南 京 工 程 学 院

毕业设计说明书(论文)

作 者：  **封光**  学 号： **202130414**

院 系： **计算机工程学院**

专 业：  **计算机科学与技术（嵌入式系统）**

题 目： **智能门禁系统开发**

指导者： **钱瑛 讲师**

评阅者： **讲师**

2017年5 月 南 京

**毕业设计说明书（论文）中文摘要**

|  |
| --- |
| 随着互联网技术和智能移动终端设备的飞速发展，人们希望移动智能终端能够代替生活中的日常使用工具，如钥匙、门锁等，期望智能终端能给他们更好的生活体验。门禁系统是整个家居安防中的最重要环节，传统门禁系统已经很难满足人们对智慧化、互联网化的生活需求。本系统将对传统门禁系统进行改造，融合智能家居的理念，使得门禁系统更加安全化、智能化、人性化。  目前市场上大部分主流旗舰机型均带有NFC进场通信模块，NFC近场通信模块相比传统的RFID门禁卡模块具有更强的安全性、便捷性，可以完美替代传统RFID门禁卡模块。此外，相比RFID模块还可以实现双向通信，可以延伸出更多的应用场景。  本文详细描述了整个智能门禁系统的开发流程。首先，本文分析了国内外智能门禁系统的研究现状以及市场销售现状，接着分析智能门禁系统需求，对需求进行功能模块的拆分，然后详细描述了系统在搭建过程中用到的技术和碰到的问题。并且对整个系统UI进行了详细介绍。最后经过系统性测试，各个功能模块正常稳定运行，人机界面友好，满足了现有的需求，并且有着很强的扩展性。  本系统选取树莓派硬件系统，烧录Google最新Android things系统，并在系统上做相应的开发，以满足系统功能需求。用android studio平台来发Android端APP，app端具有用户注册、登录、用户管理、异常管理、开锁事件等功能模块，并可以用NFC开锁和远程开锁两种开锁方式。开锁通过树莓派驱动PWM端口控制舵机模块来模拟门禁开关。整个系统融入互联网技术，极大地提高了门禁的安全性和便捷性。  关键词 Android开发 NFC Android things物联网操作系统 数据库 |

**毕业设计说明书（论文）外文摘要**

|  |
| --- |
| With the rapid development of Internet technology and intelligent mobile terminal equipment, people want mobile intelligent terminal can replace the daily use of life tools, such as keys, locks, etc., expect smart terminals to give them a better life experience. Access control system is the whole home security in the most important part of the traditional access control system has been difficult to meet people's wisdom, the Internet needs of life. The system will transform the traditional access control system, the concept of integration of intelligent home, making the access control system more secure, intelligent, user-friendly.  Currently, most mainstream flagship models on the market are equipped with NFC approach communication module. NFC near field communication module has better security and convenience than traditional RFID access card module, and can replace traditional RFID access card module. In addition, compared to RFID module can also achieve two-way communication, you can extend more of the application scenarios.  This article describes in detail the entire smart access control system development process. First of all, this paper analyzes the status quo of intelligent access control system at home and abroad as well as the status quo of market sales, and then analyzes the requirements of intelligent access control system, the functional modules of the split, and then describes the system used in the construction process and encountered The problem. And the entire system UI for a detailed description. Finally, after a systematic test, the normal operation of the various functional modules, friendly man-machine interface to meet the existing needs, and has a strong scalability.  The system chooses the raspberry dispatching hardware system, burns Google's latest Android things system, and develops the system to meet the system function demand. With the Android platform to send Android end APP, app end with user registration, login, user management, exception management, unlock events and other functional modules, and can use NFC unlock and remote unlock two ways to unlock. Unlocks the rudder by driving the PWM port to control the steering gear module to simulate the access switch. The entire system into the Internet technology, greatly improving the security and convenience of access control.  **Keywords**  Mobile Application, NFC,Android things operating system,database |

**目　　录**

[前 言 1](#_Toc483215474)

[第一章 绪论 2](#_Toc483215475)

[1.1 课题研究意义 2](#_Toc483215476)

[1.2 国内外研究现状 2](#_Toc483215477)

[1.3 课题研究内容 3](#_Toc483215478)

[第二章 技术概览 4](#_Toc483215479)

[2.1 Android things嵌入式系统 4](#_Toc483215480)

[2.2 NFC协议介绍 5](#_Toc483215481)

[2.3 Android Studio开发平台 7](#_Toc483215482)

[第三章 通讯模块设计实现 8](#_Toc483215483)

[3.2 基于AES的通讯信息加密算法 11](#_Toc483215484)

[3.3 串口通讯设计 14](#_Toc483215485)

[第四章 智能门禁系统软件设计 18](#_Toc483215486)

[4.1 系统软件总体设计 18](#_Toc483215487)

[4.2 Android开锁模块设计 19](#_Toc483215488)

[4.3后台数据分析管理模块设计 25](#_Toc483215489)

[4.4 Android things服务端模块设计 30](#_Toc483215490)

[第五章 智能门禁系统硬件设计 32](#_Toc483215491)

[5.1门禁控制-舵机模块 32](#_Toc483215492)

[5.2 Raspberry Pi 3 运行平台 33](#_Toc483215493)

[5.3 PN532 NFC模块设计 35](#_Toc483215494)

[第六章 结论 37](#_Toc483215495)

[参考文献 38](#_Toc483215496)

[致谢 39](#_Toc483215497)

[附录：英文技术资料翻译 40](#_Toc483215498)

前 言

近年来，随着互联网技术和智能家居的蓬勃发展，传统门禁系统已经不能满足用户的需求，传统门禁不能满足钥匙管理、用户管理等方面的需求，为此，本系统在做了市场调研后，决定用最新的互联网技术来对传统门禁系统做相应的改进，加入了用户管理、NFC开锁、远程开锁、异常管理、开锁事件查看等功能。

智能移动终端和NFC近场通信技术的结合，为系统的安全性提供了极大的保障和策略选择空间。使得门禁系统的发展产生了质的变化。用移动智能终端代替传统机械锁钥匙来实现门禁的控制，将成为门禁系统发展的一种新的趋势。智能移动终端和后台服务器通信，可以为虚拟用户提供电子密钥，并可以对电子密钥进行管理，以及通过对密钥绑定的用户的管理，和虚拟用户的事件记录、分析等。减少了卡复制或者密钥包盗用的情况。更为有效的降低了人为原因的门禁系统风险。

本系统客户端选用Android操作系统，Android操作系统是基于Linux的一个深度定制的操作系统，用Binder方式取代传统Linux RPC方式，并在Linux系统上运行ART虚拟机，方便开发者用java快速搭建应用程序。本系统的服务端采用Android things操作系统，Android things操作系统是Google公司于2016年11月发布的一款物联网操作系统，该系统基于Android操作系统进行了精简，使得该操作系统可以运行在系统性能较弱的嵌入式系统中。由于成本原因，搭载Android things的树莓派3b+成了本系统的硬件平台选择。

第一章 绪论

## 课题研究意义

智能门禁系统是一种新型的现代安全管理系统，将计算机自动识别技术与现代安全管理措施整体结合起来，涉及电子，机械，光学，计算机技术，通信技术，生物技术等诸多新技术。它是用来解决重要部门的出入口安全管理的措施。适用于银行，酒店，机房，军械库，保密室，办公空间，智能小区，工厂等各类保密部门。在数字网络技术快速发展的今天，智能门禁技术得到了飞速的发展。门禁系统早已不再只是单纯的门控和钥匙管理，它已经逐渐发展成为一套成熟的出入管理系统。它在人事考勤管理，工作环境安全等行政工作中发挥了巨大的作用。

在原有门禁系统的基础上增加相应的辅助设备可以实现保安巡检管理、物业消防监控、电梯控制、安全检查管理等，真正实现智能管理。

## 1.2 国内外研究现状

智能门禁系统随着自动化识别技术的发展而迅速发展，国外门禁系统的知名品牌有美国的修斯（HID）、西坞（WSE）、罗泰克（NTK），以色列的DDS、英国的集保等。国内有销售门禁系统有深圳捷顺、门吉、红门、青云等公司。目前，国内外使用和研制的门禁系统主要分为生物辨识别门禁系统和感应门禁系统。在生物识别门禁系统中，又以指纹门禁系统最为广泛应用。随着模式识别理论、计算机图像处理技术、集成电路技术等的不断成熟与发展，指纹识别模块体积的不断缩小和算法的不断优化，其价格也不断降低。据国际生物认证组织（IBG）的预测：生物识别市场份额2002年将突破7亿美元，其中虹膜识别占8%，而指纹占将近50%。生物识别市场份额到2005年将达到19亿美元，其中虹膜识别占10%左右，指纹识别占40%左右。并且生物识别市场份额将在未来十几年内仍会保持高速增长的势头，纵观当下门禁市场，预估未来1－3年内中国的门禁市场增长比将达到30％。

目前国外制作的门禁系统又较高的安全性，但是价格确比较昂贵，感应式门禁控制器的价格从5000到4万元不等，如果包括门锁、指纹模块、控制器、以及软件在内的一整套系统价格基本上在万元以上。而国内生产制作的的指纹门禁系统和感应式门禁系统价格相比起国外同类产品价格大约低14%-55%，但是系统的可靠性和安全性相对来说较差。

## 1.3 课题研究内容

本课题采用Linux和 Android 技术来设计开发一个具有远程控制门禁、NFC身份识别以及后台数据分析及管理等功能的系统。本课题主要分为三个部分：服务器、android解锁客户端、android后台统计数据分析客户端。其中服务器采用Android things物联网系统(基于Linux)，用来控制门禁开关、接收Android解锁客户端的网络请求和后台统计系统的网络请求。解锁客户端只要负责实现远程解锁和刷NFC解锁两大功能模块。后台统计系统主要负责门禁开关事件的记录、用户身份控制，异常分析等等。

第二章 技术概览

## 2.1 Android things嵌入式系统

Android Things是Google公司向物联网市场推出的最新智能系统，这让物联网市场的竞争进一步升温。目前，微软正在向发烧友和原始设备制造商(OEM)推广其Windows 10 IOT系统。尽管亚马逊并未开发相应的设备系统，但是该公司已经把AWS Lambda计算服务开放给了物联网开发者。亚马逊云计算部门AWS正在与设备制造商合作，在设备中绑定Lambda Greengrass软件。该软件是AWS Lambda的本地版本，可以被安装到物联网设备和套件中。Android Things使用和Android开发一样的工具、Android框架和Google APIs，这些都使得开发物联网变得更简单。Android things的平台架构如下：(图2-1)

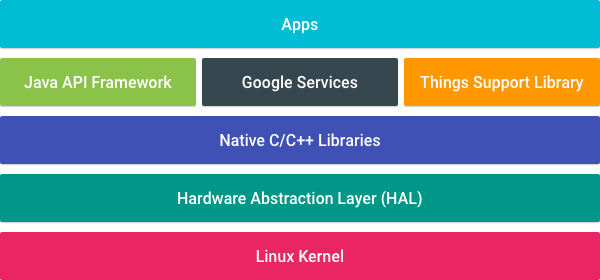


图2-1

Android Things继承了Android核心框架层并且通过支持包提供额外的APIs。这些APIs允许apps与新型的硬件之间交互，但是这些硬件并不存在于移动设备上。Android Things平台仍然为单一应用程序使用。系统app不存在，并且你的app将会自动启动来给你的用户带来沉浸式体验。

相比起之前Google发布的Brillo系统，Brillo把C++作为主要开发环境，而Android Things则面向所有Java开发者，不管开发者有没有移动开发经验。开发者可以针对物联网设备使用工具包。Android Studio是最常用的Android集成开发环境，它基于IntelliJ IDEA开源社区版本，支持通过Android Things进行物联网开发。应用开发的生命周期几乎和移动开发相同。Android Things系统支持原始Android SDK中的一个子集，不支持需要用户输入或者需要地图、搜索以及登录等验证凭证的应用程序接口(API)。和Android手机的OTA无线升级一样，开发者可以通过谷歌已经在其产品和服务中使用过的相同OTA基础设施，推送谷歌提供的系统更新和定制应用升级。谷歌推荐开发者在最初原型设计和开发时使用微型电脑“树莓派3”、英特尔微型可穿戴设备“爱迪生”(Edison)以及恩智浦半导体的Pico平台。谷歌会在Android Things正式发布前增加更多认证设备。包括云消息传递服务Firebase在内的谷歌云平台组件能够轻易与Android Things整合。开发者可以利用多个云服务进行存储、状态管理以及发送消息。Android Things整合了物联网设备通讯平台Weave。Weave Server是用来处理设备注册、命令传送、状态存储以及与谷歌助手等谷歌服务整合的云服务。Weave SDK将嵌入到设备中进行本地和远程通讯。

本系统使用树莓派3b+作为Android things系统的载体，在树莓派3b+中刷入最新的Android things系统。

## 2.2 NFC协议介绍

NFC，近场通信（英语：Near Field Communication，NFC），近距离无线通信技术，它是一种面向短距离的高频无线通信技术，它允许终端之间进行点对点非接触式数据传输，可以在十厘米内交换数据。

这个技术由RFID非接触式射频技术演变而来，由Nokia、Philips半导体公司和Sony共同研制开发，它基于RFID互连技术。NFC是一种高频短距的无线通信技术，以13.56MHz的频率运行，在20厘米距离内。其传输速度分为105 Kbit/秒、202 Kbit/秒或者434 Kbit/秒三种。目前NFC通信技术已通过成为ISO/IEC IS 18092标准、EMMA-340标准与ETTI TS 102标准。NFC采用主动和被动两种读取模式。

NFC有三种工作模式：点对点模式、卡模拟模式、读卡器模式。点对点模式：这个模式和红外线比较类似，可以用于数据双向传输，只不过传输距离比较短，但是创建连接速度较快，也有更好的传输熟读，和蓝牙一样，也有着较低的功耗。将两个搭载NFC模块的设备相连，可以实现点对点传输数据，比如下载电子书、交换音乐或者同步设备联系人等等。因此，通过NFC，多个设备之间比如数码相机、个人计算机和智能移动终端等等，都可以交换资料或者提供服务。卡模拟模式：这个模式可以模拟采用RFID技术的门禁卡。可以替代现在大部分的RFID卡（包括信用卡）的应用场景，比如商场刷卡、车票、门禁、公园门票等等。这种模式有一个极大的优势，那就是普通RFID卡片需要要通过读卡器来供电，而NFC在寄主设备（如手机）没电的情况下也可以正常工作。NFC设备如果要进行卡模拟模式，就必须内置安全NFC芯片。读卡器模式：作为非接触读卡器使用，比如可以从电子标签上或者RFID卡读取相关信息。

NFC和蓝牙都属于短距离通信技术，而且都可以很方便的集成到移动智能终端中。NFC设置程序较为简单。NFC也可以告别繁琐的蓝牙连接过程。NFC相比于蓝牙无需繁琐的连接过程，但同时无法达到蓝牙设备的的传输速率。在两台终端连接过程中，使用NFC来替代人工设置会大大加快连接的速度：少于十五分之一秒。NFC的最大传输速率434 kbit/s远小于蓝牙v2.1（2.1 Mbit/s）。虽然NFC在传输速率和距离方面比蓝牙稍弱，但NFC受到其他设备的干扰较少。这让使得NFC更适用于设备密集连接较慢的时候。

NFC兼容目前的的RFID（13.54 MHz ISO/IEC 18200-4）设备，相比于蓝牙，NFC的电量损耗更低，与V4.0低功耗蓝牙类似。不过当NFC在一台无法持续供电的设备（比如一台关机的手机）上工作时，NFC的电量消耗会要稍微大于V4.0低功率蓝牙。

对智能移动终端或是消费性电子产品来说，NFC的使用相比蓝牙更加方便。NFC以其短距离通信、耗电量低、点对点相连、有较高的安全性更收用户的欢迎。NFC可以避免信用卡交易时期被盗刷。但是也不是说NFC可以取代蓝牙等其他短距离无线通信技术，NFC和蓝牙各有各的优缺点，他们在不同的场合和领域起到相互补充的作用。

本系统中将使用NFC的点对点模式，采用PN532模块和树莓派上位机通过串口通信，来实现对NFC模块的控制和信息采集。

## 2.3 Android Studio开发平台

Android Studio是一个为Android平台开发程序的集成开发环境。2013年5月16日在Google I/O上发布，可供开发者免费使用。2013年5月发布早期预览版本，版本号为0.1。2014年6月发布0.8版本，至此进入beta阶段。第一个稳定版本1.0于2014年12月8日发布。Android Studio基于JetBrains IntelliJ IDEA，为Android开发特殊定制，并在Windows、OS X和Linux平台上均可运行。

本系统中使用的版本有以下功能可用

* 可视化布局： WYSIWYG编辑器，实时程序界面预览。
* 开发者控制台： GPU、CPU状态显示，内存占用分析，函数调用栈，函数占用时间分析等。
* Gradle构建支持
* 快速代码重构和代码格式化
* 支持java代码混淆和应用签名功能。
* 可以基于模板来自动生成常用的Android UI组件。
* 布局编辑器，可让开发者简单的拖拽UI组件，并在不同尺寸设备上预览UI显示效果。
* 支持开发Android Wear应用。

第三章 通讯模块设计实现

3.1 基于socket封装的类http通讯格式

在计算机科学技术中，套接字（英译： socket），又称网络套接字、网络插槽，是网络中不同计算机进程间通信的工具。以网络协议（Internet Protocol）为基础的套接字称为网际套接字（Internet socket）。因为网络的流行，现代绝大多数的套接字都属于网际套接字的范畴。

socket是一种操作系统提供的进程间通信机制。socket最初被翻译为把socket译为“媒介(字)”。不久，ARPANET的socket就被翻译为“套接字”，其理由是：由于每个主机系统都有各自命名进程的方法，而且常常是不兼容的，因此，要在全网范围内硬把进程名字统一起来是不现实的。所以，每个计算机网络中都要引入一种起媒介作用的、全网一致的标准名字空间。这种标准名字，在ARPA网中称作套接字，而在很多其他计算机网中称作信口。更确切地说，进程之间的连接是通过套接字或信口构成的

在操作系统中，通常会为应用程序提供一组应用程序接口（API），称为套接字接口（英语：socket API）。应用程序可以通过套接字接口，来使用网络套接字，以进行数据交换。最早的套接字接口来自于4.2 BSD，因此现代常见的套接字接口大多源自Berkeley套接字（Berkeley sockets）标准。在套接字接口中，以IP地址及通信端口组成套接字地址（socket address）。远程的套接字地址，以及本地的套接字地址完成连接后，再加上使用的协议（protocol），这个五元组（five-element tuple），作为套接字对（socket pairs），之后就可以彼此交换数据。例如，再同一台计算机上，TCP协议与UDP协议可以同时使用相同的port而互不干扰。 操作系统根据套接字地址，可以决定应该将数据送达特定的进程或线程。这就像是电话系统中，以电话号码加上分机号码，来决定通话对象一般。

手机能够使用联网功能是因为手机底层实现了TCP/IP协议，可以使手机终端通过无线网络建立TCP连接。TCP协议可以对上层网络提供接口，使上层网络数据的传输建立在“无差别”的网络之上。

HTTP协议即超文本传送协议(Hypertext Transfer Protocol )，是Web联网的基础，也是手机联网常用的协议之一，HTTP协议是建立在TCP协议之上的一种应用。

HTTP连接最显著的特点是客户端发送的每次请求都需要服务器回送响应，在请求结束后，会主动释放连接。从建立连接到关闭连接的过程称为“一次连接”。

由于HTTP在每次请求结束后都会主动释放连接，因此HTTP连接是一种“短连接”，要保持客户端程序的在线状态，需要不断地向服务器发起连接请求。通常的做法是即时不需要获得任何数据，客户端也保持每隔一段固定的时间向服务器发送一次“保持连接”的请求，服务器在收到该请求后对客户端进行回复，表明知道客户端“在线”。若服务器长时间无法收到客户端的请求，则认为客户端“下线”，若客户端长时间无法收到服务器的回复，则认为网络已经断开。

由于通常情况下Socket连接就是TCP连接，因此Socket连接一旦建立，通信双方即可开始相互发送数据内容，直到双方连接断开。但在实际网络应用中，客户端到服务器之间的通信往往需要穿越多个中间节点，例如路由器、网关、防火墙等，大部分防火墙默认会关闭长时间处于非活跃状态的连接而导致 Socket 连接断连，因此需要通过轮询告诉网络，该连接处于活跃状态。

而HTTP连接使用的是“请求—响应”的方式，不仅在请求时需要先建立连接，而且需要客户端向服务器发出请求后，服务器端才能回复数据。

很多情况下，需要服务器端主动向客户端推送数据，保持客户端与服务器数据的实时与同步。此时若双方建立的是Socket连接，服务器就可以直接将数据传送给客户端；若双方建立的是HTTP连接，则服务器需要等到客户端发送一次请求后才能将数据传回给客户端，因此，客户端定时向服务器端发送连接请求，不仅可以保持在线，同时也是在“询问”服务器是否有新的数据，如果有就将数据传给客户端。

在本系统中，由于不需要完全实现http的所有格式支持，所以采用了socket模拟http的方式，根据业务需求写了部 分属性支持。比如一个完整http请求的设计、包头包尾的定义。结合多线程/线程池以及java回调的相关知识，延伸出一套可用的sdk，并集成到了项目当中。整个请求过程可以用下图表示。（图3-1）

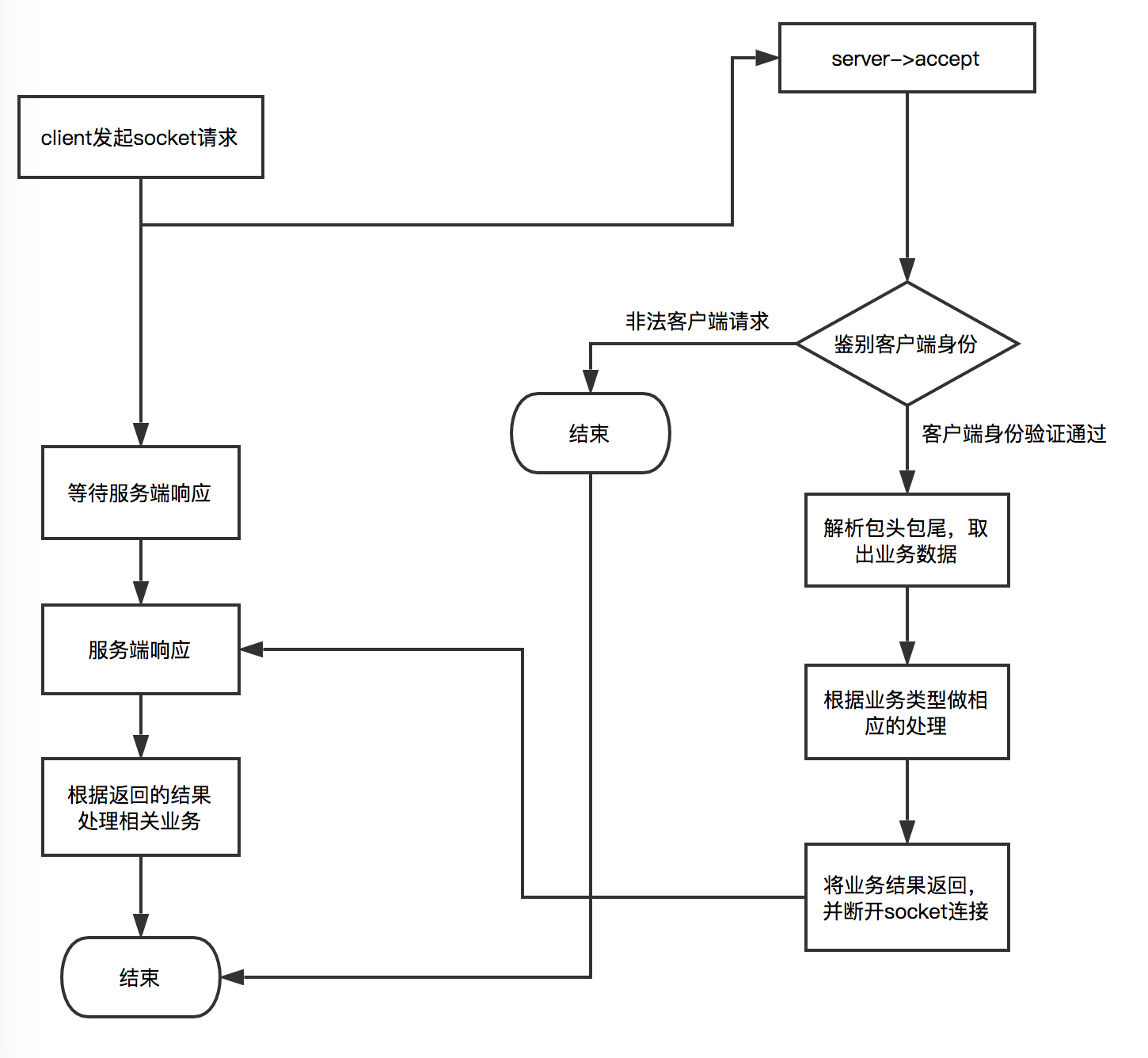


图3-1 socket通信流程图

在整个请求流程中，客户端先从线程池中抽出一个闲置的线程，通过tcp socket发起网络请求。服务端有个一直阻塞的accpet线程，用来接收客户端的请求。在来一个socket请求之后，可以得到一个客户端的套接字描述符。后期的身份鉴权以及通信将依赖这个套接字描述符展开。服务端在拿到套接字描述符之后，将用约定好的密文来进行客户端的身份验证，如果身份验证失败将直接断开连接。如果身份验证成功，则进行下一步的通信。身份验证之后，服务端将根据事先约定好的包头包尾部剥离出主要通信正文，注意这里的密文是加密过的，这段加密算法将在下一节提到。在进行通信正文的解密之后，就可以拿到客户端的头字段、type、action等字段。由于socket通信顺序的不确定性，这里设计了一个头字段的约定，头字段里面包含了一个根据时间戳生成的一个UID，这个UID将会原封不动的返回给客户端，客户端拿到服务端返回的字段之后，会解析出UID字段，并将这个通信通过java回调的方式分发给该请求的线程。然后这个线程将会处理服务端返回的字段，处理相关业务代码之后，完成一次完整的请求。其中type字段类似于http中的URL，标识了要访问的资源，在这里简化成了一个type字符串，用来标识业务请求，比如开锁、获取开锁事件等等请求。action字段则标识了这个请求的内容，可以携带一个需要的参数。然后服务端解析相关参数后完成相应的业务代码。

## 3.2 基于AES的通讯信息加密算法

作为一个门禁系统，安全问题自然也是重中之重。在本系统中，主要的安全隐患存在于通信过程中。如果使用明文加密，通过反编译app或者中间人攻击，即可拿到通信协议，并且冒充客户端来欺诈服务器，逃过安全验证。所以，本系统将采用AES对称加密算法来实现通信过程的加密。

AES，高级加密标准（英语：Advanced Encryption Standard，缩写：AES），在密码学中又称Rijndael加密法，是美国联邦政府采用的一种区块加密标准。这个标准用来替代原先的DES，已经被多方分析且广为全世界所使用。经过五年的甄选流程，高级加密标准由美国国家标准与技术研究院（NIST）于2001年11月26日发布于FIPS PUB 197，并在2002年5月26日成为有效的标准。2006年，高级加密标准已然成为对称密钥加密中最流行的算法之一。

AES加密过程涉及4种操作：字节替代（SubBytes）、行移位（ShiftRows）、列混淆（MixColumns）和轮密钥加（AddRoundKey）。下图（图3-2）给出了AES加解密的流程，从图中可以看出：1）解密过程的每一步分别对应加密的逆操作，2）加解密所有操作的顺序正好是相反的。正是由于这几点（再加上加密与解密每步的操作互逆）保证了解密能够正确地恢复明文。加解密中每轮的密钥分别由种子密钥经过密钥扩展扩展得到。算法中16字节的明文、密文和轮子密钥都以一个4x4的矩阵表示。



图3-2 AES加解密流程

在AES加密中，需要key(私钥)、iv(偏移量)、加密原文，根据这三项输入可以得到密文的输出。这时候就会暴露一个问题，就是key(私钥)、iv(偏移量)的安全存储问题。由于这两个输入项是用户自定义的，而且必须是不能暴露的。所以，如何安全地存储这两个值是一个比较令人头痛的问题。

在本系统中，采用了安全图片存储机制。顾名思义，就是把这两个值存在图片中，然后这个图片可以通过http访问、本地存储等方式来获取，也可以用直接当成资源文件放在工程目录中。

在设计安全图片存储方案的时候，大致查阅到了以下几种思路：

* 在jpg中，可以通过写入Exif区，这块区域存入了相机拍摄信息：时间、拍摄地点等等，可以在这个区域存入加密后的信息，实现图片的无损加密。
* 在png中，可以在IEND(png文件尾部)区域后追加加密信息，在此区域追加的信息不会影响图像的正常读取。但是用png格式需要注意在编译的时候会被编译器自动压缩，从而失去加密的信息。可以考虑动态获取安全图片来规避这个问题，比如从网络中加载图片再写入本地。
* 从图片本身结构入手，比如通过你要加密的信息构造出RGB、alpha等信息，然后按照标准png/jpg/bmp等格式构造出一张全新的图片，也可以在中间按照某种规则穿插一些混淆数据。这样生成的图片加密程度较高，不容易破解，但相对来说开发成本较高。

图片从工程里的任意一张图片获取，然后在app加载的时候读入这张图片，根据png的尾部信息来获取加密的信息，然后通过约定好的规则来算出key(私钥)、iv(偏移量)，并用私钥和偏移量来进行通信过程中的加解密。

在使用过程中还暴露出另外一个问题，就是防二次打包。因为如果盗窃者剥离出你的祥光代码和图片，并在自己的app中使用，就可以完全伪造客户端，所以在本系统中还加入了防止二次打包的验证，即在制作安全图片的时候也将app的签名摘要sha-1值写入图片中，在读取的过程中验证这个签名是否和图片中的是否相等。如果不等就直接杀死app进程，从而保证app的安全。整个解密流程如下图所示。（图3-3）

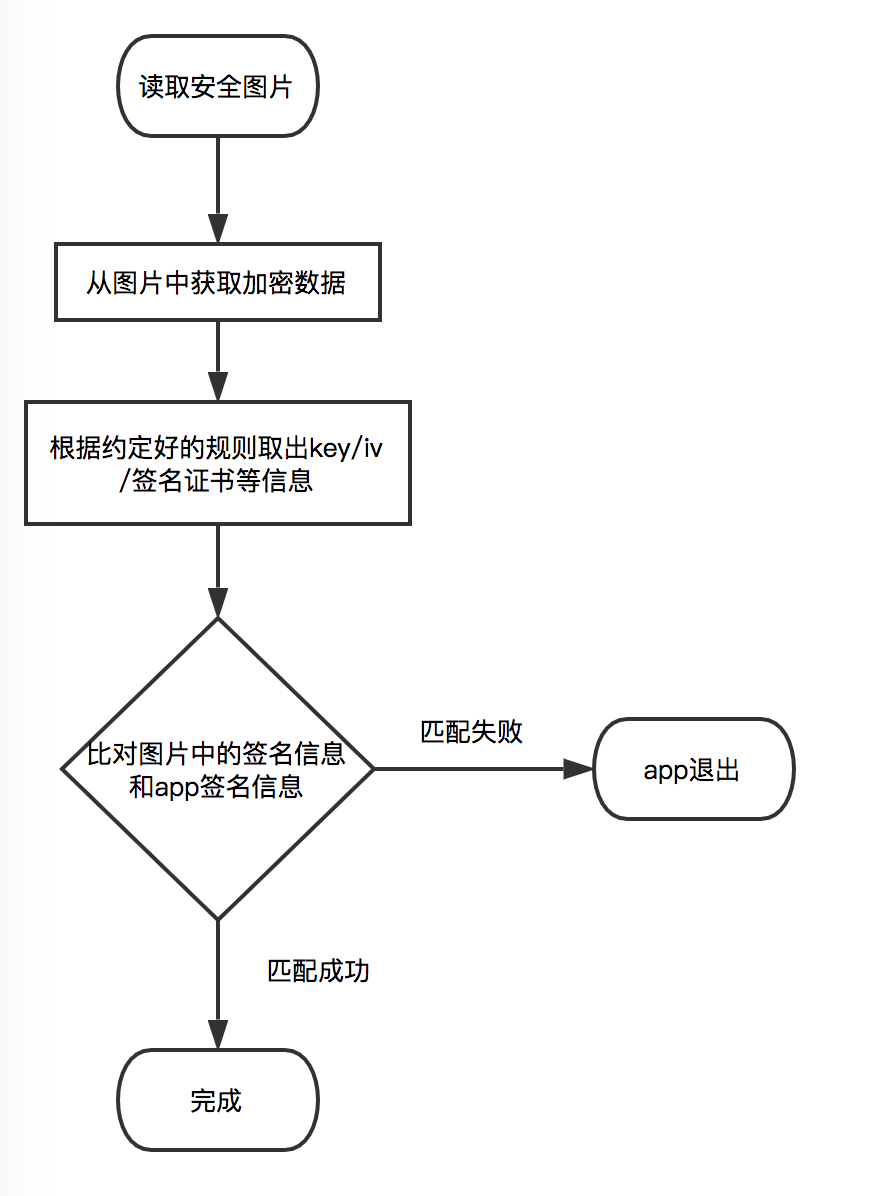


图3-3 安全图片验证流程

## 3.3 串口通讯设计

串口通信(Serial Communication)， 是指外设和计算机间，通过数据信号线 、地线、控制线等，按位进行传输数据的一种通讯方式。这种通信方式使用的数据线少，在远距离通信中可以节约通信成本，但其传输速度比并行传输低。其通信接线如图所示（图3-4）



图3-4 串口通信接线图

在本系统中，需要PN532和Android things系统进行通信，所以采用了波特率115200的串口通信。使用android things提供的UART通信API来完成模块之间的通信。

为打开指定端口号的UART串口连接，需要知道串口的名称。可以在初始化的时候，使用PeripheralManagerService的getUartDeviceList()函数来获取系统识别到的UART名称。

一旦知道目标名称，就可以使用PeripheralManagerService连接到该设备。与外围设备通信后，关闭连接以释放资源。 此外，在现有连接关闭之前，无法打开与设备的新连接。要关闭连接，请使用设备的close()方法。

在onstart()中打开device设备:

mUartDevice = manager.openUartDevice(UART\_DEVICE\_NAME);

在onstop()中关闭device设备:

if(mUartDevice != null){

mUartDevice.close();

mUartDevice = null;

}

数据帧通过UART发送的每个字符都包裹在一个数据帧中，该数据帧包含以下组件：（图3-5）



图3-5 串口通信帧格式

* 启动位 - 在发送数据之前，该行保持活动状态，持续1位持续时间的固定时间间隔，以指示新字符的开始。
* 数据位 - 表示数据字符的各位。 UART可以配置为在5-9个数据位之间发送以表示字符。 较少的位减少了数据的范围，但可以增加有效的数据传输速率。
* 奇偶校验位 - 可选错误校验值。 如果UART配置为偶校验或奇校验，则会在帧中添加一个额外的位，以指示数据位的内容是否为偶数或奇数值。 将此设置为none会从帧中删除该位。
* 停止位 - 在传输所有数据之后，线路被重置一个可配置的时间间隔，以指示该字符的结束。 这可以配置为保持空闲1或2位持续时间。

UART上的数据传输速率称为波特率。 它表示接收和发送的速度（以位/秒为单位）。 由于在通过UART连接的两个器件之间没有共享时钟，因此必须提前配置这两个器件，才能使用相同的波特率正确解码数据。

常用波特率包括9600,19200,38400,57600,115200和921600.该速率包括数据帧（开始，停止和奇偶校验位）的开销，因此有效数据传输速率将略低，并且根据 您配置的帧位数。

以下代码将UART连接配置为以115200波特，8个数据位，无奇偶校验和1个停止位（8N1）操作：

public void configureUartFrame(UartDevice uart) throws IOException {

uart.setBaudrate(115200);

uart.setDataSize(8);

uart.setParity(UartDevice.PARITY\_NONE);

uart.setStopBits(1);

}

传输输出数据:

public void writeUartData(UartDevice uart) throws IOException {

byte[] buffer = {...};

int count = uart.write(buffer, buffer.length);

Log.d(TAG, "Wrote " + count + " bytes to peripheral");

}

监听输入数据:

使用read()方法将输入数据从UART FIFO缓冲区拉入应用程序。 此方法接受空缓冲区以填充传入数据和要读取的最大字节数。 UART读取是非阻塞的，如果FIFO中没有可用的数据，则会立即返回。

UartDevice将在读取时返回FIFO中的可用字节数，并把读取的内容写入Buffer中。读取之前要确保buffer中所有数据都重置，循环读取UART，直到它返回值为-1(没有更多的数据)

public void readUartBuffer(UartDevice uart) throws IOException {

// 一次读取的最大数据量

final int maxCount = ...;

byte[] buffer = new byte[maxCount];

int count;

while ((count = uart.read(buffer, buffer.length)) > 0) {

Log.d(TAG, "Read " + count + " bytes from peripheral");

}

}

为了避免在缓冲区为空时不必要地轮询UART，用UartDevice注册一个UartDeviceCallback。 当有可用于读取的数据时，此回调将调用onUartDeviceDataAvailable()方法。当应用程序不再需要监听输入数据时，应该取消注册回调，以防止内存泄漏。

// 注册中断事件

mDevice.registerUartDeviceCallback(mUartCallback);

// 取消注册中断事件

mDevice.unregisterUartDeviceCallback(mUartCallback);

第四章 智能门禁系统软件设计

## 4.1 系统软件总体设计

本系统软件的总体设计如图所示（图4-1），主要分为三个模块，服务端控制模块、android开锁模块、android后台管理模块。

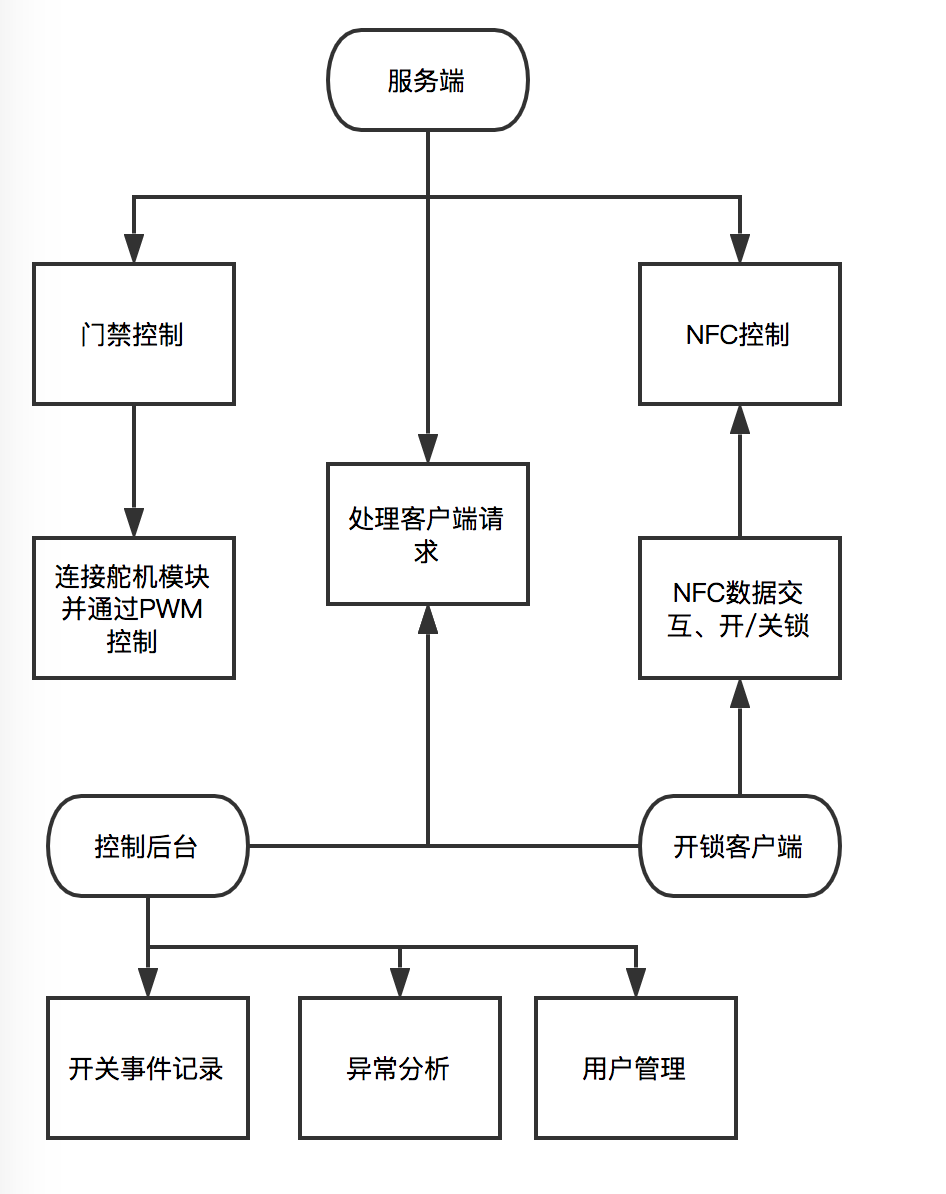


图4-1系统设计流程图

## 4.2 Android开锁模块设计

android开锁模块主要包含了开锁、关锁、NFC开锁、身份验证等功能。第一次进入app的时候需要注册账号，这个账号以后会作为开关锁的时候记录的凭据，方便后台管理。在实现注册之后，后台控制系统将会收到一条注册申请。

在后台通过注册申请之后，客户端需要设置一下解锁密码。解锁密码用九个点的手势密码来实现。在设置完密码之后，后台将记录下当前的密码，并存入android提供的sharedPreferences中。以后每次进入app都会有一个身份认证的界面。只有输入正确的密码才能进入开锁界面。

进入开锁界面后，可以选择远程开锁和关锁两个选项。因为是用的舵机模拟的门禁，所以这里的关锁只是为了将舵机复位来方便模拟。在实际生活中，只需要一个开锁选项，而关锁是由人工手动关锁。也可以贴近NFC天线模块，在贴近的时候将会跳出一个对话框来让用户确认是否解锁，如果选择了解锁即会向Android things后台服务器发送开锁请求，在服务器验证通过后即可让舵机旋转90度，来实现开锁的模拟。

4.2.1用户注册界面设计

在用户注册界面中(见图4-2)，用户需要填写账号、密码、申请理由三个选项。账号可以自行填写，但是在发送登录请求的时候会验证一下账号，如果账号是数据库中已经存在的账号则会注册失败，不通过。密码使用一个type为passwd属性的edittext，并使用hint属性来提示用户输入密码，使得界面更加友好。申请理由可以填写100字以内的申请理由，在注册的时候将会将这个理由发送到Android things后台服务器中，在后台管理界面收到请求后将会显示申请理由，并可以选择通过或者拒绝。

  
 图4-2注册界面

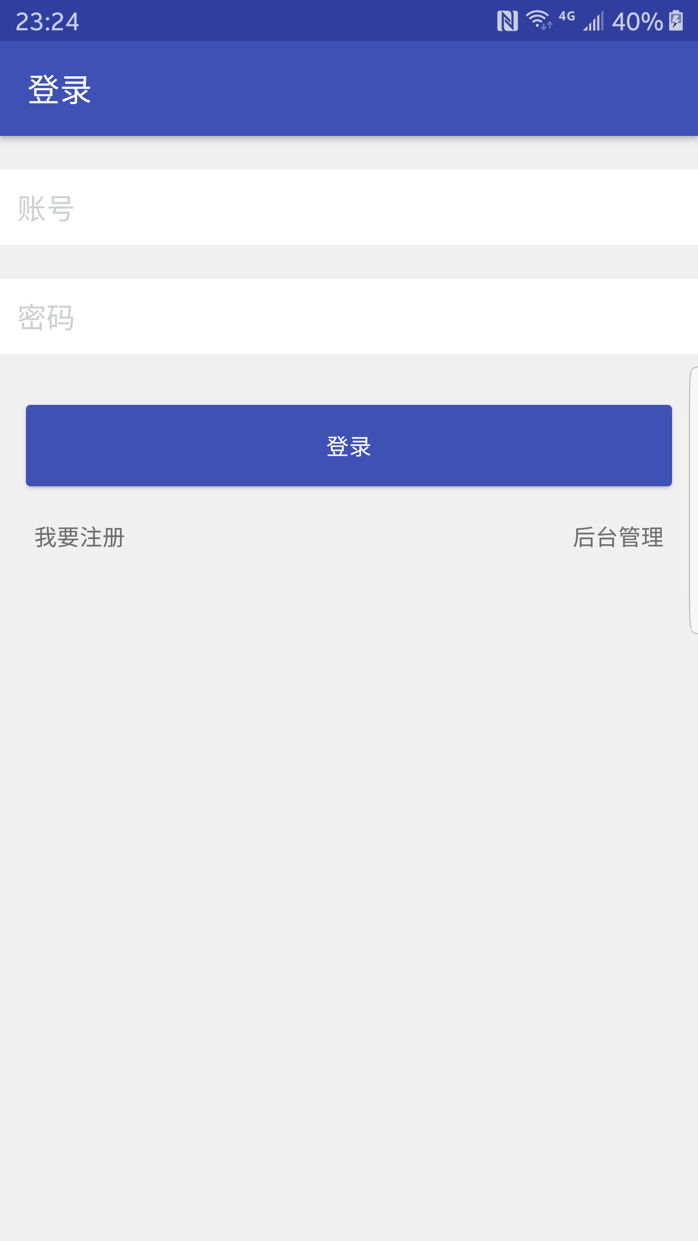


图4-3登录界面

在通过注册申请后，将会有一次手势密码的录入。采用九点手势。在很多安全性比较高的应用程序中，每次打开程序，都会有让用户输入密码，这些安全性比较高的程序，密码也会相对复杂，在手机上输入起来，就会不大方便，至少要切换一次输入法，对用户体验造成不好的影响，在移动互联网时代，用户体验至高无上，在这种情况下，手势密码不失为一个很好解决方案，既安全，又快捷，用户很容易记住，一个手势即可解锁，不用输入繁琐的密码。（图4-4）

1. 正上方的提示区域，用一个类(LockIndicator.Java)来实现，自定义view来绘制9个提示图标；

2. 手势密码绘制区域，用一个类(GestureContentView.java)来实现，它继承自ViewGroup里面, 添加9个ImageView来表示图标, 在onLayout()方法中设置它们的位置；

3. 手势路径绘制，用一个类(GestureDrawline.java)来实现，复写onTouchEvent()方法，在这个方法里面监听TouchEvent事件: ACTION\_DOWN、ACTION\_MOVE、ACTION\_UP事件，来绘制手势连接不同点之间的路径；

4. 9个点的对象，用一个类(GesturePoint.java)来实现，保存它的位置、状态、背景图片等相关信息；

5. 手势密码的获取，判断手指当前的位置，根据滑动路径经过的点，按顺序保存绘制的点的顺序，不能有重复的点。

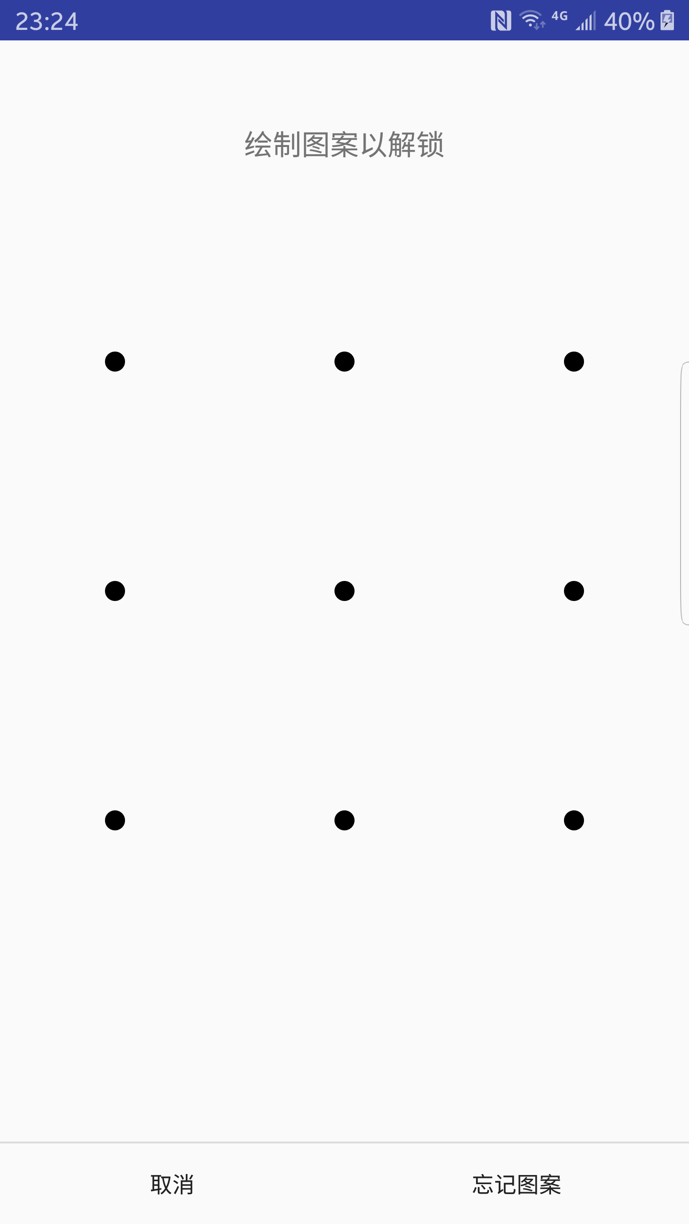


图4-4手势密码界面

在isPatternCorrect(List<PatternView.Cell> pattern)函数中判断手势密码是否正确，下面这行代码用来对比数据库中保存的用户手势密码sha1值和用户输入的密码sha1值，如果比对成功，则通过密码解锁进行下一步操作，不成功则会让用户再来一次。

TextUtils.equals(PatternUtils.patternToSha1String(pattern), dbSavedPatter);

在登录的时候如果密码输入错误还会向服务器记录错误的值和错误的时间，用来记录事件HttpUtil.getRetrofitInstance().create(ExceptionService.class)

.sendTrySuperAdminFail(System.currentTimeMillis()).enqueue(new Callback(){.....})

在设置密码界面会进行两次输入密码流程，如果两次输入的密码相同就通过校验，并把用户的密码加密后保存在本地数据库中。

protected void onSetPattern(List<PatternView.Cell> pattern) {

PatternSp.saveUser(userName, pattern);

}

4.2.2 NFC开锁设计

Android提供了NFCAdapter类来管理android的NFC功能。可以通过以下代码获取手机附带的NFC管理器。

this.nfcAdapter = NfcAdapter.getDefaultAdapter(context);

在onstart中判断获取到的nfcadapter是否为空则能判断该手机是否支持NFC。

if(nfcadapter != null){

if(nfcadapter.isEnabled()){

…}

}

通过PendingIntent包裹一个activity对象，来设置当系统发现一张nfc卡的时候要跳转的activity，相当于给系统创建了一个快捷方式。

pendingIntent = PendingIntent.getActivity(activity, 0,new Intent(activity.

getApplicationContext(),getClass()).addFlags(Intent.FLAG\_ACTIVITY\_SINGLE\_TOP), 0);

intentFilters=newIntentFilter[]{new IntentFilter(NfcAdapter.ACTION\_NDEF\_DISCOVERED)};

techList = new String[][]{new String[]{}};

setReaderMode();

然后给NFCadapter设置一个卡读取模式的监听，当有卡识别的时候将会有相应的回调。

nfcAdapter.enableReaderMode(activity,new NfcAdapter.ReaderCallback(),NfcAdapter.FLAG\_READER\_NFC\_A, Bundle.EMPTY);

在这个函数里会有一个tag对象的回调参数。

public void onTagDiscovered(Tag tag){}

之后拿到Tag对象就可以作为整个通信过程中的上下文。从TAG对象中可以获取java输入输出流，只要按照标准的输入输出流处理方式就可以完成整个通信过程。

这个开锁界面用三个cardview实现，第一个cardview存放用户信息，用户信息中会显示账户用户名信息和是否为管理员信息。第二个cardview存放开锁按钮，第三个cardview存放锁复位按钮。复位按钮用来测试时候使用，为了让舵机复位。（图4-5）

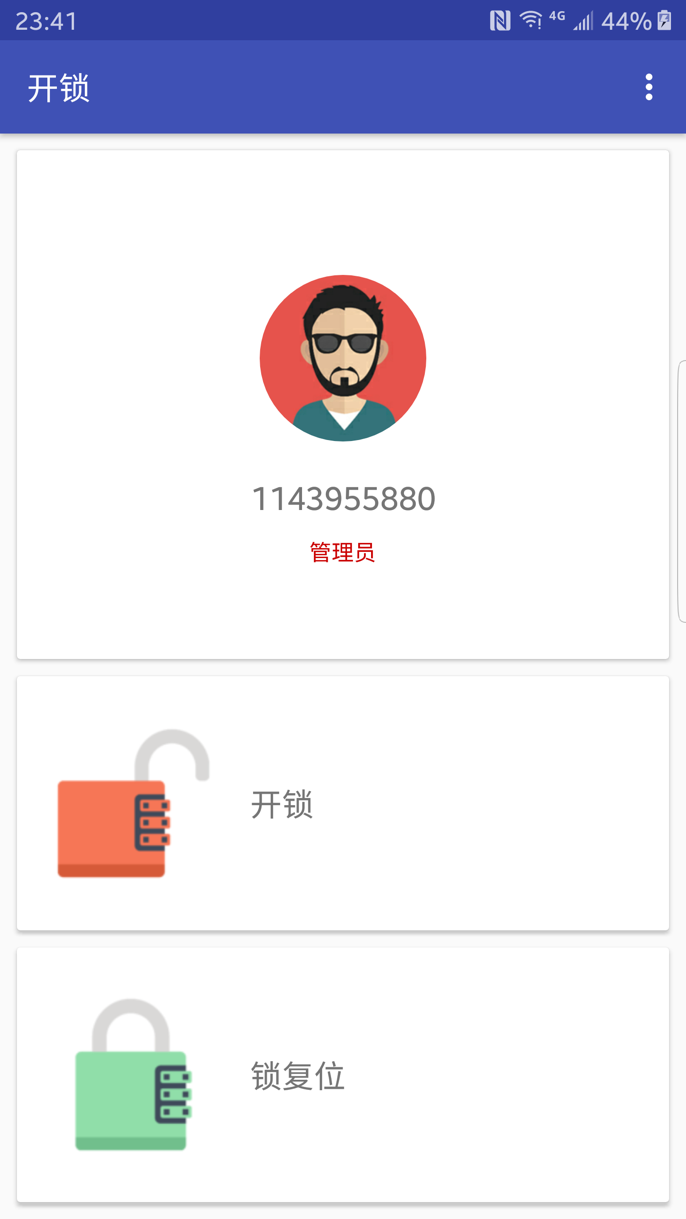


图4-5 开锁界面

## 4.3后台数据分析管理模块设计

**4.3.1 开锁事件记录**

每次客户端发送开锁请求都会被服务端记录下来，记录的字段有开锁的时间戳，开锁用户，开锁方式(远程开锁/NFC开锁)，然后会生成一个时间线，在后台展示。（图4-6）

整个开锁记录事件用RecyclerView + Cardview来展示，从后台拉取的有三个字段：开锁用户名、开锁时间戳、和开锁方式。

从服务器获取开锁事件list之后，将按照时间戳进行排序，最新事件在最上面，并通过SimpleDateFormat进行时间戳的格式化：

SimpleDateFormat dataFormater= new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

这样就可以以年-月-日 小时-分-秒的方式来对时间戳进行格式化，并在列表上展示。

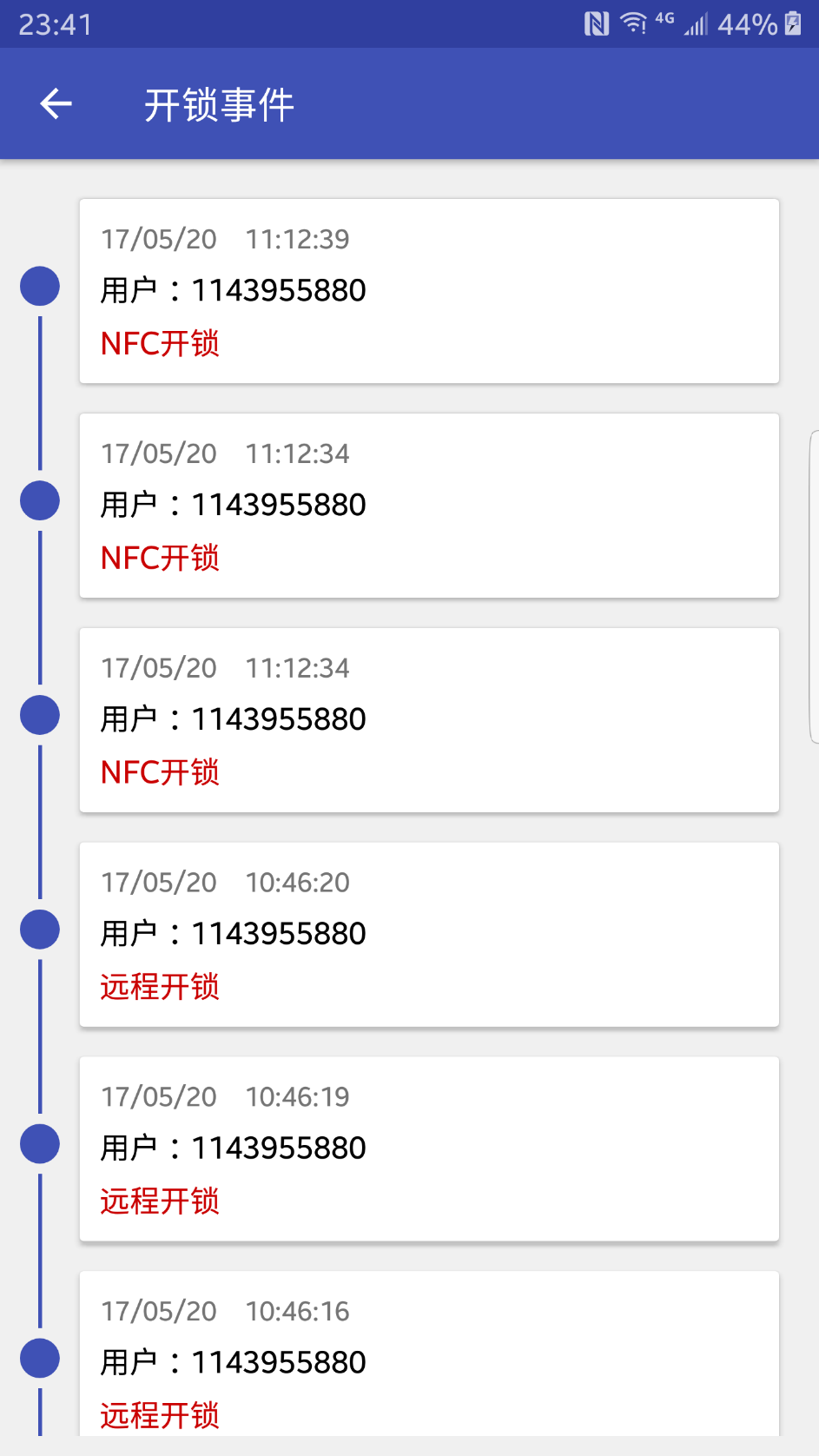


图4-6 开锁事件界面



图4-7 异常事件界面

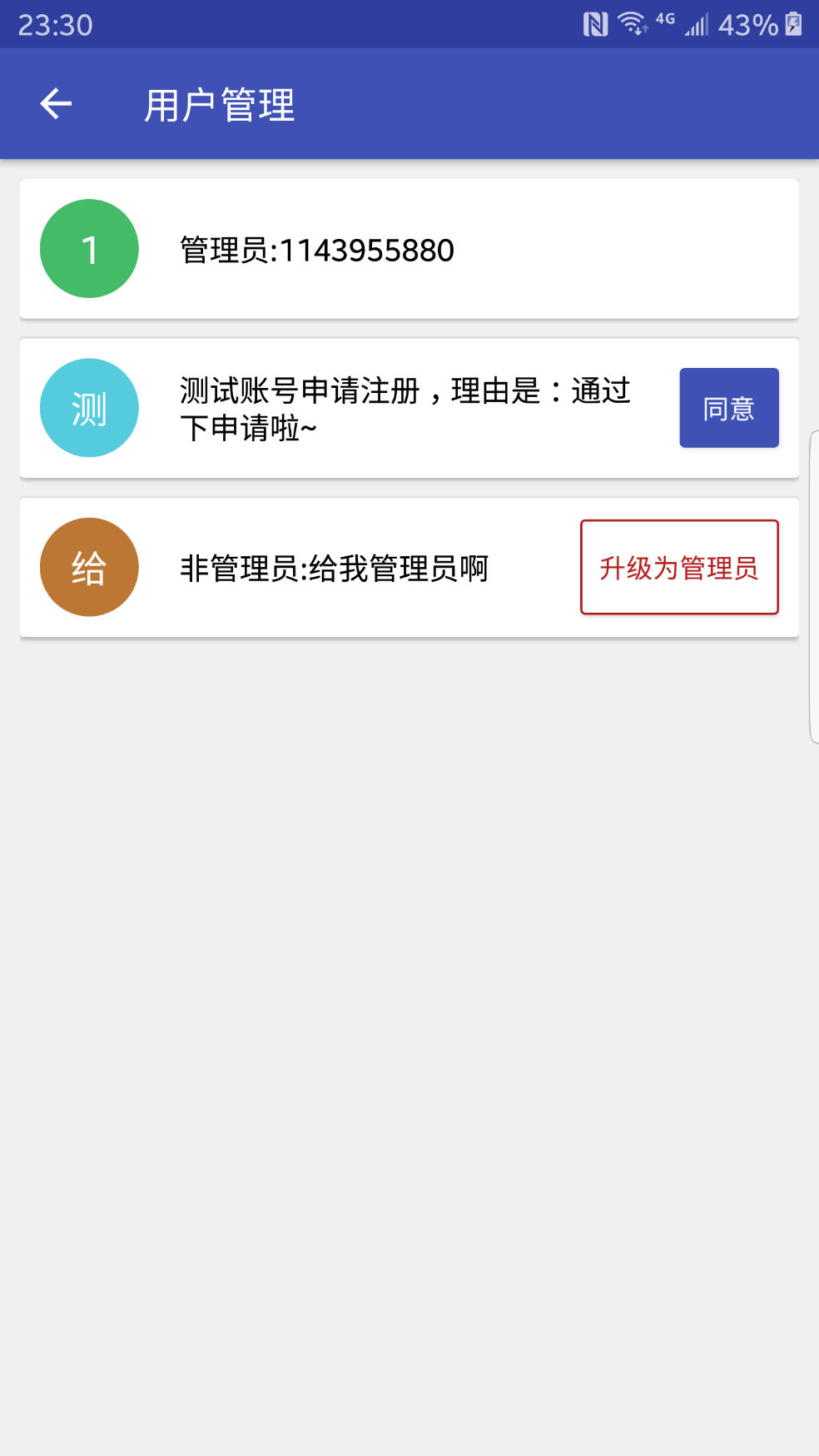


图4-8 用户管理界面

**4.3.2 用户管理**

用户管理只有管理员或者超级管理员才能管理和查看（图4-8）。在用户管理中，用户的状态分为三种，第一种是待通过申请的用户，这个用户可能在注册界面注册了一个用户，然后给后台发送了注册申请，在用户管理界面上可以看到申请的账号用户名和申请理由。管理员可以选择是否通过请求。如下图（图4-9）

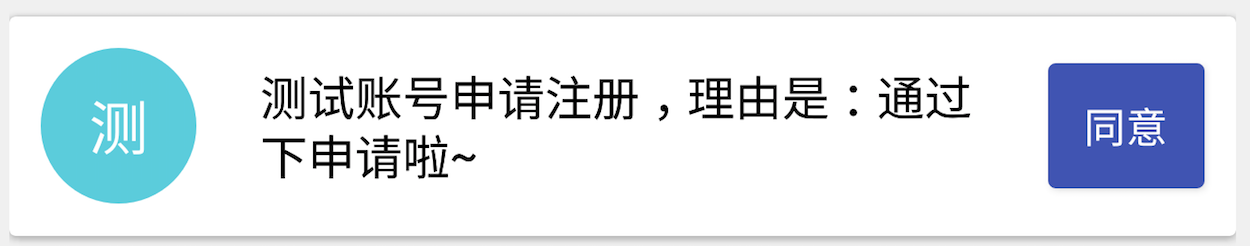


图4-9 待通过申请用户

第二种是已经通过申请的用户，会出现一个升级为管理员按钮，如果点击升级为管理员按钮，就可以把用户提权到管理员用户。如下图（图4-10）

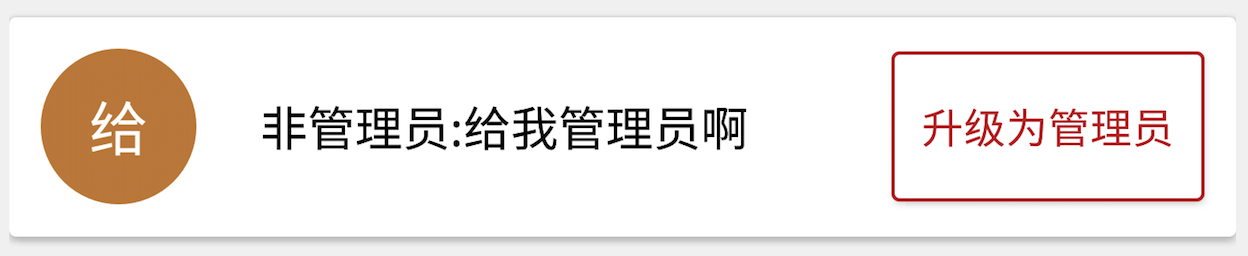


图4-10 待升级为管理员用户

第三种就是管理员用户，管理员用户有查看后台管理界面的权限，可以用户管理、异常事件查看、开锁事件查看，并且可以删除用户，拥有完全的管理员权限。（图4-11）

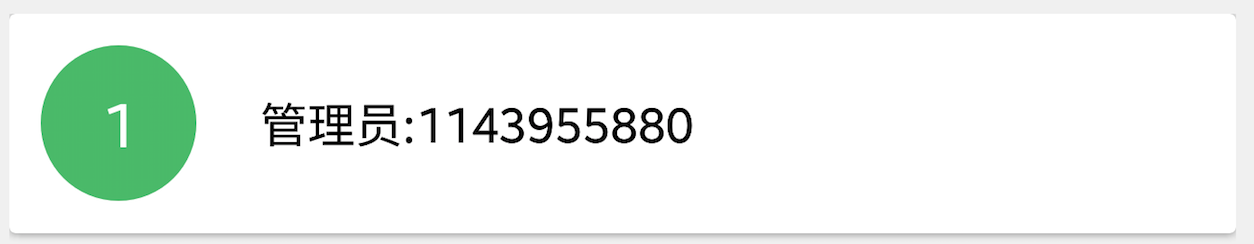


图4-11 管理员用户

**4.3.3 异常管理**

本系统处理三种异常事件，第一种是用户登录的时候，手势密码尝试失败，系统会记录下手势密码尝试的事件和当前用户名。（图4-12）



图4-12 尝试手势密码登录失败

第二种是在登录界面，账号输入正确，但是密码错误，系统会记录下密码错误的时间和错误的账号，最后在时间轴中展示。（图4-13）

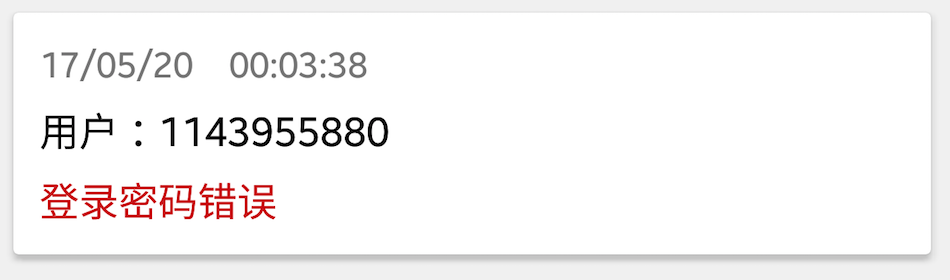


图4-13 登录密码错误

第三种是尝试超级管理员密码错误，系统会记录下尝试的时间，在时间轴中以下图的方式展示（图4-14）



图4-14 尝试超级管理员密码失败

在异常事件和开锁事件中，事件都以时间轴的形式排列展示，事件按照最新的在上面，老的事件在下面的形式排列展示。有着更好的用户体验。

## 4.4 Android things服务端模块设计

本服务端采用轻量级数据库Sqlite来保存客户端的一些信息，包括账户密码，用户开关锁事件等的记录。

Android 为了让我们能够更加方便地管理数据库，专门提供了一个 SQLiteOpenHelper 帮助类，借助这个类就可以非常简单地对数据库进行创建和升级。SQLiteOpenHelper 是一个抽象类，这意味着如果我们想要使用它的话，就需要创建一个自己的帮助类去继承它。SQLiteOpenHelper 中有两个抽象方法，分别是onCreate()和 onUpgrade()，我们必须在自己的帮助类里面重写这两个方法，然后分别在这两个方法中去实现创建、升级数据库的逻辑。

SQLiteOpenHelper 中还有两个非常重要的实例方法，getReadableDatabase() 和getWritableDatabase()。 这两个方法都可以创建或打开一个现有的数据库（如果数据库已存在则直接打开，否则创建一个新的数据库），并返回一个可对数据库进行读写操作的对象。不同的是，当数据库不可写入的时候（如磁盘空间已满）getReadableDatabase()方法返回的对象将以只读的方式去打开数据库，而 getWritableDatabase()方法则将出现异常。

SQLiteOpenHelper 中有两个构造方法可供重写，一般使用参数少一点的那个构造方法即可。这个构造方法中接收四个参数，第一个参数是 Context，这个没什么好说的，必须要有它才能对数据库进行操作。第二个参数是数据库名，创建数据库时使用的就是这里指定的名称。第三个参数允许我们在查询数据的时候返回一个自定义的 Cursor，一般都是传入 null。第四个参数表示 当前数据库的版本号，可用于对数据库进行升级操作。构建出SQLiteOpenHelper 的实例之后，再调用它的 getReadableDatabase()或 getWritableDatabase()方法就能够创建数据库了，数据库文件会存放在/data/data/<package name>/databases/目录下。此时，重写的 onCreate()方法也会得到执行，所以通常会在这里去处理一些创建表的逻辑。

开始的时候，系统会有一个默认的管理员账号root，之后管理员账号可以自行更改密码，还可以给普通用户提权升级到管理员账号，管理员有看开锁事件的权限、看异常分析的权限和对用户管理的权限。

创建表：

create table user (username long primary key, passwd varchar(50) not null, isAdmin BOOLEAN not null)

设置默认管理员账号：

insert into user (username, key, isAdmin) values (“root”, “admin”, true)

管理员账号和密码会加密并放到本地数据库中，在进入后台管理界面的同时要输入管理员手势密码。这个手势密码也可以更改。passwd中记录的是手势密码的走向的md5值，仅用来做登录时验证比对。

第五章 智能门禁系统硬件设计

## 5.1门禁控制-舵机模块

本系统门禁舵机模块使用SG-90舵机作为门禁控制模块，当舵机转到0度时，舵机为开启状态，180度时为关闭状态。舵机是一种位置（角度）伺服的驱动器，适用于那些需要角度不断变化并可以保持的控制系统。目前在高档遥控玩具，如航模，包括飞机模型，潜艇模型；遥控机器人中已经使用得比较普遍。舵机是一种俗称，其实是一种伺服马达。

控制信号由接收机的通道进入信号调制芯片，获得直流偏置电压。它内部有一个基准电路，产生周期为20ms，宽度为1.5ms的基准信号，将获得的直流偏置电压与电位器的电压比较，获得电压差输出。最后，电压差的正负输出到电机驱动芯片决定电机的正反转。当电机转速一定时，通过级联减速齿轮带动电位器旋转，使得电压差为0，电机停止转动。当然我们可以不用去了解它的具体工作原理，知道它的控制原理就够了。就象我们使用晶体管一样，知道可以拿它来做开关管或放大管就行了，至于管内的电子具体怎么流动是可以完全不用去考虑的。

舵机的控制一般需要一个20ms左右的时基脉冲，该脉冲的高电平部分一般为0.5ms~2.5ms范围内的角度控制脉冲部分。以180度角度伺服为例，那么对应的控制关系是这样的：

0.5ms--------------0度；

1.0ms------------45度；

1.5ms------------90度；

2.0ms-----------135度；

2.5ms-----------180度；

在本系统中舵机连接到树莓派的PWM1引脚，在android things平台中先获取pwm1管理器。

PeripheralManagerService manager = new PeripheralManagerService();

pwmController = manager.openPwm(PWM0);

设置舵机频率：

pwmController.setPwmFrequencyHz(50);

//先通过以下代码获取转动角度对应的脉冲时间

public float getDutyCycle(int degree) {

int length = 10;

int startDegree = -90;

int len = degree - startDegree;

return (((float) length \* ((float) len / 180f)) + 2.5f);

}

然后控制舵机转动角度：

pwmController.setEnabled(false);

pwmController.setPwmDutyCycle(getDutyCycle(degree));

pwmController.setEnabled(true);

最后在onstop中关闭舵机管理器

pwmController.close();

## 5.2 Raspberry Pi 3 运行平台

**5.2.1树莓派平台介绍**

树莓派（英语：Raspberry Pi），是一款基于Linux的单板机电脑。它由英国的树莓派基金会所开发，目的是以低价硬件及自由软件促进学校的基本计算机科学教育。

树莓派的生产是通过有生产许可的两家公司：Element 14/Premier Farnell和RS Components。这两家公司都在网上出售树莓派。

树莓派配备一枚博通（Broadcom）出产的ARM架构700MHz BCM2835处理器，256MB內存（B型已升级到512MB内存），使用SD卡当作存储媒体，且拥有一个Ethernet、两个USB接口、以及HDMI（支持声音输出）和RCA端子输出支持。树莓派只有一张信用卡大小，体积大概是一个火柴盒大小，可以运行像《雷神之锤III竞技场》的游戏和进行1080p视频的播放。操作系统采用开源的Linux系统如Debian、ArchLinux，自带的Iceweasel、KOffice等软件，能够满足基本的网络浏览、文字处理以及电脑学习的需要。分A、B两种型号，售价分别是A型25美元、B型35美元。树莓派基金会从2012年2月29日开始接受B型的订货。

**5.2.2 树莓派刷android things 操作系统**

（1）首先需要下载Android Things系统镜像，目前最新的android Things是预览版，可以到下面的地址下载相对应芯片的系统镜像。Android Things镜像地址 https://developer.android.google.cn/things/preview/download.html

（2）用sdcard读卡器连接sdcard到电脑上。

（3）解压刚才下载的系统镜像。

（4）在windows上进行烧录https://sourceforge.net/projects/win32diskimager/这个网站下载Win32 Disk Imager工具。运行win32 DiskImager ，最右边选择你sdcard的盘符（备注不要选到了其他的硬盘或移动存储设备，导致造成不必要的损失），然后选择镜像路径，再然后点击write等待写入结果。写入完后把sdcard插到树莓派上，插上网线（网线另一端连接路由），插上视频输出线连接到显示器，USB数据线连接电脑到树莓派，树莓派就会开始启动系统了，系统启动后，屏幕会出现局域网的ip。

（5）打开命令行工具，输入下面命令（ip -address就是显示器上android things系统显示的界面），不出情况的话就提示connected to <ip-address>:5555 这样就成功连接到树莓派。

**5.2.3 android things开发环境搭建**

添加依赖

（1）往app的模块里的build.gradle文件添加如下依赖

dependencies {

provided 'com.google.android.things:androidthings:0.4-devpreview'

}

（2）往 manifest .xml文件添加

<application ...>

<uses-library android:name="com.google.android.things"/>

...

</application>

这段声明是为了让系统识别此app作为androidthings启动app

（3）声明主Activity要把应用运行在嵌入式设备（本教程中的树莓派），必须包含一个 CATEGORY\_LAUNCHER 这样的Intent Filter，这样，才能在部署和调试应用的时候，Android Studio才能启动默认的Activity。具体需要在Manifest声明的设置如下。

<application

android:label="@string/app\_name">

<uses-library android:name="com.google.android.things"/>

<activity android:name=".HomeActivity">

<!-- Launch activity as default from Android Studio -->

<intent-filter>

<action android:name="android.intent.action.MAIN"/>

<category android:name="android.intent.category.LAUNCHER"/>

</intent-filter>

<!-- Launch activity automatically on boot -->

<intent-filter>

<action android:name="android.intent.action.MAIN"/>

<category android:name="android.intent.category.IOT\_LAUNCHER"/>

<category android:name="android.intent.category.DEFAULT"/>

</intent-filter>

</activity>

</application>

## 5.3 PN532 NFC模块设计

PN532是风火轮科技推出的一款 NFC 模组，主要是实现 NFC 手机与外部设 备之间进行 NFC 通信功能，外部设备通过串口连接 PN532模组，可以非常快速 添加 NFC 功能，本模组就是设备与 NFC 手机之间的一条通信渠道，对设备这一端 来说只是串口通信，要发送或接受数据直接操作串口就行，不需要关注 NFC 通信 协议栈，而对手机端来说，外部设备就是一个 TAG，Type A 的标准标签，操作起 来非常容易和简便，兼容性更好。PN532通过动态模拟卡方式实现与手机快速交互数据，可以交换任意长度 的数据，只要手机不离开天线区域。Pn532模组结构如图(图5-1)

PN532模组通过静态卡模拟方式，可以模拟成 NFC Forum 定义的标准功能 卡:蓝牙配对卡，TEXT 文本卡，网址标签卡。板载一个蜂鸣器驱动电路(可以外接蜂鸣器并通过串口命令控制它的开关) ，一个 LED 驱动电路(可以外接 LED 灯并通过串口命令控制它的开关) ，GPIO:4 个(P0.6/P0.5/P0.4/P3.1)，可以通过串口命令控制和读取，一个IRQ 中断口:当模组要输出串口数据前 1 毫秒，它会输出高电平，平时为低，可用于唤醒外部主控。

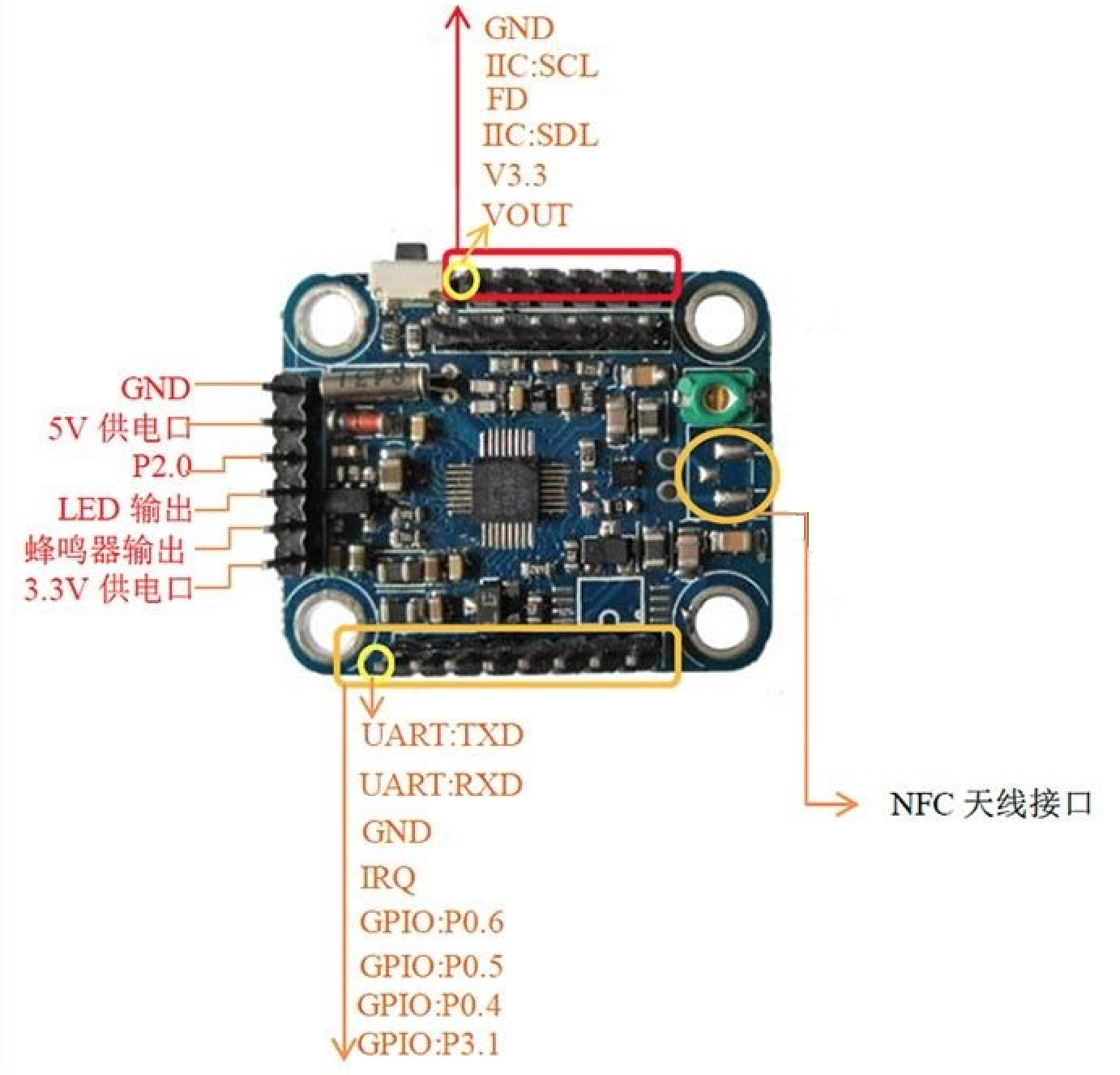


图5-1 pn532模组图

将PN532的UART串口和树莓派串口相连，tx->rx，rx->tx，gnd->gnd，vcc->vcc。这样就可以实现与树莓派的串口通信。

第六章 结论

未来的社会将会朝着便捷化、信息化和智能化的方向发展，随着近几年互联网的兴起，移动智能终端的技术水平不断提升，将会给人们带来更多无限的想象空间。互联网技术给人类带来了更加便捷的生活方式。NFC技术凭借着更强的安全性和可定制性，在未来将会以智能移动终端为载体融入人们的生活中。在金融、门禁、安防、用户鉴权等领域将发挥其作用。门禁系统与互联网技术的结合，将会成为智能家居领域的新的发展方向。

本系统针对国内外智能门禁系统做了深入分析，寻找发展瓶颈，确定了课题意义和目标方向，然后针对已有的智能门禁系统进行功能分析，确定了整个业务流程。接着针对业务流程做了整个系统的功能模块拆解、技术难点分析。确定使用的技术框架。包括服务器系统选取、通信格式定义、数据库设计。然后进行编码完成项目设计。最后进行整个框架的功能、稳定性和安全性测试。以确保其安全性和可用性。

本系统属于智能家居领域的一个分支，通过物联网接入更多的传统设备，有效的提升了整个系统的智能化特性。

本地通对于我来说是本科阶段一次非同寻常的做项目的经历。之前本科阶段的课程设计只是对某一知识点进行实践操作，而毕业设计综合了本科阶段所学的大部分知识，是对自身所学知识的一个极大的考验。通过这次项目经历，我接触到了很多新的东西，比如NFC技术，作为新的技术方向，正是毕业设计的这次机会使我对此进行了深入的了解。还有Android things智能物联网操作系统和android开发技术。学会了如何使用Google等搜索工具，android控件自定义、socket通信、串口通信、app安全等一些新潮技术。

在开发过程中，我遇到了很多棘手的问题，难解决的系统bug，感谢指导老师和同学们对我的帮助和支持。同时也学到了如何去查资料以及搜索引擎的使用，此次毕设经验将会让我在以后的工作中终生受益。

参考文献

致谢

不知不觉大学生活已经步入尾声，当初入学时的情景还历历在目。在本系统的设计中，从开题报告，到英文翻译、搜集资料、系统搭建、系统测试、毕业设计编写，诸多的环节都得到了指导老师和同学们的帮助和指导，在此表示衷心地感谢。

首先感谢我的指导老师-钱瑛老师。在初期毕业设计选课题的时候，我提出了几个创新的点，比如NFC通信模块的选取，由于NFC技术的不成熟，曾一度想放弃。这时候指导老师对我的想法给予了充分的肯定，并给我寻找相关的资料，对于系统中的很多技术细节进行了相应的指导。在开发过程中，每周的周会指导老师都会对我进行一些技术难点的指导，并解答在开发过程中遇到的困难，凭借丰富的经验对我系统中可能存在的漏洞进行了分析，督促了我对整个系统的不断地完善。

其次，要感谢学校图书馆提供的相关资料，每个星期我都会去图书馆翻阅相应的资料，比如安卓开发中遇到的一些问题，好多都能在图书馆的资料中查阅到。在这还要感谢Google公司开发的Android things智能物联网操作系统，使得嵌入式也可以拜托c/c++沉重的包袱，能让android 开发人员也能轻易地上手。感谢其在官网提供详细的教程和API文档，填平了嵌入式开发的填坑之路。让我这样的初级开发者也能很顺利的开发出完善的产品。

最后，我还要感谢学校四年来老师们的辛勤付出，让我打下了计算机科学与技术相关技术坚实的基础。感谢学校的培养，科学的课程体系，今日的收获离不开学校的精心栽培。最后再次感谢所有在毕业设计中对我提供帮助的老师和同学，感谢大家对本系统的测试和提出的相关建议。鼓励帮助我顺利完成毕业设计。

附录：英文技术资料翻译

**英文原文**

**中文译文：**