嵌入式系统技术实习报告

二〇二一年十二月二十四日

目 录

第一部分：智能病房系统设计.....1
1.1 设计概述.....1
1.1.1 整体设计.....1
1.1.2 硬件设计.....2
1.1.3 软件设计.....3
1.2 嵌入式系统设计.....6
1.2.1 多线程设计.....6
1.2.2 医疗设备监控.....6
1.2.3 数据管理与使用.....7
1.2.4 TCP 通信.....8
1.3 单片机功能板设计.....9
1.3.1 智能电器.....9
1.3.2 紧急情况报警.....9
1.4 主站设计.....10
1.4.1 数据集中管理.....10
1.4.2 紧急情况集中处理.....11
1.5 实习心得.....12
1.5.1 遇到问题及解决.....12
1.5.2 收获感想.....16
第二部分：Linux 系统体验实习.....17
2.1 Linux 操作系统.....17
2.2 Linux 操作系统移植.....18
2.3 驱动开发.....19
2.4 基于 Linux 系统应用开发与裸机应用开发.....19
2.5 实习心得.....20
2.5.1 遇到问题及解决.....20
2.5.2 收获感想.....21

第一部分：智能病房系统设计

1.1 设计概述

1.1.1 整体设计

本系统旨在设计一个智能病房系统，用于实现医院病房乃至整个医院医疗系统的智能化，提升病房的安全性和管理的高效性。整体系统构思的拓扑图如图 1.1 所示。

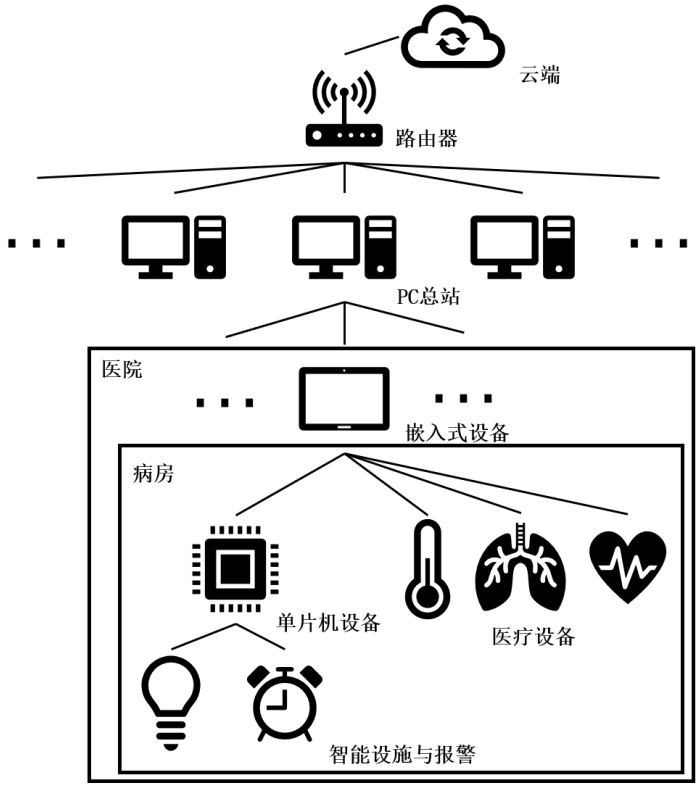

图 1.1 智能病房系统拓扑结构图

系统的主体部分由病房内部的嵌入式系统构成，嵌入式系统通过串口与单片机设备相连，用于实现类似智能家居的控制功能例如紧急报警，开关门/灯的一系列功能。同时嵌入式系统也与病房内的医疗设备通过串口进行通讯，实时获取当前病人的生理状况。每个病房的嵌入式设备通过自组局域网的形式处于同一网络，可以通过 TCP 协议向主机 PC 上传病人信息，同时若病房出现紧急状况在 PC 端也可以进行查看。PC 主机还可以通过路由器接入互联网使用云端存储和云端管理的相应功能。由于实习的时间有限，最终只针对病房嵌入式部分以及 PC 总站进行了相应开发。

1.1.2 硬件设计

系统的硬件设计主要包括四个部分，包括嵌入式开发所使用的树莓派，用于功能拓展的 STM32 单片机，同时还有用于模拟医疗设备和总站的 PC 机等。

1.1.2.1 嵌入式设备

本次设计使用树莓派搭载 Ubuntu 操作系统用于嵌入式开发，可以搭配触摸屏进行使用。树莓派 4B 采用 BCM2711 四核 CPU，实物图如图 1.2 所示，采用 Cortex-A72 处理器内核。Cortex-A72 最早发布于 2015 年年初，也是基于 ARMv8-A 架构，采用台积电 16nm FinFET 制造工艺，Cortex-A72 可在芯片上单独实现性能。在相同的移动设备电池寿命限制下，Cortex-A72 能相较基于 Cortex-A15 的设备提供 3.5 倍的性能表现，相比于 Cortex-A57 也有约 1.8 倍的性能提升，展现出了优异的整体功耗效率。

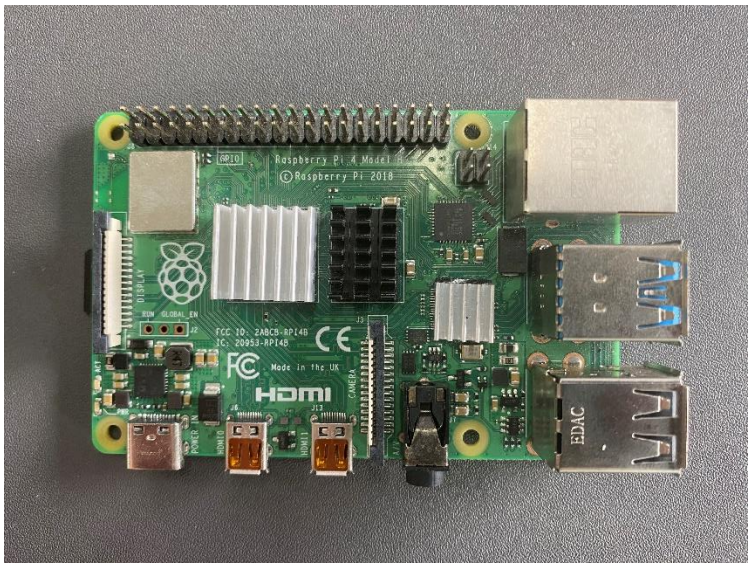

图 1.2 树莓派 4B 实物图

1.1.2.2 单片机设备

单片机采用的是自主制作的基于 STM32F103C8T6 的功能板，其实物图如图 1.3 所示。单片机可以与树莓派通过串口连接进行通讯，采用舵机模拟开门关门，使用 LED 灯模拟开关灯以及蜂鸣器用于报警。在整体系统中单片机作为嵌入式系统的额外拓展模块，提供功能化的使用接口，便于移植以及同时接入多个单片机设备。

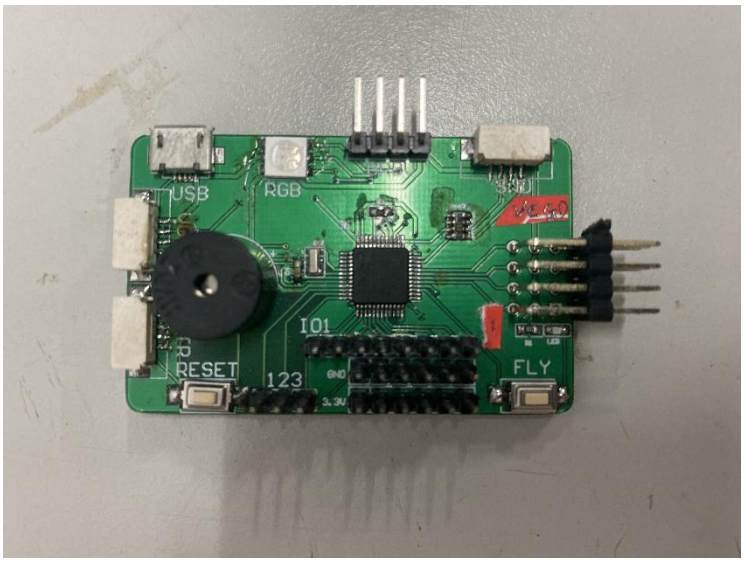

图 1.3 STM32F103C8T6 功能板实物图

1.1.2.3 医疗设备与 PC 主站设备

由于没有真实的医疗设备供使用，我们采用笔记本模拟的方式进行，笔记本通过 TCP 协议向树莓派随机放松数据信息用于模拟实际病房过程中的医疗设备信息。

本次实习采用笔记本电脑作为主站设备，通过 TCP 与每个病房中的嵌入式设备进行通信，从而实现整个医院病房信息记录以及查询等功能。

1.1.3 软件设计

系统软件设计主要包括四大部分，如图 1.5 所示。第一部分是底层数据的管理，即图中橙色部分；第二部分为功能板的控制功能，即图中绿色部分；第三部分为系统交互界面设计模块，即图中紫色部分；最后一部分是通信部分，为图中黄色部分。

1.1.3.1 数据管理模块

考虑到实际医院医护信息多，由护士直接进行登记容易造成信息的丢失以及错误，采用数据库的技术可以很好解决这一问题，有利于数据的保存和统一管理。本次实习采用的数据库管理系统基于 SQLite 进行搭建，SQLite 是一款轻型的数据库，它的设计目标是嵌入式的，而且已经在很多的嵌入式产品中使用到了它，它占用资源非常的低，在嵌入式设备中可能只需要几百 K 的内存就可以使用，

因此选其作为嵌入式开发的数据库。

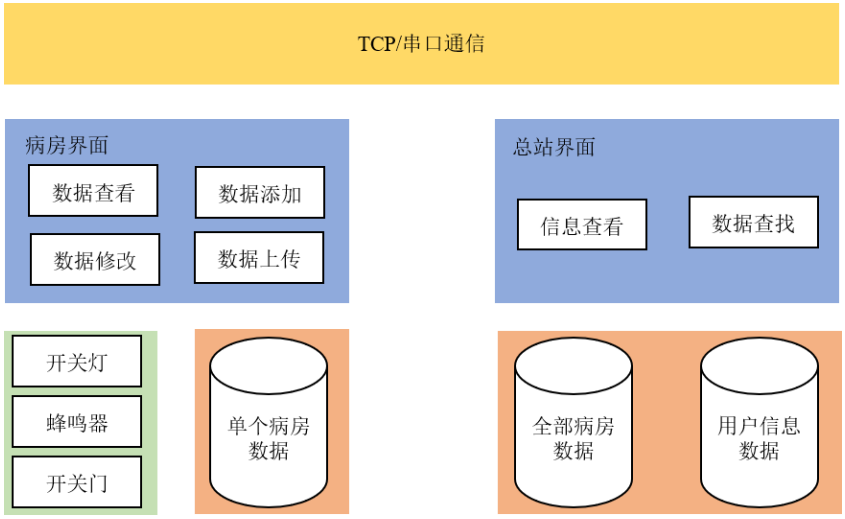

图 1.5 系统软件设计框图

1.1.3.2 控制模块

采用模块化的设计可以增加设计的可移植性以及有利于二次开发，本系统将病房内的部分控制功能转移至专门的功能拓展板，针对功能板的开发采用相关的单片机开发技术。

STM32CubeMX 是一个全面的软件平台，包括了 ST 产品的每个系列。平台包括了 STM32CubeMX 硬件抽象层和一套的中间件组件，模块化的开发方式可以很大程度减少开发时间，针对本次使用的单片机系统，使用 STM32CubeMX 配置相关内容如图 1.6 所示。

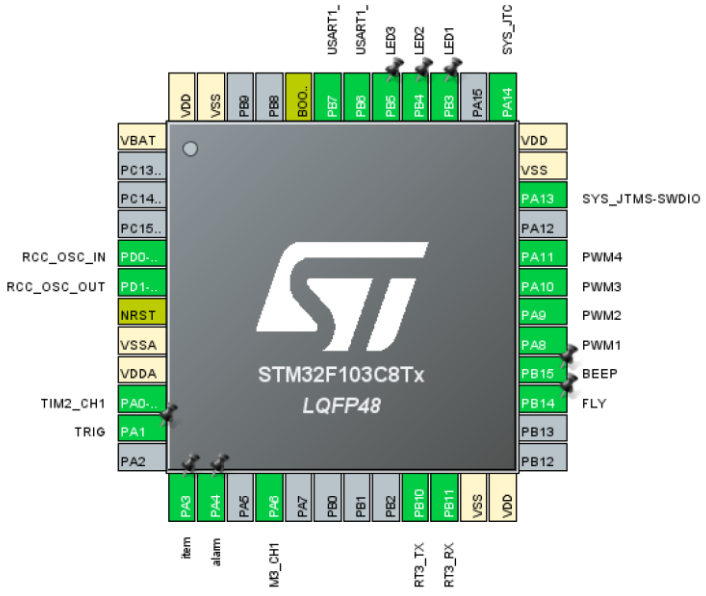

图 1.6 STM32CubeMX 引脚配置

1.1.3.3 交互界面模块

交互界面的设计有利于在系统封装完成后对系统采用上层的方式进行使用，同时可以在交互界面进行一些基础设置的修改还有功能的实现。考虑到本次使用的硬件设施为树莓派，其自带的 Ubuntu 系统支持 Python 开发，本次实习的人机交互功能由 Tkinter 进行实现。

Tkinter 模块(Tk 接口)是 Python 的标准 Tk GUI 工具包的接口，TCL/Tk 的 GUI 工具组。Tk 和 Tkinter 可以在大多数的 Unix 平台下使用，也可以应用在 Windows 和 Mac 系统里。在使用 Tkinter 时，我们不需要关注 TCL/Tk 的实现，而可以将 Tkinter 看成 python 一个独立的扩展。其方便以及轻量的特点作为嵌入式交互界面的开发正合适。

1.1.3.4 通讯模块

在本次实习过程中设计到设备与设备之间的通信，其中功能板与嵌入式采用串口通信，是单片机与嵌入式开发的常规通信方式。医院信息的通信需要保证数据的正确，嵌入式系统与主站之间的通信采用 TCP 协议进行通信，并采用无线的方式，保证信息准确无误的同时也减少了线路的搭建成本。

1.2 嵌入式系统设计

在本节内容中，将对具体的细节设计进行展开，不会对软件和硬件进行明确的区分，同时会给出嵌入式系统设计部分的最终效果图。

1.2.1 多线程设计

针对嵌入式系统的开发，我们在实际使用中发现在数据显示方面如果需要一直刷新医疗设备传回数据，在TCP通信方面容易导致程序的运行顺序出现混乱，因此采用多线程的设计方式，将 TCP 通信与正常的屏幕显示采用多线程运行的方式可以很好解决这一问题。多线程设计代码如下所示：

```
try:
    thread1 = threading.Thread(target=show_client, args=(0.1,))
    thread3 = threading.Thread(target=tcp_recieve, args=(7890,
data_recieve, 0.2,))
    thread1.setDaemon(True)
    thread1.start()
    print('STARTING Thread-1')
    thread3.start()
    print('STARTING Thread-3')
except:
    print('ERROR: 无法启动线程')

thread1.join()
thread3.join()
print("main process end.")
```

1.2.2 医疗设备监控

为了实时监测病人的生命体征情况，系统会不断采集监控病人体征的传感器的数据，显示屏幕会实时更新数据，实时的显示病人的信息，显示界面设计如图1.7 所示。

每一个病房内都会放置一块如图 1.7 所示的显示屏幕，这块屏幕上会显示病房号、病患信息以及病人的各项生命体征数据。同时，屏幕右上方还配有一个上传 按键供医护人员使用，当医护人员查房时，若想要进行病人信息的快捷上传，不需要再使用 纸质统计表进行数据登记，只需在病房内点击此按键便可以完成每次的数据登记。

图 1.7 医疗设备情况显示图

1.2.3 数据管理与使用

在医疗设备情况显示界面右上角点击上传界面即可以进入登记记录界面。考虑到如今医院内部护士查房均采用笔记方式，容易导致数据收集不易且出错率较高，采用无纸化的记录方式一定程度上可以提高效率以及后期管理。界面设计如图 1.8 所示。其中可以在界面右侧对病人状况进行记录，点击上传按钮即可以将当前的所有数据上传至总站。

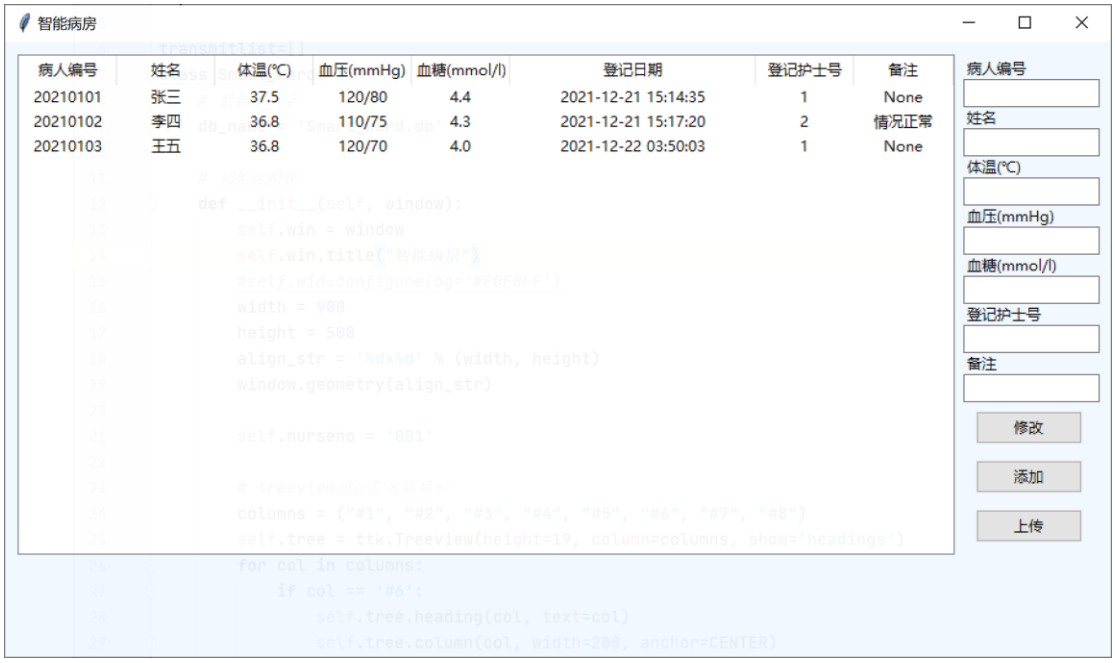

图 1.8 护士登记与上传界面

1.2.4 TCP 通信

考虑到平时的病房各种设备走线繁杂琐碎，其中用于数据传输的线路很大程度上会影响医用器械的使用安全，故我们对各个部分之间设计了基于 TCP 连接的通信方式，无论是传感器与病房内的显示屏幕，还是屏幕与总站之间，均为无线连接。又考虑到医院内的数据均属于个人隐私，故我们并没有将数据传输接入互联网，而是使用自组局域网的方式来构建数据传输的基础。因此所有的数据传输过程都在医院的掌控内，大大见笑了病患数据被窃取或者意外流失的可能性，提高了系统安全性与可靠性。TCP 收发的关键代码如下所示：

```
import socket
# 数据接收
tcp_server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
tcp_server_socket.bind(("", port))
tcp_server_socket.listen(128)
new_client_socket, client_addr = tcp_server_socket.accept()

# 数据发送
tcp_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_ip = ip
server_port = port
server_addr = (server_ip, server_port)
tcp_socket.connect(server_addr)
while True:
    # send_data = input("请输入要发送的数据:")
    tcp_socket.send(data.encode("utf-8"))

    time.sleep(delay)
```

1.3 单片机功能板设计

1.3.1 智能电器

为方便病人在病房内的活动，在设计时使用 LED 模拟走道灯光，舵机模拟房门，使用红外传感器检测物体，设计目的是当红外传感器检测到物体时 LED 灯点亮，以及舵机转动用于开门的效果。实物效果如图 1.9 所示，粉灯点亮，舵机转动。

图 1.9 智能家居实际效果图

1.3.2 紧急情况报警

考虑到医院病房的安全性，使用红外传感器模拟烟雾报警，当传感器检测到烟雾时，需要有蜂鸣器报警以及红灯点亮，并能够通过串口将信息传输给嵌入式设备由嵌入式设备向主站汇报。实际效果如图 1.10 所示，可以看到在检测到紧急情况后功能板能够提供红灯常亮功能，在实际中还有蜂鸣器报警。

图 1.10 报警实际效果图

1.4 主站设计

1.4.1 数据集中管理

在进入主站显示界面前设计了相应的登录界面用于增强数据的安全性，在登录界面可以进行注册以及登录，登录后若通过验证即可以直接显示主界面。登录界面的设计如图 1.11 所示。

图 1.11 主站登录界面

针对主站的设计主要目的是对各个病房的数据进行收集用于集中管理，同时支持主站的查询功能，界面展示如图 1.12 示。

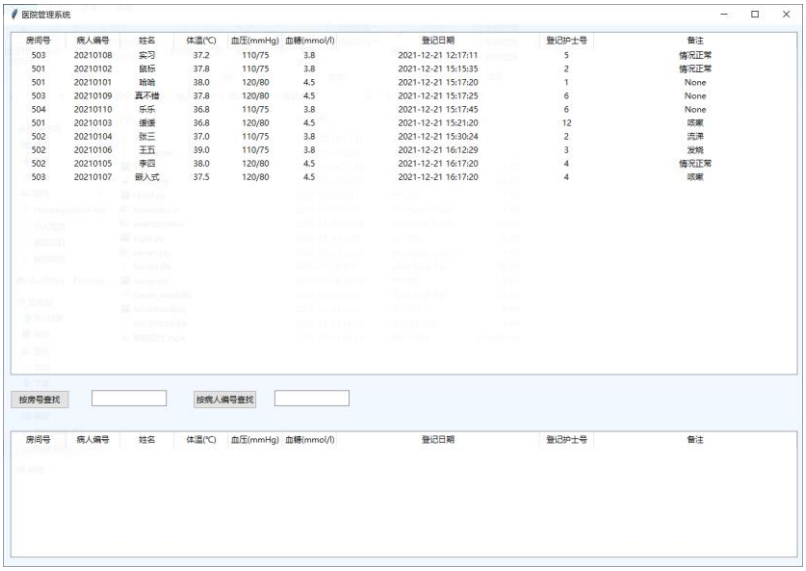

图 1.12 站数据管理界面

界面主要分为两大部分，上半部分用于数据展示，显示了实时上传的病房信

息，即嵌入式设备在各个病房点击上传按钮后上传的结果。界面下半部分主要是信息查询的结果界面，支持房号以及病人编号的查询方式。查询功能的展示如图 1.13 示，查询 502 病房的相关信息可以在下半部分得到展示。

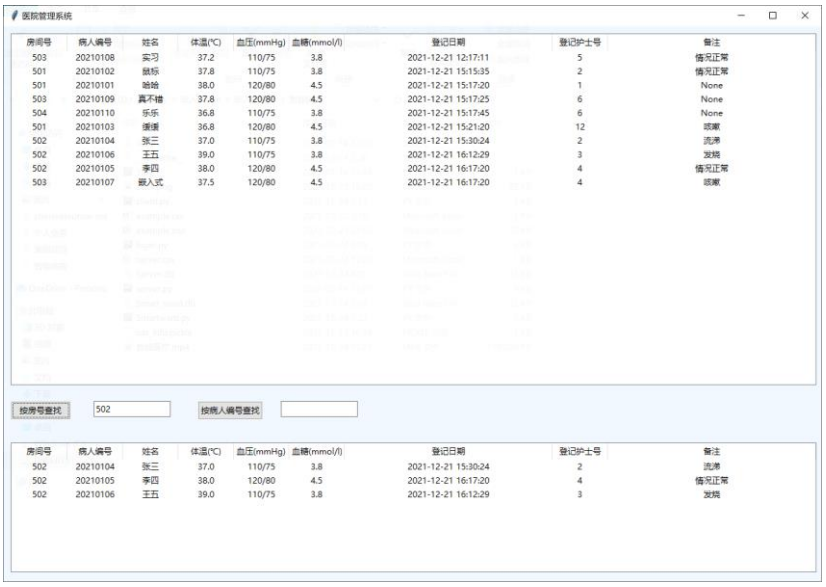

图 1.13 查询功能展示

1.4.2 紧急情况集中处理

由于无法保证实际紧急情况发生时病房内有医护人员在现场，容易出现报警不及时的问题存在，针对这一问题可以采用总站提示的方式便于问题的及时发现。参照在嵌入式系统中的设计，针对紧急情况开辟了单独的线程用于接收紧急情况的 TCP 报文，在情况发生时会在主界面有提示窗口弹出，实际效果如图 1.14。

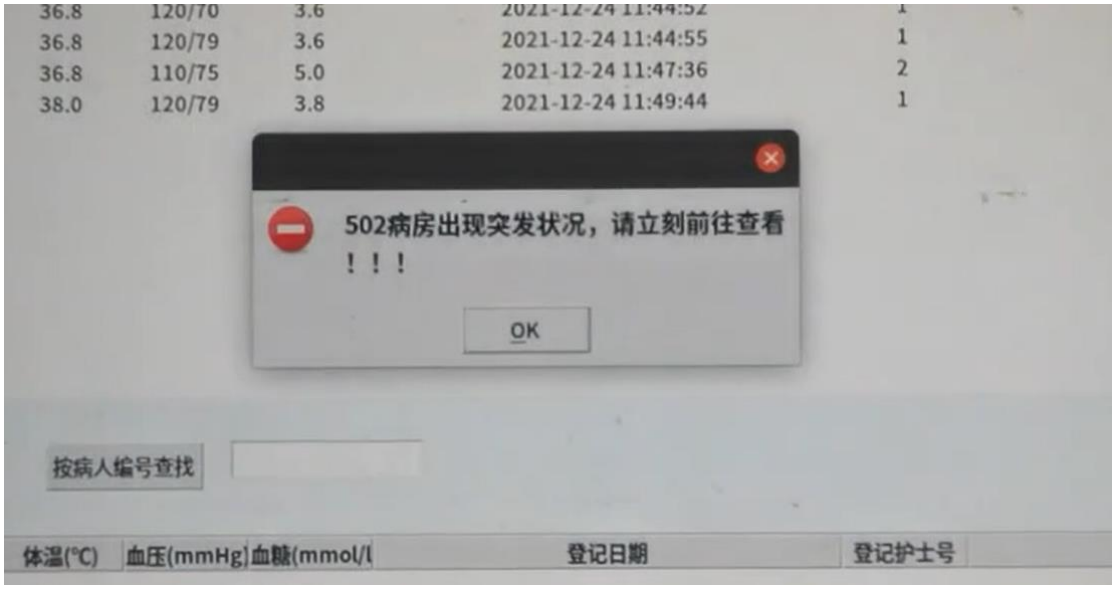

图 1.14 报警提示效果

1.5 实习心得

1.5.1 遇到问题及解决

1.5.1.1 问题一

【问题描述】

在嵌入式系统开发阶段，由于医疗设备的数据是实时更新的，即相应的显示画面需要实时更新，同时由于设备存在接收和发送信息的相关任务，以紧急情况为例，当紧急情况发生时，系统需要立刻向总站进行汇报，但是由于数据更新的程序代码在运行中，可能触发竞态条件导致程序出现错误。

【解决方法】

初步的解决方法是采用中断的方式，将 TCP 的通讯以中断触发形式进行，能够在更新医疗数据的同时不定时的发送和接收数据，但是在调试过程中发现由于 TCP 通讯过于频繁则会不断调用中断，虽然程序能够进入中断但是 TCP 通讯的进行最终导致画面显示的停滞即程序大部分时间处于中断处理阶段并不能很好处理主函数的相关任务内容。

最终，考虑到本次实习的嵌入式板子自带操作系统，可以采用多线程的编程方式解决这一问题，通过将两个线程进行独立的方式，互不干扰且能同时运行。最终测试发现系统能够实时更新画面同时能够不停发送和接收 TCP 报文，虽然可能存在一定的时序问题但是是在可以忍受的范围之内，具体的解决方法可以参考 1.2.1。

【问题解决】

由于多线程的展示不利于文书形式表现，即展示最终标识符的输出表征系统多线程的正常运行如下：

```
STARTING Thread-1
STARTING Thread-2
STARTING Thread-3
STARTING Thread-4
```

1.5.1.2 问题二

【问题描述】

设计总站的登录界面是为了数据的安全性并保证病人的隐私，因此采用了需要账号和密码的相关设计，但是如果采用程序内部使用列表或者是字典的方式进行存储，导致整体程序量大幅上升，及其占用内存空间。

【解决方法】

在嵌入式开发中由于还需要考虑整体的内存性能，因此采用外部文件的方式来搭建相应的用户信息数据库。

pickle 是 Python 的一个标准模块，实现了基本的数据序列化和反序列化。通过 pickle 模块的序列化操作可以将程序中的运行对象信息保存到文件当中，永久存储；通过反序列化操作，能从文件中创建上一次程序保存的对象。序列化和反序列化的操作有利于存储，将文本信息转化为二进制数据流容易存储在硬盘之中，针对 Python 程序中的数据类型有较好的适配性。同时也便于传输，当两个进程在进行远程通信时，采用 pickle 封装的文件可以直接进行发送。

具体的实施代码参考如下：

```
try:
    with open('usr_info.pickle', 'rb') as usr_file:
        usrs_info = pickle.load(usr_file)
except FileNotFoundError:
    with open('usr_info.pickle', 'wb') as usr_file:
        usrs_info = {'admin': 'admin'}
        pickle.dump(usrs_info, usr_file)
# 判断用户名和密码是否匹配
if usr_name in usrs_info:
    if usr_pwd == usrs_info[usr_name]:
        os.popen('python3 ./server.py')
        # server.Smart_ward(logwindow)
        logwindow.destroy()
    else:
        tk.messagebox.showerror(message='密码错误')
# 用户名密码不能为空
elif usr_name == '' or usr_pwd == '':
    tk.messagebox.showerror(message='用户名或密码为空')
# 不在数据库中弹出是否注册的框
else:
    is_signup = tk.messagebox.askyesno('欢迎', '您还没有注册，是否现在注册')
    if is_signup:
        usr_sign_up()
```

【问题解决】

运行程序后，会在目录列表列出新建的 usr_info.pickle 文件同时整体系统支持用户进行登录以及注册，在这里只展示注册界面的结果如图 1.15 所示。

图 1.15 用户注册与信息管理

按照图中方式进行注册后即可以在主登陆界面使用用户名和密码成功登录。

1.5.1.3 问题三

【问题描述】

不管是登录界面跳转至总站主界面还是嵌入式设备从医疗设备信息显示界面跳转至数据上传界面，都存在双界面跳转以及关闭上一界面的问题存在。通过初步尝试 Tkinter 库相比于 Qt 不能很好支持界面跳转的功能，在界面跳转后存在画面或者控件丢失的问题，不符合设计要求。

【解决方法】

os 库时 Python 标准库，提供通用的，基本的操作系统交互功能，可以将两个画面进行多文件编写，通过 os 库的相关命令在条件触发后运行另一个 py 文件，从而实现两个画面之间的跳转。具体代码如下：

```
if usr_pwd == usrs_info[usr_name]:
    os.popen('python3 ./server.py')
    logwindow.destroy()
else:
    tk.messagebox.showerror(message='密码错误')
```

【问题解决】

通过重新运行程序的方式很好解决了多界面跳转的问题，但是这一解决方法也存在一定的局限性，例如两个文件之间的全局变量不能使用，存在无法交互的问题，这一问题会在后续的学习过程中尝试进行解决。由于跳转是过程结果，不方便展示。

1.5.1.4 问题四

【问题描述】

在嵌入式系统设计部分，需要为护士在病房内的数据操作提供修改功能，在实际 SQL 语言的修改操作中，需要确定相应修改信息的主码信息或者是能够对信息进行唯一确认的相关信息，但是通过手动输入的方式过于繁琐，且过于耗费时间，需要一种简便的方式来确定需要修改信息的相关基本信息，方便护士在修改过程中只需要修改部分信息即可。

【解决方法】

在界面设计过程中在数据管理界面左侧预留出了空白区间用于放置 Treeview 控件，可以采用 Treeview 的 selection()方法锁定当前鼠标点击选中信息，从而可以通过触摸屏点击来获取需要修改病人的相关信息，节省了手动输入的大量时间。具体代码如下：

```
r = self.tree.item(self.tree.selection(), "values")
if state[2] == 1:
    query = "update Patient set temp='%s' where Pno='%s'" %
(temp_get, r[0])
    params = ()
    with sqlite3.connect(self.db_name) as conn:
        cursor = conn.cursor()
        db_rows = conn.execute(query, params)
        conn.commit()
```

【问题解决】

针对以上提到的解决方法，系统能够快速进行信息修改，图 1.16 展示了系统原始图片，图 1.17 展示了选中信息后用右边输入栏进行修改后的最终结果。

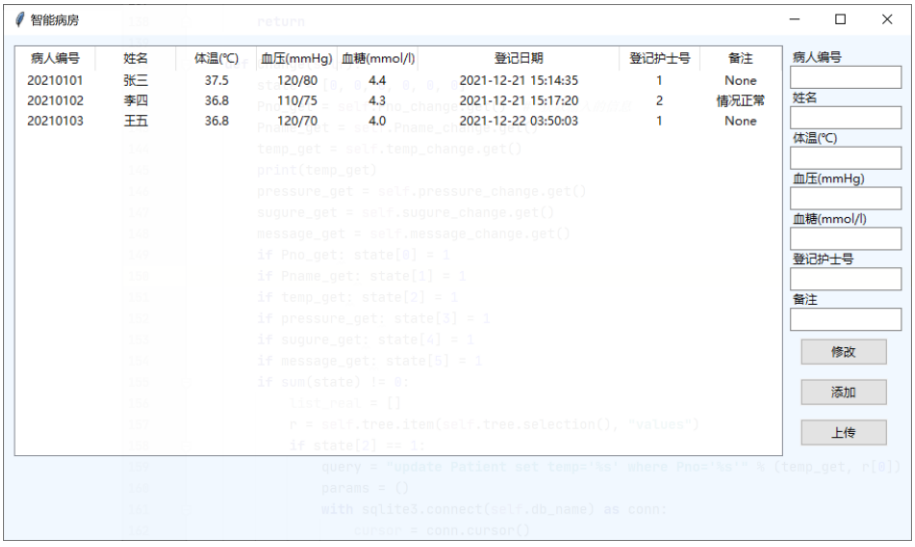

图 1.16 原始界面展示

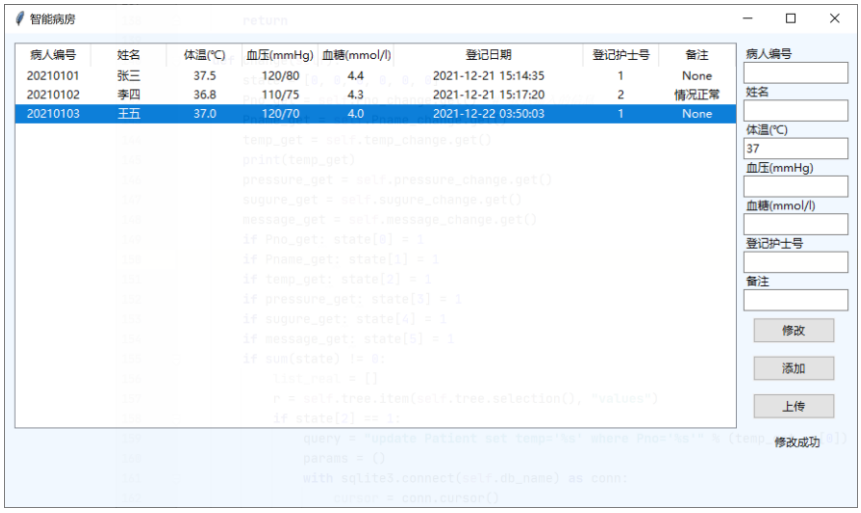

图 1.17 修改信息结果展示

1.5.2 收获感想

通过第一部分的实习，我们小组通过努力搭建了智能病房系统，初步实现了设计要求，包括病房内的医疗设备实时数据监控，紧急情况报警，护士的无纸化登记以及信息的无线上传，同时还开发了相应的总站系统用于信息的汇总于集中管理。

设计内容不仅仅是嵌入式系统相关开发，还包括了单片机开发，数据库系统设计以及计算机网络的相关知识。将整个学期的大部分专业课进行了有机的整合并以一个实习课设的结果形式展现，是一件非常有成就感的事情。

由于时间有限，在短时间内能够完成具有一定规模的系统设计本身是存在一定的难度的，但是小组成员之间互相配合，能够非常合理的进行项目分工，最终整合调试，开发期间能够互相讨论，共同解决问题，这是我们这个设计能够快速完成的最主要原因。

针对该系统的开发，最大的感想是体会到了需求分析的重要性。在项目开发之初，在团队对需求不清晰的情况下快速开工反而容易导致事倍功半，只有真真正正坐下来好好讨论，每位成员都能够对需求有较深刻的理解后，再分工协作才能真正实现事倍功半的效果。

除此之外，我还在实习过程中深刻体会到了硬件平台的关键作用，在此次实习中我们使用的是装载 Ubuntu 的树莓派，其对我们使用的 Python 语言有较好的支持，同时我们的单片机设备是在实习之初绘制完成的，引出尽可能多的功能引脚使更多的功能能够被实现。在一定的硬件基础之上才能进行较为完善的开发。正是应验了一句话：硬件决定下限，软件决定上限。

第二部分：Linux 系统体验实习

2.1 Linux 操作系统

Linux 其实指的是 Linux 内核，各大厂商和团队在此基础上开发出了各式各样的操作系统，例如 Red Hat、Ubuntu 等等。由于并没有学习过操作系统的底层知识，对其理解还仅仅停留在用户的使用体验之上，但是其鲜明特点给我留下了深刻的印象：

相比一些常用的操作系统，Linux 给我的感受是更加“接地气”，没有吸引人眼球的交互界面，而是让用户更加接近操作系统底层，这一特点在命令行体现的淋漓尽致。一个常用 Linux 操作系统的人一定不会采用双击的方式去打开一个文件，而是使用命令行通过输入命令的形式进行打开，更像是用户跟着操作系统一步步走进到文件所处在的空间，并亲手打开想要打开的文件。

初步体验 Linux 操作系统时，会存在上手难的问题，熟练后却又十分便捷。大二是我第一次接触 Linux 操作系统，由于 Windows 使用的习惯性操作，常常点击开文件界面寻找自己所需要的文件，往往打开文件就耗费了我大量的时间。但在长时间的使用过程中，随着对 Linux 操作系统的熟练，通过命令行方式所需要的时间已经远远超过 Windows 的文件打开方式。同时能体现这一点的还有 Linux 的 vi 编辑器，vi 编辑器对于初学者而言可能连输入代码都难以执行，但是在熟练后可以进行一系列的快捷操作，真的是可以出现手指在键盘上飞舞的情形。同时对于 Linux 操作系统的熟练后，对于同样以 Linux 为内核的操作系统其上手非常之快，反观若是从 Windows 资深用户转到 Mac OS 上，往往需要花费大量的时间。由于以前主要使用的是 Ubuntu 操作系统，此次实习在使用 RedHat 的时候能够快速上手，体会到了只需要打开命令行，所有 Linux 操作系统就是同一个系统的独特体验。

在 Linux 开发过程中往往会遇到很多的问题，由于其开源的特性，网上相关的资料很多，针对常规的问题可以很好的解决。但是一些比较少遇到却又比较致命的问题可能会存在解答不全的情况，同时由于官方的文档较少，在进阶开发过程中也会遇到一定的困难，同时也不能完全保证系统的稳定运行。其开源的特性使 Linux 操作系统能够通过下包的方式不断拓展功能，根据需求不断扩充，缩减了系统的初始大小，方便移植至嵌入式进行开发。

2.2 Linux 操作系统移植

所谓系统移植，是将已经有的软件，根据硬件平台的差异，做少量的代码修改或者裁剪，使得软件可以在新的硬件平台上运行起来的过程。Linux 是一个通用的内核，获取源码后，需要先经过相应的配置，使其与硬件平台相匹配后才能进行编译和安装。

本次实习过程中，针对系统移植主要包括 bootloader、Linux 内核、根文件系统的移植工作。其在内存中的存放如图 2.1 所示。

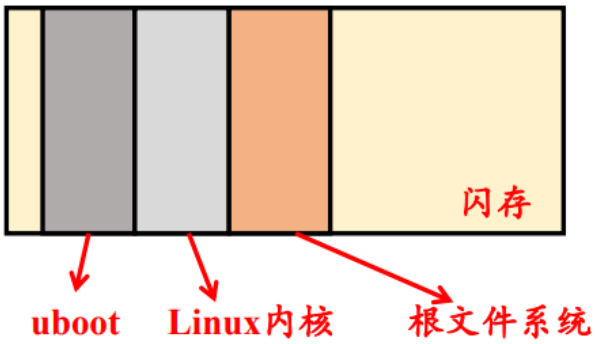

图 2.1 Linux 操作系统移植内存存放示意图

uboot 是一个主要用于嵌入式系统的引导加载程序，可以支持不同的计算机系统结构，也是一套在 GNU 通用公共许可证之下发布的自由软件。使用 Bootloader 主要是为了初始化软硬件环境，引导加载 Linux 内核，执行用户命令。

Linux 内核必须要由 bootloader 加载和启动，可以看成一个大的逻辑程序，自上而下运行最后进入 while(1)循环，内核加载后，会根据 bootloader 传递的参数去闪存中寻找根文件系统，并将内核控制权转交给根文件系统。

当内核根据参数找到根文件系统后，运行/sbin/init 第一号进程，随后便等待输入命令。

按照个人理解，uboot 作为底层驱动，更像是为后续的系统移植配置好环境，根据后续的内核和根文件系统进行特殊定制。而 Linux 操作内核则是系统的主要部分，开启整体的运行。根文件系统构建出系统的主体文件内容，也是平时在使用 Linux 系统中的文件内容，是系统必不可少的部分。

三者之间环环相扣，一个环节的问题都会最终导致系统运行失败。在实习过程中，根文件系统出现过编译不正确的问题，导致整个系统就一直处于重启检测的状况并不能正常进入系统。

2.3 驱动开发

一个嵌入式系统的 Linux 操作系统没有相应的外设驱动会受到许多的限制，例如键盘鼠标 U 盘无法正常使用，难于在嵌入式设备上进行操作。同时开发板往往自带许多的外设包括 LED 灯，蜂鸣器，摄像头，音频输入输出等，不添加相应的驱动程序便不能进行使用，最终导致无法满足嵌入式系统开发的要求。

Linux 对驱动程序有统一的框架，以文件的形式定义系统的驱动程序。往往是作为文件系统的一部分加入到内核当中。这一特点与 Linux 自身的包管理十分相似，Linux 可以通过下包的方式给自身添加功能，Python 也有类似的机制，通过下载功能包来扩展功能，这种模块化的方式可以在需要驱动时进行添加，在不再需要驱动时进行删除，十分方便。

驱动程序的根据实际需求进行选择下载在达到同样功能的前提下一定程度可以减小系统内核的整体大小，十分适合嵌入式这种内存性能存在限制的开发环境。

2.4 基于 Linux 系统应用开发与裸机应用开发

在嵌入式实习之前，我们更加熟悉的是裸机应用开发，这种开发方式更加接近于单片机的应用开发，通过在 PC 机上进行编写程序、编译程序最后将程序烧录到开发板中，而基于 Linux 系统的应用开发提供了操作系统的相关支持，能够实现在开发板上直接进行开发，同时其支持的功能更加丰富。

以显示界面开发为例，在裸机开发中采用的方式是对显示屏像素点进行操作，通过对像素点的操作最终实现图像的显示，这种操作方法针对大型的显示屏幕存在很大的弊端，虽然能够支持自定义设计但是设计结果往往不太美观。反观基于 Linux 的操作系统，使用 Qt 等开发软件可以十分方便的设计出美观的图形交互界面，其提供的外设功能也更加丰富。

在此次实习中体会较深的还有基于操作系统的多线程机制，在裸机开发中更多使用的是中断触发的方式使程序进行跳转，但是多个程序跳转存在中断优先级处理等一系列问题。而操作系统支持多线程的开发，可以很好解决这一问题，进一步说明了嵌入式开发选用基于 Linux 操作系统的应用开发的优越性。

2.5 实习心得

2.5.1 遇到问题及解决

2.5.1.1 问题一

【问题描述】

将 uboot，Linux 内核，根文件系统编译后装载至开发板后启动发现串口汇报错误：

```
Failed to execute /linuxrc. Attempting defaults…
Kernel panic -not syncing: No init found.
```

【问题解决】

通过串口信息可以知道系统使根文件系统的编译出现问题导致系统不能正常运行，uboot 和 Linux 内核的编译结果可以正常使用。在遇到错误时，尝试按照网上的操作进行，例如关闭看门狗，将文件系统下 etc/inittab 文件属性改为 777 等操作，包括更换操作系统均得到了相同的错误信息。

最后在老师的提醒下发现初始的压缩包文件不能在 Windows 环境下进行解压操作，而是要将压缩包移放至 Linux 环境下进行解压，最终才编译通过。

2.5.1.2 问题二

【问题描述】

在实习阶段，我们组还尝试了将 Qt 移植到 TQ2410 的开发板之上，设计相应的相应的 Qt 界面，但是在移植环境过程中就遇到了问题，首先是 Qtopia 的移植，由于 Qtopia 只支持 32 位的操作系统，我们尝试 64 位操作系统并下载 32 位依赖包的方式进行编译尝试，还是会出现错误，当采用实验室 32 位 Redhat 操作系统时，会存在 g++编译器缺失的相关问题，随后尝试使用 32 位的 Ubuntu 系统发现出现了文件.h 的丢失，即压缩包内存在文件丢失。

在 Qtopia 编译失败后我们又尝试了将 Qt4 进行移植，但是到最后还是存在编译器无法使用的问题，查阅网上所有的相关资料，由于是比较老的开发板，很多方法存在不适用的问题，最后也被迫放弃。

【问题解决】

虽然最终没能解决问题，但是在编译的实际过程中针对一些常见问题还是得到了很好的解决，在此不一一罗列，而是将主要的问题以链接的形式展示，可以

便于查看。

- https://blog.csdn.net/ting1231/article/details/24987167
- https://my.oschina.net/guizimo/blog/4268087
- https://zhidao.baidu.com/question/2268774367369084908.html

2.5.2 收获感想

经过第二部分的实习内容，初步尝试了基于 Linux 的系统移植，包括 uboot，系统内核以及根文件系统的移植。除此之外还尝试了将 Qt 环境在开发板上进行部署。

这部分实习是本次实习过程中花费心思最多的一部分，也是最有遗憾的一部分。我们选择提前进入 Linux 的开发部分，在没有相关移植经验的情况下尝试烧录从而完成配置，按照官方给出的文档一步步执行，但是总能遇到一定的问题。有种完闯关类游戏的感觉，在解决一个问题后紧接着就回来另一个问题，接近一周的时间我们就是在这种解决问题遇到问题的过程中不断循环。

在系统移植部分，我们遇到了根文件系统无法正常读取的问题，按照网上的方案在各个贴吧，网站寻找答案，都无法解决问题，最后是由于将压缩包在 Windows 中解压导致，这一教训给我留下了深刻的印象。

在完成系统移植后，我们尝试给开发板下载 Qt 的环境，这一部分才是真正困难的开始，为了这个问题，那一周基本每天需要熬夜到三点钟，很可惜还是没有在实习结束前成功解决编译的问题。

塞翁失马焉知非福，在整个实习过程中，虽然最后没能够实现 Qt 的编译，但是整个过程中对于 Linux 操作系统有了更加深刻的体会，对于系统的移植丰富了经验。最明显的改变就是在命令行使用时的数目，在不断的输入 cd 命令后，感觉整个 Linux 系统的文件树就复制在了自己的脑海中，真正实现了指哪打哪，能够快速访问文件。同时也对于系统的快捷操作更加熟悉，相信这一段经历必定对我未来的工作或是学习都大有裨益。

在实习过程中也感慨技术发展之快，由于实习采用的板子比较老，导致开发的系统或者工具并不能完全适配，从而导致问题不能及时解决。最终我们采用的是树莓派搭载 Ubuntu 的方案，在系统设计部分也得到了很好的性能体现。

最后，还是要感谢指导老师在实习期间的帮助，老师愿意花费自己的时间帮我们寻找解决方案并分享经验，在很多次我们接近放弃时，老师总会给我们一些不同的看待问题的角度，让我们能够继续进行尝试，对此不甚感激。同时也要感谢所有为了此次嵌入式实习辛苦付出的老师们和助教们，也祝大家新年快乐，工作顺利！