2018 年广东省大学生电子设计竞赛

设计报告

一种基于蓝牙和语音控制的智能家居系统

A smart home system based on Bluetooth and voice control

摘要：本项目使用 STM32F407 作为主控芯片，通过蓝牙和语音控制实现安全、便宜的智能家居系统。实现了手机实现家电控制；语音识别+人脸识别开门/关门；语音控制开关灯、开关风扇以及切歌，关闭音乐等；地震、CO、温度、湿度、光强监测等；用户可通过 LCD 显示屏操作 APP 实现人脸管理、人脸登记、语音留言、文本留言、查看语音、查看文本留言、查看门外情况(监控)，以及控制开门/关门、开灯/关灯、开风扇/关风扇等等。相比于现有的智能家居方案，我们的方案通过使用 Android ID 作为手机用户唯一识别码以及语音识别+人脸识别双重开门检测，使得整个系统更为安全。同时，我们的方案成本更低，可用于高校学生宿舍推广。

关键词：智能家居；STM32；蓝牙；语音识别；人脸识别

ABSTRACT This project uses STM32F407 as the main control chip to realize a safe and cheap smart home system through Bluetooth and voice control. Achieve mobile phone to achieve home appliance control; voice recognition + face recognition to open / close the door; voice control switch lights, switch fans and cut songs, turn off music, etc.; earthquake, CO, temperature, humidity, light intensity monitoring, etc.; users can pass LCD Display operation APP implements face management, face registration, voice message, text message, view voice, view text message, view door condition (monitoring), and control door opening/closing, turning on/off, turning on/off Fans and so on. Compared with the existing smart home solution, our solution makes the whole system more secure by using the Android ID as the unique identification code of the mobile phone user and the dual recognition detection of voice recognition + face recognition. At the same time, our program costs are lower and can be used to promote college students' dormitory.

INDEX TERMS Smart home; STM32; Bluetooth; Speech recognition; Face recognition

原 创 性 声 明

本团队郑重声明：所呈交的作品及论文，是本团队在指导老师的指导下，独立进行研究所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律责任由本人承担。

作者签名：_____ 日 期：_____

目　录

查新说明

在大雅相似度分析官网上查新结果：

http://dsa.dayainfo.com/

标题	相似度	检测时间	检测状态	付费状态	操作
一种基于蓝牙和语音控制的智能家居系统	22.75%	2018-08-20	完成		

第 1 章 引言

1.1 选题背景和研究意义

智能家居的概念起源很早，但一直未有具体的建筑案例出现，直到 1984 年美国联合科技公司（United Technologies Building System）将建筑设备信息化、整合化概念应用于美国康涅狄格州（Connecticut)哈特佛市（Hartford）的 CityPlaceBuilding 时，才出现了首栋的智能型建筑[1]，从此揭开了全世界争相建造智能家居派的序幕。美国、加拿大、欧洲、澳大利亚和东南亚等经济比较发达的国家先后提出了各种智能家居的方案。

智能家居系统是人们理想中的一种居住环境，其以住宅为平台安装有智能家居系统，实现家庭生活更加安全，节能，智能，便利和舒适。智能家居不仅能给用户提供安全、健康和舒适的生活环境，而且用户能够远程监控自己的家居状态[2]和控制家庭电器设备。出门在外，我们可以通过电话、电脑来远程遥控家居各智能系统，例如在回家的路上提前打开家中的空调；到家开门时，借助人脸和语音识别技术[3]，可实现不用钥匙开门，安防撤防，开启家中的照明灯具和窗户迎接我们的归来；回到家里，可以使用手机或者遥控器方便地控制房间内各种电器设备，可以通过智能化照明系统选择预设的灯光场景，可以用语言控制音响电视播放歌曲……这一切，主人都可以安坐在沙发上从容操作，一个手机或者直接用语音可以遥控家里的一切，比如开风扇，给植物浇水，调整窗帘、灯光的状态；在公司上班时，家里的情况还可以显示在办公室的电脑或手机上，随时查看；门口机具有语音留言和拍照等功能，家里没人而有客人来的时候，系统会记录相关信息供我们回来查询。这一切都是家居智能化给我们带来的福利，摆脱了传统的机械的家居控制，节省了人力，使居家更加的方便，增强了家庭中各种安全问题的监控，提高了住户的生活质量和幸福感[4]。

1.2 智能家居研究现状

智能家居最初的发展主要以灯光遥控控制、电器远程控制和电动窗帘控制为主，随着行业的发展，智能控制的功能越来越多，控制的对象不断扩展，控制的联动场景要求更高，其不断延伸到家庭安防报警[5]、背景音乐、可视对讲、门禁指纹控制等领域。

智能家居领域由于其多样性和个性化的特点，也导致了技术路线和标准众多，没有统一通行技术标准体系的现状，从技术应用角度来看主要有三类主流技术：第一类是总线技术类；总线技术的主要特点是所有设备通信与控制都集中在一条总线上，是一种全分布式智能控制网络技术，其产品模块具有双向通信能力，以及互操作性和互换性，其控制部件都可以编程。典型的总线技术采用双绞线总线结构[6]，各网络节点可以从总线上获得供电，亦通过同一总线实现节点间无极性、无拓扑逻辑限制的互连和通信。第二类是无线通信技术类；无线通信技术众多，已经成功应用在智能家居领域的无线通信技术方案主要包括：射频（RF）技术[7]（频带大多为 315 和 433.92MHz）、VESP 协议、IrDA 红外线技术、HomeRF 协议[8]、Zigbee 标准[9]、Z-Wave 标准、Z-world 标准、X2D 技术等。无线技术方案的主要优势在于无需重新布线，安装方便灵活，而且根据需求可以随时扩展或改装，可以适用于新装修用户和已装用户。第三类是电力

线载波通信技术；电力线载波通信技术充分利用现有的电网，两端加以调制解调器，直接以 50Hz 交流电为载波，再以数百 KHz 的脉冲为调制信号，进行信号的传输与控制。

目前智能家居也面临一个大数据时代所不可回避的问题就是安全。目前对于物联网设备，可以采取的安全措施大概可以分为四类。一是前端，即用户的设备端，通过增强设备本身的可靠性来降低风险，比如提高生物识别模块的准确度；二是网络，即防范数据传输过程中可能出现的安全风险；三是系统和数据库，即防范操作系统、通用应用平台系统方面存在的风险以及威胁到信息数据库的安全风险；四是应用、管理、终端，即防范实现业务应用自身及应用交互过程中的安全风险，防范网络和系统管理不善产生的风险，防范控制终端及其操作系统面临的风险[10]。不过放眼整个智能家居行业，很少有企业能完全做到上述措施，因为涉及的领域极其广泛。另一方面，从市场的角度来说，智能家居的发展不仅受限于技术的发展，也受限于用户的体验。对于市场来说，智能家居系统作为一款产品[11]，用户关心的是他的用户体验以及价格方面的问题，但就目前而言，智能家居系统受到价格的影响，他的使用范围还仅仅局限在那些高档住宅小区，对于那些相对老式的住房，智能家居的寥寥无几。

1.3 论文研究内容和结构安排

本论文主要是采用 STM32 作为主单片机，使用蓝牙通信和语音识别实现智能控制，使用摄像头、蜂鸣器、温湿度传感器、光敏传感器、LCD 显示屏、继电器、马达等硬件模拟整个智能家居系统。

论文的内容安排如下:

第 1 章：引言。本章包括选题背景和研究意义，概述智能家居系统的发展现状。

第 2 章：智能家居实现的理论分析。本章主要智能家居系统的蓝牙通信技术、语音识别技术、人脸识别技术。

第 3 章：智能家居系统设计。本章对比分析了 STM32 单片机和 51 单片机、蓝牙通信和 WIFI 通信，并展开硬件设计和软件设计，最终搭建整个智能家居系统。

第 4 章：智能家居系统实现。本章主要对系统方案进行实现并测试相关功能，验证设计的合理性与有效性，并进行整体测试。

第 5 章：结论与展望。本章总结了本论文所做的工作，并指出可改进的地方，深入认识基于蓝牙和语音控制的智能家居系统的优势与技术难点，预测智能家居的发展走向。

附录给出了智能家居系统的程序清单、使用说明、Android 源码、STM32 源码。

第 2 章 智能家居实现的理论分析

2.1 蓝牙通信技术

蓝牙(Bluetooth)通信技术，实际上是一种短距离无线电技术，利用“蓝牙”技术，能够有效地简化笔记本电脑和智能手机、单片机等终端设备之间的通信，从而使这些设备之间的数据传输变得更加迅速高效，为无线通信拓宽道路，其实际应用范围可以拓展到各种家电产品、消费电子产品和汽车等信息家电，组成一个巨大的无线通信网络。“蓝牙”技术属于一种短距离、低成本的无线连接技术，是一种能够实现语音和数据无线传输的开放性方案。蓝牙通信的传输速率最高为 1Mb/s，以时分方式进行全双工通信，通信距离为 10 米左右，配置功率放大器可以使通信距离进一步增加。

HC-05 是一款高性能的主从一体蓝牙串口模块，该蓝牙模块的特点如下：

（1）采用 CSR 主流蓝牙芯片，蓝牙 V2.0 协议标准；

（2）输入电压:3.6V--6V,禁止超过 7V；

（3）波特率为 1200，2400，4800，9600，19200，38400，57600，115200 用户可设置；

（4）带连接状态指示灯,LED 快闪表示没有蓝牙连接;LED 慢闪表示进入 AT 命令模式；

（5）板载 3.3V 稳压芯片,输入电压直流 3.6V-6V;未配对时,电流约 30mA(因 LED 灯闪烁,电流处于变化状态)；配对成功后,电流大约 10mA。

（6）用于 GPS 导航系统，水电煤气抄表系统，工业现场采控系统；

（7）可以与蓝牙笔记本电脑、电脑加蓝牙适配器等设备进行无缝连接。

（8）HC-05 嵌入式蓝牙串口通讯模块（以下简称模块）具有两种工作模式：命令响应工作模式和自动连接工作模式，在自动连接工作模式下模块又可分为主（Master） 、从（Slave）和回环（Loopback）三种工作角色。

2.2 语音识别技术

语音识别部分的原理图如以下框图所示：

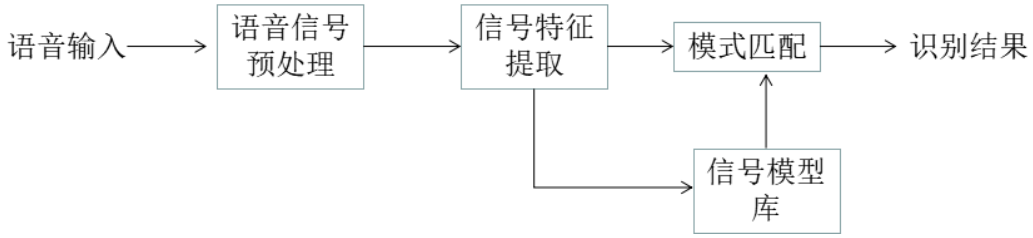


图 1 语音识别原理

采集到语音信号后，我们对语音信号进行预处理，即将采集到的连续的语音模拟信号进行数字化，最常见的方法为脉冲编码调制。脉冲编码调制主要分为采样、量化、编码三个过程。

首先，根据采样定理，采用过采样方法对语音信号进行采样，将模拟电信号转换成二进制码。之后进行量化操作，量化是按四舍五入的原则把幅值连续的样本值变化成由一定间距数(量化级差!)表示的有限样本值。经过采样和量化后，连续的语音模拟信号就变成了离散数字信号。而编码是指将量化好的采样值表示成二进制码的过程。

将语音信号转换为数字信号后，我们采用线性预测编码等技术对语音信号进行分析，用过去的若干个语音样本值的线性组合来预测以后的样值，从而提取去信号的特征参数。将提取出来的特征参数与信号模型库中的模板进行模式匹配，从而得到识别结果。

2.3 人脸识别技术

特征脸方法是从主成分分析(PCA)导出的一种人脸识别和描述技术。它将包含人脸的图像区域看作一随机向量，采用 K-L 变换得到正交 K-L 基，对应其中较大特征值的基具有与人脸相似的形状，因此又被称为特征脸。利用这些基的线性组合可以描述、表达和逼近人脸图像，所以可进行人脸识别与合成。识别过程就是将人脸图像映射到由特征脸组成的子空间上，并比较其在特征脸空间中的位置，然后利用对图像的这种投影间的某种度量来确定图像间的相似度，最常见的就是选择各种距离函数来进行度量分类实现人脸识别。

主成分分析原理：

对于一个样本资料，观测 p 个变量 $x_1, x_2, \cdots x_p$ ， n 个样品的数据资料阵为：

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} = \left(x_1, x_2, \mathrm{L}\ x_p \right)$$

其中： $x_j = \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{nj} \end{pmatrix}, \quad j = 1,2,\cdots p$

主成分分析就是将 p 个观测变量综合成为 p 个新的变量（综合变量），即

$$\begin{cases} F_1 = a_{11}x_1 + a_{12}x_2 + \mathrm{L} + a_{1p}x_p \\ F_2 = a_{21}x_1 + a_{22}x_2 + \mathrm{L} + a_{2p}x_p \\ \mathrm{L} \\ F_p = a_{p1}x_1 + a_{p2}x_2 + \mathrm{L} + a_{pp}x_p \end{cases}$$

简写为：

$$F_j = \alpha_{j1}x_1 + \alpha_{j2}x_2 + \mathrm{L} + \alpha_{jp}x_p$$

$$j = 1, 2, \mathrm{L}\ , p$$

要求模型满足以下条件：

①F_i, F_j 互不相关（$i \neq j$，$i, j = 1, 2, \mathrm{L}\ , p$）

②F_1 的方差大于 F_2 的方差大于 F_3 的方差，依次类推

③$a_{k1}^{\ 2} + a_{k2}^{\ 2} + \mathrm{L} + a_{kp}^{\ 2} = 1 \quad k = 1, 2, \mathrm{L}\ \ p$.

于是，称 F_1 为第一主成分，F_2 为第二主成分，依此类推，有第 p 个主成分。主成分又叫主分量。这里 a_{ij} 我们称为主成分系数。

上述模型可用矩阵表示为：

$F = AX$，其中

$$F = \begin{pmatrix} F_1 \\ F_2 \\ \mathrm{M} \\ F_p \end{pmatrix} \qquad X = \begin{pmatrix} x_1 \\ x_2 \\ \mathrm{M} \\ x_p \end{pmatrix}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \mathrm{L} & a_{1p} \\ a_{21} & a_{22} & \mathrm{L} & a_{2p} \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} \\ a_{p1} & a_{p2} & \mathrm{L} & a_{pp} \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ \mathrm{M} \\ a_p \end{pmatrix}$$

A 称为主成分系数矩阵。

本论文将采用基于主成分分析的方法实现人脸识别。

第 3 章 智能家居系统设计

3.1 智能家居系统整体设计

3.1.1 系统整体架构

如图 2 所示，智能家居系统包括+12 V 电源、主控芯片 STM32F407、LCD 显示屏、SD 卡、按键、TPAD 模块、蜂鸣器模块、继电器模块、蓝牙 HC-05 、语音模块、六轴传感器 MPU6050、摄像头 OV2640、温湿度传感器 DHT11、一氧化碳检测模块、以及升压模块、推拉式门锁、电灯、风扇。

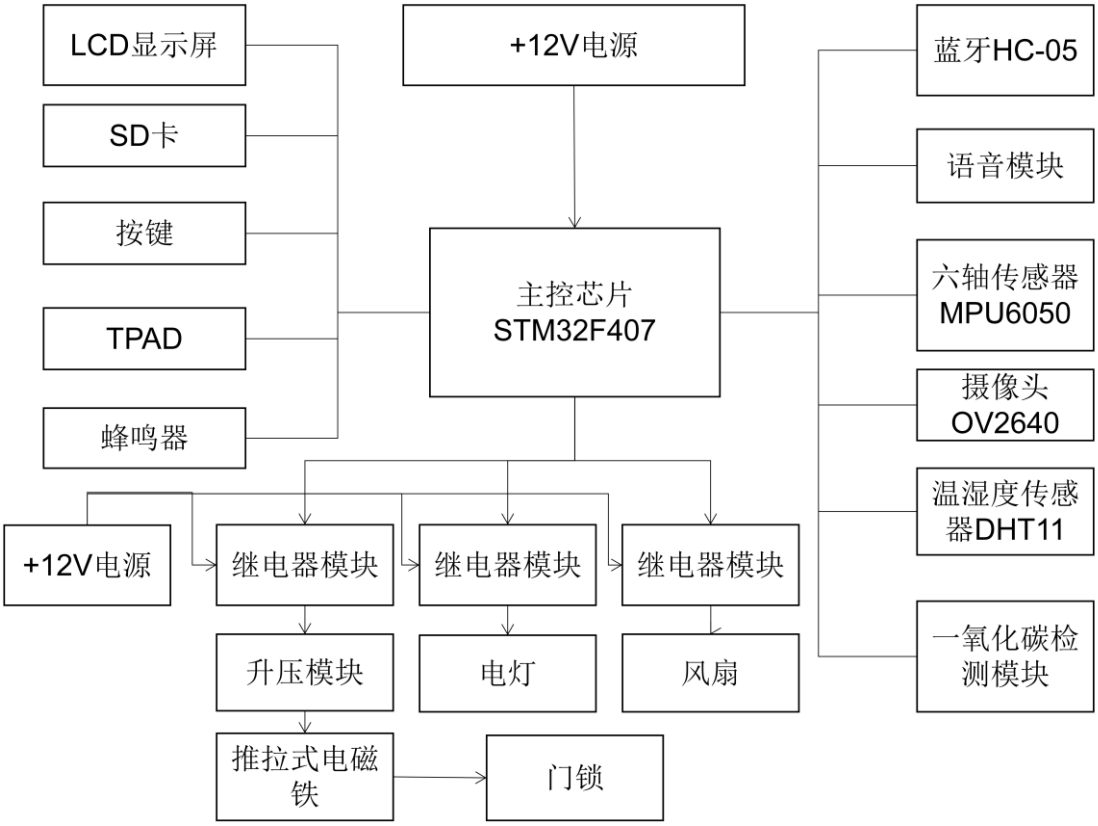


图 2 智能家居系统整体架构图

一、主控芯片方案

方案一：C51 单片机

一个全双工串行通信口，外部数据存储器寻址空间为 64kB，外部程序存储器寻址空间为 64kB，逻辑操作位寻址功能，双列直插 40PinDIP 封装，单一+5V 电源供电，CPU:由运算和控制逻辑组成，同时还包括中断系统和部分外部特殊功能寄存器；RAM:用以存放可以读写的数据，如运算的中间结果、最终结果以及欲显示的数据；ROM:

用以存放程序、一些原始数据和表格；I/O 口:四个 8 位并行 I/O 口，既可用作输入，也可用作输出；T/C:两个定时/记数器，既可以工作在定时模式，也可以工作在记数模式；五个中断源的中断控制系统；一个全双工 UART(通用异步接收发送器)的串行 I/O 口，用于实现单片机之间或单片机与微机之间的串行通信；片内振荡器和时钟产生电路，石英晶体和微调电容需要外接。最佳振荡频率为 6M-12M。

方案二：STM32F407 单片机

它拥有的资源包括： 集成 FPU 和 DSP 指令，并具有 192KB SRAM、1024KB FLASH、12 个 16 位定时器、2 个 32 位定时器、2 个 DMA 控制器(共 16 个通道)、 3 个 SPI、2 个全双工 I2S、3 个 IIC、6 个串口、2 个 USB（支持 HOST/SLAVE）、2 个 CAN、3 个 12 位 ADC、2 个 12 位 DAC、1 个 RTC（带日历功能）、1 个 SDIO 接口、1 个 FSMC 接口、1 个 10/100M 以太网 MAC 控制器、1 个摄像头接口、1 个硬件随机数生成器、以及 112 个通用 IO 口等。

STM32F407 系列面向需要在小至 10 x 10 mm 的封装内实现高集成度、高性能、嵌入式存储器和外设的医疗、工业与消费类应用。STM32F407 提供了工作频率为 168 MHz 的 Cortex™-M4 内核（具有浮点单元）的性能。

性能：在 168 MHz 频率下，从 Flash 存储器执行时，STM32F407 能够提供 210 DMIPS/566 CoreMark 性能，并且利用意法半导体的 ART 加速器实现了 FLASH 零等待状态。DSP 指令和浮点单元扩大了产品的应用范围。

功效：该系列产品采用意法半导体 90 nm 工艺和 ART 加速器，具有动态功耗调整功能，能够在运行模式下和从 Flash 存储器执行时实现低至 238 µA/MHz 的电流消耗（@ 168 MHz）。

丰富的连接功能：出色的创新型外设：与 STM32F4x5 系列相比，STM32F407 产品还具有符合 IEEE 1588 v2 标准要求的以太网 MAC10/100 和能够连接 CMOS 照相机传感器的 8~14 位并行照相机接口。可以利用支持 Compact Flash、SRAM、PSRAM、NOR 和 NAND 存储器的灵活静态存储器控制器轻松扩展存储容量。

考虑到我们要用到较多 FLASH 存储单元存储用户信息，以及需要进行蓝牙通信等，本项目采用方案二，使用 STM32F407ZG 作为主控芯片。

二、无线传输方案

方案一：采用 WiFi 的无线传输方案

Wi-Fi 模块又名串口 Wi-Fi 模块，属于物联网传输层，功能是将串口或 TTL 电平转为符合 Wi-Fi 无线网络通信标准的嵌入式模块，内置无线网络协议 IEEE802.11b.g.n 协议栈以及 TCP/IP 协议栈。传统的硬件设备嵌入 Wi-Fi 模块可以直接利用 Wi-Fi 联入互

联网，是实现无线智能家居、M2M 等物联网应用的重要组成部分。

方案二：采用蓝牙的无线传输方案

蓝牙（Bluetooth®）：是一种无线技术标准，可实现固定设备、移动设备和楼宇个人域网之间的短距离数据交换（使用 2.4—2.485GHz 的 ISM 波段的 UHF 无线电波）。蓝牙技术最初由电信巨头爱立信公司于 1994 年创制，当时是作为 RS232 数据线的替代方案。蓝牙可连接多个设备，克服了数据同步的难题。

方案三：采用 ZigBee 的无线传输方案

ZigBee 是一种基于标准的远程监控、控制和传感器网络应用技术。为满足人们对支持低数据速率、低功耗、安全性和可靠性，而且经济高效的标准型无线网络解决方案的需求，ZigBee 标准应运而生。核心市场是消费类电子产品、能源管理和效率、医疗保健、家庭自动化、电信服务、楼宇自动化以及工业自动化。围绕 ZigBee 芯片技术推出的外围电路，称之为“ZigBee 模块”，常见的 ZigBee 模块都是遵循 IEEE802.15.4 的国际标准，并且运行在 2.4GHZ 的频段上，另外，欧洲的标准是 868MHZ、北美是 915MHZ。

WiFi 模块传输的数据量大，功耗较大，蓝牙 ble 传输的数据量小，功耗低，但是传输的距离近，只有 10 米左右，ZigBee 技术传输的数据量小，但是传输距离较远，有 200 米左右，而且功耗较低。由于本次项目要求实现智能家居系统控制数据的无线传输，数据量较小，功耗要求低，距离要求低，故采用蓝牙通信技术。

3.1.2 系统功能

本智能家居系统基于蓝牙和语音控制，实现以下功能：

一、通过手机实现家电控制：

1、手机发送 Android ID 到 STM32 端实现 root 用户注册；

2、root 用户可使用手机添加或删除家庭成员；

3、root 用户手机遗失时，可通过智能家居系统的 STM32 端的 APP 实现一键重置蓝牙用户；

4、登记在 STM32 的 FLASH 里面的 Android ID 用户可实现手机控制开门/关门、开灯/关灯、开风扇/关风扇。

二、语音识别+人脸识别开门/关门。

三、语音控制开关灯、开关风扇以及切歌，关闭音乐等。

四、STM32 端通过 MPU6050 实现地震监测，通过 DHT11 和光敏传感器实现环境监测，通过 CO 检测模块实现一氧化碳检测报警。

五、用户可通过 LCD 显示屏操作 APP 实现人脸管理、人脸登记、语音留言、文本留言、查看语音、查看文本留言、查看门外情况(监控)，以及控制开门/关门、

开灯/关灯、开风扇/关风扇。

六、用户可播放音乐、播放视频等。

3.1.3 系统指标

1、语音识别准确率：80%以上；

2、人脸识别准确率：95%以上；

3、蓝牙通信准确率：99.99%；

4、CPU 耗内存：不工作时低于 10%；

5、MPU6050 能监测 3 级以上地震，并且抗干扰。

3.2 智能家居硬件设计

3.2.1 蓝牙通信模块

HC-05 蓝牙模块的电路原理图见图 3 所示。

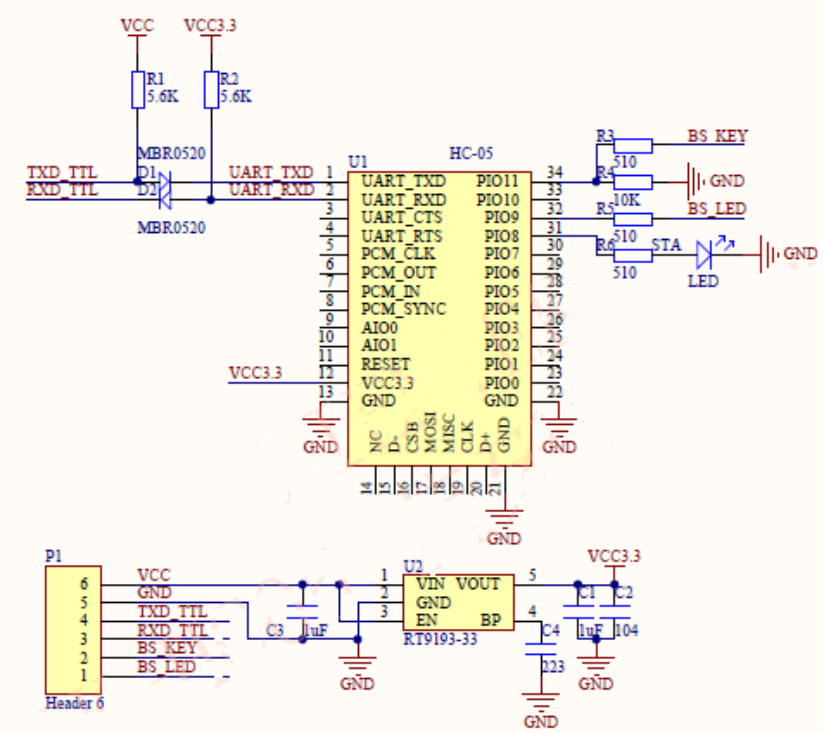


图 3 HC-05 蓝牙模块电路

模块与单片机连接最少只需要 4 根线即可：VCC、GND、TXD、RXD，VCC 和 GND 用于给模块供电，模块 TXD 和 RXD 则连接单片机的 RXD 和 TXD 即可。本模块兼容 5 V 和 3.3 V 单片机系统，所以可以很方便的连接到你的系统里面去。ATK-HC05 模块与单片机系统的典型连接方式如图 4 所示。

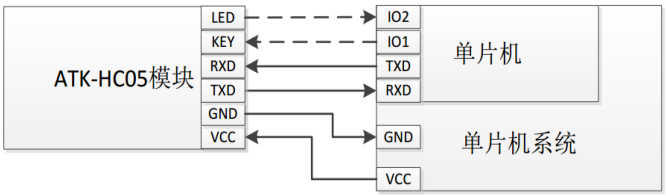


图 4 模块与单片机系统连接图

HC05 蓝牙模块接收手机蓝牙发送的指令，并发送至单片机进行信息处理。整个过程中采用串行通信。串行通信是指数据一位一位串行按顺序传送的通信方式，即构成的二进制代码序列在 1 条信道上，以位（码元）为单位，按时间顺序且按位输入方式。典型的串行传输通常由 2 根信号线构成，包括数据信号线和时钟信号线。按数据流的方向分可分为单工、半双工和全双工等 3 种方式；按数据信号和时钟信号同步与否可以分为同步通信方式和异步通信方式 2 种。生活中我们通常称呼的串行通信，其实是 UART 接口的通信，它是一种异步通信，我们在本次设计中也是采用这种方式。

异步通信中，每传输 1 帧字符，在字符的前面都必须加上起始位“0”，后面加个停止位“1”，这是一种起止式的通信方式，字符之间没有固定的间隔长度，但占用了传输时间，在要求传送数据量较大的场合，速度就慢得多。异步数据发送器先送出 1 个起始位，再送出具有一定格式的串行数据位、奇偶检验位和停止位。在不传送字符时，应插入空闲位，空闲位保持为“1”。接收端不断检测线路的状态，当数据发送器要发送 1 个字符数据时，首先发送 1 个起始位信号“0”，数据接收器检测到这个“0”，就开始准备接收。所以起始位用于表示字符传送开始，同时还被用做同步接收端时钟，以保证以后的接收正确。起始位后面是数据位，数据位可以有 5、6、7 或 8 位数据，数据位从最低位开始传送。数据位之后发送奇偶检验位，它只占据 1 位，通信双方在通信时须约定一致的奇偶校验位和数据位（在没有奇偶检验时）之后发送停止位，停止位有 1 位、1 位半和 2 位，它一定是“1”，停止位用来表示 1 个字符数据的结束。数据接收器收到停止位后，知道前一个符传送结束，同时也为接收下一个字符做准备，如果再收到“0”信号，就表示有新的字符要传送，否则就表示目前的通信结束。

异步通信的数据格式如下：①1 位起始，为低电平；②5-8 位数据位接着起始位，表示要传送的有效数据；③1 位奇偶检验位（可加也可不加）；④1 位或 1 位半或 2 位停止位，为高电平。

每一个字符由起始位、数据位、检验位、停止位构成，称为 1 帧，其典型的格式

如图 5 所示。

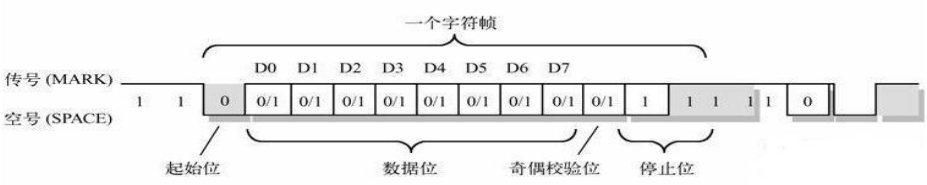
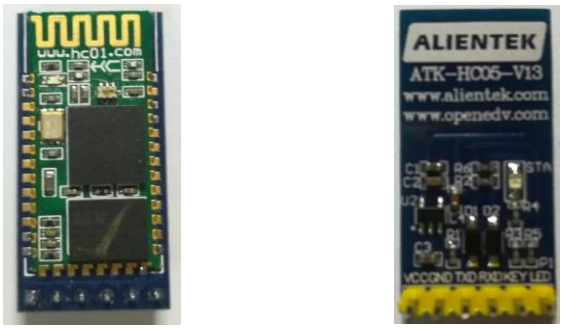


图 5 异步传送一帧数据格式

HC-05 蓝牙实物图如图 6 所示。



(a) 正面　　(b) 背面

图 6 HC-05 蓝牙模块实物图

3.2.3 摄像头模块

OV2640 是 OV（OmniVision）公司生产的一颗 1/4 寸的 CMOS UXGA（1632*1232）图像传感器。该传感器体积小、工作电压低，提供单片 UXGA 摄像头和影像处理器的所有功能。通过 SCCB 总线控制，可以输出整帧、子采样、缩放和取窗口等方式的各种分辨率 8/10 位影像数据。该产品 UXGA 图像最高达到 15 帧/秒（SVGA 可达 30 帧，CIF 可达 60 帧）。用户可以完全控制图像质量、数据格式和传输方式。所有图像处理功能过程包括伽玛曲线、白平衡、对比度、色度等都可以通过 SCCB 接口编程。OmmiVision 图像传感器应用独有的传感器技术，通过减少或消除光学或电子缺陷如固定图案噪声、拖尾、浮散等，提高图像质量，得到清晰的稳定的彩色图像。

ATK -OV2640 模块原理图如图 7 所示。

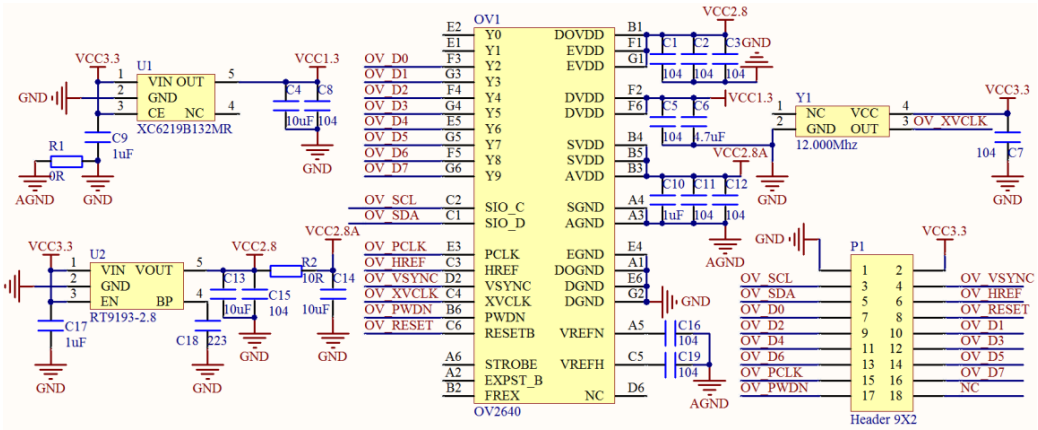


图 7 OV2640 模块电路

OV2640 的寄存器通过 SCCB 时序访问并设置。OV2640 的特点有：

- 高灵敏度、低电压适合嵌入式应用
- 标准的 SCCB 接口，兼容 IIC 接口
- 支持 RawRGB、RGB(RGB565/RGB555)、GRB422、YUV(422/420)和 YCbCr（422）输出格式
- 支持 UXGA、SXGA、SVGA 以及按比例缩小到从 SXGA 到 40*30 的任何尺寸
- 支持自动曝光控制、自动增益控制、自动白平衡、自动消除灯光条纹、自动黑电平校准等自动控制功能。同时支持色饱和度、色相、伽马、锐度等设置。
- 支持闪光灯
- 支持图像缩放、平移和窗口设置
- 支持图像压缩，即可输出 JPEG 图像数据
- 自带嵌入式微处理器

3.2.4 MPU6050 模块

MPU6050 是 InvenSense 公司推出的全球首款整合性 6 轴运动处理组件，相较于多轴运动处理组件方案，它免除了组合陀螺仪和加速度传感器时之轴间差问题，并且减少了安装空间。MPU6050 的模块电路如下图所示。

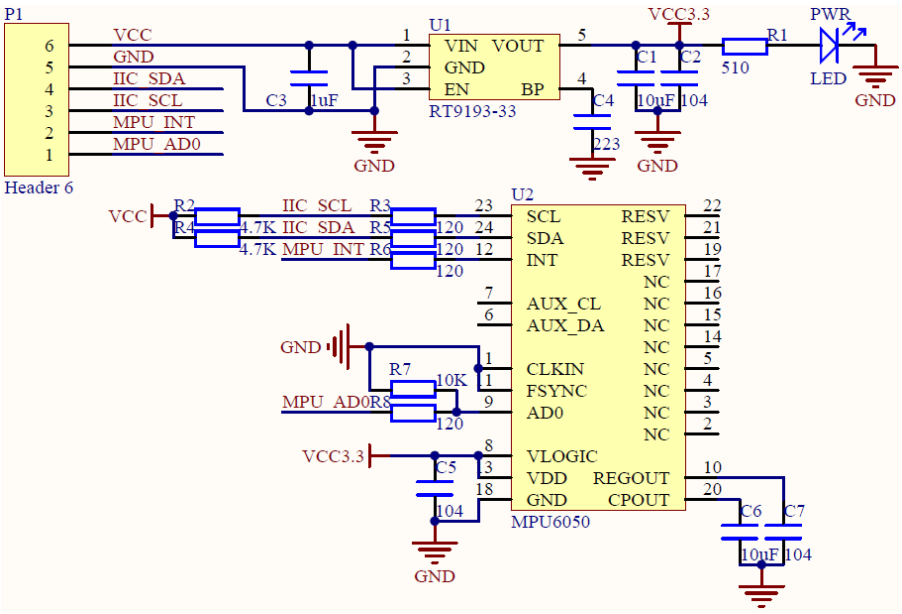


图 8 MPU6050 模块电路

MPU6050 传感器的检测轴如下图所示。

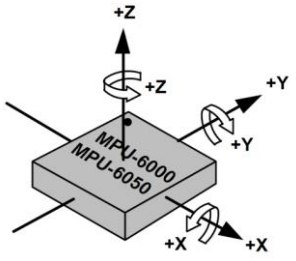
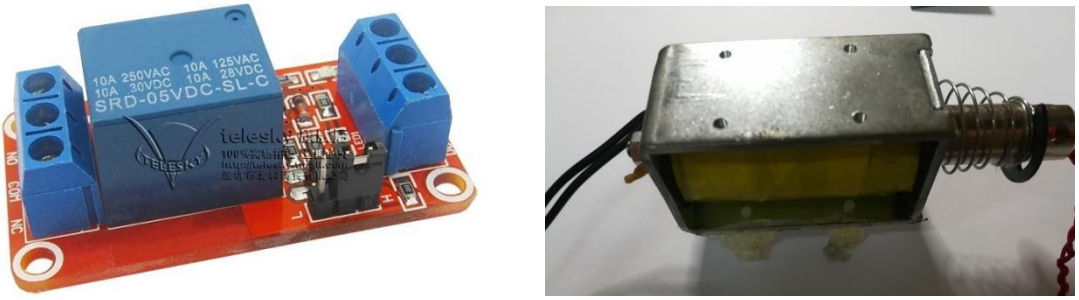


图 9 MPU6050 检测轴

3.2.5 继电器模块和电磁铁模块

智能控制模块包括继电器模块和电磁铁模块，如下图所示。



(a) 继电器模块　　(b) 电磁铁模块

图 10 智能控制模块

3.2.6 SD 卡接口电路

SD 卡接口电路如下图所示。STM32F4 的 SDIO 控制器支持多媒体卡、SD 存储卡等设备。

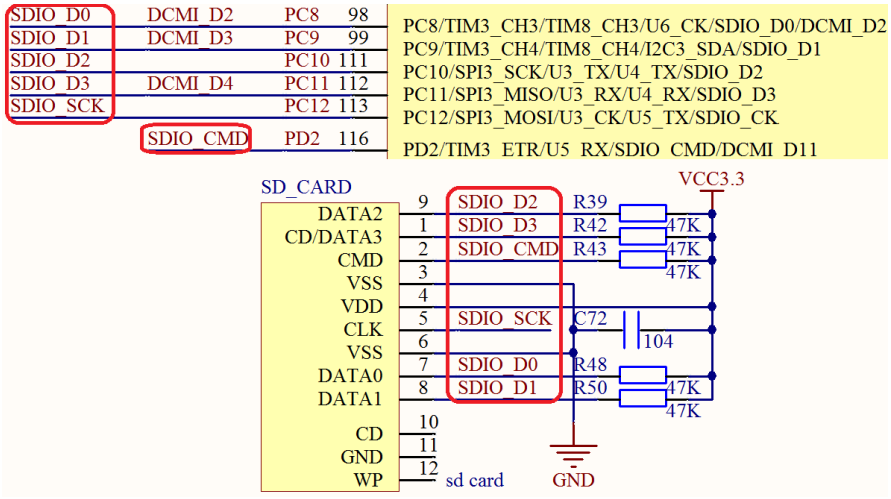


图 11 SD 卡接口电路

3.2.7 数字温湿度传感器 DHT11

DHT11 是一款湿温度一体化的数字传感器，其管脚分布如下图所示。

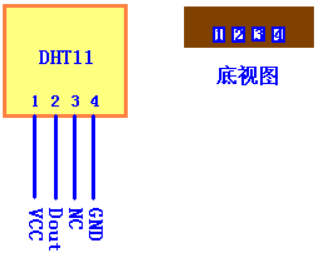


图 12 数字温湿度传感器 DHT11 管脚图

3.3 智能家居软件设计

3.3.1 智能家居 APP 设计

一、APP 界面设计

(1) 主界面

图 13 智能家居 APP 主界面

(2) 功能界面

(a) 连接界面

(b) 帮助界面

(c) 关于界面　　(d) 退出

图 14 智能家居系统 APP 界面设计

二、APP 功能实现流程

APP 的程序流程如下图所示，源码见**附录 2**。

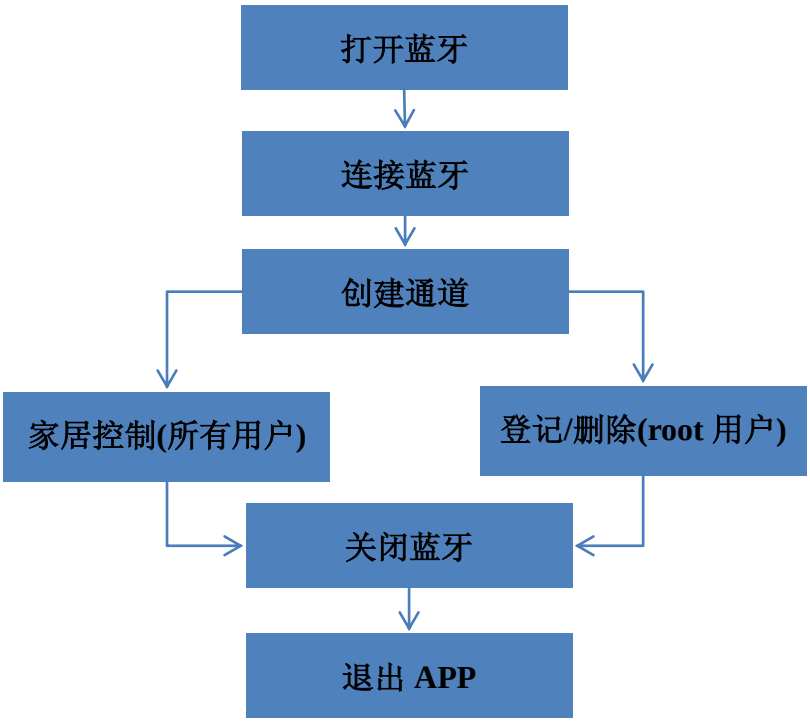


图 15 APP 功能流程

3.3.2 智能家居 STM32 控制设计

一、LCD 显示界面设计

LCD 的主界面和功能菜单界面如下图所示。其中主界面包含：人脸管理、查看语音、查看留言、系统管理、人脸登记、语音留言、文本留言、蓝牙用户、关于我们、智能家居(APP 中心)、室内环境。功能菜单包含：帮助、地震监测、人脸识别、人脸管理、电灯、风扇、门、音乐、视频、天气、设置、系统、蓝牙管理、查看留言。

(a) 主界面　　(b) 功能菜单

图 16 LCD 显示界面设计

三、STM32 源码设

STM32 程序框架如下图所示，源码见**附录 3**。

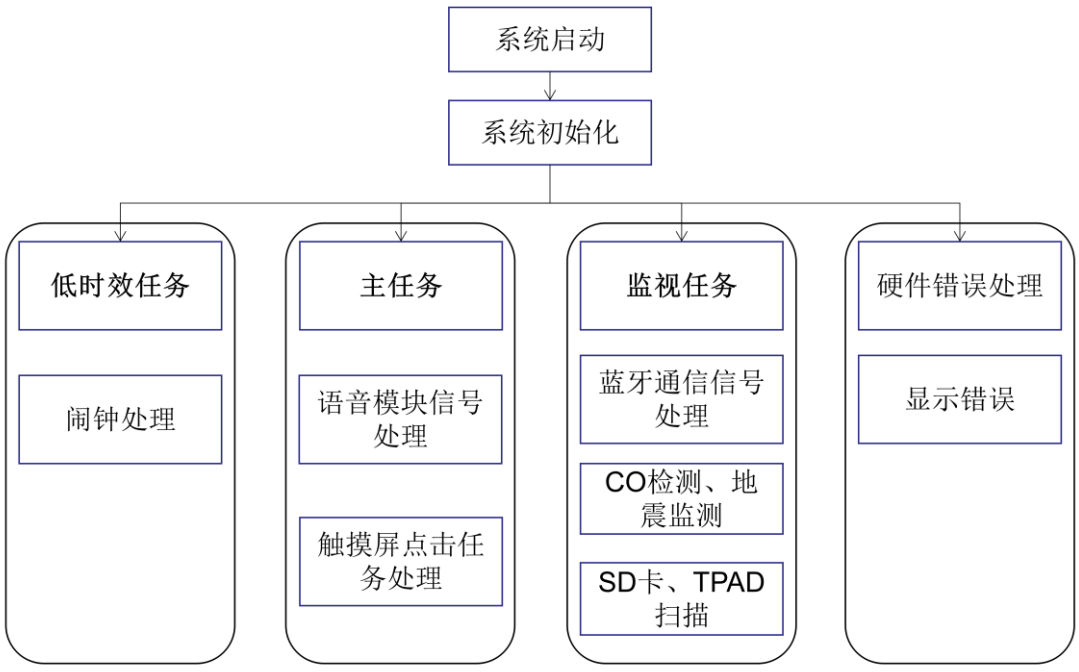


图 17 STM32 程序框架

3.4 通信协议

3.4.1 蓝牙通信

本系统采用以下蓝牙通信协议：

表 1 通信协议

发送端	接收端	指令	功能
普通用户手机	root 用户手机	Android ID	普通用户登记
root 手机	智能家居系统	Android ID	root 用户注册
手机	智能家居系统	‘D’+root 用户 Android ID+’0’	关门
手机	智能家居系统	‘D’+root 用户 Android ID+’1’	开门
root 手机	智能家居系统	‘R’+root 用户 Android ID+’T’+普通用户 Android ID	添加普通用户
root 手机	智能家居系统	‘R’+root 用户 Android ID+’S’+普通用户 Android ID	删除普通用户
手机	智能家居系统	‘L’+root 用户 Android ID+’0’	关灯
手机	智能家居系统	‘L’+root 用户 Android ID+’1’	开灯
手机	智能家居系统	‘F’+root 用户 Android ID+’0’	关风扇
手机	智能家居系统	‘F’+root 用户 Android ID+’1’	开风扇

3.4.2 系统硬件设置操作

按 RESET 复位进入开机界面(自检界面)后,迅速按以下按键:

按住 KEY0 不放:可以强制进入校准界面,对触摸屏进行校准(仅限电阻屏,对电容屏无效).

按住 KEY1 不放:可以强制进入字库更新,更新字库.

按住 KEY2 不放:可以强制擦除整个 SPI FLASH,方便使用 SD 卡快速更新系统文件.

---提示:按住 KEY0~KEY2 是要一直按住,并耐心等待.直到进入相应界面/出现提示.

第 4 章 智能家居系统实现

4.1 整体系统的搭建

搭建的智能家居系统整体效果如下图所示：

图 18 智能家居系统外观

4.2 系统测试方案

本系统的测试方案根据住户的生活情景进行整个系统的模块及整体测试：
情景一：住户不在家时，陌生人来访，进行语音及文本留言的测试
情景二：住户回家时进行人脸识别加语音开门模块的测试
情景三：用户进入房间后进行温湿度，光照强度以及留言的查看功能测试
情景四：用户使用语音控制风扇和电灯，调节室内温湿度及光照强度的测试
情景五：由于附近施工语音模块不灵敏，用户使用 APP 进行控制
情景六：夜间用户睡觉后，进行安全监测，包括地震监测报警，煤气泄漏监测报警模块的测试

4.3 智能家居系统的测试

4.3.1 智能家居 APP 调试

测试内容：按钮的有效性验证，APP 的正常进入和退出，目标功能的验证（登记，连接，发送，调用蓝牙模块等），出错处理（闪退，系统兼容性等问题的测试）

4.3.2 蓝牙通信部分调试

测试内容：有效连接范围（14 米），用户数量（1 个）经过测试同一时刻一个用户连接目的是确保系统的稳定性，防止出现多个用户同时控制电器，导致系统出错

4.3.3 语音控制部分调试

测试内容：有效范围（宿舍范围内有效），能较好，较准确的识别相关指令；分不同时间段对语音模块进行了测试：分别在中午和傍晚时刻进行 20 次测试。测试结果如下：

中午：识别正确 20 次，识别率 100%

傍晚：识别正确 14 次，识别率 70%

从以上结果可知，本系统的语音识别功能准确率足以满足使用需求。

4.3.4 人脸识别部分调试

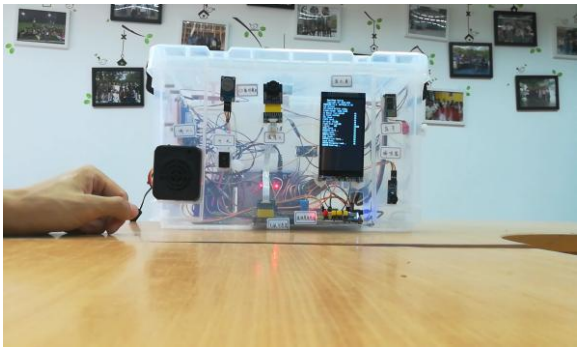
测试内容：人脸识别准确率。

白天：95%；

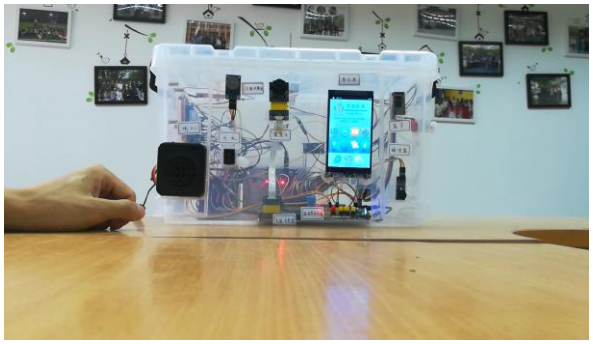
夜间：80%。

4.3.5 地震监测部分调试

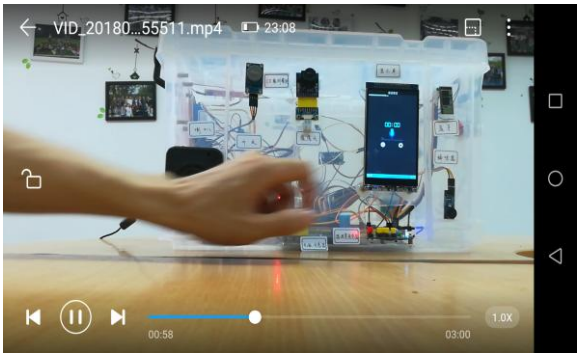
测试内容：为了对系统的地震报警功能进行检测，我们模拟地震时的摇晃场景去摇晃我们的系统模型。当系统实验箱震动超过一定角度后，系统便开始发出警报声。



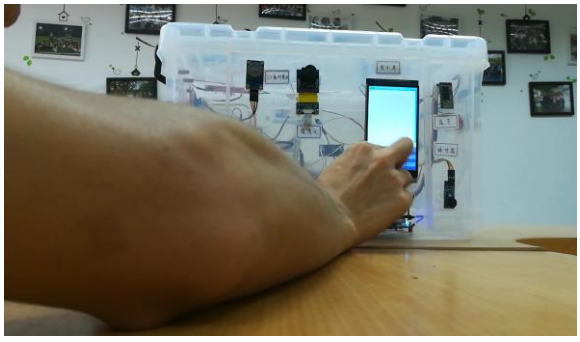
(a) 初始化



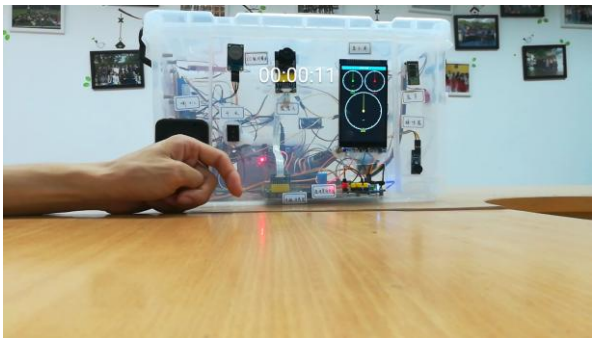
(b) 进入系统



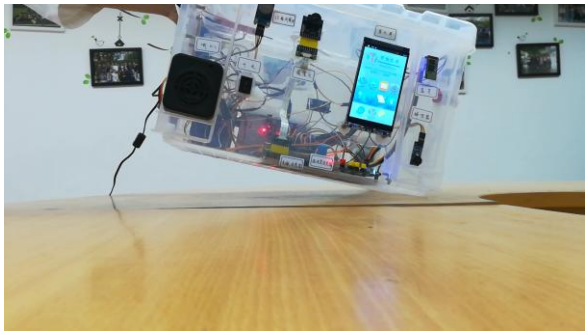
(c) 语音留言



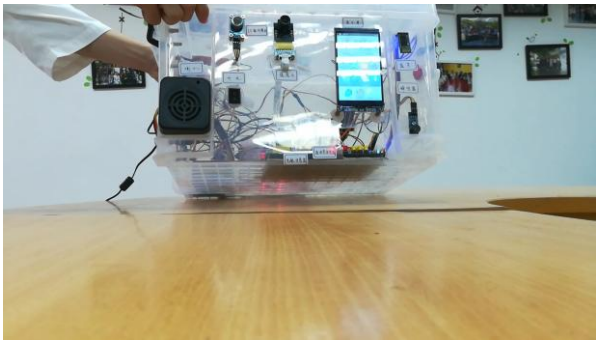
(d) 文本留言



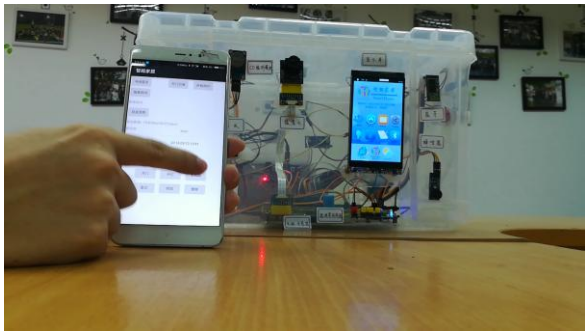
(e) 地震监测初始化



(f) 地震监测 1



(g) 地震监测 2



(h) 手机控制家电测试

图 19 系统测试

第 5 章 结论与展望

本项目通过使用 STM32F407 搭建智能家居系统，通过蓝牙和语音控制实现智能开门/关门、开灯/关灯、开风扇/关风扇功能，本系统配备智能家居硬件系统一套以及智能家居 APP 一个。通过使用本系统，可实现方便、高效、安全的家居控制。同时，我们创新性地使用 Android ID 作为手机用户的识别码，极大地提高了系统的安全性和自主性。

概括之，本文主要做了以下工作。

第一点：概括总结了目前智能家居领域的现状与发展方向。学习智能家居系统必备的专业知识，包括蓝牙通信、语音识别、人脸识别等。

第二点：学习 STM32 芯片知识，温习 C 语言知识，掌握 STM32 库函数代码及相应寄存器的使用，实现 STM32 的操作系统编写，从而实现多功能的智能家居系统。

第三点：提出使用 Android ID 作为手机用户的识别码，将智能家居系统用户分为 root 用户和普通用户，方便进行管理。

第四点：完成前三点的基础上，搭建了完整的智能家居系统，并测试了系统的功能和稳定性。

总之，本系统的高度安全以及便宜，使得它十分适合应用在高校学生宿舍、酒店、旅馆等住宿人员变动大，需要不断变换房间主人的场合。

附 录

附录 1 智能家居系统使用说明

硬件资源:

1,DS0(连接在 PF9),DS1(连接在 PF10)

2,蜂鸣器(连接在 PG14)

3,按键 KEY0(PE4)/KEY1(PE3)/KEY2(PE2)/KEY_UP(PA0,也称之为 WK_UP)

4,外部 SRAM(IS62WV51216,通过 FSMC 驱动,FSMC_NE3 接 SRAM)

5,串口 1(波特率:115200,PA9/PA10 连接在板载 USB 转串口芯片 CH340 上面)

6,串口 3(波特率:115200,PB10/PB11)

7,ALIENTEK 4.3 寸 TFTLCD 模块(通过 FSMC 驱动,FSMC_NE4 接 LCD 片选/A6 接 RS)

8,TPAD 电容触摸按键(右下角 LOGO,即 TPAD,连接在 PA5)

9,定时器 TIM1,TIM2,TIM3,TIM6,TIM7,TIM12

10,RTC(实时时钟)

11,ADC(读取内部温度传感器)

12,WM8978(音乐播放/视频播放/NES 游戏)

13,光敏传感器(连接在 PF7)

14,24C02(IIC 连接在 PB8/PB9 上面)

15,W25Q128(SPI FLASH 芯片,连接在 SPI1 上)

16,触摸屏(TFTLCD 模块自带)

17,DS18B20 传感器一个(接在 PG9 上).

18,MPU6050(IIC 连接在 PB8/PB9 上面).

19,NRF24L01 模块(SPI1(PB3/PB4/PB5)/IRQ(PG8)/CS(PG7)/CE(PG6)).

20,ALIENTEK OV2640 摄像头模块

21,USB_SLAVE 接口(通过跳线帽将 D-/D+连接在 PA11/PA12 上)

22,USB_HOST 接口(通过跳线帽将 D-/D+连接在 PA11/PA12 上)

23,LAN8720(通过 RMII 接口连接 STM32)

24,SD 卡,通过 SDIO(SDIO_D0~D4(PC8~PC11),SDIO_SCK(PC12),SDIO_CMD(PD2))连接

注意事项:

1,4.3 寸和 7 寸屏需要比较大电流,USB 供电可能不足,请用外部电源适配器(推荐外接 12V 1A 电源).

2,本例程在 LCD_Init 函数里面(在 ILI93xx.c),用到了 printf,如果不初始化串口 1,将导致液晶无法显示!!

3,自备配件包括:

a,SD 卡

b,ALIENTEK OV2640 摄像头模块

c,ALIENTEK GSM/GPRS 模块

d,DHT11 数字温度传感器

4,按 RESET 复位进入开机界面(自检界面)后,迅速按以下按键:

按住 KEY0 不放:可以强制进入校准界面,对触摸屏进行校准(仅限电阻屏,对电容屏无效).

按住 KEY1 不放:可以强制进入字库更新,更新字库.

按住 KEY2 不放:可以强制擦除整个 SPI FLASH,方便使用 SD 卡快速更新系统文件.

---提示:按住 KEY0~KEY2 是要一直按住,并耐心等待.直到进入相应界面/出现提示.

5,人脸识别功能,按 KEY_UP 按键可以进行添加模板，按住 KEY0 可进行识别.

6,在音乐播放器界面,调节的音量不会自动保存,在设置界面设置好音量,并退回主界面,才会保存.

7,P10 端口,跳线帽默认连接方式为:PB10(TX)连 COM3_RX,PB11(RX)连 COM3_TX.

8,P12 端口(多功能端口),默认用跳线帽连接:ADC 和 TPAD.

9,P6 端口,跳线帽连接方式为:PA9(T)连 RXD,PA10(R)连 TXD.

10,P11 端口,跳线帽连接方式为:PA11 连 D-,PA12 连 D+.

附录 2 智能家居系统 Android 源程序

```
package com.example.administrator.new_dianzixitongzonghesheji;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothServerSocket;
import android.bluetooth.BluetoothSocket;
import android.content.BroadcastReceiver;
import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;

import android.provider.Settings;
import android.support.v7.app.ActionBarActivity;
import android.telephony.TelephonyManager;
import android.view.View;
```

```
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import com.example.administrator.new_dianzixitongzonghesheji.BlueToothDeviceAdapter;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.PublicKey;
import java.util.ArrayList;
import java.util.List;
import java.util.Set;
import java.util.UUID;

import com.example.administrator.new_dianzixitongzonghesheji.R;


/**
 * Created by 56305 on 2018/6/28.
 */
public class bluetooth_lianjie extends ActionBarActivity implements View.OnClickListener   {

    public  MainActivity a;
    private BluetoothAdapter bTAdatper;
    private ListView listView;
    private ListView listView1;


    private BlueToothDeviceAdapter adapter;
    String androidId = null;
    static String androidId_data;
    static String new_androidId_data;
    Toast tst;
    TextView text_state;
    TextView textView1;
    private TextView text_msg;
    boolean kaimenorguanmen=true;
```

```
boolean kaidengorguandeng=true;
boolean kaifengshanorguanfengshan=true;
boolean switch_bluetooth=true;
private final int BUFFER_SIZE = 1024;
byte[] buffer = new byte[BUFFER_SIZE];
int bytes;


private static final String NAME = "BT_DEMO";
private static final UUID BT_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
boolean lianjie_state=false;
ConnectThread connectThread;
private ListenerThread listenerThread;
Button btn_kaimen,btn_openBT,btn_kaideng,btn_kaifengshan;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.bluetooth_lianjie);
    initView();
    bTAdatper = BluetoothAdapter.getDefaultAdapter();
    initReceiver();
    listenerThread = new ListenerThread();
    listenerThread.start();
    btn_kaimen=(Button)findViewById(R.id.btn_kaimen);
    btn_kaimen.setOnClickListener(this);

    btn_kaideng=(Button)findViewById(R.id.btn_kaideng);
    btn_kaideng.setOnClickListener(this);
    btn_kaifengshan=(Button)findViewById(R.id.btn_kaifengshan);
    btn_kaifengshan.setOnClickListener(this);

    btn_openBT=(Button)findViewById(R.id.btn_openBT);
    btn_openBT.setOnClickListener(this);


    listView1 = new ListView(this);


    androidId_data="7659483c763e8649";
```

```
        if (!bTAdatper.isEnabled())
        {btn_openBT.setText("打开蓝牙");
            switch_bluetooth=true;
        }
        else
        {btn_openBT.setText("关闭蓝牙");
            switch_bluetooth=false;
        }
    }

    private void initView() {
        // findViewById(R.id.btn_openBT).setOnClickListener(this);
        findViewById(R.id.btn_search).setOnClickListener(this);
        findViewById(R.id.btn_send).setOnClickListener(this);
        findViewById(R.id.btn_dengji).setOnClickListener(this);
        // findViewById(R.id.btn_kaimen).setOnClickListener(this);
        findViewById(R.id.btn_tianjia).setOnClickListener(this);
        findViewById(R.id.btn_shanchu).setOnClickListener(this);
        findViewById(R.id.btn_record).setOnClickListener(this);
        findViewById(R.id.btn_user).setOnClickListener(this);
        // findViewById(R.id.btn_kaideng).setOnClickListener(this);
        // findViewById(R.id.btn_kaifengshan).setOnClickListener(this);
        text_state = (TextView) findViewById(R.id.text_state);
        text_msg = (TextView) findViewById(R.id.text_msg);
        textView1=(TextView) findViewById(R.id.textView1);
        androidId = Settings.Secure.getString(getContentResolver(), Settings.Secure.ANDROID_ID);
        listView = (ListView) findViewById(R.id.listView);
        adapter = new BlueToothDeviceAdapter(getApplicationContext(), R.layout.device_list);
        listView.setAdapter(adapter);
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                if (bTAdatper.isDiscovering()) {
                    bTAdatper.cancelDiscovery();
                }
                BluetoothDevice device = (BluetoothDevice) adapter.getItem(position);
                //连接设备
                connectDevice(device);
            }
        });
    }
```

```
    private void initReceiver() {
        //注册广播
        IntentFilter filter = new IntentFilter();
        filter.addAction(BluetoothDevice.ACTION_FOUND);
        filter.addAction(BluetoothAdapter.ACTION_DISCOVERY_STARTED);
        filter.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
        registerReceiver(mReceiver, filter);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_openBT:
               if(switch_bluetooth)
               { openBlueTooth();
                   btn_openBT.setText("关闭蓝牙");
                   switch_bluetooth=false;
               }
                else
               {
                   bTAdatper.disable();
                   btn_openBT.setText("打开蓝牙");
                   Toast.makeText(getApplicationContext(), "蓝牙已关闭", Toast.LENGTH_LONG);
                   switch_bluetooth=true;
               }
                break;
            case R.id.btn_search:
                if( bTAdatper.isEnabled()) {
                    searchDevices();
                    textView1.setText("房间名  \t\t
\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\tMAC");
                }
                else {
                    Toast.makeText(bluetooth_lianjie.this, "请先打开蓝牙！",
Toast.LENGTH_SHORT).show();
                }
                break;
            case R.id.btn_send:
                if (connectThread != null) {
                    connectThread.sendMsg("111");
```

```
                }
                break;
            /* case R.id.fanhui:
             {  Intent intent =new
Intent(bluetooth_lianjie.this,com.example.administrator.new_dianzixitongzongheseji.MainActivity.cl
ass);
                startActivity(intent);
                //  finish();
            }
            break;*/
            case R.id.btn_dengji: {
                if( lianjie_state) {

                    tst = Toast.makeText(bluetooth_lianjie.this, androidId, Toast.LENGTH_SHORT);
                    tst.show();
                    //  CopyToClipboard(context,androidId);
                    ClipboardManager cm = (ClipboardManager)
getSystemService(Context.CLIPBOARD_SERVICE);
                    // 创建普通字符型ClipData
                    ClipData mClipData = ClipData.newPlainText("本机Android_ID是", androidId);
                    cm.setPrimaryClip(mClipData);
                   // connectThread.sendMsg(getPhoneNumber()+androidId);         登记时发送 手机号码
+AndroidID,
                    connectThread.sendMsg(androidId);
                }
//即添加用户时发送 手机号码+AndroidID 给stm32

                else
                    Toast.makeText(bluetooth_lianjie.this, "请先连接蓝牙！",
Toast.LENGTH_SHORT).show();
            }
            break;

            case R.id.btn_kaimen:    //开门
            {
                if( lianjie_state) {
                    if (kaimenorguanmen)

                    {
```

```
                    connectThread.sendMsg("D" + androidId + "1");

                    kaimenorguanmen = false;
                    btn_kaimen.setText("关门");
                } else {

                    connectThread.sendMsg("D" + androidId + "0");


                    kaimenorguanmen = true;
                    btn_kaimen.setText("开门");

                }
            }
            else
                Toast.makeText(bluetooth_lianjie.this, "请先连接蓝牙！",
Toast.LENGTH_SHORT).show();
            }
            break;

            case R.id.btn_kaideng:
            {
                if( lianjie_state) {
                    if (kaidengorguandeng)

                    {

                        connectThread.sendMsg("L" + androidId + "1");    //开灯
                    kaidengorguandeng = false;
                    btn_kaideng.setText("关灯");
                } else {

                        connectThread.sendMsg("L" + androidId + "0");     //关灯


                    kaidengorguandeng = true;
                    btn_kaideng.setText("开灯");

                }
            }
            else
```

```
                    Toast.makeText(bluetooth_lianjie.this, "请先连接蓝牙！",
Toast.LENGTH_SHORT).show();
            }
            break;
            case R.id.btn_kaifengshan:
            {
                if( lianjie_state) {
                    if (kaifengshanorguanfengshan)

                    {

                        connectThread.sendMsg("F" + androidId + "1");      //开风扇
                        kaifengshanorguanfengshan = false;
                        btn_kaifengshan.setText("关风扇");
                    } else {

                        connectThread.sendMsg("F" + androidId + "0");  //关风扇


                        kaifengshanorguanfengshan = true;
                        btn_kaifengshan.setText("开风扇");

                    }
                }
                else
                    Toast.makeText(bluetooth_lianjie.this, "请先连接蓝牙！",
Toast.LENGTH_SHORT).show();
            }
            break;

            case R.id.btn_user:     //返回用户
            {
                if( lianjie_state) {
                    connectThread.sendMsg("U" + androidId );




                }
                else
                    Toast.makeText(bluetooth_lianjie.this, "请先连接蓝牙！",
```

```
Toast.LENGTH_SHORT).show();
            }
            break;
            case R.id.btn_record:     //返回开门记录
            {
                if( lianjie_state) {
                    connectThread.sendMsg("J" + androidId );
                    ;



                }
                else
                    Toast.makeText(bluetooth_lianjie.this, "请先连接蓝牙！",
Toast.LENGTH_SHORT).show();
            }
            break;
            case R.id.btn_tianjia: {

             if(text_msg.getText().toString()!=null)

{    if((returnResultMultiple(text_msg.getText().toString()).length()==16)||(returnResultMultiple(t
ext_msg.getText().toString()).length()==15))
                {
                    connectThread.sendMsg( androidId +
"T"+returnResultMultiple(text_msg.getText().toString()));
                    Toast.makeText(bluetooth_lianjie.this, "添加权限成功",
Toast.LENGTH_SHORT).show();
                }
                else//添加
                {
                    Toast.makeText(bluetooth_lianjie.this, "请录入正确的要添加的身份码",
Toast.LENGTH_SHORT).show();

                }
                }
                else  {
                    Toast.makeText(bluetooth_lianjie.this, "请录入要添加的身份码",
Toast.LENGTH_SHORT).show();

                }
                //   finish();
```

```
            }
        break;
        case R.id.btn_shanchu: {
                if(text_msg.getText().toString()!=null)
                    {

if((returnResultMultiple(text_msg.getText().toString()).length()==16)||(returnResultMultiple(text_m
sg.getText().toString()).length()==15))
                        {
                            connectThread.sendMsg( androidId +
"S"+returnResultMultiple(text_msg.getText().toString()));
                            Toast.makeText(bluetooth_lianjie.this, "删除权限成功",
Toast.LENGTH_SHORT).show();
                        }
                        else
                        {
                            Toast.makeText(bluetooth_lianjie.this, "请录入正确的要删除的身份码",
Toast.LENGTH_SHORT).show();

                        }
                        //删除
                    }
                    //   finish();
                    else  {
                        Toast.makeText(bluetooth_lianjie.this, "请录入要添加的身份码",
Toast.LENGTH_SHORT).show();

                    }
            }
            break;
        }
    }


    /**
     * 开启蓝牙
     */
    private void openBlueTooth() {
        if (bTAdatper == null) {
```

```
            Toast.makeText(this, "当前设备不支持蓝牙功能", Toast.LENGTH_SHORT).show();
        }
        if (!bTAdatper.isEnabled()) {
            /* Intent i = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
             startActivity(i);*/
            bTAdatper.enable();
        }
        else
            Toast.makeText(bluetooth_lianjie.this, "蓝牙已打开", Toast.LENGTH_SHORT).show();
        //开启被其它蓝牙设备发现的功能
        if (bTAdatper.getScanMode() != BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE) {
            Intent i = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
            //设置为一直开启
            i.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, 0);
            startActivity(i);
        }
    }

    /**
     * 搜索蓝牙设备
     */
    private void searchDevices() {
        if (!bTAdatper.isEnabled()) {
            /* Intent i = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
             startActivity(i);*/
            Toast.makeText(bluetooth_lianjie.this, "蓝牙未打开，请打开蓝牙",
Toast.LENGTH_SHORT).show();
        }
        if (bTAdatper.isDiscovering()) {
            bTAdatper.cancelDiscovery();
        }
        getBoundedDevices();
        bTAdatper.startDiscovery();

    }

    /**
     * 获取已经配对过的设备
     */
    private void getBoundedDevices() {
        //获取已经配对过的设备
```

```
        Set<BluetoothDevice> pairedDevices = bTAdatper.getBondedDevices();
        //将其添加到设备列表中
        if (pairedDevices.size() > 0) {
            adapter.clear();
            for (BluetoothDevice device : pairedDevices) {
                adapter.add(device);
            }
        }
    }

    /**
     * 连接蓝牙设备
     */
    private void connectDevice(BluetoothDevice device) {

        text_state.setText(getResources().getString(R.string.connecting));

        try {
            //创建 Socket
            BluetoothSocket socket = device.createRfcommSocketToServiceRecord(BT_UUID);
            //启动连接线程
            connectThread = new ConnectThread(socket, true);
            connectThread.start();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        //取消搜索
        if (bTAdatper != null && bTAdatper.isDiscovering()) {
            bTAdatper.cancelDiscovery();
        }
        //注销 BroadcastReceiver，防止资源泄露
        unregisterReceiver(mReceiver);
    }

    private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
        @Override
```

```
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            if (BluetoothDevice.ACTION_FOUND.equals(action)) {
                BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                //避免重复添加已经绑定过的设备
                if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
                    adapter.add(device);
                    adapter.notifyDataSetChanged();
                }
            } else if (BluetoothAdapter.ACTION_DISCOVERY_STARTED.equals(action)) {
                Toast.makeText(bluetooth_lianjie.this, "开始搜索", Toast.LENGTH_SHORT).show();
            } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
                Toast.makeText(bluetooth_lianjie.this, "搜索完毕", Toast.LENGTH_SHORT).show();
            }
        }
    };

    /**
     * 连接线程
     */
    public class ConnectThread extends Thread {

        private BluetoothSocket socket;
        private boolean activeConnect;
        InputStream inputStream;
        OutputStream outputStream;

        public ConnectThread(BluetoothSocket socket, boolean connect) {
            this.socket = socket;
            this.activeConnect = connect;
        }

        @Override
        public void run() {
            try {
                //如果是自动连接 则调用连接方法
                if (activeConnect) {
                    socket.connect();
                }
                text_state.post(new Runnable() {
                    @Override
```

```
                public void run() {
                    text_state.setText(getResources().getString(R.string.connect_success));
                    lianjie_state=true;
                }
            });
            inputStream = socket.getInputStream();
            outputStream = socket.getOutputStream();

            //byte[] buffer = new byte[BUFFER_SIZE];
            //int bytes;
            while (true) {
                //读取数据
                bytes = inputStream.read(buffer);
                if (bytes > 0) {
                    final byte[] data = new byte[bytes];
                    System.arraycopy(buffer, 0, data, 0, bytes);

                    text_msg.post(new Runnable() {
                        @Override
                        public void run() {
                                                                    //省赛补充-周泽鑫 2018/7/31
                                text_msg.setText(getResources().getString(R.string.get_msg) +
new String(data));

                        }
                    });
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
            text_state.post(new Runnable() {
                @Override
                public void run() {
                    text_state.setText(getResources().getString(R.string.connect_error));
                    lianjie_state=false;
                }
            });
        }
    }

    /**
```

```
     * 发送数据
     *
     * @param msg
     */
    public void sendMsg(final String msg) {

        byte[] bytes = msg.getBytes();
        if (outputStream != null) {
            try {
                //发送数据
                outputStream.write(bytes);
                text_msg.post(new Runnable() {
                    @Override
                    public void run() {
                        text_msg.setText(getResources().getString(R.string.send_msgs)+msg);
                    }
                });
            } catch (IOException e) {
                e.printStackTrace();
                text_msg.post(new Runnable() {
                    @Override
                    public void run() {
                        text_msg.setText(getResources().getString(R.string.send_msg_error)+msg);
                    }
                });
            }
        }
    }
}

/**
 * 监听线程
 */
private class ListenerThread extends Thread {

    private BluetoothServerSocket serverSocket;
    private BluetoothSocket socket;

    @Override
    public void run() {
        try {
```

```
                serverSocket = bTAdatper.listenUsingRfcommWithServiceRecord(
                        NAME, BT_UUID);
                while (true) {
                    //线程阻塞，等待别的设备连接
                    socket = serverSocket.accept();
                    text_state.post(new Runnable() {
                        @Override
                        public void run() {
                            text_state.setText(getResources().getString(R.string.connecting));
                        }
                    });
                    connectThread = new ConnectThread(socket, false);
                    connectThread.start();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    private static String returnResultMultiple(String str) {
        String string = "";
        if (str.equals("")) {
            return "";
        }
        for (int i = 0; i < str.length(); i++) {
            char ch = str.charAt(i);
            if ((ch>=65&&ch<=122)||(ch>=48&&ch<=57)){
                string = string + ch;
            }
        }
        return string;
    }
}
```

附录 3 智能家居系统 STM32 源程序主函数

```
#include "includes.h"
#include "malloc.h"
```

```
#include "spblcd.h"
#include "spb.h"
#include "common.h"
#include "ebook.h"
#include "settings.h"
#include "picviewer.h"
#include "audioplay.h"
#include "videoplay.h"
#include "calendar.h"
#include "nesplay.h"
#include "notepad.h"
#include "exeplay.h"
#include "paint.h"
#include "camera.h"
#include "recorder.h"
#include "usbplay.h"
#include "netplay.h"
#include "wirelessplay.h"
#include "calculator.h"
#include "phoneplay.h"
#include "appplay.h"
#include "smsplay.h"

#include "usart3.h"
#include "sim900a.h"
#include "mpu6050.h"
#include "wm8978.h"
#include "lan8720.h"
#include "lsens.h"
#include "usb_app.h"
//-------------------------------------------------------------------------------------------
//  设置 FLASH  保存地址(必须为偶数，且所在扇区,要大于本代码所占用到的扇区.
//  否则,写操作的时候,可能会导致擦除整个扇区,从而引起部分程序丢失.引起死机.
#define FLASH_DATA_SAVE        0X080FF000

// 0~3:num
// 4~19:root user Android ID
// 20~35:common user Android ID 1
// ...
// 132~147:common user Android ID 8
u8 flash_data_save[148];
```

```
// 用于存放 root ID
const u8 id_root_buf[]={"7659483c763e8649"};
// 用户 id 索引，其中 find_id[0]为 root 用户 id
u8 find_id[9]={4,20,36,52,68,84,100,116,132};

#define COMMON_BUF_LENTH sizeof(id_common_buf)
#define ROOT_BUF_LENTH sizeof(id_root_buf)
#define COMMON_BUF_SIZE COMMON_BUF_LENTH/4+((COMMON_BUF_LENTH%4)?1:0)
#define ROOT_BUF_SIZE ROOT_BUF_LENTH/4+((ROOT_BUF_LENTH%4)?1:0)

//////////////////////////提示信息/////////////////////////////////

//删除所有蓝牙用户
u8*const delethbt_remindmsg_tbl[GUI_LANGUAGE_NUM]=
{
"你已经删除所有蓝牙用户",
"你已经删除所有蓝牙用户",
"你已经删除所有蓝牙用户",
};

//删除所有蓝牙用户
u8*const ndelethbt_remindmsg_tbl[GUI_LANGUAGE_NUM]=
{
"你没有删除蓝牙用户",
"你没有删除蓝牙用户",
"你没有删除蓝牙用户",
};

//关于我们提示信息
u8*const aboutus_remindmsg_tbl[GUI_LANGUAGE_NUM]=
{
"2018 年广东省大学生电子设计竞赛\r\
一种基于蓝牙和语音控制的智能家居系统\r\
手机通过蓝牙连接本系统可以实现控制\r\
通过语音控制+人脸识别实现安全的家居控制\r\
参赛队伍号:00237\r\
作品属性:智能家居",
"2018 年广东省大学生电子设计竞赛\r\
一种基于蓝牙和语音控制的智能家居系统\r\
```

```
手机通过蓝牙连接本系统可以实现控制\r\
通过语音控制+人脸识别实现安全的家居控制\r\
参赛队伍号:00237\r\
作品属性:智能家居",
"2018 年广东省大学生电子设计竞赛\r\
一种基于蓝牙和语音控制的智能家居系统\r\
手机通过蓝牙连接本系统可以实现控制\r\
通过语音控制+人脸识别实现安全的家居控制\r\
参赛队伍号:00237\r\
作品属性:智能家居",
};

//地震提示信息
u8*const dizhen_remindmsg_tbl[GUI_LANGUAGE_NUM]=
{
"危险警报：\r\
地震啦！\r\
地震啦！！\r\
地震啦！！！\r\
地震啦！！！\r\
地震啦！！！",
"危险警报：\r\
地震啦！\r\
地震啦！！\r\
地震啦！！！\r\
地震啦！！！\r\
地震啦！！！",
"危险警报：\r\
地震啦！\r\
地震啦！！\r\
地震啦！！！\r\
地震啦！！！\r\
地震啦！！！",
};



//////////////////////////UCOSII 任务设置///////////////////////////////////
//START 任务
//设置任务优先级
#define START_TASK_PRIO                     10 //开始任务的优先级设置为最低
```

```
//设置任务堆栈大小
#define START_STK_SIZE                  64
//任务堆栈，8 字节对齐
__align(8) static OS_STK START_TASK_STK[START_STK_SIZE];
//任务函数
void start_task(void *pdata);

//串口任务
//设置任务优先级
#define USART_TASK_PRIO                 7
//设置任务堆栈大小
#define USART_STK_SIZE                  128
//任务堆栈，8 字节对齐
__align(8) static OS_STK USART_TASK_STK[USART_STK_SIZE];
//任务函数
void usart_task(void *pdata);

//主任务
//设置任务优先级
#define MAIN_TASK_PRIO                  6
//设置任务堆栈大小
#define MAIN_STK_SIZE                   1200
//任务堆栈，8 字节对齐
__align(8) static OS_STK MAIN_TASK_STK[MAIN_STK_SIZE];
//任务函数
void main_task(void *pdata);

//监视任务
//设置任务优先级
#define WATCH_TASK_PRIO                 3
//设置任务堆栈大小
#define WATCH_STK_SIZE                  256
//任务堆栈，8 字节对齐
__align(8) static OS_STK WATCH_TASK_STK[WATCH_STK_SIZE];
//任务函数
void watch_task(void *pdata);
////////////////////////////////////////////////////////////////////////////////

//外部内存测试(最大支持 1M 字节内存测试)
//x,y:坐标
//fsize:字体大小
```

```
//返回值:0,成功;1,失败.
u8 system_exsram_test(u16 x,u16 y,u8 fsize)
{
    u32 i=0;
    u16 temp=0;
    u16 sval=0;  //在地址 0 读到的数据
    LCD_ShowString(x,y,lcddev.width,y+fsize,fsize,"Ex Memory Test:      0KB");
    //每隔 1K 字节,写入一个数据,总共写入 1024 个数据,刚好是 1M 字节
    for(i=0;i<1024*1024;i+=1024)
    {
        FSMC_SRAM_WriteBuffer((u8*)&temp,i,2);
        temp++;
    }
    //依次读出之前写入的数据,进行校验
    for(i=0;i<1024*1024;i+=1024)
    {
        FSMC_SRAM_ReadBuffer((u8*)&temp,i,2);
        if(i==0)sval=temp;
        else if(temp<=sval)break;//后面读出的数据一定要比第一次读到的数据大.

        LCD_ShowxNum(x+15*(fsize/2),y,(u16)(temp-sval+1),4,fsize,0);//显示内存容量
    }
    if(i>=1024*1024)
    {
        LCD_ShowxNum(x+15*(fsize/2),y,i/1024,4,fsize,0);//显示内存值
        return 0;//内存正常,成功
    }
    return 1;//失败
}
//显示错误信息
//x,y:坐标
//fsize:字体大小
//x,y:坐标.err:错误信息
void system_error_show(u16 x,u16 y,u8*err,u8 fsize)
{
    POINT_COLOR=RED;
    while(1)
    {
        LCD_ShowString(x,y,lcddev.width,lcddev.height,fsize,err);
        delay_ms(400);
        LCD_Fill(x,y,lcddev.width,y+fsize,BLACK);
```

```
            delay_ms(100);
        }
    }
    //擦除整个 SPI FLASH(即所有资源都删除),以快速更新系统.
    //x,y:坐标
    //fsize:字体大小
    //x,y:坐标.err:错误信息
    //返回值:0,没有擦除;1,擦除了
    u8 system_files_erase(u16 x,u16 y,u8 fsize)
    {
        u8 key;
        u8 t=0;
        POINT_COLOR=RED;
        LCD_ShowString(x,y,lcddev.width,lcddev.height,fsize,"Erase all system files?");
        while(1)
        {
            t++;
            if(t==20)LCD_ShowString(x,y+fsize,lcddev.width,lcddev.height,fsize,"KEY0:NO  /
    KEY2:YES");
            if(t==40)
            {
                gui_fill_rectangle(x,y+fsize,lcddev.width,fsize,BLACK);//清除显示
                t=0;

            }
            key=KEY_Scan(0);
            if(key==KEY0_PRES)//不擦除,用户取消了
            {
                gui_fill_rectangle(x,y,lcddev.width,fsize*2,BLACK);//清除显示
                POINT_COLOR=WHITE;
                return 0;
            }
            if(key==KEY2_PRES)//要擦除,要重新来过
            {
                LCD_ShowString(x,y+fsize,lcddev.width,lcddev.height,fsize,"Erasing          SPI
    FLASH...");
                W25QXX_Erase_Chip();
                LCD_ShowString(x,y+fsize,lcddev.width,lcddev.height,fsize,"Erasing          SPI
    FLASH OK");
                delay_ms(600);
                return 1;
```

```
            }
            delay_ms(10);
        }
    }
    //字库更新确认提示.
    //x,y:坐标
    //fsize:字体大小
    //返回值:0,不需要更新;1,确认要更新
    u8 system_font_update_confirm(u16 x,u16 y,u8 fsize)
    {
        u8 key;
        u8 t=0;
        u8 res=0;
        POINT_COLOR=RED;
        LCD_ShowString(x,y,lcddev.width,lcddev.height,fsize,"Update font?");
        while(1)
        {
            t++;
            if(t==20)LCD_ShowString(x,y+fsize,lcddev.width,lcddev.height,fsize,"KEY0:NO  /
    KEY2:YES");
            if(t==40)
            {
                gui_fill_rectangle(x,y+fsize,lcddev.width,fsize,BLACK);//清除显示
                t=0;

            }
            key=KEY_Scan(0);
            if(key==KEY0_PRES)break;//不更新
            if(key==KEY2_PRES){res=1;break;}//要更新
            delay_ms(10);
        }

        gui_fill_rectangle(x,y,lcddev.width,fsize*2,BLACK);//清除显示
        POINT_COLOR=WHITE;
        return res;
    }
    //系统初始化
    void system_init(void)
    {
        u16 okoffset=162;
        u16 ypos=0;
```

```
u16 j=0;
u16 temp=0;
u8 res;
u32 dtsize,dfsize;
u8 *stastr=0;
u8 *version=0;
u8 verbuf[12];
u8 fsize;
u8 icowidth;

Stm32_Clock_Init(336,8,2,7);//设置时钟,168Mhz
exeplay_app_check();              //检测是否需要运行 APP 代码.
delay_init(168);                  //延时初始化
uart_init(84,115200);             //初始化串口波特率为 115200
usart3_init(42,115200);           //初始化串口 2 波特率为 115200
usmart_dev.init(84);              //初始化 USMART
LED_Init();                       //初始化 LED
BEEP_Init();                      //初始化蜂鸣器
Lsens_Init();                     //初始化光敏传感器
CODETECT_Init();                  //初始化 CO 监测传感器
LCD_Init();                       //LCD 初始化
HC05_Init();                      //蓝牙初始化
//mpu_dmp_init();                 //初始化 MPU6050
RS485_Init(42,9600);              //初始化 RS485
//MOTOR_Init();                   //初始化电机
KEY_Init();                       //按键初始化
FSMC_SRAM_Init();                 //初始化外部 SRAM
AT24CXX_Init();                   //EEPROM 初始化
W25QXX_Init();                    //初始化 W25Q128
Adc_Init();                       //初始化内部温度传感器
Lsens_Init();                     //初始化光敏传感器

my_mem_init(SRAMIN);              //初始化内部内存池
my_mem_init(SRAMCCM);             //初始化 CCM 内存池

tp_dev.init();
gui_init();
piclib_init();                    //piclib 初始化
slcd_dma_init();
usbapp_init();
exfuns_init();                    //FATFS  申请内存
```

```
    version=mymalloc(SRAMIN,31);//申请 31 个字节内存
REINIT://重新初始化
    LCD_Clear(BLACK);               //黑屏
    POINT_COLOR=WHITE;
    BACK_COLOR=BLACK;
    j=0;
////////////////////////////////////////////////////////////////////////
//显示版权信息
    ypos=2;
    if(lcddev.width==240)
    {
        fsize=12;
        icowidth=18;
        okoffset=190;

    //app_show_mono_icos(5,ypos,icowidth,24,(u8*)APP_ALIENTEK_ICO1824,YELLO
W,BLACK);
    }else if(lcddev.width==320)
    {
        fsize=16;
        icowidth=24;
        okoffset=250;

    //app_show_mono_icos(5,ypos,icowidth,32,(u8*)APP_ALIENTEK_ICO2432,YELLO
W,BLACK);
    }else if(lcddev.width==480)
    {
        fsize=24;
        icowidth=36;
        okoffset=370;

    //app_show_mono_icos(5,ypos,icowidth,48,(u8*)APP_ALIENTEK_ICO3648,YELLO
W,BLACK);
    }

    LCD_ShowString(icowidth+5*2,ypos+fsize*j++,lcddev.width,lcddev.height,fsize,
"SmartHome System");
    LCD_ShowString(icowidth+5*2,ypos+fsize*j++,lcddev.width,lcddev.height,fsize,"Copy
right (C) 2018-2028");
    app_get_version(verbuf,HARDWARE_VERSION,2);
```

```
        strcpy((char*)version,"HARDWARE:");
        strcat((char*)version,(const char*)verbuf);
        strcat((char*)version,", SOFTWARE:");
        app_get_version(verbuf,SOFTWARE_VERSION,3);
        strcat((char*)version,(const char*)verbuf);
        LCD_ShowString(5,ypos+fsize*j++,lcddev.width,lcddev.height,fsize,version);
        sprintf((char*)verbuf,"LCD ID:%04X",lcddev.id);         //LCD ID 打印到 verbuf 里面
        LCD_ShowString(5,ypos+fsize*j++,lcddev.width,lcddev.height,fsize, verbuf);//  显  示
LCD ID
////////////////////////////////////////////////////////////////////////////
//开始硬件检测初始化
        WM8978_Init();              //防止喇叭乱叫
        app_wm8978_volset(0);   //关闭音量输出
        Light=1; //开灯
      Fan=1;       //开风扇
      Door=1;    //开门

        if(HC05_Get_Role()==1)
        {
            HC05_Set_Cmd("AT+ROLE=0");
            HC05_Set_Cmd("AT+RESET");   //复位 ATK-HC05 模块
            delay_ms(200);
        }



        LCD_ShowString(5,ypos+fsize*j++,lcddev.width,lcddev.height,fsize,
"CPU:STM32F407ZGT6 168Mhz");
        LCD_ShowString(5,ypos+fsize*j++,lcddev.width,lcddev.height,fsize,  "FLASH:1024KB
SRAM:192KB");
        if(system_exsram_test(5,ypos+fsize*j,fsize))system_error_show(5,ypos+fsize*j++,"EX
Memory Error!",fsize);
        LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize,"OK");

        my_mem_init(SRAMEX);            //初始化外部内存池,必须放在内存检测之后
        Light=0; //关灯
      Fan=0;       //关风扇
      Door=0;    //关门

        if(W25QXX_ReadID()!=W25Q128)//检测不到 W25Q128
        {
```

```
        system_error_show(5,ypos+fsize*j++,"Ex Flash Error!!",fsize);
    }else temp=16*1024;//16M 字节大小
    LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize,  "Ex  Flash:
KB");
    LCD_ShowxNum(5+9*(fsize/2),ypos+fsize*j,temp,5,fsize,0);//显示 flash 大小
    LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize, "OK");
    //检测是否需要擦除 SPI FLASH?
    res=KEY_Scan(1);//
    if(res==KEY2_PRES)
    {
        res=system_files_erase(5,ypos+fsize*j,fsize);
        if(res)goto REINIT;
    }
     //RTC 检测
    LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize, "RTC Check...");

    if(RTC_Init())system_error_show(5,ypos+fsize*(j+1),"RTC Error!",fsize);//RTC 检测
    else
    {
        calendar_get_time(&calendar);//得到当前时间
        calendar_get_date(&calendar);//得到当前日期
        LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize,
"OK");
    }
    //检查 SPI FLASH 的文件系统
    LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize,  "FATFS
Check...");//FATFS 检测
    f_mount(fs[0],"0:",1);          //挂载 SD 卡
    f_mount(fs[1],"1:",1);          //挂载挂载 FLASH.
    LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize, "OK");

    //SD 卡检测
    LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize,  "SD  Card:
MB");//FATFS 检测
    temp=0;
    do
    {
        temp++;
        res=exf_getfree("0:",&dtsize,&dfsize);//得到 SD 卡剩余容量和总容量
        delay_ms(200);
    }while(res&&temp<5);//连续检测 5 次
```

```
    if(res==0)//得到容量正常
    {
        gui_phy.memdevflag|=1<<0;  //设置 SD 卡在位.
        temp=dtsize>>10;//单位转换为 MB
        stastr="OK";
    }else
    {
        temp=0;//出错了,单位为 0
        stastr="ERROR";
    }
    LCD_ShowxNum(5+8*(fsize/2),ypos+fsize*j,temp,5,fsize,0);                    //  显
示 SD 卡容量大小
    LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize,stastr);
    //SD 卡状态
    //W25Q128 检测,如果不存在文件系统,则先创建.
    temp=0;
    do
    {
        temp++;
        res=exf_getfree("1:",&dtsize,&dfsize);//得到 FLASH 剩余容量和总容量
        delay_ms(200);
    }while(res&&temp<20);//连续检测 20 次
    if(res==0X0D)//文件系统不存在
    {
        LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize,    "Flash    Disk
Formatting...");  //格式化 FLASH
        res=f_mkfs("1:",1,4096);//格式化 FLASH,1,盘符;1,不需要引导区,8 个扇区为 1 个
簇
        if(res==0)
        {
            f_setlabel((const TCHAR *)"1:SmartHome");                   //设置  Flash  磁盘
的名字为：SmartHome
            LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize,
"OK");//标志格式化成功
            res=exf_getfree("1:",&dtsize,&dfsize);//重新获取容量
        }
    }
    if(res==0)//得到 FLASH 卡剩余容量和总容量
    {
        gui_phy.memdevflag|=1<<1;  //设置 SPI FLASH 在位.
        LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize,    "Flash    Disk:
```

```
KB");//FATFS 检测
            temp=dtsize;
        }else system_error_show(5,ypos+fsize*(j+1),"Flash Fat Error!",fsize);     //flash  文 件
系统错误
        LCD_ShowxNum(5+11*(fsize/2),ypos+fsize*j,temp,5,fsize,0);                          //
显示 FLASH 容量大小
        LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize,"OK");
            //FLASH 卡状态
        //U 盘检测
        usbapp_mode_set(0);                                                    //设置为 U 盘模式
        temp=0;
        while(usbx.hdevclass==0&&temp<1000)  //等待 U 盘被检测,最多等待 5 秒
        {
            usbapp_pulling();
            if((usbx.bDeviceState&(1<<6))==0&&temp>300)break;//1.5 秒钟之内,没有检测
到设备插入,则直接跳过,不再等待
            delay_ms(5);
            temp++;
        }
        if(usbx.hdevclass==1)//检测到了 U 盘
        {
            fs[2]->drv=2;
            f_mount(fs[2],"2:",1);                          //挂载挂载 U 盘
            res=exf_getfree("2:",&dtsize,&dfsize); //得到 U 盘剩余容量和总容量
        }else res=0XFF;
       LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize, "U Disk:        MB");
        //U 盘容量大小
        if(res==0)//得到容量正常
        {
            gui_phy.memdevflag|=1<<2;       //设置 U 盘在位.
            temp=dtsize>>10;//单位转换为 MB
            stastr="OK";
        }else
        {
            temp=0;//出错了,单位为 0
            stastr="ERROR";
        }
        LCD_ShowxNum(5+7*(fsize/2),ypos+fsize*j,temp,5,fsize,0);                  //  显
示 U 盘容量大小
        LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize,stastr);
        //U 盘状态
```

```
    //TPAD 检测
    LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize, "TPAD Check...");

    if(TPAD_Init(8))system_error_show(5,ypos+fsize*(j+1),"TPAD  Error!",fsize);//触摸按键检测
    else        LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize, "OK");
    //MPU6050 检测
    LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize,                "MPU6050 Check...");
    if(MPU_Init())system_error_show(5,ypos+fsize*(j+1),"MPU6050 Error!",fsize);//ADXL345 检测
    else        LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize, "OK");
    //24C02 检测
    LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize, "24C02 Check...");

    if(AT24CXX_Check())system_error_show(5,ypos+fsize*(j+1),"24C02 Error!",fsize);//24C02 检测
    else        LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize, "OK");
    //WM8978 检测
   LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize, "WM8978 Check...");

    if(WM8978_Init())system_error_show(5,ypos+fsize*(j+1),"WM8978 Error!",fsize);//WM8978 检测
    else
    {
        LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize, "OK");
        app_wm8978_volset(0);                   //关闭 WM8978 音量输出

    }
    //字库检测
    LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize, "Font Check...");
    res=KEY_Scan(1);//检测按键
    if(res==KEY1_PRES)//更新？确认
    {
        res=system_font_update_confirm(5,ypos+fsize*(j+1),fsize);
    }else res=0;
    if(font_init()||(res==1))//检测字体,如果字体不存在/强制更新,则更新字库
```

```
    {
        res=0;//按键无效
        if(update_font(5,ypos+fsize*j,fsize,"0:")!=0)//从 SD 卡更新
        {
            TIM3_Int_Init(100-1,8400-1);//启动 TIM3  轮询 USB,10ms 中断一次
            if(update_font(5,ypos+fsize*j,fsize,"2:")!=0)//从 U 盘更新
            {
                system_error_show(5,ypos+fsize*(j+1),"Font Error!",fsize); //字体错误
            }
            TIM3->CR1&=~(1<<0);//关闭定时器 3
        }
        LCD_Fill(5,ypos+fsize*j,lcddev.width,ypos+fsize*(j+1),BLACK);//填充底色
        LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize, "Font Check...");

    }
    LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize, "OK");//
字库检测 OK
    //系统文件检测
    LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize, "SYSTEM Files
Check...");
    while(app_system_file_check("1"))//系统文件检测
    {
        LCD_Fill(5,ypos+fsize*j,lcddev.width,ypos+fsize*(j+1),BLACK);        //填充底
色
        LCD_ShowString(5,ypos+fsize*j,(fsize/2)*8,fsize,fsize, "Updating");//    显    示
updating
        app_boot_cpdmsg_set(5,ypos+fsize*j,fsize);        //设置到坐标
        temp=0;
        TIM3_Int_Init(100-1,8400-1);                    //启动 TIM3  轮询 USB,10ms 中断
一次
        if(app_system_file_check("0"))                    //检查 SD 卡系统文件完整性
        {
            if(app_system_file_check("2"))res=9;        //标记为不可用的盘
            else res=2;                                    //标记为 U 盘
        }else res=0;                                    //标记为 SD 卡
        if(res==0||res==2)                                //完整了才更新
        {
            sprintf((char*)verbuf,"%d:",res);
            if(app_system_update(app_boot_cpdmsg,verbuf))//更新?
            {
                system_error_show(5,ypos+fsize*(j+1),"SYSTEM File Error!",fsize);
```

```
            }
        }
        TIM3->CR1&=~(1<<0);                                    //关闭定时器 3
        LCD_Fill(5,ypos+fsize*j,lcddev.width,ypos+fsize*(j+1),BLACK);//填充底色
        LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize, "SYSTEM Files Check...");
        if(app_system_file_check("1"))//更新了一次，再检测，如果还有不全，说明 SD 卡文件就不全!
        {
            system_error_show(5,ypos+fsize*(j+1),"SYSTEM File Lost!",fsize);
        }else break;
    }
    LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize, "OK");
    //触摸屏检测
    LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize, "Touch Check...");

    res=KEY_Scan(1);//检测按键
    if(TP_Init()||(res==KEY0_PRES&&(tp_dev.touchtype&0X80)==0))// 有更新/按下了 KEY0 且不是电容屏,执行校准
    {
        if(res==1)TP_Adjust();
        res=0;//按键无效
        goto REINIT;                    //重新开始初始化
    }
    LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize, "OK");//触摸屏检测 OK
    //系统参数加载
    LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize,          "SYSTEM Parameter Load...");
    if(app_system_parameter_init())system_error_show(5,ypos+fsize*(j+1),"Parameter Load Error!",fsize);//参数加载
    else      LCD_ShowString(5+okoffset,ypos+fsize*j++,lcddev.width,lcddev.height,fsize, "OK");
    LCD_ShowString(5,ypos+fsize*j,lcddev.width,lcddev.height,fsize,          "SYSTEM Starting...");
    myfree(SRAMIN,version);
    delay_ms(1500);
}
//main 函数
int main(void)
{
```

```
    system_init();          //系统初始化
    OSInit();
    OSTaskCreate(start_task,(void                                    *)0,(OS_STK
*)&START_TASK_STK[START_STK_SIZE-1],START_TASK_PRIO );//创建起始任务
    OSStart();
}
extern OS_EVENT * audiombox;  //音频播放任务邮箱
//开始任务
void start_task(void *pdata)
{
     OS_CPU_SR cpu_sr=0;
    pdata = pdata;
    OSStatInit();      //初始化统计任务.这里会延时 1 秒钟左右
    app_srand(OSTime);
    audiombox=OSMboxCreate((void*) 0);     //创建邮箱
    OS_ENTER_CRITICAL();//进入临界区(无法被中断打断)
    OSTaskCreate(main_task,(void
*)0,(OS_STK*)&MAIN_TASK_STK[MAIN_STK_SIZE-1],MAIN_TASK_PRIO);

    OSTaskCreate(usart_task,(void
*)0,(OS_STK*)&USART_TASK_STK[USART_STK_SIZE-1],USART_TASK_PRIO);

    OSTaskCreate(watch_task,(void
*)0,(OS_STK*)&WATCH_TASK_STK[WATCH_STK_SIZE-1],WATCH_TASK_PRIO);

    OSTaskSuspend(START_TASK_PRIO);    //挂起起始任务.
    OS_EXIT_CRITICAL(); //退出临界区(可以被中断打断)
}

//主任务
void main_task(void *pdata)
{
    //  语音模块
    u8 len=0;
    u8 rs485buf[5];
    u8 t=0;
    u8 res_bt=0;
    u8 selx;
    u16 tcnt=0;
    spb_init();            //初始化 SPB 界面
    spb_load_mui();        //加载 SPB 主界面
```

```
        slcd_frame_show(0); //显示界面

        while(1)
        {
    //-------------------------------------------------------------------------------
    //   语音模块
    //   语音控制开门/关门、开灯/关灯、开风扇/关风扇
            RS485_Receive_Data(rs485buf,&len);
            if(len)//接收到有数据，语音模块与 STM32 通信
            {
                if(len>5)len=5;//最大是 5 个数据.
                    switch(rs485buf[1])
                    {
                            case 1:face_yuyin_play();break;       //开门:01
                          case 2:Door=0;break;          //关门:02
                            case 3:Light=1;break;          //开灯:03
                            case 4:Light=0;break;      //关灯:04
                            case 5:Fan=1;break;              //开风扇:05
                            case 6:Fan=0;break;        //关风扇:06
                            case 9:
                            {
                                rs485buf[0]=0;
                                rs485buf[1]=0;
                                audio_stop_req(&audiodev);//请求停止播放

                            }

                                break;//关音乐
                            case 0xA:alarm.ringsta&=~(1<<7);break;//关闹钟
                            case 0xC:OPEN_Window();break;//开窗
                            case 0xD:CLOSE_Window();break;//关窗
                            case 0xE:       //睡眠模式
                            {
                                Door=0;
                                Light=0;
                                Fan=0;
                            };break;
                            case 0xF:       //省电模式
                            {
                                Door=0;
                                Light=0;
```

```
                    Fan=0;
                };break;
            }


        }
        selx=spb_move_chk();
        system_task_return=0;//清退出标志
        switch(selx)//发生了双击事件
        {

            case 8:face_manager_play();break;     //人脸管理
            case 9:audio_play();break;                  //查看语音
            case 10:ebook_play();break;                 //查看留言
            case 11:sysset_play();break;                //系统管理
            case 12:appplay_frec();break;               //人脸登记
         case 13:recorder_play();break;              //语音留言
            case 14:notepad_play();break;               //文本留言    原记事本
            case 15:
            {

    res_bt=window_msg_box((lcddev.width-200)/2,(lcddev.height-80)/2,200,80,"",(u8*)AP
P_BLUETOOTH_CAPTION_TBL[gui_phy.language],12,0,0X03,0);

                    if(res_bt==1)
                    {
                    flash_data_save[0]=0;   //记录保存的 Android_ID 个数，格式化为 0


    STMFLASH_Write(FLASH_DATA_SAVE,(u32*)flash_data_save,37);

    app_muti_remind_msg((lcddev.width-250)/2,(lcddev.height-200)/2,250,106,APP_REMI
ND_CAPTION_TBL[gui_phy.language],delethbt_remindmsg_tbl[gui_phy.language]);
                    res_bt=0;
                    }
                    else
                    {

    app_muti_remind_msg((lcddev.width-250)/2,(lcddev.height-200)/2,250,106,APP_REMI
ND_CAPTION_TBL[gui_phy.language],ndelethbt_remindmsg_tbl[gui_phy.language]);
                    res_bt=0;
```

```
                    }
                }
                break;                          //查看手机,蓝牙用户
                case 16:

    app_muti_remind_msg((lcddev.width-250)/2,(lcddev.height-200)/2,250,200,APP_REMI
ND_CAPTION_TBL[gui_phy.language],aboutus_remindmsg_tbl[gui_phy.language]);
                    break;                      //关于我们
                case 17:app_play();break;                   //智能家居 Home  查看说明
                case 18:calendar_play();break;                      //天气预报

            }

            RS485_Receive_Data(rs485buf,&len);
            if(len)//接收到有数据，语音模块与 STM32 通信
            {
                if(len>5)len=5;//最大是 5 个数据.
                    switch(rs485buf[1])
                    {
                            case 1:face_yuyin_play();break;       //开门:01
                        case 2:Door=0;break;          //关门:02
                            case 3:Light=1;break;          //开灯:03
                            case 4:Light=0;break;      //关灯:04
                            case 5:Fan=1;break;              //开风扇:05
                            case 6:Fan=0;break;         //关风扇:06
                            case 9:
                            {
                                rs485buf[0]=0;
                                rs485buf[1]=0;
                                audio_stop_req(&audiodev);//请求停止播放

                            }

                                break;//关音乐
                            case 0xA:alarm.ringsta&=~(1<<7);break;//关闹钟
                            case 0xC:OPEN_Window();break;//开窗
                            case 0xD:CLOSE_Window();break;//关窗
                            case 0xE:       //睡眠模式
                            {
                                Door=0;
                                Light=0;
```

```
                    Fan=0;
                };break;
                case 0xF:       //省电模式
                {
                    Door=0;
                    Light=0;
                    Fan=0;
                };break;
            }

        }


        if(selx!=0XFF)spb_load_mui();//显示主界面
        delay_ms(1000/OS_TICKS_PER_SEC);//延时一个时钟节拍
        tcnt++;
        if(tcnt==500)     //500 个节拍为 1 秒钟
        {
            tcnt=0;
            spb_stabar_msg_show(0);//更新状态栏信息，顶部信息
        }


    }
}
extern vu8 frec_running;
//执行最不需要时效性的代码
void usart_task(void *pdata)
{
    // 语音模块
    u8 len2=0;
    u8 rs485buf2[5];
    u8 t2=0;
    u16 alarmtimse=0;
    pdata=pdata;
    while(1)
    {
        delay_ms(100);
     RS485_Receive_Data(rs485buf2,&len2);
        if(len2)//接收到有数据，语音模块与 STM32 通信
        {
```

```
                if(len2>5)len2=5;//最大是 5 个数据.
                    if(rs485buf2[0]==0&&rs485buf2[1]==0X0A)
                    {
                            alarm.ringsta&=~(1<<7);//关闹钟
                 }

               }
               if(alarm.ringsta&1<<7)//执行闹钟扫描函数
               {
                    calendar_alarm_ring(alarm.ringsta&0x3);//闹铃
                    alarmtimse++;

                    if(alarmtimse>300)//超过 300 次了,5 分钟以上
                    {
                        alarm.ringsta&=~(1<<7);//关闭闹铃
                        alarmtimse=0;
                    }

               }else if(alarmtimse)
               {
                    alarmtimse=0;

               }
               if(sim900dev.mode==3)phone_ring();//蜂鸣器,来电提醒
               if(systemset.lcdbklight==0)app_lcd_auto_bklight(); //自动背光控制

     if(frec_running==0)printf("in:%d,ex:%d,ccm:%d\r\n",my_mem_perused(0),my_mem_p
erused(1),my_mem_perused(2));//打印内存占用率
     }
}

vu8 system_task_return;        //任务强制返回标志.
//监视任务
void watch_task(void *pdata)
{
     OS_CPU_SR cpu_sr=0;
    // CO 监测
    u8 COflag=0;
    // MPU6050
    float pitch,roll,yaw;
    short temp1,temp2,temp3;
```

```
    u8 tmpu6050=0;
    u8 dizhenflag=0;
  // 语音模块
    u8 len=0;
    u8 rs485buf[5];
    u8 t=0;
    //
    u8 rerreturn=0;
    u8 res;
    u8 key;
    pdata=pdata;


    while(1)
    {

//-------------------------------------------------------------------------------
//   语音模块
//   语音控制开门/关门、开灯/关灯、开风扇/关风扇
        RS485_Receive_Data(rs485buf,&len);
        if(len)//接收到有数据，语音模块与 STM32 通信
        {
            if(len>5)len=5;//最大是 5 个数据.
            //for(i=0;i<len;i++) LCD_ShowxNum(30+i*32,500,rs485buf[i],1,16,0X80);    //
显示数据

//              if(rs485buf[0]==0)
//              {
                    switch(rs485buf[1])
                    {

                            //case 1:face_yuyin_play();break;       //开门:01
                            case 2:Door=0;break;       //关门:02
                            case 3:Light=1;break;          //开灯:03
                            case 4:Light=0;break;      //关灯:04
                            case 5:Fan=1;break;              //开风扇:05
                            case 6:Fan=0;break;        //关风扇:06
                            case 9:
                            {
                                rs485buf[0]=0;
                                rs485buf[1]=0;
```

```
                    audio_stop_req(&audiodev);//请求停止播放

                }

                    break;//关音乐
                case 0xA:alarm.ringsta&=~(1<<7);break;//关闹钟
                case 0xC:OPEN_Window();break;//开窗
                case 0xD:CLOSE_Window();break;//关窗
                case 0xE:       //睡眠模式
                {
                    Door=0;
                    Light=0;
                    Fan=0;
                };break;
                case 0xF:       //省电模式
                {
                    Door=0;
                    Light=0;
                    Fan=0;
                };break;

            }

        if(Door==0)
            {
                LCD_Fill(250,5,350,25,BLACK);     //清除显示
                POINT_COLOR=RED;
                LCD_ShowString(250,0,200,16,16,"Door Close!");

            }
            else
            {
                LCD_Fill(250,5,350,25,BLACK);     //清除显示
                POINT_COLOR=GREEN;
                LCD_ShowString(250,0,200,16,16,"Door Open!");
            }
//      }

        }
```

```
//-------------------------------------------------------------------------------------------
//    一氧化碳监测
      if(COD==0)
          {
              BEEP=0; //蜂鸣器报警
              Show_Str(350,5,80,16,"CO 超标",16,0);
              COflag=1;
          }
          else
          {
              if(COflag==1)
                          {
                              BEEP=1;     //解除报警
                              LCD_Fill(350,5,450,25,BLACK); //清除显示
                              COflag=0;
                          }
          }



//-------------------------------------------------------------------------------------------
//    蓝牙控制
          if(USART3_RX_STA&0X8000)            //蓝牙接收到一次数据了，手机与
STM32 通信
          {
              // 蓝牙通信部分参数
              u8 i,j,temp;
              u8 time=0;
              u8 flag=1;
              u8 reclen=0;
              u8 num;
              LCD_Fill(100,550,400,240,WHITE);  //清除显示
              reclen=USART3_RX_STA&0X7FFF; //得到数据长度
              USART3_RX_BUF[reclen]=0;          //加入结束符
              //LCD_Fill(100,550,500,580,WHITE);//清除显示
              //LCD_ShowString(100,550,300,16,16,USART3_RX_BUF);//显示接收到的数
据
              USART3_RX_STA=0;
```

```
// 登记 root 用户
// root 用户 Android ID                                    登记 root 用户，只在首次使用或者格式化后使用有效，否则无效
            if(reclen==16)        //登记 root 用户
            {
                //   添加增加人脸模板功能



                STMFLASH_Read(FLASH_DATA_SAVE,(u32*)flash_data_save,37);
                num=flash_data_save[0];
          //LCD_ShowxNum(130,600,num,4,16,0);
          if(num==0)
                {
                        flash_data_save[0]=1;   //记录保存的 Android_ID 个数，初始化为 1
                        num=flash_data_save[0];
                  //LCD_ShowxNum(130,600,num,4,16,0);
                        //flash_data_save[148]
                        for(j=0;j<9;j++)     //初始化时，全部 id 设置为 root 用户的 id
                        {
                            for(i=0;i<16;i++)
                                { //初始化 ROOT
                                    temp=find_id[j]+i;
                                    flash_data_save[temp]=USART3_RX_BUF[i];

                                }
                        }
                        //LCD_ShowString(30,500,500,16,16,"Message:Start          Write FLASH...."); //提示

     STMFLASH_Write(FLASH_DATA_SAVE,(u32*)flash_data_save,37);
                        //LCD_ShowString(30,500,500,16,16,"Message:Initial  root  user Finished!"); //提示
                        //u3_printf("%s\r\n","You are root user!"); //发送指令
                }
          else
                {
                  //u3_printf("%s\r\n","Error:Root user has existed!"); //发送指令
                    //LCD_ShowString(30,500,500,16,16,"Error:Root          user          has existed,could not change!");   //提示
```

```
            }

        }

// 开门指令-----Door
// 'D'+root 用户 Android ID+'0'                PF10 置 0，LED1 亮，关门
// 'D'+root 用户 Android ID+'1'                PF10 置 1，LED1 暗，开门
// 'D'+普通用户 Android ID+'0'                 PF10 置 0，LED1 亮，关门
// 'D'+普通用户 Android ID+'1'                 PF10 置 1，LED1 暗，开门

// 开灯指令-----Light
// 'L'+root 用户 Android ID+'0'                PF7 置 0，关灯
// 'L'+root 用户 Android ID+'1'                PF7 置 1，开灯
// 'L'+普通用户 Android ID+'0'                 PF7 置 0，关灯
// 'L'+普通用户 Android ID+'1'                 PF7 置 1，开灯

// 风扇指令-----Fan
// 'F'+root 用户 Android ID+'0'                PF5 置 0，关风扇
// 'F'+root 用户 Android ID+'1'                PF5 置 1，开风扇
// 'F'+普通用户 Android ID+'0'                 PF5 置 0，关风扇
// 'F'+普通用户 Android ID+'1'                 PF5 置 1，开风扇
```

```
if(reclen==18)     //指令长度为 18
 {
    STMFLASH_Read(FLASH_DATA_SAVE,(u32*)flash_data_save,37);
    num=flash_data_save[0];
     if(USART3_RX_BUF[0]==68)     //D:68   开门指令-----Door
    {
        for(j=0;j<num;j++)
        {
            for(i=0;i<16;i++)
                {
```

```
if(flash_data_save[i+find_id[j]]!=USART3_RX_BUF[i+1]) break;
                                        else if(i==15)
                                                {
                                                    //LCD_Fill(30,500,400,550,WHITE); //
清除显示
                                                    if(USART3_RX_BUF[17]==48)
                                                    {
                                                        Door=0; //关门,0:48
                                                        //u3_printf("%s\r\n","Door
Closed!"); //发送指令

    //LCD_ShowString(30,500,500,16,16,"Message:Door Closed!!");
                                                    }
                                                    if(USART3_RX_BUF[17]==49)
                                                    {
                                                        Door=1; //开门,1:49
                                                        //u3_printf("%s\r\n","Door
Opened!"); //发送指令

    //LCD_ShowString(30,500,500,16,16,"Message:Door Opened!");
                                                    }
                                                }
                                    }
                                }


                            }

                             if(USART3_RX_BUF[0]==76)     //L:76    开灯指令-----Light
                                {
                                for(j=0;j<num;j++)
                                    {
                                        for(i=0;i<16;i++)
                                            {

if(flash_data_save[i+find_id[j]]!=USART3_RX_BUF[i+1]) break;
                                                    else if(i==15)
                                                            {

//LCD_Fill(30,500,400,550,WHITE);  //清除显示
```

```
    if(USART3_RX_BUF[17]==48)
                                                    {
                                                        Light=0;      //PF7 置 0，
关灯
                                                        //u3_printf("%s\r\n","Light
Closed!"); //发送指令

    //LCD_ShowString(30,500,500,16,16,"Message:Light Closed!!");
                                                    }

    if(USART3_RX_BUF[17]==49)
                                                    {
                                                        Light=1;      //PF7 置 1，
开灯
                                                        //u3_printf("%s\r\n","Light
Opened!"); //发送指令

    //LCD_ShowString(30,500,500,16,16,"Message:Light Opened!");
                                                    }

                                                }
                                            }
                                        }
                                    }
                                if(USART3_RX_BUF[0]==70)     //F:70   风扇指令-----Fan
                                    {
                                    for(j=0;j<num;j++)
                                        {
                                            for(i=0;i<16;i++)
                                                {

if(flash_data_save[i+find_id[j]]!=USART3_RX_BUF[i+1]) break;
                                                    else if(i==15)
                                                        {

//LCD_Fill(30,500,400,550,WHITE);  //清除显示

    if(USART3_RX_BUF[17]==48)
                                                    {
                                                        Fan=0;      //PF5 置 0，关
风扇
```

```
                                                    //u3_printf("%s\r\n","Fan
Closed!"); //发送指令

    //LCD_ShowString(30,500,500,16,16,"Message:Fan Closed!!");
                                                    }

    if(USART3_RX_BUF[17]==49)
                                                    {
                                                        Fan=1;      //PF5 置 1，开
风扇
                                                    //u3_printf("%s\r\n","Fan
Opened!"); //发送指令

    //LCD_ShowString(30,500,500,16,16,"Message:Fan Opened!");
                                                    }
                                                }
                                            }
                                        }
                                    }

            }

// 查询所有用户
// 'U'+root 用户 Android ID                    返回所有用户信息
// 'U'+普通用户 Android ID                     返回个人用户信息

// 查询开门记录
// 'J'+root 用户 Android ID                    返回所有用户开门记录
// 'J'+普通用户 Android ID                     返回个人用户开门记录
            if(reclen==17)     //指令长度为 17
             {
                STMFLASH_Read(FLASH_DATA_SAVE,(u32*)flash_data_save,37);
                num=flash_data_save[0];
                 if(USART3_RX_BUF[0]==85)     //U:85
                {
                    //LCD_ShowString(30,400,500,16,16,"USART3_RX_BUF[0]==85");

                    for(j=0;j<num;j++)
                    {
                        for(i=0;i<16;i++)
                            {
```

```
if(flash_data_save[i+find_id[j]]!=USART3_RX_BUF[i+1]) break;
                                        else if(i==15)
                                                {
                                                    //if(j==0)    //root 用户
                                                    //{

//LCD_ShowString(30,400,500,16,16,"Root chaxun test!");

//u3_printf("%s\r\n",(u32*)flash_data_save); //发送指令



                                                    //}

                                                }
                                    }
                                }
                            }

                             if(USART3_RX_BUF[0]==74)      //J:74
                                {
                                for(j=0;j<num;j++)
                                    {
                                        for(i=0;i<16;i++)
                                            {

if(flash_data_save[i+find_id[j]]!=USART3_RX_BUF[i+1]) break;
                                                else if(i==15)
                                                    {
                                                        if(j==0)    //root 用户
                                                        {
                                                        }

                                                    }
                                            }
                                    }

                        }
```

```
// 注册用户
// root 用户 Android ID+'T'+普通用户 Android ID 添加普通用户
// root 用户 Android ID+'S'+普通用户 Android ID 删除普通用户
                if(reclen==33)     //添加或删除用户
                {
                    STMFLASH_Read(FLASH_DATA_SAVE,(u32*)flash_data_save,37);
                    num=flash_data_save[0];
            LCD_ShowxNum(130,600,num,4,16,0);
            //判断是否为 root 用户操作
                    for(i=0;i<16;i++)
                      {
                        if(!((USART3_RX_BUF[i]==flash_data_save[i+find_id[0]])))
                          {
                              //LCD_ShowString(30,500,500,16,16,"Error:Sorry,you       are
mot my manager!");
                               flag=0;
                          }
                      }
                    //是 root 用户
                      if(flag)
                            { //root 用户存入一个新的 common id
                              //LCD_ShowString(30,500,500,25,25,"Message:HELLO,my
manager!");
                              if(USART3_RX_BUF[16]=='T')     //添加用户
                              {
                                    if(num<8)
                                    {
                                          for(i=0;i<16;i++)
                                                {

flash_data_save[i+find_id[num]]=USART3_RX_BUF[17+i];
                                                }
                                          num=num+1;     //普通用户数目增加 1
                                          flash_data_save[0]=num;
                                          num=flash_data_save[0];
                                          LCD_ShowxNum(130,600,num,4,16,0);

    STMFLASH_Write(FLASH_DATA_SAVE,(u32*)flash_data_save,37);

    //LCD_ShowString(30,500,500,16,16,"Message:Add  User  Successfully!");//提示传送
```

```
完成
                                        }
                                        else
                                        {
                                            LCD_ShowString(30,500,500,25,25,"Error:User
registered full!");
                                        }

                                    }
                                    if(USART3_RX_BUF[16]=='S')     //删除用户
                                    {
                                        u8 delete_temp=0;
                                        if(num>1)
                                        {
                                            for(j=num;j>=1;j--)
                                            {
                                                for(i=0;i<16;i++)
                                                    {

if(flash_data_save[i+find_id[j]]!=USART3_RX_BUF[17+i]) break;
                                                        if(i==15) delete_temp=j;
                                                    }
                                                if(delete_temp>0) break;
                                            };
                                            if(delete_temp>0)
                                            {
                                                for(i=0;i<16;i++)      //将第 num-1 个
common 用户的 id 移到被删除用户的 id 那里
                                                    {

flash_data_save[i+find_id[delete_temp]]=flash_data_save[i+find_id[num]];
                                                    }
                                                for(i=0;i<16;i++)
                                                    {

flash_data_save[i+find_id[num]]=flash_data_save[i+find_id[0]]; //用 root 用户 id 覆盖最后
的位置
                                                    }
                                                num=num-1;     //普通用户数目减少 1
                                                flash_data_save[0]=num;
                                                num=flash_data_save[0];
```

```
                                                LCD_ShowxNum(130,600,num,4,16,0);

        STMFLASH_Write(FLASH_DATA_SAVE,(u32*)flash_data_save,37);

        //LCD_ShowString(30,500,200,16,16,"Delete User Finished!");//提示传送完成


                                                }
                                                else
                                                {

        //LCD_ShowString(30,500,200,16,16,"Error:No this user!");//没有要删除的用户
                                                }
                                        }
                                        else
                                        {
                                            //LCD_ShowString(30,500,210,25,25,"Error:No
common user!"); //没有普通用户

                                        }
                                    }
                                }
                }

                //   显示屏提示开关门
                if(Door==0)
                    {
                        LCD_Fill(250,5,350,25,BLACK);     //清除显示
                        POINT_COLOR=RED;
                        LCD_ShowString(250,0,200,16,16,"Door Close!");
                    }
                    else
                    {
                        LCD_Fill(250,5,350,25,BLACK);     //清除显示
                        POINT_COLOR=GREEN;
                        LCD_ShowString(250,0,200,16,16,"Door Open!");
                    }
            }
```

```
//-------------------------------------------------------------------------------------
//    MPU6050 地震监测
        if(tmpu6050==5)
        {
            tmpu6050=0;
            if((audiodev.status&(1<<7))==0)        //没有在放歌
            {

                if(mpu_dmp_get_data(&pitch,&roll,&yaw)==0)//读取 DMP 数据
                {
                    temp1=pitch*10;
                    temp2=roll*10;
                    temp3=yaw*10;
                    if(temp1<-40.0||temp1>40.0)   //小于-4 度或者大于 4 度
                    {
                        BEEP=0;     //蜂鸣器报警
                        Show_Str(350,5,50,16,"地震",16,0);
                        dizhenflag=1;
                    }
                    else if(temp2<-40.0||temp2>40.0)   //小于-4 度或者大于 4 度
                    {
                        BEEP=0;     //蜂鸣器报警
                        Show_Str(350,5,50,16,"地震",16,0);
                        dizhenflag=1;
                    }
                    else if(temp3<-40.0||temp3>40.0)   //小于-4 度或者大于 4 度
                    {
                        BEEP=0;     //蜂鸣器报警
                        Show_Str(350,5,50,16,"地震",16,0);
                        dizhenflag=1;
                    }
                    else
                    {
                        if(dizhenflag==1)
                        {
                            BEEP=1;     //解除报警
                            LCD_Fill(350,5,450,25,BLACK); //清除显示
                            dizhenflag=0;
                        }
```

```
                    }
                }
                delay_ms(10);
            }
        }
        tmpu6050=tmpu6050+1;

//  闹钟处理
        if(alarm.ringsta&(1<<7))//闹钟在执行
        {
            calendar_alarm_msg((lcddev.width-200)/2,(lcddev.height-160)/2);//闹钟处理
        }
//  gif 解码
        if(gifdecoding)//gif 正在解码中
        {
            key=pic_tp_scan();
            if(key==1||key==3)gifdecoding=0;//停止 GIF 解码
        }

//  TPAD 扫描
        if(rerreturn)//再次开始 TPAD 扫描时间减一
        {
            rerreturn--;
            delay_ms(15);//补充延时差
        }else if(TPAD_Scan(0))      //TPAD 按下了一次,此函数执行,至少需要 15ms.
        {
            rerreturn=10;           //下次必须 100ms 以后才能再次进入
            system_task_return=1;
            if(gifdecoding)gifdecoding=0;//不再播放 gif
        }
//  SD 卡检测
        if((t%60)==0)//900ms 左右检测 1 次
        {
            //SD 卡在位检测
            if((DCMI->CR&0X01)==0)//摄像头不工作的时候,才可以查询 SD 卡
            {
                OS_ENTER_CRITICAL();//进入临界区(无法被中断打断)
                res=SD_GetState();  //查询 SD 卡状态
                OS_EXIT_CRITICAL(); //退出临界区(可以被中断打断)
                if(res==0XFF)
                {
```

```
                    gui_phy.memdevflag&=~(1<<0);//标记 SD 卡不在位
                    OS_ENTER_CRITICAL();//进入临界区(无法被中断打断)
                    SD_Init();              //重新检测 SD 卡
                    OS_EXIT_CRITICAL(); //退出临界区(可以被中断打断)
                }else if((gui_phy.memdevflag&(1<<0))==0)//SD 不在位?
                {
                    f_mount(fs[0],"0:",1);          //重新挂载 sd 卡
                    gui_phy.memdevflag|=1<<0;  //标记 SD 卡在位了
                }
            }
        }
        delay_ms(10);
    }
}
//硬件错误处理
void HardFault_Handler(void)
{
    u32 i;
    u8 t=0;
    u32 temp;
    temp=SCB->CFSR;                 //fault 状态寄存器(@0XE000ED28)包括:MMSR,BFSR,UFSR
    printf("CFSR:%8X\r\n",temp);    //显示错误值
    temp=SCB->HFSR;                 //硬件 fault 状态寄存器
    printf("HFSR:%8X\r\n",temp);    //显示错误值
    temp=SCB->DFSR;                 //调试 fault 状态寄存器
    printf("DFSR:%8X\r\n",temp);    //显示错误值
    temp=SCB->AFSR;                 //辅助 fault 状态寄存器
    printf("AFSR:%8X\r\n",temp);    //显示错误值
    while(t<5)
    {
        t++;
        for(i=0;i<0X1FFFFF;i++);
    }

}
```

参考文献

[1] 苏红艳.建筑家族中的新成员——智能型建筑[J].山西建筑,2004(20):11-12.
[2] 文翔. 基于物联网的智能家居远程监控子系统软件设计[D].西安电子科技大学,2014.
[3] Wei Li,Bai Hui Cui,Fa Wei Zhang,Xing Guo. A Smart Home System Based on Speech Recognition Technology[J]. Applied Mechanics and Materials,2015,3744(713).
[4] 张娉. 智能家居产品情感化设计[D].昆明理工大学,2014.
[5] 李元元.基于 Android 平台的智能家居安防系统设计[J].制造自动化 2012,34(12):138-140.
[6] 杨龙山,王丽芳.屏蔽双绞线在车用 CAN 总线中的抗干扰能力研究[J].汽车技术,2006(12):9-12.
[7] 李要伟.射频技术在物联网中的应用[J].物联网技术,2011,1(01):49-51.
[8] Kaveh Pahlavan,Xinrong Li,Mika Ylianttila,Matti Latva-aho. Wireless Communication Technologies: New Multimedia Systems[M].Springer US:2002-06-15.
[9] Mitsugu Terada. Application of ZigBee sensor network to data acquisition and monitoring[J]. Measurement Science Review,2009,9(6).
[10] 申华.智能家居发展现状及前景[J].信息与电脑(理论版),2017(03):153-154.
[11] 朱敏玲,李宁.智能家居发展现状及未来浅析[J].电视技术,2015,39(04):82-85+96.