



本科毕业设计（论文）

基于web的教室管理系统设计

**学 院 物理与光电工程学院**

**专 业 光电信息科学与工程**

**年级班别 2016级（2）班**

**学 号**  **3116007113**

**学生姓名 陈艺豪**

**指导教师 刘汉瑞**

2020 年 5 月

**基于web的教室管理系统系统设计 陈艺豪 物理与光电工程学院**

**摘要**

随着计算机技术的发展，网络技术和移动互联网技术逐渐成熟，各种管理事务的信息化和自动化应用也越来越多。教室是特别宝贵的资源，特别是在高校中，多媒体教室的设备和环境都是学生学习的最佳选择之一，因此多媒体教室的管理显得尤为重要，因为这关系到教室资源的合理利用。

本论文在对广东工业大学的教室编号进行采集之后，围绕着教室的晚上利用信息，针对广东工业大学多媒体教室的运行现状，运用web（World Wide Web）技术开发了一套教室管理系统。本系统注重于实现学生和管理员都可以查看本周教室的利用信息，如上课、空闲、社团活动等，并分别采用不同页面对教室进行管理。该系统采用前端和后端分离的方式，后端语言采用java语言开发接口，为前端提供数据；前端采用vue.框架+ elementUI实现功能及展示页面。

**关键词：**web技术，教室管理系统，java，vue+elememtUI

**Abstract**

# 目 录

[目 录 4](#_Toc37516353)

[1绪论 6](#_Toc37516354)

[1.1课题的研究背景和研究意义 6](#_Toc37516355)

[1.2 web技术概述 7](#_Toc37516356)

[1.2.1 web技术的概念 7](#_Toc37516357)

[1.2.2嵌入式系统发展现状 7](#_Toc37516358)

[1.3门禁系统概述 8](#_Toc37516359)

[1.3.1门禁系统简介 8](#_Toc37516360)

[1.3.2门禁系统国内外发展现状 10](#_Toc37516361)

[1.4本论文的研究工作和内容 11](#_Toc37516362)

[1.5本论文的内容和结构安排 12](#_Toc37516363)

[2嵌入式门禁系统软硬件方案选取 12](#_Toc37516364)

[2.1系统硬件方案的选取 12](#_Toc37516365)

[2.1.1门禁系统控制器 12](#_Toc37516366)

[2.1.2RFID通信协议读写模块 13](#_Toc37516367)

[2.1.3图形用户界面选择 14](#_Toc37516368)

[2.2门禁系统软件实现的关键技术 14](#_Toc37516369)

[2.2.1系统交叉编译环境的搭建 14](#_Toc37516370)

[2.2.2交叉编译工具链简介 15](#_Toc37516371)

[2.2.3应用程序的移植 15](#_Toc37516372)

[2.2.4Linux下多线程编程 17](#_Toc37516373)

[2.2.5Linux字符设备驱动编程 18](#_Toc37516374)

[2.2.6系统GUI制作 21](#_Toc37516375)

[2.2.7RFID通信协议 23](#_Toc37516376)

[2.2.8Linux系统编程 25](#_Toc37516377)

[3嵌入式门禁系统主控系统软件设计 27](#_Toc37516378)

[3.1系统设备驱动程序 27](#_Toc37516379)

[3.2门禁系统程序编写 32](#_Toc37516380)

[3.2.1门禁系统程序界面构建 33](#_Toc37516381)

[3.2.2界面按键触发操作类 33](#_Toc37516382)

[3.2.3RFID数据采集类 34](#_Toc37516383)

[3.2.4门禁控制线程类 40](#_Toc37516384)

[3.2.5密码信息链表类 46](#_Toc37516385)

[3.2.6非接触卡卡片信息链表类 47](#_Toc37516386)

[3.3门禁系统主程序 48](#_Toc37516387)

[4软件测试与应用 50](#_Toc37516388)

[4.1系统总体测试 50](#_Toc37516389)

[4.2总结 50](#_Toc37516390)

[总结与展望 51](#_Toc37516391)

[参考文献 53](#_Toc37516392)

[致谢 54](#_Toc37516393)

# 1绪论

## 1.1课题的研究背景和研究意义

随着信息时代的到来，当代信息技术的不断发展，网络技术也逐渐成熟，其应用也越来越广泛。近年来，由于web技术的不断发展和成熟，各种功能的网页不断出现，同时也不断改变着我们原有的生活方式，从原本需要一个APP应用才能实现的复杂系统，到现在只需要一个网页或者小程序就能实现其全部功能，这不仅让网页和小程序替代繁琐的APP应用，也大大提高我们的生活效率和办事效率。

在四年的大学生活中，我熟悉到多媒体教室在白天的时候一般用于教学活动，晚上则大多是用于学生自习和社团活动（少数教室用于上课）。白天时有教务系统相应的页面可查看当前教室的信息，这给我们查找空闲教室带来了极大的便利。但是到了晚上，我们只能查到教室的上课信息，却查不到社团活动的信息，因此经常出现在一间空教室学习到一半就有人进来开展社团活动的情况，这无疑给一些热爱学习的人带来极大的困扰；而且社团活动一般比较吵闹，这也给周边上课和自习的教室带来影响。因此，假如对教室资源的利用不规范，会给高校教室使用效率带来严峻的挑战，甚至会让学生彼此产生矛盾进而引发斗殴等更严重的后果。同时作为一个学生，我希望对教室使用情况有一个比较清楚的认识，便于我安排学习或社团等活动，但苦于没有相应的渠道；而对于教师来说，想了解目前教室的安排以及教室的具体使用情况，也没有比较好的解决办法。

基于广东工业大学的基本情况——教学楼数目多，每幢教学楼层数不同，每层楼教室数目也不相同；同时，由于硬件条件的限制，没有更好的地方可供学生进行社团活动，只能占用教室。在当前信息时代的背景下，学生仍然采用跑腿的方式申请社团活动，而后又需经过相关人员进行审批，这对无疑是对学生时间和精力的巨大浪费。当然，教学活动作为教室的主要功能，如何将规模庞大的教室资源合理地分配，做到应用尽用，既不浪费教室资源又不影响彼此进行活动，并且能够提供给学生和教师查询使用，这将是一个比较大的难题。

因此，为了充分利用广工的多媒体教室资源，提高基础设施的管理效率，使教学管理更加信息化、自动化、高效化；同时，既方便教师和学生对当前的多媒体教室情况有一个充分的了解和快速查看，又能提高教室管理者对多媒体教室的管理效率，我设计此套教室管理系统。

## 1.2 web技术概述

### 1.2.1 web技术的概念

Web（World Wide Web）技术，即全球广域网，也称万维网，本意是蜘蛛网，在网页设计中称为网页，表现为三种方式，即超文本（hypertext）、超媒体（hypermedia），超文本传输协议（HTTP），是建立在Internet上的一种网络服务，为浏览者在Internet上查找和浏览信息提供了图形化的、易于访问的直观界面。Web开发技术大体上可分为客户端技术和服务端技术。

1. 客户端技术：

客户端技术也称前端技术，主要任务是展示信息内容，主要包括HTML语言，CSS语言，JS语言。

1. HTML语言：即超文本标记语言，它包括一系列标签，开发者可通过这些标签实现对网页结构的控制，并通过专用的浏览器，将所需要表达的信息按照规则表现出来，是网页的内容构成部分。
2. CSS语言：即层叠样式表，是用来控制网页样式的语言，它通过属性和属性值的方式为网页的样式提供丰富的定义，使网页更加美观，其中包含但不仅限于可以修改网页的位置、背景、字体、边框，甚至还提供了一系列丰富多彩的动画。随着技术的发展，现在已经开发出多套UI可供开发者使用，其中比较常用的有：element-UI、LayUI、MUI等等。
3. JS语言：即JavaScript，它是一种解释性脚本语言（代码不进行预编译），通过API为网页页面添加交互行为，是网页“动起来”的技术支持。JS不同于服务端的的脚本语言，它是在浏览器上运行的，因此不需要服务器的支持；同时它还具有另一个特点：跨平台，这使得它不依赖于操作系统，因此它也受到众多程序员的青睐。而随着技术的发展，JS程序员不满足强大的JS语言只能在客户端使用，因此服务端语言node.js诞生了，这标志着JS同时能在服务端和客户端编写程序；而且，现在发展出许许多多成熟的JS框架，使得代码更加简洁，也使得开发效率大大提高，其中，现在比较流行的有：Vue、Angular、React。这三大框架各有各的特点，所以用于不同的场合。
4. 服务端技术：

服务端也称后端技术，主要为任务是提供信息。主要包括CGI、ASP、PHP、ASP.NET、JSP和Servlet技术。

1. GCI：即公共网关接口技术，是最早用来创建动态网页的技术，它可以使浏览器与服务器之间产生互动。
2. ASP：即活动服务器页面技术，它通过在页面代码中嵌入VBScript或JavaScript脚本语言来生成动态的内容。但是必须在服务端安装适当的解释器，才可以通过调用此解释器来执行脚本程序，然后将执行结果与静态内容部分结合并传送到客户端浏览器上。
3. PHP：即超文本预处理器，它是一种开源的Web服务器脚本语言，与ASP一样可以在页面中加入脚本代码来生成动态内容。对于一些复杂的操作可以封装到类或函数中，在PHP中提供了许多已经定义好的函数。
4. ASP.NET：这种建立动态Web应用程序的技术，是.NET框架的一部分，可以使用任何.NET兼容的语言来编写ASP.NET应用程序。
5. JSP和Servlet：JSP的组合让Java开发者同时拥有了类似CGI程序的集中处理功能和类似PHP的HTML嵌入功能，此外，Java的运行时编译技术也大大提高了Servlet和JSP的执行效率。Servlet和JSP被后来的javaEE平台吸纳为核心技术。

### 1.2.2 web技术的发展现状

随着信息互联网时代的到来，web技术的发展迎来了空前的机遇。现阶段，客户端技术与服务端技术的发展正推动着web技术的快速发展，目前对于web技术的研究和应用的情况如下：

1. 随着web2.0的到来和发展成熟，不仅让web技术更加完善，而且把web技术推向一个更高的高峰，使web技术应用越来越广泛。
2. 框架化是web技术现阶段的重大特点，框架带来的好处数不胜数，它不仅使程序员可以开发动态网页，也使用户有更好的视觉体验。
3. UI框架的发展则是web技术的另一个重要特点，它使得程序员有更多精力去注重逻辑实现，而不需要像以前一样花费很多精力在样式上。
4. 相关语言的发展也为web技术的发展起到了重要作用，HTML5新增了一系列功能强大的语义化标签、CSS3新增的样式属性则丰富了网页样式、ES6（ES为JS的语法标准）中新增的方法以及更加简洁强大的代码风格则受到无数web技术人员的青睐。
5. 前后端的分离使程序员分工明确，更好地各司其职，降低维护成本。
6. 随着互联网技术的参数和web技术的发展，程序web化、应用web化成为必然的趋势。

## 1.3教室管理系统概述

### 1.3.1教室管理系统简介

教室管理系统是专门用来管理多媒体教室的系统，主要是用来管理教室资源，在高校中较常见，包括预约上课、查看教室安排、预约活动等功能。随着现代网络技术的发达，教室管理系统已经由过去的线下人工方式转变为线上智能管理方式，教师学生可根据自己的安排，线上查看或预约教室，这样不仅极大地方便了教师和学生等等用户快速查找空闲教室，也极大地方便了教务处管理人员对教室的管理。一般的教室管理系统应该由页面展示、逻辑实现和数据修改三大部分组成，其中逻辑实现部分除了要实现整个系统的功能之外，还应该注意赋予用户和管理员不同的权限，这也是所有系统在设计时需要注意的一个问题。

根据教室管理系统的发展历史，按照不同的实现方式，可分为以下几类：

1. 传统的线下人工教室管理系统

由于之前网络技术还未发展起来，教室管理只能采用线下人工管理的方式，这种方式不仅浪费人力，而且还不能实时查看和更新教室状态，极大地影响教室的使用，大大地降低了教师和学生的使用体验，同时也使得管理人员的办公效率无法提高。经常出现教师或学生想申请教室，但管理人员不在的情况；或者，教室的状态未能实时更新，而造成一间教室多人使用或者多种活动使用的情况，这无疑给教师和学生以及管理人员带来极大的困扰。故随着网络技术的发达，这种传统的线下的教室管理系统逐渐被舍弃，取而代之的是线上教室管理系统。

1. 基于RBAC的物联网教室管理系统

RBAC，即Role Based Access Control（角色访问控制），采用角色访问控制技术进行权限管理，赋予教室管理人员和普通用户不同级别的权限。该系统根据用户在登陆系统时的实际身份赋予相应的权限，若用户想要更高的权限，可以向拥有更高权限的管理员申请，但必须经过管理员同意之后，才能获得相应的权限，保证了不越权访问。在用户登陆成功之后，会根据不同级别的权限，为用户展示不同的页面，如管理人员登陆之后，可对信息进行审批等操作；而普通用户登陆之后，只能进行查看和申请等操作。

1. 基于ARM+Linux的嵌入式教室管理系统

ARM是对一类微处理器的统称，因为其优越的表现和优良的性能，而被广泛地应用于嵌入式开发。该教室管理系统包括硬件系统和软件系统，其中硬件系统主要包括主控装置、ZigBee协调器和节点硬件模块、RFID阅读器与电子标签、红外遥控器功能学习模块，软件系统主要包括嵌入式Linux操作系统裁剪、交叉开发环境、Qt/Embedded应用程序、嵌入式数据库sqlite和嵌入式服务器。

1. 基于web的教室管理系统

基于web的教室管理系统一般由两部分组成：前端（客户端）和后端（服务端），两部分的分工也很明确，后端负责提供教室和用户的数据，并把数据以接口的方式提供给前端调用。前端通过调用接口发送请求，后端响应请求之后把数据传给前端。前端在拿到后端传来的数据之后可进行一系列操作，登陆时可以对数据进行匹配，用户和密码匹配即登陆成功，进入下一个页面，并把教室的状态以标签的形式渲染到浏览器上，展示出页面。教师和学生可以通过前端页面，发送请求，改变后端数据。这样可以保证实时地更新数据，因为数据是双向绑定的，只要刷新页面就会重新调用接口，而调用接口传来的数据是最新的。

综上所述，以上四种教室管理系统中，传统的线下人工教室管理系统因为耗费巨大的人力物力，并且效率低下，体验感差，因此已基本被淘汰。基于RBAC的物联网教室管理系统虽然能实现基本功能，但是因其技术方面还不够成熟，还有很多方面有待进一步完善，国内外都在努力地研究该方面的知识，相信在不久的将来，在其各方面完善之后，将会广泛应用于国内外大部分学校中。基于ARM+Linux的教室管理系统因为比较复杂，同时涉及硬件和软件方面的知识，虽然在硬件方面可以采用市面上成熟的开发板和开发板模块进行二次开发，但是由于其实用性不太大，同时涉及物联网、嵌入式、以太网及自动控制等方面技术的综合应用，因此要想设计一套成熟的教室管理系统，必将耗费较大的人力物力，所以应用也较少，一般用于教学案例而已。基于web的教室管理系统，因为其轻量级、实用性、高效性等特点，简单易上手，只需一台电脑（服务器）就可以进行开发，因此开发成本也较小，并且现在网络技术的发达也使其安全性有了较好的保障，特别适用于教师和学生等用户的需求，因此被广泛应用。

本设计采用了基于web的教室管理系统设计。可以预见，在未来教室管理系统的发展历程中，更加强大高效的教室管理系统将会进一步得到应用和普及，而效率低、功能少的教室管理系统将逐步被多功能、智能化、自动化的系统所替代。

### 1.3.2国内外教室管理系统发展现状

在国内，现阶段基于互联网开发的教室管理系统已经基本取代了落后的人工教室管理系统，但是由于学校的教室管理系统一般由生产集成商独立研发，其重要目的是实现基本功能即可，一般没有充分了解用户的需求，所以由公司研发出来的教室管理系统，虽说在性能上有一定的优化，但是没办法做到对学校教室有百分之百的了解，因此在细节上也不能百分之百地满足教师和学生的需求。

从其他国家和地区的实际情况来看，由于对传统教室管理系统的漏洞多和教室信息化问题的都有较早认识，以及有强大的网络技术和经济基础做支撑，因此国外比较早教室管理系统的网络化。并且由于起步早，并且不断地推进技术的进步，使得国外一度在教室管理方面遥遥领先国内。

但是近些年来，由于在国内涌现出一大批优秀的开发人员，大大提高了我国在教室管理方面信息化的发展速度。开发出前端框架Vue.js的尤雨溪便是其中之一，Vue.js在近些年可谓是风靡全球，也可以说是web程序员必须掌握的一门技术，它的出现改变了无数网页。正是由于在国内有无数个像尤雨溪这样优秀的开发人员，在细节上不断完善，功能上更加强大，使得多媒体教室管理系统更加多样化。从目前来说，各种网络技术基本都可以实现教室的自动化和实时化管理，其区别只是用不同的手段和方法去实现，应用比较广泛的有基于web的教室管理系统和基于嵌入式的教室管理系统，而作为新技术的物联网，在融合其他技术之后开发出来的教室管理系统也将是一个前景不错的方向。

总而言之，多媒体教室管理的实现是一个不断提升的持续过程。它不仅要随着技术的发展表现出与时俱进的特点，而且要随着用户的需求不断完善其功能。例如，在过去，社团活动等实践活动较少的时候，教室足够举办这些活动，但随着社团活动和其他各种活动的增加，使得原本足够使用的教室呈现出资源不够、影响彼此使用的情况，所以我们需要更智能的、更全面的教室管理系统。因此教室管理系统的提升也是一个借鉴性的提升，我们不仅可以借鉴过去的系统，还可以借鉴别人的系统，并结合自身的需求，开发出最合适的教室管理系统。

## 1.4本论文的研究工作和内容

本文研究工作的目的是为了设计和实现一个实用的嵌入式门禁系统软件，实现系统对门禁管理的稳定高效控制。

嵌入式门禁系统主要由嵌入式门禁控制器和相关门禁设备组成。门禁控制器是门禁系统的核心，也是本次设计的重点，控制器上移植了嵌入式Linux操作系统，采用高级语言C/C++实现对系统软件的编码，系统前端信息识别[8]采用非接触卡技术。通过设计，系统能够有效地管理、控制大门的关闭与开启，识别出入人员的权限，并且能够添加或删除用户，并将用户信息进行记录保存，完成了门禁系统对管制区域的实时控制及安全管理。

本设计主要涉及门禁系统的软件编码，本文也主要介绍该嵌入式门禁系统应用软件的设计及其实现过程，论文的主要工作如下：

1.搭建嵌入式系统软件开发环境；

2.移植tslib、qt库；

3. Linux下的多线程编程技术；

4.实现对RFID通信协议的编码；

5.系统设备驱动的编码。

## 1.5本论文的内容和结构安排

本设计采用基于三星的S5PV210CPU开发板，通过外接RFID模块，LCD显示器等外部设备来组成门禁系统的控制器。通过LCD界面可以操作控制门禁管理系统，实现对门禁区域的管理。

本设计的目的如下：

* 掌握Cotex-A8的相关系统开发；
* 掌握RFID通信协议；
* 掌握GUI系统的开发与设计。
* 掌握部分设备驱动的编写和系统编程。

第二章讲述本设计硬件模块的选取，论证了软件方案的整体架构以及开发过程中使用的关键技术，最终确定整个系统设计的方案。

第三章是整个门禁系统设计的灵魂，介绍了每个功能模块的的程序设计，包括本门禁系统的界面设计及内部应用程序的具体实现。

第四章对设计进行了必要的软件测试，验证设计方案的正确性。

最后则是对本次毕业设计作了总结和期望。

# 2嵌入式门禁系统软硬件方案选取

## 2.1系统硬件方案的选取

### 2.1.1门禁系统控制器

本门禁系统控制器采用了以ARMCortex-A8内核，主频可达1GHZ的三星芯片S5PV210，该芯片具有较强的运算性能。同时，开发板提供了丰富的外设硬件资源，支持3.5寸到12.1寸的LCD显示屏，屏幕分辨率可达到1024\*768像素，网卡采用DM9000AEP，同时拥有1个RS232串口，3个TTL电平串口，1个miniUSB2.0接口和1个USBHost2.0接口，1个VGA接口和1个TV接口等，能够满足开发的大部分需求，并且支持u-boot、Linux内核、根文件系统以及一些开发软件库的移植，便于开发。

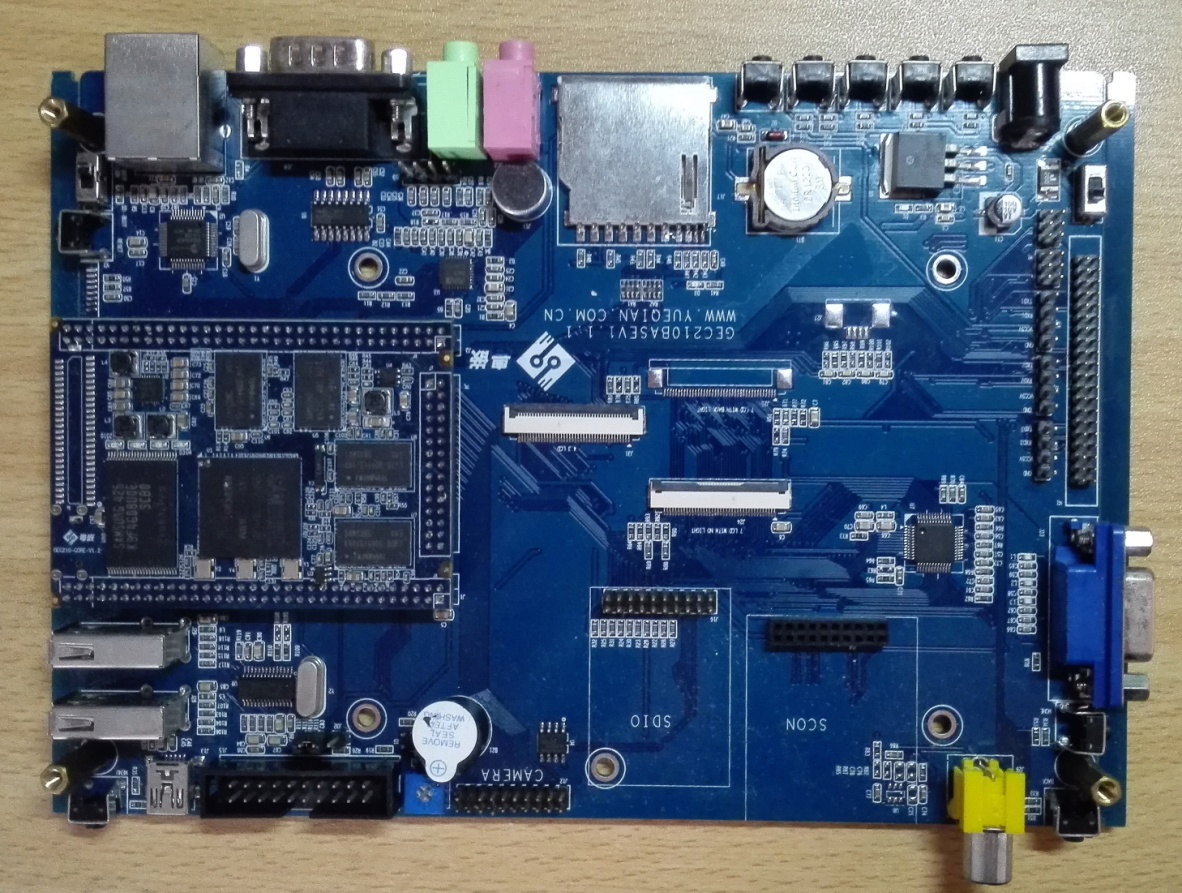


图2.1S5PV210开发板底板

### 2.1.2RFID通信协议读写模块

本设计采用了以NXP原装芯片MFRC522开发的13.56MHz智能IC卡读卡模块，其具有易用、稳定、可靠、体积小的特点。该模块不但支持Mifare卡[9]的数据读写操作，还支持CPU卡读写操作。该模块产品广泛应用于考勤系统、电子感应门锁、门禁系统等不同场合。



图2.2RFID通信协议读写模块

### 2.1.3图形用户界面选择

qt是一个适用于多平台的图形用户界面(GUI)应用程序框架,其使用面向对象编程,易于扩展。qt工具包其实上就是一个C++类库,并且能够通过使用不同编译工具链编译代码来构建不同平台的图形用户界面。

qt/Embedded是qt的嵌入式版本,它在原有qt的基础上做了许多调整以适应于嵌入式环境。qt的可移植性好,并且能够支持多个平台的交互开发。目前，qt已被广泛地应用于各种嵌入式系统产品开发。因此本系统选择qt来开发本系统设计的图形用户界面。

## 2.2门禁系统软件实现的关键技术

### 2.2.1系统交叉编译环境的搭建

嵌入式系统硬件资源无法比拟PC机，无法建立其开发工作所需的开发环境，而且对于面向产品的嵌入式设备来说，完全没有必要将其发展成为一个集运行环境和开发环境于一身的系统，因此，交叉开发环境是目前嵌入式系统开发中必不可少的开发编程环境。交叉开发环境[10]，实际上就是在主机上开发能够在目标机上运行的可执行程序。

本设计采用嵌入式Linux操作系统，PC开发环境则选用了LinuxUbuntu操作系统，其内核是Linux3.5.0版本。系统交叉开发环境的创建过程有以下几个步骤：

1. 从三星官网下载交叉编译工具链软件包arm-2009q3.tar.bz2。

2. 将arm-2009q3.tar.bz2工具包解压，并移动到操作系统目录/usr/local。

3 将工具链路径添加到系统环境变量PATH中，PATH环境变量的指定一般放在~/.bashrc中或是/etc/profile文件中。

### 2.2.2交叉编译工具链简介

嵌入式开发软件的编译和链接使用建立好的交叉开发环境来完成软件的编译和链接工作。一般情况下，我们称对嵌入式软件进行交叉编译的主机称为宿主机，即普通的PC计算机，而把嵌入式软件实际运行的环境称为目标机，即嵌入式系统环境。通常的编译过程包括汇编、编译、链接等步骤，所以嵌入式系统的交叉环境也包括交叉汇编、交叉编译、交叉链接等步骤。本设计使用的是ARM提供的arm-linux-gcc交叉编译器，交叉链接器则为arm-linux-ld。由于嵌入式系统本身硬件资源有限的原因，嵌入式软件开发一般都希望代码简练，因此，应用程序和相关库文件交叉编译完成之后，一般都会使用arm-linux-strip去除相关的调试信息，以减小应用程序和库文件的大小。

### 2.2.3应用程序的移植

在目标板上搭建完应用程序运行所需的软硬件平台之后，便可以进行嵌入式系统应用程序的移植。首先，我们需要将应用程序运行时依赖的库文件拷贝到根文件系统中，接着，我们需要指定库文件的路径到系统的环境变量中，最后，通过串口或网口将应用程序下载到目标板上的文件系统，便可运行该应用文件，完成嵌入式应用文件的移植。

本设计通过LCD显示屏提供按键来实现用户交互，用户可通过界面输入用户密码来获得权限开启大门。由于使用到了LCD屏幕以及触摸屏，触摸屏容易存在抖动问题，故需在软件设计上移植tslib库，实现对触摸屏驱动获得的采样特性提供滤波、去抖、校准等功能。

1.tslib移植步骤

（1）解压tslib源码包，并运行autogen.sh生成配置文件configure，执行configure生成Makefile文件。

（2）执行命令如下：

./configure --host=arm-linux --prefix=/home/linux/tslib--cache-file=$ARCH-linux.cache

其中，--host指的是交叉编译器的前缀，--prefix变量指定了执行makeinstall命令后编译生成的可执行文件、库文件以及配置文件所安装的目录。

（3）执行make、makeinstall操作完成对tslib源码的编译以及安装。

（4）到指定安装目录下，将tslib相关文件拷贝到目标板的根文件系统下。

（5）修改tslib的配置文件配置，设置/etc/profile的相关环境变量。

（6）运行/tslib/bin下的测试文件，实现触摸屏的校准。

2.qt库移植步骤

qt是一套能够跨平台运行的C++图形用户界面库，其具有时间过滤器、信号和槽[11]的通信机制等特性。而这些特性的存在使得用户开发软件界面的工作变得更加方便。因此本设计通过使用qt库来实现人机交互界面的设计。

（1）解压qt源码包。

（2）执行configure生成Makefile文件。命令执行如下：

./configure-prefix/qt-everywhere-embeddedarm-release-opensource-fast-no-accessibility-no-scripttools-no-mmx-no-multimedia-no-svg-no-webkit-no-3dnow-no-sse-no-sse2-silent-qt-libpng-qt-libjpeg-no-libtiff-no-multimedia-makelibs-nomaketools-nomakeexamples-nomakedocs-nomakedemo-no-nis-no-cups-no-iconv-no-dbus-no-openssl-xplatformqws/linux-arm-gnueabi-g++-little-endian-qt-freetype-depths16,18-qt-gfx-linuxfb-no-gfx-transformed-no-gfx-multiscreen-no-gfx-vnc-no-gfx-qvfb-qt-kbd-linuxinput-no-glib-qt-mouse-tslib-I/home/linux/tslib/include-L/home/linux/tslib/lib

（3）执行make、makeinstall操作完成对qtlib源码的编译以及安装。

（4）到指定安装目录下，将qtlib相关文件拷贝到目标板的根文件系统下。

### 2.2.4Linux下多线程编程

线程相对于进程来说，是一个更加接近独立体[10]的调度概念，它是进程内调度最基本的单位，并且可以与同一进程中的其他线程共享数据，同时具有更强的并发执行能力，从而提高了程序的运行效率和缩短了响应时间。

本设计通过qt库提供的线程类来实现系统的多线程设计。Qt库中的QThread类提供了与系统无关的线程，每一个QThread对象代表在程序中一个单独的线程。在多任务操作系统中，它和同一进程中的其它线程共享数据，同时运行时并发性强，效率高效。它并不是像系统调用的pthread\_create（）接口一样在main()中开始，QThread类是在run()中开始运行线程的。开发人员只需在QThread对象中重写run()函数，并在需用启动线程时调用start（）成员函数来启动线程。

本嵌入式门禁系统的软件架构按照线程功能可分为若干模块,其中包括RFID采集线程、按键采集线程、门禁控制线程，按键采集线程为主线程，整个应用程序程序由此启动，而门禁控制线程及RFID采集线程均其创建产生。整个系统系统运行期间，按键采集线程负责实现系统的初始化设置，然后启动其他线程运行；RFID采集线程负责RFID模块对于卡片信息的采集；门禁控制线程主要负责将RFID采集线程和按键采集线程所采集的数据进行处理，识别权限并控制大门的开关。系统各线程之间通过全局的公共变量来达到实现数据共享的目的，其中，公共变量的访问采用线程间互斥锁来实现读写互斥目的，而线程间运行条件的等待采用信号量来实现线程间的同步。系统软件总体结构如下图所示：

图2.3系统软件架构图

### 2.2.5Linux字符设备驱动编程

本设计中使用到了继电器来模拟门禁系统开门的操作，以及利用蜂鸣器设计了门铃，系统对这些设备的管理均通过字符设备驱动来实现。

字符设备，属于Linux系统设备之一，指只能通过字节流来实现读写的设备，其不能随机读取数据。常见的字符设备有鼠标，键盘、led设备等。每一个字符设备对应/dev下一个设备文件。Linux用户程序通过设备文件来实现对字符设备的操作。Linux字符设备驱动模型的建立需要如下操作：

1.字符设备驱动程序基础

(1) 主设备号和次设备号

一个字符设备需要向系统申请一个主设备号和一个次设备号，设备的主设备号主要用于表示与设备文件相连的驱动代码，用来反映注册设备的类型，而次设备号则是被驱动程序用来分辨系统所操作的具体设备，主要用于区分同类型下的设备。

在linux2.6的内核中，字符设备通过dev\_t结构体来描述，在驱动代码中分配cdev，主要任务是分配一个结构体和申请设备号。

设备号的申请可以通过使用以下宏来获得主次设备号：

MAJOR(dev\_tdev)；

MINOR(dev\_tdev)；

也可以使用下列宏通过主次设备号生成dev\_t:

MKDEV(intmajor,intminor)；

(2) 分配设备号

Linux设备号的申请可通过两种方式来实现，分别为动态申请和静态申请。

对于设备号的操作可有以下接口来实现：

静态申请：

intregister\_chrdev\_region(dev\_tfrom,unsignedcount,constchar\*name)；

动态分配：

intalloc\_chrdev\_region(dev\_t\*dev,unsignedbaseminor,unsignedcount,constchar\*name)；

注销设备号：

voidunregister\_chrdev\_region(dev\_tfrom,unsignedcount)；

2.字符设备驱动程序相关的数据结构：

（1）structfile：

代表一个打开的文件描述符，用户每打开系统中的一个文件，内核中都会生成一个关联的structfile。它由内核在调用系统IO函数open时创建，并传递给该设备文件上的IO操作函数，直到调用close（）关闭。

（2）structinode:

该结构体用来记录文件的物理信息。它与file结构体不同，一个文件只有一个inode结构，但是可以对应多个file结构。Inode结构体一般作为file\_operations结构中函数参数传递给内核。  
　　inode指的是索引节点。每个存储设备或存储设备的分区分为inode和Block，Block是用来存储数据。而inode用来存储这些数据的信息，包括文件大小、属主、归属的用户组、读写权限等。操作系统根据系统调用，能通过inode值快速索引相对应的文件。

（3）structfile\_operations

该结构体的每个成员均对应着一个系统调用接口。用户进程通过系统调用在对设备文件进行文件IO操作时，系统通过设备文件的主设备号找到相应的设备驱动程序，然后读取到相关调用函数接口的指针，最后执行调用函数。

3.字符设备驱动程序设计

一个简单的字符设备驱动程序的框架需要通过以下步骤来实现：

（1）分配定义一个字符设备cdev

cdev是一个描述一个字符设备的结构体，设计一个字符设备，就必须定义一个cdev结构体，然后对cdev结构进行初始化。初始化结束后，将cdev注册到内核，这样，内核中就添加了一个字符设备驱动。

（2）定义一个文件操作集，并对文件操作集进行初始化：

这部分主要完成对设备硬件初始化，硬件初始化主要是硬件资源的申请与配置以及实现对设备的操作，用户空间的应用程序通过访问文件的形式访问相应的字符设备，一般都通过open、read、write、close等系统调用来实现。而这些系统调用的执行过程则对应于file\_operations结构体中成员函数，它们是字符设备驱动与内核的接口。

（3）字符设备初始化

将文件操作集以及对应字符设备结构体相关联。

voidcdev\_init(structcdev\*cdev,conststructfile\_operations\*fops)；

（4）申请设备号

动态申请：

intalloc\_chrdev\_region(dev\_t\*dev,unsignedbaseminor,unsignedcount,constchar\*name)

静态申请：

intregister\_chrdev\_region(dev\_tfrom,unsignedcount,constchar\*name)

（5）注册cdev，将字符设备加入到内核

intcdev\_add(structcdev\*p,dev\_tdev,unsignedcount)

（6）注销设备号

调用unregister\_chrdev\_region()以释放原先申请的设备号。

（7）删除cdev

在字符设备驱动模块代码中，卸载函数调用cdev\_del()函数删除系统中的一个cdev，完成字符设备的注销。

voidcdev\_del(structcdev\*p)

### 2.2.6系统GUI制作

1.qt简介

qt是一个跨平台的C++应用程序开发框架。信号和槽是qt通信机制中的一种高级接口，主要应用于应用于对象之间的通信。信号和槽函数通信机制是qt自行定义的一种通信机制，它独立于标准的C/C++语言。本设计在LCD屏幕上设计了一个键盘界面，在qt界面制作中，界面按钮与目标执行代码的关联主要通过信号与槽函数机制来实现。

大多GUI软件工具包中，窗口小部件(widget)触发的动作均是由一个回调函数来响应，而回调函数通常是一个指向某个执行函数的指针。但是，qt的信号和槽取代了这些函数指针，使得我们编写程序时变得更为简洁。信号和槽通信机制能携带不同数量和不同类型的函数参数，而且类型完全安全，不会像回调函数产生coredumps错误。

qt中所有从QObject或其子类中派生出来的类均能使用信号和槽通信机制。当对象发生特定的触发动作时，信号就由该对象发送，而发送信号的对象并不关心接收信号对象，由此实现了信息的封装。槽函数作为普通的对象成员函数，用于接收信号，但一个槽并不知道是否有任何信号与本身相连接。多个信号可以与单个槽进行连接，单个信号也可以与多个槽进行连接。总之，信号与槽函数通信机制构造了qt中的一个部件编程机制。

2.基于qt的系统界面的实现

本设计采用QTCreator集成开发环境上进行系统界面的设计。首先，在虚拟机上的Ubuntu系统打开集成开发环境QTCreator，创建一个工程，点击File->NewFileOrProject，会弹出下图所示窗口：

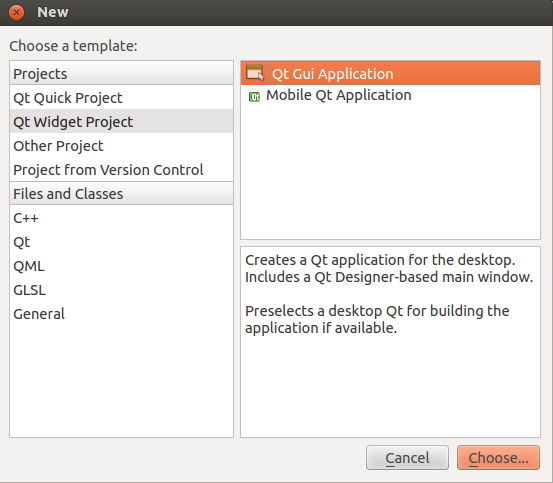


图2.4创建qt工程

按照上图所示，正确选择创建工程模板的类型之后，便可进入下一步操作，接着，设置工程的名称和保存路径，之后会弹出以下窗口：

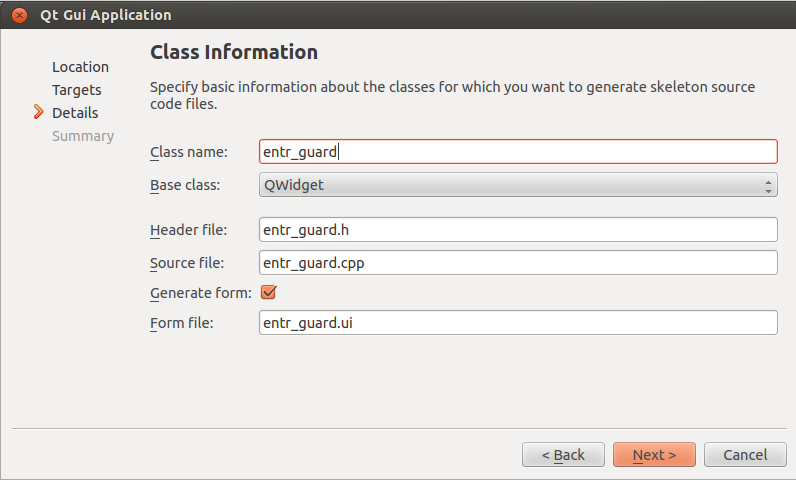


图2.5设置基类相关选项

在设置完系统基类的名称和继承父类的类型之后，qt工程创建完成，工程自动生成5个文件，分别是main.cpp、entr\_guard.cpp、entr\_guard.h、entr\_guard.ui、entr\_guard.pro。

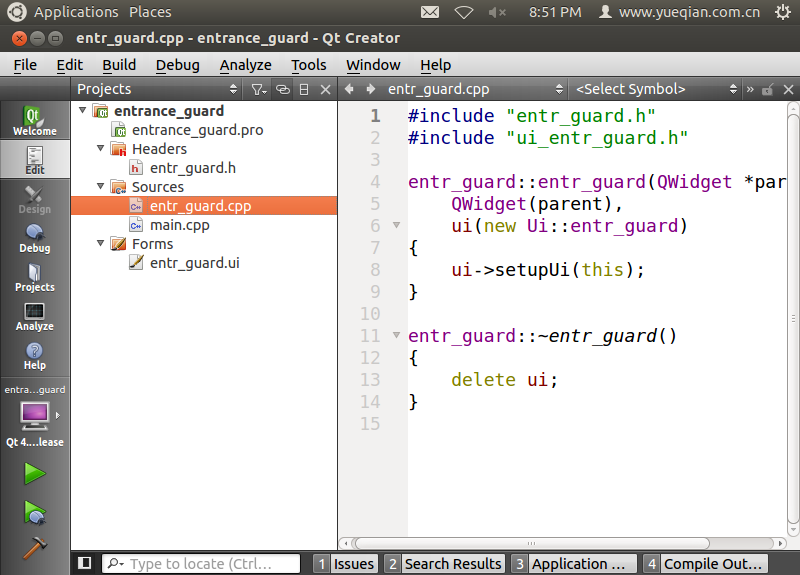


图2.6工程创建完成

根据系统功能的需要，本系统界面主要由PushButton控件组成，其中包含0-9九个数字按钮，一个门铃按钮，一个确认按钮，一个用户添加按钮和一个用户删除按钮组成，一个保存按钮和一个恢复按钮。界面效果图如图2.7所示。

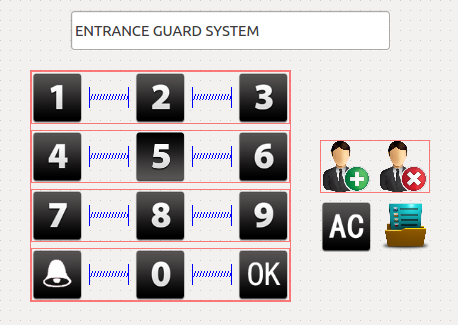


图2.7系统UI界面

### 2.2.7RFID通信协议

本设计采用了以NXP原装芯片MFRC522[12]芯片开发的13.56MHz智能IC卡读卡模块，该模块以命令——响应的方式进行工作，通常是由主机首先发出命令，然后等待模块响应。本设计对于该模块的操作主要通过串口来通信，模块通信命令如表2.1所示。

表2.1RFID模块命令

|  |  |
| --- | --- |
| 命令符 | 功能 |
| A | 请求-——Request |
| B | 防碰撞——CassAnticoll |
| C | 选择——CascSelect |
| D | 卡挂起——Halt |
| E | 证实E2——Authentication |
| F | 证实——AuthKey |
| G | 读——Read |
| H | 写——Write |

前4条命令（命令A—D）是ISO14443A标准定义的命令，只要符合该标准的卡都应能发出响应。后4条命令（命令E—H）为Mifare1卡的专用命令，只有先进行验证（命令E、F）成功之后才能进行。命令的发送以数据帧的形式发送，数据帧的结构如表2.2所示。

表2.2命令数据帧格式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 帧长 | 包号/命令类型 | 命令/状态 | 信息长度 | 信息 | 校验和 | 帧结束符 |
| 1byte | 1byte | 1byte | 1byte | 1byte | 1byte | 1byte |

基于本设计的需求，由于本模块在本设计中主要用于识别非接触卡的卡号，因而未用到读卡器对卡片的数据写操作。系统控制器只是通过对RFID模块的轮询发送读卡命令，防碰撞接收卡片卡号的功能，因此只使用了A、B命令，其命令结构如下：

1．请求（Cmd=A）

命令类型（CmdType）：0x02

命令（Command）：‘A’

数据长度（Length）：1

数据信息（Info）：请求模式（1字节）：0x26——IDLE

0x52——ALL

2．防碰撞（Cmd=B）

命令类型（CmdType）：0x02

命令（Command）：‘B’

数据长度（Length）：若位计数=0，则长度=2

若位计数≠0，则长度=6

数据信息（Info）：选择代码（1字节）：0x93——第一级防碰撞

0x95——第二级防碰撞

0x97——第三级防碰撞

位计数（1字节）：已知的序列号的长度

序列号（4字节）（若位计数≠0）

### 2.2.8Linux系统编程

本设计采用了多线程编程来完成对系统多个设计模块的控制，但由于多线程中共享着一些共同的资源是全局变量，因此需要使用线程通信的同步互斥机制。

线程同步是指不同线程之间所存在的一种制约关系，一个线程的执行依赖另一个线程的消息或者状态，当其没有得到另一个线程的消息时应等待挂起，直到消息发送回来到达时才被唤醒。  
线程互斥是指对于共享的进程系统资源，在各个线程访问该资源时具有排它性。而当有多个线程需要同时使用某一共享资源时，该资源又只允许在任何时刻最多只有一个线程能够访问，其它需要访问该资源的线程必须等待，直到占用资源的线程访问结束并释放该资源。

对于以上两种不同的情况，本设计分别使用了互斥锁和信号量机制来实现对线程资源的管理。

线程引入互斥锁的目的在于保证在对进程共享资源数据操作的完整性。互斥锁主要用来保护临界资源，线程必须先获得互斥锁才能访问临界资源，访问完临界资源后释放该锁。如果无法获得锁，线程会阻塞直到获得锁为止。线程互斥锁的操作如下：

初始化互斥锁  
intpthread\_mutex\_init(pthread\_mutex\_t\*mutex,pthread\_mutexattr\_t\*attr)

申请互斥锁：  
intpthread\_mutex\_lock(pthread\_mutex\_t\*mutex)  
释放互斥锁：  
intpthread\_mutex\_unlock(pthread\_mutex\_t\*mutex)

信号量代表系统中的某一类资源，其数值代表的是系统中该资源的可用数量。信号量是一个受保护的变量，一般只能通过以下操作来访问信号量：初始化P操作（申请资源），V操作（释放资源）。

pthread库常用的信号量操作函数如下：

intsem\_init(sem\_t\*sem,intpshared,unsignedintvalue)  
 intsem\_wait(sem\_t\*sem)  
 intsem\_post(sem\_t\*sem)  
 intsem\_trywait(sem\_t\*sem)  
 intsem\_getvalue(sem\_t\*sem,int\*svalue)

# 3嵌入式门禁系统主控系统软件设计

## 3.1系统设备驱动程序

本设计采用继电器模块来模拟门禁系统中对于门锁的控制，利用蜂鸣器模块来模拟门禁系统中的门铃，系统通过字符设备驱动来管理以上两个设备，并通过在用户代码中调用ioctl()接口来实现对设备的控制，其具体驱动代码如下：

程序清单3.1继电器驱动代码程序清单

|  |
| --- |
| #defineDEVICE\_NAME"relay"  staticintrelay\_gpios[]={  S5PV210\_GPJ2(0),//继电器模块控制IO口  };  staticlonggec210\_relay\_ioctl(structfile\*filp,unsignedintcmd,  unsignedlongarg)//ioctl接口实现  {  switch(cmd){  case0:  gpio\_set\_value(relay\_gpios[0],cmd);//关闭继电器，模拟关门操作  break;  case1:  gpio\_set\_value(relay\_gpios[0],cmd);//启动继电器，模拟开门操作  break;  default:  return-EINVAL;  }  return0;  }  staticstructfile\_operationsgec210\_relay\_dev\_fops={//配置文件操作集成员  .owner =THIS\_MODULE,  .unlocked\_ioctl =gec210\_relay\_ioctl,  };  staticstructmiscdevicegec210\_relay\_dev={//配置杂项设备结构体  .minor =MISC\_DYNAMIC\_MINOR,  .name =DEVICE\_NAME,  .fops =&gec210\_relay\_dev\_fops,  };  staticint\_\_initgec210\_relay\_dev\_init(void){//驱动初始化函数  intret=gpio\_request(relay\_gpios[0],"RELAY");//申请IO资源  if(ret){  printk("%s:requestGPIO%dforRELAYfailed,ret=%d\n",DEVICE\_NAME,  relay\_gpios[0],ret);  returnret;  s3c\_gpio\_cfgpin(relay\_gpios[0],S3C\_GPIO\_OUTPUT);  gpio\_set\_value(relay\_gpios[0],1);  }  ret=misc\_register(&gec210\_relay\_dev);//注册杂项设备  returnret;  }  staticvoid\_\_exitgec210\_relay\_dev\_exit(void){//驱动卸载函数  gpio\_free(relay\_gpios[0]);  misc\_deregister(&gec210\_relay\_dev);  }  module\_init(gec210\_relay\_dev\_init);//驱动的入口函数  module\_exit(gec210\_relay\_dev\_exit);//驱动的出口函数  /\*驱动的描述信息\*/  MODULE\_AUTHOR("yangchaofeng");//驱动的作者  MODULE\_DESCRIPTION("thedriverofrelay");//驱动的描述  MODULE\_LICENSE("GPL");//遵循的协议 |
|  |

程序清单3.2蜂鸣器驱动代码程序清单

|  |
| --- |
| #defineDEVICE\_NAME"buzzer"  staticintbuzzer\_gpios[]={  S5PV210\_GPD0(0),//蜂鸣器控制IO  };  staticlonggec210\_buzzer\_ioctl(structfile\*filp,unsignedintcmd,  unsignedlongarg)//ioctl接口实现  {  switch(cmd){  case0:  gpio\_set\_value(buzzer\_gpios[0],cmd);//关闭蜂鸣器  break;  case1:  gpio\_set\_value(buzzer\_gpios[0],cmd);//打开蜂鸣器  break;  default:  return-EINVAL;  }  return0;  }  staticstructfile\_operationsgec210\_buzzer\_dev\_fops={//配置文件操作集成员  .owner =THIS\_MODULE,  .unlocked\_ioctl =gec210\_buzzer\_ioctl,  };  staticstructmiscdevicegec210\_buzzer\_dev={//配置杂项设备结构体  .minor =MISC\_DYNAMIC\_MINOR,  .name =DEVICE\_NAME,  .fops =&gec210\_buzzer\_dev\_fops,  };  staticint\_\_initgec210\_buzzer\_dev\_init(void){//驱动初始化函数  ret=gpio\_request(buzzer\_gpios[0],"buzzer");//申请IO资源  if(ret){  printk("%s:requestGPIO%dforbuzzerfailed,ret=%d\n",DEVICE\_NAME,  buzzer\_gpios[0],ret);  returnret;  }  s3c\_gpio\_cfgpin(buzzer\_gpios[0],S3C\_GPIO\_OUTPUT);  gpio\_set\_value(buzzer\_gpios[0],0);  }  ret=misc\_register(&gec210\_buzzer\_dev);//注册杂项设备  printk(DEVICE\_NAME"\tinitialized\n");  returnret;  }  staticvoid\_\_exitgec210\_buzzer\_dev\_exit(void){//驱动卸载函数  gpio\_free(buzzer\_gpios[0]);  misc\_deregister(&gec210\_buzzer\_dev);  }  module\_init(gec210\_buzzer\_dev\_init);//驱动的入口函数  module\_exit(gec210\_buzzer\_dev\_exit);//驱动的出口函数  /\*驱动的描述信息\*/  MODULE\_AUTHOR("yangchaofeng");//驱动的作者  MODULE\_DESCRIPTION(thedriverofbuzzer");//驱动的描述  MODULE\_LICENSE("GPL");//遵循的协议 |
|  |

本设计设置了一个按键来模拟门禁系统的关门操作，对于按键设备的管理采用在内核注册外部中断形式。当系统接收到按键的中断信号时，驱动将在中断执行函数中启动内核定时器延时，当定时时间到达再回调定时器执行函数实现按键功能，由此来实现按键的防抖功能。其具体驱动代码如下：

程序清单3.3按键驱动代码程序清单

|  |
| --- |
| structtimer\_listmy\_timer;  voidtimer\_func(unsignedlongdata){//定时器回调函数  gpio\_set\_value(S5PV210\_GPJ2(0),1);  }  irqreturn\_tkeys\_isr(intirq,void\*dev\_id){//中断处理函数  mod\_timer(&my\_timer,jiffies+25);//启动定时器定时  returnIRQ\_HANDLED;  }  staticint\_\_initeint\_init(void){//驱动的初始化函数  intret;  init\_timer(&my\_timer);//初始化定时器  my\_timer.function=timer\_func;  ret=request\_irq(IRQ\_EINT(18),keys\_isr,IRQF\_TRIGGER\_FALLING|IRQF\_DISABLED,"eint16\_k2",(void\*)a);//从内核中申请中断资源  if(ret<0) {  printk("cannotrequestIRQ\_EINT18\n");  returnret;  }  returnret;//驱动注册成功，返回0；驱动注册失败，返回一个负数错误码  }  staticvoid\_\_exiteint\_exit(void){//驱动退出函数  free\_irq(IRQ\_EINT(18),NULL);//释放资源  }  module\_init(eint\_init);//驱动的入口函数  module\_exit(eint\_exit);//驱动的出口函数  /\*驱动的描述信息\*/  MODULE\_AUTHOR("yangchaofeng");//驱动的作者  MODULE\_DESCRIPTION("thedriverofkey");//驱动的描述  MODULE\_LICENSE("GPL");//遵循的协议 |

## 3.2门禁系统程序编写

本设计使用qt库来规划门禁系统的人机交互界面，并遵循了qt编程的一般规则，将程序分为若干个任务，并将其通过不同的类来实现不同的模块功能。

### 3.2.1门禁系统程序界面构建

本设计选择qt中的Widget类来构建界面，通过鼠标拖动对应的窗口控件到界面布局中的适当位置，添加所需要的按钮来构建初步界面，整体界面如图3.1所示。

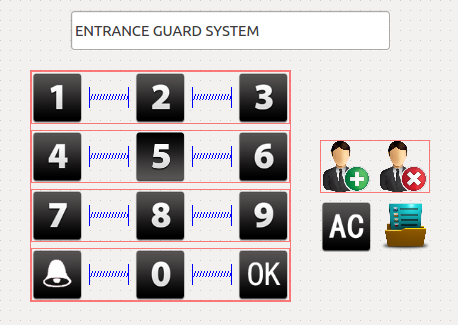


图3.1门禁系统界面

### 3.2.2界面按键触发操作类

在qt工程中添加界面按键触发操作类的头文件，该头文件给出该类的声明。该类主要完成本系统基础界面的构建，利用信号与槽函数机制实现界面按钮与触发动作的关联，并将用户通过界面密码输入方式开门时的密码信息传递给控制线程，其主要内容如下所示：

程序清单3.4界面按键触发操作类代码清单

|  |
| --- |
| #include<QWidget>  namespaceUi{  classWidget;  }  classWidget:publicQWidget{  Q\_OBJECT  public:  explicitWidget(QWidget\*parent=0);  ~Widget();  privateslots://界面按键槽函数  voidon\_pushButton\_clicked();  voidon\_pushButton\_2\_clicked();  voidon\_pushButton\_3\_clicked();  voidon\_pushButton\_4\_clicked();  voidon\_pushButton\_5\_clicked();  voidon\_pushButton\_6\_clicked();  voidon\_pushButton\_7\_clicked();  voidon\_pushButton\_8\_clicked();  voidon\_pushButton\_9\_clicked();  voidon\_pushButton\_10\_clicked();  voidon\_pushButton\_11\_clicked();  voidon\_pushButton\_13\_clicked();  voidon\_pushButton\_14\_clicked();  voidon\_pushButton\_12\_pressed();  voidon\_pushButton\_12\_released();  voidon\_pushButton\_15\_clicked();  voidon\_pushButton\_16\_clicked();  private:  Ui::Widget\*ui;  }; |

### 3.2.3RFID数据采集类

该类主要负责系统与RFID模块的通信操作，当用户使用该模块来获取系统的开门权限时，本模块将把采集到的卡片卡号信息发送给门禁控制线程，由门禁控制线程来完成对用户权限的识别，其主要内容如下：

程序清单3.5RFID数据采集类程序清单

|  |
| --- |
| #include<QThread>  classrfid\_proc:publicQThread{  Q\_OBJECT  public:  explicitrfid\_proc(QObject\*parent=0);  QStringreadCardID();  intPiccAnticollS(int);/\*防碰撞，获取范围内最大ID\*/  intPiccRequestS(int);/\*请求天线范围内的卡\*/  voidinit\_ttyS(int);/\*设置窗口参数:9600速率\*/  unsignedcharCalBCCS(unsignedchar\*,int);/\*计算校验和\*/  protected:  voidrun();  signals:  publicslots:  private:  QStringstr\_id;  }; |

PiccRequestS：请求天线范围的卡，调用该函数，将执行模块的寻卡指令操作，检查感应区是否有符合条件的卡，若有符合条件的卡进入并且被选中，建立与卡片的第一步的通信联系。然后程序退出循环请求过程，进入防冲突过程。

PiccAnticollS：进行防碰撞，获取天线范围内最大的ID，通过模块内部处理的防碰撞算法实现对卡片的选择，避免了多个卡片同时接近读卡器而造成信息冲突的情况。

readCardID：读卡器读取非接触卡信息，通过主机对模块的轮训发送读卡命令，实现该模块对于卡片的实时读取操作。对于RFID模块的操作程序清单如下所示：

程序清单3.6RFID通信程序清单

|  |
| --- |
| /\*请求天线范围内的卡\*/  intrfid\_proc::PiccRequestS(intfd){  unsignedcharWBuf[128],RBuf[128];  intret;  fd\_setrdfd;  memset(WBuf,0,128);  memset(RBuf,1,128);  WBuf[0]=0x07; //帧长=7Byte  WBuf[1]=0x02; //包号=0,命令类型=0x01  WBuf[2]=0x41; //命令='C'  WBuf[3]=0x01; //信息长度=0  WBuf[4]=0x52; //请求模式:ALL=0x52  WBuf[5]=CalBCCS(WBuf,WBuf[0]-2); //校验和  WBuf[6]=0x03; //结束标志  FD\_ZERO(&rdfd);  FD\_SET(fd,&rdfd);  write(fd,WBuf,7);;  ret=select(fd+1,&rdfd,NULL,NULL,&timeout);  switch(ret){  case-1:  perror("selecterror\n");  break;  case0:  printf("Requesttimedout.\n");  break;  default:  ret=read(fd,RBuf,8);  if(ret<0){  printf("ret=%d,%m\n",ret,errno)  break;  }  if(RBuf[2]==0x00){ //应答帧状态部分为0则请求成功  return0;  }  break;  }  return-1;  }  /\*防碰撞，获取范围内最大ID\*/  intrfid\_proc::PiccAnticollS(intfd){  unsignedcharWBuf[128],RBuf[128];  intret;  unsignedintcardID;  fd\_setrdfd;;  memset(WBuf,0,128);  memset(RBuf,0,128);  WBuf[0]=0x08; //帧长=8Byte  WBuf[1]=0x02; //包号=0,命令类型=0x01  WBuf[2]=0x42;//命令='B'  WBuf[3]=0x02;//信息长度=2  WBuf[4]=0x93; //防碰撞0x93--一级防碰撞  WBuf[5]=0x00; //位计数0  WBuf[6]=CalBCCS(WBuf,WBuf[0]-2); //校验和  WBuf[7]=0x03; //结束标志  FD\_ZERO(&rdfd);  FD\_SET(fd,&rdfd);  write(fd,WBuf,8);  ret=select(fd+1,&rdfd,NULL,NULL,&timeout);  switch(ret){  case-1:  perror("selecterror\n");  break;  case0:  perror("Timeout:");  break;  default:  ret=read(fd,RBuf,10);  if(ret<0){  printf("ret=%d,%m\n",ret,errno);  break;  }  if(RBuf[2]==0x00){//应答帧状态部分为0则获取ID成功  cardID=(RBuf[4]<<24)|(RBuf[5]<<16)|(RBuf[6]<<8)|RBuf[7];  if(cardID>0){  sem\_wait(&sem\_deal);  pthread\_mutex\_lock(&key\_mutex);  key=cardID;  pthread\_mutex\_unlock(&key\_mutex);  sem\_post(&sem\_reqt);  cardID=0;  }  return0;  }  }  return-1;  }  /\*轮询向RFID模块发送命令\*/  QStringrfid\_proc::readCardID(){  intfd;  fd=open(DEV\_RFID,O\_RDWR|O\_NOCTTY|O\_NONBLOCK);  if(fd<0){  fprintf(stderr,"open/dev/ttySAC1fail!\n");  return"error";  }  /\*初始化串口\*/  init\_ttyS(fd);  while(1){  timeout.tv\_sec=1;  timeout.tv\_usec=0;  /\*请求天线范围的卡\*/  if(PiccRequestS(fd)){  sleep(1);  printf("Therequestfailed!\n");  }  /\*进行防碰撞，获取天线范围内最大的ID\*/  if(PiccAnticollS(fd){  sleep(1);  printf("Couldn'tgetcard-id!\n");  }  printf("cardID=%x\n",key);  }  } |

### 3.2.4门禁控制线程类

本类主要实现系统采集完用户密码信息或非接触卡卡片信息之后，将采集来的数据信息与系统所保存的用户信息相互检验，识别用户的操作，其主要内容如下：

程序清单3.7门禁控制线程类代码清单

|  |
| --- |
| /\*定义一个枚举类型变量，用于标记按键的状态\*/  enumkey\_state{  user\_add,  user\_del,  init,  passwd  };  handle::handle(Ui::Widget\*p,QObject\*parent):/\*处理线程与界面控制线程为不同线程，因此本线  QThread(parent){调用界面控件需要外部传UI的指针\*/  ui=p;  }  voidhandle::run(){//重新线程run函数  intfd;  pkey\_nodepkeytmp;  pstr\_nodepstrtmp;  fd=open("/dev/relay",O\_RDWR);  if(fd<0){  printf("open/dev/relayfailed.\n");  ::close(fd);  }  while(1){  sem\_wait(&sem\_reqt);//申请信号量，判断是否采集到数据  if(ksta==passwd){  /\*当添加或删除用户时判断管理员密码\*/  if(strcmp(str1.toStdString().c\_str(),"123456")==0){  if(temp==user\_add){  ui->lineEdit->setText("Correct.Pleaseaddinguser.");  }  else{  ui->lineEdit->setText("Correct.Pleasedeletinguser.")  }  ksta=temp;  }  else{  ui->lineEdit->setText("Error.Inputagain.");  }  str1.clear();  }  else{  /\*RFID模块信息处理\*/  pthread\_mutex\_lock(&key\_mutex);  if(key!=0){  /\*RFID模块识别用户权限\*/  if(ksta==init){  pkeytmp=pkeyhead->next;  while(pkeytmp!=NULL){  if(key==pkeytmp->key){  ui->lineEdit->setText("Identifycorrectly.");  ioctl(fd,0,0);//识别正确，执行开门动作  key=0;  tmp=1;  break;  }  pkeytmp=pkeytmp->next;  }  if(tmp==0){  ui->lineEdit->setText("Erroruser.");  }  key=0;  tmp=0;  }  /\*添加RFID用户\*/  elseif(ksta==user\_add){  pkeytmp=pkeyhead->next;  while(pkeytmp!=NULL){//判断用户是否存在  if(key==pkeytmp->key){  flag=1;  ui->lineEdit->setText("Theuserexisted.");  break;  }  pkeytmp=pkeytmp->next;  }  if(flag==0){  key\_list::kadd(pkeyhead,key);//添加用户  ui->lineEdit->setText("Addusersuccessfully.");  }  ksta=init;  key=0;  flag=0;  }  /\*删除用户\*/  elseif(ksta==user\_del){  key\_list::kcut(pkeyhead,key);  ui->lineEdit->setText("Deleteuser.");  ksta=init;  }  }  pthread\_mutex\_unlock(&key\_mutex);  pthread\_mutex\_lock(&str\_mutex);  if(str1.isEmpty()!=true&&str1.length()=={  /\*按键密码识别\*/  if(ksta==init){  pstrtmp=pstrhead->next;  while(pstrtmp!=NULL){  if(strcmp(str1.toStdString().c\_str(),pstrtmp->passwd)==0){  ui->lineEdit->setText("Identifycorrectly.");  ioctl(fd,0,0);  tmp=1;  str1.clear();  }  pstrtmp=pstrtmp->next;  }  if(tmp==0){  ui->lineEdit->setText("Erroruser.");  }  tmp=0;  str1.clear();  }  /\*添加RFID卡\*/  elseif(ksta==user\_add){  pstrtmp=pstrhead->next;  while(pstrtmp!=NULL{  if(str1==pstrtmp->passwd){  ui->lineEdit->setText("Userexisted.");  flag=1;  break;  }  pstrtmp=pstrtmp->next;  }  if(flag==0){  char\*p=(char\*)str1.toStdString().c\_str();  str\_list::sadd(pstrhead,p);  ui->lineEdit->setText("Addusersucessfully.");  }  ksta=init;  str1.clear();  flag=0;  }  /\*删除卡片用户\*/  elseif(ksta==user\_del){  char\*p=(char\*)str1.toStdString().c\_str();  str\_list::scut(pstrhead,p);  ui->lineEdit->setText("Deleteuser.");  ksta=init;  }  }  else{  ui->lineEdit->setText("Thelengthofpasswdshouldbe6.");  str1.clear();  }  pthread\_mutex\_unlock(&str\_mutex);  }  sem\_post(&sem\_deal);  }  ::close(fd);  } |

### 3.2.5密码信息链表类

本类主要用于保存在系统注册的用户密码信息，控制线程在接收到数据采集线程发送的信息后，便会将所采集的信息与本类中的所有用户信息比较，以此来判别用户权限，其主要内容如下：

程序清单3.6密码信息链表类代码清单

|  |
| --- |
| typedefstructstr\_Node{  charpasswd[7];//存放按键密码  structstr\_Node\*next;  }str\_node,\*pstr\_node;  classstr\_list{  public:  str\_list();  pstr\_nodeinit\_list();//初始化链表  staticintslength(pstr\_nodephead);//计算链表成员个数  staticintsadd(pstr\_nodephead,char\*p);//添加链表信息  staticintslist\_next(pstr\_nodephead);//遍历链表  staticvoidscut(pstr\_nodephead,char\*passwd);//删除链表成员  intlist\_create(pstr\_nodephead);//创建链表头指针  }; |

### 3.2.6非接触卡卡片信息链表类

本类主要用于已在系统注册的非接触卡卡片信息，控制线程在接收到数据采集线程发送的信息后，便会将所采集的信息与本类中的所有用户信息比较，以此来判别用户权限，其主要内容如下：

程序清单3.7非接触卡卡片信息链表类代码清单

|  |
| --- |
| typedefstructkey\_Node{  unsignedintkey;//存放卡号信息  structkey\_Node\*next;  }key\_node,\*pkey\_node;  classkey\_list:publicQObject{  Q\_OBJECT  public:  explicitkey\_list(QObject\*parent=0);  pkey\_nodeinit\_list();//初始化链表  staticintklength(pkey\_nodephead);//计算链表成员个数  staticintkadd(pkey\_nodephead,unsignedintkey);//添加链表信息  staticintklist\_next(pkey\_nodephead);//遍历链表  staticvoidkcut(pkey\_nodephead,unsignedintkey);//删除链表成员  intlist\_create(pkey\_nodephead);//创建链表头指针  signals:  publicslots:  }; |

## 3.3门禁系统主程序

本系统设计的主程序首先创建了一个QApplication对象，qt中的QApplication类负责管理图形用户界面应用程序的控制流和一些主要设置，可以说其负责着qt的整个后台管理，它是一个主事件循环，在循环中处理和调度来自窗口系统和其它资源的所有事件。它也能够处理应用程序的初始化和结束，并且提供系统与用户的对话管理，同时也处理大多数系统和应用程序的设置。QApplication::exec()语句的执行，便是使线程进入一个主事件循环,直到exit（）、quit（）或者用户关闭了应用程序的主窗口。

接着，程序初始化了两个信号量变量，这两个信号量主要用于将数据采集线程与控制线程的同步功能。QPalette类主要负责对于主界面背景颜色的修改，实现界面显示的美化工作，最后，初始化两个保存用户信息的数据结构链表，用于保存用户信息。而在本线程主事件循环开始后,它将会接收来自于用户界面事件以及其他事件源的事件的信号，并向相应的窗口进行分发和处理，实现了人机界面的交互任务。

程序清单3.8门禁系统主程序代码清单

|  |
| --- |
| pthread\_mutex\_tkey\_mutex;  pthread\_mutex\_tstr\_mutex;  unsignedintkey;  QStringstr;  sem\_tsem\_reqt;  sem\_tsem\_deal;  pkey\_nodepkeyhead=NULL;  pstr\_nodepstrhead=NULL;  intbuzzer\_fd;  intmain(intargc,char\*argv[])  {  QApplicationa(argc,argv);  key\_listkey;  str\_listsstr;  pkeyhead=key.init\_list();//初始化卡片信息链表  pstrhead=str.init\_list();//初始化卡片密码链表  handle::load\_keylist(pkeyhead);//读取配置文件，初始化卡片信息链表  handle::load\_strlist(pstrhead);//读取配置文件，初始化用户密码信息链表  sem\_init(&sem\_reqt,0,0);//初始化信号量  sem\_init(&sem\_deal,0,1);  buzzer\_fd=open("/dev/buzzer",O\_RDWR);//打开蜂鸣器设备  Widgetw;  w.setAutoFillBackground(true);  QPalettepalette;  palette.setColor(QPalette::Background,QColor(0,0,0));//设置界面背景颜色  w.setPalette(palette);  rfid\_procrfid;  handlehdl（w.ui）;  /\*启动系统控制线程\*/  hdl.start();  /\*启动rfid数据采集线程\*/  rfid.start();  w.show();  returna.exec();  } |

# 4软件测试与应用

为了保证本次设计软件的质量，必须对软件进行相应的测试与错误纠正。软件测试是指软件开发工作初步完成后，评价软件系统的质量并发现软件运行中的错误。从软件的运行过程来看，软件测试可分为模块局部运行和整体运行；从运行的实际工作环境来看，又可以分为仿真运行和实际运行。软件的故障一般都分布在软件开发的整个过程中，所以软件测试也是软件开发初步完成后必须进行的一项工作。

## 4.1系统总体测试

将LCD模块,RFID模块与开发板连接好后，运行本设计应用软件，便可进行系统测试。搭建好的门禁系统测试实物如图4.1所示。



图4.1门禁系统实物

在开发板启动系统后，通过串口工具SecureCRT输入./entrance\_guard–qws,执行已交叉编译后的可执行代码,便可进行本设计的软件测试。

## 4.2总结

通过对本系统各软件模块的运行测试，并对测试结果以及测试现象进行观察，同时通过串口工具查看应用程序打印输出信息，证明设计达到了系统要求。

# 总结与展望

针对目前市面上的门禁系统应用产品，本文研究了基于RFID数据采集和界面输入密码的识别方式的智能门禁系统，并重点研究了RFID通信协议的内容，并且掌握了非接触卡片与RFID读卡器之间数据传递的工作过程。

RFID是一种基于无线射频通信实现的非接触式自动识别新型技术，RFID标签，同时也称为电子标签，具有体积小、容量大、可重复使用、寿命长等特点，同时支持快速读写、移动识别、多目标识别等特性。因此，RFID技术目前广泛应用于物流、制造等行业，大幅提高运作与管理的效率，降低运作生产成本。而且随着物联网相关技术的不断完善和发展，RFID产业将成为一个新兴的高科技产业。

由于射频通信技术相对于目前门禁系统的其它自动识别技术具有许多独特的优势，再加上电子标签成本较低，本技术已在各行各业中得到广泛应用。本文首先明确了全文的研究重点，即开发一个基于RFID识别和密码输入识别的嵌入式智能门禁系统，然后全面地分析了当前门禁系统现有的技术背景及原理。接着，对整个系统软件的各个模块进行了规划和分析，逐步完成各模块的代码编写。最后将各模块的代码整合，并对已实现的各个模块以及整个系统进行测试，再对测试结果及测试现象进行分析，进一步分析了本系统设计的性能。全文重点在于实现基于RFID标签识别和密码输入识别的嵌入式智能门禁系统软件设计，主要的研究成果如下：

1.通过目前市面上的智能门禁系统及物联网的发展趋势以及现状进行调研，最终确定本设计的需求，提出了本系统设计的研究方案，并确定嵌入式Linux系统为软件核心，进行嵌入式门禁系统系统软件开发，并选购搭建了本系统的硬件开发平台，同时根据硬件平台搭建了系统软件设计的开发环境，其中涉及到操作系统启动前的引导加载程序的移植以及配置，用于本系统设计所需的内核的配置、编译和移植，单板上根文件系统的制作和移植，以及用于开发用户界面的qt使用。

2.研究了射频识别技术，了解掌握了无线射频技术的系统标准、通信协议，着重研究了ISO14443A类命令的详细通信过程，并且对通信过程中的数据格式进行了分析。同时，还研究分析了RFID读卡芯片发生卡片信息碰撞的原因以及防碰撞算法，开发具有RFID射频识别和用户密码输入功能的智能门禁系统。

3.集成了多卡识别时防碰撞功能，并且能够添加删除用户权限，适应了物联网技术的发展，本门禁系统同时支持门铃功能。

虽然本系统设计已基本完成了课题项目的要求，但对于技术已经逐渐成熟的物联网行业来说，由于设计开发时间以及个人能力有限的关系，暂时还不能对系统中的缺点和不足进行纠正或改进，本文在RFID标签识读地研究过程中也存在一些不足，主要包括：

1.本设计的软件系统功能还有待进一步提高和改进；

2.只对系统基本功能进行了有限地测试，系统具体性能还有待于进一步测试和研究；

3.研究非接触卡识别门禁系统安全方面的问题。随着门禁系统的智能化程度的不断提升，RFID安全问题也越来越受到人们的重视。目前，新型的攻击手段在不停地发展，所以门禁系统的安全问题必然要面临更加严峻的挑战，安全系统的改进和普及将是未来门禁系统的另一个研究重点。

# 参考文献

|  |
| --- |
| [1]瞿小玲.基于RFID的低功耗智能门禁系统的设计与研究[D].成都理工大学,2012.  [2]瞿小玲,王洁.RFID在智能门禁系统中的应用[J].黑龙江科技信息,2011,21:32-33.  [3]周海龙.嵌入式门禁系统的设计与实现[D].西安电子科技大学,2012.  [4]姜涛.ARM技术在门禁系统中的应用研究[D].南京邮电大学,2012.  [5]张永生.非接触式IC卡智能门禁系统[D].燕山大学,2012.  [6]朱晓林.基于射频识别的嵌入式门禁系统设计[D].河北工业大学,2012.  [7]王刚.基于非接触IC卡的智能门禁系统设计与研究[D].上海海事大学,2006.  [8]陆璐.物联网中RFID智能门禁系统研究[J].信息技术,2013,07:87-90.  [9]柯资颖.嵌入式门禁系统软件的设计与实现[D].西安电子科技大学,2008.  [10]佟英杉.嵌入式多功能门禁系统的设计与实现[D].大连理工大学,2008.  [11]SuxiaLi.Thedevelopmentofintelligenthomefurnishingentranceguardsystembased  onB/Sarchitecture[A].2014:6.  [12]FeiLao.DesignAndImplementationsOfKindergartenManagementSystemBased  OnFingerprintIdentificationAndRFIDTechnology[A].Proceedingsof2015  InternationalConferenceonEducationTechnology,ManagementandHumanitiesScience  (ETMHS2015)[C],2015:6. |
|  |

# 致谢

光阴似箭，随着毕业论文的完成，大学生涯也即将画上一个圆满的句号。回顾大学四年。一切历历在目，记忆犹新，通过不断地努力学习，本人学习到了许多知识，开拓了自己的眼界，拓宽了自己的思维，并且进一步提高了自己分享问题、解决问题的能力。这些势必会在我今后的工作和学习中带来帮助，我也因此对未来充满信心。

在即将顺利完成毕业设计之际，我在此想要感谢这四年以来给予我热情指导的老师们以及帮助，支持过我，和我一起奋斗过的同学朋友们。

首先要感谢刘汉瑞老师和徐胜老师，正是因为你们辛勤教学而是我开始着迷于嵌入式软件学习，并以此作为自己就业的技术方向，同时也找到了一份较为体面的工作。在不断地学习软件知识期间，我不断深化自己专业理论基础，同时，解决实际问题的工作能力和开发能力也不断提高。另外，在选择就业方向时也是得到了两位老师的指导。

我还要真诚地感谢给我鼓励和与我一起奋斗的同学朋友们。正是因为遇到了志同道合的你们，并且在软件编程和硬件设计相互鼓励和学习，我才能不断坚持着走到今天。在此，我想预祝你们学业有成，工作美满！

最后，我还要感谢我的父母，正是你们源源不断的爱与支持给予了我无穷的动力！