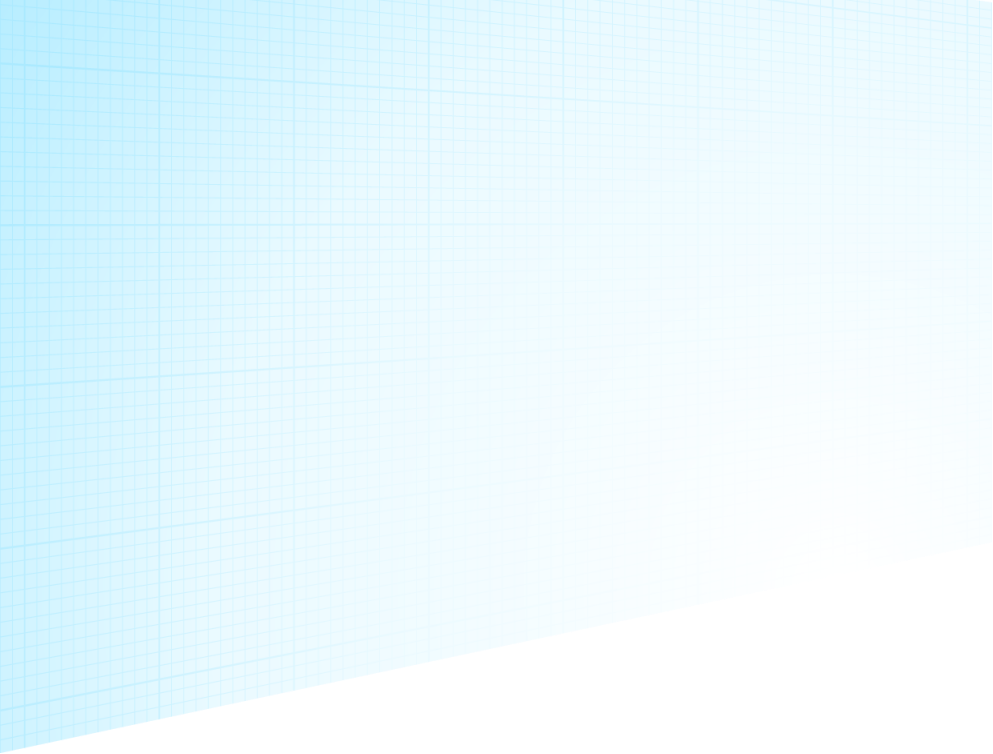
****

**基于物联网的智能心率健康检测系统**

**物联网感知与传输生产实习**

**个人报告**

|  |  |  |  |
| --- | --- | --- | --- |
| **文件编号：物联网生产实习** | | **生效日期：2025 年 1 月 03日** | |
| **编制： 夏书淇（组长）**  **日期：2025-1-3** | **审核：周思语(组员)**  **日期：2025-1-3** | | **批准：韩鑫（组员）**  **日期：2025-1-3** |

**物联网工程2201班 夏书淇 2022003725**

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布  [ ] 正在修改 | 文件标识： | 物联网2201-夏书淇个人报告 |
| 当前版本： | v0.0.0.0.02 |
| 作 者： | 夏书淇 |
| 完成日期： | 2025年01月03日 |

**目录**

[1. 任务概述 5](#_Toc24807)

[1.1. 生产实习分工 5](#_Toc1929)

[1.2. 个人工作 5](#_Toc3011)

[1.3. 目标 7](#_Toc14500)

[1.4. 系统的要求及特点 7](#_Toc24448)

[2. 基于物联网的智能心率健康检测系统需求分析 8](#_Toc5092)

[2.1. 系统要求 8](#_Toc10114)

[2.2. 系统功能目标 8](#_Toc5946)

[2.3. 需求概述 8](#_Toc20879)

[2.3.1. 功能描述 8](#_Toc10883)

[2.3.2. 系统界面需求 9](#_Toc23161)

[2.4. 结构化分析 9](#_Toc25107)

[2.4.1. 数据需求 9](#_Toc27905)

[2.4.2. 层次方框图 10](#_Toc28277)

[2.4.3. 数据流图 10](#_Toc11534)

[2.4.4. 数据字典 10](#_Toc27480)

[2.5. 其他需求 11](#_Toc14700)

[2.5.1. 数据库需求 11](#_Toc3780)

[2.5.2. 安全需求 11](#_Toc175)

[2.5.3. 性能需求 11](#_Toc775)

[3. 基于物联网的智能心率健康检测系统总体设计 12](#_Toc10728)

[3.1. 总体设计的目标和任务 12](#_Toc18682)

[3.2. 需求规定 12](#_Toc6225)

[3.2.1. 功能 12](#_Toc11670)

[3.2.2. 性能 12](#_Toc2336)

[3.2.3. 输入数据 12](#_Toc29924)

[3.2.4. 输出数据 13](#_Toc1308)

[3.3. 系统硬件选型及设计 13](#_Toc11737)

[3.3.1. 系统硬件介绍 13](#_Toc27561)

[3.4. 软件结构设计 14](#_Toc32150)

[3.4.1. 模块设计说明 14](#_Toc19300)

[3.4.2. 模块结构图设计 14](#_Toc26149)

[3.5. 程序运行方式 14](#_Toc7314)

[3.5.1. 数据库 14](#_Toc27567)

[3.5.2. E-R图（概念结构设计） 15](#_Toc21468)

[3.5.3. 逻辑结构设计 15](#_Toc21686)

[4. 基于物联网的智能心率健康检测系统详细设计 17](#_Toc14396)

[4.1. 详细设计的目的和任务 17](#_Toc25823)

[4.1.1. 详细设计目的 17](#_Toc32006)

[4.1.2. 详细设计任务 17](#_Toc12110)

[4.2. 结构化程序设计 17](#_Toc26987)

[4.2.1. 检测配置初始化模块 18](#_Toc17574)

[5. 基于物联网的智能心率健康检测系统编码 19](#_Toc24814)

[5.1. 程序设计语言 19](#_Toc25119)

[5.1.1. 串口通信 19](#_Toc31902)

[5.1.2. 数据库SQL 19](#_Toc12894)

[5.1.3. 前端 19](#_Toc8984)

[5.1.4. 硬件 19](#_Toc30187)

[5.2. 系统开发环境搭建 19](#_Toc15772)

[5.3. 源代码目录结构说明 20](#_Toc22651)

[5.4. 源程序编写 20](#_Toc23223)

[5.4.1. 串口通信代码编写 20](#_Toc8554)

[5.4.2. 实时绘图代码编写 21](#_Toc21328)

[5.4.3. 数据库SQL代码编写 21](#_Toc8466)

[5.4.4. Streamlit代码编写 22](#_Toc27935)

[5.4.5. 接口函数编写 27](#_Toc2540)

[5.4.6. 数据库功能代码编写 28](#_Toc3542)

[5.4.7. 大模型Agent代码编写 29](#_Toc7064)

[5.4.8. 硬件代码编写 33](#_Toc27773)

[6. 基于物联网的智能心率健康检测系统测试 34](#_Toc30527)

[6.1. 软件测试目的 34](#_Toc23254)

[6.2. 软件测试方法 34](#_Toc28653)

[6.3. 单元测试 34](#_Toc20166)

[6.4. 集成测试 34](#_Toc32467)

[6.5. 测试结论 34](#_Toc21521)

[7. 基于物联网的智能心率健康检测系统总结 35](#_Toc664)

[7.1. 个人总结 35](#_Toc4333)

[7.2. 改进的地方 35](#_Toc27164)

[7.3. 致谢 35](#_Toc7225)

# 

# 任务概述

## **生产实习分工**

在当今物联网技术飞速发展的时代背景下，智慧医疗作为物联网应用的重要领域之一，正以前所未有的速度改变着传统医疗行业的面貌。本次《物联网感知与传输生产实习》生产实习中，我们团队致力于开发一款基于物联网的智慧心率检测系统，旨在通过先进的物联网技术，实现对人体心率的实时监测与数据传输，以及结合可视化手段以及当下流行的大语言模型Agent框架为智慧心率这一方向实现我们的设计想法。该系统融合了智能传感器技术、射频识别技术、数据库系统原理以及Python语言程序设计、大语言模型Agent思维链等多学科知识。

## **个人工作**

在项目的实施过程中，我们团队充分发挥了每位成员的专业特长和团队协作能力，形成了高效、有序的工作模式。我所参与的实习具体分工如下：

**确定成员：**作为队长，在第一天，17周周四老师布置完生产实习的内容以后我便定下了组员：韩鑫和周思语，一方面是我们在传感器实验的时候也是一组、软工实验的时候也是一组，团队配合默契。

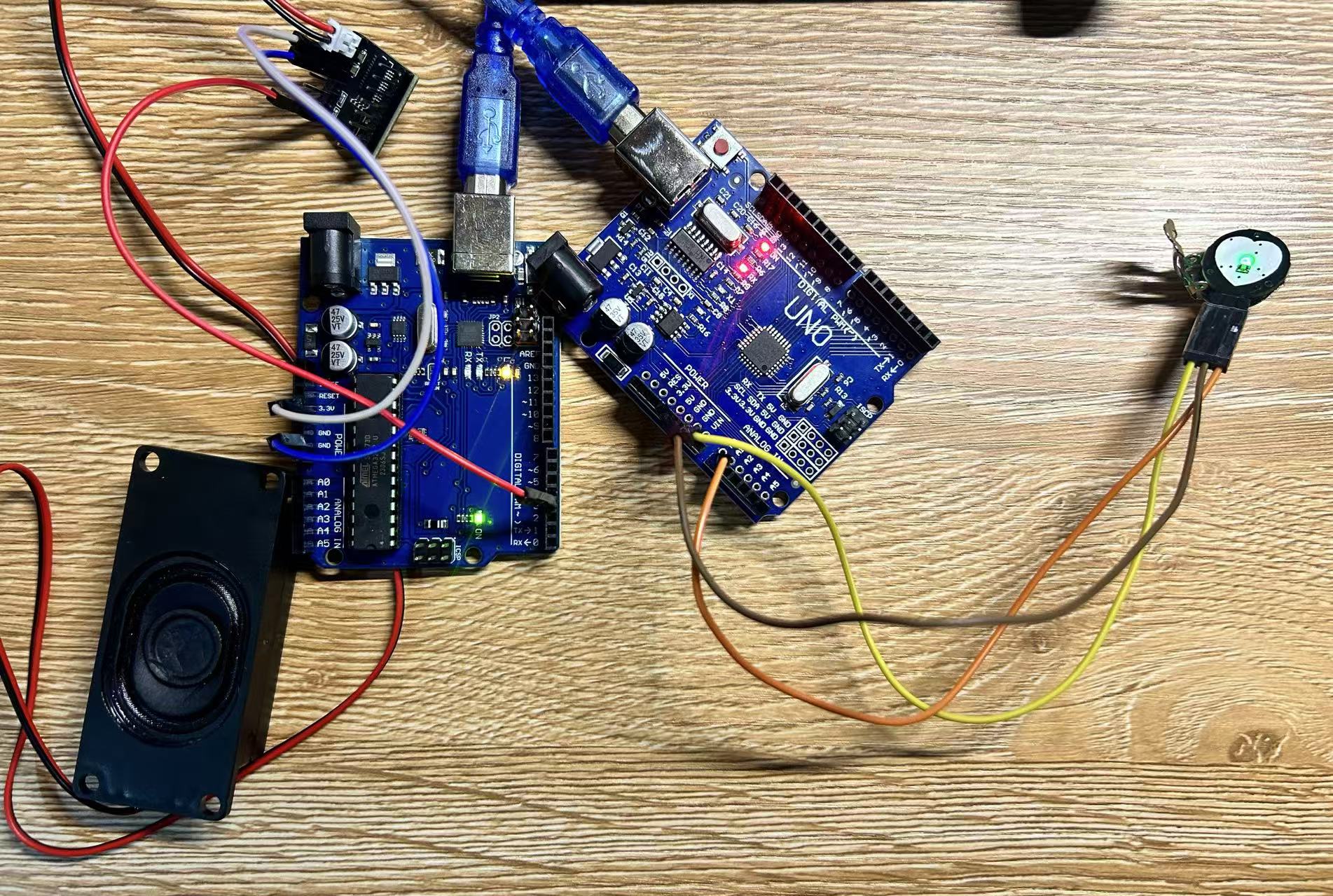
**确定选题（可行性分析）：**我对生产实习指导书中的选题和对应的要求进行查看后觉得大部分选题比较没有心意，同时E308实验室的物联网设备不支持其中的一部分选题，因此在个人能力足够的前提下我想开新的生产实习选题。考虑到课外在实验室做的内容是和医疗与大语言模型相关的工作，因此我想将课上所学的传感器部分知识、智慧医疗、大语言模型Agent相关知识进行有机结合，开发基于物联网的智能心率健康检测系统作为实习的选题。17周周5早上我和组员一起讨论了从硬件选型、烧录代码、程序语言、系统功能、数据库设计等多个部分的可行性以后，最终确定了本选题。本选题的所有编码程序设计（部分硬件烧录代码除外）均有小组三人独立完成、不涉及抄代码、抄袭等情况。同时，所有的系统构思、模块设计、语言架构、功能实现均不参考各种类似已实现的作品，后续我们全部的代码我将在GitHub上开源公共仓库上传，并保持代码优化更新，补充基于STM32的实现方案。运行代码所需的环境、Python库等一系列操作流程我将在后文编码部分详细描述，以下是仓库的地址（程序优化后我将上传完整工作代码，使用Git Clone前你需要在本地安装Git版本控制工具，在Git Bash终端输入下述命令即可，不建议使用download zip的方式进行下载）：

Git Clone <https://github.com/Qi2212/PulseSensor_Visualization.git> (仓库地址)

**硬件选型：**17周周5上午，我和组员确认选题以后，我和我们组的硬件专家韩鑫进行了设备的选型讨论。考虑到STM32需要一定的基础，上手困难，我们选择了Pulse sensor官方使用Arduino uno R3开发板作为我们的嵌入式微控制器，确定了使用串口通信作为我们的设备和上位机（笔记本）的通信方式，完成了硬件和软件部分的选型和通信设计。购置了Pulse Sensor传感器\*4(一个套件、三个传感器，￥42)，Arduino uno R3\*2(国产版，￥40)，语音控制模块(JQ8900-16P，￥21)，USB\*2（实验室的设备老有问题，我买了两根）参考连线设计方案如下：



**图1.1 传感器、下位机、上位机通信设计**



**图1.2 硬件采购实物图**

**串口通信demo代码编写:**

通过对Python库进行搜索，了解到PySerial作为Python的嵌入式自动化测试工具可以进行串口通信，与我们的方案设计符合，最终敲定PySerial作为编码通信的包。我的Python技术较好，我阅读文档

https://pyserial.readthedocs.io/en/latest/以后进行了串口通信的demo代码编写，并将demo代码交给周思语进行绘图部分的编码参考

**数据库模块代码编写：**我有丰富的Python后端代码经验，对于数据库的存储模块非常熟练，这部分我完成了主要的工作，从数据库逻辑设计到用PyMySQL进行采集数据的存储和可视化展示均参与完成。

**Streamlit前端及数据可视化代码编写：**我在课外的大模型工作的时候了解到这个专注数据可视化、大模型展示的Python Web框架，查看了官方文档、接口简单、非常适合我这种前端小白，因此我也参与绝大部分的前端编码和用户界面设计。

**大语言模型Agent代码编写：**我将课外实验室的课题方向大语言模型的Agent融入了本次实习设计中，完成了Agent的Function calling的流程，包括函数体的编写，tools脚本的编写，stream流式输出的实现，streamlit的session记忆实现的全部编码，可以实现大模型根据自己的理解去调用编写好的SQL Agent工具查询最近一次的检测记录数据并基于自身的知识进行回答用户，降低了平台用户的医疗知识依赖性。这部分的算法流程、编码实现将在后文详细阐述。

**反馈模块的代码编写：**软件部分考虑到一个用户反馈功能，额外增加了这个小功能，包括格式的校验、信息的校验、以及数据库的存储等实现了反馈模块。

**系统测试：**编码完成后，我合并了组员的全部代码，组装了代码，和组员对每个模型进行了单元测试、集成测试，确保了答辩时的所有模块、系统流程的完美演示~

**撰写报告：**和组员一起撰写报告、我主要负责可行性分析部分的系统流程图、结构化设计的有关算法程序流程图、详细设计的数据库部分以及测试部分的全部内容，最终定稿。

综上，为我这次参与物联网感知与传输生产实习的全部工作，后文将对我参与的部分进行阐述，需求分析、总体设计、详细设计、测试部分的内容请老师详见我们组的打报告。

## **目标**

随着人们对健康关注度的不断提升，尤其是心血管疾病的高发，实时、精准地监测心率变化对于预防疾病、保障健康至关重要。本基于物联网的智能心率健康检测系统旨在利用先进的物联网技术与医疗传感技术，打破传统心率监测的时空限制，为用户提供便捷、持续、可靠的心率监测服务，助力个人健康管理。本系统在心率监测功能模块上1具备高度自主性。从传感器驱动、数据采集到初步处理，均由自主研发的软件代码实现，不依赖外部心率监测专用软件，能够独立完成核心的心率检测与基础数据分析任务。

面向普通大众，无论是日常健身爱好者、上班族还是老年人等关注健康的人群，均可随时随地使用该系统监测心率。在运动过程中，实时了解心率变化，确保运动强度适宜，避免过度运动引发健康问题；日常生活里，持续追踪心率趋势，能早期发现心率异常波动，如因疲劳、压力等因素导致的心率变化，及时调整生活方式，预防疾病发生。

## **系统的要求及特点**

系统实时收集和显示心率数据，通过串口与传感器进行通信，确保数据的及时性和准确性。采用动态图表（如Matplotlib）展示心率变化，用户可直观了解健康状况，便于分析与决策。除了实时监测，系统还支持心率数据的历史记录及分析，用户可以查看特定时间段内的心率变化趋势。系统提供简洁直观的用户界面，便于各年龄层用户操作，不需要专业的技术背景，用户可以在日常生活中定期 (如每天多次) 使用该系统进行心率监测，特别是在运动、休息和睡眠等特定时间段。同时，系统提供给历史检测数据的查看，检测使用者的测量记录并进行一定的统计，对测量结果提供一定的解读方案。

# **基于物联网的智能心率健康检测系统需求分析**

## **系统要求**

系统应具备高度的可靠性，确保在各种复杂环境下稳定运行，心率监测数据准确无误，数据传输不中断、不丢失。无论是日常使用还是极端条件下（如高温、高湿、强电磁干扰等），都能持续为用户提供可靠的健康监测服务。

操作流程简洁明了，无论是专业医护人员还是普通大众，都能轻松上手。界面设计友好，采用直观的图形化界面展示心率数据、分析结果等信息，减少用户学习成本，引导用户快速完成各项操作，如设备连接、参数设置、查看报告等。

软件架构采用模块化设计，各个功能模块独立封装，便于后期的升级、优化与故障排查。系统提供在线用户反馈功能，使用者在系统留言后即可被技术人员查看并进行相应的系统改进和维护。

硬件方面，心率监测终端需适配多种主流智能穿戴设备平台以及常见的移动终端设备；软件方面，Web端应用利用http/https协议可在多种浏览器如edge、Chrome、FireFox等主流浏览器进行使用，可以确保不同用户群体在使用不同的浏览器时，系统都能正常运行，展现一致的功能与性能。同时基于http/https的传输协议应具有良好的可移植性，用户在手机/电脑设备都能访问使用，并且前端页面具有良好的适配。

## **系统功能目标**

整个系统的软件界面，包括心率监测终端的显示屏界面（若有）、移动端应用界面以及网页端管理界面，应遵循统一的设计风格与布局规范。采用相似的色彩搭配、图标样式、字体大小等元素，使用户在不同终端操作时能够快速适应，减少认知混淆，提升操作效率。

当系统出现各类错误情况，如设备连接失败、数据传输异常、心率数据超出正常范围等，发出统一的、辨识度高的错误声音提示，提醒用户及时关注系统状态。同时，在界面上明确显示错误信息详情，指导用户采取相应的解决措施。

可接入时下流行的AI大语言大模型，内置在线帮助文档与教程，涵盖系统的基本使用方法、常见问题解答等内容。用户在使用过程中遇到问题时，可通过专门的反馈界面进行反馈技术人员改进或提供实时的在线技术支持窗口帮助用户更好的使用产品。

## **需求概述**

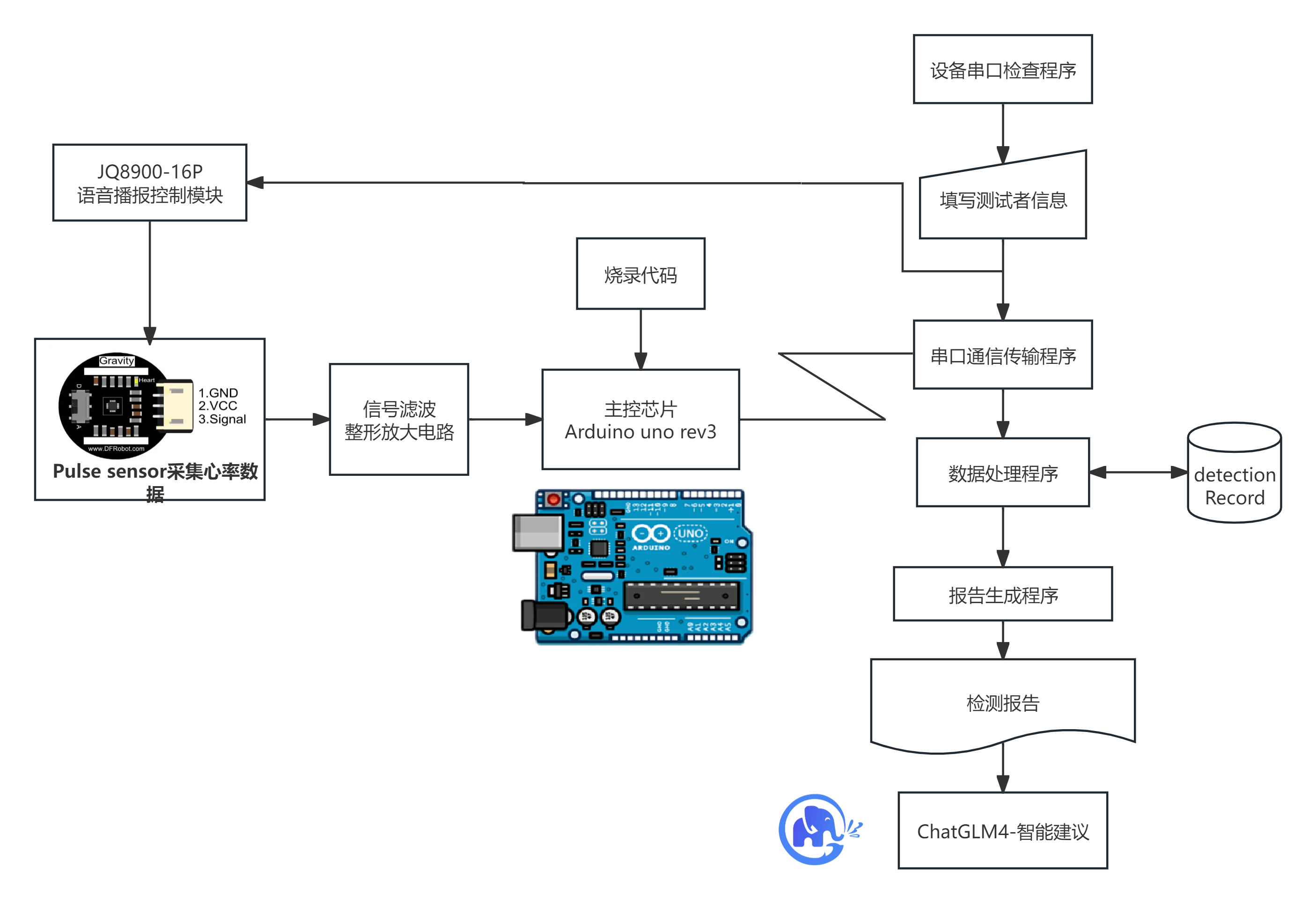
### 功能描述

通过高精度心率传感器实时采集人体心率数据，测量范围覆盖常见心率区间。支持连续监测模式，自动适应人体不同状态（运动、睡眠、静止等），稳定输出心率数值，并以设定的频率（如每 5 秒更新一次）传输数据。

在云端服务器或本地终端（根据设备性能与网络条件）运行智能算法，对心率数据进行深度分析。构建安全可靠的数据库系统，存储海量用户心率数据及相关个人信息。实时接收数据，绘制动态心率曲线图，随后将数据存入数据库，再绘制出心率值和脉搏值的静态曲线图。

移动端应用支持与心率监测终端快速配对连接，连接过程自动化、简单化，耗时短。用户可在移动端便捷地查看实时心率、历史数据、健康报告，进行系统参数设置（如监测频率、预警阈值等），增加用户体验。此外，交互功能还应涵盖与 AI大语言大模型的问答交互界面，设计简洁易用的输入框与展示区域，方便用户提问与查看回答，提升用户体验。

AI大语言大模型，实现数据分析结束后，用户可结合心率脉搏测试数据进行深入提问，例如询问当前心率异常波动可能对应的潜在健康风险、近期心率变化趋势反映出的身体状态改变、基于个人历史数据给出特定场景下（如运动、工作压力大时）的心率管理策略等。AI大语言大模型将基于海量医疗知识、大数据分析以及深度学习算法，实时理解用户问题意图，快速给出专业、准确且通俗易懂的回答，辅助用户更深入了解自身健康状况。



**图2.1 智能心率健康检测系统-软硬件系统流程图**

### 系统界面需求

移动端界面采用简洁直观的设计风格，以大字体、鲜明色彩突出显示实时心率数值，方便用户一眼看清。心率曲线以折线图形式呈现，可缩放、平移查看历史趋势。设置明显的预警提示区域，当有预警信息时醒目显示。参数设置页面布局合理，操作按钮易于点击。在与AI大语言大模型交互的区域，设计专门的问答界面，输入框提示清晰，回答展示区域排版美观，便于用户阅读。

网页显示屏，四个步骤：选择传感器；设备就绪检测；开始检测（正确完成初始化相关配置）；展示结果。显示出实时测试的心率曲线图、存入数据库的最终测试结果、计算出来的平均值以及结合数据库的最终静态心率和脉搏折线图。

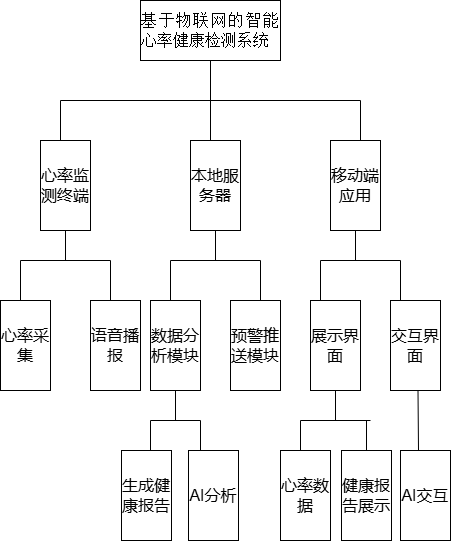
## **结构化分析**

### 数据需求

心率数据：记录每次心率测量的具体数值、测量时间、测量时人体状态（如运动、休息、睡眠等），以时间序列形式存储，便于后续分析与绘制曲线。同时，为了支持 AI大语言大模型的深度分析，还需存储心率数据的变化速率、波动幅度等衍生数据，为用户提问提供更丰富的数据基础。

历史检测报告数据：由数据库提供历史数据支持，包含心率趋势总结、异常情况分析、健康建议等文本内容，以及相关图表（如心率实时折线图、脉搏图等），方便用户查看与分析。同时，健康报告应具备动态更新功能，随着用户新数据的产生以及与 AI大语言大模型的交互反馈，及时优化报告内容。

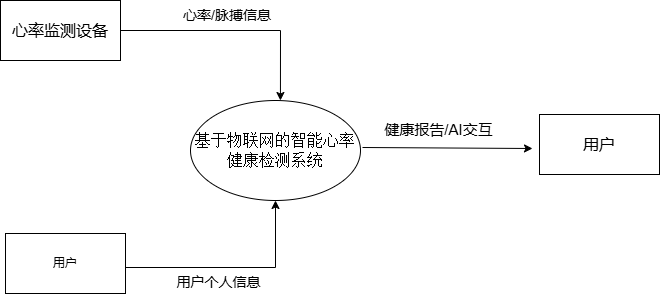
### 层次方框图



**图2.2 软件层次方框图**

### 数据流图

顶层数据流图：



**图2.3 顶层数据流图**

### 数据字典

1.数据流条项目：

(1)测试者信息 = 测试用户姓名 + 添加时间 + 登录IP + 用户位置信息 + 年龄 +邮箱

(2)个人档案 = 用户姓名 + BPM + IPI + 脉搏数据 + 监测时间

(3)健康报告 = 用户姓名 + BPM + IPI + 脉搏数据 + 监测时间 + 是否异常 + AI建议

(4)数据库记录 = 用户姓名 + 监测时间 + 监测记录

(5)咨询记录 = 用户姓名 + 咨询时间 + AI建议

(6)技术人员 = 工号 + 技术人员姓名 + 年龄 + 所在部门 + 解决问题

(7)测量数据 = BPM +IBI + 脉搏数据

(8)传感器信息 = 传感器编号 + 传感器名称 + 传感器状态

(9)预警记录 = 记录编号 + 预警时间 + 预警信息

## **其他需求**

### 数据库需求

能够存储海量用户数据，数据库初始设计至少支持 10万以上的心率数据及相关信息存储，且具备良好的扩展性，随着用户数量增长能够方便地扩容。采用分布式数据库架构，提高数据存储与处理能力，应对大数据量的挑战。

设计合理的数据表结构，将心率数据、用户信息、健康报告等分类存储，建立必要的索引，提高数据查询与检索效率。例如，以用户 ID 为索引关联心率数据表、用户信息表等，确保数据的完整性与一致性。同时，为了支持 AI大语言大模型的数据分析与问答功能，构建专门的数据关联模型，方便快速提取相关数据。

### 安全需求

### 性能需求

-----------------------其余需求分析内容详见组报告4-14页---------------------------

# **基于物联网的智能心率健康检测系统总体设计**

## **总体设计的目标和任务**

（1）目标：构建一个高效、精准、可靠且用户友好的基于物联网的智能心率健康检测系统，实现对用户心率的实时、连续监测，通过数据分析、可视化以及AI大模型为用户提供个性化的健康报告与建议，助力个人健康管理。

（2）任务：整合心率监测硬件设备（心率传感器）、数据可视化以及网页应用，打造一个完整的系统生态。确保系统满足不同用户群体（普通健康人群、慢性病患者、运动爱好者等）在各种场景下（日常活动、运动锻炼、睡眠等）的使用需求。优化系统性能，保障数据传输的及时性、准确性，以及数据分析的高效性与可靠性。设计简洁直观的用户界面与便捷的操作流程，降低用户使用门槛，提升用户体验。

## **需求规定**

### 功能

心率采集功能：依托高精度心率传感器，适配多种佩戴方式（如指尖式、耳夹式等），能够在不同环境与人体状态下稳定、精准地采集心率数据，测量范围至少涵盖 50 - 200 次 / 分钟，测量误差控制在 ±3 次 / 分钟以内，支持实时连续监测，每 5 秒更新一次数据，确保心率数据实时、稳定传输至云端服务器或网页端应用。

数据可视化功能：为用户的每次测量的结果进行可视化展示，动态绘图以及数据存入数据库后进行的全局静态绘图，让用户直观了解测量数据，并在历史检测记录相关模块展示数据库的相应数据。

交互功能：移动端应用具备简洁美观的界面设计，以直观图表（折线图）展示实时心率、历史心率曲线等信息，方便用户快速了解健康状况。支持用户与系统的交互操作，如设置监测参数、反馈使用体验，接入AI大语言模型提升系统的交互性与功能丰富性。

### 性能

响应时间：从心率传感器采集数据到本地服务器接收并存储数据的全过程响应时间不超过 2 秒，确保数据的及时性；用户在移动端应用进行操作（如点击查看心率数据、切换图表、设置参数等）的响应时间不超过 1 秒，提供流畅的操作体验。

吞吐量：能够同时处理至少 1000 个心率监测终端的数据上传请求，且保证数据不丢失、不混淆，满足大规模用户使用场景；在高峰时段（如运动健身高峰期、健康监测集中时段），系统整体吞吐量满足需求，确保各个组件稳定运行，数据传输顺畅。

可用性：系统整体可用性不低于 99.9%，确保用户在任何时间都能可靠地使用该系统进行心率监测与健康管理；各组件具备冗余设计与故障自动切换机制，心率监测终端配备备用电源，当主设备出现故障时，能够迅速切换至备用设备，保障系统持续运行。

准确性：心率测量精度达到医疗级标准，数据分析对常见心率异常的识别准确率不低于 90%，确保系统输出的健康建议与预警信息具有可靠性。

### 输入数据

心率数据：由心率监测终端实时采集的人体心率数值，附带采集时间戳、测量均值、是否正常等标注，数据精度达到小数点后一位，以时间序列形式持续输入系统。

监测参数设置：用户根据自身需求自定义的心率监测相关参数，如测量数据下界、监测频率（如每 5 秒、10 秒监测一次）、数据传输方式偏好等，输入后系统实时生效并按设定规则运行。

### 输出数据

实时心率显示：在网页端应用上，以醒目的数字形式实时展示当前心率数值，字体清晰、颜色鲜明，方便用户随时查看。

健康报告：用户可结合AI大模型生成的包含心率趋势分析、异常情况总结、个性化健康建议的报告，以图文并茂的形式呈现，可进行查看、下载、分享，为用户提供全面的健康洞察。

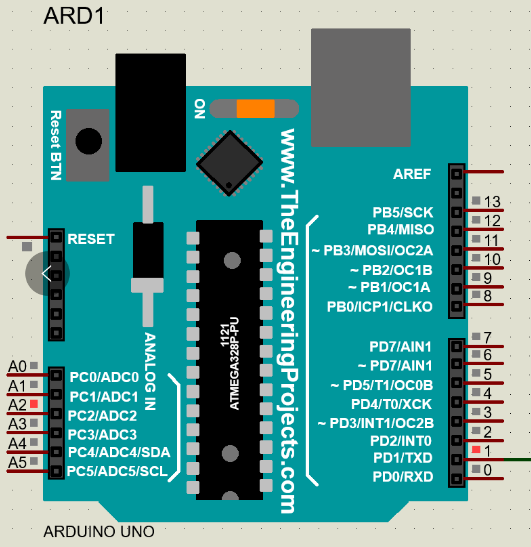
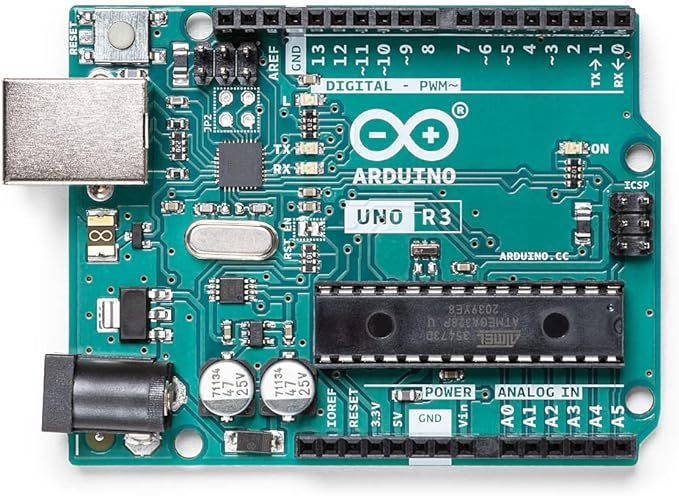
预警信息：当心率异常时，及时向用户推送的包含心率数值、异常类型、发生时间、应对措施建议等内容的通知，通过多种渠道确保用户能及时接收并知晓健康风险。

数据统计图表：根据用户历史心率数据生成的各类统计图表（心率变化曲线），辅助用户直观了解心率变化规律，为自我健康管理提供可视化参考。

## **系统硬件选型及设计**

### 系统硬件介绍

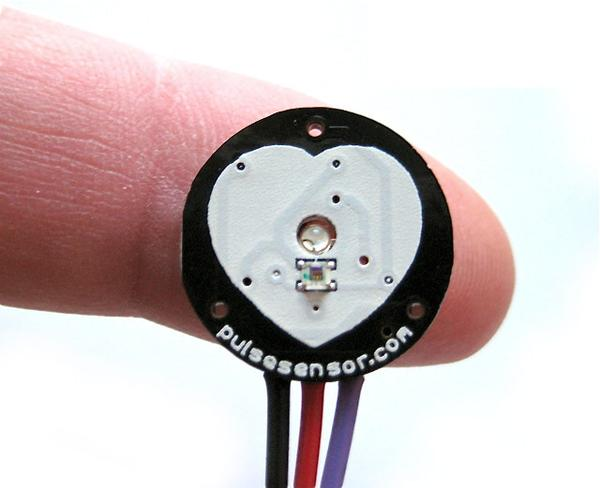
1. Arduino UNO板



**图3.1 Arduino uno Rev3**

Arduino UNO R3是用来做电路连接的Arduino电路板，配套以Arduino IDE软件使用。

1. Pulse sensor心率传感器



**图3.2 Pulse sensor心率传感器**

## **软件结构设计**

### 模块设计说明

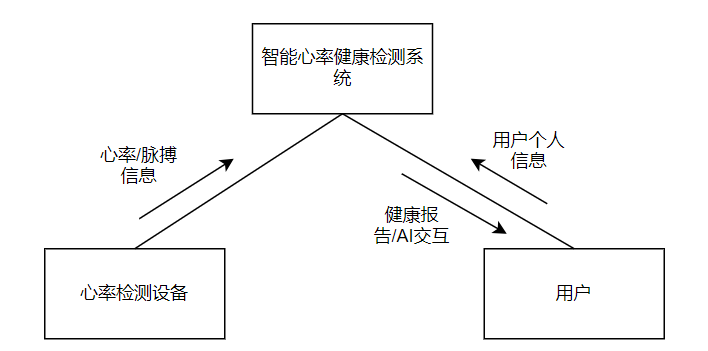
心率采集模块：部署在心率监测终端，负责驱动心率传感器工作，实时采集心率数据，对数据进行初步的滤波、降噪处理，去除因传感器接触不良、人体运动干扰等因素产生的异常数据点，将处理后的数据封装成标准格式，准备传输。

数据分析模块：接收来自各个心率监测终端的数据，运用内置的智能算法对心率数据进行深度分析。结合数据库存储的数据以及AI大模型，构建心率变化、识别异常模式、生成个性化健康建议，将结果存储至结果数据库，为后续的报告生成与预警推送提供数据支持。

预警推送模块：依托云端服务器，实时监控心率数据与预警阈值，一旦发现异常，按照用户设置的通知方式（如短信、推送通知、邮件等，若支持），向用户的智能穿戴设备等发送预警信息，同时记录预警日志，便于后续追溯与分析。

数据展示模块：集成在移动端应用，从云端服务器获取实时心率、健康报告、数据统计图表等信息，以直观、友好的界面设计呈现给用户，接收用户的交互操作指令，如参数设置、图表切换、分享等，并反馈至相应模块处理。

### 模块结构图设计



**图3.11 顶层结构图**

## **程序运行方式**

### 数据库

选用如 MySQL主流关系型数据库管理系统，使用MySQL 8.0版本，数据库可视化工具使用Navicat。

### E-R图（概念结构设计）



**图3.16 E-R图**

### 逻辑结构设计

检测记录表表：包括用户 ID（主键）、传感器类型、存储测量数据的文件名、用户名称、BPM数值、IBI数值、Pluse数值、测量的最低下界值threshold、是否正常以及检测时间字段。

**表 detectionRecord**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 主关键字 | 外部关键字 | 参照表 | 取值说明 |
| ID | int | Yes |  |  | 取值不为空 |
| Sensor | Varchar(255) |  |  |  | 取值不为空 |
| Filename | Varchar(255) |  |  |  | 默认为空 |
| username | Varchar(255) |  | Yes | user(username) |  |
| BPM | int |  |  |  | 默认为空 |
| IBI | int |  |  |  | 默认为空 |
| Pulse | int |  |  |  | 默认为空 |
| Threshold | int |  |  |  | 默认为空 |
| is\_normal | Varchar(20) |  |  |  | 默认为空 |
| time | timestamp |  |  |  | 自动插入当前时间 |

**表 Feedback**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 主关键字 | 外部关键字 | 参照表 | 取值说明 |
| ID | int | Yes |  |  | 取值不为空 |
| uid | int |  | Yes | user(uid) |  |
| email | Varchar(255) |  |  |  | 取值不为空 |
| feedback | Varchar(255) |  |  |  | 默认为空 |

**表 user**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 主关键字 | 外部关键字 | 参照表 | 取值说明 |
| uid | int | YES |  |  | 取值不为空 |
| username | Varchar(255) |  |  |  | 不能为空，取值唯一 |
| time | timestamp |  |  |  | 自动插入当前时间 |

-----------------------其余需求分析内容详见组报告15-26页-------------------------

# **基于物联网的智能心率健康检测系统详细设计**

## **详细设计的目的和任务**

### 详细设计目的

本智能心率健康检测系统的详细设计旨在打造一个集数据采集、传输、处理、分析及可视化于一体的心率健康检测及可视化平台。通过集成先进的PulseSensor传感器技术和强大的AI大语言模型agent，系统能够实时、准确地监测用户的心率数据，提供个性化的健康建议与预警，帮助用户更好地了解自己的健康状况，并采取积极的干预措施。同时，系统还具备数据下载、用户反馈和设备连接检验等功能，以确保用户能够便捷地获取和管理自己的健康数据，与系统实现高效互动。

### 详细设计任务

* 数据采集与传输设计：

选用高性能的PulseSensor传感器，确保心率数据的准确采集。

设计稳定的数据传输机制，实现传感器与上位机之间的实时、无丢失数据传输。

* 数据处理与分析设计：

利用Python编程语言开发数据处理模块，对采集到的心率数据进行清洗、整理和分析。

集成AI大语言模型agent，根据心率数据提供个性化的健康建议或预警信息。

* 数据可视化设计：

在上位机上开发数据可视化界面，实时显示心率数据、心率趋势图等关键信息。

优化界面设计，提高用户交互体验，使用户能够直观地了解自己的健康状况。

* 数据库存储设计：

设计合理的数据库结构，确保心率数据的安全、可靠存储。

提供数据下载功能，使用户能够方便地获取自己的健康数据。

* 用户反馈机制设计：

开发用户反馈界面，允许用户向系统提交反馈信息。

设计后台管理系统，用于接收、处理和回复用户反馈，确保用户与系统之间的有效沟通。

* 设备连接检验设计：

设计设备连接检测功能，实时监测设备的连接状态。

在设备未连接或连接异常时，提供明确的提示或警告信息，确保用户能够及时发现并解决问题。

* 系统优化与测试：

对系统进行全面的优化，提高系统的稳定性、响应速度和用户体验。

制定详细的测试计划，对系统的各项功能进行严格的测试，确保系统的性能和可靠性满足设计要求。

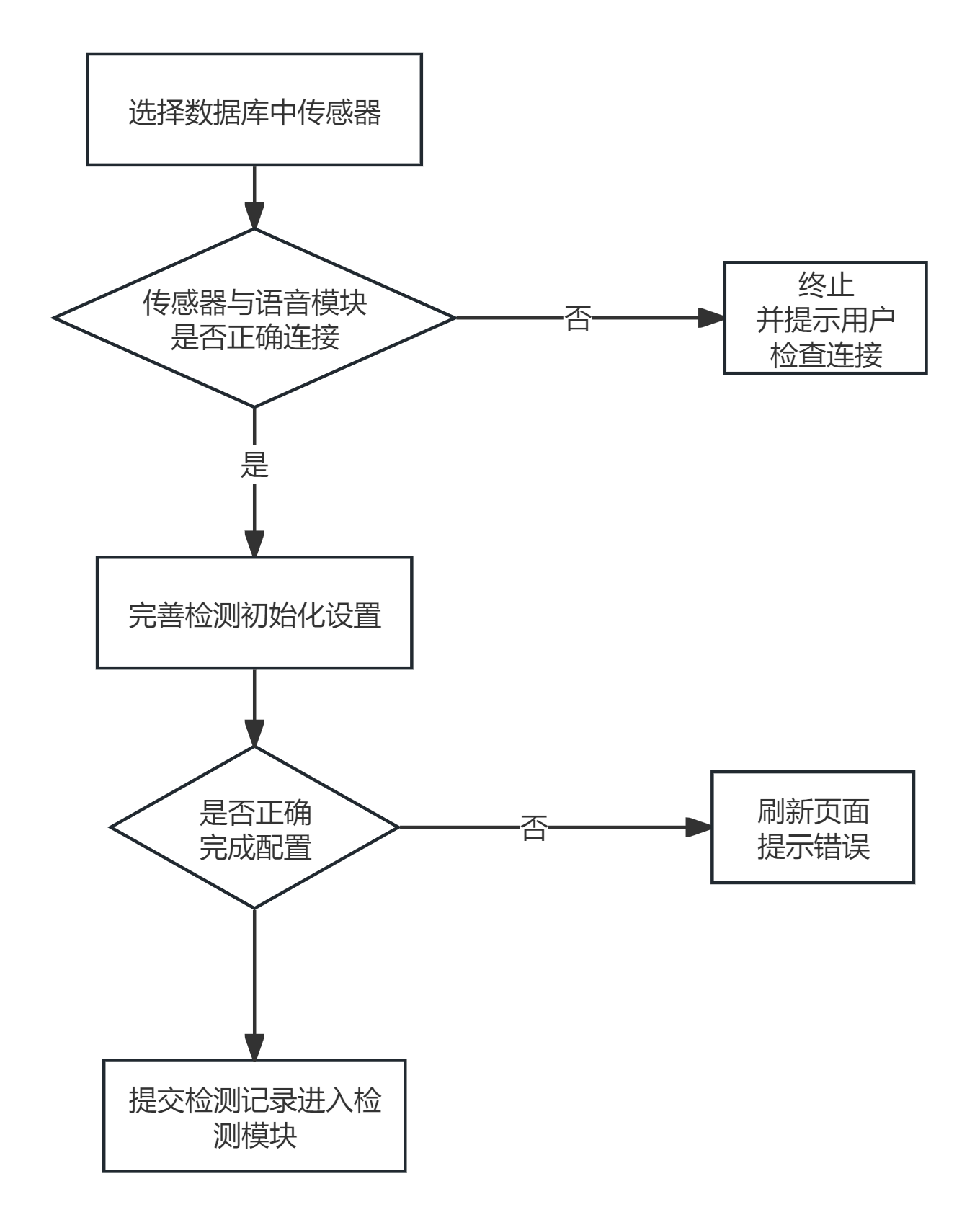
通过完成上述设计任务，我们将能够打造一个功能强大、性能卓越、用户友好的智能心率健康检测系统，为用户提供全面、准确、可靠的健康监测服务。

## **结构化程序设计**

本节将对上位机的主模块程序流程进行详细说明，重点聚焦于为用户提供传感器与上位机交互体验的核心模块。通过绘图的方式，我们将直观展示检测配置初始化、报告可视化、问题反馈以及基于ChatGLM的大语言模型Agent等关键模块。其中，4.2.1节“检测配置初始化”将阐述如何设置和初始化传感器检测参数，为后续操作奠定基础；4.2.2节“报告可视化”则关注于将传感器收集的数据转化为直观易懂的图表和报告，便于用户快速理解分析结果；4.2.3节“问题反馈”模块旨在提供一个用户反馈渠道，以便及时收集和处理用户在使用过程中遇到的问题；而4.2.4节“基于ChatGLM的大语言模型Agent”则引入了先进的AI技术，利用ChatGLM构建的智能代理，不仅能够解答用户关于传感器和上位机使用的疑问，还能提供个性化的操作建议，极大地提升了用户体验和系统交互的智能化水平。

### 检测配置初始化模块

在用户使用交互实时检测平台之前，需要先对检测配置进行初始化，一个时设备的串口连接是否正确的校验，一个是初始化配置信息，如数据点的下界。两项步骤均正确无误完成后即可进去其他模块。本模块的程序流程图如下：



**图4.1 检测配置初始化模块程序流程图**

----------------------其余需求分析内容详见组报告28-32页--------------------------

# 基于物联网的智能心率健康检测系统编码

## **程序设计语言**

### 串口通信

本系统串口通信完全使用Pyserial实现，不涉及其他第三方插件。Pyserial是一个Python的第三方库，用于在串口通信中进行数据的读写操作。它提供了一种简单、方便的方式来与串口设备进行通信，如GPS、条码扫描仪、RFID读卡器等。Pyserial支持多种平台，包括Windows、Linux和macOS，具有跨平台性。本系统利用Pyserial与选型硬件中的传感器进行串口通信，采集数据并存储

### 数据库SQL

SQL（Structured Query Language），即结构化查询语言，是一种专门用来与数据库进行通信的编程语言。本系统中利用PyMySQL结合SQL原生语句编写数据库相关的查询等语句。

### 前端

本系统前端完全使用Python库所支持的Streamlit来编写，无任何第三方插件，Streamlit官网文档地址：https://docs.streamlit.io/。Streamlit是一个开源的Python库，旨在帮助开发者轻松创建和共享美观的自定义Web应用程序。它提供了一个简洁且强大的API，使得即使是非Web开发背景的编程人员也能迅速构建出交互式的Web界面，用于展示数据分析结果、机器学习模型等。

### 硬件

本系统硬件使用为PulseSensor传感器以及JQ8900-16P语音模块。PulseSensor官方网址https://pulsesensor.com/。PulseSensor传感器是一种基于光学原理的心率传感器，可以通过测量心脏跳动时的血液流动情况来检测心率。它由一个LED和一个光敏元件组成，LED发出的光线透过皮肤照射到血液中，光敏元件接收反射回来的光线，然后将光信号转换为电信号，从而测量出心率。JQ8900-16P语音模块用来播报语音。

## **系统开发环境搭建**

本系统软件部分语言全部使用Python语言进行搭建，为了运行本系统需要的Python环境3.12.1及以上。

使用Python的时候，强烈建议创建虚拟环境来搭建开发环境，可以使用source或者conda进行创建虚拟环境，用conda创建虚拟环境是使用conda create env -n sensor创建一个名为sensor的虚拟环境，使用conda activate sensor进行激活，pip install/conda install相关的python库，首次下载是conda会自动安装最新的python版本，可以使用conda list进行查看python版本。创建完虚拟环境以后，需要下载运行项目所需的全部Python库，我已将需要的库打成requirements.txt放在源代码目录下，可以使用pip install -r requirements.txt进行安装。部分库及版本如下：

accessible-pygments==0.0.5

alabaster==0.7.16

altair==5.5.0

annotated-types==0.7.0

anyio==4.7.0

argon2-cffi==23.1.0

argon2-cffi-bindings==21.2.0

arrow==1.3.0

asttokens==3.0.0

attrs==24.3.0

babel==2.16.0

beautifulsoup4==4.12.3

streamlit==1.41.1

zhipuai==2.1.5.20241204

zipp==3.21.0

zundler==0.2.2

1. S:其中streamlit版本必须为1.41.1才支持代码中的部分接口函数，zhipuai建议与我版本一致为2.1.5.2024.1204,否则相关的接口函数参数不一致无法成功运行。

## **源代码目录结构说明**

## **源程序编写**

### 串口通信代码编写

#语音模块的通信代码-->实现实时播报的通信

def sent\_byte(message,port='COM6',bt=115200):

try:

ser = serial.Serial(port, 115200,timeout=1)

while True:

if ser.in\_waiting > 0:

time.sleep(0.1)

ser.write(message.encode())

break

time.sleep(0.1)

while True:

if ser.in\_waiting > 0:

ser.flushOutput()

ser.flushInput()

ser.write(b'E')

ser.close()

break

time.sleep(0.05)

except Exception as e:

print(f'打开串口遇到问题:{e}')

return False

device\_detection(port='COM6')

sent\_byte()

print("检测设备状态:")

device\_detection()

print("开始写数据:")

generate\_csv\_file()

--------------------------------------------------------------------------------------------------

**其余通信编写详见周思语报告**

### 实时绘图代码编写

**代码编写：周思语-详见个人报告**

### 数据库SQL代码编写

数据库建表SQL由我编写：

**Feedback表对应SQL语句：**

CREATE TABLE feedback (

ID INT AUTO\_INCREMENT PRIMARY KEY, -- 主关键字，自增

uid INT NOT NULL, -- 非空字段，外部关键字

email VARCHAR(255) NOT NULL, -- 非空字段

feedback VARCHAR(255) DEFAULT NULL, -- 默认为空

FOREIGN KEY (uid) REFERENCES user(uid) -- 外键约束，参照user表的uid字段

);

**User表对应SQL语句：**

CREATE TABLE user (

uid INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL UNIQUE,

time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**detectionRecord表对应SQL语句**

CREATE TABLE `detectionrecord` (

`id` int NOT NULL AUTO\_INCREMENT,

`sensor` varchar(255) NOT NULL,

`filename` varchar(255) DEFAULT NULL,

`username` varchar(255) DEFAULT NULL,

`BPM` varchar(255) DEFAULT NULL,

`IBI` int DEFAULT NULL,

`Pulse` int DEFAULT NULL,

`threshold` int DEFAULT NULL,

`is\_normal` varchar(20) DEFAULT NULL,

`time` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=50 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

### Streamlit代码编写

由我编写的部分：

#以下是实时检测的部分

import streamlit as st

#from draft\_tool\_code.read\_port import device\_detection

import serial

import sys

import re

import pandas as pd

import matplotlib.pyplot as plt

import plotly.graph\_objects as go

import streamlit as st

import time

import threading

from matplotlib.animation import FuncAnimation

from datetime import datetime

import subprocess

from draft\_tool\_code.read\_port import device\_detection

from plot\_code.plot\_dy import create\_plotly\_plot

import os

import serial

import sys

import random

st.title(':blue[开始体验]🔔')

st.divider()

st.markdown("""

## step3 开始检测

""")

st.info(

"""

请正确佩戴或穿戴设备,穿戴说明如下图所示|请确保您已正确佩戴传感器，否则会造成数据采集误差

"""

)

col1, col2= st.columns(2)

col1.image(r"D:\Users\Administrator\Desktop\sensor\fronted\_page\pic\_static\light1.png",

caption="指示灯闪烁",width=450)

col2.image(r"D:\Users\Administrator\Desktop\sensor\fronted\_page\pic\_static\sensor5.png",caption="指尖式佩戴示例",width=450)

st.subheader("请正确完成下述初始化检测相关配置🏃", divider="blue")

username = st.text\_input(":red[\*]您的姓名:")

st.divider()

low\_Threshold = st.slider("心率时序图下界 (150-200):", 0, 100)

col1, col2, col3 = st.columns(3)

with col2:

submit\_button = st.button("提交&开始检测🏃")

if submit\_button:

ser = serial.Serial('COM6', 115200, timeout=1)

placeholder = st.empty()

placeholder.markdown("正在初始化硬件配置...")

time.sleep(0.5)

placeholder.progress(0, "正在连接JQ8900-16P")

time.sleep(0.5)

t1 = random.randint(40,60)

placeholder.progress(t1, "正在连接Pulse Sensor")

while True:

if ser.in\_waiting > 0:

time.sleep(0.2)

ser.write(b'S') # 将字符串编码为字节并发送

t2= random.randint(61,90)

placeholder.progress(t2, "准备检测")

ser.close()

break

time.sleep(0.1)

params = [

'--username', username,

'--value\_min', str(low\_Threshold)]

subprocess.run(["python", r"D:\Users\Administrator\Desktop\sensor\fronted\_page\plot\_code\plot\_dy.py"]+ params)

#关闭串口避免占用

col1, col2,col3 = st.columns(3)

placeholder.progress(100, "检测完成")

with col2:

st.page\_link("report.py", label="查看检测结果➡️(click me!)")

def generate\_csv\_file(port='COM3',bt=115200,username="Unknown"):

try:

ser = serial.Serial('COM3',115200, timeout=0)

ser.write(b'S')

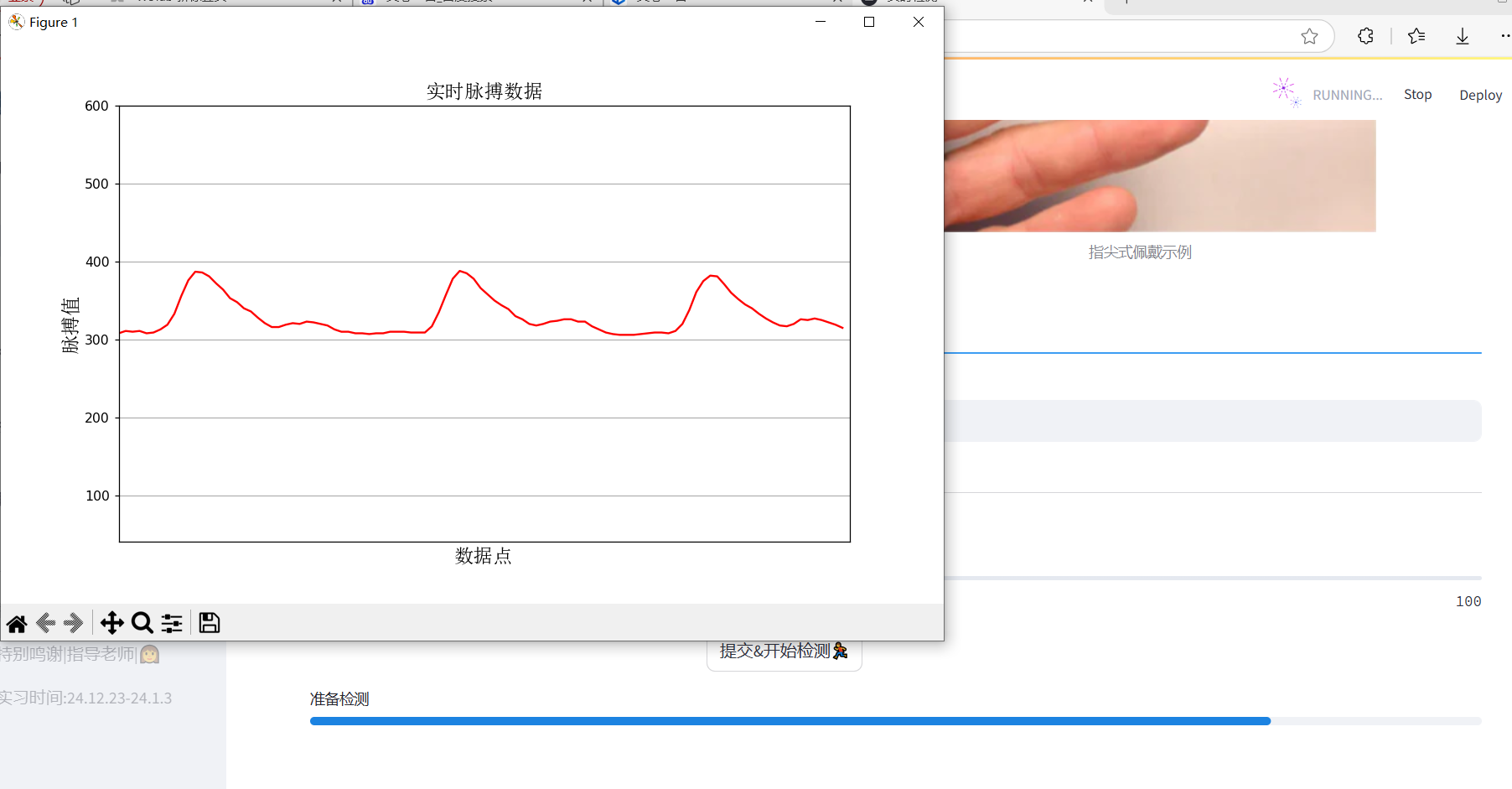
except Exception as e:

print(f'打开串口遇到问题:{e}')

return False



图5.1 Streamlit实时检测页面代码-1



**图5.2** Streamlit实时检测页面代码-2

#反馈模块功能代码：

import streamlit as st

import re

from db\_config.db\_info import insert\_mysql\_feedback

st.set\_page\_config(page\_title='TYUT - 问题反馈', layout='wide', page\_icon=':thought\_balloon:')

st.title("问题反馈")

st.divider()

st.title("技术人员正在马不停蹄地debug:blue[!] :sunglasses:")

#提交邮箱格式校验

email\_regex = re.compile(

r"(^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$)"

)

multi = '''

\* 前端支持：周思语 :ok\_woman: | 韩鑫 :runner: | 夏书淇 :speech\_balloon:

\* 后端支持：夏书淇 :speech\_balloon: | 周思语 :ok\_woman: | 韩鑫 :runner:

\* 传感器技术支持：韩鑫 :runner: | 夏书淇 :speech\_balloon: | 周思语 :ok\_woman:

\* [排名不分先后 :yum:]

'''

st.markdown(multi)

with st.expander("联系我们"):

with st.form(key='contact', clear\_on\_submit=True):

email = st.text\_input('联系邮箱')

problem\_description = st.text\_area("遇到问题", "请将您遇到的问题填写在此处以便我们能及时解决")

submit\_button = st.form\_submit\_button(label='发送信息')

if submit\_button:

# 校验邮箱地址格式

if not email\_regex.match(email):

st.error("请输入有效的邮箱地址。")

# 校验问题描述是否为空

elif not problem\_description.strip():

st.error("问题描述不能为空。")

else:

insert\_mysql\_feedback(email,problem\_description)

st.success('您的信息已成功提交！')

print(email)

print(problem\_description)



**图5.3** Streamlit反馈页面代码-1

#历史检测记录表格代码

import pymysql

from pymysql import cursors

import streamlit as st

import random

import pandas as pd

import streamlit as st

from db\_config.db\_info import fetch\_mysql

import numpy as np

from db\_config.db\_info import get\_record\_count,calculate\_column\_IBI\_mean,calculate\_column\_Pulse\_mean

import os

st.title("历史检测记录")

st.divider()

main\_path= r'D:\Users\Administrator\Desktop\sensor\data\_saved'

result = fetch\_mysql()

history\_df = pd.DataFrame(result)

pulse\_values = []

for index, filename in enumerate(history\_df['filename']):

file\_path = os.path.join(main\_path, filename)

data\_pulse = pd.read\_csv(file\_path, encoding='gbk')

pulse\_value = data\_pulse["Pulse/mS"].tolist()

print(type(pulse\_value))

pulse\_values.append(pulse\_value)

pulse\_series = pd.Series(pulse\_values, index=history\_df.index)

history\_df['plot'] = pulse\_series

st.dataframe(

history\_df,

column\_config={

"id": "检测序号",

"time": "检测日期",

"sensor": "传感器型号",

"filname": "检测数据文件名",

"plot":st.column\_config.LineChartColumn("Pulse mS", y\_min=200, y\_max=700)

},

hide\_index=True,

)



**图5.4** Streamlit历史检测页面代码-1

--------------------------------------------------------------------------------------------------

**其余Streamlit前端页面代码编写：周思语-详见个人报告 韩鑫-详见个人报告**

### 接口函数编写

#生成并存储CSV报告

#这部分代码合并在周思语的绘图代码中

def save\_csvfile(self):

csv\_filename = file\_time\_rename()

csv\_filepath = os.path.join(self.main\_path,csv\_filename)

with open(csv\_filepath, mode='w', newline='') as csvfile:

csvwriter = csv.writer(csvfile)

csvwriter.writerow(["Username","Index","BPM/mS(Threshold:200)", "IBI/mS", "Pulse/mS","RecordTime","Sensor"])

for item in self.data\_list:

csvwriter.writerow(item)

avg\_BPM,avg\_IBI,avg\_Pulse = self.cal\_avg()

csvwriter.writerow(["--","--",avg\_BPM, avg\_IBI, avg\_Pulse,"--","PulseSensor"])

print("写入完成！")

#插入记录到数据库

if avg\_BPM=='未检测到数据':

avg\_BPM = int(60000/avg\_IBI)

if (avg\_BPM > 50 and avg\_BPM<150):

is\_normal = "正常"

elif avg\_BPM>150:

is\_normal = "测量异常，请重新测量"

insert\_plot\_db(self.sensor,csv\_filename,self.username,avg\_BPM,avg\_IBI, avg\_Pulse,self.value\_min,is\_normal)

print("插入完成！")

--------------------------------------------------------------------------------------------------

**其余接口函数代码编写：周思语-详见个人报告 韩鑫-详见个人报告**

### 数据库功能代码编写

由我编写的部分：

import pymysql

from pymysql import cursors

import pandas as pd

#本地数据库连接信息

config = {

'host': 'localhost',

'user': 'root',

'port':3306,

'password': 'Xsq031124',

'db': 'sensor',

'charset': 'utf8mb4',

'cursorclass': pymysql.cursors.DictCursor

}

#数据库用的是本地的 db\_name:sensor| 反馈表:infoBack 检测记录表: detectionRecord

#数据库连接函数

def local\_b\_connect():

connection = pymysql.connect(\*\*config)

#按用户名检索函数

def fetch\_by\_username(username, table="detectionRecord"):

connection = pymysql.connect(\*\*config)

try:

with connection.cursor() as cursor:

sql = f"SELECT \* FROM {table} WHERE username = %s ORDER BY time DESC"

cursor.execute(sql, (username,))

result = cursor.fetchall()

return result

except Exception as e:

print(f"遇到错误: {e}")

return False

#按传感器型号检索函数

def fetch\_by\_sensor(sensor, table="detectionRecord"):

connection = pymysql.connect(\*\*config)

try:

with connection.cursor() as cursor:

sql = f"SELECT \* FROM {table} WHERE sensor = %s ORDER BY time DESC"

cursor.execute(sql, (sensor,))

result = cursor.fetchall()

return result

except Exception as e:

print(f"遇到错误: {e}")

return False

#按是否正常检索函数

def fetch\_by\_is\_normal(is\_normal, table="detectionRecord"):

connection = pymysql.connect(\*\*config)

try:

with connection.cursor() as cursor:

db\_is\_normal = "正常" if is\_normal else "疑似异常"

sql = f"SELECT \* FROM {table} WHERE is\_normal = %s ORDER BY time DESC"

cursor.execute(sql, (db\_is\_normal,))

result = cursor.fetchall()

return result

except Exception as e:

print(f"遇到错误: {e}")

return False

其余数据库连接代码编写：周思语-详见个人报告xx页 韩鑫-详见个人报告xx页

### 大模型Agent代码编写

全部由我编写：

import streamlit as st

from zhipuai import ZhipuAI

import pymysql

import json

db\_config = {

'host': '127.0.0.1',

'user': 'root',

'password': 'Xsq031124',

'database': 'sensor',

'charset': 'utf8mb4',

'cursorclass': pymysql.cursors.DictCursor,

}

connection = pymysql.connect(\*\*db\_config)

def pulse\_detect():

"""

检索数据库存储的最近一次的心率检测记录数据,给出详细的检测记录,包括测量人姓名,BPM(每分钟心脏跳动次数),Pulse(每毫秒),检测时间

"""

try:

#连接数据库

connection = pymysql.connect(\*\*db\_config)

with connection.cursor() as cursor:

sql = "SELECT username,filename,BPM,IBI,Pulse,is\_normal,time FROM detectionRecord ORDER BY time DESC LIMIT 1"

cursor.execute(sql)

result = cursor.fetchone()

if result:

pulse\_info = result

pulse\_info = f"测量者{result['username']}的心率测量记录如下: BPM(每分钟心脏跳动次数):{result['BPM']},Pulse(每毫秒):{result['Pulse']},检测时间:{result['time']},测量数据在理论范围是否正常:{result['is\_normal']}"

print(pulse\_info)

return pulse\_info

except pymysql.MySQLError as e:

print(f"Error: {e}")

return pulse\_info

tools = [

{

"type": "function",

"function": {

"name": "pulse\_detect",

"description": "检索数据库存储的最近一次的心率检测记录数据,给出详细的检测记录，包括测量人姓名，BPM(每分钟心脏跳动次数)，Pulse(每毫秒)，检测时间:{result['time']}",

"parameters": {

"type": "object",

"properties": {},

"required": []

}

}

}

]

#工具函数传参

function\_list = [pulse\_detect]

available\_functions = {func.\_\_name\_\_:func for func in function\_list}

def sql\_Agent\_pulse(prompt,available\_functions):

prompt = prompt

message = [{"role":"user","content":prompt}]

client = ZhipuAI(api\_key=api\_key)

response = client.chat.completions.create(

model="glm-4-plus",

messages= message,

tools =tools,

tool\_choice = "auto"

)

try:

function\_to\_call = available\_functions[response.choices[0].message.tool\_calls[0].function.name]

function\_args = json.loads(response.choices[0].message.tool\_calls[0].function.arguments)

fucntion\_response = function\_to\_call(\*\*function\_args)

message.append({

"role":"tool",

"name":response.choices[0].message.tool\_calls[0].function.name,

"content":"最近一次检测记录的信息："+str(fucntion\_response)

})

return message

except Exception as e:

print(f"{e}")

return False

api\_key = ""

st.set\_page\_config(page\_title='TYUT - ChatGLM智能建议', layout='wide', page\_icon=':robot:')

st.title("💬 智能建议专家")

st.divider()

st.caption("🚀 一个基于ChatGLM4的智能心率健康建议专家")

if "messages" not in st.session\_state:

st.session\_state["messages"] = [{"role": "assistant", "content": "你可以向我询问有关健康方面的知识"}]

for msg in st.session\_state.messages:

st.chat\_message(msg["role"]).write(msg["content"])

def stream\_out(messages):

client = ZhipuAI(api\_key=api\_key)

response = client.chat.completions.create(

model="glm-4-plus",

messages = messages,

stream=True,)

for chunk in response:

bot\_response = chunk.choices[0].delta.content

yield str(bot\_response)

#原始回答

def raw\_stream\_out(prompt):

client = ZhipuAI(api\_key=api\_key)

response = client.chat.completions.create(

model="glm-4-plus",

messages=[

{"role": "user", "content": prompt},

],

stream=True,

)

for chunk in response:

bot\_response = chunk.choices[0].delta.content

yield str(bot\_response)

if prompt := st.chat\_input(f"Q:分析我最近一次的心率检并给出建议"):

st.session\_state.messages.append({"role": "user", "content": prompt})

st.chat\_message("user").write(prompt)

with st.chat\_message("assistant"):

with st.spinner("Thinking..."):

message = sql\_Agent\_pulse(prompt,available\_functions) #append调用了我的agent的message

print(message)

if not message:

print(prompt)

response = raw\_stream\_out(prompt)

else:

response = stream\_out(message)

full\_response = st.write\_stream(response)

message = {"role": "assistant", "content": full\_response}

st.session\_state.messages.append(message) #添加历史会话信息



**图5.4** Streamlit历史检测页面代码-1

### 硬件代码编写

**代码编写：韩鑫-详见个人报告**

# **基于物联网的智能心率健康检测系统测试**

## **软件测试目的**

智能心率健康检测系统的测试目的是确保系统能够正常地运行，并满足用户需求；及时发现需求与实现之间的差距，通过反馈意见改进系统；验证系统的可靠性和稳定性，发现其中的漏洞检验系统的性能，看系统在主要的功能模块下是否能够正常运行。

## **软件测试方法**

## **单元测试**

## **集成测试**

## **测试结论**

经过一系列详尽的测试，我们针对智能心率健康检测系统的主要功能及新增特性进行了全面评估。测试范围涵盖了数据采集、数据传输、数据可视化、AI大语言模型agent、数据库存储、下载功能、反馈功能以及设备可使用性等多个方面。

在数据采集方面，PulseSensor传感器展现出了卓越的性能，能够稳定、准确地采集心率数据，实时性和准确性均达到了预期标准。数据传输测试也表明，从传感器到上位机的数据传输过程稳定可靠，未出现数据丢失或延迟现象。

数据可视化功能在上位机上得到了很好的实现，用户能够直观地看到实时心率数据以及心率趋势图，这对于用户了解自身健康状况具有重要意义。同时，AI大语言模型agent也展现出了强大的分析能力，能够根据心率数据提供合理、准确的健康建议或预警。

在数据库存储方面，系统能够正确地将心率数据存储到数据库中，且数据完整、无丢失。此外，我们还特别对下载功能进行了测试，结果显示，用户能够轻松地将测量数据下载到本地，且文件内容与数据库中存储的数据完全一致。

用户反馈的信息能够迅速、准确地发送到后台，并被技术人员及时查看和处理。这为用户与系统之间的有效沟通提供了有力保障。

最后，在设备可使用性测试方面，我们验证了设备连接本地串口的稳定性和可靠性。系统能够实时检测设备的连接状态，并在设备未连接或连接异常时给出相应的提示或警告信息。

综上所述，智能心率健康检测系统在各个方面均表现出了卓越的性能和稳定性。我们坚信，该系统将为用户提供一个全面、准确、可靠的健康监测解决方案。

------------------------其余软件测试内容详见组报告38-54页------------------------

# 基于物联网的智能心率健康检测系统总结

## 个人总结

本次实习由我和组员一起选题完成，从选题设计到方案设计、可行性研究、需求分析、总体设计、详细设计、编码、测试等，每个部分我与组员共同参与完成，我们团队的氛围很好，虽然大家都遇到了不会的东西，但是我们都齐心协力一起完成。尽管，中间还有考试的任务安排，组员也没有抛下实习的任务，大家精益求精，不断优化每个部分，韩鑫遇到硬件问题一直尝试，虽然最后没能解决模块组合的问题，但是尽力优化与调试了硬件。周思语负责绘图部分，无论是PyQt5的实现或者是嵌入页面的难题，尽管最后我们发现streamlit没办法实现嵌入的功能，遇到了很多挫折，耽误了两天时间没什么进度，但我认为错误也是工作的一部分。调试代码发现无法实现，也是工作的一部分，虽然这两个模块一直没进度，但是我们也一直在优化。这一周的时间非常感谢组员的支持。

我个人在这次实习中，协调组员的每个模块，将个人的课外知识与课内知识结合完成了这次实践。由于时间的紧迫，一些附加功能还没能实现，没法展示给老师，算是一个遗憾，但是这个过程中和组员一起从0到1的过程，每天实验进步一点进度的过程体验非常好，技术和读文档、撰写规范都有提升。希望后续自己还有时间去优化一些功能模块，这次没实现的功能尽量补足。同时我也在实习过程中发现了自己硬件知识方面的短板，我将在下一个实习中逐步学习STM32的内容后，完善这次实习中的硬件部分。

## 改进的地方

1. 硬件选型部分，我们组的步骤有错。硬件的步骤应该是，先设计电路原理图，在protesus中进行仿真后，确认每个硬件的模块功能无误以后再去订购硬件。我们组先购买了硬件，回来以后发现语音控制模块和心率检测模块没办法在一块Arduino uno R3板中完成，这是这次的一个错误的地方，也是没能实现语音控制的遗憾，这个模块最后只能作为语音播报使用。
2. 自己对硬件的知识有所欠缺，这次只参与硬件的选型和串口通信设计，其他部分交给了韩鑫完成，这是我在硬件工作部分的一个欠缺，下一个实习改进。

## 致谢

感谢这次实习以及这个学期老师教授的知识。无论是传感器部分的知识或者是物联网软件工程的知识，都在这次实习中得到了很好的应用，传感器部分的知识帮助我们完成了硬件和上位机的通信部分、物联网软件工程部分的知识指导我们的系统搭建全流程、给我们撰写方案提供了指导。全面详尽的需求分析使得我们的开发工程并没有出现需要返工的情况。

感谢我的组员这一周在编码、资料搜索等方面的全部支持，正是我们每个人的参与使得这次系统开发非常流畅的进行，完成了我们基于物联网的智能心率健康检测平台的选题。