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**

|  |
| --- |
| 随着智能移动终端数量上的爆炸式增长和互联网的蓬勃发展，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。本系统将基于传统门禁的设计而重新设计，并对其加入互联网思想，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，本文对国内外智能门禁系统的研究现状以及市场销售现状进行了详细的分析和调研，接着对智能门禁系统需求进一步设计，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**

|  |
| --- |
| **Title Intelligent entrance guard system**  **Abstract**  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](#_Toc483843408)

[第一章 绪论 2](#_Toc483843409)

[1.1 课题研究意义 2](#_Toc483843410)

[1.2 国内外研究现状 2](#_Toc483843411)

[1.3 课题研究内容 3](#_Toc483843412)

[第二章 技术概览 4](#_Toc483843413)

[2.1 Android things嵌入式系统 4](#_Toc483843414)

[2.2 NFC协议介绍 5](#_Toc483843415)

[2.3 Android Studio开发平台 7](#_Toc483843416)

[第三章 通讯模块设计实现 8](#_Toc483843417)

[3.2 基于AES的通讯信息加密算法 10](#_Toc483843418)

[3.3 串口通讯设计 13](#_Toc483843419)

[第四章 智能门禁系统硬件设计 17](#_Toc483843420)

[4.1门禁控制-舵机模块 17](#_Toc483843421)

[4.2 Raspberry Pi 3 运行平台 18](#_Toc483843422)

[4.3 PN532 NFC模块设计 20](#_Toc483843423)

[第五章 智能门禁系统软件设计 22](#_Toc483843424)

[5.1 系统软件总体设计 22](#_Toc483843425)

[5.2 Android开锁模块设计 23](#_Toc483843426)

[5.3 后台数据分析管理模块设计 29](#_Toc483843427)

[5.4 Android things服务端模块设计 34](#_Toc483843428)

[第六章 测试 36](#_Toc483843429)

[6.1 Android things服务端模块测试 36](#_Toc483843430)

[6.2 Android app模块测试 36](#_Toc483843431)

[6.3 NFC开锁和远程开锁模块测试 37](#_Toc483843432)

[第七章 结论 38](#_Toc483843433)

[参考文献 39](#_Toc483843434)

[致谢 40](#_Toc483843435)

[附录：英文技术资料翻译 41](#_Toc483843436)

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。[1]

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

## 1.1 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。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

智能门禁是一种现代化的新型门禁系统，它将自动化技术和物联网技术进行融合，用智能移动终端进行管理，极大地方便了用户的使用。它用来解决重要部门的出入口安全和管理问题，适用于家庭、银行、酒店等地方。在互联网蓬勃发展的今天，安全门禁技术也得到了飞速的发展。智能门禁系统已经不再是单纯的钥匙开关，它已经成为安全和管理上的一种成熟的解决方案，它在各个领域发挥着重大作用。[2]

目前的智能门禁系统还不是很普及，在大部分小区中还是使用的传统门禁，国外的智能门禁系统有着相对较高的安全性，他们用了顶尖的生物识别技术，如指纹识别、虹膜识别等，先进的科技保证了门禁的安全性。而反观国内，一般都是以光学指纹做为门禁验证，比起同类的国外产品大概成本低30%左右。相应的系统稳定性和安全性比国外的稍差点。[3][4]

## 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最新的互联网智能系统，这使得互联网的竞争更加激烈。目前，微软正在向爱好者和原始设备制造商（OEM）推广其Windows 10 IOT系统。虽然亚马逊没有开发相应的设备系统，但该公司已经将AWS Lambda计算服务开放给物联网开发商。 亚马逊的云计算部门AWS正在与设备制造商合作，将Lambda Greengrass软件绑定到设备。 该软件是AWS Lambda的本地版本，可以安装到物联网设备和套件中。 Android Things使用与Android，Android框架和Google API相同的工具，使开发更容易。Android things的平台架构如下：(图2-1)

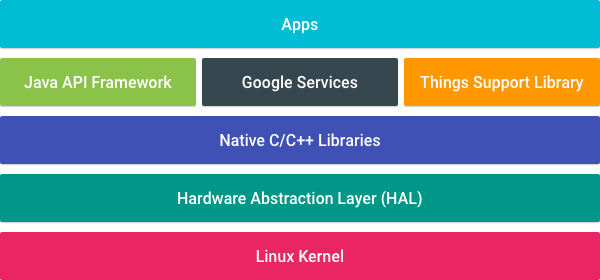


图2-1 Android things平台架构图

Android Things继承了Android核心框架层，并通过支持包提供了其他API。这些API允许应用程序与较新的硬件交互，但它们不存在于移动设备上。 Android Things平台仍然用于单个应用程序。系统应用程序不存在，您的应用程序将自动为您的用户提供绝妙的体验。

与Google的Brillo系统相比，Brillo使用C ++作为主要开发环境，而Android Things则面向所有Java开发人员，无论开发人员是否具有移动开发经验。开发人员可以使用物联网设备的工具包。 Android Studio是最受欢迎的Android集成开发环境，它是基于IntelliJ IDEA社区开源版本，支持Android things物联网系统开发。应用程序开发的生命周期与移动开发几乎相同。 Android Things系统支持原始Android SDK的一部分，并且不支持需要用户输入或需要身份验证凭据（如地图，搜索和登录）的应用程序界面（API）。与Android手机的OTA无线升级技术一样，开发人员可以通过接入Google已经在其产品和服务中使用的相同的OTA基础架构来推送Google提供的系统更新和定制应用程序升级。谷歌建议开发商在最初的原型开发过程中使用微型计算机“Raspberry 3”，Intel的微型可穿戴设备“Edison”和EnPico半导体的Pico平台。 Google将在Android Things正式发布之前添加更多身份验证设备。包括云消息服务Firebase，包括Google Cloud平台组件可以轻松地与Android的集成。开发人员可以使用云服务进云晕存储，状态管理和消息发送。 Android Things整合了物联网平台Weave。 Weave Server是一种云服务，用于处理设备注册，状态存储，命令传递以及与Google Assistant的Google服务集成。 Weave SDK将嵌入设备中进行本地和远程通信。

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。[5]

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。[6]

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。[7]

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是为Android平台程序开发而开发的集成开发环境。 发布于2013年5月16日Google I / O大会上，供开发者免费使用。 2013年5月发布早期版本，版本号0. 1。2014年6月发布版本0.8，进入beta阶段。 第一个稳定版本发布于2014年12月8日。Android Studio基于JetBrains IntelliJ IDEA社区版，专为Android定制，并可以在OSX，WIndows和Linux平台上运行、构建。

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是由操作系统提供的进程间通信机制。 套接字最初被翻译成网络插座。 很快，套接字被翻译成“套接字”，原因是：因为每个主机系统都有自己的进程命名过程，而且往往是不兼容的，所以在整个网络内硬进程是统一的名。 因此，每个计算机网络都应该引入媒体角色，整个网络的标准命名空间一致。 该标准名称在ARPA网络中称为套接字，并在许多其他计算机网络中称为信头。 更准确地说，进程之间的连接是由插座或信头

操作系统中，通常为用户态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）。应用程序可以利用网络套接字通过套接字接口进行数据交换。现代通用的套接字接口大多遵循Berkeley套接字标准。在套接字接口中，IP地址和通信端口形成套接字地址。远程套接字地址和本地套接字地址完成连接，加上使用协议，作为一对套接字对，可以交换数据与信息。例如，在同一台计算机上，TCP协议和UDP协议可以同时使用同一个端口，而不会相互干扰。根据套接字地址，操作系统可以决定将数据传送到特定进程或线程。这就像一个电话系统，一般通过电话号码加分机号来确定通话对象地址。

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）

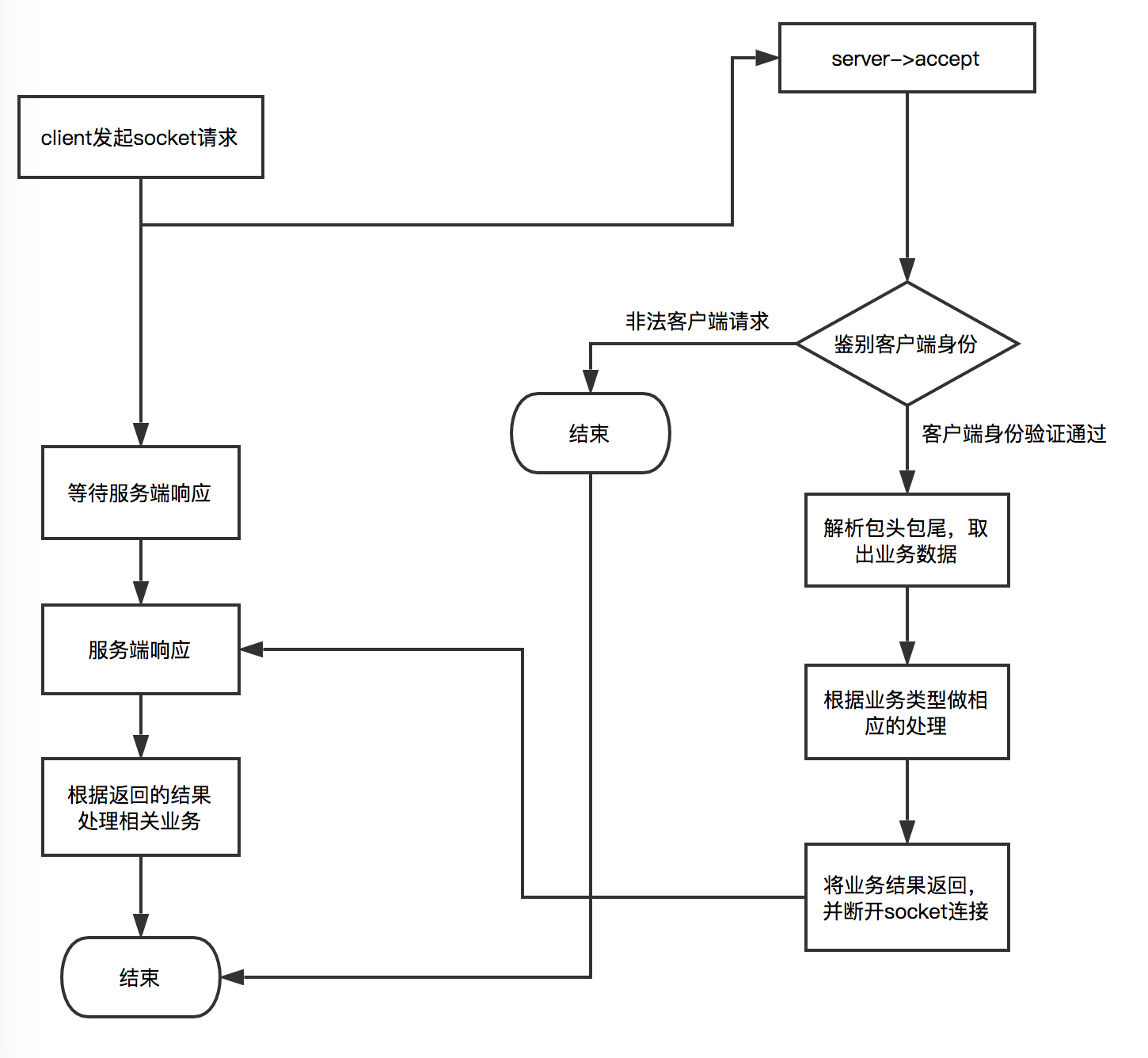


图3-1 socket通信流程图

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，Advanced Encryption Standard（缩写AES），在密码学里面又称为Rijndael加密技术，是美国联邦政府采用的块加密标准。截止2006年，AES加密标准已经成为全世界最被欢迎的对称密钥加密算法之一。

AES加密过程涉及四个操作：SubBytes，ShiftRows，MixColumns和AddRoundKey。中文分别为字节替换，行移位，列混淆，密钥加法。 AES加密和解密过程如图3-2所示。 从图中可以看出：

* 解密过程的每个步骤对应于加密的反向操作。
* 所有加解密的顺序完全相反，正是由于这些点（加密和解密每个步骤操作的反向），以确保解密算法可以正确恢复明文。通过种子密钥的密钥扩展来扩展每轮解密的密钥。算法中的16字节明文，密文和轮键由4×4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：

* jpg格式的图片中有一块存储区域叫Exif区，是专为数码相机设定的区域，可以记录数码照片的信息和拍摄数据，我们可以在Exif区存入加密后的信息，这样就可以实现安全无损地加密密文信息。
* png格式的图片中，可以在文件尾部区域后追加加密信息，这块区域被称为IEND区，在这段区域写入的信息不会影响android操作系统对图像的正常读取。不过，有利有弊，使用png格式的时候需要注意不要在编译的时候被编译器压缩掉，从而丢失加密信息。可以用把后缀名改成jpg等格式，这样就可以绕过编译器对图片的检查，但是在android操作系统中，不会收到后缀名的影响，操作系统会加载文件的头部来判断图片的格式，然后用相应的解码算法来加载图片，所以不会受到影响。
* 从图片的编码格式入手，比如根据你需要加密的信息自行构造出RGB信息，然后按照标准的图片编码格式构造出一张全新的安全图片，不过这样的图片看起来不太正常。你也可以在原来的图片中间按照某种自定义算法穿插一些数据。

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）

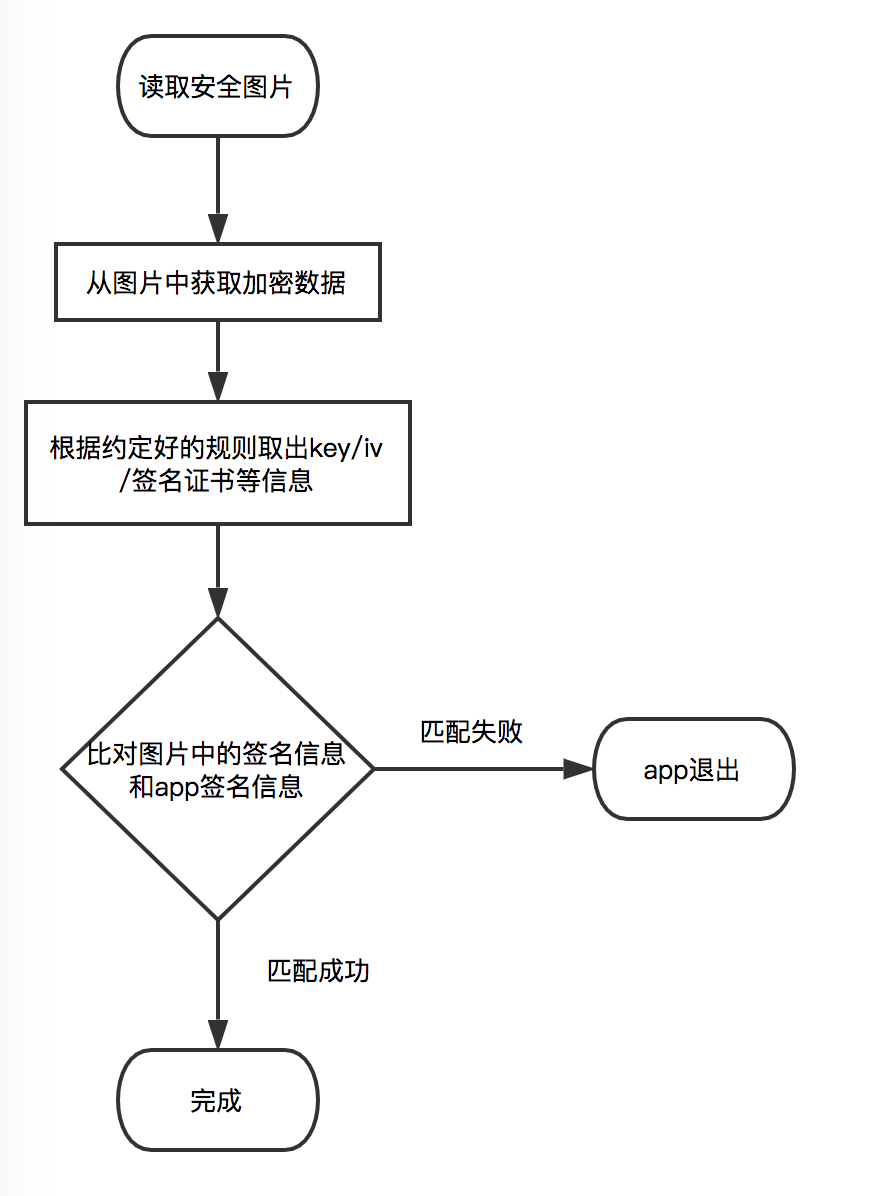


图3-3 安全图片验证流程

## 3.3 串口通讯设计

串行通信，是指外设和计算机之间通过数据信号线，地线，控制线等，按位数据传输的一种通信方式。 这种通信方式使用较少的数据线，可以节省长途通信中的通信成本，但其传输速度低于并行传输。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。

知道目标名称之后，就可以使用PeripheralManagerService来打开并初始化该设备。在结束语外围设备通信之后，需要及时关闭连接来释放资源。此外，在关闭现有连接之前，不可以打开与新设备的连接。如果需要关闭连接，可以使用设备的close方法。

在onstart()中打开device设备:

|  |
| --- |
| fgUartDevice= fgManager.openUartDevice(UART\_DEVICE\_NAME); |

在onstop()中关闭device设备:

|  |
| --- |
| if(fgUartDevice!= null){  fgUartDevice.close();  fgUartDevice = 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中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配置，才能使用相同的波特率正确编解码数据。

在串口通信中，常用的波特率有9600,19200,38400,57600,115200几种，这个速率里包括数据帧的开销，因此有效数据传输的速率将会稍微低一点。

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 {  myuart.setbaudrate(115200);  myuart.setdatasize(8);  myuart.setparity(Uartdevice.PARITY\_NONE);  myuart.setstopbits(1);  } |

传输输出数据:

|  |
| --- |
| public void writedata(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缓冲区输入应用程序。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。串口读取是非阻塞的，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中所有数据都被重置，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 {  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

|  |
| --- |
| feDevice.registeruartdviceallback(myCallBack); |

// 取消注册中断事件

|  |
| --- |
| feDevice.unregisterdeviceCllback(myCallBack); |

第四章 智能门禁系统硬件设计

## 4.1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

控制信号从接收器的通道进入信号调制芯片，以获得直流偏置电压。 它具有内部的参考电路，以产生20ms的周期和1.5ms的宽度的参考信号。 当然，我们不明白其具体的工作原理，知道其控制原则就足够了。 当我们使用相同的晶体管时，知道您可以在线上做开关管或放大管，以及如何不考虑管中电子的特定流量。

转向装置的控制通常需要大约20ms的时基脉冲，并且脉冲的高电平部分通常是在0.5ms至2.5ms的范围内的角度控制脉冲部分。 以180度角伺服为例，相应的控制关系为：

0.5ms----------0°

1.0ms----------45°

1.5ms----------90°

2.0ms----------135°

2.5ms----------180°

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);  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(); |

## 4.2 Raspberry Pi 3 运行平台

**4.2.1树莓派平台介绍**

树莓派（英文：Raspberry Pi）是一款基于Linux的单板计算机。它是为了使学生更有兴趣地学习计算机而开发的一款小型计算机。

树莓派配备了博通ARM架构BCM2835处理器，拥有256MB内存（B型已升级到512MB内存），使用SD卡作为硬盘（存储介质），还板载以太网网口，四个USB端口和HDMI（支持声音输出）和RCA终端输出的支持。树莓派体积只有信用卡一般大，体积大概和火柴盒差不多大，您可以在官方系统上运行游戏，比如雷电竞技场，你也可以在树莓派上播放1080p视频。你可以在树莓派上使用开源操作系统，如Android things, Debian，还可以以满足基本的上网冲浪，文字处理和计算机学习需求。

**4.2.2 树莓派刷android things 操作系统**

（1）首先你需要下载最新的Android Things刷机包，目前谷歌官网提供的最新的镜像是预览版，你可以Google搜索Android things Download就可以下载。

（2）插入SD卡

（3）用winrar等压缩工具解压第一步下载的刷机包。

（4）解压之后你会得到一个img镜像。你可以用win32DiskManager软件来烧录的的镜像，选好镜像之后，在右边的文件夹图标上点击，选择你SD卡的盘符，然后点击write，就可以把你的镜像烧录到SD卡中。

（5）打开路由器管理页，找到主机名为raspberry的主机，并记录下这个主机对应的ip地址。然后打开命令行工具，输入命令adb connect xxx.xxx.xxx.xxx，显示adb connected之后就表示你的计算机已经连接到树莓派上，这时候打开Android studio你就可以运行你的app啦。

**4.2.3 android things开发环境搭建**

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'  } |

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android things app

（3）你需要在Mainifest中声明以下代码，先是生命主activity，这个activity将在你app启动之后自动启动。这个activity需要包含一个CATEGORY\_LAUNCHER的Intent filter，这样Android studio就可以默认启动Manactivity。一个IOT\_LAUNCHER的Intent filter是为了让树莓派在启动的时候自动运行你这个app，这样就可以实现app自启动。

|  |
| --- |
| <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> |

## 4.3 PN532 NFC模块设计

PN532是一款由风火轮科技开发的NFC模块，主要为了实现NFC移动设备与外部设备之间的NFC通讯功能​​，PN532通过串口和外部设备之间进行数据通讯。可以实现卡模拟、卡的点对点模式。该模块是设备和NFC手机手机之间的通信通道，设备和外部设备只需要进行串行通信，直接发送或接收数据就行，不需要程序员了解NFC通信协议栈，而在移动端，外部设备是TAG类型一个标准的标签，操作非常简单，有着更好的兼容性。 PN532动态模拟卡通过方式实现与移动终端数据的快速交互，可以实现任意长度的数据通信，只要移动设备不离开天线区域。 Pn532模块结构如图（图4-1）

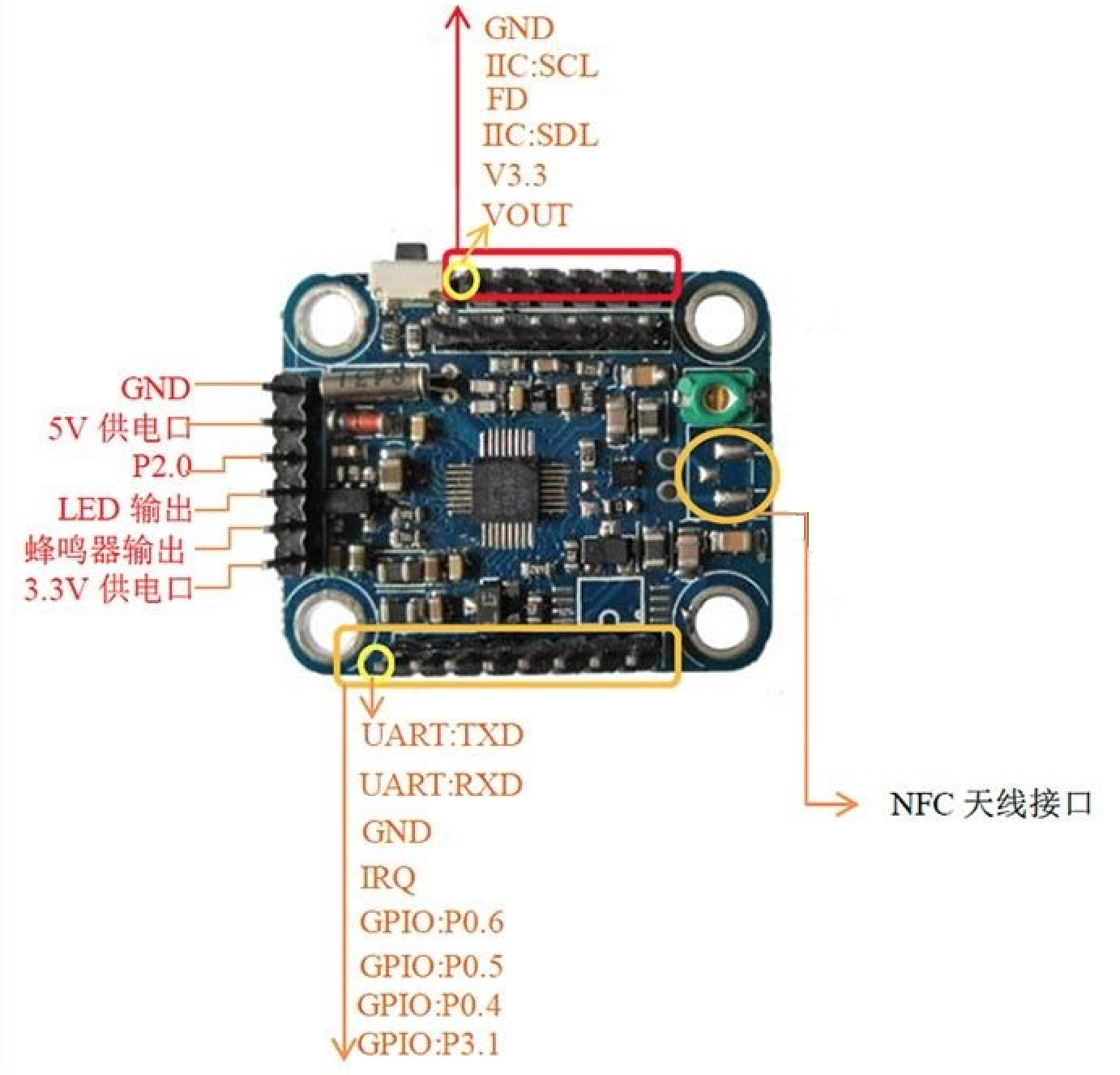


图4-1 pn532模组图

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。

第五章 智能门禁系统软件设计

## 5.1 系统软件总体设计

本系统软件的总体设计如图所示（图5-1），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。

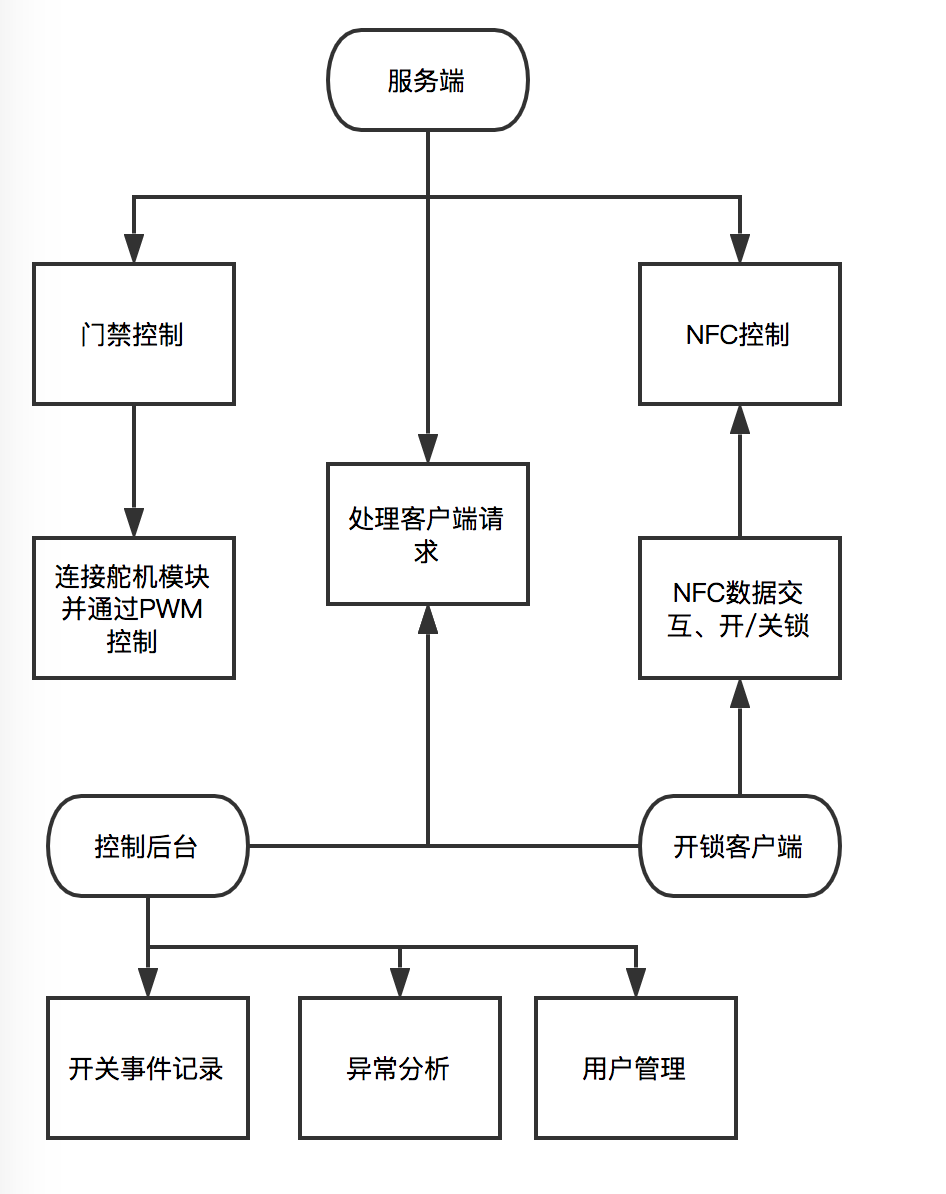


图5-1系统设计流程图

## 5.2 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。

在进入开锁界面之后，你可以选择远程开锁或者关锁两个选项。因为本系统是用的舵机来模拟门禁系统，所以这里的关锁只是为了把舵机复位，为了更方便模拟。然而在实际生活中，只需要一个开锁选项，关锁是由人工手动关锁。你也可以把手机贴近NFC的天线，在贴近的时候将会app跳出对话框来让用户确认是否需要解锁，如果选择了解锁就会向Android things后台服务器发送一个开锁请求，在Android things服务器验证通过之后就会控制舵机旋转90度，就可以实现开锁的模拟。

5.2.1用户注册界面设计

在用户注册界面中(见图5-2)，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5-2注册界面

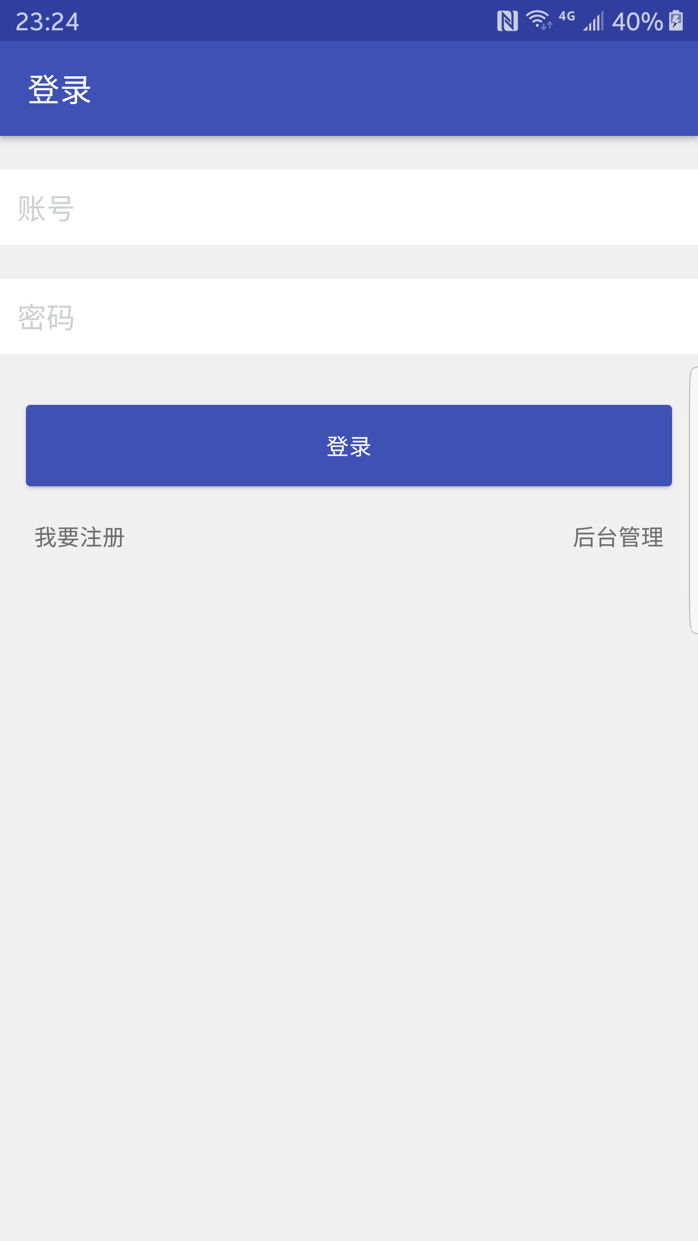


图5-3登录界面

在通过注册申请后，会有一个手势密码输入。使用九点手势密码。在许多高安全性的应用程序中，每次打开程序时，都会有用户输入密码，这些相对较高的安全程序，密码会比较复杂，进入app时会不是那么便捷，至少要切换输入法一次，用户体验很差，在移动互联网时代，用户体验是极其重要的，在这种情况下，手势密码是一个很好的解决方案，既安全又快捷，用户很容易记住，手势可以解锁，不要输入繁琐的密码。（图5-4）

* 在最上面的提示区域，用一个类(LockMessage.Java)来实现，用自定义view来绘制提示图标。
* 手势密码输入区域用一个类(UserGustPasswd.java)来实现，它继承自ViewGroup, 通过添加9个imageview来进行绘制, 然后在onLayout()方法中把他们放到该放的位置。
* 绘制手势路径，用DrawLineEvent.java来实现，需要复写onTouchEvent()方法，并且在这个方法里面实现监听TouchEvent事件: ACTION\_UP、ACTION\_DOWN、ACTION\_MOVE事件，这样就可以绘制不同点之间的连接路径。
* 九个点的记录对象，用 Point.java来实现，他可以保存状态、位置、背景图片、是否选中等相关信息。

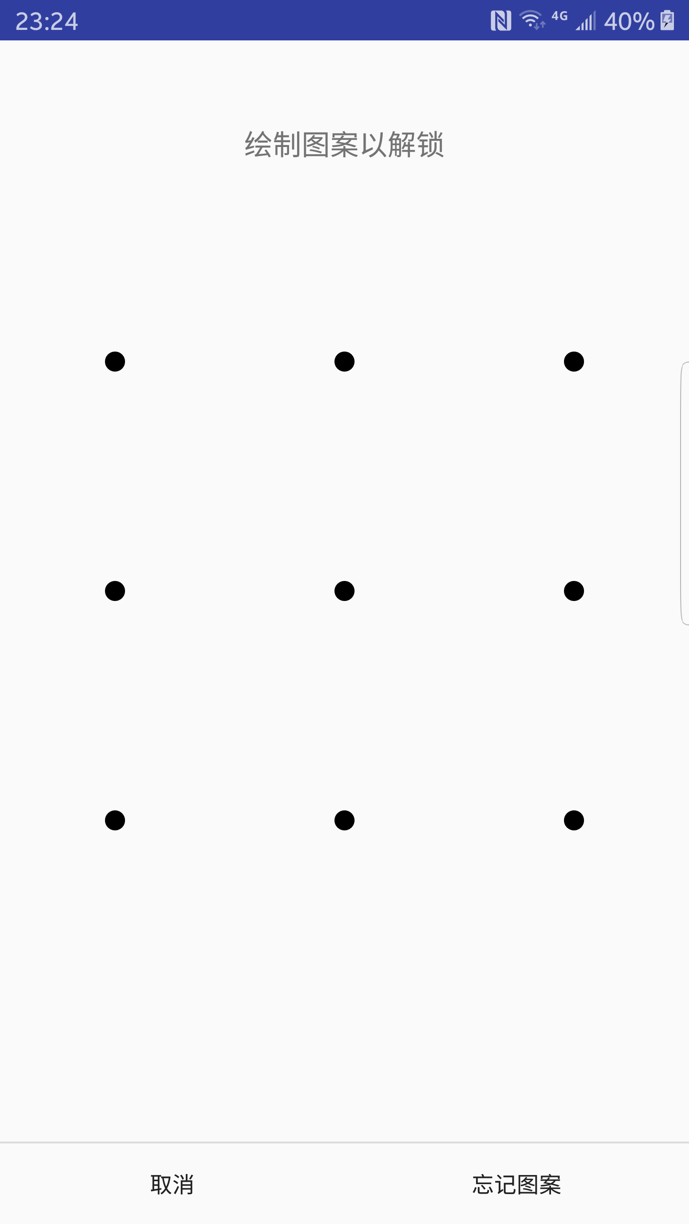


图5-4手势密码界面

最后在patternIsCorrect(List<ViewPattern.Call> userPattern)函数中判断用户输入的手势密码是否正确，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);  } |

5.2.2 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(getNfcAdapter()!= null){  if(getNfcAdapter().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。

|  |
| --- |
| pdItnt = 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5-5）

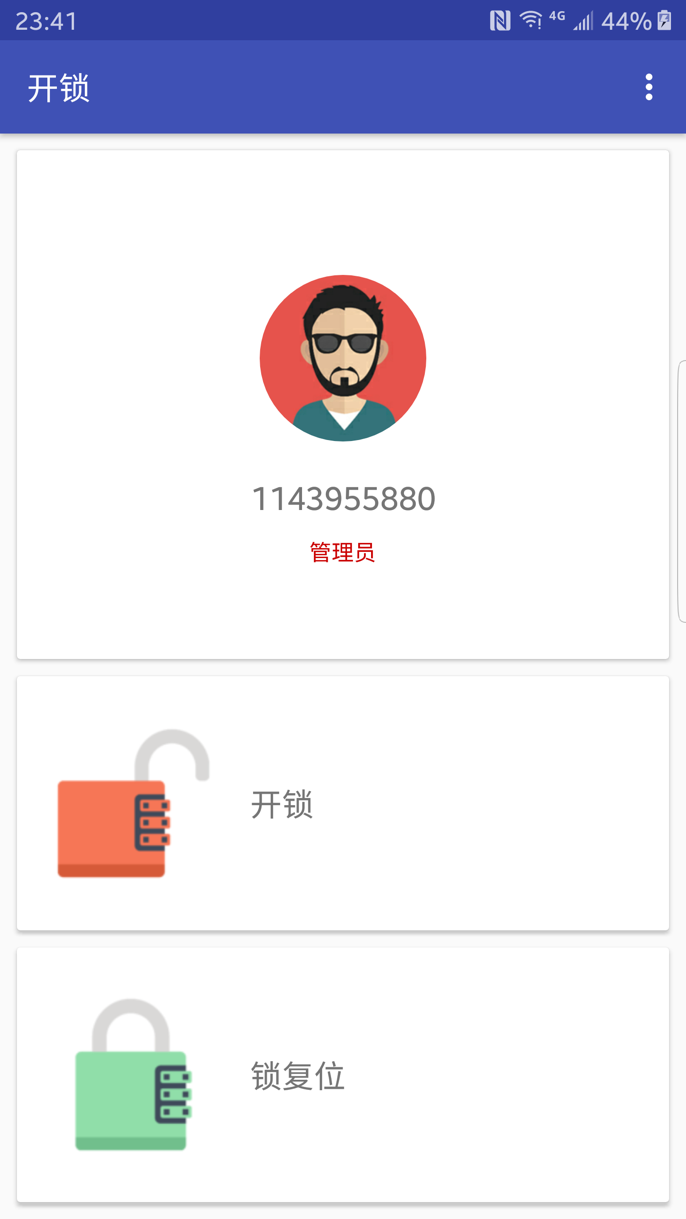


图5-5 开锁界面

## 5.3 后台数据分析管理模块设计

**5.3.1 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5-6）

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。

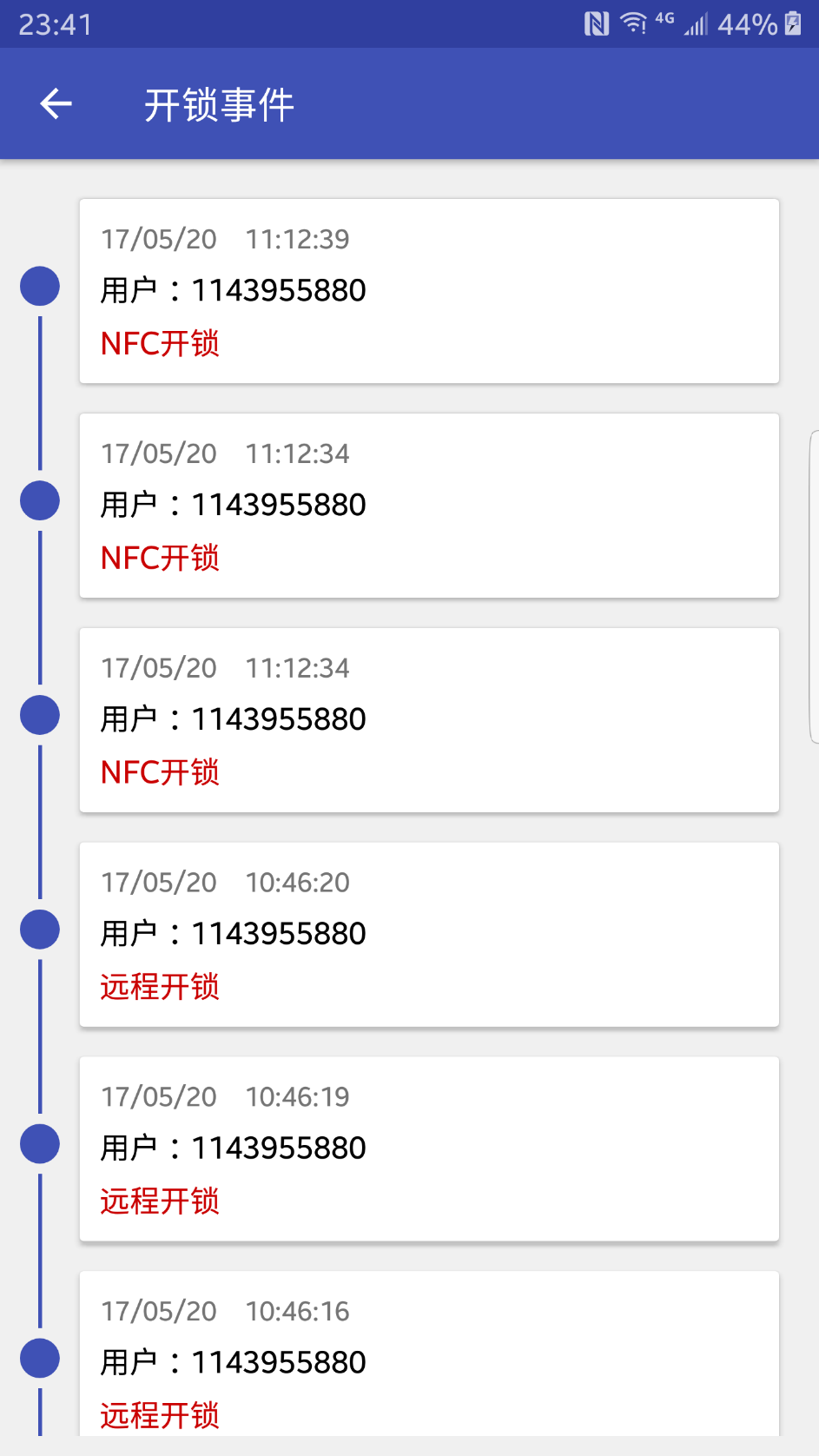


图5-6 开锁事件界面



图5-7 异常事件界面

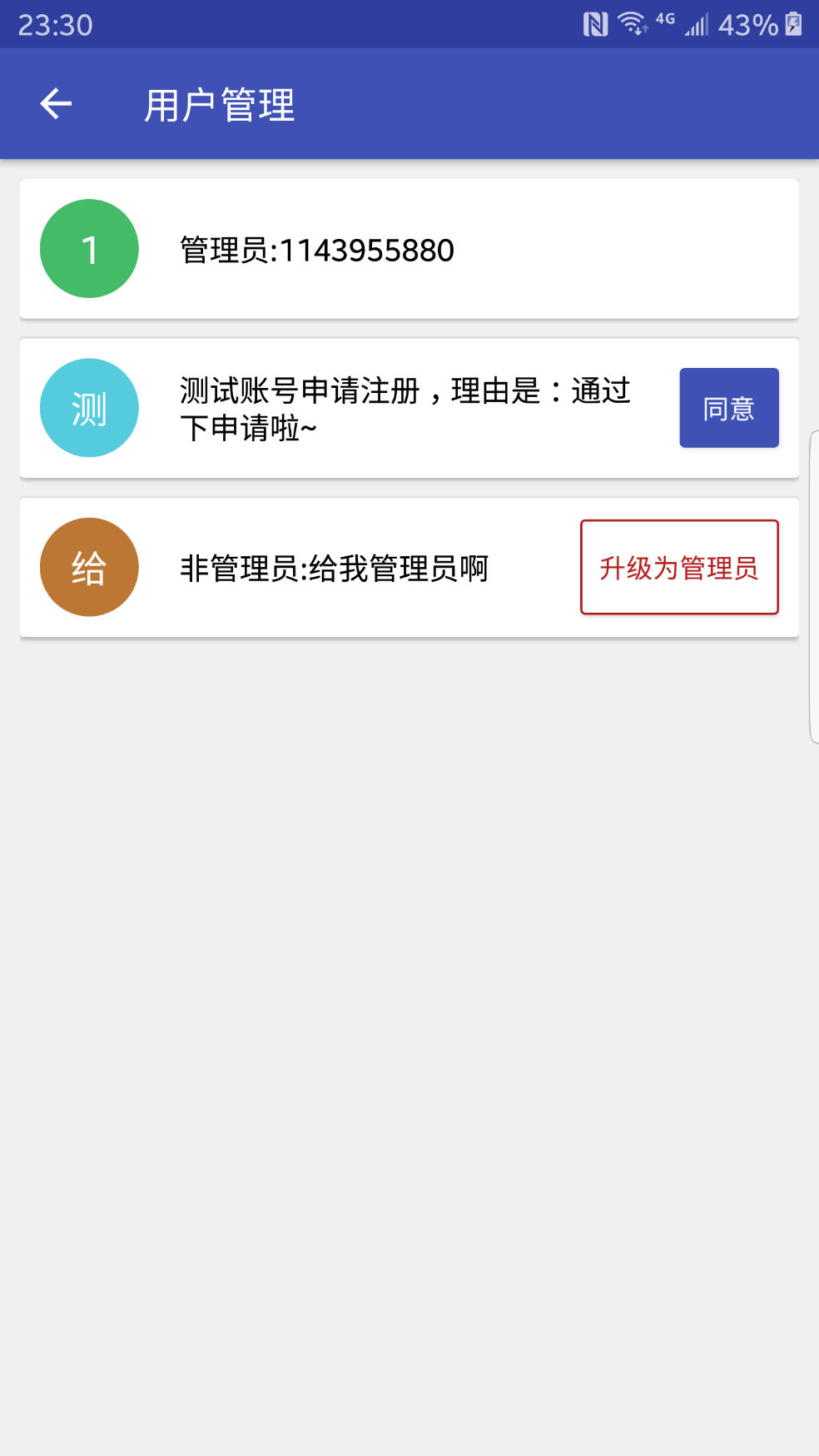


图5-8 用户管理界面

**5.3.2 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5-8）。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5-9）

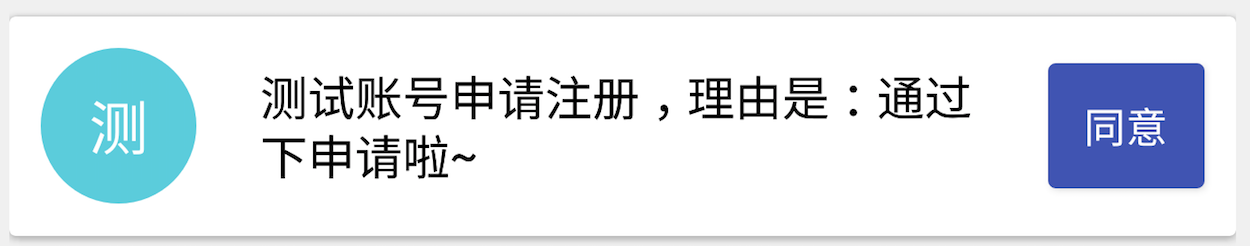


图5-9 待通过申请用户

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5-10）

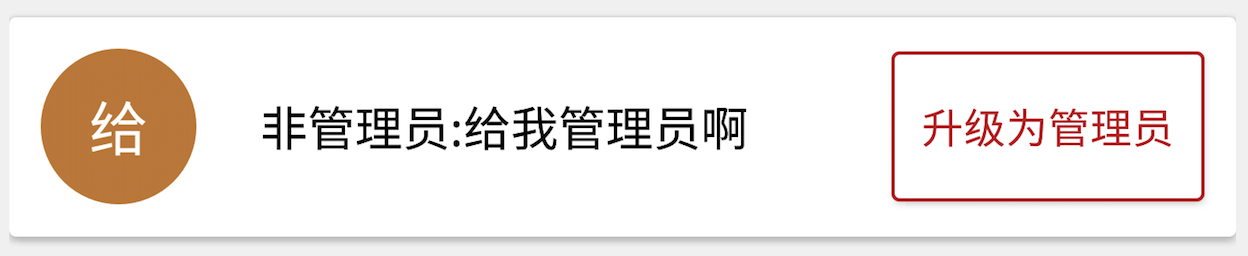


图5-10 待升级为管理员用户

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5-11）

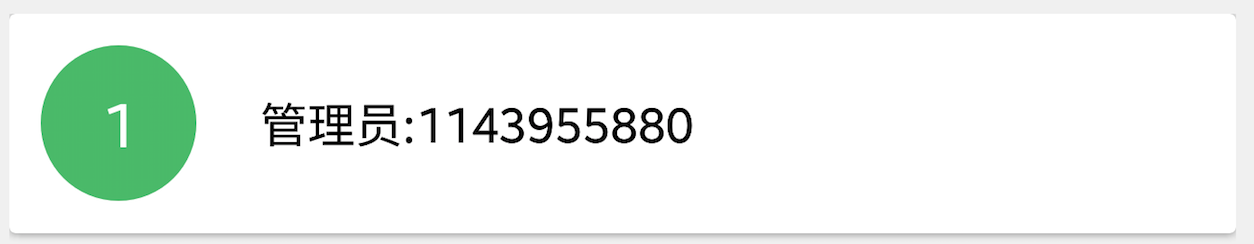


图5-11 管理员用户

**5.3.3 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5-12）



图5-12 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5-13）

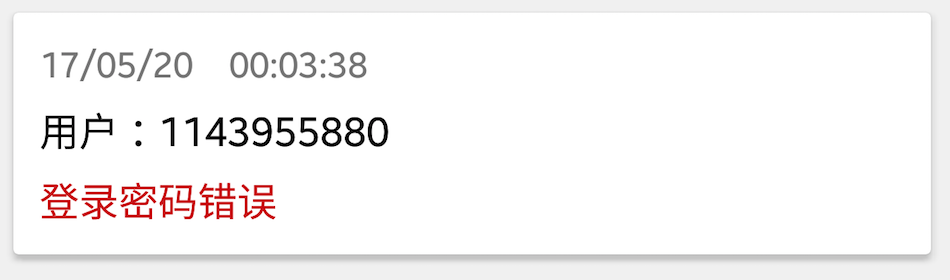


图5-13 登录密码错误

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5-14）



图5-14 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。

## 5.4 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为了让我们能够更轻松地管理数据库，特地为我们提供一个SQLiteOpenHelper帮助类，这个类可以非常简单的创建和升级数据库。 SQLiteOpenHelper是一个抽象类，这意味着如果我们要使用它，则需要创建一个帮助类来继承它。在SQLiteOpenHelper实体类中有两个抽象需覆盖的方法，即onCreate()和onUpgrade()，我们必须在自己的帮助内部重写这两个方法，然后在这两个方法中实现数据库的创建和升级逻辑。

SQLiteOpenHelper有两个用于重写的构造函数，一般使用的参数可以少一些。这个构造函数接收四个参数，第一个参数是Context，这是没有什么可说的，它必须让它在数据库上操作。第二个参数是数据库的名称，用于创建数据库的名称是此处指定的名称。第三个参数允许我们在查询数据时返回一个自定义的Cursor，通常是传入null。第四个参数表示当前数据库的版本号，可用于升级数据库。在实例化SQLiteOpenHelper(数据库助手类)实例后，调用getReadableDatabase()或getWritableDatabase()方法创建数据库，数据库文件将存储在/ data / data / <package name> / databases /目录中。 此时，重写onCreate()方法将被实现，所以通常这里要处理一些逻辑来创建表。

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。

第六章 测试

## 6.1 Android things服务端模块测试

**6.1.1 服务器开机自启动测试**

在本系统中，进入MainActicity的时候会自动把舵机复位到中间的位置，所以只要听到舵机转动的声音就表示系统已经启动。经测试，从树莓派上电到舵机转动大概需要1分钟20秒的时间，测试通过。

**6.1.2 服务器稳定性测试**

在android端上写一个socket连接->断开，再连接->再断开的小程序，并开启100个线程模拟压力测试，并随机发送一些脏数据、测试数据。并在服务端打印连接、请求日志，监控服务器稳定性。测试通过。

## 6.2 Android app模块测试

**6.2.1 注册模块测试**

用户注册栏所有edittext不能为空，否则无法完成注册流程，测试通过。

用户注册时服务器会返回该账号是否已经注册的信息，如果已经注册过的用户名不能通过二次注册。测试通过。

用户密码少于8位的时候显示注册失败，并提示用户名密码过短。测试通过。

**6.2.2 登录模块测试**

用户登录时需要填写账户名和密码，两项均不能为空，否则不能登录。测试通过。

用户登录的时候如果填写了错误（不存在）的用户名，则服务器返回账号不存在的消息，登录失败，测试通过。

用户登录时填写的账号正确但密码错误时，服务器返回密码错误的信息。测试通过。

**6.2.3 用户管理模块测试**

用户注册后会在后台管理页面会显示请求信息，并在最右边有同意按钮，点击后变成“升级成管理员”按钮，测试通过。

用户注册同意后，按钮变成“升级成管理员”，点击按钮，用户信息变成“管理员：xxx”，测试通过。

长按用户信息会跳出删除用户的按钮，点击删除后会删除用户的信息，测试通过。

**6.2.4开锁事件模块测试**

用户在用NFC开锁后，服务端将会记录NFC开锁事件和用户信息，在点击开锁事件按钮后会显示该事件，测试通过。

用户在用远程开锁后，服务端将会记录远程开锁事件和该用户的用户名信息，在点击开锁事件按钮后会显示该事件，测试通过

**6.2.5 异常事件模块测试**

用户尝试超级管理员密码失败后，服务端将会记录下用户名信息和尝试时间，测试通过

用户在登录时密码错误后，服务端将会记录下用户名信息和尝试时间，测试通过。

用户在尝试手势密码解锁失败后，服务端将会记录下用户名信息和失败时间，测试通过。

## 6.3 NFC开锁和远程开锁模块测试

当手机靠近NFC天线后，将会自动向服务器发送NFC开锁请求，并完成开锁，然后在开锁事件中记录下本次开锁事件，测试通过。

当用户点击“开锁”按钮后，将会自动向服务器发送远程开锁的请求，并完成开锁，然后在开锁事件中记录下本次开锁事件，测试通过。

第七章 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

[01] 夏文豪. 浅谈智能门禁系统的探索与应用. 电子制作，2013 (23)

[02] 胡丽. 新型现代化门禁安全管理系统概述. 金卡工程，2002(5):80-83

[03] 王伟贾慧娟. 门禁系统时间统计的实现. 福建电脑, 2006(8):152-152

[04] 曹建清，马文新，张旭璟. 一种智能门禁系统的设计. 微计算机信息, 2007(29):30-31

[05] 王刚. 基于非接触IC卡的智能门禁系统设计与研究. 上海海事大学, 2007

[06] 杨晓娟. NFC在手机中的应用——NFC芯片用于世博刷卡及蓝牙识别. 移动通信，2011(11):70-74

[07] 陈文正. 基于Android系统的NFC技术实现. 大连理工大学， 2012

[08] 王洋．Java就该这样学． 电子工业出版社，2013.6

[09] 叶核亚．JAVA程序设计实用教程（第四版）．电子工业出版社，2013

[10] 黄宏程，胡敏，陈如松．Andriod移动应用设计与开发．北京：人民邮电出版社，2013

[11] 孙更新，邵长恒，宾晟．Andriod从入门到精通．北京：电子工业出版社，2007

[12] 余成锋，李代平，毛永华. Android3.0 内存管理机制分析. 计算机应用与软件，2013:55-80

[13] 李刚. 疯狂 Android 讲义[M]. 北京：电子工业出版社，2013:25-42

[14] Sayed Y. Hashimi ．Pro Android 2． Apress，2010.1

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：**