学位论文检测系统

文本复制检测报告单(全文标明引文)

№:ADBD2021R_20140321160247202111101739195086100622O3　　检测时间:2021-11-10 17:39:19

篇名: 21_ZF1821334_张加杰
作者: 张加杰
指导教师:
学位类型:
检测机构: 北京航空航天大学图书馆
提交论文IP: 106.39.43.187
文件名: 21_ZF1821334_张加杰.docx
检测系统: 学位论文检测系统
检测类型: 博硕士学位论文
检测范围: 中国学术期刊网络出版总库
中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库
中国重要会议论文全文数据库
中国重要报纸全文数据库
中国专利全文数据库
图书资源
优先出版文献库
互联网资源(包含贴吧等论坛资源)
英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)
港澳台学术文献库
互联网文档资源
CNKI大成编客-原创作品库
时间范围: 1900-01-01至2021-11-10

可能已提前检测，检测时间：2021/11/9 8:51:51，检测结果：11.7%

检测结果

去除本人文献复制比: 0.7%　　跨语言检测结果: 0%
去除引用文献复制比: 0.7%　　总文字复制比: 0.7%
单篇最大文字复制比: 0.3%　（基于智能语音交互技术的智慧语音助理系统实现）

重复字数:	[283]	总段落数:	[6]
总字数:	[40447]	疑似段落数:	[4]
单篇最大重复字数:	[103]	前部重合字数:	[134]
疑似段落最大重合字数:	[169]	后部重合字数:	[149]
疑似段落最小重合字数:	[29]		

指标: ☐ 疑似剽窃观点　☑ 疑似剽窃文字表述　☐ 疑似整体剽窃　☐ 过度引用

相似表格: 0　　相似公式: 没有公式　　疑似文字的图片: 1(已OCR处理)

2%(169)	2%(169)	21_ZF1821334_张加杰_第1部分（总8550字）
0%(0)	0%(0)	21_ZF1821334_张加杰_第2部分（总2729字）
1%(36)	1%(36)	21_ZF1821334_张加杰_第3部分（总3651字）

中图分类号：TP3
论文编号：10006ZF1821334
夕匕京甬宝航天大學
专业硕士学位论文
基于语音的家居控制系统
的设计与实现
作者姓名张加杰
学科专业软件工程
指导教师王丽华
培养院系软件学院
Design and Implementation of Home Control System Based on Voice
A Dissertation Submitted for the Degree of Master
Candidate: Zhang Jiajie
Supervisor: Prof. Wang Lihua
School of Software
Beihang University, Beijing, China
中图分类号：TP3
论文编号：10006ZF1821334
硕士学位论文
基于语音的家居控制系统的设计与实现
作者姓名张加杰申请学位级别硕士学位
指导教师姓名王丽华职称教授
学科专业软件工程研究方向人工智能
学习时间自 2018年9月10日起至年月日止
论文提交日期 2021年12月12日论文答辩日期年月日
学位授予单位北京航空航天大学学位授予日期年月日
关于学位论文的独创性声明

本人郑重声明：所呈交的论文是本人在指导教师指导下独立进行研究工作所取得的成果，论文中有关资料和数据是实事求是的。尽我所知，除文中已经加以标注和致谢外，本论文不包含其他人已经发表或撰写的研究成果，也不包含本人或他人为获得北京航空航天大学或其它教育机构的学位或学历证书而使用过的材料。与我一同工作的同志对研究所做的任何贡献均已在论文中作出了明确的说明。

若有不实之处，本人愿意承担相关法律责任。

学位论文作者签名： 日期： 年月日

学位论文使用授权书

本人完全同意北京航空航天大学有权使用本学位论文（包括但不限于其印刷版和电子版），使用方式包括但不限于：保留学位论文，按规定向国家有关部门（机构）送交学位论文，以学术交流为目的赠送和交换学位论文，允许学位论文被查阅、借阅和复印，将学位论文的全部或部分内容编入有关数据库进行检索，采用影印、缩印或其他复制手段保存学位论文。

保密学位论文在解密后的使用授权同上。
学位论文作者签名： 日期： 年月日
指导教师签名： 日期： 年月日

摘要

随着互联网到移动互联网的转变，互联网本身的结构发生了巨大的结构变化，而随着移动互联网的成熟，智能化已经成为新的发展趋势和主要需求。在大规模数据和大规模算力的驱使下，人工智能技术也进入了智能化引领的发展阶段，人工智能技术正在越来越广泛地应用在移动互联网领域。语音识别技术作为人机交流接口的关键技术也在根本性地改变人们使用互联网的方式，5G的规模部署和物联网普及应用使我们进入了万物互联时代。而人工智能语音识别与智能家居的结合作为一个典型的应用，具有代表性的是各互联网厂商推出的人工智能音箱，将音箱与智能家居结合作为智能生态，这是互联网当前发展的一个方向。简单高效的控制，智能的语音对话等，为我们提供了便利同时，支持设备不够完善，识别率低，误识别等问题也是凸显出来。如何在支持控制大部分家居的同时保证在有一定外接干扰的情况下仍能正常工作是本文的目标。

本文结合当前市面上智能音箱与智能家居的能力，针对上述问题进行进一步设计与完善。通过需求分析和整体架构的调研，首先确定了系统以C/S为基础架构。在保证算力的同时，为尽量多的支持控制家居种类，确定了开发板和系统的选型，Linux系统下具有丰富的驱动和完善的系统调用接口。硬件接口层和控制层使用C语言实现，应用层使用少量C++和shell语言进行处理，在调研对比了当前比较新的语音技术后，确定了使用Kaldi作为语音识别模块的解决方案，在此之上使用bash和python脚本做了一些工具具有良好的扩展性。然后针对上述模块和核心算法进行深入分析，进行了详细的设计和实现。最后针对本系统进行全面测试和分析评估，最终的测试结果可以证明该系统满足预期。

该系统在落地实现后可充当红外遥控控制大部分家电产品，包括一些比较老的产品，可切实的解放双手，并且在没有外网的情况仍然可以使用。另外展示与公司产品经理后受到一致好评，激发了产品经理进一步完善产品的热情。

关键词：物联网，语音识别，智能家居，红外

Abstract

As the development of the mobile Internet enters a new direction, intelligence in the mobile Internet has become a new development trend and main demand. In the stage where intelligence leads the development, artificial intelligence technology is becoming more and more widely used in the field of mobile Internet.Speech recognition technology, as the key technology of human-machine communication interface, is also fundamentally changing the way people use the Internet. The large-scale deployment of 5G and the popular application of the Internet of Things have brought us into the era of the Internet of Everything.As a typical application, the combination of artificial intelligence speech recognition and smart home is representative of artificial intelligence speakers introduced by various Internet manufacturers. The combination of speakers and smart homes is a smart ecology, which is a direction of the current development of the Internet. Simple and efficient control, intelligent voice dialogue, etc., provide us with convenience. At the same time, the support equipment is not perfect, the recognition rate is low, and the problems of misrecognition are also highlighted.How to support the control of most homes while ensuring that it can still work normally with a certain amount of external interference is the goal of this article.

This article combines the capabilities of smart speakers and smart homes currently on the market to further design and improve the above problems. Through demand analysis and overall structure research, it is first determined that the system is based on C/S. While ensuring computing power, in order to support as many types of homes as possible, the selection of chips and systems has been determined. The Linux system has a wealth of drivers and a complete system call interface.The hardware interface layer and control layer are implemented in C language, and the application layer is processed in C++ and shell language. After investigating and comparing the current relatively new voice technology, it is determined to use Kaldi as a solution for the speech recognition module. On top of this, some tools have been made with bash and python scripts with good scalability. Then it conducts in-depth analysis on the above-mentioned modules and core algorithms, and carries out detailed design and implementation. Finally, a comprehensive test, analysis and evaluation is carried out for the system, and the final test result can prove that the system meets expectations.

The system can be used as an infrared remote control to control most home appliances, including some older products, after being implemented on the ground. It can effectively free your hands and can still be used without an external network. In addition, after the presentation and the company's product manager, they were unanimously praised, which inspired the product manager's enthusiasm to further improve the product.

Key words: Internet of Things, Speech recognition, Smart home, infrared

目录

绪论

研究背景及意义

本课题主要是受当前流行的智能语音对话音箱启发，结合我当前从事的公司相关项目，我相继在百度智能生活事业群组小度智能音箱业务部和语音技术部门工作，主要工作内容是小度智能音箱的开发，先后开发了小度智能音箱的语音SDK，OTA（Over the Air）空中升级，蓝牙配网，底层Linux系统和驱动等功能。当前智能音箱普遍重云轻端，控制功能薄弱，强依赖网络，将主要算法如ASR（Automatic Speech Recognition），NLP（Natural Language Processing）等放在云端，音

箱端仅负责数据采集和语音合成（Text To Speech）播放，端云之间通过一套自研协议进行通信，基于这种架构形式可以有效削减硬件成本，相应的音箱端可实现的功能受到限制，无法发挥嵌入式系统控制优势，音箱端可新增功能受限，降低了灵活性。基于此，设计一款基于深度学习的嵌入式高性能语音控制系统非常有必要，应用场景可涵盖手机，车载，智能家居，行业智能终端等，该系统可将唤醒，识别和控制功能全部放在音箱本地系统中，可实现离线语音唤醒和识别功能，并能极大程度增加系统的控制功能，对该控制系统进行有效设计和实现是本课题的主要内容。

研究背景

在移动互联网上一轮的发展周期中，最具代表性的PC互联网时代，人们利用键盘、鼠标和显示器来获取信息，而且可以很好的解决人机交互问题。随着移动互联网的新一轮发展周期，语音技术和智能家居的进一步发展，人们通过移动互联网，可以随时随地的利用各种移动终端设备访问网络。但是伴随着移动互联网的快速发展，原来获取信息和人机的沟通的方式已经不能得到充分满足，社会向着信息化和智能化方向快速的发展，新的一种人机交互方式产生了，人类一直以来通过语音进行交流，是人类最自然最便捷的信息获取和沟通的方式，人机通过语音交互的方式引起了业界的广泛关注。另外一个方面，随着人工智能领域的发展，机器学习、深度学习技术的突破，大数据技术以及自然语言理解语音合成等能力的提升，进一步带动了一波产业热潮。国内包括继科大讯飞、捷通华声之后，互联网巨头像阿里，百度，腾讯等都在往智能语音领域发力。具有代表性的基于语音交互的智能音箱已经走入千家万户，越来越多的人们习惯于用口语化的指令进行查询和接受信息，如播放音乐，和智能音箱聊天对话，通过语音进行查询天气情况，还可以设置日程提醒、预订机票和酒店网上购物等非常实用的生活服务和功能。伴随着智能家居产业的蓬勃发展，制造业，家电设备，安防，工控等方面的成熟，带动主要制造商雨后春笋般推陈出新了许多智能家居产品，通过无线传感技术来增加使用体验和舒适度。对于大众用户来说，随着消费需求逐渐升级成熟，通过语音进行集中控制更多的智能产品已日渐成为主流趋势。

研究意义

本选题在智能音箱普遍重云轻端的大背景下，实现一套在线离线一体的语音控制系统，相较于市面语音智能音箱具有更强力的控制功能，即使在一些没有网络的场景下仍然能离线识别控制，稳定性强，识别率高，适应场景能力更强。另一方面，基于算法和工程的完整系统可以回馈算法，使工程和算法互补互益，更好的提升算法能力和效率。基于当前工作项目，目前主要实现工程化及算法周边内容，对一些优秀的开源项目进行调研，利用算法和深度学习的强大能力进行深入的设计和实现，基于此选题，可以更深入的学习深度学习在智能语音交互系统中的实现，通过自己的思考和实现可以巩固自己所学和工作任务，更希望能在这个领域做出自己的一份贡献。

国内外相关研究现状及对比分析

国外研究现状

目前国外语音技术与智能家居的结合产物最具代表性的也属智能语音音箱，而且已经被广泛的推广，如Amazon的Echo，谷歌的GoogleHome，苹果的HomePod等均已发布上市，备受消费者欢迎。亚马逊在2014首次发布基于语音控制的智能音箱Echo，该产品让人眼前一亮，亚马逊将其定位为一位家庭助理，赋予其语音交互和灵活处理语音需求的能力，它装载了一个名为Alexa的语音交互系统。这个系统在后续的迭代中能力不断增强，又赋予了它家居控制的能力，不仅可以通过手机进行被动控制，还可以通过语音进行高级的外部智能家居设备进行操控。不止是家庭助理，它还可以像一个知心的朋友一样随时随地的跟人沟通交流，甚至在你无聊的时候给你讲笑话。另一方面，不管是购物还是家居控制，包括一些信息查询等功能都是基于语音的交互来完成，这样全新的交互方式已经突破人们的传统观念。语音交互一直处于实验阶段，一直到Echo的成功发布和售卖，也同样带动了语音交互的发展。随后各个互联网巨头和厂家开始推出自己的造型各异的语音交互机器人。这些机器人更加注重家居场景，而从手机端到智能家居控制的垂直场景的过度，最根本的是各大厂商从技术角度突破并解决了远场拾音与多种场景下的语义理解问题，利用先进的声学算法和硬件创新使得语音交互成为可能，并且在家庭场景中和智能家居结合，面向真实客户群进入千万家庭，为人们生活提供了更多乐趣和便利。

国内研究现状

目前语音交互和家居控制在国内发展日益蓬勃，当前国内各厂家的普遍具有自己的语音交互和控制能力。家居生态中首屈一指的是小米科技，其推出米家家居生态做的比较全面，而家居生态中以小爱音箱为核心，打造基于语音的家居控制生态。科大讯飞联合京东在2015年首次推出国内的第一款基于语音交互的智能音箱“叮咚”，其在功能和形态上都比较类似亚马逊的Echo。虽然比国外稍微晚了一些，但国内的发展势头非常强劲，从2016年开始，语音交互和家居市场突然爆发，各大厂家均发布了自己的带有家居控制的语音交互音箱，而且在很短的时间内各个厂家的各类语音和家居交互产品的版本都会进行迭代，不断优化用户体验，而且效果都会出现非常大的提升，产品的落地和用户的使用和反馈推动了语音交互和家居系统的不断完善，使语音交互技术链条越来越成熟。而家居方面各个厂家开始向智能方向转型。作为语音交互的基础底座，语音前端信号处理是一项极具挑战技术难点，声智科技和科大讯飞在这方面首先发力，并持续改进这项技术。随着市面上新产品的落地和数量的增加，真实场景下的用户数据和音频被不断的生产和收集，这对于互联网大规模的数据云存储能力得到了用武之地，可以轻松存储终端产品在使用过程中上报上来的数据进行收集并加以筛选和利用。反过来大量的数据正是人工智能的前提，机器学习和深度学习在海量数据的加持下不断训练优化模型，目前小米、阿里、百度等语音识别准确率高达97%，各互联网厂家在智能语音的基础上不断拓展家居生态，海尔推出了海尔智家，小米推出的小米生态链，阿里基于天猫精灵的智能家居等。

对比分析

全球范围内智能音箱销量都在持续增长，带动了智能家居生态的发展，据报告显示，以智能家居为核心的语音控制音箱在全球市场的规模达到4000万。中国已经成为仅次于美国的全球第二大智能产品消费市场，其中国产的智能音箱更是占据了大头。随着人工智能语音技术的不断突破，国外的互联网公司如亚马逊，谷歌，国内的互联网公司如百度，阿里巴巴，小米等均推出了自己特色的基于语音的智能家居生态体系，如亚马逊的Amazon Robot，百度的语音交互系统 DuerOS和小度智能家居生态，小米的小爱同学与小米生态链等都非常强劲，支持的家居种类也非常全面。

研究目标及研究内容

研究目标

本论文的研究目标是实现一套基于深度学习的智能语音控制系统，探索最前沿的技术和解决方案，并加以研究和优化，并增进工程落地能力。具体的研究目标如下：

嵌入式开发板和控制系统的选型
语音关键词唤醒功能设计实现
语音识别的设计实现
控制系统的设计实现
语音合成功能设计实现
研究内容
本基于语音的家居控制系统的研究内容根据研究目标可分为具体下列几个方面:
1）嵌入式开发板和系统的选型
依据市面已有的智能音箱进行开发板对比，确定开发板型号，结构设计上对麦克风采集到的信号的影响，包括结构震动，腔体效应，导音孔，音腔谐振自激放大，现在大多数产品同时有麦克风和扬声器，而扬声器放出的声音会被麦克风采集到，回采（硬件或者软件实现）到的扬声器信号对最终的AEC（回声消除）影响比较大。另外操作系统可选择Linux系统来搭建开发和执行环境。
2）语音唤醒功能
对于智能产品的用户来说,唤醒就是语音交互的第一入口,唤醒效果的好坏直接影响到用户的第一体验。用户使用时通过指定唤醒词将系统唤醒，如”小贾小贾”。该功能依托音频的声学算法和深度学习，对声学算法AEC进行研究和调优，设计关键词唤醒训练模型并进行调参训练。
3）语音识别
用户唤醒系统后，发起语音请求控制家居设备，如”打开电灯”，系统收到指令音频，经过一系列声学算法，最后送入识别引擎转成文字。该模块同唤醒模块一样，需要研究声学算法并进行语音识别模型训练。经过声学算法对音频进行降噪增强，在语音识别中声学算法+深度学习模型具有非常好效果，这一部分主要研究语音识别相关解决方案和开源工具，以及如何进行音频处理和并根据语音识别模型搭建有效的识别引擎。
4）控制系统
语音识别结果为文本内容，通过字典匹配查找待执行指令，匹配成功后通过局域网通信协议将指令发与外设，电灯成功打开。这一部分需要研究局域网通信协议如MQTT或蓝牙BLE，红外控制等，需要在开发板选型时考虑芯片是否支持相关功能,电灯打开后发送打开成功指令给系统，系统通过语音合成合成打开成功音频并播放，该模块可采用默认音频+简易语音合成合成引擎结合实现。
语音合成
许多互联网公司都提供了AI开发平台，平台提供了语音合成功能，本次研究会体验该功能并成功应用到当前系统。
本文组织结构
第一章是绪论。绪论部分主要讲述了我的论文基于语音的家居控制系统需要进行的研究背景和意义的探索分析，梳理并分析市面上控制系统的发展和现有技术成果。
第二章是系统需求分析。本章主要阐述了围绕基于语音的家居控制系统的核心功能进行相关的需求分析，并将系统需求从功能角度划分为功能性需求和非功能性需求，本系统的功能性需求比较注重该系统支持的功能和承载该功能所需要的底层功能性需求，最后说明非功能性需求，该需求强调系统性能和系统的实现效率等，与功能性需求相辅相成。
第三章是系统总体设计。本章从系统架构角度切入，对系统的功能结构，网络拓扑，系统数据库和接口等进行了设计。
第四章是系统详细设计与实现。本章对系统各个模块进行了详细的设计并进行开发和落地实现。另外本章还对语音唤醒，语音识别等关键技术进行了分析并给出了解决方案。
第五章是系统测试分析。本章从系统的角度出发，设计了比较全面的测试用例和测试方法，对系统功能和非功能需求进行了摸底和测试，测试结果符合预期。是系统设计后对系统整体实现完整度和能力的评估，是比较重要的一环。
第六章是总结与展望。本章对整个系统从需求到设计再到实现进行了整体总结，从总结内容得到后续需要继续完善的细节，并做出了具体的规划。对后续的系统落地和商业化进行了构思和展望。
本章小结
本章首先介绍了基于语音的家居控制系统需要进行的意义背景等方面的研究，根据当前的技术和产品方面的现状进行了国内和国外的对比，另外对于本论文的研究目标进行了设定，基于研究目标，设计了本系统在研究过程中需要实现的内容。本章是整个论文的基础，对后续的研究和设计工作具有指导意义。

第二章系统需求分析
2.1 现状分析
目前市面上的音箱具有重云轻端的特点，往往丢失了嵌入式系统本身灵活和控制的特点。目前家居虽然向着智能的方向发展，支持移动互联网的远程控制。但仍有大量的家居设备是按键控制，好一点的是红外或蓝牙，但相对功能单一，无法通过一个智能的语音中控进行统一控制。另一方面，智能语音的代表智能音箱，都是在有网络的环境下才能使用，在弱网或没有网的情况下基本罢工。

针对上述问题，本文从各个角度考虑，目标是设计一套能尽量多覆盖家居家电设备，对其进行控制，并且能够在有线离线环境下都能正常工作的家居控制系统。该系统具有如下特点：

尽量多的支持控制家居家电，包括已经支持WiFi，红外，蓝牙的偏智能家居类产品，以及不支持无线控制的家居，可以使用智能插座来控制开关。

可以胜任有网和无网环境，在弱网或没有网络的情况下仍然支持最基本的开关和简单控制指令。在有网的条件下可以支持更复杂的指令，并且可以做到网络实时监测，动态切换有网无网模式。

2.2 功能性需求分析

基于语音的家居控制系统功能性需求比较明确，示例如图1所示，用户通过“小贾小贾”唤醒系统，语音指令是“打开空调”，系统将识别该语音并分析用户意图，发送红外控制指令给空调，空调打开后，系统会通过扬声器播放“空调已打开”的提示术语。

图1 基于语音的家居控制系统功能示意图

由功能示意图，该系统功能性需求分为如下几个部分，各部分内容如下：

语音唤醒功能

作为整个系统的触发入口，唤醒功能是必需功能。当用户喊出语音关键词时可以唤醒系统。如功能示意图中的“小贾小贾”即为唤醒关键词，当系统接收到该关键词时，经过唤醒算法计算，判定系统被唤醒，系统唤醒后，通过网络探测模块镜像网络通路判断，将录音送入下一阶段，进行离线或在线语音识别。

语音识别功能

该功能的作用是将音频转化为文字。该功能根据系统当前网络状况分为在线语音识别和离线语音识别。当系统网络通畅的情况下，采用在线语音识别，在线语音识别的优点是识别率高，识别准确，该功能的实现选择三方平台支持，如使用百度AI开发平台提供的在线语音识别服务，该平台提供了优秀的开发服务，具有比较全面的API接口。

当系统网络不好的情况下，采用离线语音识别，离线语音识别的优点是不需要依赖网络，使系统具有一定的抗干扰能力。该功能需要处理后的音频，需在录音后将音频经过AEC算法进行内噪消除，之后将音频经过离线识别引擎，该离线识别引擎基于深度学习，通过采集语料训练识别模型。语音识别需求如图2所示。

信号处理

语音输入

解码器

文本输出

声学模型

语言模型

图2 语音识别需求图

网络探测功能

该功能可以是一个独立任务，每隔一分钟进行网络状态判断并设置网络状态标志位。网络状态的判断可以通过访问因特网上特定网址进行判断，如www.baidu.com或www.sina.com。当用户发起识别请求时能立即拿到相应状态进行在线离线模式切换，并且网络探测与状态获取应设置为异步模式，防止因网络探测造成的系统卡顿。

识别结果关键词分类和匹配功能

系统应对语音识别转化后的文字进行有限的关键词分类和匹配，匹配到可控制设备关键词和动作关键词后查找对应设备的控制指令，这一块可通过预先设置字典记录设备关键词，关键动作和控制指令。

外围通信控制功能

作为一个嵌入式系统，该系统硬件上应具备最基本的控制功能模组，如红外，蓝牙，WiFi。开发板选型时应当注意支持的外设类型及相关驱动模块。

麦克风音频数据采集功能

该系统处于安静环境中应处于静默状态，麦克风正常拾音，送入离线唤醒引擎。该阶段需要注意硬件上支持参考路信号回采，否则做不了AEC降噪，无法在有内噪的情况下唤醒系统。

外设通信控制功能

系统通过匹配到的设备关键词，关键动作和控制指令对设备发起控制指令。这一块要实现对应的内设的驱动和与外部设备交互的协议。

控制模块可以接受外部设备发送回来的状态，同样的通过字典的形式匹配对应状态文本。结合唤醒与外设控制，用户可以使用的功能如图3所示

图3 使用控制用例图

语音合成功能

该功能的作用是文本转音频，将传入的文本经过处理后转化为音频结果，在处理状态文本时可以通过判断网络状态选择在线语音合成或离线音频，语音合成将文本转化为可播放的音频文件。在线功能同样可接入三方平台，使用其提供的语音合成功能。离线音频为系统默认提供音频，数量比较有限。语音合成功能为用户提供可识别的控制结果，是与用户交互非常重要的一环，其交互如图4所示。

图4 控制结果返回图

音频播放功能

系统唤醒后，不管语音识别与控制成功与否，都应有相应的提示话术，如“空调打开成功”“指令无法匹配”等。该功能依赖硬件设计，需要将播放音频进行回采。系统将音频文件解码，送入Audio驱动，通过扬声器播放音频数据。

2.3 非功能性需求分析

唤醒率与误唤醒率需求

系统唤醒率与误唤醒率非常重要，唤醒是该语音控制系统的入口，若唤醒率低，会严重影响系统的可用性，要保证唤醒率

在90%以上（100次唤醒有90次以上可以唤醒）。另一方面，误唤醒率高，也会在用户没有操作的情况下对用户造成干扰，误唤醒率应控制在2次每12小时（电视噪音下）以下。

识别率，字准句准需求

系统的识别率是影响用户控制体验的另一个关键指标，在线识别率依赖于服务商，识别率会比较高，离线识别字准应在90%以上，句准应在80%以上。

安全性需求

安全性需求是一个比较重要的问题，因为系统会不断录音（送入唤醒引擎），会有泄露用户隐私的风险。本地录音数据应使用AES加密存储。另外防止系统入侵，应关闭相关登录接口，如串口，telnet，ssh等服务。

外设控制满足度需求

外设控制应满足用户需求，在用户表达不够明确的情况，应尽量揣摩用户心理，如”小贾，小贾，空调”，若当前空调处于关闭状态，可以发送空调打开指令，用户控制满足度应在80%以上。

完整性需求

该系统应满足至少三个智能家电的控制，并且控制能力包括开、关、一项自带功能控制三种能力。

可扩展性需求

在满足最基本完整性需求下，系统应具备充分的扩展性，可扩展能力如下：一唤醒词定制；二离线识别内容扩展；三可控制外设的扩展等。

2.4 本章小结

本章首先对市面上的一些智能家居语音控制系统进行了现状分析，针对分析出的痛点进行需求分析，将本系统的需求分为功能性需求和非功能性需求系统。从功能性需求角度将系统分为几个功能模块，并对功能模块进行一一分析。从非功能性需求角度分析，本系统应具有一定性能和使用性。

3. 21_ZF1821334_张加杰_第3部分　　总字数：3651

相似文献列表

去除本人文献复制比：1%(36)　　文字复制比：1%(36)　　疑似剽窃观点：(0)

1	基于矢量地图的车辆监控系统的设计与实现	1.0%（36）
	郝向宁(导师：周东清) - 《大连理工大学硕士论文》- 2011-04-15	是否引证：否
2	基于B/S的网上考试系统的设计与实现	1.0%（36）
	吴天(导师：刘显德;吕桂友) - 《东北石油大学硕士论文》- 2011-06-16	是否引证：否

原文内容

第三章系统总体设计

3.1 系统设计原则

本系统设计应遵循以下几个原则：

系统性原则。在本控制系统设计中，遵循系统性的原则，从整个语音控制系统的角度进行设计，系统分为多个子系统，每个子系统又分为几个模块，子系统之间和模块之间都要相互配合，按流程进行设计和实现保证系统的完整性一致性等。

灵活性及可变性原则。本控制系统要有一定的灵活性，随着可控制外设的增加，要有一定的灵活性和扩展性，能够对外界环境变化的适应能力，比如在网络可用的情况下支持在线功能，在网络不稳定或断连的情况下支持离线模式。

可靠性原则。本控制系统要有一定的抗干扰能力，语音唤醒和语音识别对于外部环境具有一定的要求，在系统的设计过程中为了加强噪音情况下的唤醒率和语音识别准确度，要考虑增加一些降噪功能，在硬件和软件算法上都可以采取必要的降噪或语音增强方法能够抵御外界干扰和受外界干扰时的恢复能力。

经济性原则。本控制系统硬件上在能够满足性能开销的基础上尽量采用经济性的芯片和外网控制模组，这样可适当节约成本，减少系统不必要的开销，提高性价比，为以后商业提供基础。

3.2 系统架构设计

整个基于语音的家居控制系统分为硬件适配层，控制层，算法层，应用层四个层次。每个层次按功能模块又各分成几个功能，层次之间相互通信交互。如图5所示的系统架构设计图，该架构设计图比较直观的展示了整个系统的层次结构。

算法层

硬件接口层

麦克风

红外

WiFi&蓝牙

播放器

网络探测

控制信号接收

控制信号发送

播音

录音

指令执行
在线识别
离线识别
语音唤醒
降噪
传输层
应用层
语音合成
指令识别
硬件系统
指令结果识别
软件系统
图5 系统架构图

系统及硬件：该层包括操作系统，驱动，硬件及相关硬件外围接口。操作系统使用Linux，该系统包含了必要的硬件及接口驱动，本系统使用的相关外围接口包括麦克风，扬声器，WiFi和蓝牙模组，红外发射器等。

硬件适配层：该层主要作用是向上为应用提供统一接口，向下屏蔽硬件和驱动细节。该层集成了包括麦克风，播放器，WiFi蓝牙，红外等功能模块的驱动接口，提供了统一的读写和控制接口。该层是可移植性的桥梁，简化上层设计，提高开发效率，使本系统结构更加清晰。

传输层：该层包含了录音，播音，控制信号发送，控制信号接收四个功能模块。传输层在硬件适配层的基础上实现了基本的输入输出控制等逻辑，为算法层和应用层提供了必要数据来源和控制通路。

算法层：该层将系统中相关算法进行了整合，包含语音唤醒，离线识别，在线识别，降噪算法，语音合成等功能模块，是整个系统计算的核心。

应用层：该层承载了主要的业务逻辑，是基于语音的家居控制系统的控制核心。主要包括网络探测，识别结果解析，控制结果解析等功能。

3.3 系统功能结构设计

结合语音控制和智能家居的实际情况，并通过对该系统的需求分析设计出的系统的功能结构如图6所示。

图6 系统功能结构图

唤醒子系统：该模块包含麦克风数据采集并通过声学算法模块进行处理，最终将数据送入唤醒引擎，唤醒引擎采用Snowboy，该工具是一款高度可定制的唤醒词检测引擎，具有高度可定制，轻巧，可嵌入的特点，比较符合当前系统需求。唤醒模块流程如图7所示。

降噪音频
录音
麦克风
声学算法
唤醒引擎
图7 唤醒模块流程图

语音识别子系统：该子系统主要包含离线识别和在线识别引擎。在线识别引擎主要通过三方接口高质量的完成语音识别转化；离线识别通过深度学习训练的识别模型，当在线识别不可用时采用离线识别，大大提高系统的可用性，系统离线语音识别模块的结构设计如图8所示。

图8 离线语音识别模块结构图

家居控制子系统：主要包含指令识别和指令执行，对语音识别得到的指令进行匹配，外设的驱动控制和指令发送，另外该子系统还接收外设的指令执行结果，最终用来通知用户指令的执行情况。

语音合成子系统：该子系统包含在线语音合成和音频数据播放两个模块。在线语音合成将执行执行结果的文字通过线上平台提供的语音合成能力转化为音频文件，而音频播放功能则将音频文件通过扬声器播放出来，是人机交互的重要模块。

3.4 系统网络拓扑

整体网络拓扑如图9所示，其中家中智能家电包括冰箱，蓝牙开关，空调，手机，智能灯等，这些智能设备均通过无线WiFi模组与家中路由器WiFi相连，同路由器下的设备组成局域网，本控制系统通过WiFi局域网与冰箱连接。本控制系统支持红外发射，可充当空调遥控器开关。

图9 系统网络拓扑图

另外本控制系统上有蓝牙模组，可通过蓝牙配对与家中蓝牙智能开光相连。路由器WiFi外部与通信厂商局端路由器相连，通过该路由器可访问外网，图中我们访问了两个服务地址，一个是www.sina.com，一个是www.ai.baidu.com。www.sina.com这个服务地址的作用是作为外网探测的地址，我们可以通过这个地址来判断当前网络是否可以访问外网，同时也作为外网是否通畅的测试地址。www.ai.baidu.com这个服务地址提供了我们在线模式下的语音识别功能，我们通过API接口访问这个服务地址并实现在线的语音识别。与www.sina.com相同的一个作用是www.ai.baidu.com也同样作为外网探测地址。

3.5 接口设计

1) 音频采集

```
void thread_audio_recorder_by_alsa();
```

降噪接口

```
void thread_audio_aec_process();
```

网络探活

```
void thread_network_accessable();
```

接口
离线语音识别请求
void thread_asr_process_offline();
在线语音识别请求
void thread_asr_process_online();
结果解析
void system_instruct_process();
控制指令发送
void system_instruct_2_hal();
语音合成接口
在线语音合成请求
void thread_语音合成_request_online();
离线语音合成请求
void thread_语音合成_request_offline()
音频播放接口
void thread_audio_play();

3.6 系统采用的关键技术、难点

本课题的关键技术难点在于离线语音识别模块和外设控制模块。

离线语音识别模块是通过人工智能语音识别技术将音频信号转变为人类可识别的文字，进一步通过自然语音理解技术转变为机器可以执行的指令的技术。而这一切都在本地系统上执行，不需要通过网络和服务器。目的就是给机器赋予人的听觉特性，听懂人说什么，并作出相应的行为。本系统采用的离线语音识别系统需要语言模型和底层的声学模型，两个模型都需要通过训练得到。需要语音样本库进行特征提取，然后通过声学模型训练。语言模型同样需要语音样本库进行语言模型训练。执行流程是语音输入经过特征提取后进行语音解码和搜索再加上词汇模块得到最后的文本输出，一个连续语音识别系统的结构如图10所示。

图10 语音识别系统结构图

离线语音识别难点有两方面，一是噪声的困扰，因为噪声环境有各种各样的声源，人声，汽车声，走路声，音乐等，而各类声源的处理是目前公认的技术难题，在机器的角度根本无法从复杂多样的背景噪音中分辨出人声，对于千差万别的背景噪声，即使是深度学习也无法训练出完全匹配的真实环境。因此，要实现噪音下的语音识别要比在安静的场景下难得多，这也是无法避免的问题。二是模型的有效性，识别系统中包括语言模型、词法模型在短句和少量词汇的情况下效果还可以，但要放在在大词汇量和一些连续语音识别的场景中还不能完全正常的工作，为此，除了基于大数据的训练，还需要有效地结合语言学、心理学及生理学已经一些语言习惯等其他学科的知识。另外，语音识别系统在实验室演示效果虽然不错，但真正要转化到商品的还有许多具体的技术细节等问题需要解决。

以上难题的解决，需要很高的技术门槛，独自解决不太现实，因此选择站在巨人的肩膀上，使用开源语音识别框架结合音频降噪算法AEC进行处理解决。目前开源世界里提供了多种不同的语音识别工具包，为开发者构建应用提供了很大帮助。但这些工具各有优劣，需要根据具体情况选择使用。大多基于传统的 HMM 和N－Gram 语言模型的开源工具包，通过对比选择Kaldi作为本次语音识别难点的解决方案。

3.7 本章小结

本章主要介绍了系统的总体设计，从系统的整体架构开始，详细设计了各个子系统，又对各个子系统所包含的模块进行了细致的分析和设计，展示了系统的各个模块和整体运行的俯视图，对于后续的详细设计和实现具有指导作用。

4. 21_ZF1821334_张加杰_第4部分　　总字数：10864

相似文献列表

去除本人文献复制比：0%(0)　　文字复制比：0%(0)　　疑似剽窃观点：(0)

原文内容

第四章系统详细设计与实现

4.1 控制系统的底层设计与实现

我们都知道智能系统的“智能”主要依靠它的心脏，主控芯片提供算力。一个完整的控制系统，其核心为底层芯片和系统组成的平台，该平台为上层应用和算法提供了基本的环境和算力。探索AI在嵌入式设备上的使时式和性能表现，指定可于评估的智能嵌式案，让AI赋能评估，使系统的实现和表现更加准确效。

4.1.1 开发板选型

从系统、计算芯类型、开发环境、态完善度等多个因素考虑，选择以下案进探索并调研。

瑞芯微RK3399开发板
瑞芯微RK3399PRO开发板
华为Atlas200DK开发板
Nvida JetSon TX2开发板
Nvida Jetson Xavier NX开发板

选型调研过程，以体关键点识别为例，对比如表1所示。

表1 开发板功能对比表

序号	方案	加速芯片类型	环境搭建难度	操作系统	开发语言	参数/算力	主流深度学习框架支持	主流框架的模型支持
1	瑞芯微RK3399	GPU	中等（RKNN等环境编译较麻烦）	AndroidLinux	Python、C/C++Java	预估2.0Tops	主流框架无加速	同框架
2	瑞芯微RK3399PRO	GPUNPU	中等（RKNN等环境编译较复杂）	AndroidLinux	Python、C/C++、Java	3.0Tops	主流框架NPU加速，只能官RKNN、Rockx框架	Caffe、 TF、TF- Lite、ONNX、Darknet 等模型转换
3	华为Atlas200DK	AI芯片	复杂	Linux	C/C++	22Tops	主流框架无AI芯片加速，只能官方框架	Caffe、 TF
4	Nvida JetSon TX2	GPU	中等	Linux	Python、C/C++	1.26TFlops	主流框架都支持但无加速TensorRT可加速	同框架
5	Nvida Jetson Xavier NX	GPU	中等	Linux	Python、C/C++	21Tops	都支持但无加速TensorRT可加速	同框架

序号方案加速芯片类型环境搭建难度操作系统开发语言参数/算力主流深度学习框架支持主流框架的模型支持

1 瑞芯微RK3399 GPU 中等（RKNN等环境编译较麻烦） AndroidLinux Python、C/C++Java 预估2.0Tops 主流框架无加速同框架

2 瑞芯微RK3399PRO GPUNPU 中等（RKNN等环境编译较复杂） AndroidLinux Python、C/C++、Java 3.0Tops 主流框架NPU加速，只能官 RKNN、 Rockx框架 Caffe、 TF、TF- Lite、 ONNX、 Darknet 等模型转换

3 华为Atlas200DK AI芯片复杂 Linux C/C++ 22Tops 主流框架无AI芯片加速，只能官方框架 Caffe、 TF

4 Nvida JetSon TX2 GPU 中等 Linux Python、C/C++ 1.26TFlops 主流框架都支持但无加速TensorRT可加速同框架

5 Nvida Jetson Xavier NX GPU 中等 Linux Python、C/C++ 21Tops 都支持但无加速TensorRT可加速同框架

以上数据是根据实际测量与开发板官网数据进行整合的，另外为了有一个更全面的认识和体验，做了更进一步的探索，开发板功能深度体验如表2所示。

表2 开发板功能深度体验表

序号	方案	深度体验	使用感受
1	瑞芯微RK3399	a.TF-Lite框架，posenet模型，5-10FPS。	性能般，对性能要求较低的场景可以使
2	瑞芯微RK3399PRO	a. Rockx框架，pose模型，40FPS，但精度很差。 b.RKNN框架，posenet模型，10FPS。 c.加载Caffe的openpose模型，4FPS。	性能般，相RK3399有NPU加速，对性能要求般的场景可以使
3	华为Atlas200DK	a. 验证官Demo。	从搭建环境Demo跑通，踩坑最多的案，档不明确、模型持少，使较为繁琐，态环境差。宣称性能不错，但不适合快速开发产品。
4	Nvida JetSon TX2	Pytorch下 yolo+pose_resnet50模型串联，10FPS。	性能满足大部分使用场景，同PC一样，基于CUDA，生态完善，部署成本低。
5	Nvida Jetson Xavier NX	Pytorch下 yolo+pose_resnet50模型串联，10FPS。	性能强于TX2，满足大部分使场景，同PC一样，基于CUDA生态完善，部署成本低。

序号方案深度体验使用感受

1 瑞芯微RK3399 a.TF-Lite框架，posenet模型，5-10FPS。 性能般，对性能要求较低的场景可以使

2 瑞芯微RK3399PRO a. Rockx框架，pose模型，40FPS，但精度很差。 b.RKNN框架，posenet模型，10FPS。 c.加载Caffe的openpose模型，4FPS。 性能般，相RK3399有NPU加速，对性能要求般的场景可以使

3 华为Atlas200DK a. 验证官Demo。 从搭建环境Demo跑通，踩坑最多的案，档不明确、模型持少，使较为繁琐，态环境差。宣称性能不错，但不适合快速开发产品。

4 Nvida JetSon TX2 Pytorch下 yolo+pose_resnet50模型串联，10FPS。 性能满足大部分使用场景，同PC一样，基于CUDA，生态完善，部署成本低。

5 Nvida Jetson Xavier NX Pytorch下 yolo+pose_resnet50模型串联，10FPS。 性能强于TX2，满足大部分使场景，同PC一样，基于CUDA生态完善，部署成本低。

选型调研总结：

1）国产瑞芯微RK3399和RK3399PRO，性能般，开发难度般，可以满模型精度、以及实时性要求不的场景。优点是持Android系统。

2）华为Atlas200DK，算虽然强，但开发部署过于繁琐，且模型持较少态不完善，遇到问题不好解决，不太适合快速开发产品。

3）Nvidia JetSon系列，基于CUDA，性能强，迁移部署成本低，态完善，主流框架都可持，适合快速开发产品。Xavier NX TX2增加了TensorCore以及CUDA核数，性能更强，优先选择Nvidia Xavier NX。

4.1.2 Nvida Jetson Xavier NX环境搭建

Nvidia Jetson Xavier NX形状，外接口类似于树莓派的嵌入式主板，搭载了6核NVIDIA CarmelARMv8.264位CPU，GPU则是有384g个NVIDIA CUDA内核和48个Tensor内核的NVIDIA Volta架构，具有以太网接口，USB主机从机，两个SPI接口，四个TWI接口，六个UART接口，三个SD卡接口，两个I2S/PCM数字音频接口。采用八核cortex-A7处理器，SGX544 GPU，支持60帧的1080P视频播放，采用HawkView ISP，最高支持800万像素摄像头。内存为8GB的LPDDR4x@51.2GB/s，支持DDR3/DDR3L/LPDDR3/LPDDR2多

种规格，满足不同成本定位应用。存储支持EMMC4.5，采用FCBGA封装，345pin，14*14mm面积。支持4K 60Hz视频解码。基本上可以将Jetson Xavier NX视为一台适用于深度学习的算力强大的微型电脑。

材料准备，NX开发板+19V电源电流1A，32G以上TF卡，路由器、线X2，电脑和开发板组。

烧录固件，在Linux电脑上下载并安装Nvidia SDK Manager并运。下载镜像，使USB连接开发板和电脑，照软件提进烧录。安装所需软件和环境，与Ubuntu类似。安装Pytorch。使开发板进开发测试 。

4.1.3 硬件适配层的设计与实现

硬件适配层包括含了录音，播放，WiFi，蓝牙，红外等模块，这些模块依赖系统驱动的类型和版本，我们以Nvidia Jetson Xavier NX自带Linux系统为基准，对以上几个功能模块的驱动进行读写和控制的封装。

4.2 控制层设计与实现

4.2.1 蓝牙外设控制

蓝牙可以分为经典蓝牙和低功耗蓝牙，本系统采用低功耗蓝牙（BLE)。BLE协议栈组成如图11所示。

Physical Layer（PHY）
Link Layer(LL)
Host-Controller Interface (HCI)
Controller
Host
Logical Link Control and Adaptation Protocol(L2CAP)
Generic Access Profile(GAP)
Security Manager (SM)
Attribute Protocol(ATT)
Generic Attribute Profile(GATT)

图11 BLE协议栈

控制器层Controller包含了物理层，链路层，HCI层。物理层是BLE协议栈最底层，规定了BLE通信的基础射频参数，包括信号频率、调制方案等。BLE4的物理层是1Mbps的GFSK调制。LL层用于控制设备的射频状态，可工作于advertising、sacnning、Initiating、connecting四中状态。advertising和sacnning状态是配对出现的，一个设备处于advertising状态，就会向外部设备发送广播包，处于scanning状态的设备可以接收广播包。设备之间的连接流程为处于sacnning状态的设备收到advertising设备发来的广播包，然后发送一个连接请求给到这个广播设备，处于advertising的设备如果接受了连接请求报文，则两个设备就会进入连接状态，也就是成功建立了连接。HCI层是一种通信接口，它为主机内报文的通信提供了标准协议，HCI不仅可以属于硬件接口如USB，SPI，IIC等，还可以是软件定义的接口。L2CAP层对LL进行了一次简单封装。

GATT负责主从设备之间的应用数据交换。GATT作为使用的ATT的子流程的一个服务型框架。为主从设备交互数据提供Profile、Service、Characteristic等概念的抽象、管理。当两个设备建立连接后，就处于GATT服务器或者GATT客户端的角色。GAP是对LL层的一部分进行封装，主要用来广播，扫描和发起连接等。Security Manager属于一种安全方式，通过加密方式进行秘钥分配，该秘钥在配对过程中使用。GAP层的角色有广播者，观察者，外设和集中器四种。处于advertising状态的设备称为广播者，等待scanning设备称为观察者，当scanning设备发送连接请求，连接建立后，advertising状态的设备作为从机，scanning设备作为主机。GATT层分为客户端和服务器，服务器顾名思义是为客户端提供服务，客户端则是通过服务器提供的服务来获取需要的数据，服务器与客户端相互配合进行通信。本系统的蓝牙插座采用BLE进行控制。插座的控制时序如图12所示。

图12 BLE插座控制时序图

蓝牙插座是长期带电并且处于广播状态，针对BLE5.0协议栈，广播设备利用37、38、39主信道和其他的信道作为第二信道向外部发送广播数据，此时并没有指定接收者，所有的处于scanning状态的设备都可以接收到该广播数据。广播状态的数据包pdu结构如图13所示。

LSB
MSB
Header
（16 bits）
Payload
（as per the Length field in the Header）
LSB
MSB
PDU Type
（4bits）
TxAdd
（1bit）
RFU
（2bits）
Length
（6bits）
RxAdd
（1bit）
RFU
（2bits）

图13 BLE广播信道数据表PDU组成图

由16bit Header+payload组成。4bits PDU Type决定了广播状态，可以分为如下表格种类。

基于语音的家居控制系统上电后应用程序调用扫描插座接口，该接口扫描所有BLE广播，发现广播中包含蓝牙插座UUID的设备，得到返回状态“发现插座”。应用层开始连接插座，通过协议栈建立与插座的BLE连接，当两个设备建立连接后，GATT开始发挥作用，一个设备作为服务器，另一设备作为客户端，连接建立后，应用层主要任务就是通过GATT服务进行数据的获取和设置。GATT服务可以包含在一个GATT的服务器中，包括的服务有设备开关，电流增大等。插座的开关实际上是GATT profile写特性值，控制系统向插座特定的服务地址请求写入要设置的特性值，插座获取设置结果并将数据是否写入成功的信息反馈给控制系统，控制系统即可完成对插座的开关控制。

4.2.2 红外外设控制

本系统使用了红外控制功能，红外控制功能可以控制带有红外接收的家电。一般的红外遥控系统包含两个部分，分别为红外发射端和红外接收端，通用红外系统的结构图如图14所示。

外设红外接收端

主控红外发射端

编码调制

发射管

信号放大解调

解码输出

图14 红外系统结构图

主控的角色就是负责红外信号的发射，红外发射的部分主要包括红外编码调制和发射晶体管。外设的角色是红外信号的接收，主要包括光电信号的转换和解调解码放大等电路。我们平时用的遥控器的信号就是将我们摁下的按键对应的红外码进行编码调制，编码的信号一般调制在32KHz到56kHz之间的载波上，放大电路将信号进行放大，最后经过驱动电路通过红外晶体管发出，家电设备负责接收并解码识别，执行相应的功能。红外码发射时序如图15所示。

图15 红外码发射时序图

红外发射采用了NEC编码nec 编码格式

- 头节码： 8ms高电平 + 5.5ms低电平
- 节码0： 0.65ms高电平 + 0.65ms低电平
- 节码1： 0.65ms高电平 + 2.86ms低电平
- 结束节位: 0.75ms高电平
- 红外发码顺序为高位最后，先发低位
- 单键子码 ：头子码 + 16位系统子码 + 8位数据子码 + 8位数据子码 + 6反码
- 连续键子码：8ms低电平 + 1.15ms高电平 + 结束子位
- 简码发送周期：112ms

引导子码+系统子码（16位）+数据子码（8位）+数据子码反子码（8位）+结束码

全码发送：引导子码 + 系统子码（16位）+ 数据子码（8位）+ 数据子码反码（8位）+ 结束码。

简码发送：9ms高电平 + 2.25ms低电平 + 结束位。红外简码发送时序如图16所示。

图16 红外简码发送时序图

编码时高低电平以1，0表示，红外发送接口流程如图17所示。

图17 红外发射接口流程图

红外发射接口启动时需要先查询操作对应的红外编码，如“关闭空调”，则需要空调关机的红外编码，本系统控制的美的空调红外指令编码如表x所示。

表3 美的空调红外指令编码

命令	主编码	子编码
开机编码	L,A,A',B,B',C,C'；	
关机	S,L,A,A',B,B',C,C'	
风速	B7 B6 B5	自动 101低风 100中风 010高风 001
模式	C3 C2	制冷 00制热 01抽湿 10送风 11
温度	C7 C6 C5 C4	17℃ 000018℃ 000119℃ 001120℃ 0010...

命令主编码子编码

开机编码 L,A,A',B,B',C,C'；

关机 S,L,A,A',B,B',C,C'

风速

B7 B6 B5

自动 101低风 100中风 010高风 001

模式 C3 C2 制冷 00制热 01抽湿 10送风 11

温度 C7 C6 C5 C417℃ 000018℃ 000119℃ 001120℃ 0010...

GPIO节点需要通过Linux系统GPIO驱动接口进行设置，设置命令为echo 12 > /sys/class/gpio/export，命令成功后生成/sys/class/gpio/gpio12目录，即为该GPIO节点。设置GPIO为输出命令为echo out > /sys/class/gpio/gpio12/direction，只需将out字符串写入direction即可。

红外发射部分当前系统采用的是NEC Protocol 的PWM(脉冲宽度调制)标准。遥控载波的频率为29kHz，占空比为1:3，一般的简码重复中的延时为101毫秒，也就是两个上升沿中间需要有98毫秒的延时。本系统采用的NTC编码，是由指令高位编码，指令字码引导，指令编码键数据，指令反码键数据组成，指令字码引导是5ms的间隔关断和8ms的波形载波统一时序组成的。红外

码的发射会有一个引导码，接收端在接收的时候会先处理接收到的引导码，在后续的处理中能更好的控制引导码与指令编码的数据，控制其检测和各项时序之间的关系。主控发射端编码通过脉冲位置调制的方式，脉冲高点即为1，脉冲低点即为0，中间间隔0.55ms，为了防止传输过程中的误码，传送8位数据后会传送一个反码，。连续发送两包，第二包总是抓到，两包红外发送不能间隔太短，否则处理不过来，如果第二包不识别，可以加大发送间隔来解决。

4.2.3 WiFi外设控制

本系统可通过WiFi模块控制外部的智能开关，无线WiFi智能开关的组成是把传统的开关或者插座做成小的模组放到智能开关中，通过串口与WiFi模组进行通信，WiFi模组负责与主控制系统连接通信，内置TCP/IP协议簇。当主控系统发送开关指令时，WiFi模组通过协议解析出指令，并将指令通过串口发送给开关模块，开关模块执行指令对应的开或关操作。通常这些智能开关可以通过手机APP进行控制，本系统正是利用了APP的控制接口，模拟APP的开关控制。

4.3 音频处理模块设计与实现

4.3.1 音频采集

音频的采集和播放在软件上主要是写音频的驱动程序，同时提供接口给上层调用。

Linux中跟音频相关的就是大名鼎鼎的ALSA(Advanced Linux Sound Architecture)了。它是Linux上的音频子系统，在kernel space和user space都有相应的代码。kernel space里主要是音频的驱动程序，user space里主要是alsa-lib,也就是提供接口给上层应用程序调用。User space和kernel space通过字符设备进行交互。

Linux下本地录音模块流程如图18所示。

线程初始化
开始
创建录音对象
录音读取
录音完成等待
音频交织
回采信号
存放缓冲区
资源释放
开始

图18 录音模块流程图

1）通过读.wav格式的音频流，将音频流写入DMA缓存区，实现语音播放。

2）通过读DMA缓存区的音频流，实现录音。

首先要配置硬件参数，包括设置采样位数、通道数、采样率等，然后向DMA缓存区写或者读，实现播放和录音，接口设计如下：

打开PCM设备，录制回采信号，接口为snd_pcm_open(&handle,"default",SND_PCM_STREAM_PLAYBACK, 0); 打开PCM设备，录音音频snd_pcm_open(&handle, "default", SND_PCM_STREAM_CAPTURE, 0);

申请录音参数对象snd_pcm_hw_params_alloca(&params);初始化录音对象snd_pcm_hw_params_any(handle, params);设置录音对象模式为SND_PCM_ACCESS_RW_INTERLEAVED，调用接口为snd_pcm_hw_params_set_access(handle,params,SND_PCM_ACCESS_RW_INTERLEAVED);设置录音格式为SND_PCM_FORMAT_S16_LE，调用接口为snd_pcm_hw_params_set_format(handle, params,SND_PCM_FORMAT_S16_LE);设置PCM为双通道snd_pcm_hw_params_set_channels(handle, params, 8);设置采样率为48000

调用接口为snd_pcm_hw_params_set_rate_near(handle, params, 48000, &dir);

设置一个录音周期为32包snd_pcm_hw_params_set_period_size_near(handle,params, 32, &dir);使能配置，将参数设置到驱动上snd_pcm_hw_params(handle, params);

申请录音存储buffer，调用接口为snd_pcm_hw_params_get_period_size(params, &frames,&dir);打开一个文件，将录音音频流写入文件snd_pcm_writei(handle, buffer, frames);

编译时，需要引入alsa库，-L指定具体的alsa-utils类库的asound等，按如下方式进行编译。

```
gcc -o local_player local_player.c -L ./alsa-utils-1.1.5 -lasound -lm -ldl
```

测试包括播放和录音，通过扬声器是否正常播放音频文件，确认播放成功。通过dump出来的文件，用audiocity或其他软件打开并播放，确认录音功能正常。

4.3.2 音频降噪

完成对音频的采集后，我们需要对音频进行降噪处理，因为音频质量对于本系统至关重要，不仅影响系统的唤醒率，还对后续的语音识别字准句准影响非常大。降噪方法我们选择使用回声消除算法。对于音频，回声消除是为了消除机器自身发出的声音，不影响外界传递过去的声音。本系统在进行音频播放时，来自扬声器放出来的声音会混叠人声一起被麦克风录进去，严重影响信号质量。

本系统采用了Speex高性能语音编解码库，该库提供了声学中回声消除AEC算法，通过API接口将该库封装到我们系统的语音处理模块中，为了使用方便，本系统将Speex中音频处理相关的API统一提取出来，形成一个中间的API层封装在libaec.so中。本系统音频降噪流程如图19所示。

图19音频降噪流程图

流程中采用多线程的方式，一个线程用于音频读取和降噪，创建Speex回声消除状态器函数为speex_echo_state_init(m_nFrameSize, m_nFilterLen);创建Speex预处理状态前的函数为speex_preprocess_state_init(m_nFrameSize, m_nSampleRate)。音频读取是从音频采集模块的CaptureCacheBuffer获取的，每次读取帧大小为8ms，读取32ms 3路麦克风数据和1路参考信号各32ms数据，也就是4帧数据。读取数据成功后进行Speex帧预处理，函数为speex_echo_cancellation(m_pState, mic, ref, out);然后进行Speex回声消除，函数为

speex_preprocess_run(m_pPreprocessorState, (__int16 *)out);处理完的数据存入AECCacheBuffer，待唤醒模块读取。另一个线程也就是主线程等待数据处理线程完成，完成后销毁Speex回声消除器，函数为speex_echo_state_destroy(m_pState);销毁Speex预处理状态器speex_preprocess_state_destroy(m_pPreprocessorState);至此音频降噪模块实现完成。

4.4 唤醒模块设计与实现

语音唤醒算是语音识别领域里最基础的应用，简单来说就是在后台静默地运行着一个占用较少系统资源的语音识别组件服务，该组件一直处于监视麦克风输入的状态，如果有检测到特定的语音输入，即唤醒词，则激活与之绑定的某个程序“开关”。相当于一个简化版的语音助手，只对某一个特定的词汇进行响应，识别后也只完成某一件指定的任务。如果说同语音助手的交互是一段持续的交流，那么语音唤醒即可作为这种连续交流的入口。

本系统采用snowboy，一个开源的，轻量级语音唤醒引擎，来定制我们的系统唤醒词“小贾小贾”。snowboy唤醒引擎的主要特性如下：

高度可定制性。可自由创建和训练属于自己的唤醒词

始终倾听。可离线使用，无需联网，保护隐私。精确度高，低延迟

轻量可嵌入。耗费资源非常低（单核 700MHz 树莓派只占用 10% CPU）

开源跨平台。开放源代码，支持多种操作系统和硬件平台，可绑定多种编程语言

snowboy安装配置下面的使用方法，也是相对比较简单方法。

4.4.1 获取源代码并编译

1）安装依赖

安装pulseaudio软件以减少音频配置的步骤：sudo apt-get install pulseaudio，安装 sox 软件测试录音与播放功能：sudo apt-get install sox，安装完成后运行 sox -d -d 命令，对着麦克风说话，确认可以听到自己的声音。安装 PyAudio：sudo apt-get install python3-pyaudio。安装 SWIG（>3.0.10）：$ sudo apt-get install swig。安装 ATLAS：$ sudo apt-get install libatlas-base-dev。获取源代码：$ git clone https://github.com/Kitt-AI/snowboy.git。编译python3绑定：$ cd snowboy/swig/Python3 && make

2）唤醒测试

进入官方示例目录 snowboy/examples/Python3 并运行以下命令：

$ python3 demo.py resources/models/snowboy.umdl

命令中的 snowboy.umdl 文件即语音识别模型。修改 snowboy/examples/Python3 目录下的 snowboydecoder.py 文件后运行，然后对着麦克风清晰地讲出“snowboy”，如果可以听到“滴”的声音，则安装配置成功。命令行为snowboy test。

4.4.2 设置自己的唤醒词

将包含“小贾小贾”唤醒词的音频文件上传至 snowboy 官网，训练生成本系统需要的语音模型。需要上传3个wav格式的音频文件，可直接在线录制。训练完成并测试通过后，下载 PMDL 后缀的模型文件，该文件即为“小贾小贾”唤醒词的模型文件。

4.4.3 定制唤醒词测试

将上一步中下载好的 model.pmdl模型文件复制到自己的项目目录下。另外将 snowboy/swig/Python3目录下的 _snowboydetect.so库，snowboy/examples/Python3目录下的 demo.py，snowboydecoder.py，snowboydetect.py 文件和 resources目录均复制到项目目录下。

在项目目录下执行 $ python3 demo.py model.pmdl，唤醒程序已启动，喊出“小贾小贾”，程序被成功唤醒，唤醒测试如图20所示。

图20 snowboy唤醒测试图

4.4.4 定义唤醒响应

系统唤醒作为一个触发，直接影响到系统接下来的动作，因此按照需求，重新定义唤醒后的动作，唤醒模块流程如图21所示。

图21 唤醒模块流程图

唤醒模块分为两个线程任务和一个回调函数。主线程负责创建唤醒任务线程并检查唤醒结束标记，如果用户要关闭唤醒，主线程会结束掉唤醒任务线程，结束函数为detector.terminate()。唤醒任务线程负责唤醒回调注册，注册函数为snowboydecoder.HotwordDetector(interrupt_callback)；设置唤醒模型，设置函数为snowboydecoder.HotwordDetector(model)；唤醒阈值设置，设置函数为snowboydecoder.HotwordDetector(model, sensitivity);之后进入主循环，从AECCacheBuffer读取降噪后的音频，并调用snowboy唤醒词检测接口，接口为detector.start(aec_audio_buffer,sleep_time=0.03)。

5. 21_ZF1821334_张加杰_第5部分　　总字数：8462

相似文献列表

去除本人文献复制比：0.6%(49)　　文字复制比：0.6%(49)　　疑似剽窃观点：(0)

1	基于深度学习的语音识别方法研究	0.6%（49）
	王成(导师：李云红;种国华) - 《西安工程大学硕士论文》- 2018-05-28	是否引证：否

原文内容

唤醒词识别成功以后，程序响应的具体内容由回调函数定义，本系统唤醒的后续操作是创建线程并进行网络检测和语音识别。

唤醒定制完成后的测试结果如图22所示。

图22 程序唤醒响应图

4.5 语音识别模块设计与实现

本系统的核心模块是语音识别，语音识别作为信息技术领域非常重要的科技发展技术之一，近二十年来逐渐从实验室走向实际应用场景。语音识别技术可以将人类发出的声音通过一系列算法转化为计算机可以识别的输入，甚至是人类可以识别的文本信息。本系统支持两种形式的语音识别，一种是在线识别，目前市面上大多数产品采用这种形式，将识别模型和处理算法放到云端，以C/S架构的形式提供服务。另一种是离线识别，顾名思义，可以支持在没有网络的情况下进行语音转文字处理，识别模型就在本地系统上，作为在线识别的补充，使系统在网络异常情况下仍能正常工作。

4.5.1 离线识别

通过前期调研对比，选择Kaldi作为离线识别的解决方案。Kaldi是当前最流行的开源语音识别工具，它使用WFST来实现解码算法。Kaldi的主要代码是C++编写，在此之上使用bash和python脚本做了一些工具.语音识别，大体可分为“传统”识别方式与“端到端”识别方式，其主要差异就体现在声学模型上。Kaldi的整体模块结构如图23所示。

External LibrariesKaldi C++ Library(Shell) Scripts

Kaldi C++ Executables

Transforms

Feat

SGMM

GMM

Utils

Matrix

Decodable

LM

Tree

FST ext

HMM

Decoder

BLAS/LAPACK

OpenFST

图23 Kaldi模块结构图

图中ExternalLibraries模块中包含BLAS/LAPACK和OpenFST，OpenFST的主要作用是构造有限状态机，对于有限的状态进行构造遍历搜索优化等。在计算过程中常用的线性代数相关的计算封装在ATLAS库中，提供了C++的使用接口。IRSTLM是一个统计语言模型的工具包。sph2pipe是处理SPHERE_formatted数字音频文件的软件，它可以将LDC的sph格式的文件转换成其它格式。SRILM（SRILM - The SRI Language Modeling Toolkit）是由SRI International提出的一套工具集，主要用于创建和使用统计语言模型。传统方式的声学模型一般采用隐马尔可夫模型（HMM），而“端到端”方式一般采用深度神经网络（DNN），Kaldi主要用脚本来驱动，每个recipe下会有很多脚本。local目录下的脚本通常是与这个example相关，不能移植到别的例子，通常是数据处理等一次性的脚本。而util下的脚本是通用的一些工具。steps是训练的步骤，最重要的脚本。

Kaldi识别训练过程如图24所示。基础环境搭建部分主要搭建相关基础开发环境，使用使用docker搭建Ubuntu18.04的开发环境。docker启动参数为docker run -it -d -m 4G fengzhishang/kaldi_env:1 /bin/bash

图24 Kaldi识别训练过程如图

Kaldi环境搭建部分包含代码下载，依赖检查，依赖软件安装，外部依赖安装，Kaldi软件配置，编译等模块。因编译过程比较消耗内存，使用Makefile 4线程进行编译。

准备数据集部分，Kaldi中文语音识别公共数据集一共有4个，分别是aishell: AI SHELL公司开源178小时中文语音语料及基本训练脚本，位于Kaldi-master/egs/aishell。gale_mandarin: 中文新闻广播数据集(LDC2013S08,LDC2013S08）。hkust: 中文电话数据集(LDC2005S15, LDC2005T32)。thchs30: 清华大学30小时的数据集。本系统选择thchs30语料库。首先制作我们需要语料库，语料包含3个文件，具体内容如表4所示。

表4 thchs30数据表

包名	大小	内容
data_thchs30	6.4G	语音数据和训练脚本
test-noise	1.9G	标准无噪音测试数据
resource	24M	补充资源，包括培训数据词典、噪音样本

包名大小内容

data_thchs30 6.4G 语音数据和训练脚本

test-noise 1.9G 标准无噪音测试数据

resource 24M 补充资源，包括培训数据词典、噪音样本

将数据包解压到egs/thchs30/s5/thchs30-openslr下，另外包含训练好的语言模型word.3gram.lm和phone.3gram.lm以及相应的词典lexicon.txt。这个数据集包含以下内容如表5所示。

表5 thchs30数据集内容

数据集	音频时长(h)	句子数	词数
train(训练)	26	20000	386504
dev(开发)	2:15	1611	35486
test(测试)	7:16	4980	98170

数据集音频时长(h) 句子数词数

train(训练) 26 20000 386504
dev(开发) 2:15 1611 35486
test(测试) 7:16 4980 98170

训练集train包含的数据量最多，主要用于模型的训练。Dev开发集的作用是为了训练出更好，与train训练集进行交差验证。训练的结果分为音素和词，测试也根据音素和词进行目标数据分类。local/thchs-30_data_prep.sh主要工作是从thchs/data_thchs30（下载的数据）三部分分别生成word.txt（词序列），phone.txt（音素序列），text（与word.txt相同），wav.scp（语音），utt2pk（句子与说话人的映射），spk2utt（说话人与句子的映射）。

MFCC features是提取MFCC特征，分为两步，先通过steps/make_mfcc.sh提取MFCC特征，再通过steps/compute_cmvn_stats.sh计算倒谱均值和方差归一化。language stuff是构建一个包含训练和解码用到的词的词典。而语言模型已经由王东老师处理好了。基于词的语言模型包含48k基于三元词的词，从gigaword语料库中随机选择文本信息进行训练得到，训练文本包含772000个句子，总计1800万词，1.15亿汉字。基于音素的语言模型包含218个基于三元音的中文声调，从只有200万字的样本训练得到，之所以选择这么小的样本是因为在模型中尽可能少地保留语言信息，可以使得到的性能更直接地反映声学模型的质量。这两个语言模型都是由SRILM工具训练得到。

数据准备，monophone单音素训练， tri1三因素训练， trib2进行lda_mllt特征变换，trib3进行sat自然语言适应，trib4做quick，后面就是dnn。执行 run.sh 即可开始训练。

查看训练效果cat tri1/decode_test_word/scoring_kaldi/best_wer：

%WER 36.06 [29255 / 81139, 545 ins, 1059 del, 27651 sub] exp/tri1/decode_test_word/wer_10_0.0

执行语音识别，将声音文件复制到 online-data/audio/ 目录，然后运行 run.sh 执行识别操作，识别效果如下：

online-wav-gmm-decode-faster --verbose=1 --rt-min=0.8 --rt-max=0.85 --max-active=4000 --beam=12.0 --acoustic-scale=0.0769 --left-context=3 --right-context=3 scp:./work/input.scp online-data/models/tri1/final.mdl online-data/models/tri1/HCLG.fst online-data/models/tri1/words.txt 1:2:3:4:5 ark,t:./work/trans.txt ark,t:./work/ali.txt

File: A1_00 “灯光调亮”“灯光调暗”“打开电视”“关闭电视”“打开空调”“关闭空调”

由此验证，该模型可以作为本系统离线识别模型。

4.5.2 在线识别

在线识别选择接入百度AI开放平台，下载BDSSDKMessage语音识别SDK包，接入流程如图25所示。

图25 在线识别流程图

在线识别模块分为两个线程任务和一个回调函数。主线程负责创建识别任务线程并检查识别结束标记，如果识别结束，主线程会结束掉识别任务线程。识别任务主要调用BDSSDKMessage接口进行识别相关任务。

获取实例接口为BDSpeechSDK::get_instance(bds::SDK_TYPE_ASR, err_msg);每次识别一个音频流，都需要从获取实例到释放实例完整地执行一遍。即get_instance每个音频流获取一次，get_instance最多可以保持10个实例，即最多同时识别10个音频。BDSSDK接口统一使用命令字进行配置，由一个标明意向的name，及其它参数组成，然后通过post函数传递命令。支持的命令字如表6所示。

表6 语音在线识别BDSSDK命令字表

命令字	说明
ASR_CMD_CONFIG	设置配置参数
ASR_CMD_START	设置启动参数
ASR_CMD_PUSH_AUDIO	传递音频数据
ASR_CMD_STOP	停止当前当前音频流输入
ASR_CMD_CANCEL	取消当前的整个识别过程

命令字说明

ASR_CMD_CONFIG 设置配置参数

ASR_CMD_START 设置启动参数

ASR_CMD_PUSH_AUDIO 传递音频数据

ASR_CMD_STOP 停止当前当前音频流输入

ASR_CMD_CANCEL 取消当前的整个识别过程

设置回调监听器的任务是当SDK有返回结果时调用该函数，设置接口为sdk->set_event_listener(&asr_output_callback, (void*)& thread_seq);回调产生在SDK内部的线程中。SDK配置参数根据官网提供信息，本系统需要支持普通话和远场识别功能，配置函数为cfg_params.set_parameter(bds::ASR_CMD_CONFIG, sdk_log_level);选择配置如表7所示。

表7 语音识别输入参数配置表

PID	语言	模型	是否有标点	在线语义
1936	普通话	语音近场识别模型	有标点（逗号）	不支持

PID 语言模型是否有标点在线语义

1936 普通话语音近场识别模型有标点（逗号） 不支持

设置SDK 启动参数只需要填写ASR_PARAM_KEY_APP参数，填写自定义的应用名称”MyTest”，接口为start_params.set_parameter(bds::ASR_PARAM_KEY_APP, "MyTest")。传递音频数据模块的音频流是从AECCacheBuffer获取的，音频流的音频格式是pcm输入流，3声道，16bits，小端序，接口为push_params.set_parameter(bds::DATA_CHUNK, audio_buf, (int)read_cnt); SDK音频流已经输入完毕，不再有后续音频。 需要调用push_params.set_parameter(bds::DATA_CHUNK, audio_buf, 0)；push_params.set_parameter(bds::DATA_CHUNK, audio_buf, 0)。

识别时序如图26所示。

图26 在线语音识别时序图

取消识别时需要告诉SDK 本次识别取消，即用户不再需要识别结果，调用接口为sdk->post(cancel_params, bds::ASR_CMD_CANCEL));在设置的event_listener输出回调中，SDK返回EvoiceRecognitionClientWorkStatusCancel事件。

识别取消或识别完成需要主动释放SDK资源，接口为bds::BDSpeechSDK::release_instance(sdk);清理所有线程池，所有识别结束，不需要发起新的识别。SDK空闲时执行清理动作bds::BDSpeechSDK::do_cleanup()。

4.4 应用层设计与实现

4.4.1 指令识别

从流程上，语音识别结果出来后需要对识别结果进行解析，解析出用户所发出语音指令的含义。本系统设计支持对蓝牙开关，红外空调，智能电灯的控制，所有支持指令如表8所示。

表8 控制指令表

设备	支持指令
蓝牙开关	打开电视
	关闭电视
红外空调	打开空调
	关闭空调
智能电灯	打开电灯
	关闭电灯
	灯光调亮
	灯光调暗

设备支持指令

蓝牙开关打开电视

关闭电视

红外空调打开空调

关闭空调

智能电灯打开电灯

关闭电灯

灯光调亮

灯光调暗

由于蓝牙开关连接的是非智能电视，本系统将蓝牙开关的指令主体设置为电视。系统需要具有一定灵活性。因此需要将指令进行泛化，如用户的语音指令为“空调关掉”指令，系统应具有一定的识别和纠错能力，可以匹配到“关闭空调”指令的动作。控制指令泛化支持如表9所示。

表9 控制指令泛化支持表

标准指令	泛化指令
打开电视	电视打开
关闭电视	电视关闭，电视关掉，关掉电视
打开空调	空调打开
关闭空调	空调关闭，空调关掉，关掉空调
打开电灯	电灯打开，开灯
关闭电灯	电灯关闭，关灯
灯光调亮	调亮灯，灯调亮
灯光调暗	调暗灯，灯调暗

标准指令泛化指令

打开电视电视打开

关闭电视电视关闭，电视关掉，关掉电视

打开空调空调打开

关闭空调空调关闭，空调关掉，关掉空调

打开电灯电灯打开，开灯

关闭电灯电灯关闭，关灯

灯光调亮调亮灯，灯调亮

灯光调暗调暗灯，灯调暗

识别结果与指令的匹配中，本系统采用两次定向查找加一次不定向查找的方式进行匹配。第一次，在识别结果中遍历标准指令，如果匹配到，则执行指令动作，如果匹配失败，则重新在识别结果中遍历泛化指令，如果匹配到，则执行指令动作，如果匹配失败，则进行不定向指令词匹配。指令匹配流程如图27所示。

图27 指令匹配流程图

当标准指令匹配和泛化指令匹配均失败时，系统进行不定向指令匹配，不定向指令匹配是最小化主体和动作的匹配，如“我要打开客厅的灯”，系统根据关键字主体“灯”和关键字动作“开”进行模糊匹配，最小化关键字仍然是以数组的形式存储，控制指令最小化主体动作如表10所示。

表10 控制指令最小化主体动作表

标准指令	最小化主体	最小化动作
打开电视	电视	开
关闭电视	电视	关
打开空调	空调	开
关闭空调	空调	关
打开电灯	电灯	开

关闭电灯	电灯	关
灯光调亮	灯	亮
灯光调暗	灯	暗

标准指令最小化主体最小化动作
打开电视电视开
关闭电视电视关
打开空调空调开
关闭空调空调关
打开电灯电灯开
关闭电灯电灯关
灯光调亮灯亮
灯光调暗灯暗

4.4.2 指令执行

所有指令包括泛化指令和非定向指令匹配成功后最终返回的结果都对应到标准指令上，每个标准指令对一个控制指令，该控制指令用来执行指令动作，比如“打开灯”指令会向WiFi外设控制模块发送打开的指令。标准指令注册了一个指令动作处理函数，当标准指令匹配到后会执行回调函数，控制指令动作对照如表11所示。

表11 控制指令动作对照表

标准指令	外设类型	指令动作
打开电视	蓝牙	command_bt_tv_on
关闭电视	蓝牙	command_bt_tv_off
打开空调	红外	command_ir_airconditioner_on
关闭空调	红外	command_ir_airconditioner_off
打开电灯	WiFi	command_wifi_led_on
关闭电灯	WiFi	command_wifi_led_off
灯光调亮	WiFi	command_wifi_led_lighten
灯光调暗	WiFi	command_wifi_led_dark

标准指令外设类型指令动作
打开电视蓝牙 command_bt_tv_on
关闭电视蓝牙 command_bt_tv_off
打开空调红外 command_ir_airconditioner_on
关闭空调红外 command_ir_airconditioner_off
打开电灯 WiFi command_wifi_led_on
关闭电灯 WiFi command_wifi_led_off
灯光调亮 WiFi command_wifi_led_lighten
灯光调暗 WiFi command_wifi_led_dark

4.4.2 网络探测模块设计与实现

网络探测模块采用守护进程的方式，使用Linux shell脚本实现。将进程启动添加到/etc/init.d/initrc中使其在开机时启动。

判断网络畅通是通过curl来访问服务器地址，从而判断服务器网络状态是否畅通，服务器地址选取sina.com和ai.baidu.com。网络探测流程如图27所示。

网络初始化
开始
网络探活
curl ai.baidu.com
curl sina.com
延时1分钟
结果收集
标志文件写入

图27 网络探测流程图

系统开机后需要进行网络初始化，相关初始化脚本会拉起wpa_supplicant进程，该进程会按照/data/wpa_supplicant.conf配置文件进行WiFi连接，WiFi连接成功后每分钟进行网络探活，探活结果会写入/data/network_status文件，格式为wifi:on/off。

关键代码为local ret_code=`curl -I -s --connect-timeout 1 “www.ai.baidu.com -w | tail -n1`。

4.5 语音合成模块设计与实现

本系统语音合成模块使用百度AI开发平台提供的语音合成能力，简要步骤如下：

注册成为百度AI开放平台的开发者

要调用百度AI开放平台的语音合成能力先要成为百度AI开放平台的开发者，注册后新建一个百度语音合成应用。然后就能看到创建完的应用和 API KEY 以及 Secret KEY了。

2）领取免费额度

创建完应用后，可以到概览页领取语音合成的免费额度。

3）准备数据

语音合成是将文本转换为可以播放的音频文件的服务，本系统语音合成文本设计如表12所示。

表12 语音合成文本设计表

设备	支持指令
蓝牙开关	电视已打开
	电视已关闭
红外空调	空调已打开
	空调已关闭
智能电灯	电灯已打开
	电灯已关闭
	电灯已调亮
	电灯已调暗

设备支持指令
蓝牙开关电视已打开
电视已关闭
红外空调空调已打开
空调已关闭
智能电灯电灯已打开
电灯已关闭
电灯已调亮
电灯已调暗
4）编写程序
安装语音合成 C++ SDK，最低支持 C++11+，使用开发包简要步骤如下
1.在官方网站下载识别、合成 RESTful API C++ SDK压缩包。
2.将下载的aip-cpp-sdk-version.zip解压，其中文件为包含实现代码的头文件。
3.安装依赖库libcurl（需要支持https） openssl jsoncpp(>1.6.2版本，0.x版本将不被支持)。
4.编译工程时添加 C++11 支持 (gcc/clang 添加编译参数 -std=c++11)，添加第三方库链接参数 lcurl，lcrypto，ljsoncpp。
5.在源码中include speech.h ，引入压缩包中的头文件以使用aip命名空间下的类和方法。
编写语音合成应用程序，在线语音合成程序流程如图29所示。
图29 在线语音合成程序流程图
实例化client是语音合成的的C++客户端，申请APPID/AK/SK后，使用接口aip::Speech client(app_id, api_key, secret_key)新建TTS client，常量APP_ID在百度云控制台中创建，接口中需要两个参数api_key和secret_key，这两个参数是开放平台创建语音合成应用后自动分配的字符串，用于授权和校验。参数设置部分可以设置音频的语速，音调，音量，发音人等。在线语音合成参数对照如表13所示。
表13 在线语音合成参数对照表

参数	类型	描述	是否必须
text	字符串	待合成的文本数据，字段长度需小于512字节	是
cuid	字符串	实例号，是用来区分用户的，系统内唯一，一般使用mac地址，字符串长度小于64字节	否
speed	字符串	语音发音速度，一般选择6作为正常中速	否
pitch	字符串	语音音调，一般选择6作为正常中文语调	否
volum	字符串	发音音量，一般选择6作为中等音量	否
persion	字符串	系统发音人类型，分为0女声，1男声。一般选择0为默认的普通女声	否

参数类型描述是否必须
text 字符串待合成的文本数据，字段长度需小于512字节是
cuid 字符串实例号，是用来区分用户的，系统内唯一，一般使用mac地址，字符串长度小于64字节否
speed 字符串语音发音速度，一般选择6作为正常中速否
pitch 字符串语音音调，一般选择6作为正常中文语调否
volum 字符串发音音量，一般选择6作为中等音量否
persion 字符串系统发音人类型，分为0女声，1男声。一般选择0为默认的普通女声否
合成文本长度必须小于1024字节，如果本文长度较长，可以采用多次请求的方式。
把系统所需文字合成为语音文件，成功返回后将数据写入文件进行保存。
返回结果分析模块根据错误码进行分析，如果失败则重新执行，最多尝试3次，语音合成错误码及含义如表14所示。
表14 在线语音合成错误码表

错误码	含义
301	不能识别的输入参数
302	输入的参数验证错误
303	合法性验证失败
304	音频合成过程中失败

错误码含义

301 不能识别的输入参数
302 输入的参数验证错误
303 合法性验证失败
304 音频合成过程中失败
4.6 关键技术和解决方案
4.7 本章小结
本章对整个系统的各个层次和功能模块进行了详细的设计和实现，对控制系统底层设计进行了开发板的选型和环境搭建。对控制层的蓝牙，红外，WiFi外设进行了设计和实现。音频处理模块详细设计了音频采集和音频AEC降噪。唤醒部分采用snowboy进行定制并完整应用到系统的唤醒模块中。语音识别模块离线部分采用了kaldi开源解决方案，并且集成了百度AI开发平台的语音识别功能。应用层对指令识别和指令执行进行了比较详细的设计，并且设计了网络探测模块。最后实现了在线语音合成功能的接入。本章是整个系统设计和实现的核心部分，整个系统的实现也是依托本章内容。

指　　标
疑似剽窃文字表述

1. SPHERE_formatted数字音频文件的软件，它可以将LDC的sph格式的文件转换成其它格式

6. 21_ZF1821334_张加杰_第6部分　　总字数：6191

相似文献列表

去除本人文献复制比：0.5%(29)　　文字复制比：0.5%(29)　　疑似剽窃观点：(0)

1	一种基于iSCSI的网络RAID设计与实现	0.5%（29）
	刘卫平;蔡皖东; - 《计算机工程与应用》- 2006-05-11	是否引证：否

原文内容

第五章系统测试分析
5.1 测试概述
本系统的测试是通过将基于语音的家居控制系统组装完整后，进行一些列的功能和性能的测试，目的是为了检验系统的设计成果与需求设计是否一致，进而对系统进行修正，起到一个促进作用。将麦克风，主板，扬声器及相关模组进行整合，烧录制作好的软件，结合成一个整体后，首先进行功能相关的测试，对比需求方案找出所开发过程中系统设计与需求不符合的地方或者矛盾冲突的地方，从而继续完善系统的设计方案。系统测试作为开发的一部分，是为保证基于语音的家居控制系统的正常运行，并在实际操作和运行环境中对基于语音的家居控制系统的严格而有效的测试。而进行系统测试的最终目的也是是为了我们设计的软件系统能够满足需求设计。当前系统测试的主要内容包括：
1）系统功能测试。即测试当前软硬件系统的功能是否按照需求设计正常执行，根据第二章需求设计文档对需求中的相关功能进行一一测试验证。系统本身就是要按需求进行设计的，是软件最重要的质量因素，因此功能测试必不可少。
2）系统稳健性测试。即测试我们的基于语音的家居控制系统能否在异常环境情况下正常运行，软件系统在异常情况下正常运行的能力。健壮性有两个含义：一是容错能力，二是恢复能力。系统测试分为功能测试，对于我们的基于语音的家居控制系统，软件部分除了能实现基本的语音识别和家居控制外未必能够满足相关的性能要求，在每一步的测试中都有对性能相关的点进行考量和测试，以最终达到系统的稳定可靠运行。
5.2 测试工具及测试环境
测试工具：Linux下录音工具arecord，程序异常定位工具gdb等gnu开发套件，网络性能测试工具netperf，unix系统性能工具unixbench，噪音检测手机APP，WiFi路由器，蓝牙音响，个人电脑等。
测试环境：唤醒识别测试会从如下几个场景进行测试评估，如表15所示。
表15 唤醒识别测试场景

	噪音类型
一米	安静办公噪音音乐噪音扫地机器人噪音电视噪音
三米	安静办公噪音音乐噪音扫地机器人噪音电视噪音
五米	安静办公噪音音乐噪音扫地机器人噪音电视噪音

噪音类型
一米安静办公噪音音乐噪音扫地机器人噪音电视噪音
三米安静办公噪音音乐噪音扫地机器人噪音电视噪音
五米安静办公噪音音乐噪音扫地机器人噪音电视噪音
由于我们主要的应用场景为家居控制，因此需要基本的家庭环境和一些测试用的必须家居设备，如家用红外空调，蓝牙开关，家用WiFi冰箱等。
需要说明的是，对于我们的系统测试场景，需要有噪音检测工具，精度较高的工具价格相对较高，因此采取比较折中的办法，使用低精度的手机噪音检测APP，这是一款噪音分贝检测应用，通过对环境中的声音进行计算得出具体的实时噪音变化分贝，以提供我们噪音场景下的测试。
本系统对计算和网络能力要求较高，需要有一定的评估手段，因此借用开源工具Unixbench对系统运行环境进行计算方面的

测试，unixbench是一个在类unix系统上运行的系统性能方面的工具，通过一系列的计算生成一个benchmark分数，通过分数可以衡量系统在计算方面的表现。该测试可以对基于语音的家居控制系统提供一个基本的性能指标，这些指标包括如下几点。

系统单任务执行方面的性能指标

系统在多任务交差运行时的性能指标

系统在处理并行任务时的性能指标

本系统的另一个性能关键指标是网络性能，我们借助Netperf工具来进行测试。这个工具分为server端和client端，可以对网络中丢包错报重试等影响网络的因素进行分析和统计，另外它是基于TCP协议和UDP协议进行的传输可以对协议层进行测试。server端作为被测试方，部署在基于语音的家居控制系统上，使用个人电脑作为client端，通过client端发起请求对server所在的家居控制系统进行网络性能测试。

5.3 测试方法及流程

本系统的测试主要包括基本的功能测试和性能测试，还要对可靠性安全性等方面进行全面的测试，有些测试过程可同时包含几个方面，但大部分需要单独设计测试用例。

1）基本的系统功能测试：测试本系统各个子系统和子系统下的功能模块是否能够执行基本的业务逻辑，如唤醒是否正常，识别结果是否正确，指令解析是否配对，外设控制是否生效，外设结果返回是否正确等。

测试方法

用该系统的路由接口进行录音（安静和餐厅噪音的1、3、5米）

将录制的音频通过工具灌入进行识别测试

根据识别结果利用wer工具计算字准和句准

测试流程

音频播放测试也就是播放一条唤醒词之后播放识别query,使用人声校准音调节调节播放query高保真音量为70dbc。噪音源为电视剧“人民的名义”、音乐“下雨天”、扫地机器人、办公室噪音。将噪音源放在距待测设备2M150度并调节音量使得在待测设备出测得噪音音量为60dbc。记录音箱识别结果计算字准句准。播测用query：按远场标准对内部识别100句。WER（Word Error Rate）是字错误率，是一个衡量语音识别系统的准确程度的度量。其计算公式是WER=(I+D+S)/NWER=(I+D+S)/N，其中I代表被插入的单词个数，D代表被删除的单词个数，S代表被替换的单词个数。也就是说把识别出来的结果中，多认的少认的，认错的全都加起来，除以总单词数。

2）系统的性能测试：测试本系统整体的性能，主要包括几个方面，计算性能，网络性能，根据测试数据进行评估，系统的计算及网络的能力是否满足顺畅可用，可支持全部功能的系统的依赖等。

3）系统的可靠性测试：根据当前系统硬件和软件方面的结构和能力，针对基于语音的家居控制系统的压力测试，首先对各个组件包括网络探活，录音，唤醒，离线语音识别，在线语音识别，外设控制，语音合成等模块进行单元化的压力测试，可通过自动化脚本进行辅助以节省人力。集中压力测试，对基于语音的家居控制系统进行整体的压力测试，可通过录制的音频进行播放测试，提前录制好唤醒词加语音控制的音频，在本阶段通过蓝牙音箱进行播放，对系统进行整体的压力测试。真实环境测试环节，在播测等压力测试进行过后，要进行真实的人力环境测试，人声唤醒加语音控制，控制端也由log辅助判断结果改为真实家居设备。随机破坏测试环节，可适当对网络限速，遮挡设备的红外发射模块等。

4）系统的安全性测试：

系统的安全性方面，借助OpenVAS漏洞扫描工具进行扫描,因本系统采用了Linux系统，基本的安全补丁已经比较完善，针对我们开发的功能进行针对性的模拟攻击,比如ssh，telnet等方式登录系统，获取录音文件，获取识别的文字结果，获取指令解析字典等。本文将针对上述的测试方式制定和设计测试用例，更换测试场景，并进行有效的性能和压力测试等手段来进行系统测试工作，以发现更多的系统问题促使系统更加安全完整。

5.4 系统功能测试

针对基于语音的家居控制系统，由于测试场景比较多样，我们先固定下一个最常用的场景，硬件方面需要准备：

支持红外控制的家居空调

支持蓝牙控制的智能插座

支持WiFi控制的智能灯

软件方面对相关的唤醒，识别等参数进行预设，根据该环境对系统进行全面测试，本系统测试用软件环境如表16所示。

表16 系统软件环境

voice_hi_pid	hi_211
voice_hi_package	com.hi.intell
voice_hi_server_url	https://hi.baidu.com/intell
debug_hi_switch	1
sdk_hi_version	intell_speech
clicklog_hi_version	2.0.5
event_hi_server_url	https://hi.baidu.com:443/saiya/ws ″
outputlog_hi_console	0
clicklog_hi_url	https://hi.baidu.com/saiya/log?
asr_hi_two_in_one	1
ais_hi_switch	1
wakeup_hi_dnn	0
voice_hi_debug_log_on	0
voice_hi_speech_sdk_version	3.0.6.15c6ba8-20170714-181721
asr_hi_confirm_request_url	https://hi.baidu.com/ws
request_hi_login_token_url	https://hi.baidu.com:443/saiya/device/token

voice_hi_pid hi_211

voice_hi_package com.hi.intell
voice_hi_server_url https://hi.baidu.com/intell
debug_hi_switch 1
sdk_hi_version intell_speech
clicklog_hi_version2.0.5
event_hi_server_url https://hi.baidu.com:443/saiya/ws ″
outputlog_hi_console 0
clicklog_hi_url https://hi.baidu.com/saiya/log?
asr_hi_two_in_one 1
ais_hi_switch 1
wakeup_hi_dnn 0
voice_hi_debug_log_on 0
voice_hi_speech_sdk_version3.0.6.15c6ba8-20170714-181721
asr_hi_confirm_request_url https://hi.baidu.com/ws
request_hi_login_token_url https://hi.baidu.com:443/saiya/device/token
语音识别测试安静场景数据如表17所示。
表17 语音识别播放测试数据-安静场景

	安静			
	字准	句准	asr_err	asr_err个数
1m60	98.37%	92.00%	0.00%	0
1m90	98.72%	93.00%	0.00%	0
3m	96.97%	87.00%	0.00%	0
5m	95.92%	86.00%	1.00%	1

安静
字准句准 asr_err asr_err个数
1m60 98.37% 92.00% 0.00% 0
1m90 98.72% 93.00% 0.00% 0
3m 96.97% 87.00% 0.00% 0
5m 95.92% 86.00% 1.00% 1
语音识别测试办公噪音场景数据如表18所示。
表18 语音识别播放测试数据-办公噪音场景

	办公噪音			
	字准	句准	asr_err	asr_err个数
1m60	97.67%	87.00%	0.00%	0
1m90	98.02%	92.00%	0.00%	0
3m	87.87%	66.00%	0.00%	0
5m	65.93%	42.00%	2.00%	2

办公噪音
字准句准 asr_err asr_err个数
1m60 97.67% 87.00% 0.00% 0
1m90 98.02% 92.00% 0.00% 0
3m 87.87% 66.00% 0.00% 0
5m 65.93% 42.00% 2.00% 2
语音识别测试音乐噪音场景数据如表19所示。
表19 语音识别播放测试数据-音乐噪音场景

	音乐噪音			
	字准	句准	asr_err	asr_err个数
1m60	95.45%	86.00%	0.00%	0
1m90	97.55%	89.00%	0.00%	0
3m	92.18%	76.00%	1.00%	1
5m	69.20%	42.00%	1.00%	1

音乐噪音
字准句准 asr_err asr_err个数
1m60 95.45% 86.00% 0.00% 0
1m90 97.55% 89.00% 0.00% 0
3m 92.18% 76.00% 1.00% 1
5m 69.20% 42.00% 1.00% 1
语音识别测试扫地机器人噪音场景数据如表20所示。
表20 语音识别播放测试数据-扫地机器人噪音场景

	扫地机器人噪音			
	字准	句准	asr_err	asr_err个数
1m60	96.03%	86.00%	0.00%	0

1m90	98.02%	88.00%	0.00%	0
3m	78.88%	51.00%	1.00%	1
5m	69.78%	49.00%	5.00%	5

扫地机器人噪音

字准句准 asr_err asr_err个数

1m60 96.03% 86.00% 0.00% 0

1m90 98.02% 88.00% 0.00% 0

3m 78.88% 51.00% 1.00% 1

5m 69.78% 49.00% 5.00% 5

语音识别测试电视剧噪音场景数据如表21所示。

表21 语音识别播放测试数据-电视剧噪音场景

	电视剧噪音			
	字准	句准	asr_err	asr_err个数
1m60	80.86%	53.00%	0.00%	0
1m90	68.03%	56.00%	0.00%	0
3m	57.06%	37.00%	0.00%	0
5m	29.41%	29.00%	0.00%	0

电视剧噪音

字准句准 asr_err asr_err个数

1m60 80.86% 53.00% 0.00% 0

1m90 68.03% 56.00% 0.00% 0

3m 57.06% 37.00% 0.00% 0

5m 29.41% 29.00% 0.00% 0

由以上数据可知，语音识别数据满足系统需求，在噪音场景下语音识别字准和句准都有下降，在5m的场景下最低达到29.41%和29.00%需要对该场景继续优化。

性能测试

在本系统上执行Unixbench，对Unixbench数据进行统计，总共执行三次取平均值，收集到的数据如表22所示。

表22 性能测试-Unixbench数据统计表

序号	score	1x_score	Double	FileCp_1024	FileCp_256	PipeCtxSw	SysCall
1	2998.4	1317.1	2242.0	1962.3	1213.7	2219.5	3500.9
2	3270.3	1525.5	2795.9	2164.9	1371.6	2652.3	3672.9
3	3356.2	1578.8	2805.2	2254.0	1442.0	2681.4	3650.6
平均值	3208.3	1473.8	2614.3	2127.0	1342.4	2517.7	3608.1

序号 score 1x_score Double FileCp_1024 FileCp_256 PipeCtxSw SysCall

1 2998.4 1317.1 2242.0 1962.3 1213.7 2219.5 3500.9

2 3270.3 1525.5 2795.9 2164.9 1371.6 2652.3 3672.9

3 3356.2 1578.8 2805.2 2254.0 1442.0 2681.4 3650.6

平均值 3208.3 1473.8 2614.3 2127.0 1342.4 2517.7 3608.1

Score是整体跑分评估，这是一个比较综合的指标，直接标识该系统的整体计算性能。由平均值3208.3可知系统整体跑分较高满足系统的功能计算需求。Double数值表示系统在浮点运算方面的性能。这两个指标是系统在内存，计算，编译优化等方面的能力，硬件的设计和软件的架构都会对这两个指标产生影响。FileCp_1024 FileCp_256这两个指标是指在文件复制拷贝过程中对于缓存和硬盘存储方面的读写速度的衡量，从平均值2127.0和1342.4来看，文件处理速度较好，满足系统的执行需求。PipeCtxSw这个指标是进程间通信中数据交换和执行的效率。SysCall这个指标是进程执行系统调用进入内核态的效率，系统调用从用户空间到内核空间执行相关的操作后重新返回用户空间，中间有大量的数据和标志进行了更新和交换。

综上，该系统整体性能较高，可满足基本数据和计算需求，多次取平均值后可见，系统性能比较稳定，在均值上下小幅波动，该项满足需求。

Netperf测试数据统计结果如表23所示。

表23 Netperf数据统计表

序号	tcp_crr	tcp_band	tcp_rr	udp_band	udp_stream	rtt_avg
1	99622.52	1430.41	483787.06	1478.17	798367.5	0.1
2	120313.95	1448.31	607314.38	1477.33	798171.06	0.1
3	120880.99	1447.9	599643.75	1477.53	798004.38	0.09
平均值	113605.65	1445.82	578234.23	1477.64	798201.44	0.096

序号 tcp_crr tcp_band tcp_rr udp_band udp_stream rtt_avg

1 99622.52 1430.41 483787.06 1478.17 798367.5 0.1

2 120313.95 1448.31 607314.38 1477.33 798171.06 0.1

3 120880.99 1447.9 599643.75 1477.53 798004.38 0.09

平均值 113605.65 1445.82 578234.23 1477.64 798201.44 0.096

tcp_crr表示在一次TCP链接中只进行一组Request/Response 通信即断开，并不断新建TCP链接时的响应效率。tcp_crr在Web服务器访问中较为普遍。tcp_band表示TCP 进行批量数据传输时的数据传输吞吐量。tcp_rr表示在同一次 TCP 长链接中进行多次 Request/Response 通信时的响应效率。tcp_rr在数据库访问链接中较为普遍。udp_band表示UDP进行批量数据传输时的

数据传输吞吐量。udp_stream表示UDP进行数据传输时整体数据流速。rtt_avg网络ping时延。

综上，系统的网络性能较好，由于受网络本身波动影响，测试过程中会有一定的数据波动，但整体波动不大，该系统的网络性能符合需求。

5.5 系统运行效果评估

本系统从各个方面的测试情况来看，整体实现度和表现都比较高，其中功能测试运行稳定，各功能模块运行符合需求和预期。性能方面从测试结果和人为体验上效果均达预期。

5.6 本章小结

本章对系统测试进行了整体概述，对测试工具及测试环境进行了研究和部署，进一步的对系统的测试方法和流程进行了设计，在系统功能测试部分，对系统的功能进行了详细的测试，并对系统性能进行了测试和分析。通过对测试结果分析，可以得到系统的整体运行情况，并且可以满足用户的需求。

总结与展望

总结

展望

参考文献

致谢

图清单

图1 流程业务图

图2 示意图

表清单

参考文献

[1] 鲍尚东,王杰.基于S3C2410X的嵌入式Linux系统的构建[J].铜陵学院电气工程学院,安徽铜陵,2014.

[2] 田阳，王泽宇.无线终端设备低功耗唤醒方法的研究范[J].中文信息学报，2018，16(5)：51-66.

[3] 李文凤，徐及，张鹏远.基于状态后验概率的语音唤醒识别系统[J].计算机工程，2017，31(10)：177-179.

[4] 张庆芳.基于语音识别的灯光控制系统[J].计算机应用,2014，24(1):14-16

天大学出版社,2015.

[6] 冼进,许振山,刘峥嵘等.基于DTW和HMM算法的语音识别系统对比研究[M]. 北京:电子工业出版社,2017.

[7]周立功.基于汉语语素对RNN语音识别系统的改进与研究[J].淮海工学院学报,2012,21(3):16-19.

[8]易克初，田斌，付强.基于DNN-HMM模型的语音识别的语音导航系统[M]. 北京：国防工业出版社，2012.

[9]蔡莲红，黄德智，蔡锐.一种GPU及深度置信网络的语音识别加速算法研究[M].北京：清华大学出版社，2013.

[10]赵力.连续音素的改进深信度网络的识别算法[M].北京：机械工业出版社，2019.

[11]胡光锐.基于ARM的嵌入式语音识别系统的研究与设计[M].上海：上海科学技术文献出版社，2004.

[12]何英，何强.一种基于层次结构深度信念网络的音素识别方法[M].北京：清华大学出版社，2012.

[13] 改进的深度置信网络分类算法研究 Lawrence Rabiner， Biing-Hwang Juang.Fundamentals of Speech Recognition[M].Beijing：Tsinghua University Press，2013.

[14]杨行峻，迟惠生.基于TTS技术的外挂式语音报警软件开发与应用[M].北京：电子工业出版社，2015.

[15]李虎生.利用TTS技术开发放射科语音播放系统[D]. 北京：清华大学，2012.

[16]俞栋，邓力，俞凯.减少驾驶人语音唤醒指令词误触发的方法及装置[M].计算机工程,2014,30(9)：94-96.

[17]王志国.基于MQTT的物联网系统文件传输方法的实现究[D].中国科学技术学,2018,31(4):524-529

[18]田国会,许亚雄.基于飞思卡尔MCU的AEC算法实现述[J].山东大学学报（工学版），2018，44(6)：47-54.

[19]薛胜尧.基于ROS 的智能语音交互系统设计与实现[J]. 电子设计工程，2015 (4)：78-81

[20]Myers C, Rabiner L, Rosenberg A. Voice Control Smart Home System [J]. IEEE Transactions on Acoustics, Speech, and Signal Processing, 2018, 28(6): 623-635.

[21] Rabiner L R.Control Method for Smart Home System[J]. Proceedings of the IEEE, 2016, 77(2): 257-286.

[22] Mohamed A, Dahl G, Hinton G.Development and analysis of Punjabi ASR system for mobile phones under different acoustic models. [J]. IEEE Transactions on audio, speech, and language processing, 2012, 20(1): 30-42.

amlc@cnki.net

http://check.cnki.net/