHT11_ReadByte();
27     /*读取结束，引脚改为输出模式*/
28     DHT11_Mode_Out_PP();
29     /*主机拉高*/
30     DHT11_Dout_1;
31     /*检查读取的数据是否正确*/
32     if(DHT11_Data->check_sum == DHT11_Data->humid_int + DHT11_Data->humid_deci +
33         DHT11_Data->temp_int+ DHT11_Data->temp_deci)
34         return SUCCESS;
35     else
36         return ERROR;
37 }
38     return ERROR;
39
40 }
```

4.4.2 ESP8266 通信

ESP8266 模块的固件支持代码函数分类较多，分别对应模块不同功能。其中，除了初始化模块和初始化 GPIO 引脚函数以外，我们需要使用对 **ESP8266** 发送指令函数 **ESP8266_Cmd()** 来控制对 WF-ESP8266 模块发送 AT 指令，AT 指令集负责控制 WIFI 模块的启停、功能和 TCP/IP 工具箱。并且在工作模式函数 **ESP8266_Net_Mode_Chose()** 选择 WF-ESP8266 模块的工作模式，物联网无线通信部分便是通过 AT 指令，选择 ESP8266 工作在 AP 模式下。除了这些函数以外，还有如 **WIFI 配置函数** **ESP8266_BuildAP()** 等诸多函数用来在 AP 模式下配置 wifi 热点信息。下面重点介绍一下 wifi 通信所用的传输数据函数 **ESP8266_SendString()** 和接收数据函数 **ESP8266_ReceiveString()**：

```

1  bool ESP8266_SendString ( FunctionalState enumEnUnvarnishTx, char * pStr, u32
    ulStrLength, ENUM_ID_NO_TypeDef ucId )
2 {
3     char cStr [20];
4     bool bRet = false;
5     if ( enumEnUnvarnishTx )
6     {
7         macESP8266_Uart ( "%s", pStr );
8         bRet = true;
9     }
10    else
11    {
12        if ( ucId < 5 )
13            sprintf ( cStr, "AT+CIPSEND=%d,%d", ucId, ulStrLength + 2 );
14        else
15            sprintf ( cStr, "AT+CIPSEND=%d", ulStrLength + 2 );
16        ESP8266_Cmd ( cStr, "> ", 0, 100 );
17        bRet = ESP8266_Cmd ( pStr, "SEND OK", 0, 500 );
18    }
19    return bRet;
20 }
21
22 char * ESP8266_ReceiveString ( FunctionalState enumEnUnvarnishTx )
23 {
24     char * pRecStr = 0;
25     strEsp8266_Fram_Record .InfBit .FramLength = 0;
26     strEsp8266_Fram_Record .InfBit .FramFinishFlag = 0;
27     while ( ! strEsp8266_Fram_Record .InfBit .FramFinishFlag );
28     strEsp8266_Fram_Record .Data_RX_Buf [ strEsp8266_Fram_Record .InfBit .FramLength ] =
29         '\0';
```

```

29     if ( enumEnUnvarnishTx )
30         pRecStr = strEsp8266_Fram_Record .Data_RX_Buf;
31     else
32     {
33         if ( strstr ( strEsp8266_Fram_Record .Data_RX_Buf, "+IPD" ) )
34             pRecStr = strEsp8266_Fram_Record .Data_RX_Buf;
35     }
36     return pRecStr;
37 }
```

4.4.3 显示界面

使用 ILI9341_DisplineStringLine 函数绘制界面以对各种可控设备标识开关状态和相关数据，包括 4 盏 led 灯、电风扇、音乐播放、报警器、室内温湿度。

```

1 strcpy(dispstate, "OFF");
2 sprintf(dispBuff, "LED1:%s", dispstate);
3 ILI9341_DisplineStringLine_EN(LINE(2), dispBuff);
4 /* LED2-LED4略 */
5 sprintf(dispBuff, "BEEP:%s", dispstate);
6 ILI9341_DisplineStringLine_EN(LINE(6), dispBuff);
7 sprintf(dispBuff, "FAN:%s", dispstate);
8 ILI9341_DisplineStringLine_EN(LINE(7), dispBuff);
9 sprintf(dispBuff, "TEMPERATURE:");
10 ILI9341_DisplineStringLine_EN(LINE(8), dispBuff);
11 sprintf(dispBuff, "HUMIDITY: ");
12 ILI9341_DisplineStringLine_EN(LINE(9), dispBuff);
13 sprintf(dispBuff, "MUSIC:%s", dispstate);
14 ILI9341_DisplineStringLine_EN(LINE(10), dispBuff);
```

当通过无线通讯收到控制信号时，对应修改该行状态为 ON/OFF 即可。

4.4.4 手机 APP 通信

在物联网 app 这一环节，本项目为了多样化物联网 APP 控制开发板的功能，将开发板打造成物联网智能家居控制终端的雏形，尝试模拟日常的家居生活，为指南者开发板增添一个直流小风扇的模块，来模拟嵌入式开发板对家居家电的控制；同时利用手机 app 控制无源蜂鸣器播放出悦耳的音乐；进一步通过连接其他 LED 灯组并且接入 app 控制，达到模拟控制家具电灯开关的效果。并且，所有的控制效果我们都将呈现在 LCD 液晶屏上。目前 APP 开发支持安卓系统。

整个项目的通信基础是 WIFI 芯片：ESP8266，已经集成在开发板上，将之设置为 AP 模式，便可向外发出 WIFI 信号，程序上设置 WIFI 信号名称为：YehuoLink，加密方式为 OPEN，不需要密码。然后用手机连接该 WIFI 信号：YehuoLink，连接成功之后，运行手机 APP “秉火物联”，即可用手机控制开发板上的 LED 的亮灭，蜂鸣器的开关，开发板上安装了 DHT11 温湿度传感器，传感器的信息也会传到手机上显示。

利用手机 APP 控制和驱动外设的原理是：利用手机 APP 各部分按键向 ESP8266 发送不同的数据，野火指南者开发板利用接收到的不同数据进行外设的驱动和控制。在项目 **test.c** 中，首先向开发板上的 ESP8266 发送一位数据来决定控制哪一个外设，然后在发送数据 ‘0’

或‘1’来控制外设开启或关闭，利用 switch 多路分支选择语句达到控制开发板各个外设的功能。其控制部分逻辑代码如下所示：

```
1 switch ( cStr[9] )
2 {
3     case '0':
4         LED(ON);
5         LCD_ClearLine(LINE(5));
6         strcpy(disppstate, "ON");
7         sprintf(dispBuff, "LED4:%s", dispstate);
8         ILI9341_DisppStringLine_EN(LINE(5), dispBuff);
9         break;
10    case '1':
11        LED(OFF);
12        LCD_ClearLine(LINE(5));
13        strcpy(disppstate, "OFF");
14        sprintf(dispBuff, "LED4:%s", dispstate);
15        ILI9341_DisppStringLine_EN(LINE(5), dispBuff);
16        break;
17    case '2':
18        FAN(ON);
19        LCD_ClearLine(LINE(7));
20        strcpy(disppstate, "ON");
21        sprintf(dispBuff, "FAN:%s", dispstate);
22        ILI9341_DisppStringLine_EN(LINE(7), dispBuff);
23        break;
24    case '3':
25        FAN(OFF);
26        LCD_ClearLine(LINE(7));
27        strcpy(disppstate, "OFF");
28        sprintf(dispBuff, "FAN:%s", dispstate);
29        ILI9341_DisppStringLine_EN(LINE(7), dispBuff);
30        break;
31    case '4':
32        BEEP(ON);
33        play_music();
34        LCD_ClearLine(LINE(10));
35        strcpy(disppstate, "ON");
36        sprintf(dispBuff, "MUSIC:%s", dispstate);
37        ILI9341_DisppStringLine_EN(LINE(10), dispBuff);
38        break;
39
40    default:
41        break;
42 }
```

手机 APP 通信示例如图 4-3 所示。

4.4.5 上位机通信

在 ESP8266 与上位机通信这一环节，我们使用开发工具——电脑网络助手向上位机传输 DHT11 温湿度数据。此时将 ESP8266 设置为 Station 模式，在程序下载到板子之前，首先要设置好局域网的 wifi 名称、密码、上位机的 IP 以及服务器端口号给开发板上电，把编译好的程序下载到开发板，用 USB 线连接好电脑和开发板的串口接口：USB TO UART，打开调试助手软件，串口部分：设置好串口号，配置好波特率；网络部分：设置好协议类型，IP 地址，端口号。点击开始监听，上位机就不断打印出读取到的数据。

上位机通信示例如图 4-4 所示。



图 4-3 手机 APP 通信示例



图 4-4 上位机通信示例

4.5 多任务并行操作系统

4.5.1 嵌入式操作系统的选择

在本项目各个成员将各自的模块和功能调试完毕后，我们开始考虑将多个模块功能集成到一块开发板上，所以项目肯定离不开对嵌入式操作系统的研究。经过了解，基于 STM 平台且满足实时控制要求操作系统，有以下 6 种可供移植选择。分别为 Clinux、C / OS-III、eCos、FreeRTOS、都江堰操作系统 (djyos) 和 Huawei Life OS。各个操作系统的优劣评价如下，因此我们经过多种因素的权衡，最终决定使用 Huawei Life OS。

1. **Clinux** 虽然是一种优秀的嵌入式 Linux 版本，但是因为没有 MMU 内存管理单元，所 以其多任务的实现需要一定技巧，移植此系统需要至少 512KB 的 RAM 空间，1MB 的 ROM/FLASH 空间，而 stmf103 指南者只拥有 512K 的 FLASH，需要外接存储器，这就增加了硬件设计的成本。Clinux 结构复杂，移植相对困难，内核也较大，其实时性也差一些。
2. **C/OS-III** 是在 C/OS 的基础上发展起来的,C/OS-II 中断处理比较简单。一个中断向量上只能挂一个中断服务子程序 ISR，而且用户代码必须都在 ISR(中断服务程序) 中完成。ISR 需要做的事情越多，中断延时也就越长。考虑到项目的创新性，还是觉得不用指南者出厂自带的 C/OS-III 操作系统。
3. 另外三个操作系统：**eCOS**、**Free RTOS** 和 **djyOS** 都多多少少存在应用不广泛、多为针对工业级嵌入式系统、需要外扩第三方 GUI 等问题。

4.5.2 Huawei Lite OS 简介

华为 LiteOS 是华为轻量级物联网操作系统，其体系架构如下图4-5所示：

华为 LiteOS 由 Huawei LiteOS kernel、互联互通中间件、开放 API 以及安全组成：

1. Huawei LiteOS Kernel 为 Huawei LiteOS 基础内核，属于最精简 RTOS，包括任务管理、内存管理、时间管理、通信机制、中断管理、队列管理、事件管理、定时器、异常管理等操作系统基础组件，可以单独运行；

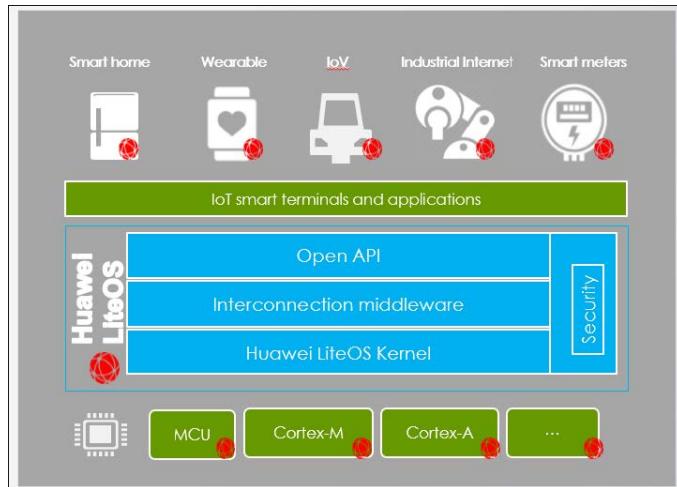


图 4-5 Huawei LiteOS 体系架构

2. Huawei LiteOS 互联互通中间件，可覆盖短距(Wifi、BT等)/广域(4G/NB-IoT)协议，可解决不同协议架构间互联互通、互操作问题；
3. Huawei LiteOS 提供面向不同 IoT 领域的业务 Profile，并以开放 API 的方式提供给第三方开发者；
4. Huawei LiteOS 构建完备的设备侧安全、轻量级 E2E 传输安全能力。

Huawei Lite OS 提供的操作系统主要模块和功能包括：

1. **任务**：提供任务的创建、删除、延迟、挂起、恢复等功能，以及锁定和解锁任务调度。任务按优先级可抢占、同优先级时间片轮转调度的方式调度。
2. **任务同步**：提供信号量、互斥锁等功能。
 - (a) **信号量**：支持信号量的创建、删除、PV 等功能。
 - (b) **互斥锁**：支持互斥锁的创建、删除、PV 等功能。
3. **硬件相关**：提供中断、定时器等功能。
 - (a) **中断**：提供中断的创建、删除、使能、禁止、请求位的清除等功能。
 - (b) **定时器**：提供定时器的创建、删除、启动、停止等功能。
4. **IPC 通信**：提供事件、消息队列功能。
 - (a) **事件**：支持读事件和写事件功能。
 - (b) **消息队列**：支持消息队列的创建、删除、发送和接收功能。
5. **时间管理**：多种时间的调用与操作。
 - (a) **系统时间**：系统时间是由定时/计数器产生的输出脉冲触发中断而产生的。
 - (b) **Tick 时间**：Tick 是操作系统调度的基本时间单位，对应的时长由系统主频及每秒 Tick 数决定，由用户配置。
 - (c) **软件定时器**：以 Tick 为单位的定时器功能，软件定时器的超时处理函数在系统创建的 Tick 软中断中被调用。
6. **内存管理**：提供静态内存和动态内存两种算法，支持内存申请、释放。目前支持的内

存管理算法有固定大小的 BOX 算法、动态申请 DLINK 算法。提供内存统计、内存越界检测功能。

7. 异常接管：异常接管是指在系统运行过程中发生异常后，跳转到异常处理信息的钩子函数，打印当前发生异常函数调用栈信息，或者保存当前系统状态的一系列动作。

4.5.3 关键代码及操作

本项目由于时间原因，在移植 Huawei Lite OS 基础上未能将所有组员完成的功能模块以 task 形式进行合并，实现多任务并行和实时交互。但是将固件库中的点亮 LED 例程和我们的温湿度传感器数据读取功能进行合并，实现并行，为将来各个组员的功能模块集成提供了实例基础和可能性。

移植 Huawei Lite OS，实现 LED 任务和 DHT11 任务并行的步骤分为四步：

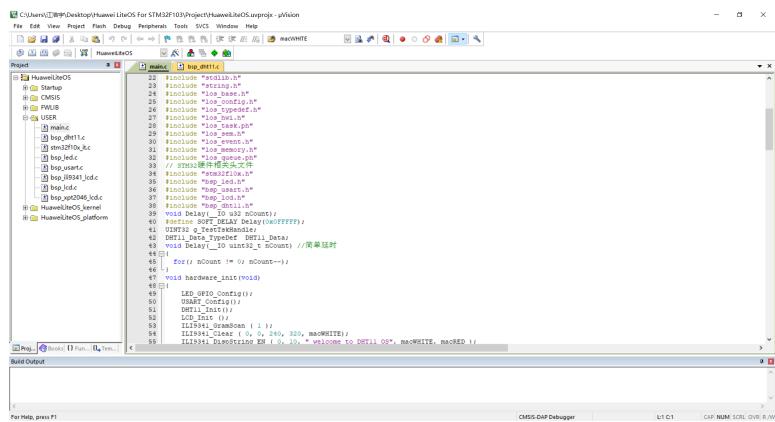


图 4-6 Huawei LiteOS 项目

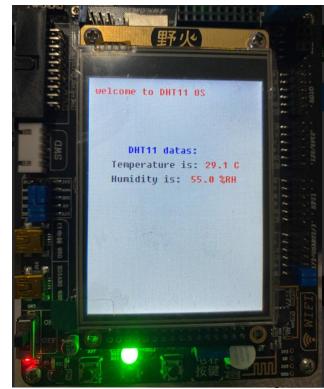


图 4-7 Huawei LiteOS 效果图

4.5.3.1 建立应用项目

新建 STM32 的工程模板，将 Huawei LiteOS Kernel 源码拷贝至项目目录，并且把点亮 LED 项目和 DHT11 温湿度读取项目文件也拷贝至同一个文件夹下，工程中右键点击 Target1，选择 Manage Project Items，如上图所示，接下来弹出新的对话框，在 Project Targets 一栏，我们将 Target 名字修改为 Huawei_LiteOS，然后在 Groups 一栏删掉 Source Group1，建立六个 Groups：分别为 STARTUP、CMSIS、FWLB、HuaweiLiteOS_Kernel、HuaweiLiteOS_Platform、USER 六个文件夹。然后点击 OK；接着把我们需要的应用代码添加进来，添加相应头文件，便可以编译；编译成功的项目如图4-6。

4.5.3.2 配置系统参数

在 los_config.h 中配置系统参数，由于移植的开发板是野火指南者，属于 STMF103，因此将 OS_SYS_CLOCK 设为系统主频 72Mhz。

```

1 #define OS_SYS_CLOCK 72000000
2 #define LOSCFG_BASE_CORE_TSK_LIMIT 15

```

```

3 #DEFINE OS_SYS\ _MEM\ _SIZE 0X00008000
4 #DEFINE LOSCFG\ _BASE\ _CORE\ _TSK\ _DEFAUL\ T_ STACK\ _SIZE SIZE(0X2D0)
5 #DEFINE LOSCFG\ _BASE\ _CORE\ _SWTMR_LIMIT 16

```

4.5.3.3 创建运行任务

创建 Huawei Lite OS 任务，实现 LED 指示灯点亮任务与 DHT11 读取温湿度数据显示在屏幕任务并行。STM32 芯片上电启动后，先执行 startup.s 汇编文件中的代码，执行堆和栈的初始化配置、中断向量表的配置，然后将程序引导到 los_config.c 文件中 main() 函数，开始进行 Huawei LiteOS 初始化，main 函数中实现 Huawei LiteOS 系统任务注册，内存池初始化等，然后调用 Huawei LiteOS 提供给开发者的入口函数 osAppInit()。开发者需要在用户程序代码中实现 osAppInit() 函数，在此函数中添加用户任务。在 Huawei LiteOS 中，我们通过函数 LOS_TaskCreate() 来创建任务，LOS_TaskCreate() 函数原型在 los_task.c 文件中定义。调用 LOS_TaskCreate() 创建一个任务以后，任务就会进入就绪状态。

此处用 DHT11 读取温湿度数据显示在屏幕任务作为例子，以下是定义该 task 和创建 task 的核心代码：

```

1 VOID DHT11_task(void)
2 {
3     UINT32 uwRet = LOS_OK;
4     char cStr1[10];
5     char cStr2[10];
6     while(1)
7     {
8         if(DHT11_Read_TempAndHumidity(&DHT11_Data)==SUCCESS)
9         {
10             sprintf(cStr1,"%d.%d C",DHT11_Data.temp_int,DHT11_Data.temp_deci);
11             sprintf(cStr2,"%d.%d %%RH",DHT11_Data.humi_int,DHT11_Data.humi_deci);
12             ILI9341_DispString_EN ( 160, 110, cStr1, macWHITE, macRED );
13             ILI9341_DispString_EN ( 140, 130, cStr2, macWHITE, macRED );
14             printf("\r\n ??DHT11??,??%d.%d C,?? %d.%d %%RH",\
15                 DHT11_Data.temp_int,DHT11_Data.temp_deci,DHT11_Data.humi_int,DHT11_Data.
16                 humi_deci);
17         }
18     else
19     {
20         printf("DHT11 ERROR!\r\n");
21     uwRet = LOS_TaskDelay(2000);
22     if(uwRet !=LOS_OK)
23         return;
24   }
25 }
26
27 UINT32 creat_DHT11_task(void)
28 {
29     UINT32 uwRet = LOS_OK;
30     TSK_INIT_PARAM_S task_init_param;
31     task_init_param.usTaskPrio = 0;
32     task_init_param.pcName = "DHT11_task";
33     task_init_param.pfnTaskEntry = (TSK_ENTRY_FUNC)DHT11_task;
34     task_init_param.uwStackSize = LOSCFG_BASE_CORE_TSK_DEFAULT_STACK_SIZE;
35     task_init_param.uwResved = LOS_TASK_STATUS_DETACHED;
36     uwRet = LOS_TaskCreate(&g_TestTskHandle,&task_init_param);
37     if(uwRet !=LOS_OK)
38     {
39         return uwRet;

```

```
40     }
41     return uwRet;
42 }
```

4.5.3.4 配置加载文件

最后一步，我们还需要修改分散加载文件 sct。由于 Huawei LiteOS 中对数据和代码位置进行了控制，代码和数据会放在多个不同的内存区域，因此需要使用分散加载文件进行描述，要是系统准确运行起来，需要重新编写一个分散加载文件，配置 MDK 的链接器选择指定的分散加载文件。在正确配置 sct 文件以后，就可以下载到板子上验证功能，经过 load 到板子以后，我们的开发板可以在 LED 灯闪烁的同时，在屏幕上显示 DHT11 温湿度数据，证明 Huawei Lite OS 移植成功，且支持多任务。

4.6 俄罗斯方块

首先，一个完整的俄罗斯方块游戏需要实现以下几个功能：

1. 游戏开始结束界面

- (a) **游戏开始的界面**，在此界面中选择游戏难度（不同的游戏难度对应不同的速度），并提示如何开始。
- (b) **游戏结束的界面**，在此界面中会统计使用者本轮游戏的得分，并提示如何重新开始游戏。

2. 游戏主界面

- (a) **游戏界面**，包括不同方块的显示、产生，变形，移动（包括左右移动、手动向下移动、自动向下移动）的显示。
- (b) **信息界面**，包括得分以及难度。
- (c) **分割线**，分割游戏界面与信息界面。

3. 关键逻辑

- (a) 方块的堆叠。
- (b) 方块的填满行时需要将填满的行删除，同时累加分数。

4.6.1 游戏开始结束界面

如下图所示，在主函数中分别设计开始结束界面，其中主要使用 ILI9341_DispStringLine_EN_CH 函数在 LCD 显示屏的不同行显示不同信息，其中 level 和 score 是程序维护的两个全局变量，其中 level 表示游戏难度可以为 1, 2, 3，对应不同的方块下落速度，score 对应本轮游戏的得分，每当出现行的消除，就相应在 score 变量中累加一定的分数。

程序首先绘制开始界面，按下开始按钮后会进入主界面，主界面分为游戏界面、信息界面、分界线。process 是游戏界面的主进程，在分界线和信息界面绘制完毕后，会修改全局变量 start，以此触发游戏界面 process() 开始工作，当满足游戏结束条件时，状态变量 end 置 1，退出 process() 和其所在的 while 循环，以此进入结束界面。界面截图如图所示。

```

1 int main(void) {
2     /* 前方配置代码此处省略 */
3     ILI9341_DispStringLine_EN_CH(LINE(7), "    欢迎来到俄罗斯方块!");
4     ILI9341_DispStringLine_EN_CH(LINE(9), "    请按START键开始游戏");
5     ILI9341_DispStringLine_EN_CH(LINE(11), "    请按LEVEL键切换游戏难度");
6     sprintf(dispBuff3, "    当前游戏难度: %d", level);
7     ILI9341_DispStringLine_EN_CH(LINE(13), dispBuff3);
8     while(1) {
9         if (start)
10             process();
11         if (end==1) break;
12     }
13     LCD_SetColors(WHITE, BLACK);
14     ILI9341_Clear(0, 0, LCD_X_LENGTH, LCD_Y_LENGTH); /* 清屏, 显示全黑 */
15     ILI9341_DispStringLine_EN_CH(LINE(7), "    游戏结束!");
16     sprintf(dispBuff3, "    你本次的得分为: %d", score);
17     ILI9341_DispStringLine_EN_CH(LINE(9), dispBuff3);
18     ILI9341_DispStringLine_EN_CH(LINE(11), "    请按RESET键重新开始游戏");
19     return 0;
20 }

```

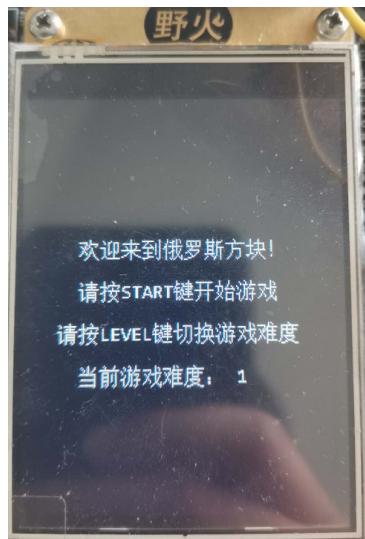


图 4-9 开始界面

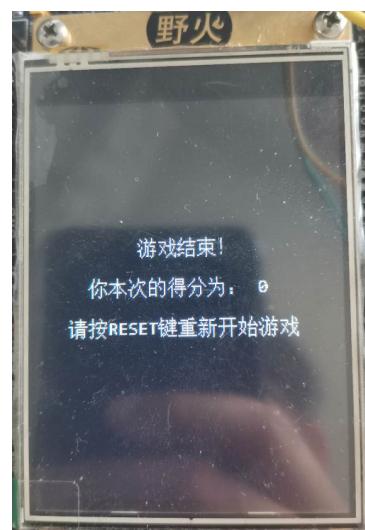


图 4-10 结束界面

4.6.2 游戏主界面

4.6.2.1 方块的显示、变形、移动、产生、速度

如下图所示，共设计 19 种不同形状的俄罗斯方块，并且将其分为 7 类，每一类使用不同的颜色。使用野火例程中的 ILI9341_DrawRectangle 函数绘制每一个正方形并填充颜色，然后若干个正方形便构成一种俄罗斯方块。使用按键中断进行变形和移动，维护全局变量 x、y、state，(x,y) 用于描述当前方块的位置，state 用于描述当前方块是哪种形状 (state 的取值为 1 到 19)。

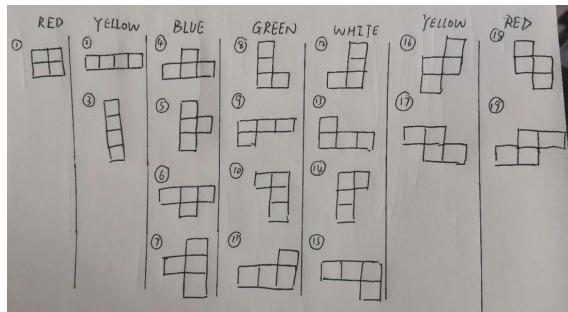


图 4-10 方块种类

通过按键中断可以改变这三个变量，当按下左移右移按钮时，可以实现 $x-=step$ 与 $x+=step$ ($step$ 为移动的步长，设置为一个方块的长度)，当按下下移按钮时，可以实现 $y+=step$ 。 x 不能超过游戏界面的左右边界，同时 y 不能超过游戏界面的下边界（不同的方块不超过右边界和下边界条件不同，需要分别给出）（此外其它方块的阻挡逻辑会在关键逻辑部分叙述）。下列代码为左移的基本逻辑，先删除原来的方块，同时在满足边界条件时 $x-=step$ ，并相应再次绘制图形，其它移动方法类似：

```

1 del_type(x,y,state);
2 if (x-step>=0 && ILI9341_GetPointPixel(x-step,y)==0)
3     x-=step;
4 draw_type(x,y,state);

```

当按下变形按钮时， $state$ 自增 1 同时检查是否越过该类的边界阈值，若越过则会跳（比如当时 $state$ 为 7，按下变形按钮后为 8，此时不属于第三类了，则 $state$ 置 4）。而后方块形状改变到 $state$ 对应的形状。方块的产生使用随机数随机生成一个 1-19 之间的值赋给状态变量，每产生一个方块，就初始化全局变量 x,y ，将其定位在上边界的中间位置，产生的方块自动下移直到到达下边界为止，下移速度与游戏难度有关。

```

1 del_type(x,y,state);
2 state+=1;
3 switch(state)
4 {
5     case 2:state=1;break;
6     case 4:state=2;break;
7     case 8:state=4;break;
8     case 12:state=8;break;
9     case 16:state=12;break;
10    case 18:state=16;break;
11    default:;
12 }
13 draw_type(x,y,state);

```

4.6.2.2 分割线与信息界面

主界面中，还需绘制一条分界线来分割游戏界面和信息界面，绘制动作在按下开始游戏按钮后触发，同时会修改全局变量 $start$ ，以此触发游戏界面开始工作。

信息界面显示当前的游戏难度和游戏分数，每次删除填满的行时会触发信息界面的修改，即修改当前的游戏分数。

4.6.3 关键逻辑

4.6.3.1 方块堆叠

对于不同种类的方块，方块下移的时候需要检查下方是否存在方块，若存在方块则退出本个方块的处理循环，此时方块停止移动并驻留在屏幕上。而后产生一个新的方块，并开启一个新方块的处理循环。实现的代码如下所示，主要使用 `ILI9341_GetPointPixel` 函数检查该俄罗斯方块的下边界是否存在其它方块。

```

1 while(1)
2 {
3     state = (rand()%19)+1;
4     x=80;
5     y=0;
6     while(1)
7     {
8         //printf("%d\n",y);
9         draw_type(x,y,state);
10
11        flag=1;
12
13        switch(state)
14        {
15            case 1:
16                if (ILI9341_GetPointPixel (x,y+20)!=0 || ILI9341_GetPointPixel (x+10,y
17                    +20)!=0) {flag =0; clear_all();} break;
18            case 2:
19                if (ILI9341_GetPointPixel (x,y+10)!=0 || ILI9341_GetPointPixel (x+10,y
20                    +10)!=0 || ILI9341_GetPointPixel (x+20,y+10)!=0 ||
21                    ILI9341_GetPointPixel (x+30,y+10)!=0)
22                {flag =0; clear_all();}
23                break;
24            case 3:
25                if (ILI9341_GetPointPixel (x,y+40)!=0) {flag =0; clear_all();}break;
26            case 4:
27                if (ILI9341_GetPointPixel (x,y+20)!=0 || ILI9341_GetPointPixel (x+10,y
28                    +20)!=0 || ILI9341_GetPointPixel (x+20,y+20)!=0) {flag =0; clear_all()
29                    ();}break;
30            case 5:
31                if (ILI9341_GetPointPixel (x,y+30)!=0 || ILI9341_GetPointPixel (x+10,y
32                    +20)!=0) {flag =0; clear_all();} break;
33            case 6:
34                if (ILI9341_GetPointPixel (x,y+10)!=0 || ILI9341_GetPointPixel (x+10,y
35                    +20)!=0 || ILI9341_GetPointPixel (x+20,y+10)!=0) {flag =0; clear_all()
36                    ();}break;
37
38        /* case7-19的情况类似，此处省略 */
39        default:
40    }
41    if (end==1) break;
42 }
```

4.6.3.2 方块清除

方块的填满行时需要将填满的行删除，同时累加分数。该过程使用 clear_all 函数实现，实现时需要调用 prejudge 函数和 clear 函数。

由于最长的一种方块是由四个正方形组成的长方形，因而一次最多消去行，由于 x,y 给出了当前方块的定位，因而只要从当前行向下检查 4 行，而后如若改行满，则标志 clear_status 置 1，而后清除 4 行中标志位为 1 的行，统计标志位为 1 的行数，并且相应累加得分。

```

1 void clear_all(void)
2 {
3     uint8_t i;
4     uint8_t count=0;
5     for (i=0;i<4;i++)
6         if ((y+10*i)<=310 && prejudge(y+10*i)) {clear_status[i]=1;}
7
8     for (i=0;i<4;i++)
9         if (clear_status[i]==1) {clear(y+i*10);count++;}
10
11    switch(count)
12    {
13        case 1:score+=10;break;
14        case 2:score+=20;break;
15        case 3:score+=40;break;
16        case 4:score+=80;break;
17    }
18
19    LCD_SetTextColor(WHITE);
20    if (count>0)
21    {
22        LCD_ClearLine(LINE(6));
23        sprintf(dispBuff, "%d", score);
24        ILI9341_DispStringLine_EN_CH(LINE(6), dispBuff);
25        ILI9341_DrawLine ( 190, 0, 190, 320 );
26    }
27
28    for (i=0;i<4;i++)
29        clear_status[i]=0;
30 }
```

prejudge 和 clear 函数的实现细节如下，prejudge 会以 step 为步长，检查该行所有方格内是否存在像素，若都存在像素，则说明该行满，返回 1，否则说明有的位置没有方块，该行不满，返回 0。

```

1 uint8_t prejudge(uint16_t y)
2 {
3     uint16_t i=0;
4     uint16_t tmp_x=0;
5     uint8_t flag=1;
6
7     //printf("this is a prejudge test\n");
8     for (i=0;i<19;i++)
9     {
10         //printf("%d:%d\n",i,ILI9341_GetPointPixel (tmp_x,y));
11         if (ILI9341_GetPointPixel (tmp_x+10*i,y)==0)
12         {
13             flag=0;
14             break;
15         }
16     }
17 }
```

```
18
19     if (flag==0)
20         return 0;
21     else
22         return 1;
23 }
```

clear 函数会将某行上方区域的所有方块下移一格，实现方式为先取上方块的像素，再用该像素覆盖本块，以消除（覆盖）该行，注意到界面最上面一行无法再取上一行，因而直接将其填充为黑色即可。

```
1 void clear(uint16_t y)
2 {
3     uint16_t i;
4     uint16_t j;
5     for (i=y;i>=10;i-=10)
6     {
7         for (j=0;j<190;j+=10)
8         {
9             LCD_SetTextColor(ILI9341_GetPointPixel (j,i-step));
10            ILI9341_DrawRectangle (j,i);
11        }
12        for (j=0;j<190;j+=10)
13        {
14            LCD_SetTextColor(BLACK);
15            ILI9341_DrawRectangle (j,0);
16        }
17 }
```

五 成果展示

5.1 指纹识别认证

指纹识别功能模块由 stm32、DAP 仿真器、ATK-AS608 高性能光学指纹识别模块以及相应的上位机构成。模块实物图如5-1所示。本指纹识别模块具有指纹录入与识别的功能，通过实际测试，指纹的识别准确率和速度都有较好的表现，符合实际功能的需要。用户可以通过录入指纹并与其他模块结合实现相应操作。通过指纹的匹配与搜索，模块可以判断并识别此智能家居系统的使用者，从而一键认证用户并实现智能家居系统其他功能。

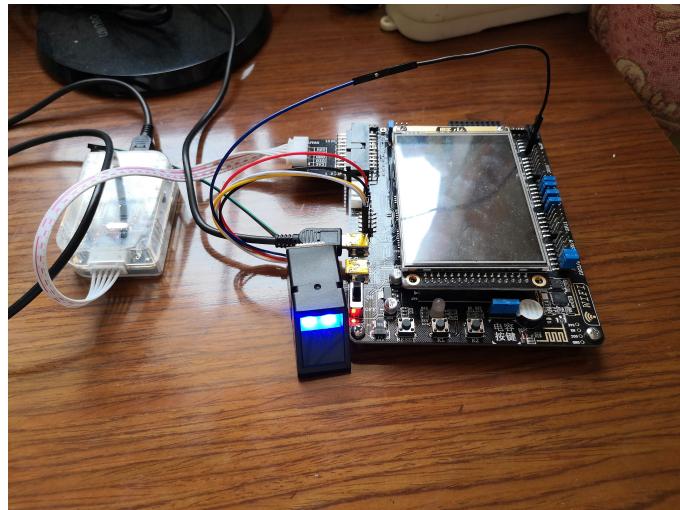


图 5-1 指纹识别模块实物图

本模块的主要功能为录入指纹并识别，可以存储指纹 300 枚，指纹搜索时间在 0.3s 内，具有较优秀功能性和性能。主要应用场景为人员识别和解锁家居系统。详细的功能展示参见演示视频。

5.1.1 串口模式

将模块的 Tx、Rx 连接到 USB 转串口设备，运行 CH340 驱动，在计算机设备管理器中可以发现对应的串口号为 COM4，如5-2所示

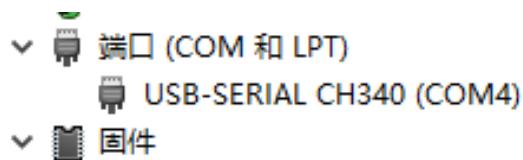


图 5-2 串口模式

5.1.2 运行上位机

打开相应的上位机 SYdemo.exe，选择对应的串口号并连接，会返回连接成功的信息、目前硬件的信息以及 ATK-AS608 中已经存储的数据。



图 5-3 上位机连接

5.1.3 录入指纹

选择“录入指纹功能”，并按照提示将待录入指纹的手指按压在模块的录入区，经过指纹录入、图像上传、综合比对等步骤后，上传的指纹图像在上位机中也可以预览。指纹录入成功并存入指纹库，每一枚指纹都有唯一对应的 FingerID，用于指纹的匹配和搜索等。截图如5-4所示



图 5-4 指纹录入

5.1.4 目标指纹匹配

指纹的匹配有两种工作模式，第一种为通过 FingerID 指定指纹与上传的指纹做比对，判断是否相同，如5-5所示，可用于解锁相关物联网终端系统。



图 5-5 单一指纹匹配

第二种为上传指纹并在指纹库里搜索相应的 FingerID，若指纹库里没有相应的指纹信息，则会返回错误信息，如5-6(a)、5-6(b)所示，可用于人员的识别与判断。



图 5-6 目标指纹匹配

5.2 疫情新闻

本部分实现的功能为上位机获取疫情最新新闻，周期性地通过串口将新闻发送给 stm32，并相应在 stm32 中查看。

```
已发送信号：苏里南大选将如期举行
已发送信号：联合国各国代表“云聚”
已发送信号：近日，马来西亚出现有
已发送信号：日本总务省22日公布
已发送信号：根据澳大利亚新南威尔
```

图 5-8 数据传输



图 5-9 传输更新前的新闻条目



图 5-10 传输更新后的新闻条目

5.3 室内温控

本部分实现的功能为室内测温并显示于屏幕，同时当温度达到阈值 1，自动打开风扇降温，低于阈值 1 风扇自动关闭，当温度达到阈值 2，自动报警，低于阈值 2，报警蜂鸣器关闭。可以使用按键调节阈值。

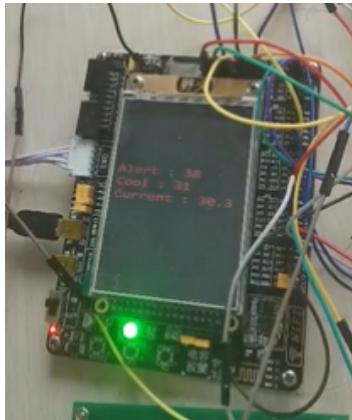


图 5-11 温控显示界面



图 5-12 超过阈值 1 风扇转动

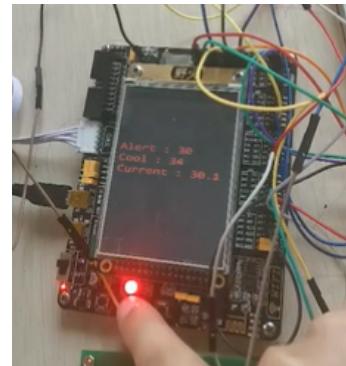


图 5-13 超过阈值 2 报警

5.4 无线物联网

本部分实现的功能为通过手机 app 与 stm32 进行无线通信，控制 stm32 上的小灯（三种颜色的 led+ 外接 led）、报警（无源蜂鸣器）、音乐（有源蜂鸣器）、风扇，此外，还可以读取室内温湿度数据并发送到手机。



图 5-14 物联网显示界面



图 5-15 以控制 LED 灯为例

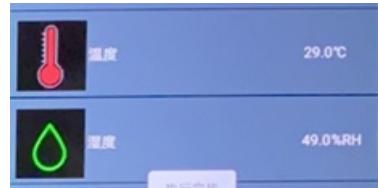


图 5-16 手机获取温湿度

5.5 俄罗斯方块

本部分实现的功能为俄罗斯方块游戏。



图 5-17 开始界面



图 5-18 游戏界面

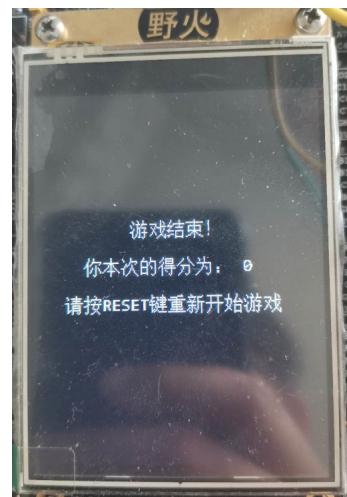


图 5-19 结束界面

六 项目总结

6.1 项目评价

6.1.1 项目特点

本项目基于全球新冠疫情居家隔离的背景，利用 STM32F103 芯片及其外围硬件模块，提出、设计并完成了一个功能较为完善的疫情背景下的物联网智能家居嵌入式系统。该系统能完成以下功能：

1. **疫情新闻模块：**及时甄选出疫情消息，让人们及时了解到抗疫进展，缓和民众对疫情紧张或恐惧的心理。
2. **音乐播放模块、游戏娱乐模块：**提供一些娱乐手段，缓解人们居家时的闭塞无聊。
3. **无线物联网模块、智能指纹锁模块、室内温湿度测量模块、智能风扇电灯模块：**提供更舒适、方便的居家环境，缓解人们居家时的焦躁，使人们爱上居家。
4. **嵌入式操作系统模块：**实现具有一定并行性的系统控制，提高用户体验和系统性能。

因此，我们的系统能够应对疫情时期人们长时间居家隔离而造成的孤独闭塞与枯燥无味，为疫情时期居家隔离中的人们提供一个方便、舒适、健康、智能、趣味的居家环境，减小疫情时的长期居家隔离对个人、对社会的负面影响。

6.1.2 改进方向

由于新冠疫情，小组成员不能聚在一起进行工作。虽然定期进行远程讨论，但由于设备的分散，小组成员只能独立进行自己模块的开发。这导致了成员之间互相不太熟悉对方的工作，难以互相帮助；遥远的距离也让最后模块的整合变得困难。加之以有限的经费与时间，这使我们的项目还存在以下不足：

1. 虽然项目整体比较复杂，但单个模块功能过于简单，不能达到实际使用的要求。
2. 各模块整合效果不佳，未实现不同应用模块间的相互自动控制，未形成一个统一的整体。
3. 不够智能，需要用户手动控制。
4. 作为物联网智能家居系统来说，功能较少，难以承担智能家居的所有需求。

因此，针对以上不足，我们希望之后在以下方向上进行改进：

1. **深入完善各模块功能**，例如智能指纹锁模块可以深入设计成集人脸识别、指纹识别、声纹识别、密码锁等多种门锁方式于一体的智能门锁系统，使用户能通过多种途径进行开锁，为用户提供一个更安全而方便的门锁系统。
2. 依据各应用模块的功能间的关系对**各应用模块进行整合**，使其能相互通信，相互控制。例如室内温湿度模块可以与智能风扇模块结合，形成一个负反馈系统，自动维持室内温湿度，为用户提供一个恒定舒适的环境；人体测温模块可与智能门锁模块结合，对来访人员进行筛查，阻止病毒的侵入。

3. 依据各模块的整合，与操作系统的总体控制，形成一个**自动运行、自动控制、自动适应**的智能家居系统。
4. **为系统加入新的应用模块**，如室内亮度感知与调节模块、智能水温模块等。

相信，在我们之后的继续完善下，我们的系统一定能成为一个实用的物联网智能家居嵌入式系统。



图 6-1 物联网智能家居嵌入式系统

6.2 心得体会

一、个人工作

我在项目中主要负责了部分模块的选择与采购，DHT11 温湿度传感器模块和 ESO8266 无线模块的资料收集与调试，参与无线物联网功能的设计与调试，移植 Huawei lite OS 实现多任务并行，项目功能整合进操作系统（仅部分），在项目报告中负责了无线物联网与 Huawei lite OS 部分软硬件的撰写。

二、心得体会

本学期由于疫情原因课程通过网络进行，又因客观条件限制难以进行协同开发，板子和模块之间的小组成员协同操作比较困难，因此也只能尽力将自己所负责的功能模块和报告做好，还是要感谢各位小组成员的包容理解和辛勤付出。在进行嵌入式开发板调试的时候，让我更加深入理解嵌入式原理，没有动手之前觉得老师上课讲的内容比较难，有些听不懂，毕竟以前没有接触过。但是发现课后的搜索资料、购买模块、亲自动手编写项目，设 GPIO，调中断，每一次尝试就离成功越进一步，发现渐渐的就理解了老师上课讲的内容。而且，借此疫情，也锻炼了自己单独开发的能力，线上在家里，遇到了难以解决的问题难以直接向同学老师询问，需要自己花时间钻研出来。并且在调节温湿度传感器和 ESP8266 模块时，偶然发生淘宝野火旗舰店的客服技术支持非常厉害，向他询问具体模块的问题，讲解的非常透彻。就课程内容而言，老师在课程中讲授了嵌入式系统的原理和相关概念，我们也通过野火开发板进行了实际的项目开发。这其中许多知识感觉还是很实用的，对软硬件的交互以及系统的设

计开发有了更深刻的理解。至于相关的建议的话，其实开发过程中也体会到，STM32 板子的问题是性能不是很好，有时限制了项目的实现，同时有些模块的支持也不完整，给项目带来了一些困难。因此感觉其实可以考虑树莓派等其他平台的开发，或者对嵌入式设备不作太多限制，让同学们选择喜欢的设备。

致 谢

感谢周老师一直以来的指导教诲与帮助支持！

感谢各位小组成员的通力合作与辛勤付出！

感谢其他同学的交流讨论与意见建议！