****

电子综合课设

项目报告书

**题目：一种基于STM32和Android的**

**智能开门系统**

**学 院 电子与信息学院**

**学生姓名 陈艺荣201530301043**

**周泽鑫201530431528**

**何晨晖201561301128**

**吴子莹201530572429**

**陈靖康201530311059**

**指导教师 徐向民、舒琳**

[一、选题背景 1](#_Toc11417)

[二、方案论证 2](#_Toc213)

[2.1 设计任务与功能介绍 2](#_Toc6807)

[2.2 方案论证 3](#_Toc27744)

[2.2.1 主控芯片方案 3](#_Toc12937)

[2.2.2 无线传输方案 3](#_Toc9072)

[三、系统硬设计及实现 4](#_Toc25798)

[3.1 硬件总体设计 4](#_Toc2580)

[3.2 STM32F407模块硬件设计 5](#_Toc11752)

[3.2.1 STM32F407简介 5](#_Toc22558)

[3.2.2 STM32F407硬件设计图 6](#_Toc26007)

[3.2.3 STM32F407硬件实物图 9](#_Toc29003)

[3.3 HC-05蓝牙模块硬件设计 10](#_Toc26758)

[3.3.1 HC-05蓝牙方案简介 10](#_Toc20418)

[3.3.2 HC-05蓝牙模块电路设计 11](#_Toc8053)

[3.3.3 HC-05蓝牙模块实物图 13](#_Toc7062)

[3.4 遥控器通信模块 13](#_Toc13115)

[3.5 门锁控制模块 14](#_Toc7967)

[3.5 整机电路 14](#_Toc32293)

[四、系统软件设计及实现 15](#_Toc1906)

[4.1 Android APP设计 15](#_Toc3361)

[4.1.1开发环境介绍 15](#_Toc6462)

[4.1.2 Android APP界面设计 15](#_Toc22429)

[4.1.3 Android APP蓝牙通信实现 16](#_Toc1485)

[4.2 通信协议简介 26](#_Toc19200)

[4.3 STM32驱动代码 27](#_Toc10511)

[4.3.1 固件开发介绍 27](#_Toc28442)

[4.3.2 按键处理 27](#_Toc198)

[4.3.3 登记root用户 29](#_Toc21258)

[4.3.4 开门或关门 30](#_Toc28242)

[4.3.5 添加或删除用户 31](#_Toc11885)

[4.3.6 扩展功能语音控制开门 33](#_Toc8117)

[五、项目成果 34](#_Toc16585)

[5.1 完整的软件设计 34](#_Toc8632)

[5.2 产品外观设计 34](#_Toc2453)

[5.3 专利申请 34](#_Toc27755)

[5.4 产品应用 35](#_Toc1365)

[六、项目收获与感悟 35](#_Toc1094)

[七、附件 35](#_Toc16531)

[附件1 支出情况 35](#_Toc23995)

[附件2 同类产品比较 36](#_Toc22474)

[附件3 市场调研 36](#_Toc24059)

# 一、选题背景

锁的历史很长，从草绳、门闩、三簧锁、叶片锁、套筒转心锁、弹子锁到现在的智能锁，无论前世今生，安全与便捷都是锁不变的使命。随着社会的不断发展以及生活水平的不断提高，人们对于家居安全的需求越来越高，普通的机械锁对于防范现代化的入室盗窃显得越来越无能为力。而且，安全措施的强度直接体现的是个人的身份和地位。智能锁使用新一代的门禁识别技术（包括蓝牙、密码、指纹、虹膜、人脸识别、语音识别等）大大改善家居安全的同时新的问题也开始显现。门锁在遭到攻击时，用户却无从知晓，直接导致各种盗窃案件居高不下。

让家居更智能更安全是现在很多人的一种生活体验追求，好的智能产品往往具备以下特点：安全、节能、经济实惠、便捷操作。智能锁作为智能家居发展的重要代表，它的各方面性能一直是消费者和制造商关注的重点。智能锁与传统机械锁的不同在于将开锁方式非机械化，智能化、多功能化，以提高用户的安全保障和使用便捷性，智能锁的发展大体可以分为两个阶段，前期是数字化开锁方式，如刷卡，密码，指纹，人脸，虹膜，声纹，静脉等等，随后是智能化开锁方式，让锁变得聪明，如利用大数据监测学习使用者的开锁习惯和生活特点，能够预先识别异常状况进行预警等，让锁像个看管员一样守护门。智能锁在设计过程中常保留原有的机械开锁方式作为保障，拓展多种非机械开锁方式,如远程蓝牙、指纹识别、RFID及智能卡感应、智控close键、zigbee通信、WIFI通信、GPRS通信、电子密码、人脸识别、语音识别等等。

但是，目前的多数智能锁价格过高，例如三星（SAMSUNG）SHS-P718 指纹电子密码锁指纹+密码+刷卡，售价为2649元，并不能得到很好普及，所以现今家庭、校园宿舍仍采样传统的普通机械锁，见图1(a)。根据《中国智能锁产业白皮书(2017版)》，我国智能锁市场占有率不到3%，见图1(b)，我国97%的家庭或机构都存在从机械锁转换为智能锁的需求。根据行业统计机构的预估数据显示，未来5-10年内，我国智能门锁的总需求量将会超过3000万套，行业总值也将突破1000亿元的大关。在乐观的情况下进行估计，到2020年我国智能门锁的总需求量将会达到3200万左右。

因此，研发一种安全而且便宜的智能锁对推广智能锁，促进家居智能化，具有重要的意义。以往的智能锁解决方案大都是摒弃传统门锁。在使用智能锁的时候需要拆除现有的门锁，操作麻烦，使得智能锁迟迟不能普及。基于现有机械锁存在的问题以及蓝牙通信技术具有的优势，本项目提出一种基于STM32和Android的智能开门系统，利用现有机械锁与连接在机械锁上的电磁推拉式长行程装置，通过蓝牙通信技术，实现远程开锁。

|  |  |
| --- | --- |
| IMG_20180402_144911 | 中国智能锁 |
| 1. 当前常用机械锁(以华工宿舍为例) | 1. 2017年各国智能锁市场占有率 |

图 1 我国智能锁发展概况

与此同时，考虑到蓝牙配对密码可能被破解，我们提出了更为安全的开锁方案：开锁APP需要密码验证(可选择不用密码进入APP)；一把锁默认绑定一台手机(root用户)；其他用户手机需要提供手机标设备唯一标识码给root用户设置权限，这一步通过两步手机互联开锁APP实现；所有用户手机每次开锁时都会动态更新开锁密码。实现更加安全的远程开锁，同时结合语音识别技术为辅，设计出一款可商业化的基于STM32和Android的智能开门系统，为下一步进行产业化和市场普及提供充足的理论与实验基础。

# 二、方案论证

## 2.1 设计任务与功能介绍

1、设计遥控器控制模块；

2、熟悉蓝牙通信协议，设计蓝牙通信模块；

3、设计Android APP，实现手机间通信以及手机和门锁间的通信；

4、利用语音控制模块实现语音控制；

5、编程实现STM32的FLASH存储以及蓝牙通信和I/O控制。

## 2.2 方案论证

### 2.2.1 主控芯片方案

**方案一：C51单片机**

一个[全双工](https://baike.so.com/doc/6893019-7110564.html" \t "https://baike.so.com/doc/_blank)[串行通信](https://baike.so.com/doc/6583743-6797513.html" \t "https://baike.so.com/doc/_blank)口，外部数据[存储器](https://baike.so.com/doc/4224899-4426539.html" \t "https://baike.so.com/doc/_blank)[寻址空间](https://baike.so.com/doc/1640137-1733703.html" \t "https://baike.so.com/doc/_blank)为64kB，外部[程序存储器](https://baike.so.com/doc/4616537-4828902.html" \t "https://baike.so.com/doc/_blank)寻址空间为64kB，逻辑操作[位寻址](https://baike.so.com/doc/6882655-7100145.html" \t "https://baike.so.com/doc/_blank)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](https://baike.so.com/doc/6385713-6599367.html" \t "https://baike.so.com/doc/_blank)和部分外部[特殊功能寄存器](https://baike.so.com/doc/2409845-2547808.html" \t "https://baike.so.com/doc/_blank)；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](https://baike.so.com/doc/1064435-1126132.html" \t "https://baike.so.com/doc/_blank)的中断[控制系统](https://baike.so.com/doc/6291733-6505239.html" \t "https://baike.so.com/doc/_blank)；一个[全双工](https://baike.so.com/doc/6893019-7110564.html" \t "https://baike.so.com/doc/_blank)[UART](https://baike.so.com/doc/2255582-2386423.html" \t "https://baike.so.com/doc/_blank)(通用异步接收发送器)的[串行](https://baike.so.com/doc/6583712-6797482.html" \t "https://baike.so.com/doc/_blank)I/O口，用于实现单片机之间或[单片机](https://baike.so.com/doc/30473-31769.html" \t "https://baike.so.com/doc/_blank)与微机之间的[串行通信](https://baike.so.com/doc/6583743-6797513.html" \t "https://baike.so.com/doc/_blank)；片内振荡器和[时钟](https://baike.so.com/doc/5407848-7598712.html" \t "https://baike.so.com/doc/_blank)产生电路，[石英晶体](https://baike.so.com/doc/5717540-5930266.html" \t "https://baike.so.com/doc/_blank)和[微调电容](https://baike.so.com/doc/1949822-2063265.html" \t "https://baike.so.com/doc/_blank)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。

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**。**

### 2.2.2 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](http://baike.baidu.com/view/2696.htm" \t "http://baike.baidu.com/_blank)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[IEEE](http://baike.baidu.com/view/3933.htm" \t "http://baike.baidu.com/_blank)802.11b.g.n协议栈以及[TCP/IP](http://baike.baidu.com/view/7729.htm" \t "http://baike.baidu.com/_blank)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](http://baike.baidu.com/view/1066191.htm" \t "http://baike.baidu.com/_blank)于1994年创制，当时是作为[RS232](http://baike.baidu.com/view/196461.htm" \t "http://baike.baidu.com/_blank)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开门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。

# 三、系统硬设计及实现

## 3.1 硬件总体设计

本设计分为四大模块，分别为遥控器通信模块、蓝牙HC-05通信模块、STM32主控模块、电磁铁控制模块。系统框图如图 2。

图 2 智能门禁系统硬件系统

**智能门禁系统硬件系统**

**STM32F407主控模块**

**蓝牙HC-05模块**

**遥控器通信模块**

**门锁控制模块**

**手机**

使用者通过手机或者遥控器控制开门或关门，控制信息通过蓝牙或者遥控模块发送给STM32，STM32根据接收到的信息作出相应处理。

## 3.2 STM32F407模块硬件设计

### 3.2.1 STM32F407简介

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。

### 3.2.2 STM32F407硬件设计图

STM32单片机的最小系统包括：电源电路，复位电路，时钟电路，BOOT电路，SWD调试电路。

①STM32电源电路

由于系统中采用的是单电源设计，模拟电源和数字电源用相同的电源会相互干扰，因此在设计中必须要采用模拟电源和数字电源隔离的做法。STM32电源电路分为数字电源部分和模拟电源部分，其中数字电源部分主要是提供电源给STM32内部数字外设如IO口，SPI等外设，而模拟电源主要是提供电源给STM32内部模拟外设如ADC，系统中利用电感隔离数字电源与模拟电源，如图 3所示。

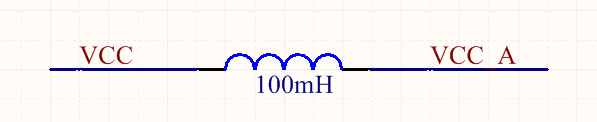


图 3 数字电源与模拟电源隔离电路

②STM32复位电路

为确保微机系统中电路稳定可靠工作，需要设计复位电路，就是利用它把电路恢复到起始状态，复位电路具有三个要求：在给电路通电时马上进行复位操作；在必要时可以由手动操作；根据程序或者电路运行的需要自动地进行。常见的复位方式有：手动按钮复位，上电复位。本系统的复位电路是上电复位，工作供电电源为4.75～5.25 V，STM32上电复位电路如图 4所示。当主板上电时，对电容充电，NRST引脚保持低电平一定时间，低电平持续的时间通过电阻值为1 k和电容值0.1 μF控制，进而实现STM32复位。

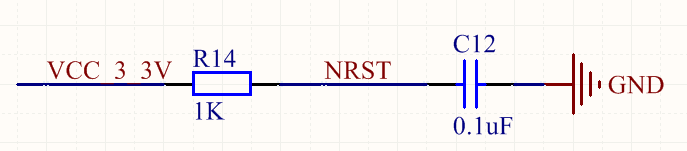


图 4 STM32上电复位电路

③STM32时钟电路

本系统利用单片机晶振提供基本的时钟信号，且一般系统共用一个晶振保持各部分同步。但是有些通信系统的基频和射频使用不同的晶振，也可以通过电子调整频率的方法保持同步。STM32的时钟由高速时钟和低速时钟构成，其中STM32内部自带高速RC和低速RC振荡器。但是内部RC时钟的不稳定将会影响STM32的内部外设，因此在条件允许的情况下，尽量使用外部时钟。在设计中我们将高速时钟使用外部晶振震荡电路。如图 5所示为STM32的时钟电路。

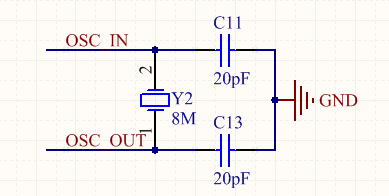


图 5 STM32时钟电路

④STM32 BOOT电路

STM32芯片内置三种启动模式：用户闪存（芯片内置的Flash）；SRAM（芯片内置的RAM区，即内存）；系统存储器（芯片内的ROM区）。在每个STM32的芯片上都有两个管脚BOOT0和BOOT1，这两个管脚在芯片复位时的电平状态决定了芯片复位后从哪个区域开始执行程序，如图 6所示为STM32的BOOT电路。用户闪存启动，正常的工作模式：BOOT1=x，BOOT0=0；厂家设置的系统存储器启动模式：BOOT1=0，BOOT0=1；内置SRAM启动的调试模式：BOOT1=1，BOOT0=1。一般不使用内置SRAM启动，因为SRAM掉电后数据就丢失。多数情况下SRAM只是在调试时使用，也可用于故障的局部诊断，写一段小程序加载到SRAM中诊断板上的其他电路，或用此方法读写板上的Flash或EEPRO等。

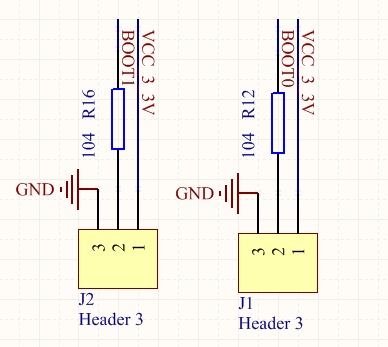


图 6 STM32 BOOT电路

⑤STM32 JTAG调试接口

JTAG（Joint Test Action Group，联合测试行动小组）是一种国际标准测试协议，主要用于芯片内部测试及对系统进行仿真、调试。目前大多数比较复杂的器件如ARM、DSP、FPGA等都含有支持JTAG协议的模块。处理器上标准的JTAG接口是4线：TMS、TCK、TDI、TDO，分别为测试模式选择、测试时钟、测试数据输入和测试数据输出。目前JTAG接口的连接有两种标准，即14针接口和20针接口。这里，我采用的是标准的JTAG接法，开发板上的处理器一般都采用标准的4线JTAG接口，即包含TMS、TCK、TDI、TDO接口（引脚）。还可能包含nTRST（测试系统复位信号）接口。STM32开发板板载的标准20针JTAG/SWD接口如图图 7所示。

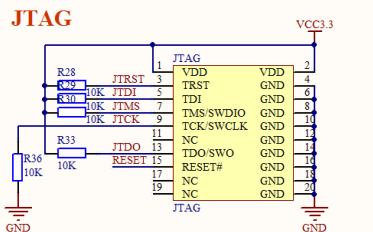


图 7 STM32 JTAG调试接口

若要用J-link工具来实现对开发板处理器中的程序的调试（在线）与仿真，则开发板上需要建立一个20针的排针，将J-Link工具上的JTAG接口通过排线连接到排针上，然后再将开发板上的处理器中有关JTAG接口引出到此排针上，以跟JTAG接口对应的引脚相连。这里，我们采用的是标准的JTAG接法，但是STM32还有SWD接口，SWD只需最少两根线（SWCLK和SWDIO）就可以下载并调试代码了，这同我们使用串口下载代码差不多。

如图 8所示为STM32单片机的最小系统的各电路模块。

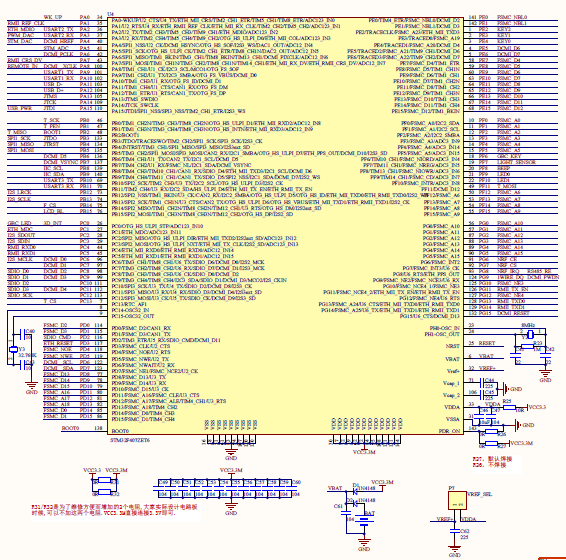


图 8 STM32F407硬件设计图

### 3.2.3 STM32F407硬件实物图

如图 9所示为系统使用的STM32F407硬件实物图。

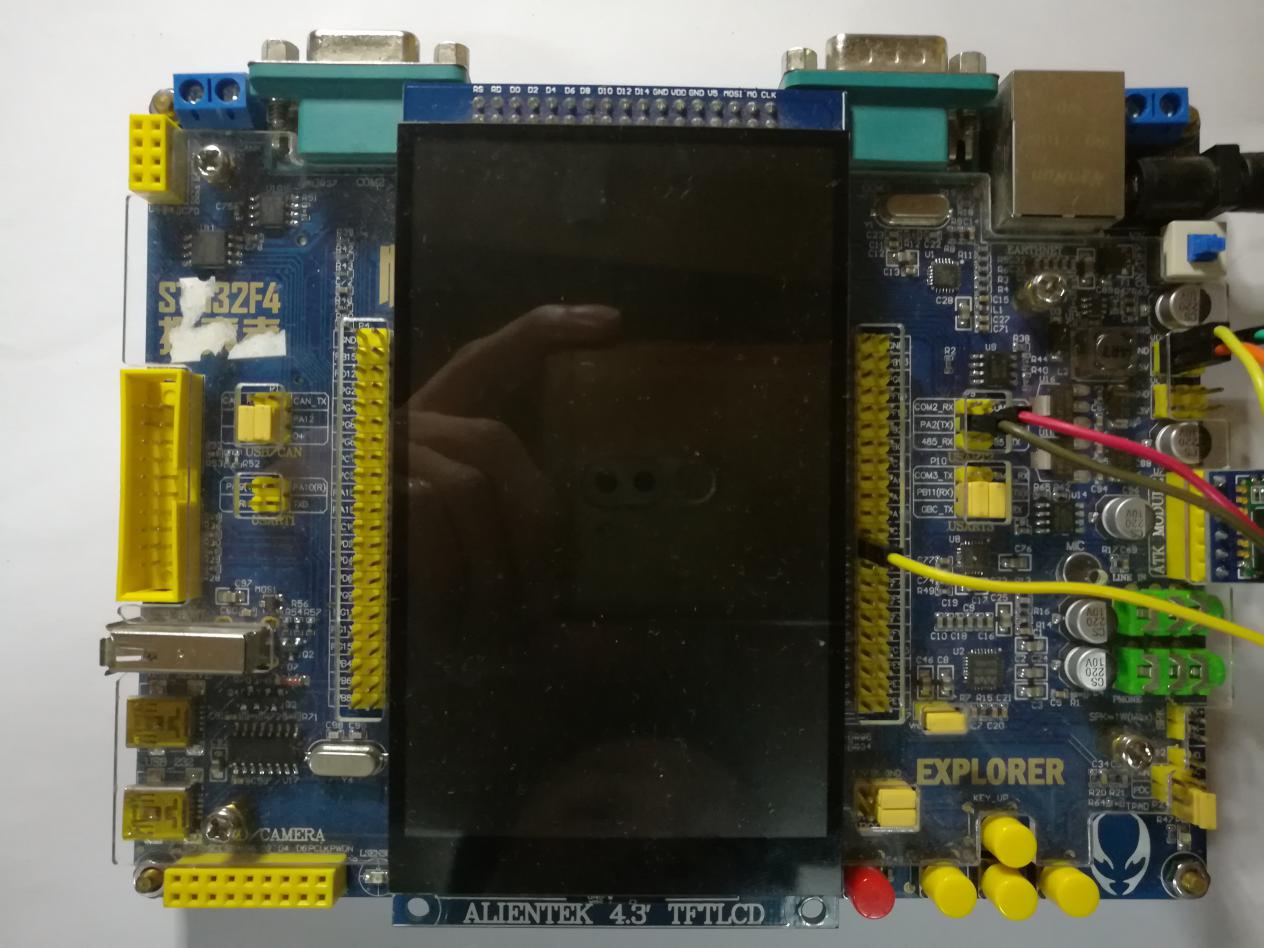


图 9 STM32F407实物图

## 3.3 HC-05蓝牙模块硬件设计

### 3.3.1 HC-05蓝牙方案简介

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；

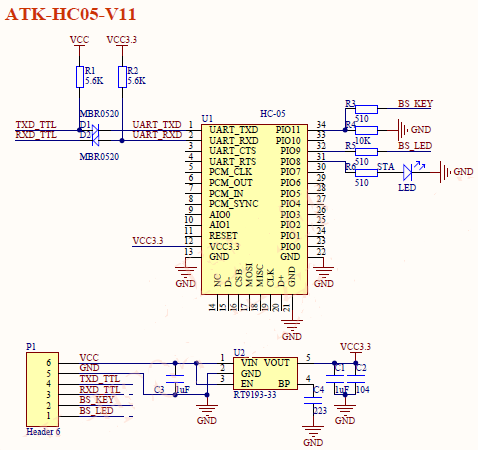
（7）可以与蓝牙笔记本电脑、电脑加蓝牙适配器等设备进行无缝连接。

（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。当模块处于自动连接工作模式时，将自动根据事先设定的方式连接的数据传输；当模块处于命令响应工作模式时能执行下述所有 AT 命令，用户可向模块发送各种 AT 指令，为模块设定控制参数或发布控制命令。通过控制模块外部引脚（PIO11）输入电平，可以实现模块工作状态的动态转换。

### 3.3.2 HC-05蓝牙模块电路设计

HC-05蓝牙模块的电路原理图见图 10所示。

图 10 HC-05蓝牙模块电路设计

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 11所示。

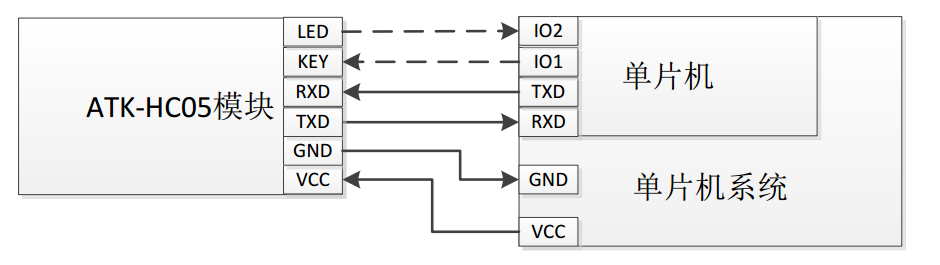


图 11 模块与单片机系统连接图

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，其典型的格式如所示。

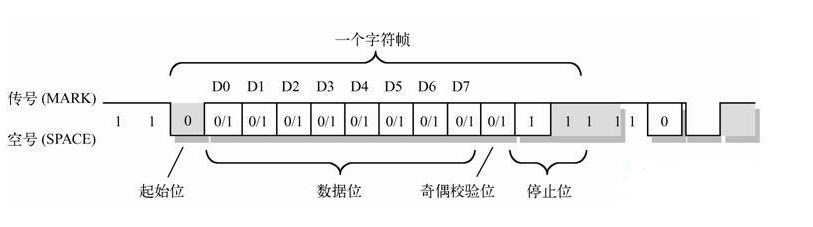


图 12 异步传送一帧数据格式

### 3.3.3 HC-05蓝牙模块实物图

HC-05蓝牙实物图如图 12所示。

|  |  |
| --- | --- |
| 1 | 2 |
| 1. 正面 | 1. 背面 |

图 13 HC-05蓝牙模块实物图

## 3.4 遥控器通信模块

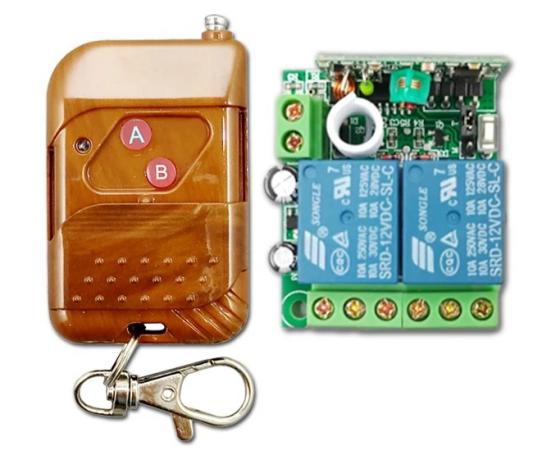


图 14 遥控器通信模块

## 3.5 门锁控制模块

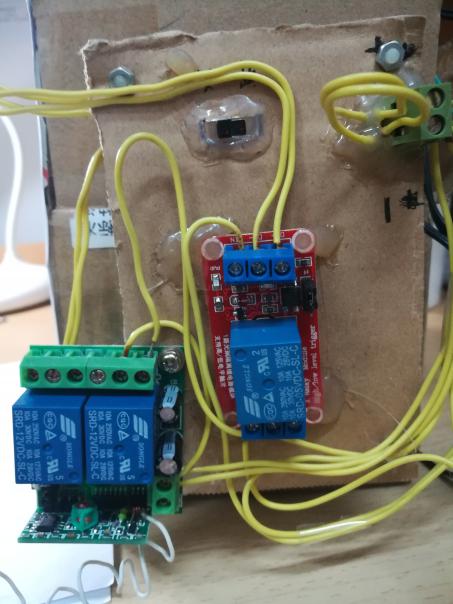
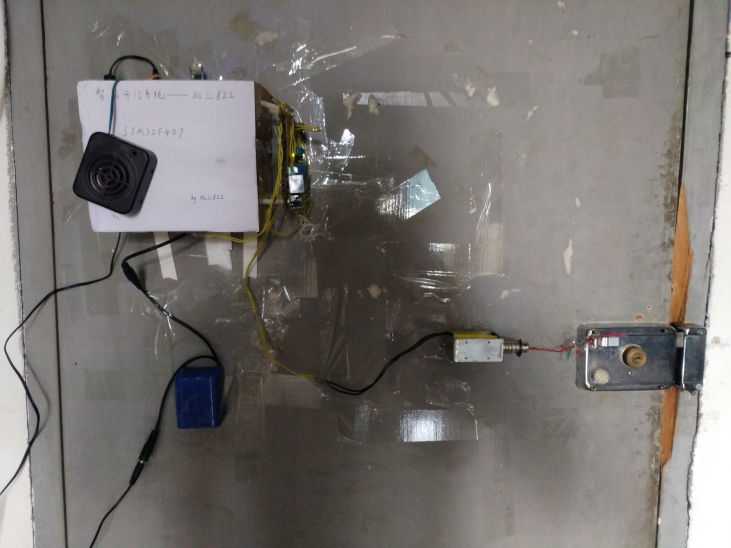
门锁控制模块包括继电器模块和电磁铁模块，如下图所示。

|  |  |
| --- | --- |
| 继电器模块 | 电磁铁 |
| 1. 继电器模块 | 1. 电磁铁模块 |

图 15 门锁控制模块

## 3.5 整机电路

整机电路由STM32主控模块，HC-05蓝牙模块和USB电路组成，实物如图 16所示。



1. 整机电路图 (b) 门锁控制部分

图 16 整机电路图

# 四、系统软件设计及实现

## 4.1 Android APP设计

### 4.1.1开发环境介绍

本次APP开发采用Android Studio实现。Android Studio 是一个全新的 Android 开发环境，基于IntelliJ IDEA。 类似 Eclipse ADT，Android Studio 提供了集成的 Android 开发工具用于开发和调试。

在IDEA的基础上，Android Studio 提供：

* 基于Gradle的构建支持
* Android 专属的重构和快速修复
* 提示工具以捕获性能、可用性、版本兼容性等问题
* 支持ProGuard 和应用签名
* 基于模板的向导来生成常用的 Android 应用设计和组件
* 功能强大的布局编辑器，可以拖拉 UI 控件并进行效果预览

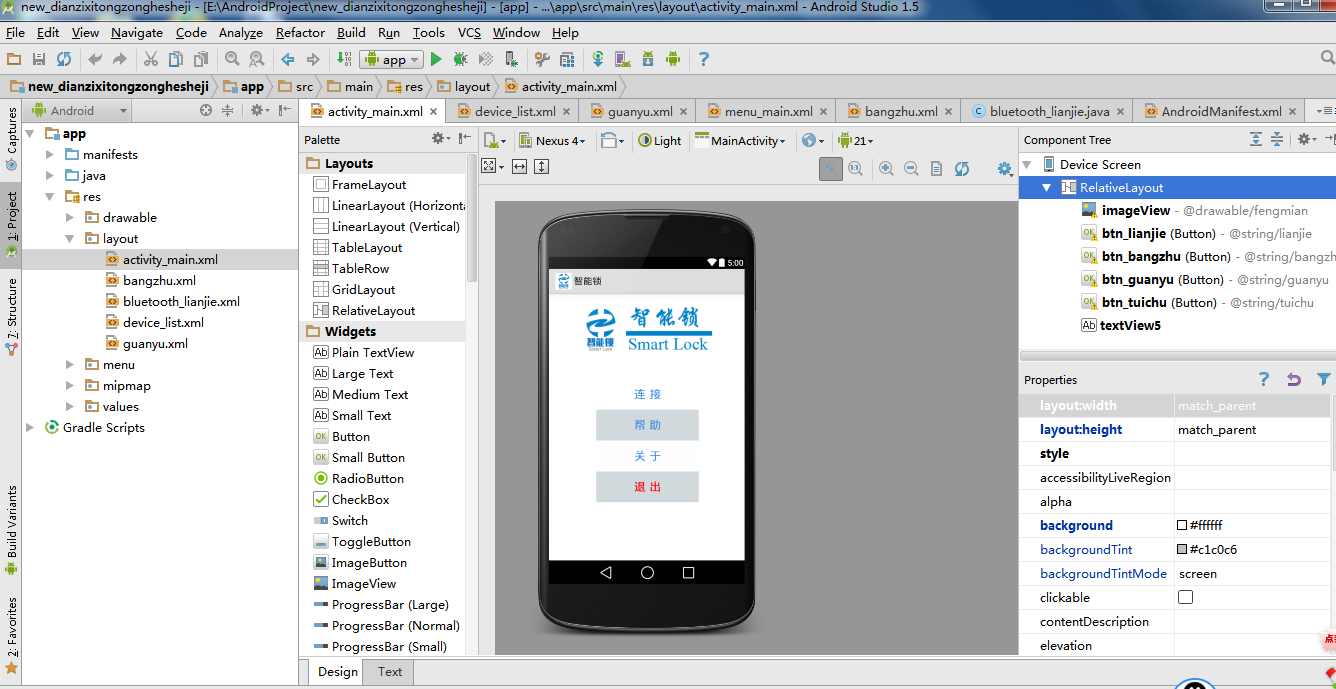


图 17 Android Studio开发环境

### 4.1.2 Android APP界面设计

APP界面设计分为主菜单和功能菜单界面设计。

其中，主菜单包含：连接、帮助、关于、退出，这几个按钮。连接页面为主要的功能页面，它包含蓝牙连接、用户添加和删除、开门和关门等功能。

主菜单设计如下：



图 18 主菜单界面设计

功能菜单设计如下：

|  |  |  |
| --- | --- | --- |
| 连接 |  |  |
| 1. 连接 | 1. 帮助 | 1. 关于 |

图 19 功能菜单界面设计

### 4.1.3 Android APP蓝牙通信实现

蓝牙通信包括：打开蓝牙、连接设备、建立通信通道，具体流程见下图。

**打开蓝牙**

**创建通道**

**连接蓝牙**

**开门/关门(所有用户)**

**登记/删除(root用户)**

**关闭蓝牙**

**退出APP**

图 20 手机APP蓝牙通信流程

其中，打开蓝牙使用openBlueTooth()，

搜索周围设备使用searchDevices()，

实现代码如下：

**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***:  
 searchDevices();  
 **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\_dianzixitongzonghesheji.MainActivity.**class**);  
 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(**androidId**);  
  
 }  
 **else** Toast.*makeText*(bluetooth\_lianjie.**this**, **"请先连接蓝牙！"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 **break**;  
 **case** R.id.***btn\_kaimen***:  
 {  
 **if**( **lianjie\_state**) {  
 **if** (**kaimenorguanmen**)  
  
 {  
 **if** (**androidId**.equals(**"7659483c763e8649"**)) { *//root用户id* **connectThread**.sendMsg(**"R"** + **androidId** + **"1"**);  
 } **else** {  
 **connectThread**.sendMsg(**"C"** + **androidId** + **"1"**);  
 }  
 **kaimenorguanmen** = **false**;  
 **btn\_kaimen**.setText(**"关门"**);  
 } **else** {  
 **if** (**androidId**.equals(**"7659483c763e8649"**)) {  
 **connectThread**.sendMsg(**"R"** + **androidId** + **"0"**);  
 } **else** {  
 **connectThread**.sendMsg(**"C"** + **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\_tianjia***: {  
  
 **if**(!**androidId**.equals(**"7659483c763e8649"**))  
 {  
 Toast.*makeText*(bluetooth\_lianjie.**this**, **"抱歉！您不是root用户，无法使用此功能"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 **else if**(**text\_msg**.getText().toString()!=**null**)  
 { **if**((*returnResultMultiple*(**text\_msg**.getText().toString()).length()==16)||(*returnResultMultiple*(**text\_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();  
  
 }}  
 **break**;  
 **case** R.id.***btn\_shanchu***: {  
 **if**(!**androidId**.equals(**"7659483c763e8649"**))  
 {  
 Toast.*makeText*(bluetooth\_lianjie.**this**, **"抱歉！您不是root用户，无法使用此功能"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 **else if**(**text\_msg**.getText().toString()!=**null**)  
 {  
**if**((*returnResultMultiple*(**text\_msg**.getText().toString()).length()==16)||(*returnResultMultiple*(**text\_msg**.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() {  
  
**text\_msg**.setText(getResources().getString(R.string.***get\_msg***)+**new** String(data));  
 *androidId\_data*=**new** String(data); *//转换* }  
 });  
 }  
 }  
 } **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;  
 }  
}

## 4.2 通信协议简介

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 | 智能锁 | Android ID | root用户注册 |
| root手机 | 智能锁 | ‘R’+root用户Android ID+’0’ | 关门 |
| root手机 | 智能锁 | ‘R’+root用户Android ID+’1’ | 开门 |
| root手机 | 智能锁 | ‘R’+root用户Android ID+’T’+普通用户Android ID | 添加普通用户 |
| root手机 | 智能锁 | ‘R’+root用户Android ID+’S’+普通用户Android ID | 删除普通用户 |
| 普通用户手机 | 智能锁 | ‘C’+root用户Android ID+’0’ | 关门 |
| 普通用户手机 | 智能锁 | ‘C’+root用户Android ID+’1’ | 开门 |

## 4.3 STM32驱动代码

### 4.3.1 固件开发介绍

在进行STM32开发时，为加快开发速度，我们采用固件开发的办法，本次开发需用到STMFLASH、HC05、KEY、LCD、LED等固件。

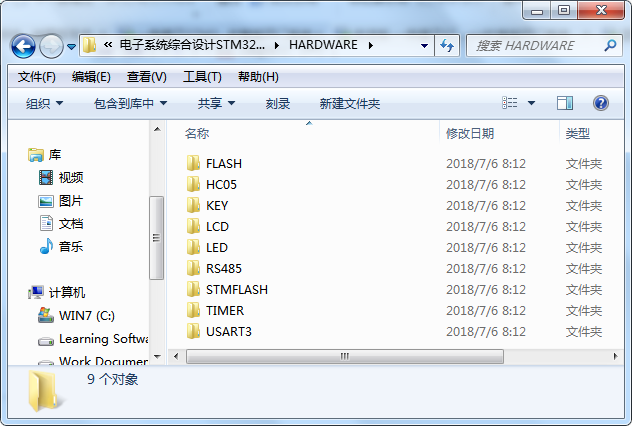


图 21 固件开发引用函数文件

### 4.3.2 按键处理

**切换蓝牙模式：**

if(key==WKUP\_PRES) { //按下WKUP，变换蓝牙主从模式

key=HC05\_Get\_Role();

if(key!=0XFF)

{

key=!key;

if(key==0)HC05\_Set\_Cmd("AT+ROLE=0");

else HC05\_Set\_Cmd("AT+ROLE=1");

HC05\_Role\_Show();

HC05\_Set\_Cmd("AT+RESET");

delay\_ms(200);

}

}

**初始化智能门锁系统**

else if(key==KEY0\_PRES) { //按下KEY0，初始化门锁

flash\_data\_save[0]=1;

num=flash\_data\_save[0];

LCD\_ShowxNum(130,600,num,4,16,0);

for(j=0;j<9;j++)

{

for(i=0;i<16;i++)

{

temp=find\_id[j]+i;

flash\_data\_save[temp]=id\_root\_buf[i]; }

}

LCD\_ShowString(30,500,500,16,16,"Message:Start Write FLASH....");

STMFLASH\_Write(FLASH\_DATA\_SAVE,(u32\*)flash\_data\_save,37);

LCD\_ShowString(30,500,500,16,16,"Message:Initial Finished!");

}

**格式化所有用户信息(包括root用户)**

else if(key==KEY1\_PRES) { //按下KEY1，格式化所有用户

flash\_data\_save[0]=0;

num=flash\_data\_save[0];

LCD\_ShowxNum(130,600,num,4,16,0);

LCD\_ShowString(30,500,500,16,16,"Message:Start Write FLASH....");

STMFLASH\_Write(FLASH\_DATA\_SAVE,(u32\*)flash\_data\_save,37);

LCD\_ShowString(30,500,500,16,16,"Message:Initial Finished!"); LCD\_ShowString(30,500,500,16,16,"Message:Format Finished!");

}

### 4.3.3 登记root用户

if(reclen==16) //判断接收长度

{ 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;

num=flash\_data\_save[0];

LCD\_ShowxNum(130,600,num,4,16,0);

for(j=0;j<9;j++)

{

for(i=0;i<16;i++)

{

temp=find\_id[j]+i;

flash\_data\_save[temp]=USART3\_RX\_BUF[i];

}

}

LCD\_ShowString(30,500,500,16,16,"Message:Start Write FLASH....");

STMFLASH\_Write(FLASH\_DATA\_SAVE,(u32\*)flash\_data\_save,37);

LCD\_ShowString(30,500,500,16,16,"Message:Initial root user Finished!"); }

else LCD\_ShowString(30,500,500,16,16,"Error:Root user has existed,could not change!");

}

### 4.3.4 开门或关门

if(reclen==18||reclen==17)

{

STMFLASH\_Read(FLASH\_DATA\_SAVE,(u32\*)flash\_data\_save,37);

if(USART3\_RX\_BUF[0]==82) //R:82

{

for(i=0;i<16;i++)

{

if(flash\_data\_save[i+find\_id[0]]!=USART3\_RX\_BUF[i+1]) kaimenflag=0;

}

if(kaimenflag==1)

{

LCD\_Fill(30,500,400,550,WHITE);

LCD\_ShowString(30,500,500,16,16,"Message:root use successful!");

if(USART3\_RX\_BUF[17]==48) LED1=0;

if(USART3\_RX\_BUF[17]==49) LED1=1,time=0;

}

kaimenflag=1;

}

if(USART3\_RX\_BUF[0]==67) //C:67

{

for(j=num;j>0;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); LCD\_ShowString(30,500,500,16,16,"Message:common use successful!");

if(USART3\_RX\_BUF[17]==48) LED1=0;

if(USART3\_RX\_BUF[17]==49) LED1=1,time=0;

}

}

}

}

### 4.3.5 添加或删除用户

if(reclen==33)

{

STMFLASH\_Read(FLASH\_DATA\_SAVE,(u32\*)flash\_data\_save,37);

num=flash\_data\_save[0];

LCD\_ShowxNum(130,600,num,4,16,0);

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;

}

}

if(flag)

{ LCD\_ShowString(30,500,500,25,25,"Message:HELLO,my manager!");

if(USART3\_RX\_BUF[16]=='T')

{

if(num<8)

{

for(i=0;i<16;i++)

{ flash\_data\_save[i+find\_id[num]]=USART3\_RX\_BUF[17+i];

}

num=num+1;

flash\_data\_save[0]=num; STMFLASH\_Write(FLASH\_DATA\_SAVE,(u32\*)flash\_data\_save,37); LCD\_ShowString(30,500,500,16,16,"Message:Add User Successfully!");

}

else LCD\_ShowString(30,500,500,25,25,"Error:User registered full!");

}

if(USART3\_RX\_BUF[16]=='S')

{ 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++)

{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]]; }

num=num-1;

flash\_data\_save[0]=num; STMFLASH\_Write(FLASH\_DATA\_SAVE,(u32\*)flash\_data\_save,37);

LCD\_ShowString(30,500,200,16,16,"Delete User Finished!");

}

else

{LCD\_ShowString(30,500,200,16,16,"Error:No this user!");

}

}

else LCD\_ShowString(30,500,210,25,25,"Error:No common user!");

}

}

}

### 4.3.6 扩展功能语音控制开门

RS485\_Receive\_Data(rs485buf,&len);

if(len) {

if(len>5)len=5;

LCD\_ShowxNum(30+i\*32,500,rs485buf[i],3,16,0X80);

if(rs485buf[0]==16&&rs485buf[1]==13) LED1=1,time=0; if(rs485buf[0]==1&&rs485buf[1]==13) LED1=0;

}

# 五、项目成果

## 5.1 完整的软件设计



图 22 智能锁APP下载链接

完整的项目产品及演示视频：<https://pan.baidu.com/s/1uUYL702afbpQiYrOaAuIwg>

## 5.2 产品外观设计

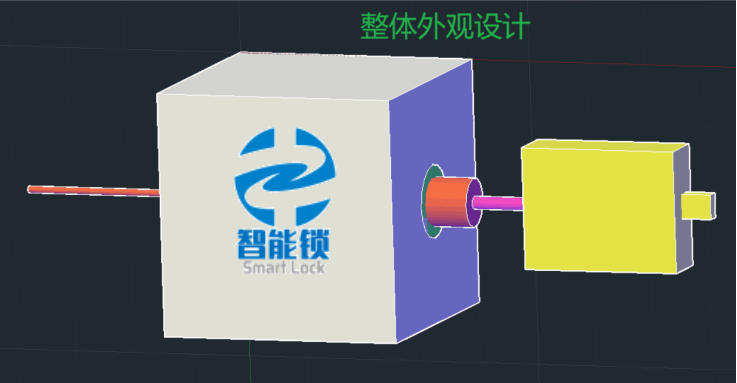


图 23 产品外观设计

## 5.3 专利申请

正在受理发明+实用新型专利《一种基于STM32和Android的智能开门方法及其系统》，已经向专利中心提交申请。

## 5.4 产品应用

目前该款智能门禁系统已经应用于华南理工大学北三822宿舍，并且大大改善该宿舍的生活质量以及方便性。

# 六、项目收获与感悟

* 在项目中锻炼了团队合作以及分工协调的能力；
* 团队成员在项目中提升了自己的学习能力和实践能力，很好的将理论应用于实际；
* 通过本次项目，初步掌握STM32的开发和Android APP的开发，以及蓝牙模块的使用；
* 感谢徐向民老师和舒琳老师的悉心指导以及严格要求。

# 七、附件

## 附件1 支出情况

表 2 项目支出情况

|  |  |  |
| --- | --- | --- |
| **时间** | **报销内容** | **报销金额（元）** |
| 2018.5 | 锂离子电池 | 84 |
| 2018.5 | 电磁铁 | 74 |
| 2018.5 | 跳线帽、蓝牙模块、1路继电器模块、STM32F407ZGT6开发板、人体感应模块 | 142.41 |
|  |  | 合计：304.42 |

(备注：专利费用800发明+500实用新型)由国创项目《一种基于可见光通信的服务机器人室内定位系统》支出)

## 附件2 同类产品比较



图 24 同类产品比较

## 附件3 市场调研



图 25 市场调研