

## 摘 要

鉴于国外芯片设计企业可能采取的技术垄断，加之疫情让人们对健康生活更加重视，本课题设计了一款基于 RISC-V 架构 MCU 的穿戴式人体健康监测装置，可实时检测佩戴者的心率和血氧饱和度、皮肤表面温度以及运动状态等健康参数。经过方案比对，最终选择乐鑫科技的 ESP32-C3 系列作为主控芯片，使用 MAX30102 传感器获取光电容积脉搏波（PPG）并计算心率、血氧饱和度(SpO2)数据，通过 MAX30205 获取实时温度、MPU-6050 惯性测量单元(IMU)获取运动状态数据，从而实现久坐、跌倒提醒等功能。

用户可通过基于 LVGL 设计的 GUI 界面进行人机交互、数据查看，还可将装置连接至 Wi-Fi 热点，数据会通过 MQTT 协议上传至阿里云 IOT 服务器。借由阿里数据可视化平台，用户端可使用配套 APP 查看历史数据、跌倒警报信息等。

装置设计完成后，通过各项功能测试，发现存在设计缺陷并进行了改正及优化，最终实现了装置的稳定与数据的准确可靠。

**关键词：**RISC-V；心率计算；血氧饱和度；接触式温度测量；跌倒检测

## Abstract

In view of foreign chip design enterprises may take the technology monopoly, coupled with the COVID-19 has made people pay more attention to a healthy life. This topic designed a wearable human health monitoring device based on RISC-V architecture MCU, which can detect the wearer's heart rate, blood oxygen saturation, skin surface temperature, and exercise status in real-time. After scheme comparison, the ESP32-C3 series of Espressif Systems CO., LTD. was selected as the master chip. The MAX30102 sensor was used to obtain photoplethysmogram(PPG) signals, which can calculate heart rate and blood oxygen saturation( $\text{SpO}_2$ ) data. MAX30205 was used to obtain real-time temperature, and MPU-6050 Inertial measurement unit(IMU) was used to obtain motion state data, so as to realize the functions of sedentary and falling reminder.

A GUI interface based on LVGL allows users to interact with the device and view data, if connected to a Wi-Fi hotspot, it will upload data to the AliCloud IOT server via MQTT. Through the AliCloud data visualization platform, users can view historical data and fall alarm information by using the supporting APP.

After finishing the design, the device defects were found and optimized through various functional tests. Finally, good stability of the device and accuracy of data was realized.

**Key words:** RISC-V ; heart rate calculation ; blood oxygen saturation( $\text{SpO}_2$ ) ; contact thermometry ; fall detection

# 目 录

<b>摘 要 .....</b>	I
<b>Abstract .....</b>	II
<b>第1章 绪论 .....</b>	1
1.1 课题研究背景和意义 .....	1
1.2 国内外研究现状 .....	1
1.3 课题主要研究内容 .....	3
1.4 文章结构安排 .....	3
<b>第2章 系统架构与技术路线 .....</b>	5
2.1 系统目标 .....	5
2.2 系统架构 .....	5
2.2.1 系统架构方案 1 .....	5
2.2.2 系统架构方案 2 .....	5
2.2.3 系统架构方案对比 .....	6
2.3 无线通信技术对比 .....	7
2.4 主控芯片选型 .....	7
2.5 云服务器与应用层通信协议 .....	8
2.6 用户界面开发技术 .....	9
2.7 系统组成介绍 .....	10
2.7.1 系统架构图 .....	10
2.7.2 装置主控板 .....	10
2.7.3 云服务器 .....	10
2.7.4 用户显示界面 .....	10
2.8 本章小结 .....	11
<b>第3章 系统硬件电路设计 .....</b>	12
3.1 装置主控板 .....	12
3.1.1 主控芯片模块电路 .....	12
3.1.2 串口调试及下载电路设计 .....	16

---

3.1.3 供源及充电管理电路设计.....	16
3.1.4 PPG 传感器电路设计 .....	17
3.1.5 温度传感器电路设计.....	18
3.1.6 姿态传感器电路设计.....	19
3.2 屏幕底板 .....	21
3.3 本章小结 .....	21
<b>第 4 章 系统软件设计 .....</b>	<b>23</b>
4.1 装置主控板 .....	23
4.1.1 装置软件整体设计.....	23
4.1.2 温度获取.....	24
4.1.3 计步测距与久坐提醒.....	26
4.1.4 跌倒检测算法.....	27
4.1.5 心率与血氧检测算法.....	28
4.1.6 屏幕交互逻辑设计.....	31
4.1.7 数据上传云服务器.....	31
4.2 服务器数据处理 .....	32
4.2.1 处理物模型数据.....	32
4.2.2 业务逻辑设计.....	33
4.3 用户 APP 设计 .....	33
4.3.1 页面交互设计.....	33
4.3.2 跌倒警报推送.....	34
4.4 本章小结 .....	35
<b>第 5 章 系统测试与分析 .....</b>	<b>36</b>
5.1 测试目的 .....	36
5.2 心率与血氧功能测试 .....	36
5.3 计步测距功能测试 .....	37
5.4 温度功能测试 .....	37
5.5 跌倒警报测试 .....	38
5.6 本章小结 .....	39

总结与展望 .....	40
6.1 总结 .....	40
6.2 展望 .....	40
参考文献 .....	42
致谢 .....	44
附录 A 电路原理图 .....	45
附录 B PCB 图 .....	48
附录 C 实物图 .....	49
附录 D 源程序 .....	51

# 第1章 绪论

## 1.1 课题研究背景和意义

课题方向源于当前半导体形势环境，大部分中低端乃至高端 MCU（Microcontroller Unit，微控制单元）被英国 ARM 公司的内核/指令集技术垄断，而同时名为 RISC-V 的开源指令集架构处于兴起时期，全球范围内产出的 RISC-V 核累计超过 20 亿颗<sup>[1]</sup>。即使受到技术制裁，中国的 MCU 乃至高端处理器市场也能借助 RISC-V，设计出高性能、低成本的处理器，从而实现我国在这块短板的突破<sup>[2]</sup>。此外，国际半导体、芯片制造业整体受疫情影响，产品价格居高不下，而 RISC-V 内核的模块化和高可复用性降低了设计成本，在当前环境下有较大优势<sup>[3]</sup>。因此，使用该架构的 MCU 既有助于支持国内企业技术发展、探索新技术路线，也能降低最终产品的成本价格。

课题的目标产品启发于疫情影响下，人们对于身体健康和科学生活的重视程度相比以往有较大幅度的提高，相关的人体健康监测设备种类增加、需求量也大幅度提升<sup>[4]</sup>。其中，拥有可穿戴便携性、体征监测及反馈等功能的智能设备均可称为可穿戴健康监测设备。根据著名市场研究机构 IDTechEx 的报告，2020 年全球可穿戴设备市场规模接近 800 亿美元，为 2014 年的三倍<sup>[5]</sup>。且根据 IDC 2021 年 Q3 季度已有的全球可穿戴设备出货量数据显示，该季度全球此类设备出货量同比增长 9.9%，达 1.384 亿台，维持了强劲增长的走势，可见这类设备的需求量、发展潜力是不断增长的，因此该课题具有较大的实际应用价值。

课题拟探究健康监测设备的技术原理，并使用 RISC-V 架构的 MCU 将其实现，以期降低成本，并为国产 MCU 技术落地和开源社区事业的发展贡献一份力量，这在当前国产芯片技术受制、全球芯片价格上涨的背景下是具有开拓性意义的。

## 1.2 国内外研究现状

本课题所采用的 RISC-V（Reduced Instruction Set Computer-FIVE，第五代精简指令集）是一种开源指令集架构，与 ARM 同属于精简指令集，其初版于 2010 年由美国加州大学伯克利分校公布<sup>[6]</sup>。其诞生是为了解决现有指令集架构（如 X86、MIPS、ARM 等）的封闭或版权费用高昂等问题，其具有开源、免费、开放、自由的特点——类似开源操作系统 Linux，开源的发展模式极大的推动了相关技术的推广和进步。经过短短十年的迅速发展，RISC-V 已有实力与 X86、ARM 竞争，形成了“三足鼎立”的局面。由于其指令集已经较为完善，并且根据适用场景的不同，划分了 5 种基础指令集(RVWMO、RV32I、

RV64I、RV32E、RV128I) 以及 24 种扩展指令集(乘除法及浮点扩展等)<sup>[7]</sup>, 其中不乏有应用于嵌入式领域的方案。因此借助 RISC-V 指令集, 可以设计出从中低端 MCU 到高端 FPGA 核心、甚至是高性能计算处理器, 且由于其开源的性质, 即使在受技术制裁的当下, 中国的优秀芯片设计企业, 仍能借此设计出拥有自主产权的高性能处理器, 弥补国内在这块领域起步晚、有短板的现状。

近年来, 对于 RISC-V 的各种研究已实现从理论到商用产品的蜕变, 各大企业基于该内核设计的各类产品也层出不穷。对于 MCU 产品, 国内企业兆易创新已于 2019 年推出全球首款 RISC-V 内核的通用 MCU 系列<sup>[8]</sup>, 而此后诸如乐鑫科技、沁恒微电子、中科蓝讯、平头哥半导体等企业都纷纷推出了各自领域适用的 RISC-V 内核芯片产品。此外, 国内外企业如芯来科技、晶芯科技、SiFive 等, 提供付费的内核设计方案, 许多小型企业凭借现有的设计, 修改后完成了芯片的自有化。

自疫情后, 人们对于个人身体健康的重视程度有所提高, 同时随着物联网的兴起, 各种可穿戴的智能健康设备层出不穷, 诸如健康手环、智能手表、便携血糖血压计、穿戴式心电贴(记录仪)等产品的市场需求量飞速提升<sup>[9]</sup>。目前市场上出货量较大的有小米、华为等厂商推出的系列智能手环或手表, 其拥有最基本的人体健康体征监测功能、数据分析反馈功能、人性化的人机交互、合适的售价等, 这些特点使得该类产品广受消费者青睐。这类产品常见的健康监测功能一般包括心率计数及血氧饱和度测量、计步与耗能换算、体温测量等<sup>[10]</sup>, 但在功能上普遍没有将多传感器数据结合或扩展、使之更贴近于生活使用。此外这类产品多数存在着识别不准、体积较大、亲肤性差(材质硬)、隐私数据泄露等问题, 而针对这些问题, 新一代的可穿戴产品正朝着小型化、柔性化的方向发展, 且随着嵌入式 AI 技术和物联网技术的发展, 可穿戴式健康设备将变得更加智能<sup>[10]</sup>。

此外, 几乎所有穿戴式健康设备都是使用 ARM 架构的 MCU 作为主控, 其好处是高性能且开发技术路线成熟, 但在当下环境中这无疑是增加了生产成本, 也使得产品的可拓展性受到制约。首先打破这一局面的是华米科技, 其推出了首款自研的 RISC-V 架构可穿戴领域 AI 处理器“黄山 1 号”<sup>[11]</sup>, 并基于此芯片推出了相关产品。不过可惜的是该芯片不对外销售, 意味着最终产品的价格受自研芯片的成本影响较大, 使得成品价格居高不下, 且没有激励其他可穿戴健康设备企业在该领域与之竞争, 但倘若华米继续延续开源精髓, 将芯片对外销售以降低自研成本, 市场上也许会是另一番景象, 越来越多厂商会加入其中。

### 1.3 课题主要研究内容

本课题主要研究如何基于 RISC-V 架构的 MCU 设计制作一个便携的穿戴式人体健康检测装置，其可无创地获取佩戴者的心率血氧、温度、姿态等信息，数据经过处理可以分析用户是否久坐、跌倒等。为便于用户查看，装置可将数据实时显示于屏幕、亦可通过网络上传至用户 APP 以便查看历史记录。

整个装置系统若完成既定的目标功能，需分为四个主要部分：主控采集电路板、屏幕及其底板、云服务器、用户交互界面，因此本文的研究工作如下：

- (1) 设计并制作主控采集电路板，通过传感器获取佩戴者的心率血氧、温度及姿态数据，整合数据并分析佩戴者的运动步数及距离、久坐时间、是否跌倒等。
- (2) 驱动显示屏幕，并能通过交互查看实时数据。
- (3) 装置可连接无线网络，并接入广域网上传数据至服务器。数据经由服务器处理后可传输至任意用户联网设备。
- (4) 用户交互界面可查询装置是否在线，并显示传感器信息及历史数据，用户跌倒时可触发通知警报。
- (5) 针对各部分功能联调优化，确保系统稳定性以及数据的准确性。

### 1.4 文章结构安排

本文为该课题的设计说明，从课题背景说明到理论设计、直至电路和软件的实现都做了详细的阐述，因此根据递进关系将本文分为以下六个章节：

第1章，绪论——本章对课题的研究背景和目的做了详细阐述，并从当前形势的角度解释了本课题的意义；而后简述了国内外相关研究的状况，并讨论了本课题的主要研究内容及目标功能。

第2章，系统架构与技术路线——本章根据讨论的目标功能，分析可行的系统架构并讨论不同方案的优劣；针对不同的器件和技术方案进行对比和选型，并对关键技术路线和思路进行概括性的阐述。

第3章，系统硬件电路设计——本章针对装置主控板的各项电路设计进行了分模块的阐述，分别介绍了最小系统设计、串口通讯电路设计、供电及充电电路设计，以及心率、温度、姿态传感器的电路设计，并简述了屏幕底板的电路接口。

第4章，系统软件设计——本章从装置主控板、服务器数据处理、用户交互界面，三个主要板块阐述整套系统的软件设计思路和操作逻辑。

第5章，系统测试与分析——本章在完成硬软件设计后，对系统进行联调和功能测

试，并对测试数据进行分析和优化，进而确保系统稳定性和数据准确性。

第6章，总结与展望——总结本课题所做的工作，对目前存在的优缺点进行分析，并对可优化的工作内容提出展望。

## 第2章 系统架构与技术路线

### 2.1 系统目标

本装置运用了传感器技术、嵌入式系统技术、物联网技术，可以在较低成本的条件下实现可靠的穿戴式人体健康状态监测。装置具有以下特点且能实现目标功能：

- (1) 使用 RISC-V 主控，装置整体成本较低。
- (2) 获取使用者的当前心/脉率波形，计算并显示实时心率数值。
- (3) 检测、显示装置附近的温度，判断装置是否被穿戴，监测体温变化状况。
- (4) 记录、显示使用者当天的运动步数，并计算运动步距。
- (5) 结合传感器信息，实现久坐提醒、跌倒报警等功能。
- (6) 装置具有人性化的交互界面以及美观的数据显示界面。
- (7) 数据可通过网络上传到云平台，用户可进行数据查看和交互。

### 2.2 系统架构

#### 2.2.1 系统架构方案 1

根据目标功能需求，装置需通过无线方式连接到互联网，因此装置的整体架构有两种选择，其一为 MCU 外部挂载射频模块——即装置除了主控芯片及传感器外，仍需使用无线通信模块进行数据传输。其架构示意如下图 2.1 所示：

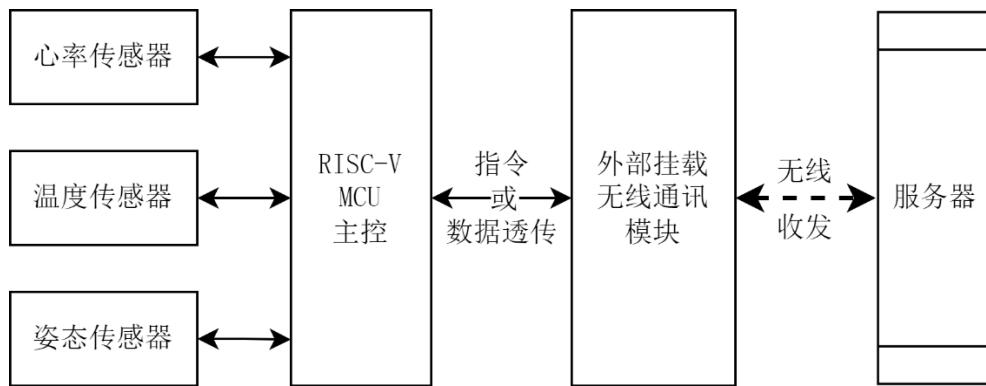


图 2.1 系统方案 1 架构图

#### 2.2.2 系统架构方案 2

第二种系统方案为 MCU 内部集成射频模块——即数据经由主控芯片处理后直接传递给内部无线射频单元，由其进行无线数据的收发。其架构示意如下图 2.2 所示：

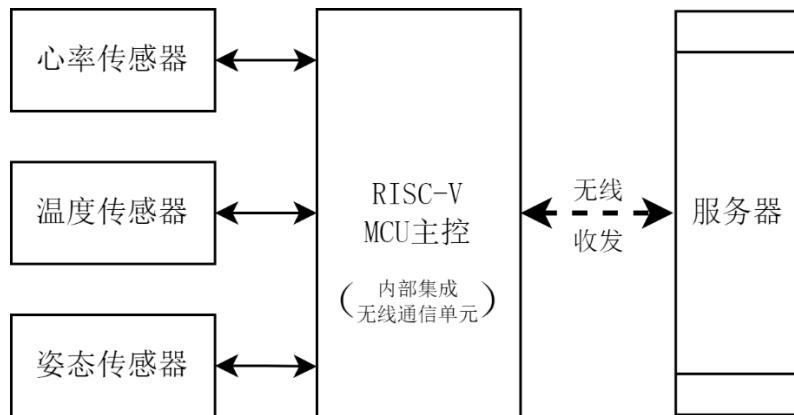


图 2.2 系统方案 2 架构图

### 2.2.3 系统架构方案对比

系统架构方案 1 是 MCU 外部挂载射频模块的方式，即选择一个芯片用作主控，挂载例如 ESP8266 或 NB-IOT 模块等作通信，此方案拓展性强、灵活且选择多。

系统架构方案 2 是使用内部集成射频通信单元的 MCU，即选择带有无线通信功能的主控，此方案不需要其他通信模块，集成度高且成本控制具有优势。

两种系统架构方案的整体对比如下表格所示：

表 2.1 架构方案对比

架构方案	集成度	整体成本	开发难度	稳定性	可拓展性
1	较低	较高	较低	较低	较高
2	较高	较低	较高	较高	较低

表 2.1 中的对比指标及其结果解释如下：

(1) 集成度：方案 1 由于外部挂载通信模块，硬件设计上既需要占用较多空间，还需要搭配外围器件，因而造成装置整体集成度较低；而方案二反之，仅需要搭配外部天线即可使 MCU 集成的射频正常工作，使得装置整体更小巧。

(2) 整体成本：普遍而言，选用带有射频单元的 MCU 会比另外购置无线通信模块的成本更低，但也跟具体选用的 MCU 和无线模块有关。

(3) 开发难度：就单片机程序开发难度而言，方案 1 通过透传进行无线通信，无需深入配置底层协议栈，因此开发难度较低；而方案 2 有可能需要经过复杂的配置从而实现通信协议，因此开发难度大、周期也较长。

(4) 稳定性：对于射频，高集成度的稳定性一般会比低集成度更好，但这也与具体选用的无线通信方式、模块单元设计相关，并不能一概而论。

(5) 可拓展性：由于方案 1 的耦合度较低，可选用的 MCU 方案更多、亦可有更多

的无线通信方式可供选择；而方案 2 已将 MCU 和无线通信方式绑定，因此可拓展性和可选择性大大降低。

综上所述，由于该装置需较多考虑减小体积和降低成本，因此更倾向于优先选择方案 2，但如若无适宜的集成无线通信 MCU 可供选择，则需考虑采取方案 1。

### 2.3 无线通信技术对比

装置的设计需考虑到可穿戴、便携性，因此向云服务器实时传输数据选择有线的方式显然不适合，且 IrDA、NFC 等近距离无线技术也并不适合，因此下表 2.2 对比该领域适合的几种无线通信技术。由于这些通信技术特性不同，因而在各自的领域各有优势，需要对比、分析后得到更优解<sup>[12]</sup>。

表 2.2 适合的无线通信技术对比

通信方式	NB-IOT	BLE	LoRa	Wi-Fi	ZigBee
主流频段	800~900MHz	2402~2480MHz	137~1050MHz	2.4GHz/5GHz	868M/915M/2.4GHz
整体功耗	低	较低	超低	高	低
通信距离	15km	50~800m	2km~15km	50~300m	100~1000m
理论速率	160~250kbps	1~2Mbps	292bps~5.4kbps	54~433Mbps	20~250kbps
应用方向	物联网 少量数据	局部快速 交换数据	超长距离 传感控制	联网高速数据 传输	低功耗、联网传输 少量数据
其他条件要求	需物联网卡 及购买流量	数据联网需要中 间设备传递	数据联网需要中 间设备传递	需接入热点源	需接入网关设备

由上表可知，选用 NB-IOT 可直接接入广域网，但需要购买物联网卡及流量，造成了成本和体积的增加，虽市面上存在 eSIM 类型模组，但其仅对企业大批量用户销售；虽 BLE 和 LoRa 有着较低的功耗，但需向中间设备传输数据转广域网才能上传至服务器，也造成了成本和开发难度的增加；而 Wi-Fi 可仅依靠用户手机热点即接入广域网、连接服务器，便携度高且开发难度不大；ZigBee 虽拥有非常低的功耗，但它仍需要网关设备才可接入互联网。

综合上表以及装置的需求考虑，ZigBee 理应成为首选，但许多集成射频的 MCU 并不支持该方式，而是对 Wi-Fi 的支持更为普遍，因此 Wi-Fi 方案也是可以接受的。

### 2.4 主控芯片选型

本课题需选择基于 RISC-V 架构的 MCU 主控芯片，虽市面上基于该开源架构的芯

片如雨后春笋般涌现，但许多厂商的产品较新，可能存在着性能不强、功能不多、稳定性差等缺点，因此需要对比市面上的成熟产品及其配置，得到更优选择。下表 2.3 列举了市面上适合本课题的 32 位 RISC-V 架构 MCU 的性能、配置及价格等信息。

表 2.3 适合的 RISC-V 架构 MCU 参数对比

主控系列	GD32VF103	CH32V305/7	CH581/2/3	BL702/4/6	ESP32-C3	AB32VG1
最高主频	108MHz	144MHz	80MHz	144MHz	160MHz	120MHz
运行内存	6~32KB	32~64KB	32KB	132KB	400KB	192KB
片上存储	16~128KB	128~256KB	248~1016KB	192KB	384KB (+4MB)	1MB
GPIO	37~80 个	17~80 个	20~40 个	15~31 个	16~22 个	22 个
射频单元	无	无	BLE 5.3	BLE 5.0 ZigBee	BLE 5.0 Wi-Fi 2.4	BLE (未开放)
单片价格	15~25 元	15~30 元	5~10 元	5~10 元	4~6 元	4.2 元
其他特点	芯来内核，编译链工具齐全	配套开发环境 MRS	配套开发环境 MRS	资料较少	配套框架 ESP-IDF	开发环境 RT-Thread Studio

综合前面的需求分析以及上表的信息，内部集成 Wi-Fi 或 ZigBee 射频单元的 BL702/4/6 系列或 ESP32-C3 系列应为较优选择。对比两者官方资料后，发现博流智能的 BL702/4/6 系列资料非常少、且处于样品实验阶段，相应的购买渠道也比较窄，因此本课题优先选择产品相对成熟的乐鑫科技 ESP32-C3 系列 MCU 作为装置主控，该芯片已经过三次修订版本，硬件功能较为稳定，同时其有着非常成熟的 ESP-IDF 配套开发框架以及详细的开发参考文档。

## 2.5 云服务器与应用层通信协议

云服务器优先选择技术成熟、可靠稳定的国内云服务商阿里云，而且相比于自己购置云主机、搭建物联网控制平台，使用阿里云现成的物联网平台更实惠、稳定，因此本课题的云服务器优先选用阿里云 IOT 平台。

根据阿里云 IOT 平台的开发文档，其支持 HTTPS、MQTT、CoAP 三种开放协议实现设的自主接入，其中前两者的传输层协议是 TCP、后者则为 UDP，有关协议层的两种协议对比如下表 2.4 所示。

表 2.4 传输层协议对比

协议类型	TCP	UDP
可靠性	可靠	不可靠
连接性	有连接	无连接
传输效率	低	高
可靠性	高	低

可见两种协议面向的使用场景不同，拥有截然相反的特性。TCP 适用于可靠、低速的数据交付；UDP 适用于非可靠、高速的数据交付场景。而对于本课题，鉴于只需要定时上传传感器参数，并不需要进行高速数据交换，所以倾向于选择稳定可靠的 TCP 协议。其中 MQTT 相比于 HTTPS，异步的消息通信相比于同步请求响应机制来讲更适合于物联网场景下的数据传输，该协议专为物联网场景设计了 QoS (Quality of Service, 服务质量) 等级、心跳机制、订阅发布机制等<sup>[13]</sup>，而且资源开销与响应及时性都是 MQTT 协议更优，因此本课题优先选择通过 MQTT 协议与阿里云 IOT 服务器通信。

## 2.6 用户界面开发技术

系统所使用的用户界面分为装置屏幕 GUI 和用户 APP 界面两部分，前者需考虑适合本项目使用的嵌入式 GUI 框架，下表 2.5 是常见嵌入式 GUI 框架的特性对比。

表 2.5 嵌入式 GUI 框架对比

GUI 框架	TouchGFX	emWin	LVGL	Qt for MCU
授权方式	收费	收费	开源、免费	非商用免费
限制性	仅 ST MCU 使用	无	无	无
特点	界面华丽 支持拖拽开发	资源占用小 提供芯片定制版	轻量、美观 纯代码开发	资源占用大 可自定义程度高

结合系统需求及上表特性综合考虑，由于 TouchGFX 和 emWin 都是收费授权，即便免费的版本也仅限于 ST 的 MCU 使用，所以不适合本课题。至于 Qt for MCU 框架，虽然美观、可自定义强，但资源占用极大。因此开源、免费，同时又轻量、美观的 LVGL 图形框架成为本项目的屏幕 GUI 框架首选。

对于用户 APP 界面部分，除了较为复杂的原生应用开发外，为了跨平台可以优先考虑使用 WebAPP 开发技术。同时由于阿里云 IOT 平台提供了免费、功能强大的数据可视化系统，可以使用低代码的方式图形开发 WebAPP，还能直接导入该平台已接入的物联网设备，因此优先采用该方式进行用户 APP 界面的开发。

## 2.7 系统组成介绍

### 2.7.1 系统架构图

经过上述讨论，最终确定了本课题所采用的系统架构方案，架构的各主要组成部分可以进行关联，得到如下图 2.3 的系统架构图。

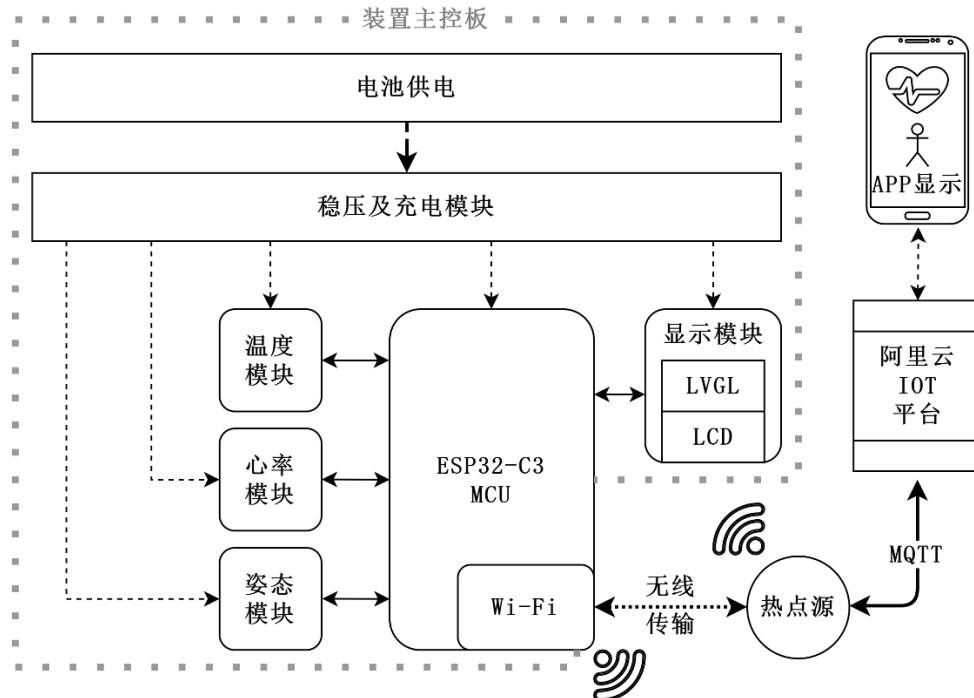


图 2.3 系统架构图

### 2.7.2 装置主控板

装置主控板由电池供电，电流经过稳压和充电模块后输送给主控 MCU 以及各模块。主控 MCU 与各传感模块进行通信以获取数据，经过整合与处理后将数据传输到显示模块，可供用户进行交互查看。

### 2.7.3 云服务器

装置可选择在连接到无线 Wi-Fi 热点源后，通过 MQTT 协议将获取的信息打包处理并上传到阿里云 IOT 平台的服务器。在将设备连接到云服务器前，需获取设备四元组信息并烧录至设备作为连接 MQTT 时的认证。

### 2.7.4 用户显示界面

整个系统的用户界面分为装置屏幕界面以及用户手机的 APP 界面两部分。

装置上的屏幕拟采用可触摸 LCD 屏，搭配 LVGL 作为 GUI 框架，可以提升用户的

交互体验，传感器数据可以进入到对应子页面进行查看，还可以通过屏幕上的软键盘连接到用户指定的热点源。

用户手机上的 APP 界面使用阿里云 IOT 平台的数据可视化系统搭建，在生成 WebAPP 后通过打包技术将其生成可安装的应用。用户 APP 获取装置上传至云服务器的各项数据后，解析并显示在用户界面上。

## 2.8 本章小结

本章主要从课题目标出发，分析了两种系统架构的优劣并确定了该课题倾向选择的架构，而后对比了几种无线通信技术的特点、市面上几款 RISC-V 架构的主控及特性、云服务器方案和几种通信协议、几种嵌入式 GUI 框架及 APP 界面方案，通过对方案进行优劣对比与分析，最终确立了系统最终的架构。

最后对系统架构中重要组成部分进行关联绘图，并简要解释了各部分的功能与意义，既明确了后续的设计思路，也清晰地解释了设计的合理性。

## 第3章 系统硬件电路设计

### 3.1 装置主控板

#### 3.1.1 主控芯片模块电路

本课题最终所选用的 RISC-V 主控芯片为乐鑫科技的 ESP32-C3 系列，其中拥有 22 个 GPIO 且包含内置 4MB Flash 存储器的型号为 ESP32-C3FN4 和 ESP32-C3FH4，前者为常温型号、后者为耐高温型号，本课题选用前者，其主要参数如下表 3.1 所示。

表 3.1 主控芯片主要参数

ESP32-C3FN4	
内核指令集(ISA)	RV32IMC
最大主频与性能	160MHz, 407.22 CoreMark
运行内存与片内存储	400 KB SRAM, 384 KB ROM (+4MB Flash)
GPIO 数目	22 个
封装	QFN32 (5mm*5mm)
数字接口	3 个 SPI、2 个 UART、1 个 IIC、1 个 I2S、红外收发器、 LED PWM 控制器、通用 DMA 控制器、1 个 TWAI 控制器 全速 USB 串口/JTAG 控制器
模拟接口	2 个 12 位 SAR 模/数转换器 (ADC)、1 个温度传感器
定时器	2 个 54 位通用定时器、3 个看门狗定时器、 1 个 52 位系统定时器
射频单元	2.4GHz 频段 Wi-Fi、BLE 5.0

从表 3.1 可知，该芯片除了拥有高性能、大运行内存以及超大存储外，基本外设功能也非常齐全，不管是 SPI (Serial Peripheral Interface Bus, 串行外设接口) 还是 IIC(Integrated Circuit, 集成电路总线) 通信的传感器，主控都可以使用硬件控制器与其进行通信，12 位的 ADC 也能满足大多数测量需求。最重要的是该芯片包含 2.4GHz 频段的 Wi-Fi 射频单元，支持 STA 模式和 SoftAP 模式，能满足本课题的需求，通过无线连接至广域网的阿里云 IOT 平台。

但正是因为该芯片包含 Wi-Fi 功能，为了发挥最佳射频性能，需要考虑 PCB 布局和天线阻抗匹配等条件<sup>[14]</sup>，因此本课题优先采用高集成度且带射频屏蔽罩的 MCU 模块进行开发。经过体积和价格的对比后，选择了安信可科技 ESP-C3-13 模块，其内部芯片

即是 ESP32-C3FN4，配合晶振、电阻电容等外围器件以及板载天线最终构成其模块，模块内部原理图可见附录 A。

关于模块的设计原理，需参考 ESP32-C3 硬件设计指南，其应包括以下几个基本组成部分：

(1) ESP32 主控芯片

(2) 外部时钟（晶振）电路

(3) 使能及复位电路

(4) 启动模式控制电路

(5) 调试烧录电路

(6) 射频匹配电路

封装为模块后的引脚原理图及实物图（正反面）如下图 3.1 所示。

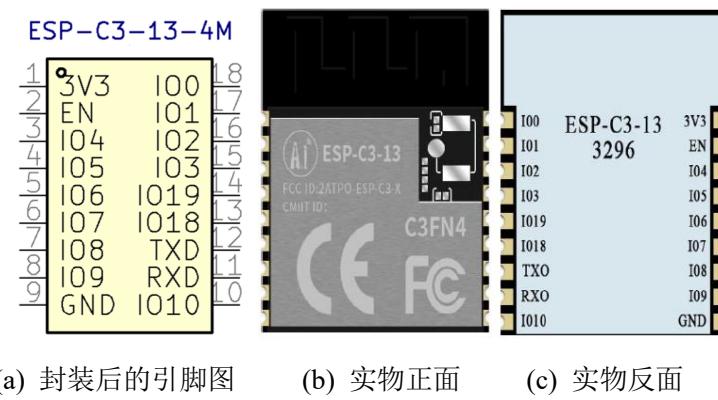


图 3.1 模块封装后原理图及实物图

下面针对模块的各主要组成部分进行设计原理的介绍，由于模块内已经包含外部时钟（晶振）电路，故以模块内的设计为准，如下图 3.2 所示。

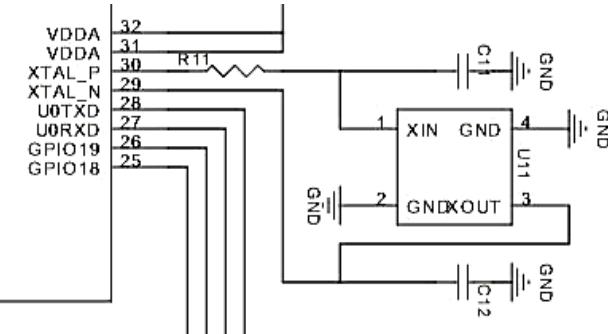


图 3.2 模块外部时钟电路

由于 ESP32-C3 系列芯片固件仅支持 40 MHz 晶振（精度需在 $\pm 10\text{ppm}$ ），且晶振外部匹配电容 C1、C2 具体值需要经过测试后才可最终确定（可选 20pf）。此外 XTAL\_P 时钟线上需串联器件以减小晶振的驱动能力，用于减弱晶振谐波对射频性能的影响，其

具体值也需要经过测试后确认（可选 24nH）。

对于使能与复位电路，实际上是对芯片 CHIP\_EN 引脚的高低电平控制，如下图 3.3。

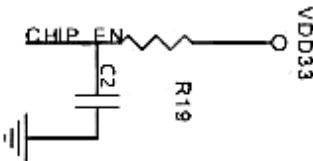


图 3.3 模块使能与复位电路

但值得注意的是，芯片上电使能和复位时该管脚需满足一定的时序要求，需求的时序如下图 3.4 所示。

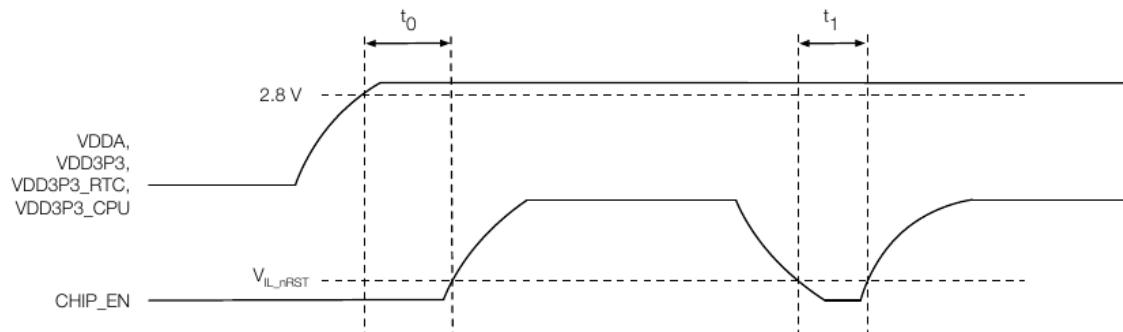


图 3.4 上电使能与复位时序

其中  $t_0$ 、 $t_1$  为电平变化的延迟时间，其最小值应不低于 50μs。为确保芯片上电时供电正常、实现如上图的启动时序，应在 CHIP\_EN 引脚处增加 RC 延迟电路，建议为 R 为 10kΩ，C 为 1μF，需根据实际使用进行调整。同理如需复位单片机，使用按键拉低 CHIP\_EN 引脚电平一段时间即可实现复位，本设计为节省按键的空间，拟通过断电并重新上电实现复位效果。

对于启动模式控制电路，实质上是通过对 Strapping 引脚电平的控制，ESP32-C3 一共有三个 Strapping 引脚分别为 GPIO2/8/9，在上电时改变其值可以控制芯片的系统启动模式，具体的模式见下表 3.2 所示。

表 3.2 ESP32-C3 系统启动模式

引脚	默认	SPI 启动模式	下载启动模式
GPIO2	无	1	1
GPIO8	无	无关	1
GPIO9	芯片内部上拉	1	0

因此综合上表，可见 GPIO2 和 GPIO8 对启动模式的影响不大，但需要进行上拉处理，GPIO9 则需要在正常启动时上拉为高电平、下载烧录时通过按键下拉到低电平。因此在装置的硬件电路设计中，启动电路如下图 3.5 所示。

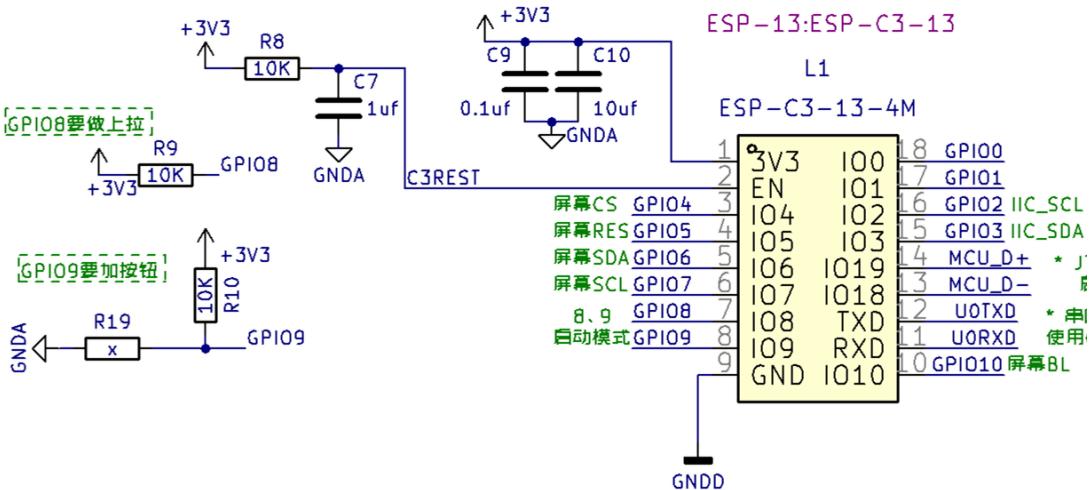


图 3.5 装置启动模式控制电路

由于本设计中拟采用 GPIO2 作为 IIC 总线 SCL 接口，在模块附近接入了上拉电阻，因此符合上表的启动模式要求；对于 GPIO8，模块内以及装置电路中都进行了上拉处理；对于 GPIO9，装置电路中使用了上拉电阻、按键下拉处理（虽然芯片内部有上拉），其中 R19 是使用电阻大小封装的按键，在烧录完成后可拆除以节省空间。

对于调试和烧录电路，在 ESP32-C3 主控芯片上最简单的方式是使用 USB 直接连接自带的二线 JTAG 接口（GPIO18 以及 GPIO19），连接关系见下表 3.3 所示。

表 3.3 JTAG 调试引脚连接关系

对象	相连引脚	
芯片引脚	GPIO18	GPIO19
USB 接口	D-	D+

但射频启动后会无法正常通过 JTAG 查看调试输出，因此选择 USB 经过串口转换芯片接入主控的 UART 引脚进行调试和烧录，对于该电路的设计详见本章 3.1.2 节。

对于射频匹配电路，设计时需添加  $\pi$  型匹配网络以便对天线进行匹配，官方建议匹配网络优先采用 CLC 结构，且其器件参数值需根据实际天线和 PCB 布局进行测试来确定，模块内的设计如下图 3.6 所示。

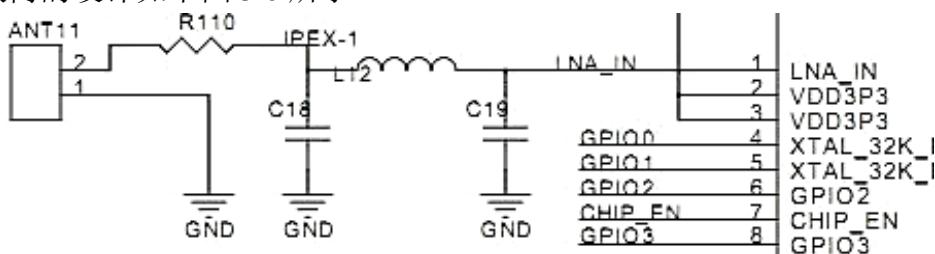


图 3.6 射频匹配电路

### 3.1.2 串口调试及下载电路设计

由于射频启动后无法使用内置 JTAG 接口配合 OpenOCD 进行单片机程序的调试与烧录，因此需要使用串口调试的方式。本装置的串口转换芯片选择沁恒微电子的 CH340C，其优点是自带内部晶振、外围器件简单、价格实惠。装置的串口调试及下载电路的相关设计如下图 3.7 所示。

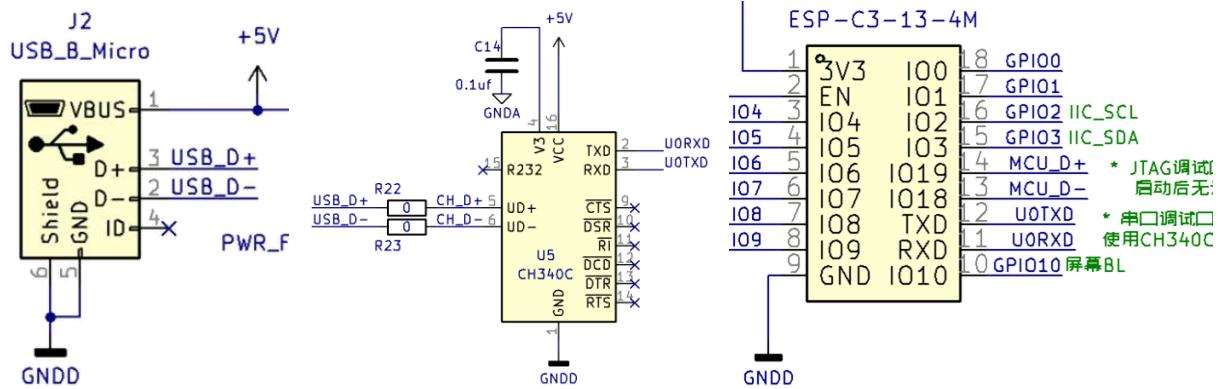


图 3.7 串口调试及下载相关电路

由图可见，USB 接口的差分信号 D+与 D-连接到串口转换芯片 CH340C 的差分输入引脚，转换为 TTL 电平后，接入主控芯片的 UART 进行调试。需要注意串口转换芯片与主控之间的 UART 接线方式，应当是收发互反进行连接的。

此外，由于串口调试仅在接入 USB 后使用，故 CH340C 使用 USB 提供的 5V 供电可以简化电路布局与外部器件，根据沁恒官方设计指导手册，仅需要注意将 V3 引脚串联  $0.1\mu F$  电容到地即可。对于 R22、R23 的零欧电阻为预留位，除具有一定程度的阻抗匹配作用外，在发现设计不合适时可更改为磁珠等器件。

### 3.1.3 电源及充电管理电路设计

装置在运行时使用 3.7V 锂电池供电，在非运行时可通过 USB 接口向电池进行充电，因此需要充电管理芯片。本装置选用的 TP4056 是一款经典的充电管理芯片，有过温保护、大电流、价格实惠等优点，其引脚与功能如下图 3.8 所示。

VCC: 电压正输入端  
GND: 公共地  
CE: 芯片使能输入端(高电平有效)  
BAT: 电池连接端  
TEMP: 电池温度检测输入端  
PROG: 恒流充电电流设置端  
CHRG: 充电状态指示端  
STDBY: 电池充电完成指示端

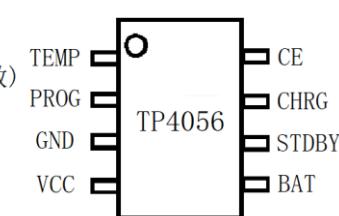


图 3.8 TP4056 引脚与功能

由上图结合芯片数据手册，其 TEMP 引脚在不需要监测电池温度时可接地，而从 PROG 引脚串联电阻到地，可实现控制充电电流大小，对于充电电流值  $I_{charge}$ (A)与采用的电阻值  $R_{PROG}$ ( $\Omega$ )的关系可用下式(3.1)近似表示。

$$R_{PROG} = \frac{1200}{I_{charge}} \quad (3.1)$$

对于本装置，为避免充电电流过大，拟控制为 250mA，故根据上式选用 4.7K $\Omega$  的电阻从 PROG 引脚串联到地即可。

由于主控及传感器等器件所需电源为 3.3V，因此仍需加入线性稳压芯片将 3.7V 锂电池的电压降至工作电压范围。对于 ESP32-C3，在一般情况下的工作电流仅有 5~50mA，而当射频单元启动时，电流瞬时峰值可达 300mA 以上，综上需要选择支持低压差、大输出电流的 LDO（线性稳压器），经过评估后选择了 TC1262-3.3VDBTR。装置的主要供源与充电管理电路如下图 3.9 所示。

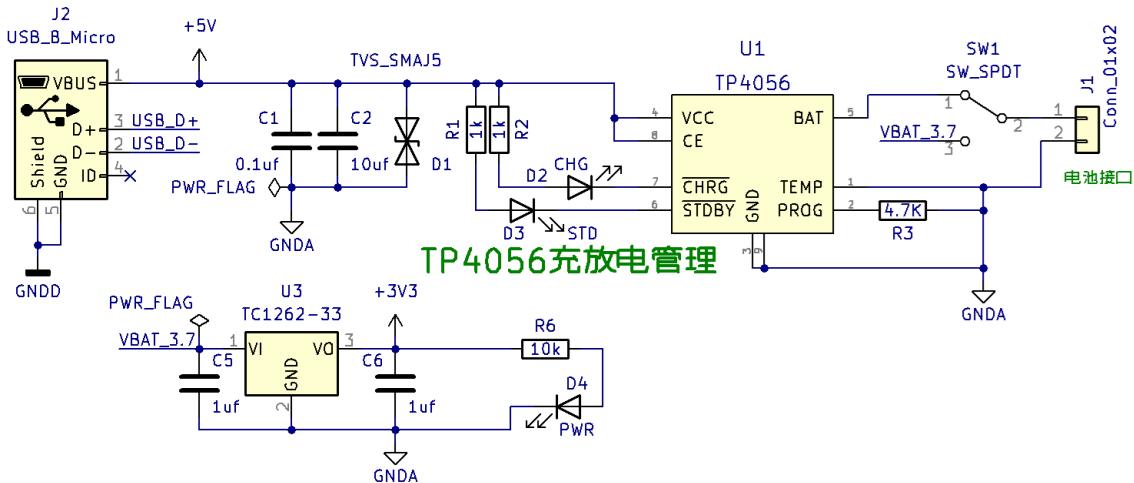


图 3.9 供源与充电管理电路

由上图可知，充电时拨动开关连接 1、2 端口，电流从 USB 接口经过电容去纹波、TVS 管抑制脉冲后进入 TP4056，其 BAT 引脚恒定输出约 250mA 的电流进入电池，并点亮 D2 的 LED，直至电池充满后 D2 灭、D3 亮。放电时，拨动开关连接 2、3 端口，3.7V 锂电池的电流经过 TC1262 稳压到 3.3V 后输送给各元件，同时指示灯 D4 亮。

### 3.1.4 PPG 传感器电路设计

本装置的心率血氧传感器拟使用 PPG 传感器，其中 Maxim 生产的 MAX30102 是一款经典、高分辨率、实惠的 PPG 传感器。它集成了两个不同波段的 LED，分别是用于测量心率的 660nm 红光、用于测量血氧的 880nm 红外光，此外还具有光电检测器、抑制环境光的低噪声电路等，下图 3.10 为传感器引脚及说明。

N. C. : 无连接  
 SCL: IIC时钟引脚  
 SDA: IIC数据引脚  
 PGND: 用于LED的电源地  
 V<sub>LED+</sub>: LED电源供应  
 V<sub>DD</sub>: 模拟电源输入  
 GND: 模拟地  
 INT: (低电平有效) 中断输入

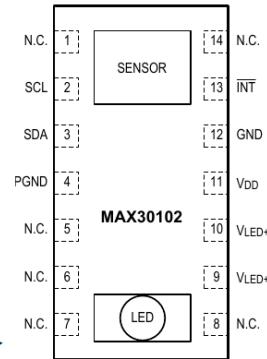


图 3.10 PPG 传感器 MAX30102 引脚及说明

结合 MAX30102 数据手册，考虑装置体积布局，简化电路设计可见下图 3.11。

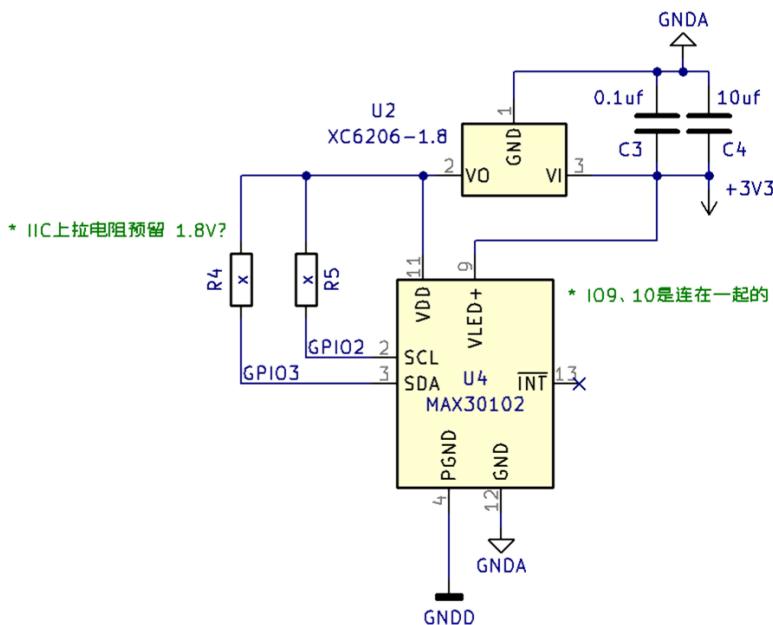


图 3.11 PPG 传感器电路设计

由于 V<sub>DD</sub> 要求 1.8V 供电，故选用 XC6206-1.8 稳压芯片，将 3.3V 的电源输入降压稳至 1.8V 接入 V<sub>DD</sub>。此外，由于设计上不使用中断功能，故将 INT 引脚悬空，并在电路中预留了 IIC 总线上拉电阻的位置，实际电路中只需要一处模块对 IIC 总线上拉即可，此处仅当电路布线过长、电平拉不高时作为备用上拉电阻使用。

### 3.1.5 温度传感器电路设计

由于装置需要获取人体温度，其常见方式有接触式、非接触式测量两种，对于本装置需实现可穿戴、小体积，因此首选用接触式测温传感器。Maxim 生产的 MAX30205 是一款专用于测量人体温度的传感器，其具有体积小、驱动简单、人体温度区间精度高等优点，因此本装置优先使用该传感器，下图 3.12 为 MAX30205 引脚及说明。

SDA:IIC数据引脚  
 SCL:IIC时钟引脚  
 OS:过温比较输出  
 GND:电源地  
 A2:IIC从机地址控制位2  
 A1:IIC从机地址控制位1  
 A0:IIC从机地址控制位0  
 VDD:电源输入(3.3V)  
 EP:底部散热焊盘

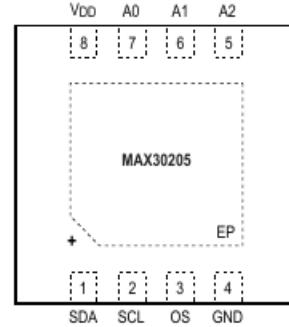


图 3.12 温度传感器 MAX30205 引脚及说明

由上图，结合 MAX30205 数据手册，装置相关电路设计如下图 3.13 所示。

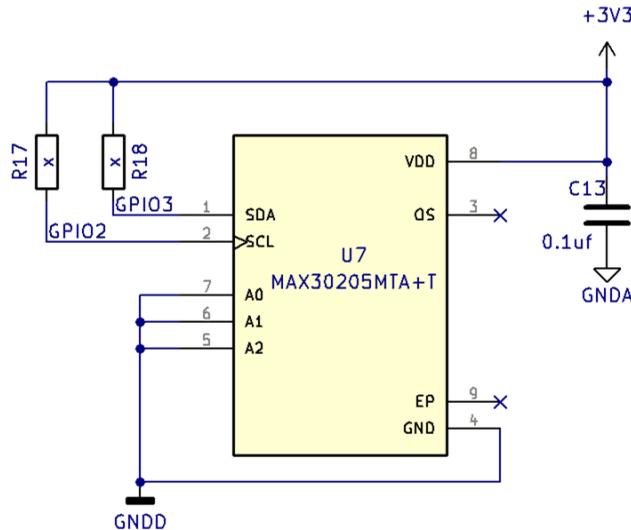


图 3.13 温度传感器电路设计

由于该传感器 IIC 从机地址与其他传感器地址并不冲突，因此将 A2、A1、A0 地址控制引脚均接地即可。对于 IIC 接口，也同样预留了上拉电阻，此外其 OS 引脚并未使用因而悬空，EP 引脚在原理图上也无相连，但 PCB 上需大面积增加散热。

### 3.1.6 姿态传感器电路设计

本装置的姿态传感器选用 MPU-6050，这是 InvenSense 公司生产的一款经典 6 轴 IMU，具有 3 轴陀螺仪与 3 轴加速度传感器，此外还具有内置 DMP (Digital Motion Processor, 数字运动引擎) 可直接融合传感器数据并滤波、解算输出四元数等姿态数据，因此大大降低了主控 MCU 的性能负担，下图 3.14 为 MPU-6050 相关引脚及功能。

CLKN: 外部时钟输入, 未使用可接GND  
 AUX\_DA: IIC主机数据引脚(连外部传感器)  
 AUX\_CL: IIC主机时钟引脚(连外部传感器)  
 VLOGIC: 数字电源电压引脚  
 ADO: 控制IIC从机地址  
 REGOUT: 连接调节器滤波电容器  
 FSYNC: 帧同步数字输入, 未使用可接GND  
 INT: 中断输出引脚  
 VDD: 电源电压引脚  
 GND: 电源地  
 RESV: 保留引脚, 无需连接  
 CPOUT: 连接电荷泵电容器  
 SCL: IIC从机时钟引脚(连外部MCU)  
 SDA: IIC从机数据引脚(连外部MCU) NC: 空引脚, 无需连接

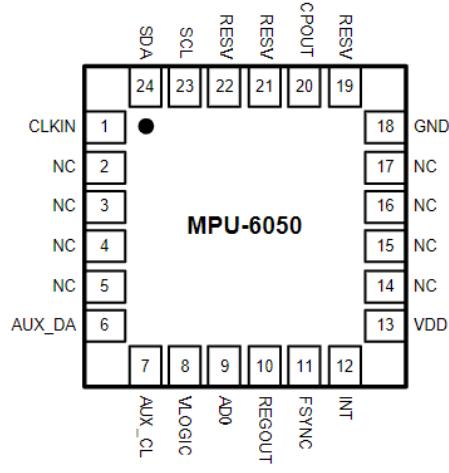


图 3.14 姿态传感器 MPU-6050 引脚及说明

由上图, 结合 MPU-6050 数据手册, 装置相关电路设计如下图 3.15 所示。

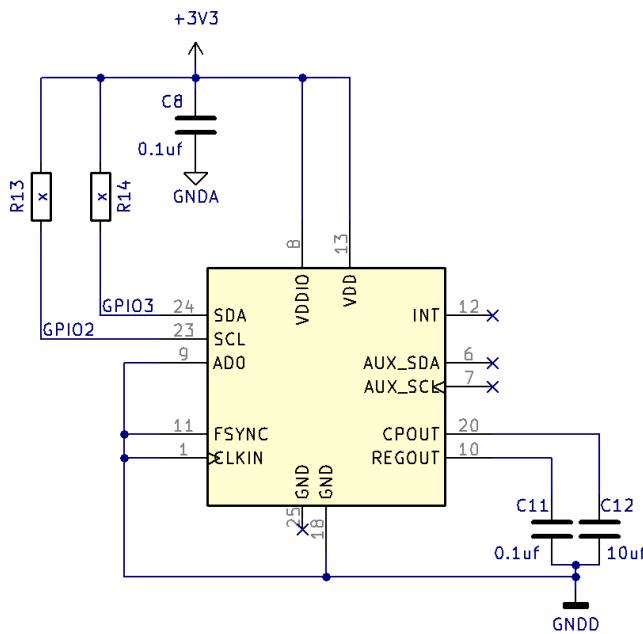


图 3.15 温度传感器电路设计

本装置并未使用到 MPU-6050 的所有功能引脚, 原理图已将部分无用引脚隐藏, 根据数据手册的介绍, 不使用外部晶振、帧同步功能时可将 CLKIN 引脚、FSYNC 引脚接地, CPOUT 以及 REGOUT 对应连接  $10\mu\text{f}$ 、 $0.1\mu\text{f}$  的电容即可; 对于 ADO, 该传感器的 IIC 从机地址也不并冲突, 因此接 GND 即可; 对于 INT、AUX\_SDA、AUX\_SCL 引脚, 装置都并未使用, 因此直接悬空即可。此外, 由于 25 号引脚对应封装底部散热焊盘, 实际上无需将其与地相连, 因此原理图上不作连接。

### 3.2 屏幕底板

本装置为实现友好的人机交互，拟采用可触摸 LCD 作为直连装置的输入、显示设备，经过对比最终选择了满足需求、价格实惠的沃乐康 WA54HC045I-12S 屏幕，该屏幕使用 ST7789V 芯片进行显示驱动、FT6236G 作为触摸驱动。由于屏幕模块提供底部控制板，底板电路原理图可详见附录 A，下图 3.16 是屏幕底板的局部原理图。

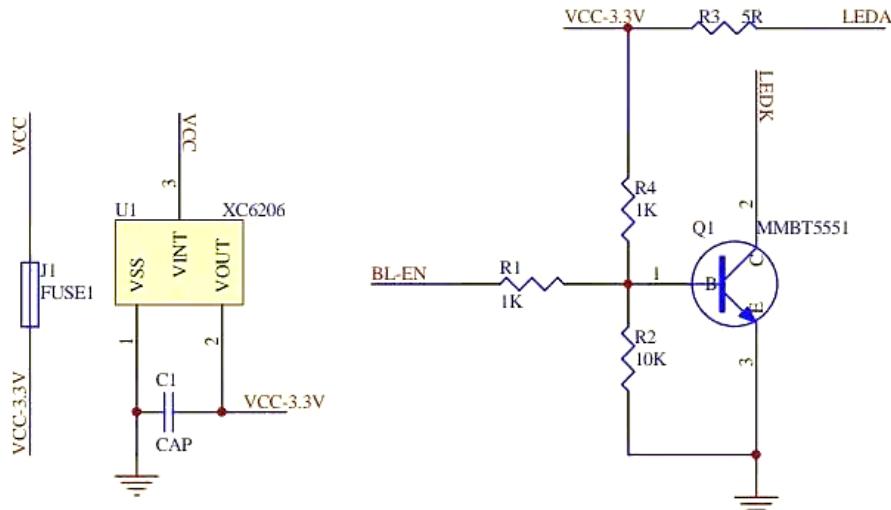


图 3.16 屏幕底板局部原理图

由上图可知，屏幕底板已经提供 XC6206-3.3 稳压芯片，因此可直接通过装置电路的锂电池 3.7V 供源，且底板已使用三极管控制背光，故装置电路中直接连接引脚即可，下图 3.17 是对接屏幕底板的装置电路原理图。

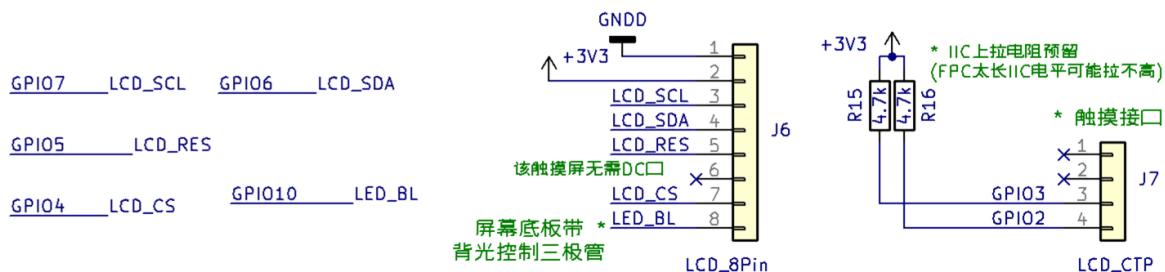


图 3.17 对接屏幕底板原理图

在装置与屏幕接口的电路设计中，触摸 IC 使用 IIC 接口通过 FPC 软排线连接，所以依旧采用了预留上拉电阻的设计，该屏幕使用特殊的 9 位 SPI 协议进行显示驱动，因此无需连接 DC 引脚，故将其悬空。

### 3.3 本章小结

本章主要介绍了装置的硬件电路设计，从目标功能出发，对主控芯片与模块电路、

供源及充电管理电路、PPG 传感器电路、温度传感器电路、姿态传感器电路、屏幕底板及装置接口各个部分进行介绍，并结合对应的芯片数据手册分析了硬件接口的连接方式，阐明了设计思路，体现了硬件设计的合理性。

## 第4章 系统软件设计

### 4.1 装置主控板

#### 4.1.1 装置软件整体设计

本装置的软件开发使用乐鑫官方的 ESP-IDF 框架，在框架之上运行 FreeRTOS（一款开源免费的嵌入式实时操作系统），在此基础上使用 LVGL 框架用于管理定时任务与屏幕 GUI 交互，而在最上层的则是用户应用功能，如传感器数据获取、显示等，因此软件的开发框架可以用下图 4.1 简化表示。

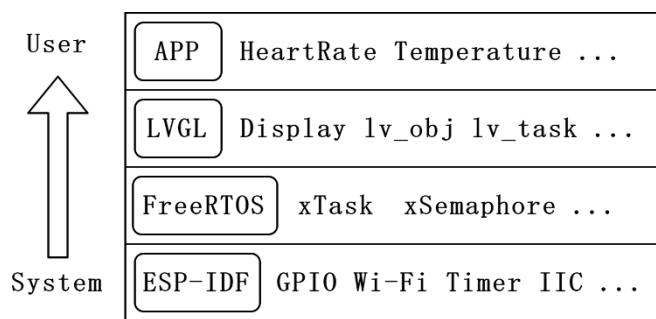


图 4.1 装置软件框架图

对于装置主控板内单片机编写的程序，主要实现了以下流程：

- (1) 装置上电后，会进行外设及 FreeRTOS 初始化，系统初始化完成后即可创建用户自定义的任务。
- (2) 在用户任务中，进行传感器、屏幕及 LVGL 图形界面的初始化，并检查初始化是否有异常，若有异常则尝试重新初始化，或提示用户检查并重启装置。
- (3) 初始化成功后使用 LVGL 自带的任务管理框架，创建秒级、毫秒级的定时循环任务——秒级任务用于计时、温度传感器更新数据、刷新界面显示数值等；毫秒级任务用于 PPG 及姿态传感器更新数据等。
- (4) 创建任务完成后进入到主循环内，调用 LVGL 处理任务队列，用于维持 LVGL 框架正常的运作与 GUI 显示，同时具有异常检测的特性。
- (5) 当用户操作网络连接成功后（仅一次），会创建定时 MQTT 上传数据的任务，之后回到主循环中执行上述第(4)条；当出现异常时会抛出异常状态，严重的异常会结束程序，否则会回到主循环中执行上述第(4)条。

上述流程可以用下图 4.2 简化表示，且相关的程序实现可详见附录 D。

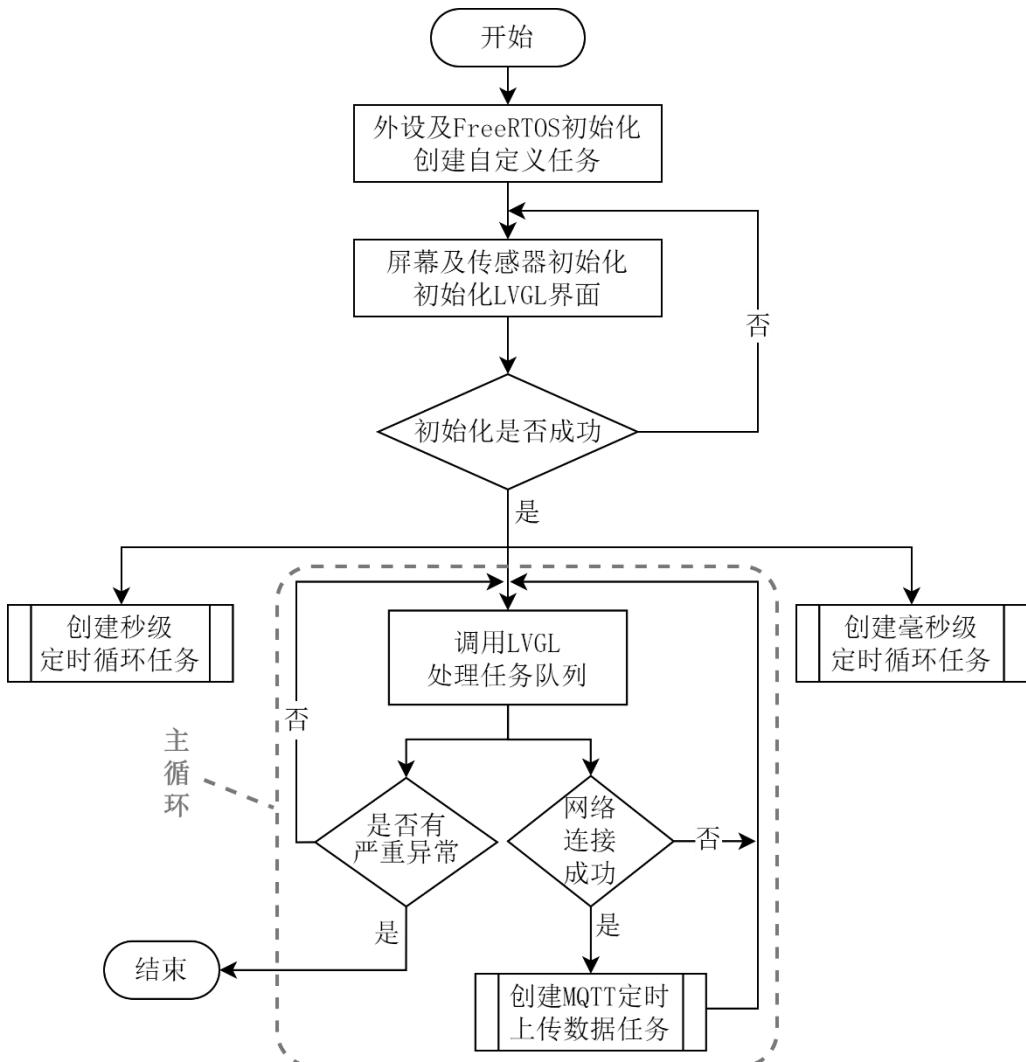


图 4.2 装置整体软件流程图

#### 4.1.2 温度获取

装置程序中的温度测量功能，是通过与 MAX30205 传感器进行 IIC 通信更新数据的，ESP-IDF 手册中讲述了使用该框架配置 IIC 的方法如下：

(1) 配置驱动程序：设置初始化参数（设置 `i2c_config_t` 结构体的配置，包括工作模式、通信引脚、时钟速度等）。

(2) 安装驱动程序：调用函数 `i2c_driver_install()` 以安装并激活 IIC 控制器。

(3) 根据是主机还是从机，来配置驱动程序，执行以下操作：

    主机模式下通信：发起通信。

    从机模式下通信：响应主机消息。

(4) 中断处理：可调用函数 `i2c_isr_register()` 注册用户自定义中断处理程序。

(5) 自定义配置：可根据需求调用函数 `i2c_param_config()` 调整默认的 IIC 通信参

数（如时序、位序等）。

(6) 错误处理：调用 `ESP_ERROR_CHECK()` 识别和处理配置和通信错误等信息。

(7) 删除驱动程序：在通信结束时调用 `i2c_driver_delete()` 释放 IIC 驱动程序所使用的资源。

对于通过 IIC 与 MAX30205 通信，需要在实现上述配置流程的情况下，加入对传感器的初始化配置、读写数据寄存器等操作，该流程可用下图 4.3 简化表示。

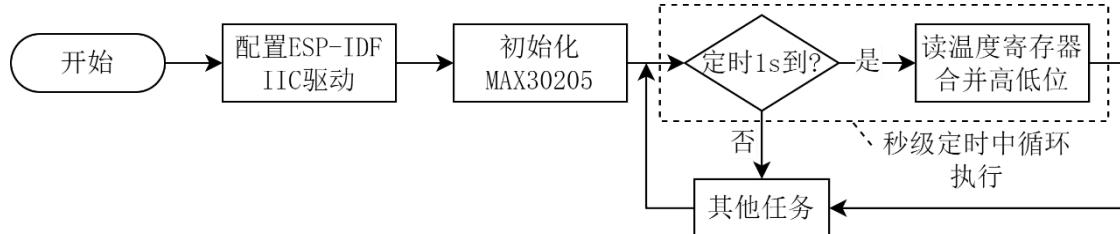


图 4.3 获取温度数据流程图

结合 MAX30205 数据手册，对其初始化实质上是对配置寄存器写入 0x00，一个典型的写入配置寄存器的时序如下图 4.4 所示。MCU 主机拉低 SDA 电平后就开始发送地址位，传感器从机收到后做出应答，随后主机发送指针字节（寄存器地址），从机收到后再次应答，收到应答后主机发送寄存器配置数据，最后收到从机的应答会由主机拉高 SDA 电平完成通信。

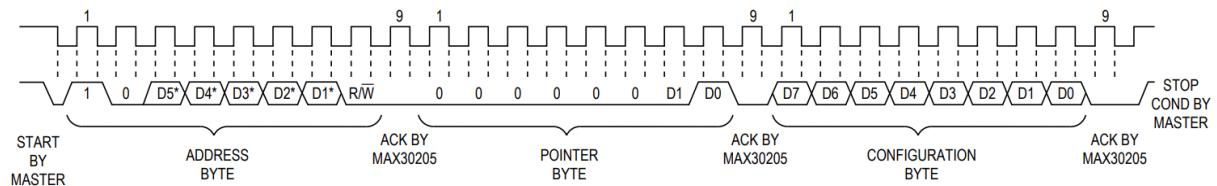


图 4.4 写入 MAX30205 配置寄存器时序图

更新温度数据是通过读取 MAX30205 的温度寄存器来完成的，由于其为 16 位数据，分高、低各 8 位进行存储，下图 4.5 是连续读取 2 字节温度寄存器数据的时序图，具体流程与上述同理故不再赘述。

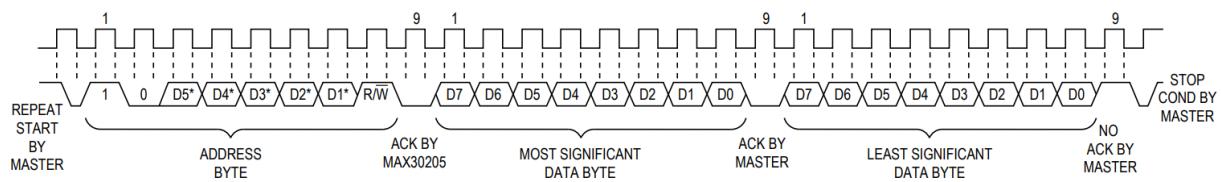


图 4.5 读取 MAX30205 温度寄存器时序图

得到十六位温度数据后，通过数据手册中如图 4.6 所示的方式，按各数据位对应相乘运算再求和，即可得到以摄氏度为单位的温度数据。

UPPER BYTE								LOWER BYTE							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
S	MSB 64°C	32°C	16°C	8°C	4°C	2°C	1°C	0.5°C	0.25°C	0.125°C	0.0625°C	0.03125°C	0.015625°C	0.0078125°C	0.00390625°C
	2 <sup>6</sup>	2 <sup>5</sup>	2 <sup>4</sup>	2 <sup>3</sup>	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>	2 <sup>-1</sup>	2 <sup>-2</sup>	2 <sup>-3</sup>	2 <sup>-4</sup>	2 <sup>-5</sup>	2 <sup>-6</sup>	2 <sup>-7</sup>	2 <sup>-8</sup>

图 4.6 温度寄存器数据合并转换图

#### 4.1.3 计步测距与久坐提醒

对于装置计步与测距功能的实现，也是通过主控与 MPU6050 进行 IIC 通信获取的，有关 IIC 的配置和使用详见本章 4.1.2 节，由于通信协议较为类似而繁琐，故不再赘述。

计步功能可通过 MPU6050 内置的 DMP 进行运算和记录，无需另外编写算法实现，大大降低了开发难度及 MCU 资源占用，在每秒的定时任务中更新计步数据即可，当步数长时间无变化则使用屏幕弹窗提醒用户久坐，上述实现流程可见下图 4.7。

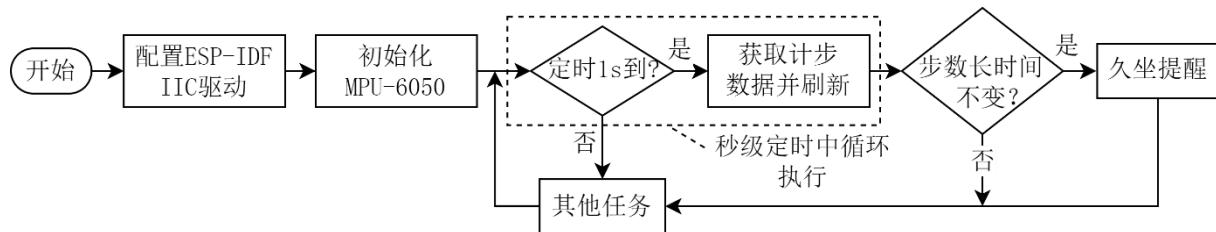


图 4.7 计步及提醒久坐实现流程图

对于步行距离的测算，可以采用刘雷等人的步长估算方法及模型，利用每一步的合加速度、步频来综合计算出步长<sup>[15]</sup>，算法的实现流程可简化为以下流程。

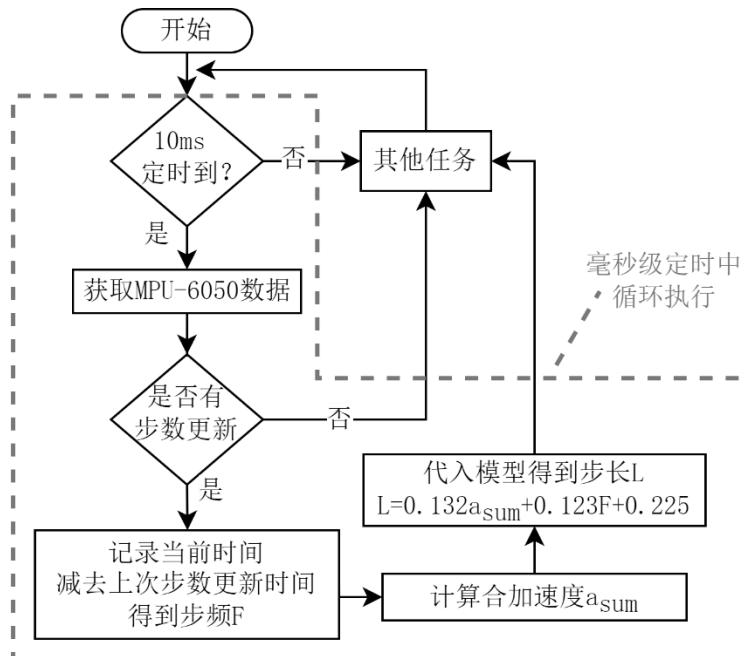


图 4.8 测距算法实现流程图

步距的测算为了保证及时性与准确性，放在了毫秒级定时器中。每隔 10ms 检查一次计步的数值，若发现步数更新则立刻记录当前时间，并减去上次计步的时间，计算出步频 F，并通过当前三轴加速度计算合加速度  $a_{sum}$ ，带入到模型中得到步长 L，模型计算式如下式(4.1)所示。

$$L = 0.132a_{sum} + 0.123F + 0.225 \quad (4.1)$$

根据刘雷等人的计算，该数学模型的相关系数  $r^2=0.928$ ，说明拟合度较好，且实测数据也仅有 1.9% 的误差，但对于不同设备可能会有所差异，所以装置需要根据实测结果对该模型进行微调。

#### 4.1.4 跌倒检测算法

本装置的跌倒检测算法主要基于郭元新等人的研究进行改良，对于嵌入式设备而言，利用 IMU 实现低资源占用、高正确识别率的跌倒检测，使用加速度数据计算 SVM (signal magnitude vector, 加速度信号向量)、配合 CV (coefficient of variation, 变异系数或离散系数) 进行估计、结合角度数据纠错是较为合理的方法，既能减少其他方法带来的大量积分运算、也能降低一定的误判<sup>[16]</sup>，其简化算法可见下图 4.9。

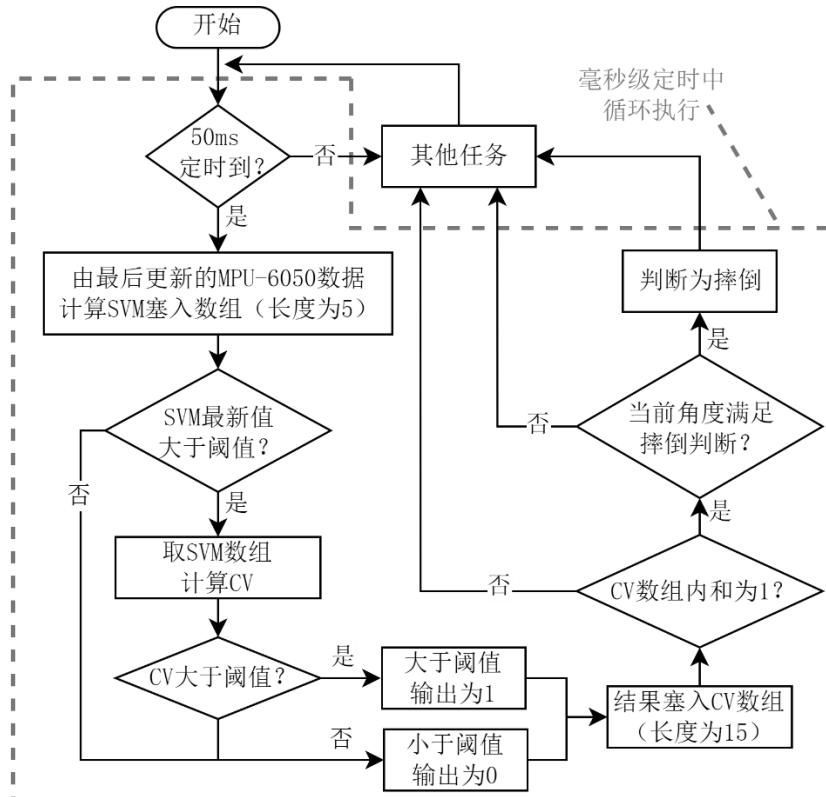


图 4.9 跌倒检测算法实现流程图

跌倒检测的程序位于 50ms 定时器中循环执行，当定时触发后，程序会将最后一次

更新的 MPU-6050 三轴加速度数据  $a_x$ 、 $a_y$ 、 $a_z$  按照下式(4.2)计算 SVM 值。

$$SVM = \sqrt{a_x^2 + a_y^2 + a_z^2} \quad (4.2)$$

最新的 SVM 数据会塞入至长度为 5 的 SVM 数组中(数组长度为 5 即设定窗口时间为 0.25s, 与人体跌倒时间接近), 若 SVM 最新值比阈值大, 则用该数组计算 CV 值, CV 值的计算方法如下式(4.3)所示。

$$\begin{aligned} \bar{x} &= \frac{1}{n} \sum_{i=1}^n x_i, \\ SD &= \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2}, \\ CV &= \frac{SD}{\bar{x}} \end{aligned} \quad (4.3)$$

这里要对郭元新等人的论文进行勘误, 变异系数 CV 应当是标准差与均值之比, 该定义式可在《概率论与数理统计》教材及相关统计学论文中可以查阅<sup>[17]</sup>。若计算出的 CV 值也大于阈值, 则将 1 装入长度为 15 的 CV 数组中、否则将 0 装入, 数组长度选为 15 是因其为 3 个窗口时间, 若 CV 数组内仅有 1 次超过阈值, 则可以高度怀疑跌倒, 此方法能一定程度上过滤掉人体剧烈活动导致的误判。由于人体跌倒时, 会本能地使用手部支撑, 因此可借助此时的角度数据做判断, 若在此基础上满足一定的角度阈值即可最终判断为摔倒, 相关代码实现可见附录 D。

#### 4.1.5 心率与血氧检测算法

本装置心率血氧检测的整体设计与温度传感器及时间相关, 拟在温度传感器检测到人体佩戴时、且每隔 5 分钟开启 30s 检测, 其整体程序逻辑可见下图 4.10。

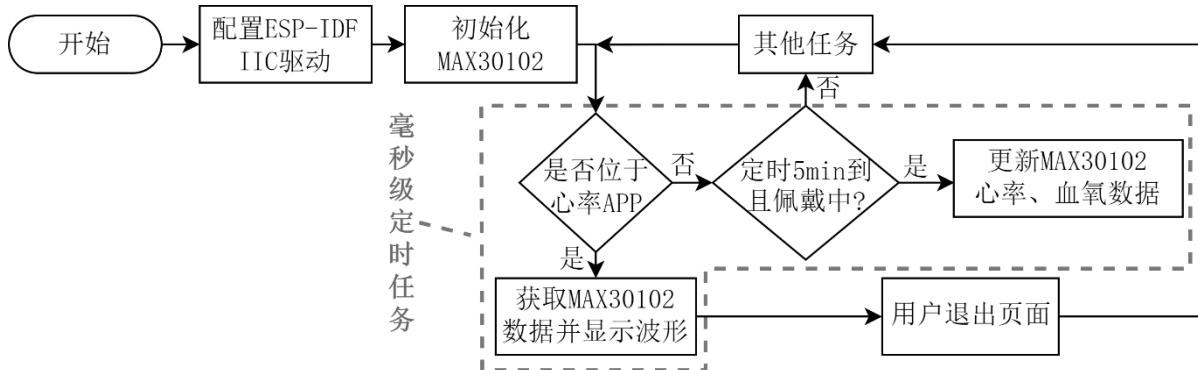


图 4.10 心率血氧检测整体逻辑

为响应及时, 在毫秒级定时任务中检查当前用户屏幕是否处于心率页面, 若是, 则

循环获取 MAX30102 数据并计算显示滤波后的 PPG 波形、当前心率及血氧数值，否则等到定时 5min 触发后台检测机制，若装置有人佩戴，则获取 30s 的 PPG 数据计算心率及血氧值并存储至变量，在联网的情况下可上传至服务器。

对于 MAX30102 数据的更新算法，主要通过 IIC 通信读取红外、红色 LED 的数据，对信号进行直流移除、平均差处理、低通滤波，之后再进行心率与血氧值的计算<sup>[18]</sup>，数据更新流程可用以下流程图 4.11 表示。

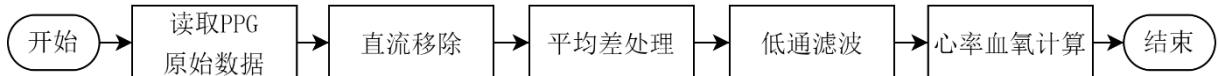


图 4.11 心率血氧数据更新流程图

对于直流移除，可以使用一种简单高效的滤波器方式，即将输入信号  $x(t)$  通过下式(4.4)进行迭代，得到移除直流后的输出  $y(t)$ 。

$$\begin{aligned} w(t) &= x(t) + \alpha \times w(t-1), \\ y(t) &= w(t) - w(t-1) \end{aligned} \quad (4.4)$$

其中， $w(t)$  为每次迭代产生的中间值，其与上次迭代产生的差即为交流部分； $\alpha$  则可称之为响应常数，其越接近 1，越能体现数据的变化性，直流移除时一般取 0.9 左右。

对于平均差处理，实际上是将移除直流后的数据放入数组中进行滑动平均，每次将新数据减去均值，以此得到平均差处理后的结果。

对于低通滤波，则是将平均差处理后的数据送入低通滤波函数中，只要确定了滤波器类型与阶数、采样率与截止频率等参数，低通滤波器的系数可以使用 MATLAB 等工具生成。装置的 MAX30102 初始化采样率为 100Hz，由于人的心率一般不会大于 220BPM，因此可以使用下式(4.5)进行换算得到对应截止频率  $f_c$ 。

$$f_c = \frac{220 \text{ (BPM)}}{60 \text{ (s)}} = 3.66 \text{ (Hz)} \quad (4.5)$$

不妨使用曲线较为平缓的一阶巴特沃斯低通滤波器，为避免波形畸变、留出截止频率的余量，故取  $f_c=10\text{Hz}$ ，结合采样率  $f_s=100\text{Hz}$ ，那么可以通过工具生成对应的滤波器系数，其原理是通过 Z 变换将转移函数转换为线性常系数差分方程从而实现的<sup>[19]</sup>，本装置的滤波器系数及代码可见附录 D。

对于心率的计算方法，实质上是对信号峰值的检测，利用装置获取时间差并进行换算，可以将该过程使用如下图 4.12 的状态机进行表示。该状态机持续监测传感器信号，一旦进入设定的阈值范围（大于最小阈值、小于最大阈值），跟随曲线并判断上升、下降状态，并在上升时获取当前时间戳直至信号到达峰值。利用两次峰值的时间差即可计算出心率。

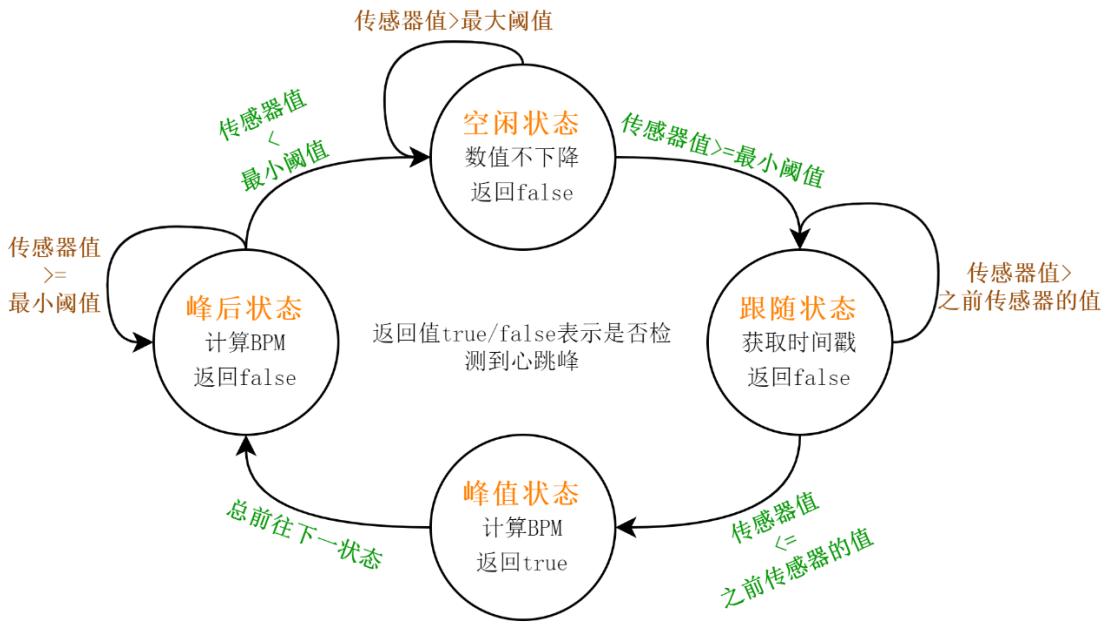


图 4.12 心率计算状态机

干扰信号可能导致 BPM 结果飘动，因此需要对不合理结果进行剔除，考虑到正常人在不同活动情况下心率范围处于 50~220 之间，故将超出范围的结果剔除掉，以提高数据准确性。此外还可使用滑动平均法，通过数组将最新的几个 BPM 结果进行求平均运算，最终在用户界面显示平均值，以提高数据稳定性。

对于 SpO<sub>2</sub>，实际为“经皮血氧饱和度”，其近似于动脉血氧饱和度(SaO<sub>2</sub>)，能够表示血液中含氧血红蛋白与总血红蛋白之比，正常值应在 95%~99%之间波动，使用 PPG 传感器时，通常需要根据光信号的交流、直流值对比计算得到 SpO<sub>2</sub>。而根据 MAX30102 官方的应用操作手册，本传感器需要红光 LED 以及红外 LED 的信号先按照下式(4.6)计算 R 值 (ratio of ratios，比值之比)。

$$R = \frac{\left(\frac{AC_{Red}}{DC_{Red}}\right)}{\left(\frac{AC_{IR}}{DC_{IR}}\right)} \quad (4.6)$$

其中， $AC_{Red}$ 、 $DC_{Red}$  分别为红光 LED 的交流、直流值； $AC_{IR}$ 、 $DC_{IR}$  分别为红外光 LED 的交流、直流值。根据此 R 值，代入到官方应用操作手册中的拟合公式，即可计算得到 SpO<sub>2</sub>，拟合公式见下式(4.7)。

$$SpO_2 = 104 - 17R \quad (4.7)$$

SpO<sub>2</sub> 的算法核心如上所示，整体上并不复杂，限于篇幅原因，一些细枝末节及代码实现可见附录 D。

#### 4.1.6 屏幕交互逻辑设计

装置使用了可触摸 LCD 屏幕作为交互的输入输出设备，为方便用户操作及查看数据，设计了一定的页面操作逻辑，其可用下图 4.13 简化表示。

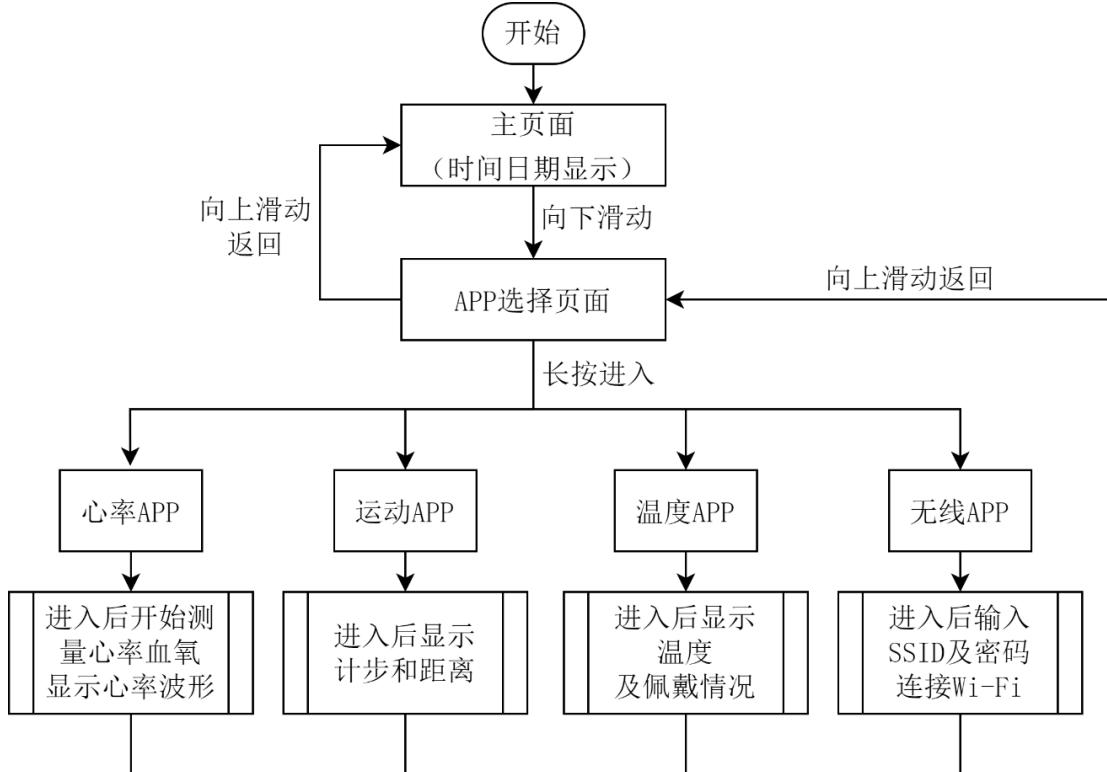


图 4.13 屏幕交互逻辑示意图

#### 4.1.7 数据上传云服务器

装置连接 Wi-Fi 后会自动联网，通过 SNTP 同步时间，用于更新主页面的时间显示。时间同步完成后会通过 MQTT 协议连接到阿里云 IOT 服务器，并每隔 5 秒主动向物模型属性上报 Topic 上传各项传感数据，该过程可由下图 4.14 简化表示。

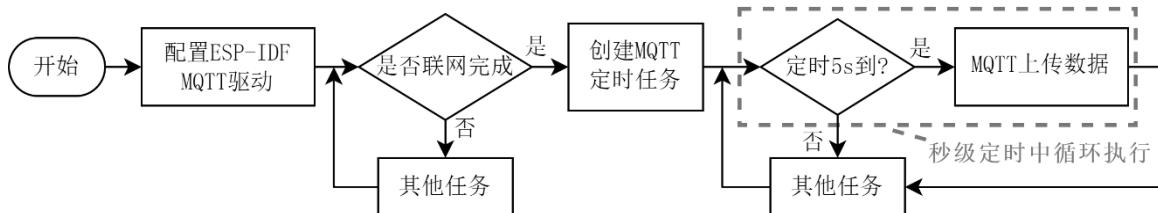


图 4.14 数据上传流程图

定时 MQTT 上传的数据按照阿里云 IOT 平台物模型的格式，包含了当前温度、计步值、最后一次测得的心率和血氧结果、是否摔倒，这几项信息，详细的代码实现方式可见附录 D。

## 4.2 服务器数据处理

### 4.2.1 处理物模型数据

若想设备的各项信息成功对接至阿里云 IOT 平台，并且可显示在 WebAPP 中，需要将数据按照规定的物模型格式进行上传，该格式实质上只是按照一定规范排列的 JSON 文本。此外，平台中要先在产品管理页面定义功能如下图 4.15。

默认模块

功能类型	功能名称 (全部) ▾	标识符 ⓘ	数据类型	数据定义	操作
属性	摔倒 <a href="#">自定义</a>	Fallen	bool (布尔型)	布尔值: 0 - 正常 1 - 摔倒	<a href="#">查看</a>
属性	跑步步数 <a href="#">自定义</a>	Step	int32 (整数型)	取值范围: 0 ~ 100000	<a href="#">查看</a>
属性	温度 <a href="#">自定义</a>	Temp	float (单精度浮点型)	取值范围: 0 ~ 50	<a href="#">查看</a>
属性	血氧饱和度 <a href="#">自定义</a>	SpO2	float (单精度浮点型)	取值范围: 0 ~ 100	<a href="#">查看</a>
属性	心率 <a href="#">自定义</a>	HR	float (单精度浮点型)	取值范围: 0 ~ 150	<a href="#">查看</a>

图 4.15 阿里云 IOT 平台功能定义

要注意各属性的标识符、数据类型，两者应当与装置上传的物模型数据格式相匹配，否则平台无法解析数据。

```
{"\"method\":\"thing.event.property.post\", \"params\":{ \"Temp\":%2.1f, \"Step\":%d, \"HR\":%2.1f, \"SpO2\":%2.1f, \"Fallen\":%d}}
```

图 4.16 装置对应上传数据格式

若数据解析成功，则可以在设备信息页面查看到设备上传的实时物模型数据，如下图 4.17 所示。

属性	值	操作
心率	74.6	<a href="#">查看数据</a>
血氧饱和度	99.8 %	<a href="#">查看数据</a>
温度	30.3 °C	<a href="#">查看数据</a>
跑步步数	0	<a href="#">查看数据</a>
摔倒	0 (正常)	<a href="#">查看数据</a>

图 4.17 解析成功后查看数据

## 4.2.2 业务逻辑设计

在阿里云 IOT 数据可视化平台中，新建如下图 4.18 所示的两个业务逻辑，分别用于查询装置在线状态、阿里钉钉机器人推送跌倒警报信息。

名称	描述	所属项目	修改时间	发布状态	发布时间	运行状态
摔倒报警	-	C3WebAPP	2022-04-13 22:00:39	已发布	2022-04-13 22:02:17	<span style="color: green;">运行中</span>
设备在线状态	-	C3WebAPP	2022-04-13 21:43:10	已发布	2022-04-13 21:43:21	<span style="color: green;">运行中</span>

图 4.18 创建业务逻辑

其中查询装置在线状态的业务逻辑如下图 4.19 所示，当用户使用 WebAPP 获取设备状态灯控件时，业务逻辑会发送 HTTP 请求给设备所在 MQTT 服务器，查询设备的在线状态并通过 HTTP 返回结果至 WebAPP 控件。

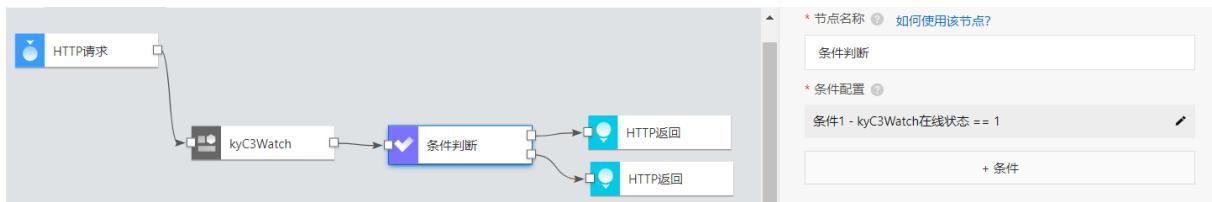


图 4.19 查询装置在线状态业务逻辑

而钉钉机器人推送跌倒警报信息的业务逻辑如下图 4.20 所示，当装置联网后且佩戴者跌倒时，设备会通过该业务逻辑触发判断，从而调用阿里钉钉机器人 API，若用户手机上安装了钉钉 APP 且加入该群聊，则会收到钉钉机器人的警报推送。

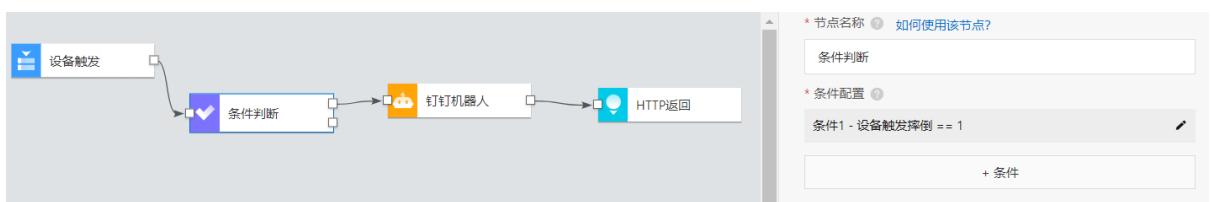


图 4.20 钉钉机器人跌倒警报推送逻辑

## 4.3 用户 APP 设计

### 4.3.1 页面交互设计

本装置的用户 APP 使用阿里云 IOT 平台的数据可视化工具搭建 WebAPP，旨在用户使用任意设备访问网页即可查看装置数据，包括当前设备在线状态、心率血氧数值、步数、温度以及是否跌倒等信息，此外还可以借助平台的大数据功能通过网络 IP 地址

定位当前装置的大致位置，点击界面上的数据卡片，可查看近半小时历史数据曲线。用户 WebAPP 界面的整体设计如下图 4.21 所示。在设计完成后，可以利用 APICloud 的 WebAPP 对其进行打包，可在移动设备上安装、点开即用，较为方便。



图 4.21 用户 WebAPP 页面整体设计

### 4.3.2 跌倒警报推送

由于跌倒警报推送采用钉钉机器人的方式，因此在使用前需要建立钉钉群聊，并开启钉钉群聊机器人，获取机器人的 Webhook 接口并填入本章 4.2.2 节业务逻辑参数中，如下图 4.22 所示。



图 4.22 机器人 Webhook 接口配置

除了推送提醒外，用户的 WebAPP 页面上也会用数据卡片直观的显示显示当前用户状态是正常或者跌倒，两种状态的界面如下图 4.23 所示。



图 4.23 正常状态与跌倒状态界面

#### 4.4 本章小结

本章先简述了装置的软件设计思路，后详细阐述了装置主要功能的实现方法、部分算法原理，包括计步测距离算法、心率血氧算法、跌倒检测算法等。后面还简述了装置屏幕的操作逻辑、服务器数据处理与业务逻辑、用户 WebAPP 的界面设计等，将整套系统完善得更加便捷、人性化。

## 第5章 系统测试与分析

### 5.1 测试目的

在系统设计完毕后，需要对各项功能进行测试，以便发现设计的缺陷与潜在问题。本装置的测试部分主要按功能分为心率血氧功能测试、计步测距功能测试、温度功能测试、跌倒报警功能测试共 4 项。

### 5.2 心率与血氧功能测试

为了测试装置获取心率、血氧值的准确性，因此该测试需要对比商用设备，并安排了两位测试者在不同状态下进行测试，以说明装置功能的普适性。但由于人体的运动对PPG波形干扰太大且具有随机性，测量只能在运动后进行，见下表 5.1。

表 5.1 心率血氧功能测试对比表

测试人员编号	测试设备	静坐时	站立时	运动后
1	商用	67BPM / 98%	73BPM / 97%	122BPM / 95%
	本装置	68.4BPM / 97.6%	78.6BPM / 96.4%	129.5BPM / 95.7%
2	商用	62BPM / 98%	69BPM / 98%	108BPM / 96%
	本装置	63.3BPM / 98.7%	68.7BPM / 98.1%	113.4BPM / 96.9 %

由表可知，该装置的测试结果在静坐时较为准确，而在站立时、运动后（站立测量）由于身体晃动，导致测试结果略有偏差。

心率血氧测试效果如下图 5.1 所示，该图的数据并未记录至表格中，但可以看到装置的测试结果与商用传感器较为接近。



图 5.1 心率血氧测试效果

此外，由于装置设计时原本仅考虑了将手指放置于传感器上获取心率，但为了装置的便捷性、无感知定时监测效果，需要将装置及传感器置于手背上，而手背的 PPG 波形变化过于微弱，这就需要对算法进行调整，于是将平均差处理放大了 8 倍，并调整了阈值，最终实现了该功能。

### 5.3 计步测距功能测试

同理，装置计步测距的结果通过与实际值（人工计步、卷尺测距）进行对比，可以得到下表 5.2。

表 5.2 计步测距功能测试对比表

实际值	10 步（慢走）/ 6.24 米	10 步（快走）/ 6.52 米	50 步（小跑）/ 34.8 米
测试结果	10 步 / 5.10 米	9 步 / 4.43 米	47 步 / 25.24 米

由表格可以看出，在慢走的情况下，装置的识别率是较为准确的；但当加快步频时，无论是计步还是测距结果，其准确性都受到了影响。推测其可能是由于 DMP 库自带的计步功能，该算法对于一步的判断阈值较高，当加快步频时往往人的手臂摆动幅度较小、身体造成的抖动较大，并不能较准确的区分是走了一步，还是由于身体抖动导致的传感器变化，而对于测距的步长模型 L，也可以通过调整固定系数的方式，使结果更接近使用者的实际值。

计步测距功能的测试效果可见下图 5.2，该图的数据并未记录至表格中，但实际快走了 20 步，其结果略有偏差。



图 5.2 计步测距功能测试效果

### 5.4 温度功能测试

对于温度测试，拟以商用红外测温枪（物体测量模式）作为标准，对传感器周围温度进行测量，在不同环境下的温度如下表 5.3 所示。

表 5.3 温度功能测试对比表

测试设备	正常放置（装置刚启动）	正常放置（启动 5 分钟后）	靠近人体（稳定 5 分钟）
测温枪	28.5°C	29.9°C	34.4°C
本装置	28.9°C	30.3°C	34.9°C

由上表可见，当装置放置一段时间后，由于电路的发热，使得传感器周边温度上升，尤其是传感器本身不能较快散热，导致测量的温度偏高，但在刚上电时、以及靠近人体一段时间稳定后，装置的测温结果与测温枪结果是比较接近的。

综上分析可以得出，对于装置电路的散热是可以再作优化的，尤其是温度传感器底部的 EP 焊盘，预计在 PCB 设计时若将其接地并在背面打孔裸露，能增强其散热性能，可以使装置测温结果更准确。

将装置刚靠近人体并使用测温枪（物体测量模式）进行测试，温度功能的测试效果如下图 5.3 所示，该图的数据并未记录至表格中，但可以看到装置的测试结果与测温枪结果较为接近。



图 5.3 温度功能测试效果

## 5.5 跌倒警报测试

对于跌倒警报测试，需要进行多次测试以衡量装置的准确性，此外还应测试跌倒检测算法是否能将正常活动、运动与跌倒区分开，因此测试了生活中常见的走路、跑步、上下楼梯、跳跃等场景，结果可见下表 5.4。

表 5.4 跌倒警报功能测试表

测试场景	走路	跑步	上楼	下楼	跳跃	跌倒
测试次（步）数	50	50	20	20	10	10
报警次数	0	2	0	1	4	8
正确率	100%	96%	100%	98%	60%	80%

可以见到，跌倒检测算法的抗干扰性还是非常好的，这主要归功于其多级检测的思想以及角度纠错的机制，因为测试过程中发现若进行较大的动作后立即平放装置也会识别为摔倒。另一方面，算法过于注重抗干扰性能反而并不能对真实的摔倒实现百分百的识别，且在剧烈运动过后立即跌倒会概率性识别失败，这是存在缺陷和隐患的，需要切换其他思路来优化跌倒检测的逻辑。

下图 5.4 是跌倒检测功能测试效果图，该图测试并未记录至表格中，可见装置有弹窗提示跌倒，手机 WebAPP 页面显示跌倒信息，手机顶部弹窗也有阿里钉钉机器人推送通知。



图 5.4 跌倒检测功能测试效果

## 5.6 本章小结

本章对设计完成的装置进行了功能性测试，通过对比测试发现了存在的问题并分析并合理推测其原因，修复了部分问题、也对部分缺陷提出了修改的思路，这为下一章的总结提供了数据支撑、对展望作出了铺垫。

## 总结与展望

### 5.7 总结

鉴于国外芯片设计企业可能采取的技术垄断，加之疫情让人们对健康生活更加重视，本课题设计了一款基于 RISC-V 架构 MCU 的穿戴式人体健康监测装置，可实时检测佩戴者的健康参数，在此基础上配合阿里云 IOT 平台实现了装置数据上传至网络、用户 WebAPP 查看数据等功能。

整套系统包含了装置和云服务器两大部分，装置可获取 PPG 传感器数据、周围温度、IMU 姿态数据，从而通过算法得到心率与血氧数值、皮肤表面温度、步数与行走距离、是否久坐以及是否跌倒。装置可通过触摸屏幕进行交互，拥有人性化的交互页面，可连接指定热点联网、同步时间。在装置联网后，数据可以通过 MQTT 上传到云服务器，用户使用 WebAPP 可以查看当前各项数据，当用户跌倒时可以通过阿里钉钉机器人进行推送提醒。

本课题完成了如下的主要工作：

(1) 根据当前的半导体行业背景、人们对可穿戴设备的需求，进行了调查并提出本课题，阐明了设计基于 RISC-V 架构 MCU 的穿戴式人体健康监测装置的意义。

(2) 对比与探究了两种不同系统架构的优劣，一种使用外挂射频单元的 MCU、另一种使用内置射频单元的 MCU，由于两者技术路线差异较大且各有优劣，最终选择了后者。

(3) 结合设计需求进行器件的选型，并使用开源工具 KiCAD 完成了相关电路设计，包括了主控芯片与模块电路、供电及充电管理电路、PPG 传感器电路、温度传感器电路、姿态传感器电路、屏幕底板及装置接口。

(4) 在 Linux 系统环境下使用乐鑫科技的 ESP-IDF 框架进行单片机软件开发，实现了对各传感器的数据读取与算法，最终完成了各项功能，并阐述了实现方法、部分算法原理等。

(5) 通过对对照实验的方式对装置及系统的各项功能进行测试，发现了系统设计潜在的问题，分析产生原因后对部分问题提出优化思路。

### 5.8 展望

本课题的设计其实还有不少可完善之处，除了测试中提到的问题及分析外，本装置离实际使用仍需要做大量的拓展，限于笔者的时间、能力原因不能使其尽善尽美，只能

简单地提出以下思路：

- (1) 对人体健康参数的获取，应当是越多越好——若能通过添加传感器，用无创的方式获取血压、呼吸波等生理数据，更能完善本装置的功能。
- (2) 对于装置的算法及功能，还有非常多的可完善之处——例如加入饮水提醒、睡眠监测、定时闹钟（添加振动马达）等等。
- (3) 对于装置各种传感器的结合运用也可以做优化——本装置只将温度与心率血氧功能作结合，在有人佩戴时定时测量心率血氧，类似的还可以添加：结合心率与 IMU 信息换算运动能量消耗；结合心率变化对跌倒检测的阈值进行动态调整，可降低剧烈运动时的误判并提高正常跌倒判断的准确率；结合温度变化综合分析健康状态等。
- (4) 云服务器端由于使用现成商业平台，其拓展性受到极大限制——比如数据导出与存储时间受限、用户 WebAPP 界面的样式及功能单调等。
- (5) 用户 WebAPP 界面未对不同设备作适应——虽然使用 Web 技术开发界面，但使用可视化平台搭建，其宽、高比例是固定的，用电脑访问的界面效果不如移动端。
- (6) 装置目前佩戴不方便——PCB 设计可以做优化，缩小体积并钻孔，可以制作相应的配套外壳、绑定带，增加美观度和佩戴舒适性。

## 参考文献

- [1] 姬晓婷.RISC-V 的进击与纠结[N].中国电子报,2022-01-18(001).
- [2] 刘畅,武延军,吴敬征,赵琛.RISC-V 指令集架构研究综述[J].软件学报,2021,32(12):3992-4024.
- [3] 种丹丹.基于 RISC-V 的开源芯片生态发展现状及未来机遇[J].中国集成电路,2021,30(08):25-30.
- [4] 赵向东,王帮德,高飞,等.新型冠状病毒肺炎疫情中可穿戴设备的应用示例与展望[J].高科技与产业化,2021,27(08):60-64.
- [5] IDTechExd.Wearable Technology Forecasts 2021-2031[R].Cambridge:IDTechEx Ltd, 2021.
- [6] Waterman A,Lee Y,Patterson D A,et al.The RISC-V instruction set manual.volume 1:user-level ISA,version 2.0[R].Defense Technical Information Center,2014.
- [7] 刘畅,武延军,吴敬征,赵琛.RISC-V 指令集架构研究综述[J].软件学报,2021,32(12):3992-4024.
- [8] 薛士然.兆易创新推出 RISC-V 架构的 GD32VF103——MCU 界的“小欢喜”[J].单片机与嵌入式系统应用,2019,19(10):93.
- [9] 谷元静,史婷奇.可穿戴设备在医学领域中的研究进展[J].全科护理,2021,19(35):4954-4958.
- [10] 李安安,石萍.人体日常健康管理可穿戴设备研究进展[J].北京生物医学工程,2021,40(04):430-436.
- [11] 全球首颗智能穿戴领域人工智能芯片发布[J].智慧城市,2019,5(10):191.
- [12] 周怡頤,凌志浩,吴勤勤.ZigBee 无线通信技术及其应用探讨 [J].自动化仪表,2005(06):5-9.
- [13] 徐侃,丁强.一种基于 MQTT 协议的物联网通信网关[J].仪表技术,2019(01):1-4+43.
- [14] 吴建辉,茅洁.射频电路 PCB 设计[J].电子工艺技术,2003(01):19-21+26.
- [15] 刘雷,慕艳艳,刘睿鑫.基于三轴加速度传感器的步长估算模型研究[J].传感器与微系统,2017,36(08):22-24+28.
- [16] 郭元新,叶玮琼.基于 MPU6050 传感器的跌倒检测算法[J].湖南工业大学学报,2018,32(03):76-80.
- [17] 王文森.变异系数——一个衡量离散程度简单而有用的统计指标[J].中国统

- 计,2007(06):41-42.
- [18] Raivis Strogonovs.Implementing pulse oximeter using MAX30100[EB/OL].(2017-08-03)[2022-05-08].<https://morf.lv/implementing-pulse-oximeter-using-max30100>.
- [19] wikipedia.Digital filter[EB/OL].(2021-11-25)[2022-05-10].[https://en.wikipedia.org/wiki/Digital\\_filter](https://en.wikipedia.org/wiki/Digital_filter).

## 致谢

## 附录 A 电路原理图

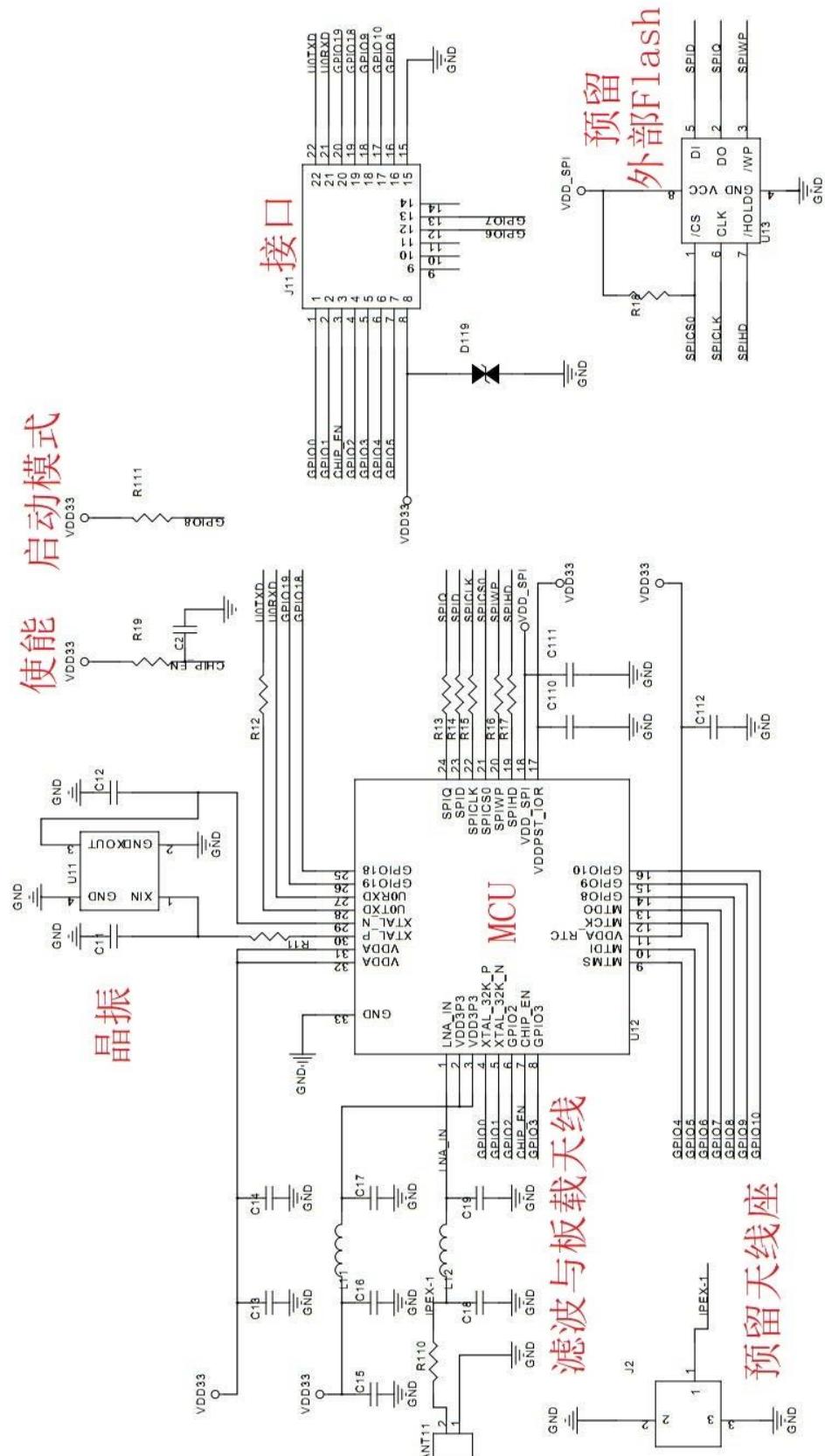


图 A1 主控模块内部电路原理图

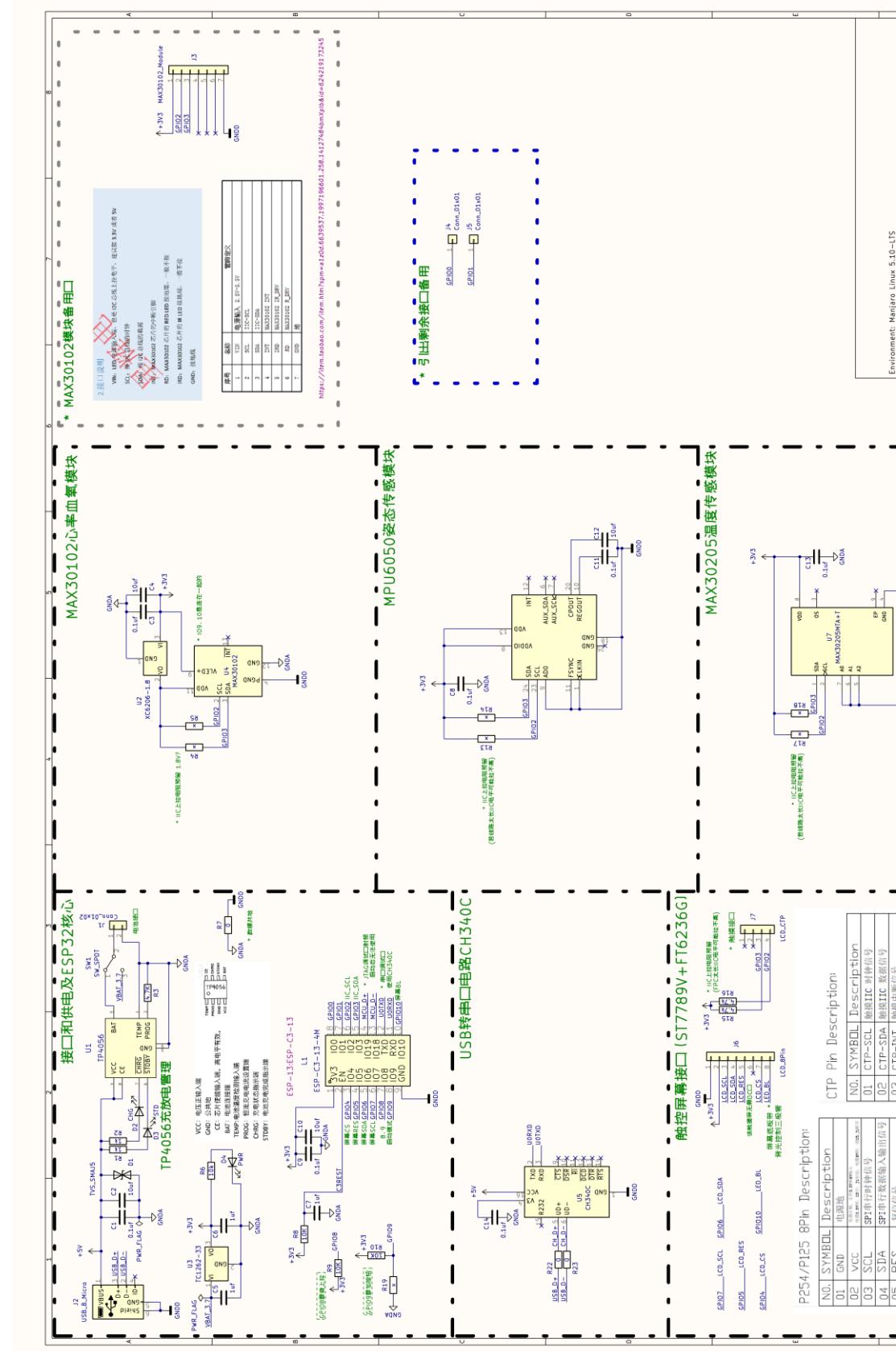


图 A2 装置电路全局原理图

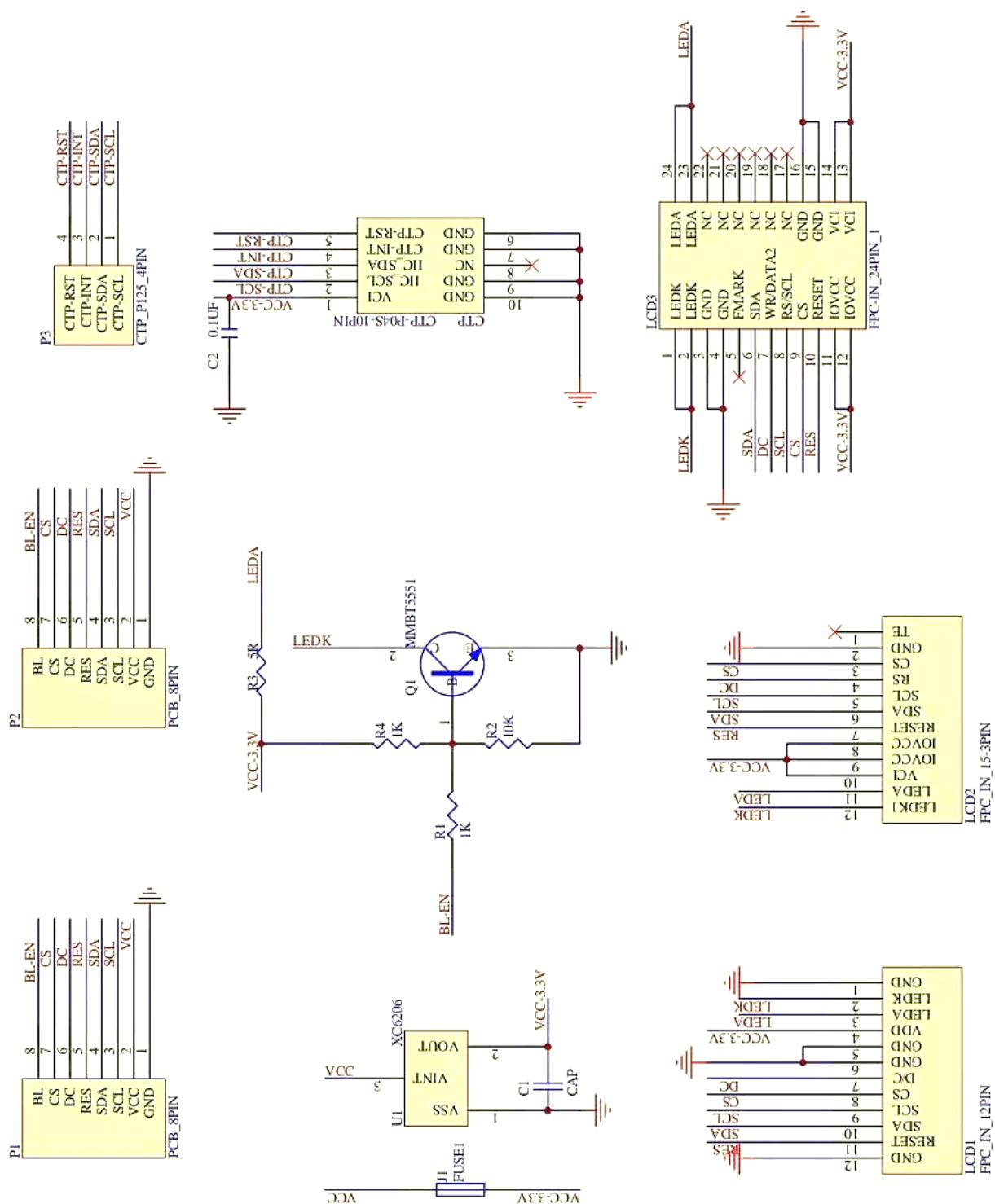


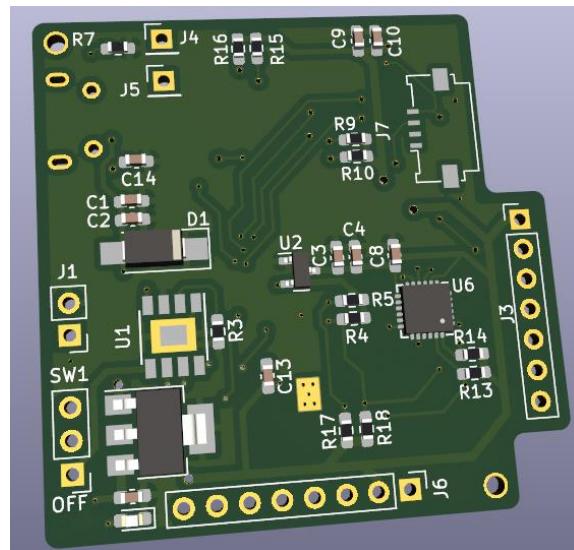
图 A3 屏幕底板原理图

## 附录 B PCB 图

(a) PCB 顶层图

(b) PCB 底层图

图 B1 装置 PCB 图



(a) PCB 顶层三视角图

(b) PCB 底层三视角图

图 B2 装置 PCB 三维图

## 附录 C 实物图



(a) 实物顶层视角图

(b) 实物底层视角图

图 C1 装置实物电路板图



图 C2 装置组装后实物图



(a) 时间页面

(b) 选择 APP 页面

(c) 联网页面

图 C3 装置运行图



图 C4 WebAPP 运行图

## 附录 D 源程序

```

/*
 * APPLICATION MAIN
 ****
void app_main() {
    /* If you want to use a task to create the graphic, you NEED to create a Pinned task
     * Otherwise there can be problem such as memory corruption and so on.
     * NOTE: When not using Wi-Fi nor Bluetooth you can pin the guiTask to core 0 */
    gpio_pad_select_gpio(ST7789_BCKL); // BCKL OFF when boot
    gpio_set_direction(ST7789_BCKL, GPIO_MODE_OUTPUT);//
    gpio_set_level(ST7789_BCKL, 0); //
    xTaskCreatePinnedToCore(guiTask, "gui", 4096*13, NULL, 0, NULL, 0);
}

/*
 **** Sensor Init ****
int ret = MAX30205_Init();
if(ret==ESP_OK){
    printf("MAX30205_Init Successful!\n");
} else{
    printf("MAX30205_Init Error!\n");
}
ret = mpu_dmp_init();
if(ret==0){
    printf("MPU6050_Init Successful!\n");
} else{
    printf("MPU6050_Init Error!\n");
}
ESP_ERROR_CHECK(max30102_init( &max30102, I2C_NUM_0,
    MAX30102_DEFAULT_OPERATING_MODE,
    MAX30102_DEFAULT_SAMPLING_RATE,
    MAX30102_DEFAULT_LED_PULSE_WIDTH,
    MAX30102_DEFAULT_IR_LED_CURRENT,
    MAX30102_DEFAULT_START_RED_LED_CURRENT,
    MAX30102_DEFAULT_MEAN_FILTER_SIZE,
    MAX30102_DEFAULT_PULSE_BPM_SAMPLE_SIZE,
    MAX30102_DEFAULT_ADC_RANGE,
    MAX30102_DEFAULT_SAMPLE_AVERAGING,
    MAX30102_DEFAULT_ROLL_OVER,
    MAX30102_DEFAULT_ALMOST_FULL,
    false ));

    gpio_pad_select_gpio(ST7789_BCKL); // BCKL ON
    gpio_set_direction(ST7789_BCKL, GPIO_MODE_OUTPUT);//
    gpio_set_level(ST7789_BCKL, 1); //
//Shutdown MAX30102 after boot
max30102_set_mode(&max30102, MAX30102_SHDN_ON);

lv_task_t * sec_task = lv_task_create(lv_sec_task, 1000, LV_TASK_PRIO_MID,NULL);
lv_task_t * xms_task = lv_task_create(lv_xms_task, 10, LV_TASK_PRIO_LOW,NULL);

if(ret != 0){
    static const char * notify_btns[] = {"Close", ""};
    guider_ui.notify_msgbox = lv_msgbox_create(lv_scr_act(), NULL);
    lv_msgbox_set_text(guider_ui.notify_msgbox, "Init Error, pls reboot!");
    lv_msgbox_add_btns(guider_ui.notify_msgbox, notify_btns);
    lv_obj_set_width(guider_ui.notify_msgbox, 200);
    lv_obj_align(guider_ui.notify_msgbox, NULL, LV_ALIGN_CENTER, 0, 0);
}

```

图 D1 主程序及初始化函数

```

void lv_sec_task(lv_task_t * sec_task){
    if(time_sec<59){
        time_sec++;
    }else{
        time_sec = 0;
        if(time_min<59){
            time_min++;
        }else{
            time_min=0;
            if(time_hour<23){
                time_hour++;
            }else{time_hour=0;}
        }
        if((lv_scr_act() == guider_ui.Timehome) && scrloading==0){
            Timeset_func(&guider_ui,time_hour,time_min,time_day);//display update every min
        }
    }
    if(time_sync_flag && (lv_scr_act() == guider_ui.Timehome) && scrloading==0)//sync time now
    {
        Timeset_func(&guider_ui,time_hour,time_min,time_day);
        time_sync_flag=0;
    }
    if(scrloading==0){
        //tempset
        temp_value = (float)GetTemperature();
        if(temp_value > 30.9)
        {
            people_flag=1;
        }else people_flag=0;
        Tempset_func(&guider_ui,temp_value);
        //stepset
        // dmp_get_pedometer_step_count(&step_count);
        Stepset_func(&guider_ui,step_count,stepDis);
        //HRset
        HR_func(&guider_ui,result.heart_bpm,result.sp02);
        //memory_print();
    }
    if (time_min%5==0 && time_sec==0){
        max30102_set_mode(&max30102, MAX30102_SHDN_OFF);//start max30102
        HR_TIM_CNT=0;
    }
}

```

图 D2 秒级定时任务函数

```

void lv_mqtt_task(lv_task_t * task)
{
    char pub_payload_c3data[150];
    sprintf(pub_payload_c3data,"(\"method\":\"thing.event.property.post\", \"params\":{\"Temp\":%2.1f,\"Step\":%d,\"HR\":%2.1f,\"SpO2\":%2.1f,\"Fallen\":%d} )",temp_value,step_count,result.heart_bpm,result.sp02,Fallen);
    esp_mqtt_client_publish(task->user_data, "/sys/elfpfa35v02/demo1/thing/event/property/post", pub_payload_c3data, 0, 0, 0);
    //printf("mem=%d\n",esp_get_free_heap_size());
}
// static uint8_t time_count =0;
if(Fallen)
{
    // time_count++;
    // if(time_count>6){
    //     time_count=0;
    //     Fallen=0;
    // }
}

```

图 D3 定时 MQTT 上传任务函数

```

void lv_xms_task(lv_task_t * xms_task){
    static unsigned long beforeStepC=0;
    static int beforeTickC=0;
    // printf("MAX: %f\n", result.ir_cardiogram);
    // if(result.pulse_detected) {
    //     printf("----\n");
    //     printf("BPM: %f | SpO2: %f%%\n", result.heart_bpm, result.sp02);
    // }
    /*----- MAX30102 UPDATE -----*/
    if(lv_scr_act() == guider_ui.APPtoHR){ //when HR_APP page now
        if(lv_debug_check_obj_valid(guider_ui.APPtoHR_chart_HeartBeep))
        {
            max30102_update(&max30102, &result);
            lv_chart_set_next(guider_ui.APPtoHR_chart_HeartBeep,APPtoHR_chart_HeartBeep_0,result.ir_cardiogram);
        }
    }else if (people_flag == 1 && HR_TIM_CNT<3000){//if people wearing, keep data updating periodically
        max30102_update(&max30102, &result);
        HR_TIM_CNT++;
        if(HR_TIM_CNT==2999) max30102_set_mode(&max30102, MAX30102_SHDN_ON);/*shutdown max30102*/
    }else{HR_TIM_CNT=8000;}
    /*----- STEPS & FALLDETECT TASK -----*/
    MPU_Get_Accelerometer(&ax,&ay,&az);
    float svm_v = sqrt((ax/16384.0)*(ax/16384.0) + (ay/16384.0)*(ay/16384.0) + (az/16384.0)*(az/16384.0));
    dmp_get_pedometer_step_count(&step_count);
    if(step_count != beforeStepC){
        stepF = 1/(((xTaskGetTickCount()-beforeTickC)*portTICK_PERIOD_MS)*0.001);
        printf("stepF=%f\r\n",stepF);
        beforeTickC=xTaskGetTickCount();
        stepDis = stepDis+(0.132*svm_v+0.123*stepF+0.225)*(step_count-beforeStepC);
        beforeStepC=step_count;
    }
    static uint8_t Div2=0;
    if(Div2==9){
        if(SVM_FallenDet(svm_v,az)==1)
        {
            Fallen=true;
            if(!lv_debug_check_obj_valid(guider_ui.notify_msgbox))
            {
                static const char * notify_btns[] = {"Close", ""};
                guider_ui.notify_msgbox = lv_msgbox_create(lv_scr_act(), NULL);
                lv_msgbox_set_text(guider_ui.notify_msgbox, "Fall Detected!");
                lv_msgbox_add_btns(guider_ui.notify_msgbox, notify_btns);
                lv_obj_set_width(guider_ui.notify_msgbox, 200);
                lv_obj_align(guider_ui.notify_msgbox, NULL, LV_ALIGN_CENTER, 0, 0);
            }
            Div2=0;
        }else{Div2++;}
        // printf("FALLEN=%d\n",SVM_FallenDet());
    }
}

```

图 D4 毫秒级定时任务函数

```

uint8_t SVM_FallenDet(float SVM_v, int16_t az)
{
    static uint8_t SVM_P = 0,CV_P = 0;
    static float SVM_BUF[SVM_Buf_size];
    static uint8_t CV_BUF[CV_Buf_size];
    bool CV_flag=0,Fall_flag=0;
    float SUM=0,SD=0;
    SVM_BUF[SVM_P] = SVM_v ;//sqrt((ax/16384.0)*(ax/16384.0) + (ay/16384.0)*(ay/16384.0) + (az/16384.0)*(az/16384.0));
    if (SVM_BUF[SVM_P] > 1.8)
    {
        for(int i = 0; i < SVM_P+1;i++)
        {
            SUM += SVM_BUF[i];
        }
        SUM = SUM / (SVM_P+1); //now SUM is AVG

        for (int j = 0;j < SVM_P+1;j++)
        {
            SD += pow((SVM_BUF[j]-SUM),2);
        }
        SD = sqrt(SD / (SVM_P+1));
        if ((SD / SUM) > 0.30)
        {
            CV_flag=1;
        }else{CV_flag=0;}
    }else{CV_flag=0;}
    /*----- MUST DO For Next -----*/
    CV_BUF[CV_P] = CV_flag;
    if (SVM_P == SVM_Buf_size-1)
    {
        for (int k = 0; k < SVM_Buf_size-1; k++)
        {
            SVM_BUF[k]=SVM_BUF[k+1];
        }
    }else{SVM_P++;}
    if (CV_P == SVM_Buf_size-1)
    {
        CV_P =0;
        SUM=0;
        for (int L = 0; L < SVM_Buf_size-1; L++)
        {
            SUM += CV_BUF[L];
            CV_BUF[L]=0;
        }
        if (SUM == 1){
            Fall_flag=1;
        }
    }else{CV_P++;}
    if(Fall_flag && az > 10000)//az~<65^o
    {
        return 1;
    }else{return 0;}
}

```

图 D5 跌倒检测算法函数

```

case MAX30102_PULSE_TRACE_UP:
    if(sensor_value > prev_sensor_value) {
        current_beat = (uint32_t)(xTaskGetTickCount() * portTICK_PERIOD_MS);
        this->last_beat_threshold = sensor_value;
    } else {
        if(this->debug) {
            printf("Peak reached: %f %f\n",
                   sensor_value,
                   prev_sensor_value);
        }

        uint32_t beat_duration = current_beat - last_beat;
        last_beat = current_beat;

        float raw_bpm = 0;
        if(beat_duration)
            raw_bpm = 60000.0 / (float)beat_duration;

        if(this->debug){
            printf("Beat duration: %u\n", beat_duration);
            printf("Raw BPM: %f\n", raw_bpm);
        }
    }
}

```

图 D6 心率算法核心

```

this->current_sp02 = 104.0 - 17 * ratio_rms;
// -45.060*R*R + 30.354 *R + 94.845
//this->current_sp02 = -45.060*ratio_rms*ratio_rms + 30.354 * ratio_rms + 94.845;
if(this->current_sp02<99.9){
    data->sp02 = this->current_sp02;
}
else{
    data->sp02 = 99.9;
}

```

图 D7 血氧算法核心

```

void max30102_lpb_filter( max30102_config_t* this, float x )
{
    this->lpb_filter_ir.v[0] = this->lpb_filter_ir.v[1];

    // //Fs = 100Hz and Fc = 10Hz
    this->lpb_filter_ir.v[1] = (2.452372752527856026e-1 * x) +
        ( 0.50952544949442879485 * this->lpb_filter_ir.v[0] );

    // //Low pass chebyshev filter order=1 (Fs=100 Fc=8 HZ ripple=-6db)
    // this->lpb_filter_ir.v[1] = (1.294569192014242698e-1 * x) +
    //                               (0.74108616159715146043 * this->lpb_filter_ir.v[0]);

    // //Fs = 100Hz and Fc = 5Hz
    // this->lpb_filter_ir.v[1] = (1.367287359973195227e-1 * x)
    //                           + (0.72654252800536101020 * this->lpb_filter_ir.v[0]);
    // //Very precise butterworth filter

    this->lpb_filter_ir.result = (this->lpb_filter_ir.v[0] + this->lpb_filter_ir.v[1]);
}

```

图 D8 滤波器系数及函数