廣東工業大學

课 程 设 计

课程名称 单片机课程设计说明书

题目名称 基于 stc15 单片机的指纹考勤器

学生学院 材料与能源学院

专业班级 15 级电子材料（2）班

学　　号 3115006353

学生姓名 吴荣东

指导教师 刘　俊

2017　年　6 月 28 日

摘　要

随着现代化各种科学新技术的快速发展，在日常生活中，我们需要各种身份认证和各种密码认证，还有对各种设备配备钥匙，对保险柜安装防盗系统等等，社会的进步，科技的发展，促使传统的安全系统的抵御能力越来越薄弱。在日常的工作和学习生活中，需要登记、出示证件，记录打卡的时间和登记人，而这个过程在传统技术手段下，很难做到准确，快速，浪费了宝贵的时间，对于接下来的工作和学习有一定不便的影响。

因此，生物特征识别应用而生，开始走进我们身边的各种安全系统，指纹识别作为生物特征识别的一个典型应用已经得到很广泛的应用和认可，指纹特征具有唯一性，是每个人终生不变的特征之一，并且各个人的各个指纹都不一样。指纹识别是近几年来被广泛应用于手机的技术，具有安全、快速，准确的特点，省去了输入密码和记住密码的烦恼。传统的指纹识别技术主要分为三类：光电式，电容式和超声波。鉴于超声波指纹识别目前还未得到广泛应用，在预算允许范围下，本次设计采用光电传感式指纹识别模块。本次设计主要利用指纹模块的快速识别，大容量的特点，结合所学单片机知识，实现指纹考勤的功能。

本次设计的指纹考勤器，采用STC公司的STC152K60S单片机作为指纹考勤器的控制核心，城章科技有限公司的R307指纹传感器作为指纹考勤器检测核心，结合OLED显示屏如按键输入、LED灯报警电路、蜂鸣器电路，最后通过编写软件和制作硬件，实现一个可以通过单片机对指纹的录入，识别，删除等功能操作的指纹识别系统,整个由单片机进行控制，使其可以稳定长期运行。

关键词：指纹识别，OLED 显示屏，考勤，单片机控制

Abstract

With the rapid development of the new technology of modern science, in daily life, we need a variety of identity authentication and password authentication, and on a variety of devices equipped with keys, the safe installation of anti-theft system and so on, the progress of the society, the development of science and technology, the traditional security system is more and more weak ability to resist. In their daily work and study life, the need for registration, identification, recording time and registration card, and this process in the traditional method, it is hard to be accurate, fast, wasting valuable time, for the next work and learning some inconvenience.

Therefore, the application of biometrics and security system, began to enter the US, fingerprint recognition has been widely used and recognized as a typical application of biometric fingerprint is unique, the characteristics of each person is a lifelong one, and each fingerprint of each individual are not the same. Fingerprint identification has been widely used in mobile phones in recent years. It has the characteristics of security, speed and accuracy. It eliminates the trouble of inputting passwords and memorizing passwords. Traditional fingerprint identification technology is mainly divided into three categories: photoelectric, capacitive and ultrasonic. In view of the fact that ultrasonic fingerprint identification has not been widely used, the design of the fingerprint sensor module is based on the photoelectric sensing module. This design mainly uses the quick identification of fingerprint module, the characteristics of large capacity, and combines the knowledge of SCM to realize the function of fingerprint attendance.

The design of the fingerprint attendance device, using STC's STC152K60S microcontroller as the control core of fingerprint attendance machine, City Chapter Technology Co. Ltd. R307 fingerprint sensor as a fingerprint attendance detector core, with OLED display as a key input, LED light alarm circuit, a buzzer circuit, finally through the software and hardware implementation. One can input, SCM for fingerprint recognition, delete and other functions of fingerprint identification system, the microcomputer control, so that it can run stably for a long time.

Key words: fingerprint identification, OLED display, attendance, SCM control

目录

1. 绪论

1.1 题目背景及目的

1.1.1. 背景

随着现代化各种科学新技术的快速发展，在日常生活中，我们需要各种身份认证和各种密码认证，还有对各种设备配备钥匙，对保险柜安装防盗系统等等，社会的进步，科技的发展，促使传统的安全系统的抵御能力越来越薄弱。同时在日常的工作和学习生活中，需要登记、出示证件，记录打卡的时间和登记人，而这个过程在传统技术手段下，很难做到准确，快速，浪费了宝贵的时间，对于接下来的工作和学习有一定不便的影响。

1.1.2. 目的

借此次单片机课程设计，希望能将所学单片机知识应用于实际中，打算打造一个实用的单片机控制系统，能够解决日常生活中的问题，带来一定的便利。

1.2 题目研究内容和方法

随着现在时代的变化，科技的进步，传统的一些安全系统已经正在变得越来越脆弱了，因此，更先进更高级的生物识别技术开始走进我们的身边，出现在各种各样的安全系统中。比如人脸识别、指纹识别、眼球视网膜识别等，而指纹识别作为生物识别里面一个比较成熟的已经获得了各行业界的认可的识别系统，广泛的被应用到安全系统中去。指纹识别具有唯一性，每个人的每一个指纹都是独一无二的，可以说指纹是一个人身份的标志。本系统正是利用指纹识别安全,简单,快速的特点,将指纹作为考勤的手段,配合计算机的数据库,便可以实际指纹考勤目的。

本系统采用的是 STC15 系列的单片机作为主控 MCU，由于其低功耗、编程灵活简单、外围设备丰富，性价比高，简言之就是简单好用便宜，在本次设计中作为主控芯片，和指纹模块进行串口通信，与 OLED 显示屏进行 SPI 通信，加上 DS1302 时钟芯片和其他一些外围器件，可以很方便地完成指纹考勤器系统所需的所用功能，这对于我更好地理解单片机的控制和各种通信方式以及指纹识别原理很有帮助。

1.3 论文构成

本文主要分成五个部分，分别是绪论部分，系统的整体概述部分，硬件的设计部分，软件的设计部分和调试部分。

第一部分阐述了本课题的研究背景和意义，同时介绍了生物识别特征的几个识别手段，最后介绍了实现指纹考勤器功能所需的知识。

第二部分介绍了本文设计的要求，包括整个系统的原理框图，系统各部分的主要功能，最后根据要求设计出合理的具体方案。

第三章介绍了本设计的硬件部分，包括主要的元器件选型，各个电路的设计和分析。

第四部分介绍了本设计的软件部分，部分功能的具体介绍，最后还简单介绍了开发环境。

第五部分介绍了本系统的焊接和调试过程，主要是在整个系统的调试过程中遇到的问题和解决方法。

其他是对本文的一个总结，以及提出对整个设计的一些不足。、

2. 指纹考勤器总体概述

2.1 设计要求

本设计要求如下：

1. 系统默认至少有一位管理员，没管理员的时候提示必须添加管理员方可操作；
2. 普通用户可以添加 300 位；
3. 录入指纹、删除指纹和清空指纹的时候必须管理员验证，验证成功才可以添加或删除用户，否则不可以添加用户；
4. 可以查看管理员和普通用户的数量；
5. 可以进行指纹识别的打卡功能；
6. 可以对打卡的结果进行查看；
7. 通过以上要求，制作出一套具有软件和硬件相结合的指纹考勤系统。

2.2 设计方案

通过上节的设计要求，根据搜集资料并且结合自身所学知识，最后制定的本设计方案如下：

本设计以 51 单片机 STC15F2K60S 作为主芯片，选取了指纹识别模块 R307 进行二次开发，该模块采用串口通信方式，按照指纹系统自定义的协议来跟单片机通信，单片机按照固定的协议去读取指纹系统的数据，同时对指纹系统发送指令进行控制，从而实现指纹的操作；而显示器选用 OLED 显示屏，液晶在系统运行中和各个不同功能模式的时候显示对应的提示内容；选用 DS1302 时钟芯片实时获取当前的时间和日期，供打卡时得到打卡的时间；设计要求可对指纹进行录入、识别、删除等操作，通过不同的按键来完成，本设计采用了 4 个独立的按键，分别对应屏幕上显示的菜单，左，右，和返回，软件部分采用轮询检测 IO 电平来判断按键值；报警提示选用的元件是蜂鸣器和 LED 灯。

3. 指纹考勤器的硬件设计

3.1 概述

指纹考勤器系统设计的第一步是硬件系统的设计，这对系统运行的稳定、控制的精确度有着基础性的重要作用。指纹考勤器系统主要由以下几部分构成：电源和烧写模块、核心板模块、指纹识别模块、OLED 显示模块、DS1302 模块、其他外围器件如蜂鸣器，按键和 LED 灯，指纹考勤器总体原理框图如图 3.1 所示。

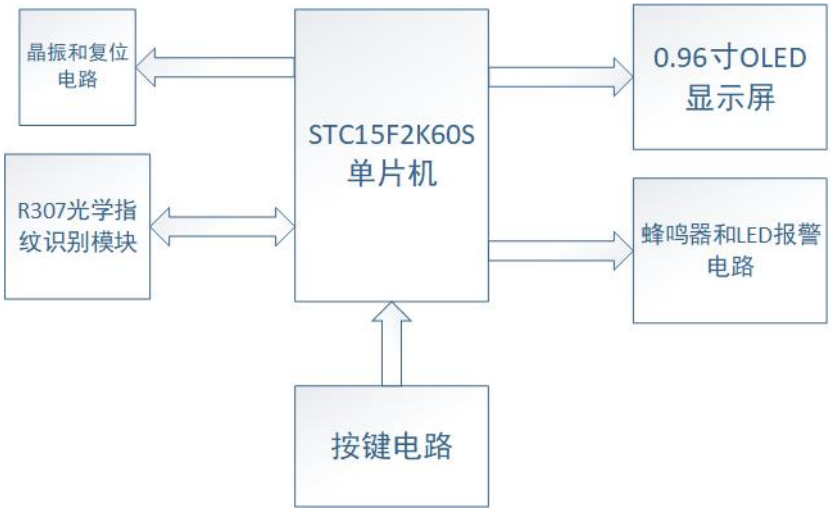


图 3.1 系统原理框图

3.2 核心板模块

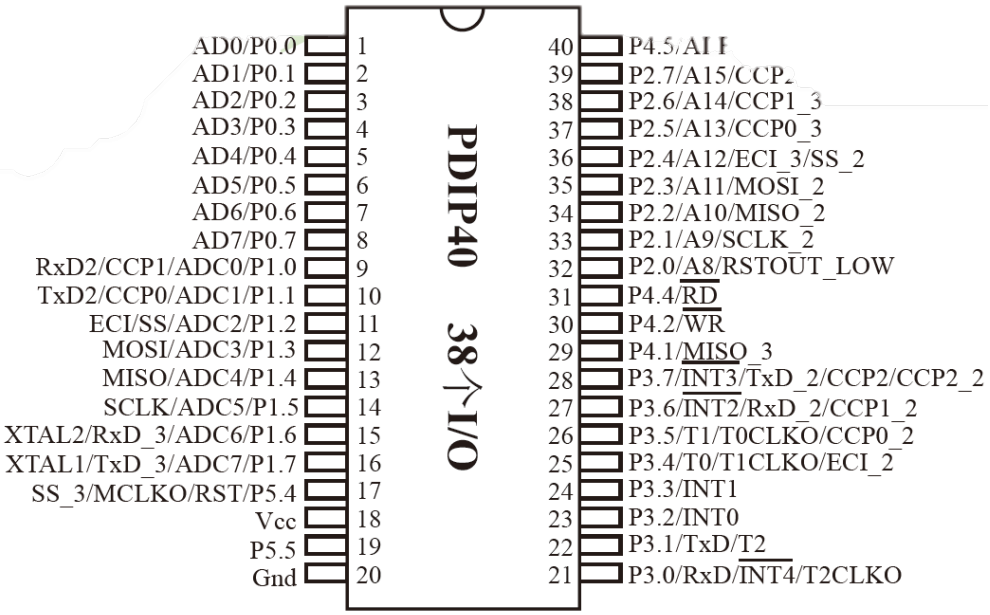


图 3.2-1 STC15F2K60S 管脚图

本设计中需要用到跟指纹模块进行通信，而经过对单片机与模块通信方式的多种比较后，决定采用比较简单的串口通信。串口通信的方式，几乎在任何一款单片机都有硬件支持的，只不过有的串口有多个，有一些少而已。按照之前自己接触过的单片机，stc15系列单片机作为本次的单片机主控芯片，功能上是完全可以实现的，所以本次决定在STC89C52RC 单片机和 STC15F2KS60K 这两个中选出一个作为主控芯片。两者都是 8051 内核，不同点有一下几点：

1、STC15F2KS60K 是 1T 模式的，比普通的 51 单片机 89C52 可以快 6~8 倍，而定时器，串口为了兼容传统的 51 单片机，是可以设置为 1T 模式或者 12T 模式的；

2、STC15F2KS60K 有两个独立的串口，而 89C52 只有一个串口，在调试过程中，有多个串口是最好的，可以调试看信息，找问题所在。

3、作为重要的对比，STC15F2KS60K 的 RAM 有 1280 个字节，ROM 有 60K 的存储空间，可见 STC15F2KS60K 的容量是非常大的，而 89C52 的 RAM 和 ROM 就相对小很多，分别是 512 字节和 8K，作为本次的功能，远远不够。

经过以上选择对比，最后决定选用 STC15F2KS60K 作为本次的主控芯片,其内部高可靠复位,可彻底省掉外部部位电路,需要复位只需给系统重新上电即可,对于频率的选择只需要在下载程序时选择适当的频率即可,本次选用的时 11.0592MHz 的频率.在 VCC 和 GND 之间就近加上电源去耦电容 C1(0.1uF),可去除电源线噪声,提供干扰能力.其电路图如图所示:

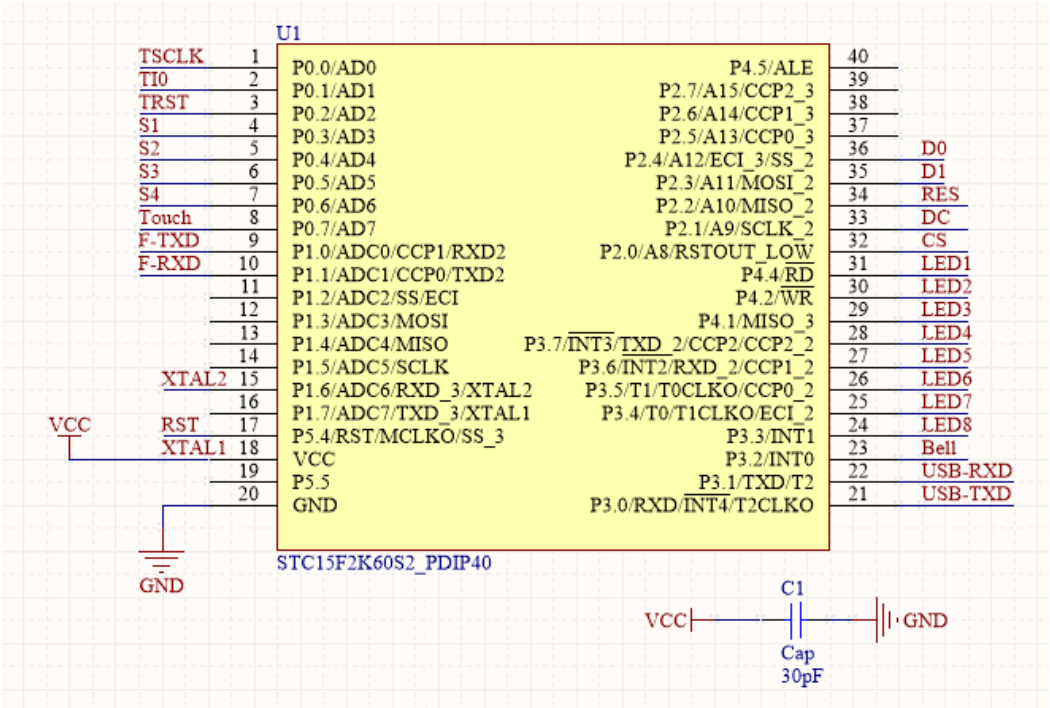


图 3.2-2 核心板模块电路图

3.3 USB 烧写模块

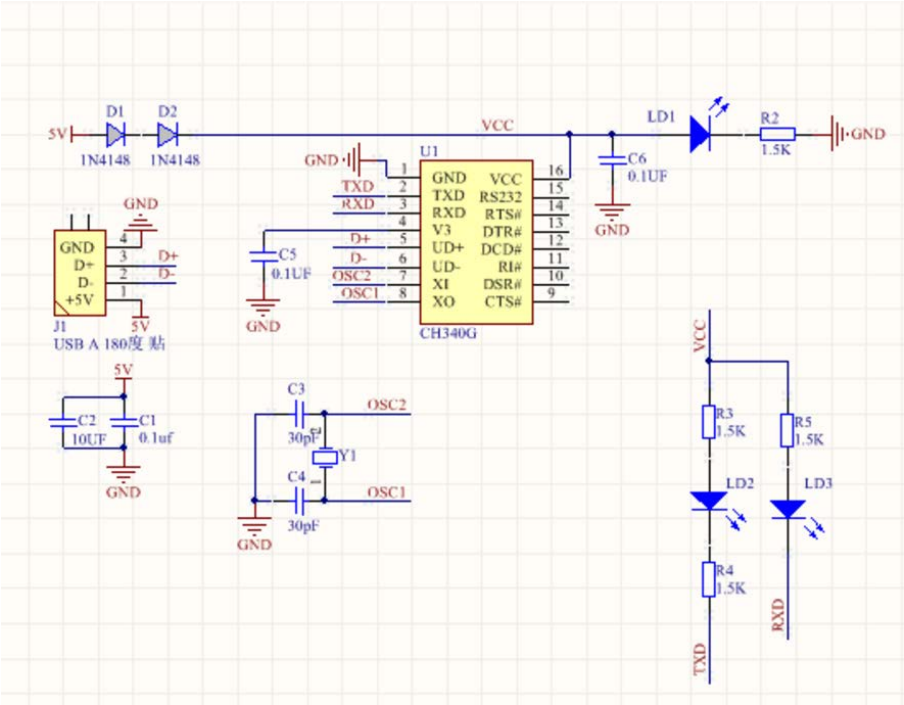


图 3.3-1 CH340g 转 ttl 小板内部电路图

CH340G 芯片，该系列芯片是 STC 官方推荐的 USB 转 TTL 芯片，不会出现由于不同的驱动/不同的电脑等导致不兼容现象，与单片机进行串口通信，有电源灯 PWR，还有 TXD 和 RXD 指示灯，方便在没有仪器的情况下直观的了解到产品是否在工作；3V3 和 5V 由短路冒选择，可的下载程序到 stc152K60 并为整个系统供电，支持 WINDOWS 10 32 位/64 位。下面是 USB 接口电路图：

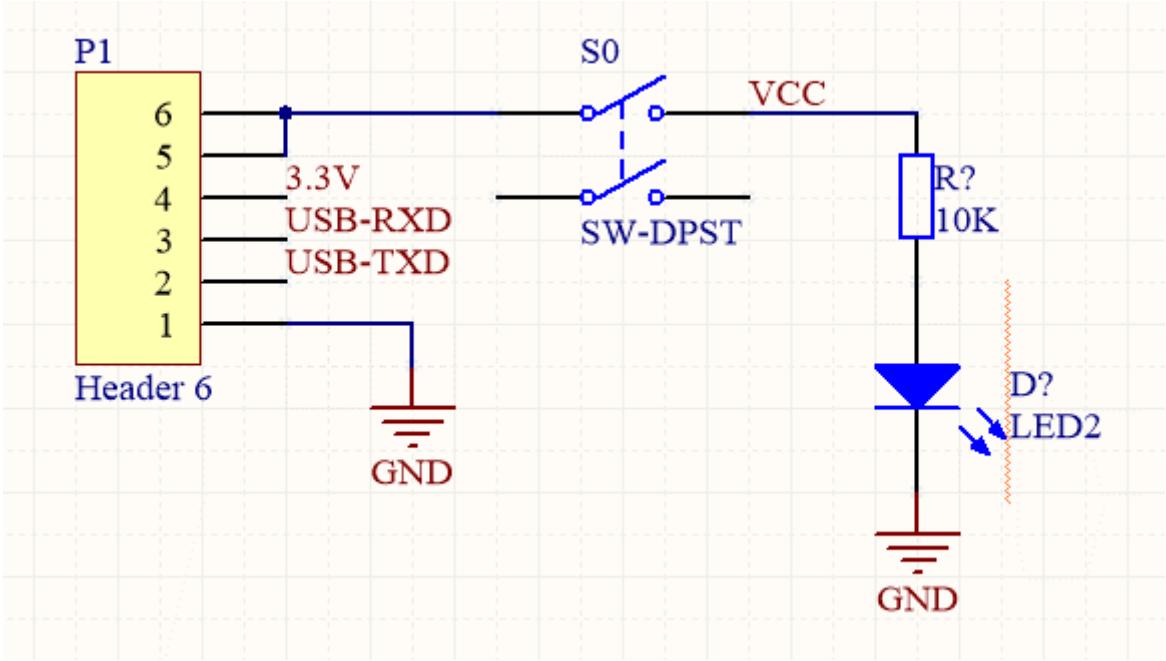


图 3.3-2 USB 接口电路图

3.4 指纹识别模块

图 3.4-1 指纹识别模块实物图

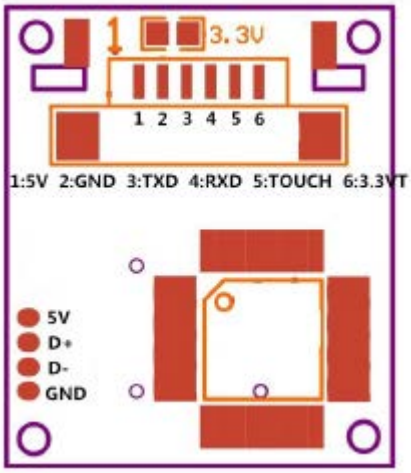


图 3.4-2 指纹识别模块接口电路图

作为本次的重要传感器，指纹识别模块无法自己去做，只能通过网上淘宝买现成的模块回来进行二次开发，市场有各种各样的指纹模块，价格也是五花八门，但实际上基本的协议还是比较相似的，只不过有的支持的协议指令多，有一些支持的指令相对来说比较少而已，本次需要用到的功能有录入指纹，识别指纹，清空指纹，显然这

三个指令，几乎所有的模块都可以实现，根据最后筛选和价格比对，最后决定选用R307 作为本次的指纹识别模块。

R307 指纹识别模块是城章科技有限公司 2015 年推出的最新产品，采用了最先进的指纹传感器和高性能的 DSP 处理器，内嵌完整的指纹识别算法和协议。具有指纹采集，指纹比对，搜索和存储等功能的智能型模块。

技术参数：供电电压：DC　4.2-6.0V

供电电流：工作电流：50mA（典型值）峰值电流：75mA

指纹图像录入时间：＜0.3 秒

匹配方式：比对方式（1:1）搜索方式（1:N）

特征文件：256 字节

模板文件：512 字节

存储容量　1000 枚

安全等级：5 级　　认假率(FAR)：＜0.0001%　拒真率(FRR)：＜1.0%

搜索时间：＜100ms　（1:500 时，均值）

上位机接口：UART

通讯串口波特率：（9600 * N）bps,N=1-12（默认 N 取 6，即 57600bqs）

R307 指纹模块通讯接口定义：

引脚号	名称	定义	描述
1	5V	电源输入	电源正输入端(DC4.2V- -6V)
2	GND	电源和信号地	电源和信号地
3	TXD	数据发送	串行数据输出，TTL 逻辑电平
4	RXD	数据接收	串行数据输入，TTL 逻辑电平
5	TOUCH	触摸信号输出	手指触摸感应信号输出，有手指时输出低电平
6	VT	触摸电源输入	触摸感应电路电源（DC3--5V、约5uA）

表格 3.4-3 R307 **指纹模块通讯接口定义**

由于本次采用的指纹识别模块是集成的模块，跟单片机通信采用的是串口方式，波特率是 57600,8 位数据，一位停止位，无校验。15F2K60S 单片机有两个串口，所以用串口 2 跟指纹模块通信，单片机的 RXD_2 接指纹模块的 TXD，而单片机的 TXD_2 接指

纹模块的 RXD。通过指纹模块的数据手册，发送对应的指令给指纹模块，即可实现录入指纹、识别指纹、清空指纹等操作。

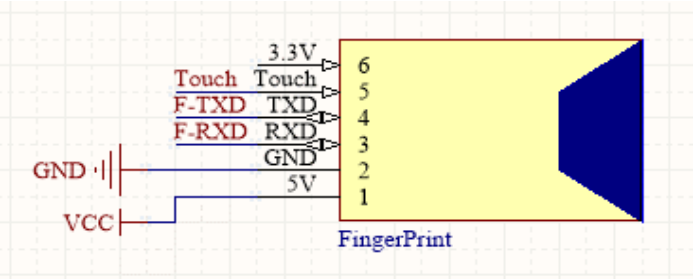


图 3.4-4 指纹识别模块电路图

3.5 OLED 显示模块

作为人机交互界面，显示器起到一个至关重要的元件，本文中需要将整个操作的过程完全显示在显示器上面，比如录取指纹，识别指纹是否正确还是错误 等等。下面就本设计的过程中对显示器的元件选型进行分析和对比

方案 1:

采用液晶 LCD1602 作为显示器，供电电压有 3.3V 和 5V 两种，能够同时显示 16*2 个字符，16 列 2 行，其内部模块里面已经存储了 160 多个我们平时很普遍用到的的点阵字符图形，每一个字符符号都有一个固定的代码编码，只需要发送对应的代码编号给液晶模块，就会自动显示出来对应的字符，共有 16 个引脚，和单片机通信采用的是并行通信方式，即 8 个 IO 口，该模块优点是可以显示基本的字符符号，价格便宜。但是也有其不足的地方，就是不能显示中文，而且只能是显示 2 行，和单片机通信需要 8 个 IO 口。加上 3 个控制引脚，一般都需要用到 11 个 IO 口。

图 3.5-1 液晶显示屏 1602 实物图

方案 2:

采用液晶 12864，显示器 12864 液晶，可以显示中文，并且是自带字库，字库中有

几千个常用的汉字，用起来基本是可以满足的，接口也是比较灵活，可以选择并行或者串行接法，串行接法只需要用到两根线。液晶的引脚图如下图所示。液晶的 D0~D7 是数据引脚，当液晶作为并行通信的时候，单片机要连接这 8 个数据口，而液晶的 PSB 引脚是选择并行或者串行的引脚，当低电平时为串行方式，当高电平时为并行方式。

图 3.5-2 **液晶显示屏** 12864 **实物图**

方案 3:

OLED，即有机发光二极管（ Organic Light Emitting Diode）。 OLED 由于同时具备自发光，不需背光源、对比度高、厚度薄、视角广、反应速度快、可用于挠曲性面板、使用温度范围广、构造及制程较简单等优异之特性，被认为是下一代的平面显示器新兴应用技术。LCD 都需要背光，而 OLED 不需要，因为它是自发光的。这样同样的显示 OLED 效果要来得好些。以目前的技术，OLED 的尺寸还难以大型化，但是分辨率确可以做到很高。在此我们使用的是 0.96 寸 OLED 显示屏，该屏有以下特点：

1) 0.96 寸 OLED 有黄蓝，白，蓝三种颜色可选；其中黄蓝是屏上 1/4 部分为黄光，下 3/4 为蓝；而且是固定区域显示固定颜色，颜色和显示区域均不能修改
2) 分辨率为 128*64
3) 多种接口方式；OLED 裸屏总共种接口包括：6800、8080 两种并行接口方式、3 线或 4 线的。

图 3.5-3 OLED **显示屏实物图**

模块接口定义：

1. GND 电源地
2. VCC 电源正（3～5.5V）
3. D0 OLED 的 D0 脚，在 SPI 和 IIC 通信中为时钟管脚

4. D1 OLED 的 D1 脚，在 SPI 和 IIC 通信中为数据管脚

5. RES OLED 的 RES#脚，用来复位（低电平复位）

6. DC OLED 的 D/C#E 脚，数据和命令控制管脚

7. CS OLED 的 CS#脚，也就是片选管脚

综上三个方案对比，最后选择 OLED 显示屏作为本次的显示器模块，OLED 显示屏内部所用的驱动 IC 为 SSD1306；其具有内部升压功能；所以在设计的时候不需要再专一设计升压电路。SSD1306 的每页包含了 128 个字节，总共 8 页，这样刚好是 128*64 的点阵大小，与单片机是以 SPI 方式通信，故只需将对应的管脚连接到单片机的 IO 口，通过程序即可控制 OLED 显示内容。电路图如图 9 所示：

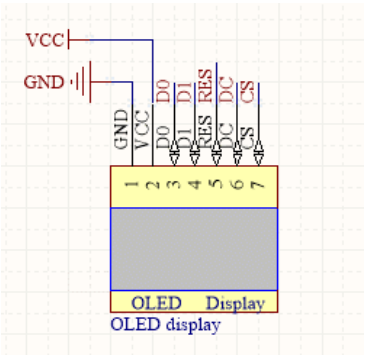


图 3.5-4 OLED 显示屏电路图

3.6 DS1302 模块

DS1302 是 DALLAS 公司推出的涓流充电时钟芯片内含有一个实时时钟/日历和 31 字节静态 RAM,可通过简单的串行接口与单片机进行通信

可提供:

--秒分时日日期月年的信息

--每月的天数和闰年的天数可自动调整

--可通过 AM/PM 指示决定采用 24 或 12 小时格式

--保持数据和时钟信息时功率小于 1mW

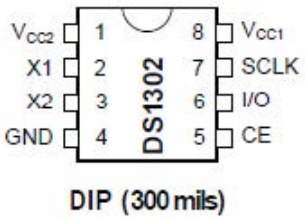


图 3.6-1 DS1302 时钟芯片管脚图

Vcc1：主电源；Vcc2：备份电源。当 Vcc2>Vcc1+0.2V 时，由 Vcc2 向 DS1302 供电，当 Vcc2< Vcc1 时，由 Vcc1 向 DS1302 供电。

SCLK：串行时钟，输入，控制数据的输入与输出；

I/O：三线接口时的双向数据线；

CE：输入信号，在读、写数据期间，必须为高。该引脚有两个功能：第一，CE 开始控制字访问移位寄存器的控制逻辑；其次，CE 提供结束单字节或多字节数据传输的方法。

DS1302 模块的电路图如图所示:

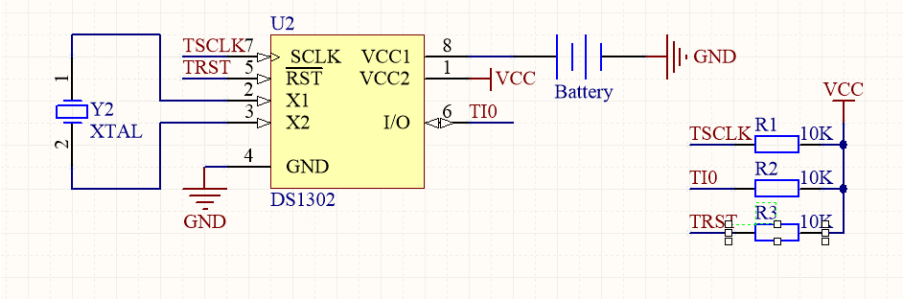


图 3.6-2 DS1302 时钟芯片电路图

3.7 其他外围器件

3.7.1. 蜂鸣器电路

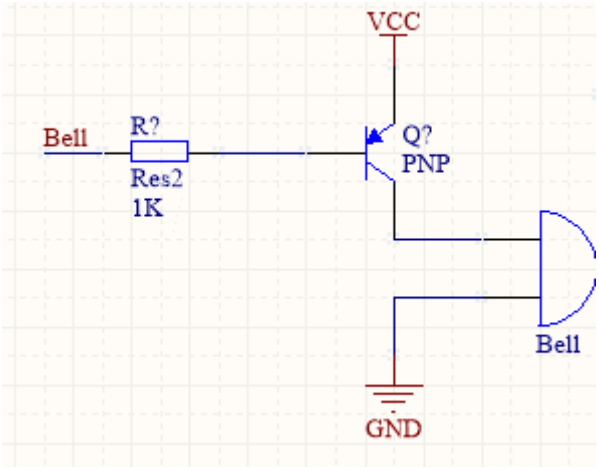


图 3.7-1 蜂鸣器电路图

本次设计中，由于需要在录入指纹和识别指纹中对正确或者错误做出判断，因此在电路设计中加入了蜂鸣器来作为提醒报警响应的作用。采用的是直流电压 5V 供电的有源蜂鸣器，但是一般的蜂鸣器在整个系统运行中需要比较大的电流，51 单片机的 IO 口除了 P0 口外其他的都有一个弱上拉电阻，但 IO 的驱动电流也是非常低，不加外部的驱动电路是很难直接驱动的。而最常用的就是用普通的三极管来放大电流驱动，该三极管在电路中的作用是开关电路并且放大电流，P37 高电平为蜂鸣器响，低电平蜂鸣器停止。报警提醒电路如图 10 所示。

3.7.2. 独立键盘电路

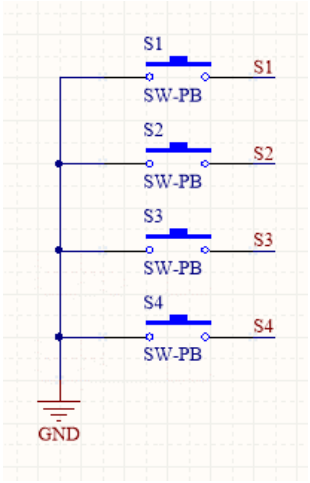


图 3.7-2 独立键盘电路图

设计中有录入指纹功能、识别指纹功能、清除指纹库功能，这些功能的切换都是通过按键来实现的，按键一般有独立按键，矩阵键盘等，由于本次涉及到的按键不需要很多个，因此采用独立按键的形式，简单方便易操作，而按键电路用的是直接练到单片机的 IO 上面，通过对该 IO 口电平读取不同的值来区别按下还是弹开，按键用的是 4 脚按键，对角的两个引脚为一对，是导通的，随便一个引脚接地，对角 IO 接入单片机，按下按键时，4 个引脚都接通，即与单片机连着的引脚也被拉低，单片机 IO 也响应被拉低，软件就认为此时按键被按下，执行按下操作，由于按下过程中有电压毛刺，所以软件一般采用 10MS 的延时来代替消抖动作。

3.7.3. LED 指示电路

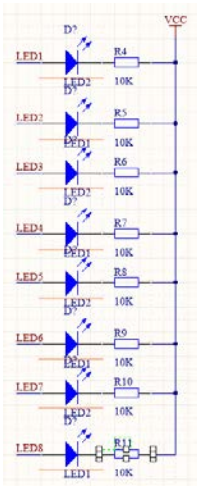


图 3.7-3 LED 指示电路图

LED 作为电子电路中最常用的指示作用，本次中也用到了两个 LED 灯来作为系统的提示灯作用，一个为红色，一个为绿色，LED 作为二极管的一个最典型元件，其两端也是有一个压降，一般 1.3V 左右，根据电子知识，只要在两端上有超过这个的压降，二极管就导通，为了保险起见，在单片机的 IO 口接上一个 1K 的限流电阻，不至于上电时容易烧坏。

4. 指纹考勤器的软件设计

4.1 概述

控制电路主要由一片 STC15F2K60S 单片机组成，STC15F2K60S 单片机主要通过接收独立键盘和指纹识别模块的输入，ds1302 时钟模块的数据,来调用指纹驱动程序和 OLED 显示屏里的驱动程序函数，来实现相应的功能。

利用此原理将指纹识别模块采集到的信号转换为数字信号并送入单片机，单片机根据收到的信号实时显示在 OLED 显示屏上。

4.2 主程序

系统上电之后，经过一系列的初始化，比如串口初始化，定时器初始化，初始化完成后延时一段时间，让外围器件上电稳定，再初始化外围器件，如ＤＳ１３０２时钟芯片，指纹识别模块，然后进入主程序。由于单片机是单线程运行的，所以在主程序中采用一个 while 循环来执行整个的主程序。

```
void main()
{
    init_mcu();                 //初始化单片机
    delay(200);                 //延时
    init_peripheral();          //初始化外设
    while(1)
    {
        key_service();          //按键服务函数
        OLED_Display();         //OLED 界面显示函数
        time_update();          //时间更新函数
    }
}
```

4.3 指纹传感器驱动程序

本系统的重要部分是指纹识别模块，单片机通过串口通信,发送命令控制指纹模块调用相应的指令，通过串口接收到的信息来判断指令执行的结果。从而实现录入指

纹，清除指纹，验证指纹的操作。每个指纹模块的指纹模板都是按照序号存放。序号定义为：0、1、2、3.....（N-1）（N 为模块指纹库容量）。用户只能根据相应序号访问指纹库的相应模板内容；相应的存储和搜索功能都是针对指纹序号进行操作。下面是一些重要函数的部分代码说明。

录入指纹是通过调用两次录入指纹指令，将录到的指纹分别存于 charbuffter1 和 charbrffter2 中，再合成指纹特征文件存于相应的区域。

```
//指纹添加新用户
unsigned char FP_add_new_user(unsigned int user_ID)
{
    unsigned char ucH_user;
    unsigned char ucL_user;
    ucH_user = (char)((user_ID >> 8) & 0xff);
    ucL_user = (char)(user_ID & 0xff);

    do {
        FP_Cmd_Get_Img();                       //获得指纹图像
        FP_Recevice_Data(12);                   //接收 12 个长度的反馈码
    }while ( UART2_FP_RECEVICE_BUFFER[9]!=0x00 );   //检测是否成功的按了指纹

    FP_Cmd_Img_To_Buffer1();        //将图像转换成特征码存放在 Buffer1 中
    FP_Recevice_Data(12);                   //接收 12 个长度的反馈码

    do{
        FP_Cmd_Get_Img();                       //获得指纹图像
        FP_Recevice_Data(12);                   //接收 12 个长度的反馈码
    }while( UART2_FP_RECEVICE_BUFFER[9]!=0x00 );

    FP_Cmd_Img_To_Buffer2();        //将图像转换成特征码存放在 Buffer2 中
    FP_Recevice_Data(12);                   //接收 12 个长度的反馈码
```

```
    FP_Cmd_Reg_Model();                         //转换成特征码
    FP_Recevice_Data(12);
    FP_Cmd_Save_Finger(ucH_user,ucL_user);
    FP_Recevice_Data(12);
    return 0;
}
```

删除指纹是通过发送删除指定指纹模板的指令包，把想要删除的指纹 ID 作为参数传入函数，合并在指令包中，达到删除指定指纹模板的目的。

```
//删除指纹模块里的指定指纹模版
void FP_Cmd_Delete_Model(unsigned int uiID_temp)
{
    volatile unsigned int uiSum_temp = 0;
    unsigned char i;
    //把 16 位的 int 变量转化位 2 个 8 位的 char 并存入指令包相应的位置
    FP_Delete_Model[4]=(uiID_temp&0xFF00)>>8;
    FP_Delete_Model[5]=(uiID_temp&0x00FF);
    for(i=0;i<8;i++)                    //计算校验和
        uiSum_temp = uiSum_temp + FP_Delete_Model[i];
    //将校验和存入指令包相应的位置
    FP_Delete_Model[8]=(uiSum_temp&0xFF00)>>8;
    FP_Delete_Model[9]=uiSum_temp&0x00FF;
    for(i=0;i<6;i++)                    //发送包头
        UART2_Send_Byte(FP_Pack_Head[i]);
    for(i=0;i<10;i++)                   //发送命令删除指定指纹模版
        UART2_Send_Byte(FP_Delete_Model[i]);
    FP_Recevice_Data(12);               //接收 12 个长度的反馈码
}
```

验证指纹是录入一次指纹在缓存区 1 中，在用缓冲区 1 中的指纹特征文件搜索指定的指纹库，并返回接受包判断搜索结果。用户指纹验证,验证成功，返回指纹对应用户的

ID，验证失败，返回-1。

```
unsigned int FP_User_Identity()
{
    unsigned int user_ID;
    unsigned char user_IDL;
    unsigned char user_IDH;
    do
    {
        FP_Cmd_Get_Img();                    //获得指纹图像
        FP_Recevice_Data(12);                //接收 12 个长度的反馈码
        }while( UART2_FP_RECEVICE_BUFFER[9]!=0x00 );//检测是否成功按了指纹
        FP_Cmd_Img_To_Buffer1();             //将图像转换成特征码存放在 Buffer1 中
        FP_Recevice_Data(12);                //接收 12 个长度的反馈码
        FP_Cmd_Search_Finger();              //搜索整个用户指纹库
        FP_Recevice_Data(12);                //接收 12 个长度的反馈码
    if( UART2_FP_RECEVICE_BUFFER[9] == 0x00)     //如果找到匹配的指纹
    {
        user_IDH = UART2_FP_RECEVICE_BUFFER[10];//该指纹所在的编号高 8 位
        user_IDL = UART2_FP_RECEVICE_BUFFER[11];//该指纹所在的编号低 8 位
        user_ID = user_IDH *100000000 + user_IDL;
        return user_ID;                      //返回 16 位的 user_ID
    }else
         return -1;                          //如果找不到匹配的指纹，返回-1
}
```

4.4 OLED 显示屏驱动程序

OLED 显示屏是与单片机进行 SPI 串口通信的，只用到了 5 个 IO 口，其中真正发送数据的只有 D0 和 D 1 两个端口，本次主要调用的是显示汉字和数字的函数。下面是部分代码展示：

```
//显示汉字
void OLED_ShowCHinese(u8 x,u8 y,u8 no)
{
    u8 t,adder=0;
    OLED_Set_Pos(x,y);
    for(t=0;t<16;t++)
    {
        OLED_WR_Byte(Hzk[2*no][t],OLED_DATA);
        adder+=1;
    }
    OLED_Set_Pos(x,y+1);
    for(t=0;t<16;t++)
    {
        OLED_WR_Byte(Hzk[2*no+1][t],OLED_DATA);
        adder+=1;
    }
}
void OLED_ShowNum(u8 x,u8 y,u32 num,u8 len,u8 size2)
{
    u8 t,temp;
    u8 enshow=0;
    for(t=0;t<len;t++)
    {
        temp=(num/oled_pow(10,len-t-1))%10;
        if(enshow==0&&t<(len-1))
```

```
            {
                if(temp==0)
            {
                OLED_ShowChar(x+(size2/2)*t,y,' ');
                continue;
            }else
                enshow=1;
            OLED_ShowChar(x+(size2/2)*t,y,temp+'0');
        }
    }
```

4.5 DS1302 驱动程序

DS１３０２时钟芯片是通过 TCLK，TIO，TRST 三个端口来与单片机进行通信的，其中数据端口只有一个就是 TIO，通过查阅数据手册，掌握其通信时序，主要调用设置日历数据和读取日历数据两个函数，下面是其部分代码展示：

函数功能：设置 DS1302 时钟日历数据，说明：把时钟日历暂存数组 TimeData 数据转换为 BCD 码并写入到 DS1302 时钟日历寄存器中

```
void Set_DS1302_Time(uchar addr)
{
    uchar i, j;
    DS1302_Clear_WP();      //清除写保护
    for(i = 0; i < 7; i++)   //写入 7 个字节的时钟初始值
    {
        j = TimeData[i]/10;       //BCD 码转换
        TimeData[i] %= 10;       //BCD 码转换
        TimeData[i] += j*16; //BCD 码转换
        DS1302_W_DAT(addr, TimeData[i]); //先写 DS1302 时钟日历起始地
      址，再写数据
        addr += 2;  //时钟日历寄存器地址+2 转向下一个寄存器
}
```

DS1302_Set_WP(); //开起写保护

函数功能：读取 DS1302 时钟数据，读取 DS1302 时钟数据 返回数据存入时钟日历暂存数组 TimeData（数据格式 BCD 码）

```
void Read_DS1302_Time(uchar addr)
{
    uchar i,j;
    DS1302_Clear_WP();          //清楚些保护
    for(i = 0; i < 7; i++) //从 DS1302 读取 7 个字节的时钟日历数据
    {
        TimeData[i] = DS1302_R_DAT(addr);//写入要读取数据的寄存器起始地址，再读出数据存入 TimeData 数组
        addr += 2;                          //时钟日历寄存器地址+2 转向下一个寄存器
        j = TimeData[i]/16;   //BCD 码转换
        TimeData[i] %= 16;        //BCD 码转换
        TimeData[i] += j*10; //BCD 码转换
    }
    DS1302_Set_WP();     //开起写保护
}
```

5. 焊接和调试

5.1 指纹考勤器的焊接

焊接时，根据原理图和各个电路模块之间的关系，先在洞洞板布线软件上进行模拟布线。布线规则按照如下规则：

- 电源引出线上正下地；
- 下载 TXD 和 RXD 靠近串口 1，指纹模块靠经串口 2；
- 以单片机为核心，上部分时人机交互输出部分，包括 OLED 显示屏，LED 指示电路和蜂鸣器；
- 下部分是人机交互输入部分，包括独立键盘和指纹识别模块。

具体的布线如图所示：

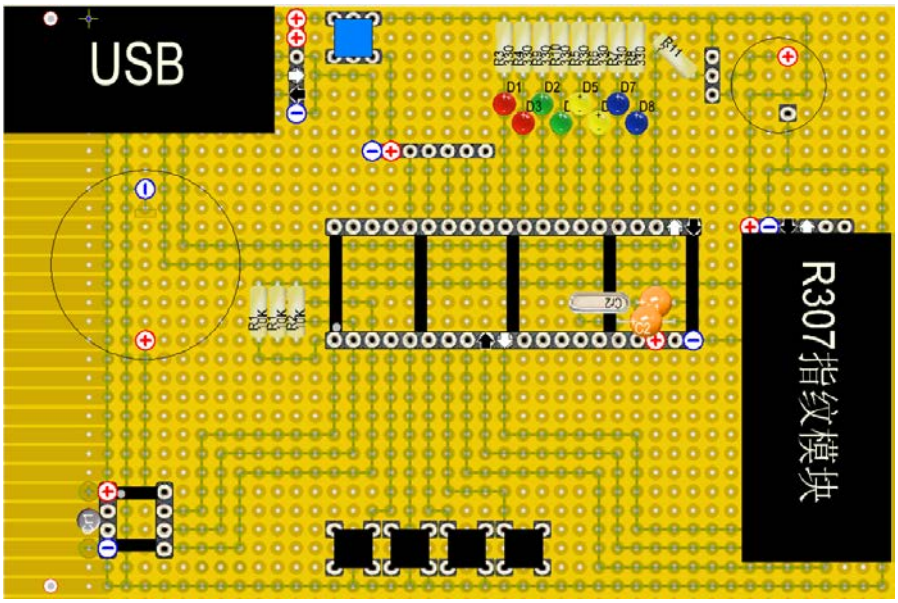


图 5.1-1 洞洞板布线图正面

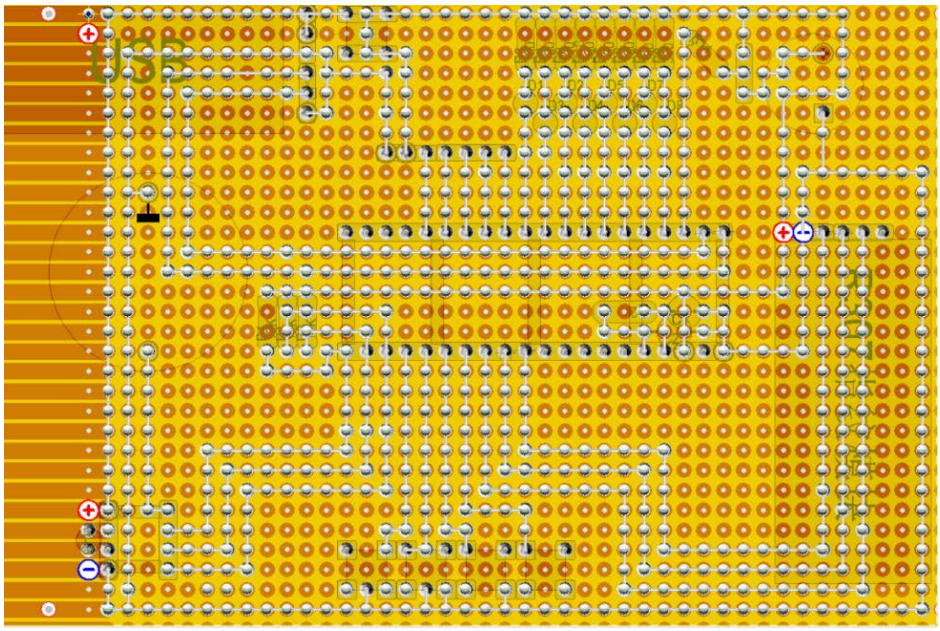


图 5.1-2　洞洞板布线图背面

5.2　指纹考勤器的调试

板子焊接好后，就要对单片机烧写程序，开始验证各个软件和硬件的部分，而本次中事实上不是一下就成功的，也遇了不少的问题，下面就对在调试过程中遇到的问题阐述说明：

- 下载不了程序

首先接上电源，用万用表量电压正常，接上 USB 转 TTL 模块，开始下载程序，发现一直没反应，电脑一直提示 USB 器件工作不正常,识别不了,检查 USB 驱动,发现驱动已安装.用万用表检查各条线路的情况,发现自锁开关焊坏了,更换后正常下载.

- 电源指示灯在未接通电源时微亮

为了方便查看电源接通情况,我在电源处接上一个 LED 灯作为指示灯．结果却发现指示灯在未接通电源时微亮．怀疑是其他接口供电．翻阅 s t c 数据手册,发现 RXD 与 TXD 需分别加上一个二极管和电阻,防止串口向单片机供电,加上后正常显示；

- 指纹识别模块不工作

单片机和指纹模块之间的通信方式采用的是串口方式，成功烧写程序之后，单片机经过串口初始化之后，为了方便测试两者之间的通信是否正常，单片机上电之后会直接通过串口发送指令给指纹模块，实现识别功能，当指纹模块接受到该指令时，指纹采集头会亮，但是指纹头却一直没有反应，确认了指纹模块的接线正确之后，指纹头还是没

有反应。将指纹模块拔掉，用 USB 转 TTL 工具来调试单片机的串口数据，接入电脑后，打开调试助手，在串口参数正确的情况下，发现单片机没有发送数据，波特率等其他参数都是正确的，同时也进行了共地，确定串口 2 不工作，去掉外部晶振,使用单片机内部晶振,串口 2 工作正常,指纹开始正常工作.

- 指纹识别模块验证指纹失败

在调用指纹验证函数时,老是验证失败,检查函数定义和指纹驱动程序头文件，发现搜索指纹库指令包检验和出错,更改正确后正常识别.

- 调用删除指纹指令陷入死循环

在调用删除指纹函数后,单片机陷入死循环,屏幕一直停留在删除界面,检查函数定义和指纹驱动程序头文件,发现单片机发送删除指定指纹模板指令后,没有接收 12 个长度的反馈码,导致通信异常,增加接收函数后指令调用正常。

结论和心得体会

时间不等人，一眨眼的功夫，3 个星期就过去了，看着桌面放着的课程设计，心里还是挺欣慰的，毕竟是本小组辛辛苦苦的劳动成果，也是对单片机课程的一个交代，心里还是比较高兴的。

在设计的过程中或多或少肯定遇到了不少的难题，设计到硬件部分还有软件部分，编写程序，画电路图，动手焊接电路板，安装调试。遇到的问题很多很多，小到函数中的一个小 BUG ,硬是找了很久才找出来,大到不小心烧坏模块,如 OLED 显示屏和指纹识别模块,烧的都是大家伙,无时无刻不在挑战我们的耐心和预算。

在这过程中，学到的东西还是很多的，毕竟平时学到的都是一些最基本的理论知识，实际动手做实验还是比较少的，最后在小组成员的共同努力下还是很惊喜的完成了。在这次的设计过程中，从最初对单片机的各种理论知识的简单了解，C 语言程序的看不懂，焊接电路的无法上手，各种各样的问题克服，体会到真正动手去实际的意义，以前总是死记硬背的去记各种知识，但是在实际应用中还是比较少的，在实际中去找问题，解决问题，更为实际。而在本次课设中，也让我懂得了一个道理，凡事都要有耐心，要善于运用学到的知识来验证问题，有问题肯定是正常的，但是在遇到问题时如何去克服，如何去解决那才是关键所在，要用尽自己所知去想，找资料，问同学，问老师，不懂就问，才有可能去解决问题。

参考文献

[1].李朝青.单片机原理及接口技术（第 3 版）.北京航天航空大学出版社，2005 年 10 月

[2].求是科技.单片机典型外围器件及应用实例.北京：人民邮电出版社，2006 年 2 月

[3].谭浩强.C 语言程序设计（第二版）。北京：清华大学出版社，1999 年 12 月

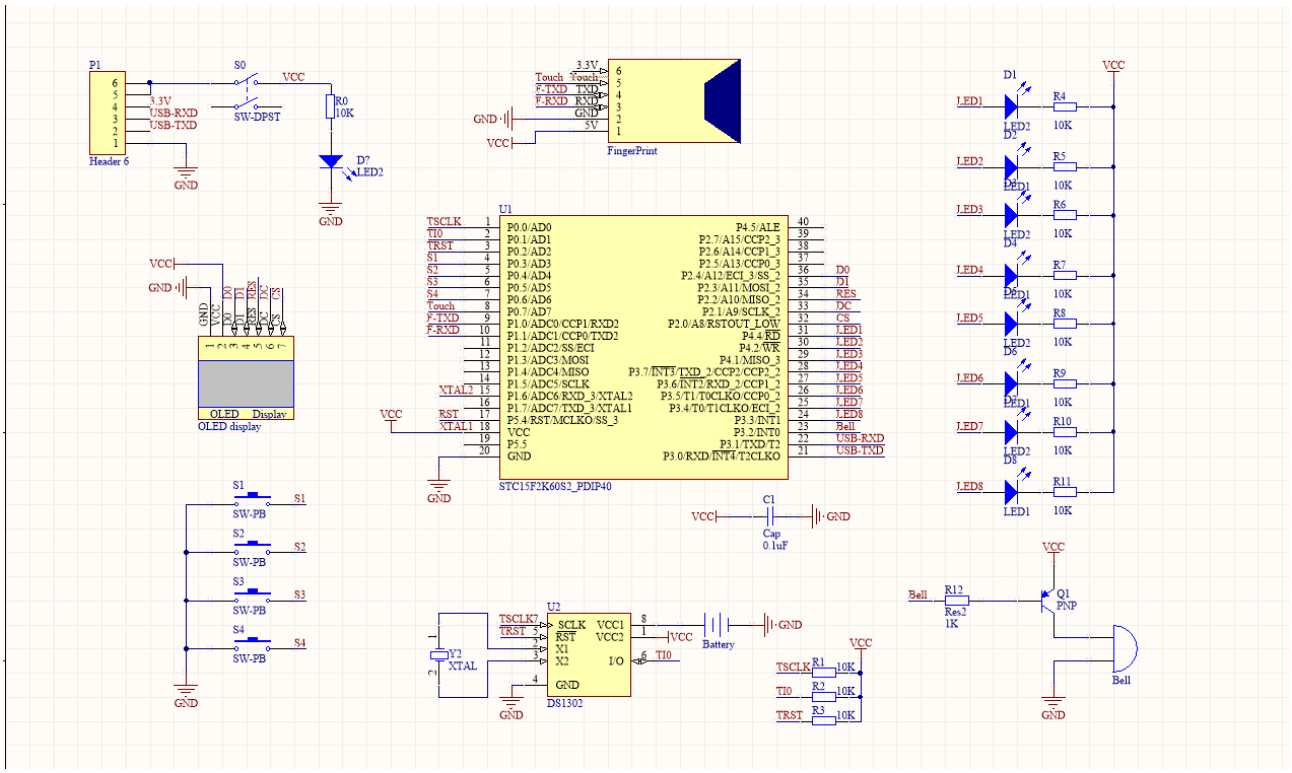
[4].吴坚鸿.手把手教你单片机程序框架（连载）,中国电子技术论坛,2015 年 11 月

[5].杭州城章科技有限公司.R307 指纹识别手册，杭州城章科技有限公司，2015 年 7 月

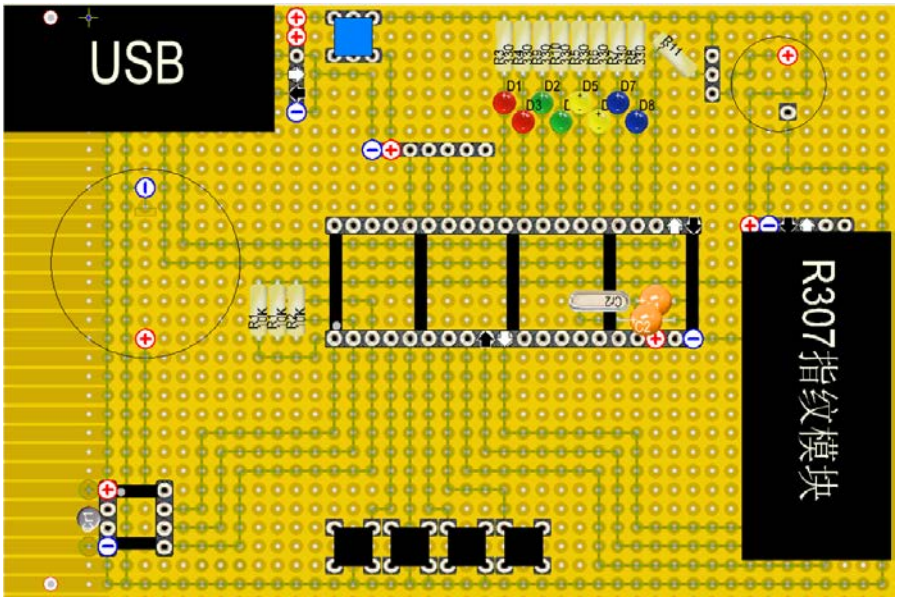
[6].STC 单片机.stc15 系列单片机数据手册，stc 宏晶科技有限公司，2015 年 6 月

[7]. 中景园电子,0.96 寸 OLED 显示屏使用手册,中景园电子，2015 年 5 月

附录 A

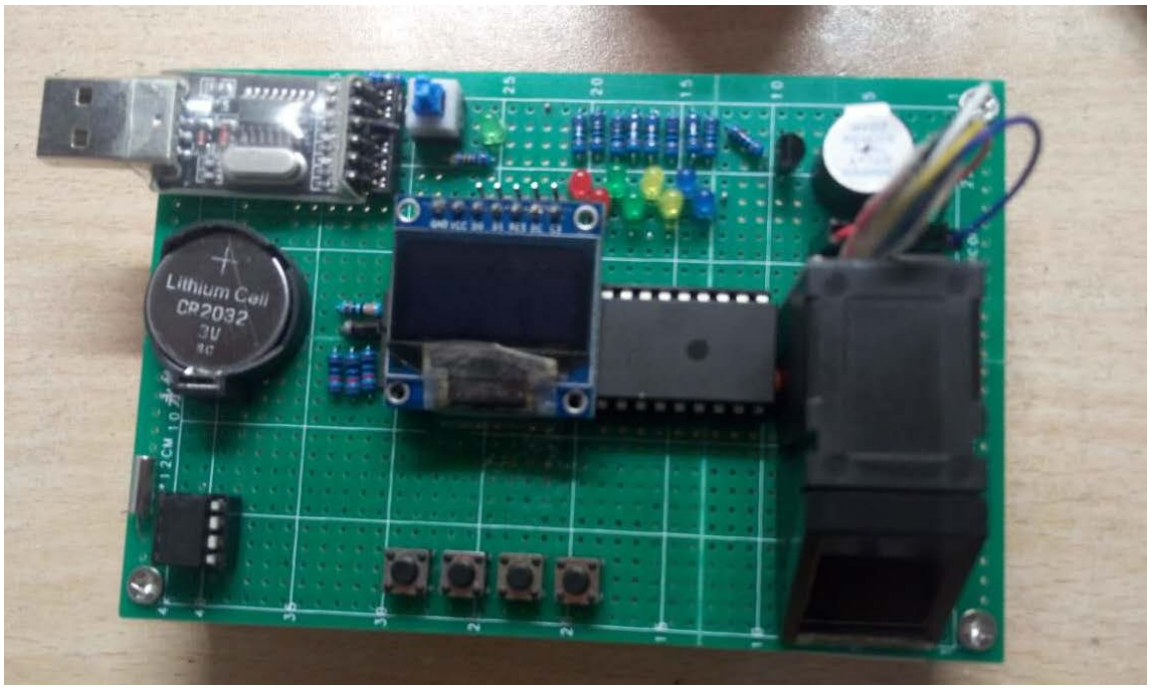


附录 A　电路原理图

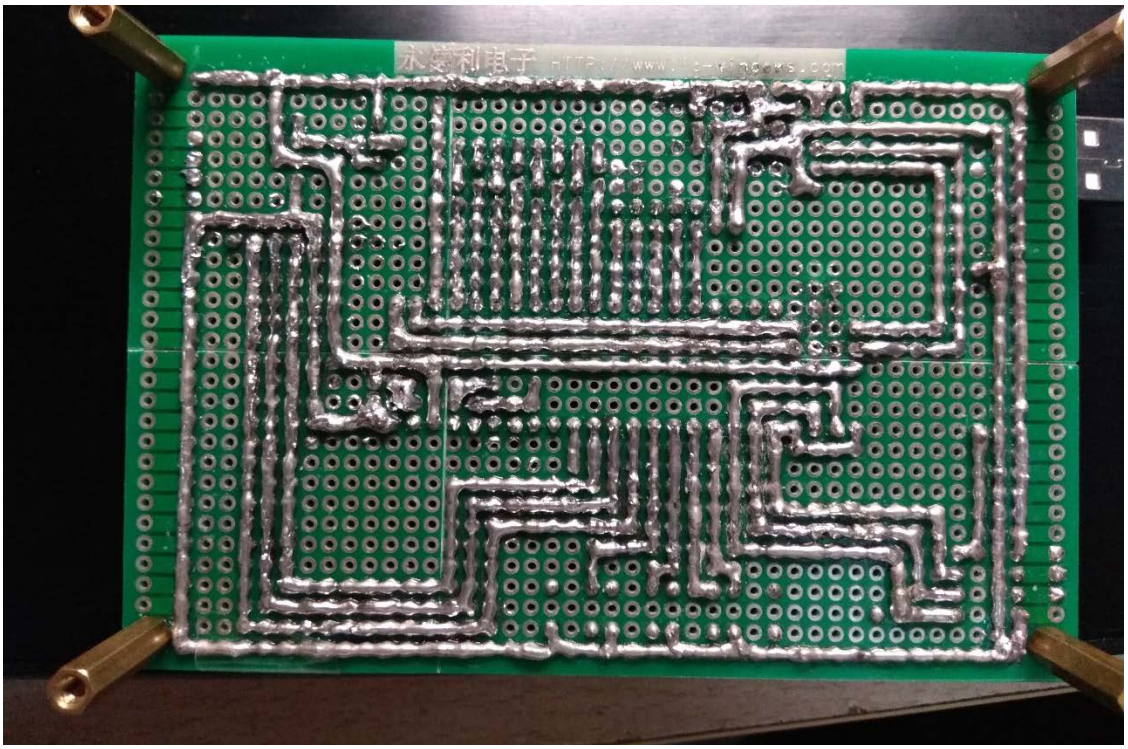


附录 A　洞洞板布线图

附录 B



附录 B　作品实物图



附录 B　实物图背面

附录 C

部分程序源码

```
#include <stc15.h>              //stc15 系列头文件
#include "ds1302.h"             //ds1302 头文件
#include "oled.h"               //oledt 头文件
#include "uart.h"               //串口头文件
#include "fingerprint.h"        //指纹模块头文件

#define FOSC 11059200L

#define uchar unsigned char
#define uint unsigned int

#define const_voice_short  40      //蜂鸣器短响
#define const_led_long   120   //led 灯常亮
#define const_key_time  40     //独立键盘消抖时间

sbit memu = P0^3;               //菜单键
sbit ret  = P0^6;               //返回键
sbit left   = P0^4;             //左翻键
sbit right = P0^5;              //右翻键

sbit touch = P0^7;              //指纹模块触摸感应 IO 口
sbit beep = P3^2;               //蜂鸣器接口
sbit red = P4^4;                //红色 LED 控制接口
sbit green = P4^1;              //绿色 LED 控制接口

uchar ucKeySec = 0;             //一些需要在不同函数之间使用的核心变量，
用全局变量
uchar uiWrongCnt = 0;           //错误计数器
```

```
uchar uiPassCnt = 0;        //通过计数器
uchar WindowSec = 1;        //窗口变量
uchar submemu_num  = 1;         //菜单编号变量
uchar wd_update=1;          //窗口整屏更新显示标志，本节最核心的变量


bit part_1_2_update = 1;        //第一屏第二行更新标志位
bit part_1_3_update = 1;        //第一屏第三行更新标志位
bit part_2_2_update = 1;        //第二屏第二行更新标志位
bit part_3_2_update = 1;        //第三屏第二行更新标志位
bit part_4_2_update = 1;        //第四屏第二行更新标志位
bit add_user = 0;               //用户添加标志位
bit add_admin = 0;              //管理员添加标志位
bit identity_admin = 1;         //管理员验证标志位
bit identity_user  = 0;         //用户验证标志位
bit delte_model    = 0;         //删除指纹模板标志位
bit delte_allmodel = 0;         //清空指纹模板标志位


uint t = 0;                     //时间更新计算器
int user_num = 1;               //用户编号变量
uint  uiVoiceCnt = 0;           //一些需要在不同函数之间使用的核心变量，只能用全局变量
fp xdata user_Ifo[301];         //用户信息结构数据
//初始化时间日期暂存数组，秒、分、时、日、月、周、年 初值为= 17 年 6 月 19 日 周 1 12:58:50
uchar TimeData[7] ={50, 13, 23, 30, 6, 5, 17};

void init_mcu();            //单片机初始化
void delay(uint xms);           //延时函数
```

```
void init_peripheral();         //外围器件初始化

/*void MCU_Recevice_Date(unsigned char ucLength);
void MCU_Synchronize_Time();
void MCU_Communicate_PC();*/

void T0_time();                 //定时器 T0 中断函数
void key_scan();              //键盘扫描程序
void key_service();             //键盘服务子程序
void time_update();             //时间更新函数
void OLED_Display();            //OLED 显示函数
void main_screen();             //主界面
void memu_screen();             //菜单界面
void ifo_screen();              //信息界面
void sub_screen();              //子菜单界面

void main()
{
   init_mcu();                      //初始化单片机
   delay(200);                          //延时
   init_peripheral();                   //初始化外设
   while(1)
   {
      key_service();                    //按键服务函数
      OLED_Display();                   //OLED 界面显示函数
      time_update();                    //时间更新函数
   }
}

void init_mcu()
```

```
{
    beep=1;                    //禁止蜂鸣器响
    TMOD=0x01;                 //设置定时器 T0 为方式一
    TH0=0xf8;                  //装初值
    TL0=0x2f;
    UART2_init_57600();          //串口 2 初始化
}

void delay(uint xms)
{
    uint i,j;
    for(i=0;i<xms+3;i++)
    {
        for(j=0;j<102;j++);
    }
}

void init_peripheral()
{
    EA=1;                              //开总中断
    ET0=1;                             //开定时器 0 中断
    TR0=1;                                 //启动定时器 0
    OLED_Init();                       //初始化 OLED
    OLED_Clear();                          //清屏
    //Set_DS1302_Time(DS1302_W_ADDR);  //先写入时钟日历寄存器起始地址再
设置时钟日历初值
    Read_DS1302_Time(DS1302_R_ADDR);   //先写入时钟日历寄存器起始地址再
读出时钟日历写入到 TimeData 数组中
    init_User_Ifo();                   //初始化用户信息结构数组
    Set_User_Ifo();                        //读取指纹模块 flash 数据放在用户
```

信息结构数组

```
        add_admin = !(user_Ifo[0].isVaild); //判断需不需添加管理员
    }
    //用户信息结构数组初始化
    void init_User_Ifo(void)
    {
        unsigned int i;
        for(i=0;i<301;i++)
        {
            user_Ifo[i].ID = 0;              //用户 ID 清零
            user_Ifo[i].isVaild = 0;     //用户有效位清零
            user_Ifo[i].sec = 0;         //用户时间清零
            user_Ifo[i].min = 0;
            user_Ifo[i].hour = 0;
        }
    }
    //用户结构数组获取指纹模块数据
    void Set_User_Ifo(void)
    {
        unsigned int i;
        unsigned char* pTable;
        UART2_init_57600();
        pTable = FP_Get_ReadIndexTable(0);//获取指纹模块有效指纹模板信息
        for(i = 0;i<256;i++)
        {
            user_Ifo[i].ID = i;
            user_Ifo[i].isVaild = (*(pTable + (i/8)) & (bit_num [(i%8)]));
        }
        UART2_init_57600();
        pTable = FP_Get_ReadIndexTable(1);
```

```
        for(i=256;i<301;i++)
        {
            user_Ifo[i].ID = i;
            user_Ifo[i].isVaild = (*(pTable + ((i/8)-32)) & (bit_num
[(i%8)]));
        }
    }
    void key_scan()
    {
        //带 static 的局部变量
        static uint  uiKeyTimeCnt1=0;
        static uchar ucKeyLock1=0;
        static uint  uiKeyTimeCnt2=0;
        static uchar ucKeyLock2=0;
        static uchar uiKeyTimeCnt3=0;
        static uchar ucKeyLock3=0;
        static uchar uiKeyTimeCnt4=0;
        static uchar ucKeyLock4=0;


        if(memu == 1)//IO 是高电平，说明按键没有被按下，这时要及时清零一些标
志位
        {
            ucKeyLock1 = 0; //按键自锁标志清零
             uiKeyTimeCnt1 = 0;//按键去抖动延时计数器清零

        }
        else if(ucKeyLock1 == 0)//有按键按下，且是第一次被按下
        {
               uiKeyTimeCnt1 ++; //累加定时中断次数
```

```
    if(uiKeyTimeCnt1 > const_key_time)
    {
        uiKeyTimeCnt1 = 0;
        ucKeyLock1 = 1;  //自锁按键置位,避免一直触发
        ucKeySec = 1;    //触发 1 号键
    }
}

if(ret == 1)
{
    ucKeyLock2 = 0;
     uiKeyTimeCnt2 = 0;
}
else if(ucKeyLock2 == 0)
{
    uiKeyTimeCnt2 ++;
    if(uiKeyTimeCnt2 > const_key_time)
    {
        uiKeyTimeCnt2 = 0;
        ucKeyLock2 = 1;
        ucKeySec = 2;
    }
}

if(left == 1)//IO 是高电平，说明按键没有被按下，这时要及时清零一些标志位
{
    ucKeyLock3 = 0; //按键自锁标志清零
     uiKeyTimeCnt3 = 0;//按键去抖动延时计数器清零，
}
```

```
        else if(ucKeyLock3 == 0)//有按键按下，且是第一次被按下
        {
               uiKeyTimeCnt3 ++; //累加定时中断次数
            if(uiKeyTimeCnt3 > const_key_time)
            {
                 uiKeyTimeCnt3 = 0;
                 ucKeyLock3 = 1;  //自锁按键置位,避免一直触发
                 ucKeySec = 3;    //触发 3 号键
            }
        }

            if(right == 1)//IO 是高电平，说明按键没有被按下，这时要及时清零一些标志位
        {
            ucKeyLock4 = 0; //按键自锁标志清零
              uiKeyTimeCnt4 = 0;//按键去抖动延时计数器清零，此行非常巧妙，是我实战中摸索出来的。
        }
        else if(ucKeyLock4 == 0)//有按键按下，且是第一次被按下
        {
               uiKeyTimeCnt4 ++; //累加定时中断次数
            if(uiKeyTimeCnt4 > const_key_time)
            {
                 uiKeyTimeCnt4 = 0;
                 ucKeyLock4 = 1;  //自锁按键置位,避免一直触发
                 ucKeySec = 4;    //触发 4 号键
            }
        }
    }
```

```
void key_service()
{
   switch(ucKeySec)
   {
      //当 memu 键被按下
       case 1:
            switch(WindowSec)
            {
                //在主菜单窗口下
                case 1:
                    WindowSec = 2;        //进入菜单窗口
                    wd_update = 1;        //切换窗口需要整屏更新
                    part_2_2_update = 1; //菜单窗口第二行更新
                    break;
                //在菜单窗口下
                case 2:
                    //前 3 个菜单的进入需要管理员认证
                    if(identity_admin && submemu_num < 4)
                    {
                        WindowSec = 5;       //弹出对话框
                        wd_update = 1;           //切换窗口
                    }
                    else
                    {
                        WindowSec = 3;
                        wd_update = 1;
                        part_3_2_update = 1;
                    }
                    if(submemu_num == 4)
                    {
```

```
                WindowSec = 5;
                wd_update = 1;
                identity_user = 1;
            }
            break;
        //在子菜单窗口下
        case 3:
            switch(submemu_num)
            {
                case 1:
                    WindowSec = 5;  //弹出对话框
                    wd_update = 1;  //切换窗口
                    add_user = 1;   //添加用户标志位置 1
                    break;
                case 2:
                    WindowSec = 5;//弹出对话框
                    wd_update = 1;//切换窗口
                    delte_model = 1;  //删除指纹标志位置 1
                    break;
                case 3:
                    WindowSec = 5;//
                    wd_update = 1;
                    delte_allmodel = 1;
                    break;
                case 4:
                    WindowSec = 5;
                    wd_update = 1;
                    identity_user = 1;
                    break;
            }
```

```
                    break;
                //在信息提示窗口下
            case 4:
                    WindowSec = 1;  //返回主界面窗口
                    wd_update = 1;
                    break;
            case 5:
                    WindowSec = 2;
                    wd_update = 1;
                    part_2_2_update = 1;
                    break;
        }
         uiVoiceCnt=const_voice_short;
         ucKeySec=0;
         break;
//当 ret 键被按下
case 2:
        switch(WindowSec)
        {
         //在主窗口下
         case 1:
                WindowSec = 4;  //进入信息窗口
                wd_update = 1;
                part_4_2_update = 1;
                break;
         //在菜单窗口下
         case 2:
                 WindowSec = 1;  //返回主菜单窗口
                 wd_update = 1;
                 submemu_num = 1;
```

```
                break;
            //在子菜单窗口下
            case 3:
                WindowSec = 2;  //返回菜单窗口
                wd_update = 1;
                user_num = 1;
                identity_admin = 1;
                identity_user = 0;
                part_2_2_update = 1;
                break;
            //在对话框窗口下
            case 5:
                WindowSec = 2; //返回菜单窗口
                wd_update = 1;
                user_num = 1;
                identity_user = 0;
                part_2_2_update = 1;
                break;
        }
        uiVoiceCnt=const_voice_short; //蜂鸣器短响
        ucKeySec=0;                   //按键编号清零，只响应一次
        break;
//当 left 键被按下时
case 3:
        switch(WindowSec)
        {
            case 1:add_admin = 1;
                   break;
            //在菜单窗口下
            case 2:
```

```
                submemu_num --;         //菜单编号自减
                if(submemu_num == 0)  //菜单编号减到0时自动跳回
最后一个菜单
                {
                     submemu_num = 5;
                }
                part_2_2_update = 1;  //菜单窗口第二行更新
                break;
            //在子菜单窗口下
            case 3:
                switch(submemu_num)
                {
                    case 1:
                    case 2:
                          user_num --;
                          if(user_num == 0)
                          {
                               user_num = 300;
                          }
                          part_3_2_update = 1;
                          break;
                    case 5:
                          user_num -= 2;
                          if(user_num == -1)
                          {
                               user_num = 299;
                          }
                          part_3_2_update = 1;
                          break;
                }
```

```
                break;
        }
         uiVoiceCnt=const_voice_short;
         ucKeySec=0;
        break;
//当 right 键被按下时
case 4:
        switch(WindowSec)
        {
             //在菜单窗口下
             case 2:
                    submemu_num ++;
                    if(submemu_num == 6)
                    {
                        submemu_num = 1;
                    }
                    part_2_2_update = 1;
                    break;
              case 3:
                     switch(submemu_num)
                    {
                       case 1:
                       case 2:
                              user_num ++;
                              if (user_num == 301)
                              {
                                   user_num = 1;
                              }
                              part_3_2_update = 1;
                              break;
```

```
                        case 5:
                                user_num += 2;
                                if (user_num == 301)
                                {
                                     user_num = 1;
                                }
                                part_3_2_update = 1;
                                break;
                        }
                        break;
                }
                 uiVoiceCnt=const_voice_short;
                ucKeySec=0;
                break;
        }
}
void OLED_Display()
{
   switch(WindowSec)
   {
       case 1:main_screen();break;          //显示主界面
       case 2:memu_screen();break;        //显示菜单界面
       case 3:sub_screen();break;             //显示子菜单界面
       case 4:ifo_screen();break;             //信息界面
       case 5:dialog_screen();break;          //对话框界面
   }
}
```