**中图分类号：TP3**

**论文编号：10006ZF1821334**



专业硕士学位论文

**智能家居语音控制系统**

**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 Intelligent Home Voice Control System**

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

智能家居语音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

随着移动互联网的发展进入新的方向，移动互联网中的智能化已经成为新的发展趋势和主要需求，在智能化引领发展的阶段中，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，确定了芯片和系统的选型，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C++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，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++ 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. Kaldi's main code is written in C++ , 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**

[第一章 绪论 11](#_Toc21205)

[1.1 研究背景及意义 11](#_Toc10865)

[1.1.1 研究背景 11](#_Toc15636)

[1.1.2 研究意义 12](#_Toc19000)

[1.2 国内外相关研究现状及对比分析 12](#_Toc24399)

[1.2.1 国外研究现状 12](#_Toc1907)

[1.2.2 国内研究现状 12](#_Toc16893)

[1.2.3 对比分析 13](#_Toc32237)

[1.3 研究目标及研究内容 13](#_Toc14887)

[1.3.1 研究目标 13](#_Toc23884)

[1.3.2 研究内容 13](#_Toc13456)

[1.4 本文组织结构 14](#_Toc21530)

[1.5 本章小结 15](#_Toc3880)

[第二章 系统需求分析 16](#_Toc13291)

[2.1 现状分析 16](#_Toc6925)

[2.2 功能性需求分析 16](#_Toc8148)

[2.3 非功能性需求分析 20](#_Toc29610)

[2.4 本章小结 21](#_Toc24470)

[第三章 系统总体设计 21](#_Toc31667)

[3.1 系统设计原则 21](#_Toc1323)

[3.2 系统架构设计 22](#_Toc5898)

[3.3 系统功能结构设计 22](#_Toc16268)

[3.4 系统网络拓扑 23](#_Toc12502)

[3.5 系统数据库设计 24](#_Toc19700)

[3.5.1数据模型结构 24](#_Toc22359)

[3.5.2 关键数据库表结构设计 24](#_Toc20457)

[3.6 接口设计 24](#_Toc17202)

[3.7 系统采用的关键技术、难点 25](#_Toc9676)

[3.8 本章小结 26](#_Toc3662)

[第四章 系统详细设计与实现 26](#_Toc4809)

[4.1 控制系统的底层设计与实现 26](#_Toc15999)

[1.5.1 芯片开发板选型 27](#_Toc30297)

[1.5.2 系统 31](#_Toc10889)

[1.5.3 外设和驱动 32](#_Toc5296)

[1.5.4 网络 32](#_Toc13382)

[1.5.5 控制协议 32](#_Toc9427)

[4.1 音频处理模块设计与实现 40](#_Toc26867)

[1.5.6 降噪 40](#_Toc4138)

[1.5.7 增强 40](#_Toc24818)

[4.2 唤醒模块设计与实现 40](#_Toc30659)

[1、获取源代码并编译 41](#_Toc17023)

[2、设置自己的唤醒词 42](#_Toc29870)

[3.测试 43](#_Toc10026)

[4.自定义响应 44](#_Toc29707)

[4.2 语音识别模块设计与实现 46](#_Toc25114)

[4.3网络检测模块设计与实现 48](#_Toc21704)

[4.4 关键技术和解决方案 48](#_Toc21854)

[4.5 本章小结 48](#_Toc23591)

[第五章 系统测试分析 48](#_Toc14046)

[5.1 测试概述 48](#_Toc32632)

[5.2 测试工具及测试环境 49](#_Toc27040)

[5.3 测试方法及流程 49](#_Toc13490)

[5.4 系统功能测试 49](#_Toc30344)

[5.5 系统运行效果评估 52](#_Toc18645)

[5.6 本章小结 52](#_Toc18358)

[总结与展望 52](#_Toc29228)

[图清单 52](#_Toc9260)

[表清单 53](#_Toc17909)

[1.6 研究内容 53](#_Toc5485)

[1.6.1 嵌入式芯片和系统的选型 53](#_Toc13306)

[1.6.2 语音唤醒功能 53](#_Toc28455)

[1.6.3 语音识别 53](#_Toc17)

[1.6.4 控制系统 53](#_Toc12592)

[2. 参考文献 54](#_Toc12462)

# 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TTS（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

当前，信息技术领域正在进入新的一轮发展周期——移动互联网蓬勃发展。通过移动互联网，人们可以利用各种移动终端设备随时随地的访问网络。在桌面互联网时代，键盘、鼠标和显示器很好的解决了人机交互问题。但是伴随着移动互联网的快速发展，社会的信息化、智能化也得到了快速的发展，原有信息获取与沟通方式已经不能满足人们的需求了，语音作为人类最自然、最便捷的信息沟通方式，逐渐进入人们的视线并引起了业界的广泛关注。随着深度学习技术的突破，大数据技术以及自然语言理解能力的提升，带动了一波产业热潮。国内继科大讯飞、捷通华声之后，阿里，百度，腾讯等巨头都在智能语音领域发力。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查询和播放音乐，和智能音箱聊天对话来获取了解天气情况、设置日程提醒、预订机票酒店等实用的生活服务。随着智能家居产业的蓬勃发展，照明、安防、工控等主要制造商已经推出多款智能家居产品，通过无线传感技术来增加使用舒适度。但因消费端需求逐渐成熟，通过智能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，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周边内容，与算法和深度学习关系密切但没有深入的设计和实现过程，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

现在国外智能语音音箱已经被广泛的推广，Amazon的echo，谷歌的GoogleHome，苹果的HomePod等纷纷上市抢占市场。2014年亚马逊的智能音箱Echo让人眼前一亮，它最大的特点是嵌入名为 Alexa语音交互大脑，它赋予了音箱语音交互的能力，让音箱能够像你的朋友一样与你交流，更重要的是，不管是购物或是信息查询等所有功能都是由语音交互来完成。可以说，Echo就是一位家庭私人助理，Echo的成功又一次带动了语音交互的发展。随后造型各异的语音交互机器人陆续推向市场，从手机端到智能音箱这种真实垂直场景的过度，需要解决的是远场拾音与固定场景语义理解的问题， 而亚马逊首先突破了这个垂直场景语义理解瓶颈，使语音交互机器人面向真实客户群。

### 国内研究现状

当前国内各厂家的音箱普遍具有语音控制功能，尤其是小米的家居生态做的比较好，而家居生态中均以在线识别为核心。2015年，京东与科大讯飞联手推出在功能上类似Echo的智能语音音箱“叮咚”。而2016年下半年开始，语音交互市场的突然爆发，几乎每隔一个多月，各类语音交互产品的用户体验效果都会出现较大的提升，而推动其发展归根到底还是依赖于语音交互技术链条的成熟。国内科大讯飞和声智科技在随后补齐了语音信号前端处理这个空缺。不管是手机还是音箱，毫无疑问产品的落地到运作带来大量真实场景下的语音数据；面对如此规模的数据量，云端能力再次彰显出来，其存储能力足以保存终端产品在日常使用过程中所产生的数据；既然云端已经拥有海量数据，剩下便是如何处理数据，而深度学习算法与硬件计算能力相结合在这方面发挥了惊人的威力，在大量数据加持下不断训练进化模型，语音识别和自然语音理解方面不断取得突破性进展，如今讯飞、搜狗、百度等语音识别准确率高达97%。

### 对比分析

全球范围内智能音箱销量都在持续增长，据报告显示，智能音箱全球市场规模突破3000万。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。

在大数据时代下，国外的微软、谷歌、亚马逊，国内的科大讯飞、百度、阿里巴巴等一批实力强劲的公司已经深入部署并提供一系列语音解决方案，如亚马逊的语音服Alexa，百度的语音交互系统 DuerOS，科大讯飞的AIUI等。

## 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：

* 嵌入式芯片和控制系统的选型和可行性分析
* 关键词唤醒功能设计实现
* 自动语音识别ASR的设计实现
* 控制系统的设计实现
* 语音合成TTS功能设计实现

### 研究内容

该智能语音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）嵌入式芯片和系统的选型

依据市面已有的智能音箱进行芯片对比，确定芯片型号，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喇叭，而喇叭放出的声音会被麦克风采集到，回采（硬件或者软件实现）到的喇叭信号对最终的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query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。该模块同样唤醒模块一样，需要研究声学算法并进行语音识别模型搭建。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去混响(Dereverberation)WPE降噪等算法，处理后的音频送与ASR模型进行训练。

4）控制系统

ASR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MESH组网，红外控制等，需要在芯片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TTS合成打开成功音频并播放，该模块可采用默认音频+简易TTS合成引擎结合实现。

## 本文组织结构

第一章是绪论。本章阐述了智能家居语音控制系统的研究背景和意义，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。本章主要阐述了智能家居语音控制系统的相关需求，将系统需求划分为功能性需求和非功能性需求，功能性需求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总结，并对后续的系统和工作做了明确计划。

## 本章小结

本章首先介绍了智能家居语音控制系统研究的背景和意义，又对国内外相关研究现状进行了对比分析，最后对研究目标及研究内容进行阐述说明，本章对后续研究及工作做了铺垫。

# 第二章 系统需求分析

## 2.1 现状分析

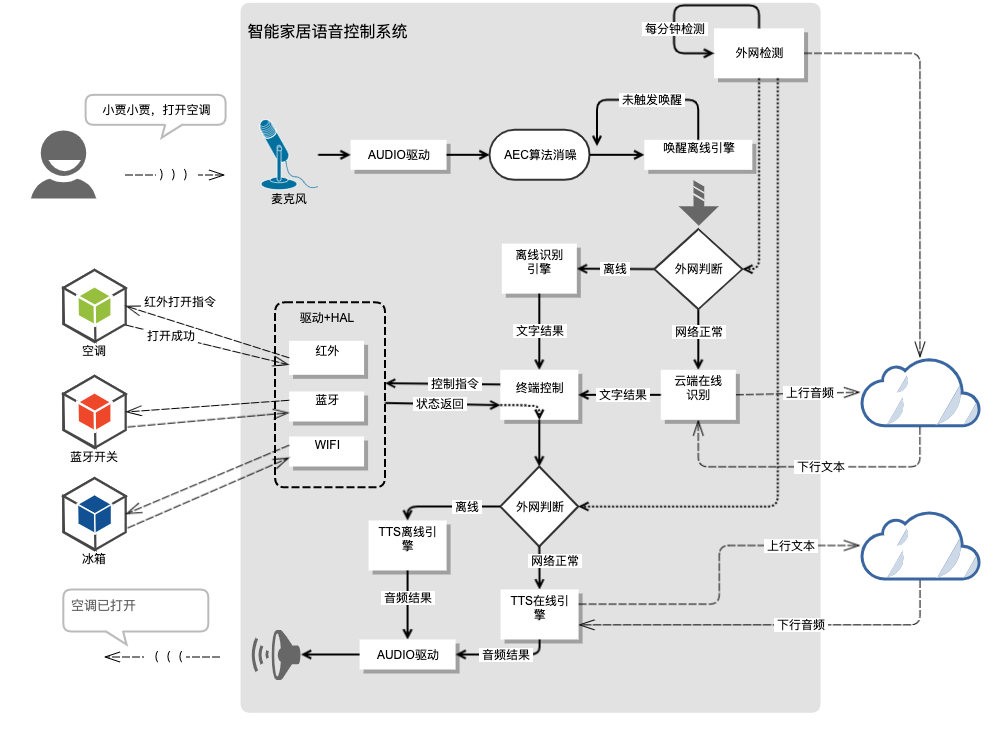
目前市面上的音箱具有重云轻端的特点，往往丢失了嵌入式系统本身灵活和控制的特点。目前家居虽然向着智能的方向发展，支持移动互联网的远程控制。但仍有大量的家居设备是按键控制，好一点的是红外或蓝牙，但相对功能单一，无法通过一个智能的语音中控进行统一控制。另一方面，智能语音的代表智能音箱，都是在有网络的环境下才能使用，在弱网或没有网的情况下基本罢工。

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智能家居语音控制系统。该系统具有如下特点：

1. 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。
2. 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

智能家居语音控制系统功能性需求比较明显，示例如图X为用户通过“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，系统会通过喇叭播放“空调已打开”的提示术语。



**图4-4 智能家居语音控制系统功能分析图**

结果功能分析图，该系统功能性需求分为十个部分，各部分内容如下：

* 外围通信控制功能

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。芯片选型时应当注意。

* 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。

* 唤醒功能

当用户喊出关键词时可以唤醒系统。含有关键词的音频经训练好的唤醒引擎后，触发唤醒，系统唤醒后，继续将录音结果送入下一阶段，进行ASR识别。

* 网络检测功能

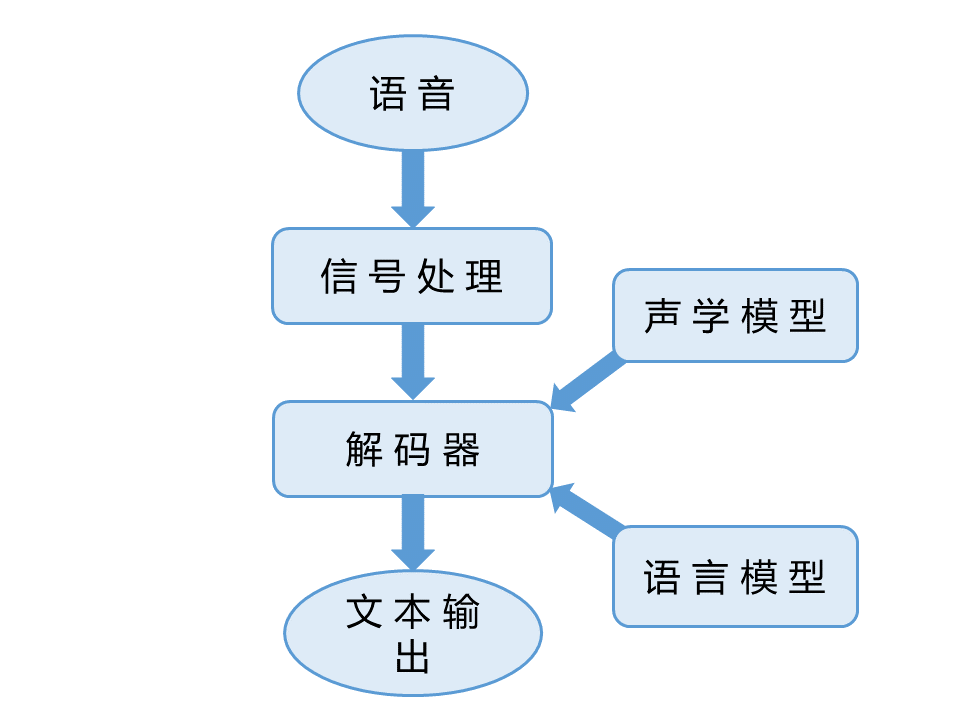
该功能可以是一个独立任务，每隔一分钟进行网络状态判断并设置网络状态标志位。在用户发起请求时能立即拿到相应状态进行在线离线模式切换。

* 在线语音识别功能

进行语音到文字转化，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，然后上传给云端。该功能依赖三方平台支持，接入百度的在线语音识别服务，百度提供了比较全面的API接口。

* 离线识别功能

该系统应同时支持在网络状况不好或未接入网络的情况下选择离线识别引擎。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。语音识别系统框架如图4-1



**图4-1 语音识别系统框架图**

* 识别结果关键词分类和匹配功能

系统应对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4-2所示



**图4-2 使用控制用例图**

* 离线在线TTS文本转音频功能

在处理状态文本时可以通过判断网络状态选择在线或离线TTS将文本转化为可播放的音频文件。在线TTS同样可接入百度的TTS服务器，离线TTS需要通过深度学习训练TTS模型。控制结果以TTS音频与用户交互，如图4-3



**图4-3 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，通过喇叭播放音频数据。

## 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，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。

# 第三章 系统总体设计

## 3.1 系统设计原则

（1）系统性原则。在本控制系统设计中，遵循系统性的原则，从整个控制系统的角度进行考虑，各个模块要相互配合，保证系统的完整性一致性等。  
（2）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。  
（3）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。  
（4）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

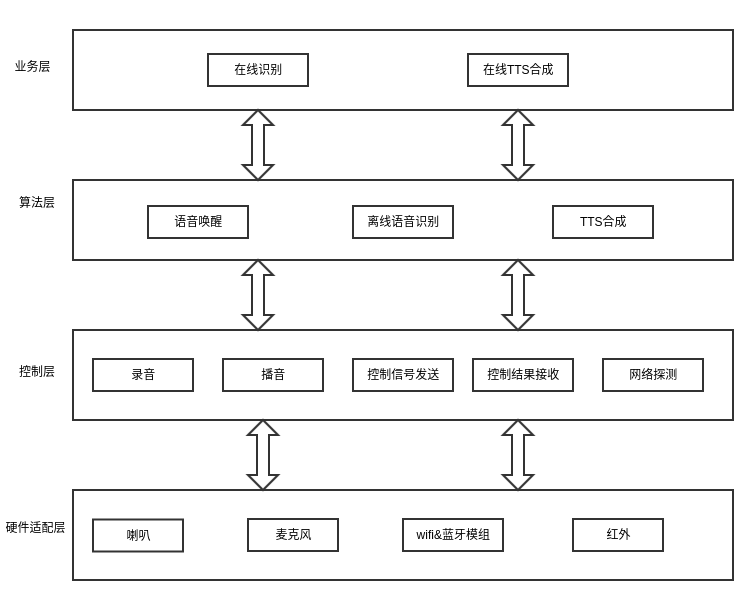
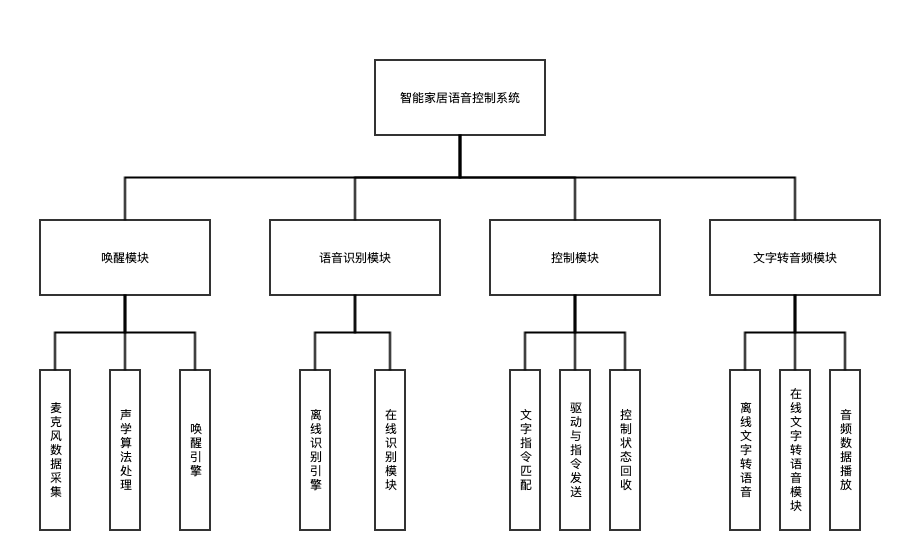
如图4-4是整个语音控制系统的架构设计图，该架构设计图比较直观的展示了整个系统的层次结构，其中包含硬件适配层，控制层，算法层，业务层四个层次。每个层次功能专一，又与其他层进行通信交互。

图1.1 系统架构图

## 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4-5所示



**图4-5 系统功能结构图**

唤醒模块：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。

降噪音频

录音

麦克风

声学算法

唤醒引擎

唤醒模块流程图

语音识别模块：该模块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ASR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。

控制模块：主要包含文字指令匹配，外设的驱动控制和指令发送，另外该模块还接受外设的指令执行结果，最终用来通知用户指令的执行情况。

文字转音频模块：该模块包含离线文字转语音模块，在线文字转语音模块和音频数据播放

## 3.4 系统网络拓扑

整体网络拓扑如图

其中家中智能家电报活冰箱，蓝牙开关，空调，手机等，这些智能设备均通过无线wifi模组与家中路由器wifki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。



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器，一个是sina.com，一个是ai.baidu.com。sina.com这个server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。ai.baidu.com这个server提供了我们在线模式下的asr语音识别功能，我们可通过api接口访问这个server并实现在线的asr识别。与sina.com相同的一个作用是ai.baidu.com也同样作为外网探测地址。

## 3.5 系统数据库设计

### 3.5.1数据模型结构

### 3.5.2 关键数据库表结构设计

## 3.6 接口设计

1) 音频采集

void thread\_audio\_recorder\_by\_alsa();

1. 降噪接口

void thread\_audio\_aec\_process();

1. 网络探活

void thread\_network\_accessable();

1. ASR接口

离线ASR请求

void thread\_asr\_process\_offline();

在线ASR请求

void thread\_asr\_process\_online();

1. 结果解析

void system\_instruct\_process();

1. 控制指令发送

void system\_instruct\_2\_hal();

1. TTS接口

在线TTS请求

void thread\_tts\_request\_online();

离线TTS请求

void thread\_tts\_request\_offline()

1. 音频播放接口

void thread\_audio\_play();

## 3.7 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。

该模块计划采用深度学习方法解决。因为语音识别的目的是对给定的波形序列，可以得到相应的单词或者字符序列。因此语音识别可以被看作是一个信道解码或者模式分类问题。统计建模是目前主流的语音识别方法。基于统计建模框架，对于给定语音波形序列O，我们可以采用贝叶斯决策的最大后验概率（Maximum A Posteriori，MAP）估计得到最优的输出序列W\*。其中条件概率P(O|W) 表示模型生成观察序列的概率，对应语音识别系统的声学模型（Acoustic Model，AM）。似然值P(W)则表示序列W出现的一个先验概率，称之为语言模型（Language Model，LM）。如图4-6是一个标注的语音识别系统的结构框图，其主要由前端处理，声学模型，语言模型，解码器四个模块组成。解码的过程主要是利用所训练的声学模型和语言模型，搜索得到最佳的输出序列。



**图4-6 语音识别系统结构图**

## 3.8 本章小结

本章主要介绍了系统的总体设计，从设计原则到系统架构设计，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。

# 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⽤时⽅式和性能表现，指定可⽤于⼈⼯评估的智能嵌⼊式⽅案，让AI赋能评估，使系统的实现和表现更加准确⾼效。

### 开发板选型

从系统、计算芯⽚类型、开发环境、⽣态完善等多个因素考虑，选择以下⽅案进⾏探索并调研：

a. 瑞芯微RK3399开发板

b. 瑞芯微RK3399PRO开发板

c. 华为Atlas200DK开发板

d. Nvida JetSon TX2 开发板

e. Nvida Jetson Xavier NX开发板

选型调研过程（以⼈体关键点识别为例）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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） | Android  Linux | Python、C/C++  Java | 预估2.0Tops | 主流框架无加速 | 同框架 |
| 2 | 瑞芯微RK3399PRO | GPU  NPU | 中等（RKNN等环境编译较麻烦） | Android  Linux | Python、  C/C++、  Java | 3.0Tops | 主流框架⽆NPU加速，只能官⽅ 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（需要科学上网） | 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（需要科学上网） | Linux | Python、  C/C++ | 21Tops | 都支持但无加速TensorRT可加速 | 同框架 |

以上数据是根据个人体验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，如表

|  |  |  |  |
| --- | --- | --- | --- |
| 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。 | 性能⼀般，对性能要求⽐较低的场景可以使⽤ |
| 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。 | 性能⼀般，相⽐  RK3399有NPU加速，对性能要求⼀般的场景可以使⽤ |
| 3 | 华为Atlas200DK | a. 验证官⽅Demo。 | 从搭建环境Demo跑通，踩坑最多的⽅案，⽂档不明确、模型⽀持少，使⽤较为繁琐，⽣态环境差。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，性能⼀般，开发难度⼀般，可以满⾜模型精度、以及实时性要求不⾼的场景。优点是⽀持Android系统。

2）华为Atlas200DK，算⼒虽然强，但开发部署过于繁琐，且⽀模型持较少⽣态不完善，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，⽣态完善，主流框架都可⽀持，适合快速开发产品。Xavier NX⽐TX2增加了TensorCore以及CUDA核⼼数，性能更强，优先选择Xavier NX。

### Nvida Jetson Xavier NX环境搭建

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 CarmelARM®v8.264位CPU，GPU则是有384g个NVIDIACUDA®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。

1）材料准备

NX开发板+19V电源电流1A，TF卡，建议32G以上，路由器、⽹线X2，电脑和开发板组⽹。

2）烧录固件

在Linux电脑上下载并安装Nvidia SDK Manager并运⾏。下载镜像，使⽤USB连接开发板和电脑，照软件提⽰进⾏烧录。安装所需软件和环境，与Ubuntu类似。安装Pytorch。使⽤开发板进⾏开发测试 。

### 硬件适配层的设计与实现

## 4.1 音频处理模块设计与实现

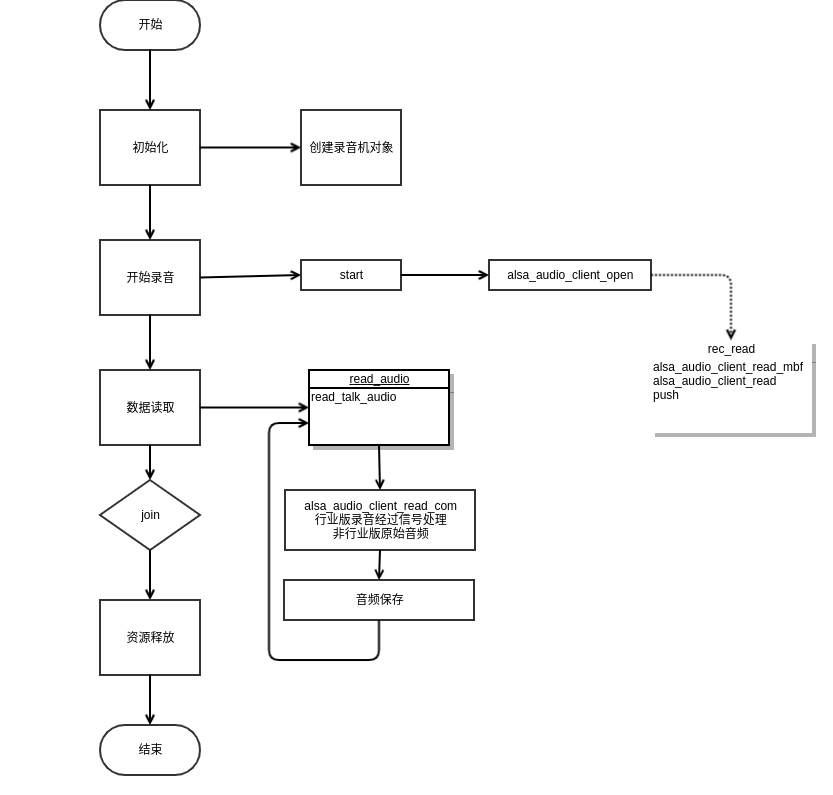
### 音频采集

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。

ALSA在user space里以ALSA-Lib存在，即提供API给应用程序调用。

linux下编写本地录音流程如图xx



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

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

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

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

val = 48000;

snd\_pcm\_hw\_params\_set\_rate\_near(handle, params, &val, &dir);

设置一个录音周期为32包

frames = 32;

snd\_pcm\_hw\_params\_set\_period\_size\_near(handle,params, &frames, &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

snd\_pcm\_hw\_params\_get\_period\_size(params, &frames,&dir);

size = frames \* 4; /\* 2 bytes/sample, 2 channels \*/

buffer = (char \*) malloc(size);

打开一个文件，将录音音频流写入文件

fopen("record\_dump.raw", "a+");

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，编译方法方法：

gcc -o local\_player local\_player.c -L ./alsa-utils-1.1.5 -lasound -lm -ldl

测试方法：

播放:通过喇叭是否正常播放音频文件，确认是否播放成功。

录音:通过dump出来的文件，用audiocity或其他软件打开并播放，确认录音是否正常。

### 音频降噪

AEC算法早期用在Voip，电话这些场景中，自从智能设备诞生后，智能语音设备也要消除自身的音源，这些音源包括音乐或者TTS机器合成声音。

回声消除的基本原理是使用一个自适应滤波器对未知的回声信道:ω \omegaω 进行参数辨识，根据扬声器信号与产生的多路回声的相关性为基础，建立远端信号模型，模拟回声路径，通过自适应算法调整，使其冲击响应和真实回声路径相逼近。然后将麦克风接收到的信号减去估计值，即可实现回声消除功能。

式中ω是回声通道的时域冲击响应函数，x是远端语音；echo是所得回声；s是近端说话人语音，d为麦克风采集到的信号，γ是对回声信号的估计值，e为误差。

为了消除较长时间的回声，需要FIR滤波器的阶数较高，时域计算法，有两个问题，一个是实时性较差，一个是计算量大。为了在实时性/计算量以及可以消除的回声时长之间找到使这三个最优的算法，采用了频谱分块自适应滤波算法。这里用到了很多信号处理算法，简单罗列涉及到的算法：

* FFT/IFFT
* 循环卷积和线性卷积的关系;重叠保留法
* 功率谱密度
* 互相关
* NLMS自适应算法

## 4.2 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](https://links.jianshu.com/go?to=https://snowboy.kitt.ai/" \t "_blank)，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。

### 1、获取源代码并编译

安装依赖

树莓派原生的音频设备是不支持语音输入的（无法录音），需要在网上购买一支免驱动的**USB音频驱动**（便携式的和 U 盘差不多），一般插上即可直接使用。  
建议安装下 pulseaudio 软件，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。

**安装其他软件依赖**：

* 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

**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**）

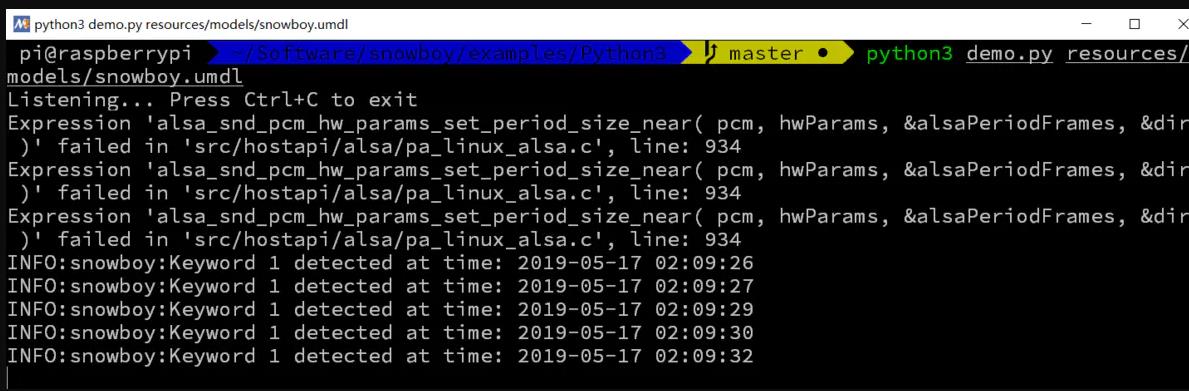
然后对着麦克风清晰地讲出**“snowboy”**，如果可以听到“滴”的声音，则安装配置成功。命令行输出如下：

snowboy test

**PS**：官方源代码使用 Python3 测试有报错，经测试需修改 snowboy/examples/Python3 目录下的 snowboydecoder.py 文件。  
将第 5 行代码 from \* import snowboydetect 改为 import snowboydetect 即可直接运行。

### 2、设置自己的唤醒词

可将包含自定义唤醒词的音频文件上传至 snowboy [官网](https://links.jianshu.com/go?to=https://snowboy.kitt.ai/dashboard" \t "_blank)（需要登录），以训练生成自己喜欢的语音模型。需要上传的音频文件数量为 3 个，wav 格式，可直接在线录制



训练完成并测试通过后，即可下载 PMDL 后缀的模型文件了。

### 3.测试

将以下文件复制到自己的项目目录下：

* 上一步中下载好的 model.pmdl 模型文件
* 之前 snowboy/swig/Python3 目录下编译好的 \_snowboydetect.so 库
* snowboy/examples/Python3 目录下的 demo.py、snowboydecoder.py、snowboydetect.py 文件以及 resources 目录
* 在项目目录下执行 $ python3 demo.py model.pmdl 并使用自己的唤醒词进行测试

### 4.自定义响应

官方提供的示例 demo.py 文件的源代码如下：

定义中断回调

def interrupt\_callback():

global interrupted

return interrupted

定义信号的处理函数

def signal\_handler(signal, frame):

global interrupted

interrupted = True

if len(sys.argv) == 1:

print("Error: need to specify model name")

print("Usage: python demo.py your.model")

sys.exit(-1)

对信号处理函数进行初始化。

model = sys.argv[1]

signal.signal(signal.SIGINT, signal\_handler)

设置热词的模型并初始化引擎。

detector = snowboydecoder.HotwordDetector(model)

主循环开始，也是进入等待唤醒模式。

detector.start(detected\_callback=snowboydecoder.play\_audio\_file,

interrupt\_check=interrupt\_callback,

sleep\_time=0.03)

detector.terminate()

唤醒词识别成功以后，程序响应的具体内容由程序末尾 detector.start() 函数的 detected\_callback 参数指定。即重新绑定 detected\_callback 对应的函数，改变程序最终的响应。

检测到唤醒后的处理函数:

def detected():

print("Great! I have recognized your words.\n")

if len(sys.argv) == 1:

print("Error: need to specify model name")

print("Usage: python demo.py your.model")

sys.exit(-1)

可通过sensitivity设置唤醒阈值

detector = snowboydecoder.HotwordDetector(model, sensitivity=0.5)print('Listening... Press Ctrl+C to exit')

主循环开始

detector.start(detected\_callback=detected,

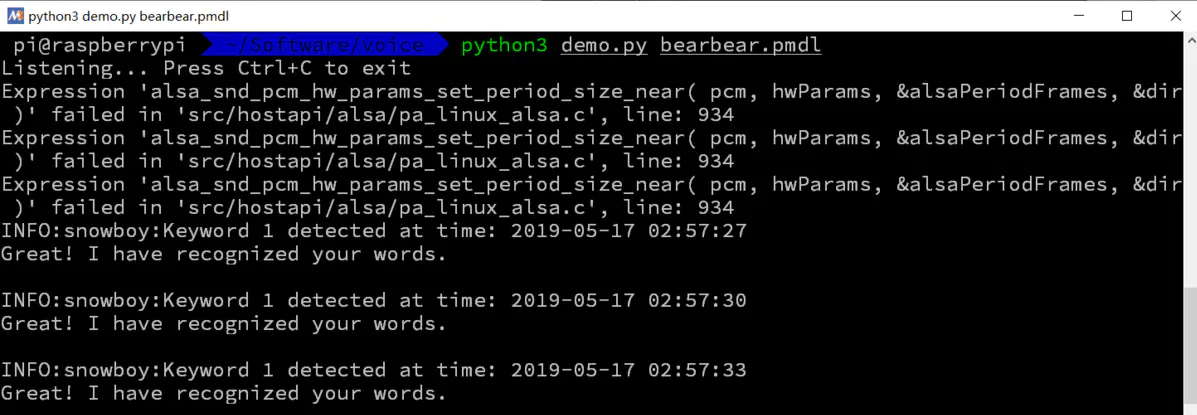
interrupt\_check=interrupt\_callback,

sleep\_time=0.03)

detector.terminate()

**注意**添加的 detected 函数。

效果如下：



本系统使用了更复杂的应用形式，唤醒后通过网络探测模块获取网络标记，并开始将录音重新组装给到语音识别模块，唤醒模块也就打开了整个系统控制的大门。

## 4.2 语音识别模块设计与实现

### 离线识别

Kaldi是当前最流行的开源语音识别工具(Toolkit)，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](https://link.zhihu.com/?target=http://www.codercto.com/category/python.html" \t "https://zhuanlan.zhihu.com/p/_blank)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。

**External Libraries**

**Kaldi C++ Library**

**(Shell) Scripts**

**Kaldi C++ Executables**

**Transforms**

**Feat**

**SGMM**

**GMM**

**Utils**

**Matrix**

**Decodable**

**LM**

**Tree**

**FST ext**

**HMM**

**Decoder**

**BLAS/LAPACK**

**OpenFST**

Kaldi功能模块结构图

1、OpenFST：Weighted Finite State Transducer library，是一个用来构造有限状态自动机的库。是一个构造、合并、优化和搜索加权有限状态机FST的库。我们知道隐马尔科夫模型就可以看成是一个有限状态自动机的。这是最终要的一个包，Kaldi的文档里面说：If you ever want to understand Kaldi deeply you will need to understand OpenFst.  
2、ATLAS：这是一个C++下的线性代数库。做机器学习自然是需要很多矩阵运算的。  
3、IRSTLM:这是一个统计语言模型的工具包。  
4、sph2pipe:这是宾夕法尼亚大学linguistic data consortium（LDC）开发的一款处理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。  
5、SCTK：Speech Recognition Scoring Toolkit是NIST（National Institute of Standards and Technology, 美国国家标准与技术协会）提出的一套工具集。NIST评分工具包  
6、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。

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。这个数字当然是越低越好。

一、docker 环境

因环境复杂性，选择使用docker环境，综合网上资料和个人习惯选择ubuntu，怕20.04太新，选择18.04作为基础环境。

二、kaldi环境

1. 代码下载

git clone https://github.com/kaldi-asr/kaldi

2. 执行依赖检查

kaldi# tools/extras/check\_dependencies.sh

按提示安装依赖

3. 按INSTALL说明，有两个方式进行构建，可选择其中之一

1）bash + Makefile

tools目录下按INSTALL执行

①extras/check\_dependencies.sh

②make 这个地方编译三方包时会出现killed问题，需要内存>4G ，修改docker启动参数如下

docker run -it -d -m 4G fengzhishang/kaldi\_env:1 /bin/bash

src 目录下按INSTALL执行

./configure --shared make depend -j 8 make -j 8

编译过程极其吃内存，多线程更严重，4个G的内存基本都要被吃掉，没办法改成make编译了好久。

cmake（提示该方式未完整测试，不建议，故采用1）

Kaldi中文语音识别公共数据集一共有4个，分别是：

1.aishell: AI SHELL公司开源178小时中文语音语料及基本训练脚本，见kaldi-master/egs/aishell

2.gale\_mandarin: 中文新闻广播数据集(LDC2013S08, LDC2013S08）

3.hkust: 中文电话数据集(LDC2005S15, LDC2005T32)

4.thchs30: 清华大学30小时的数据集，可以在http://www.openslr.org/18/下载

今天我们来运行thchs30数据集。

一、数据准备

首先我们需要下载语料库：

下载地址为：http://www.openslr.org/18/

里面共有3个文件，分别是：

data\_thchs30.tgz [6.4G] ( speech data and transcripts )

test-noise.tgz [1.9G] ( standard 0db noisy test data )

resource.tgz [24M] ( supplementary resources, incl. lexicon for training data, noise samples )

下载后随便解压到一个文件夹里，例如在egs/thchs30/s5下新建了一个文件夹thchs30-openslr，然后把三个文件解压在了该文件夹下

这个数据集包含以下内容：

数据集 音频时长(h) 句子数 词数

train(训练) 25 10000 198252

dev(开发) 2:14 893 17743

test(测试) 6:15 2495 49085

还有训练好的语言模型word.3gram.lm和phone.3gram.lm以及相应的词典lexicon.txt。

其中dev的作用是在某些步骤与train进行交叉验证的，如local/nnet/run\_dnn.sh同时用到exp/tri4b\_ali和exp/tri4b\_ali\_cv。训练和测试的目标数据也分为两类：word（词）和phone（音素）。

1.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）

2.#produce 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。

3.#prepare 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，如果不打算改语言模型，这段代码也不需要修改。

a)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

b)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。

c)这两个语言模型都是由SRILM工具训练得到。

二、修改脚本

1.首先修改s5下面的cmd.sh脚本，把原脚本注释掉，修改为本地运行：

#export train\_cmd=queue.pl

#export decode\_cmd="queue.pl --mem 4G"

#export mkgraph\_cmd="queue.pl --mem 8G"

#export cuda\_cmd="queue.pl --gpu 1"

export train\_cmd=run.pl

export decode\_cmd="run.pl --mem 4G"

export mkgraph\_cmd="run.pl --mem 8G"

export cuda\_cmd="run.pl --gpu 1"

2.然后修改s5下面的run.sh脚本，需要修改两个地方：

第一个地方是修改并行任务的数量，可以根据cpu的个数来定

#n=4 #parallel jobs

n=2 #parallel jobs

第二个地方是修改数据集放的位置，例如我修改的为：

#thchs=/nfs/public/materials/data/thchs30-openslr

thchs=/root/kaldi-trunk/egs/thchs30/s5/thchs30-openslr

三、运行

在s5下执行./run.sh，就会开始运行。

大概有几个过程：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（这个我也不懂），后面就是dnn了

因为我是在虚拟机上运行的，所以运行的非常慢，建议大家尽量不要再虚拟机上运行，找一个配置好的机器，最好有GPU，这样运行速度比较快。

当运行到dnn时候会报错，因为默认dnn都是用GPU来跑的。它会检查一下，发现只在CPU下，就终止了。这里建议不要跑dnn了，想跑dnn的话还是找GPU吧。

四、其它

算法流程

1.首先用标准的13维MFCC加上一阶和二阶导数训练单音素GMM系统，采用倒谱均值归一化（CMN）来降低通道效应。然后基于具有由LDA和MLLT变换的特征的单音系统构造三音GMM系统，最后的GMM系统用于为随后的DNN训练生成状态对齐。

2.基于GMM系统提供的对齐来训练DNN系统，特征是40维FBank，并且相邻的帧由11帧窗口（每侧5个窗口）连接。连接的特征被LDA转换，其中维度降低到200。然后应用全局均值和方差归一化以获得DNN输入。DNN架构由4个隐藏层组成，每个层由1200个单元组成，输出层由3386个单元组成。 基线DNN模型用交叉熵的标准训练。 使用随机梯度下降（SGD）算法来执行优化。 将迷你批量大小设定为256，初始学习率设定为0.008。

3.被噪声干扰的语音可以使用基于深度自动编码器（DAE）的噪声消除方法。DAE是自动编码器（AE）的一种特殊实现，通过在模型训练中对输入特征引入随机破坏。已经表明，该模型学习低维度特征的能力非常强大，并且可以用于恢复被噪声破坏的信号。在实践中，DAE被用作前端管道的特定组件。输入是11维Fbank特征（在均值归一化之后），输出是对应于中心帧的噪声消除特征。然后对输出进行LDA变换，提取全局标准化的常规Fbank特征，然后送到DNN声学模型（用纯净语音进行训练）。

训练与解码脚本解读

本节结合官方文档对主要脚本进行解读。

以下流程中的符号解释：->表示下一步，{}表示循环，[]表示括号内每一个都要进行一次，()表示不同分支下可能进行的操作

1.train\_mono.sh 用来训练单音子隐马尔科夫模型，一共进行40次迭代，每两次迭代进行一次对齐操作

gmm-init-mono->compile-train-graphs->align-equal-compiled->gmm-est->

{gmm-align-compiled->gmm-acc-stats-ali->gmm-est}40->analyze\_alignments.sh

2.train\_deltas.sh 用来训练与上下文相关的三音子模型

check\_phones\_compatible.sh->acc-tree-stats->sum-tree-stats->cluster-phones->compile-questions->

build-tree->gmm-init-model->gmm-mixup->convert-ali->compile-train-graphs->

{gmm-align-compiled->gmm-acc-stats-ali->gmm-est}35->analyze\_alignments.sh

3.train\_lda\_mllt.sh 用来进行线性判别分析和最大似然线性转换

check\_phones\_compatible.sh->split\_data.sh->ali-to-post->est-lda->acc-tree-stats->sum-tree-stats->

cluster-phones->compile-questions->build-tree->gmm-init-model->convert-ali->compile-train-graphs->

{gmm-align-compiled->gmm-acc-stats-ali->gmm-est}35->analyze\_alignments.sh

4.train\_sat.sh 用来训练发音人自适应，基于特征空间最大似然线性回归

check\_phones\_compatible.sh->ali-to-post->acc-tree-stats->sum-tree-stats->cluster-phones->compile-questions->

build-tree->gmm-init-model->gmm-mixup->convert-ali->compile-train-graphs->

{gmm-align-compiled->(ali-to-post->)gmm-acc-stats-ali->gmm-est}35->ali-to-post->

gmm-est->analyze\_alignments.sh

5.train\_quick.sh 用来在现有特征上训练模型。

对于当前模型中在树构建之后的每个状态，它基于树统计中的计数的重叠判断的相似性来选择旧模型中最接近的状态。

check\_phones\_compatible.sh->ali-to-post->est-lda->acc-tree-stats->sum-tree-stats->

cluster-phones->compile-questions->build-tree->gmm-init-model->convert-ali->compile-train-graphs->

{gmm-align-compiled->gmm-acc-stats-ali->gmm-est}20->analyze\_alignments.sh

6.run\_dnn.sh 用来训练DNN，包括xent和MPE，

{make\_fbank.sh->compute\_cmvn\_stats.sh}[train,dev,test]->train.sh->{decode.sh}[phone,word]->

align.sh->make\_denlats.sh->train\_mpe.sh->{{decode.sh}[phone,word]}3

7.train\_mpe.sh 用来训练dnn的序列辨别MEP/sMBR。

这个阶段训练神经网络以联合优化整个句子，这比帧级训练更接近于一般ASR目标。

sMBR的目的是最大化从参考转录对齐导出的状态标签的期望正确率，而使用网格框架来表示竞争假设。

训练使用每句迭代的随机梯度下降法。

首先使用固定的低学习率1e-5（sigmoids）运行3-5轮。

在第一轮迭代后重新生成词图，我们观察到快速收敛。

我们支持MMI, BMMI, MPE 和sMBR训练。所有的技术在Switchboard 100h集上是相同的，仅仅在sMBR好一点点。

在sMBR优化中，我们在计算近似正确率的时候忽略了静音帧。

{nnet-train-mpe-sequential}3->make\_priors.sh

8.train\_dae.sh 用来实验基于dae的去噪效果

compute\_cmvn\_stats.sh->{add-noise-mod.py->make\_fbank.sh->compute\_cmvn\_stats.sh}[train,dev,test]->

train.sh->nnet-concat->{{decode.sh}[phone,word]}[train,dev,test]

9.train.sh 用来训练深度神经网络模型，帧交叉熵训练，该相位训练将帧分类为三音状态的DNN。这是通过小批量随机梯度下降完成的。

默认使用Sigmoid隐藏单元，Softmax输出单元和完全连接的AffineTransform层，学习率是0.008，小批量的大小为256。

我们没有使用动量或正则化（注：最佳学习率和隐藏单元的类型不同，sigmoid的值为0.008,tanh为0.00001。

通过‘–feature-transform’和‘-dbn’将input——transform和预训练的DBN传入此脚本，只有输出层被随机初始化。

我们使用提前停止来防止过度拟合，为此我们测量交叉验证集合（即保持集合）上的目标函数，

因此需要两对特征对齐dir来执行监督训练

feat-to-dim->nnet-initialize->compute-cmvn-stats->nnet-forward->nnet-concat->cmvn-to-nnet->

feat-to-dim->apply-cmvn->nnet-forward->nnet-initialize->train\_scheduler.sh

10.train\_scheduler.sh 典型的情况就是，train\_scheduler.sh被train.sh调用。

一开始需要在交叉验证集上运行，主函数需要根据$iter来控制迭代次数和学习率。

学习率会随着目标函数相对性的提高而变化：

如果提高大于’start\_halving\_impr=0.01’，初始化学习率保持常数

否则学习率在每次迭代中乘以’halving\_factor=0.5’来缩小

最后，如果提高小于’end\_halving\_impr=0.001’，训练终止。

11.mkgraph.sh 用来建立一个完全的识别网络

12.decode.sh 用来解码并生成词错率结果

13.align\_si.sh 对制定的数据进行对齐，作为新模型的输入

14.make\_fmllr\_feats.sh 用来保存FMLLR特征

15.pretrain\_dbn.sh 深度神经网络预训练脚本

16.decode\_fmllr.sh 对发音人自适应的模型进行解码操作

17.nnet-train-frmshuff.cc 最普遍使用的神经网络训练工具，执行一次迭代训练。过程：

–feature-transform 即时特征扩展

NN输入-目标对的每帧重排

小批量随机梯度下降（SGD）训练

支持的每帧目标函数（选项 - 对象函数）：

Xent：每帧交叉熵

Mse：每帧均方误差

18.nnet-forward.cc 通过神经网络转发数据，默认使用CPU。选项：

–apply-log :产生神经网络的对数输出(比如：得到对数后验概率)

–no-softmax :从模型中去掉soft-max层

—class-frame-counts：从声学得分中减去计算对数的计数

专有缩写中文解释

cmvn：倒谱均值和方差归一化

fft：快速傅里叶变换

GMM：高斯混合模型

MFCC：梅尔倒谱系数

pcm：脉冲编码调制

pdf：概率分布函数

PLP：感知线性预测系数

SGMM：子空间高斯混合模型

UBM：通用背景模型

VTLN：特征级声道长度归一化

### 在线识别

BDSSDKMessage 包可以认为是命令的参数，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。

name 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

识别调用流程

获取实例

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，不要复用。

get\_instance最多可以保持10个实例，即最多同时识别10个音频。

std::string err\_msg;

bds::BDSpeechSDK\* sdk = bds::BDSpeechSDK::get\_instance(bds::SDK\_TYPE\_ASR, err\_msg);

设置回调监听器

设置结果的回调函数。注意回调产生在SDK内部的线程中。

void asr\_output\_callback(bds::BDSSDKMessage& message, void\* user\_arg);

sdk->set\_event\_listener(&asr\_output\_callback, (void\*)& thread\_seq);

// 此例中 thread\_seq会作为user\_arg参数，传入asr\_output\_callback回调。

设置SDK配置参数

bds::BDSSDKMessage cfg\_params;

cfg\_params.name = bds::ASR\_CMD\_CONFIG;

cfg\_params.set\_parameter(bds::COMMON\_PARAM\_KEY\_DEBUG\_LOG\_LEVEL, sdk\_log\_level);

bool is\_success =sdk->post(cfg\_params, err\_msg);

// is\_success表示设置是否成功，具体错误见err\_msg

cfg\_params.set\_parameter中可以设置的参数列表请见“参数列表:输入配置参数”一节

设置SDK 启动参数

这里只需要填写ASR\_PARAM\_KEY\_APP参数，填写您自定义的应用名称即可，方便百度服务端联调及统计。

bds::BDSSDKMessage start\_params;

start\_params.name = bds::ASR\_CMD\_START;

start\_params.set\_parameter(bds::ASR\_PARAM\_KEY\_APP, "AIPtest");

bool is\_success =sdk->post(start\_params, err\_msg);

// is\_success表示设置是否成功，具体错误见err\_msg

cfg\_params.set\_parameter中可以的参数列表说明请见“参数列表:输入启动参数”一节

传递音频数据

音频流的音频格式为raw，或者是pcm文件输入流。单声道，16bits， 小端序。

bds::BDSSDKMessage push\_params;

push\_params.name = bds::ASR\_CMD\_PUSH\_AUDIO;

push\_params.set\_parameter(bds::DATA\_CHUNK, audio\_buf, (int)read\_cnt); // read\_cnt=320

//如果实时音频，此处不需要sleep。如果是文件流，此处需要 sleep ， sleep的时长为音频时长。

//即服务端处理一段音频需要等同于音频原始时长的耗时，不能上传过快，否则服务端行为不能预期。

bool is\_success =sdk->post(push\_params, err\_msg);

// is\_success表示设置是否成功，具体错误见err\_msg

每次传递音频的大小可以根据需要来，建议320字节 。

两次音频数据调用请在10s内完成。否则有异常出现。

停止或者暂停流程

停止音频流输入

告诉SDK 音频流已经输入完毕，不再有后续音频。 需要调用以下2行代码：

push\_params.set\_parameter(bds::DATA\_CHUNK, audio\_buf, 0)

sdk->post(stop\_params, err\_msg));

push\_params.set\_parameter(bds::DATA\_CHUNK, audio\_buf, 0);

bds::BDSSDKMessage stop\_params;

stop\_params.name = bds::ASR\_CMD\_STOP;

bool is\_success = sdk->post(stop\_params, err\_msg));

// is\_success表示设置是否成功，具体错误见err\_msg

取消识别

告诉SDK 本次识别取消，即用户不再需要识别结果。

bds::BDSSDKMessage cancel\_params;

cancel\_params.name = bds::ASR\_CMD\_CANCEL;

bool is\_success = sdk->post(cancel\_params, err\_msg));

// is\_success表示设置是否成功，具体错误见err\_msg

在设置的event\_listener输出回调中，SDK返回EvoiceRecognitionClientWorkStatusCancel事件

释放资源

bds::BDSpeechSDK::release\_instance(sdk);

与get\_instance 对应 SDK不是处于空闲状态（见下面的空闲状态定义），调用 bds::BDSpeechSDK::release\_instance可能引起程序出core。

清理所有线程池

所有识别结束，不需要发起新的识别。

SDK空闲时才能执行

bds::BDSpeechSDK::do\_cleanup();

其它

SDK空闲状态定义

SDK空闲是指 没有开始识别或者实例产生了以下3个回调后：

EVoiceRecognitionClientWorkStatusLongSpeechEnd // 长语音识别过程正常结束

EVoiceRecognitionClientWorkStatusError // 发生错误

EVoiceRecognitionClientWorkStatusCancel // 用户手动取消当前识别

本次识别的最后一点数据发完之后，并且post了长度为0的数据，并且post了stop命令后，最长(通常都不会超过)12s，sdk就会产生上述3个回调。

不建议用户发送ASR\_CMD\_CANCEL命令来强制结束识别，如果有需要，也可以这样做；但在没收到上述三个回调状态时，即不是空闲状态调用 bds::BDSpeechSDK::release\_instance可能引起程序出core。

设置日志

默认日志根据您设置的日志级别输出到stderr中，可以自行指定日志文件。 注意需要在引擎空闲的时候，如一开始执行。 日志需要设置 COMMON\_PARAM\_KEY\_DEBUG\_LOG\_LEVEL = EVRDebugLogLevelTrace 开启

BDSpeechSDK::open\_log\_file(const char \*logFileName, int fileSize = 0);

logFileName 日志的路径

fileSize 日志文件的最大size， 实际size为 fileSize \*512k

如果日志文件大于size， SDK会自动将当前日志文件重命名为xxx.bak文件（并删除历史xxx.bak），并新建一个日志文件xxx。所以这样只能最多同时保留两个日志文件。

您可以根据fileSize参数定期运行一个cron任务，备份xxx.bak文件

BDSpeechSDK::close\_log\_file();

close\_log\_file函数与open\_log\_file配对使用。在引擎空闲的时候，如全部结束后执行。

统计日志格式

开启日志并且日志级别在EVRDebugLogLevelInformation以上。 Sdk日志里带有统计信息，以Statistics开头(s大写)，如下示例：

[INFO]-[2017-12-08 15:08:28.598][bds\_EventManagerAsr.cpp:307]Statistics info, sn[e89f9705-2ca9-43bb-9790-0328a69ab1f4], sn\_cost[11245]ms, response\_cost[1085]ms, hard\_delay[16]ms, recv\_interval:[11229]ms, now:[1512716908598612], recv\_first\_pack\_time:[1512716897352635], recv\_last\_pack\_time:[1512716908582388], audio\_len[10.620000]s

其中, 一个sn代表一句话的识别标志logId

audio\_len： 该句话的音频长度

sn\_cost ：产生这个sn到这个sn对应的语音识别结束耗时， 可近似当作sdk识别该句话的总耗时；

recv\_interval ：为sdk从用户侧接收这句话总的间隔时间, 即recv\_first\_pack\_time和recv\_last\_pack\_time之差

response\_cost：响应时间，为sdk收到用户post来的该句话的第一个包到第一次给出用户识别结果(可能是部分结果)的耗时；

hard\_delay ：硬延时，类似response\_cost，但是是计算最后一个包的。sdk收到用户post来的该句话的最后一个包到给出用户最终识别结果(可能报错信息)的耗时；

recv\_first\_pack\_time和recv\_last\_pack\_time：sdk用户post来的该句话的第一包和最后一包的时间点，实际用户每次传递多少数据可能有所不同，且传来时还未分句，但不影响，sdk会回做标记，并计算出每句话的首尾包是哪次传来的，以及传过来的时间点。 通常sn\_cost 应该略大于recv\_interval，而recv\_interval应该近似等于audio\_len（时间上），即一句的识别时间和该句话的音频长度相当(略长出几十到几百ms)，如果觉得sdk识别慢，请先关注这几个时间，以排查问题在大概哪一侧。

## 4.3 MQTT

由于物联网的环境是非常特别的，所以MQTT遵循以下设计原则：

（1）精简，不添加可有可无的功能；

（2）发布/订阅（Pub/Sub）模式，方便消息在传感器之间传递；

（3）允许用户动态创建主题，零运维成本；

（4）把传输量降到最低以提高传输效率；

（5）把低带宽、高延迟、不稳定的网络等因素考虑在内；

（6）支持连续的会话控制；

（7）理解客户端计算能力可能很低；

（8）提供服务质量管理；

（9）假设数据不可知，不强求传输数据的类型与格式，保持灵活性。

4.1 MQTT协议实现方式

实现MQTT协议需要客户端和服务器端通讯完成，在通讯过程中，MQTT协议中有三种身份：发布者（Publish）、代理（Broker）（服务器）、订阅者（Subscribe）。其中，消息的发布者和订阅者都是客户端，消息代理是服务器，消息发布者可以同时是订阅者。

MQTT传输的消息分为：主题（Topic）和负载（payload）两部分：

（1）Topic，可以理解为消息的类型，订阅者订阅（Subscribe）后，就会收到该主题的消息内容（payload）；

（2）payload，可以理解为消息的内容，是指订阅者具体要使用的内容。

4.2 网络传输与应用消息

MQTT会构建底层网络传输：它将建立客户端到服务器的连接，提供两者之间的一个有序的、无损的、基于字节流的双向传输。

当应用数据通过MQTT网络发送时，MQTT会把与之相关的服务质量（QoS）和主题名（Topic）相关连。

4.3 MQTT客户端

一个使用MQTT协议的应用程序或者设备，它总是建立到服务器的网络连接。客户端可以：

（1）发布其他客户端可能会订阅的信息；

（2）订阅其它客户端发布的消息；

（3）退订或删除应用程序的消息；

（4）断开与服务器连接。

4.4 MQTT服务器

MQTT服务器以称为"消息代理"（Broker），可以是一个应用程序或一台设备。它是位于消息发布者和订阅者之间，它可以：

（1）接受来自客户的网络连接；

（2）接受客户发布的应用信息；

（3）处理来自客户端的订阅和退订请求；

（4）向订阅的客户转发应用程序消息。

4.5 MQTT协议中的订阅、主题、会话

一、订阅（Subscription）

订阅包含主题筛选器（Topic Filter）和最大服务质量（QoS）。订阅会与一个会话（Session）关联。一个会话可以包含多个订阅。每一个会话中的每个订阅都有一个不同的主题筛选器。

二、会话（Session）

每个客户端与服务器建立连接后就是一个会话，客户端和服务器之间有状态交互。会话存在于一个网络之间，也可能在客户端和服务器之间跨越多个连续的网络连接。

三、主题名（Topic Name）

连接到一个应用程序消息的标签，该标签与服务器的订阅相匹配。服务器会将消息发送给订阅所匹配标签的每个客户端。

四、主题筛选器（Topic Filter）

一个对主题名通配符筛选器，在订阅表达式中使用，表示订阅所匹配到的多个主题。

五、负载（Payload）

消息订阅者所具体接收的内容。

4.6 MQTT协议中的方法

MQTT协议中定义了一些方法（也被称为动作），来于表示对确定资源所进行操作。这个资源可以代表预先存在的数据或动态生成数据，这取决于服务器的实现。通常来说，资源指服务器上的文件或输出。主要方法有：

（1）Connect。等待与服务器建立连接。

（2）Disconnect。等待MQTT客户端完成所做的工作，并与服务器断开TCP/IP会话。

（3）Subscribe。等待完成订阅。

（4）UnSubscribe。等待服务器取消客户端的一个或多个topics订阅。

（5）Publish。MQTT客户端发送消息请求，发送完成后返回应用程序线程。

五、MQTT协议数据包结构

在MQTT协议中，一个MQTT数据包由：固定头（Fixed header）、可变头（Variable header）、消息体（payload）三部分构成。MQTT数据包结构如下：

（1）固定头（Fixed header）。存在于所有MQTT数据包中，表示数据包类型及数据包的分组类标识。

（2）可变头（Variable header）。存在于部分MQTT数据包中，数据包类型决定了可变头是否存在及其具体内容。

（3）消息体（Payload）。存在于部分MQTT数据包中，表示客户端收到的具体内容。

5.1 MQTT固定头

固定头存在于所有MQTT数据包中，其结构如下：

5.1.1 MQTT数据包类型

位置：Byte 1中bits 7-4。

相于一个4位的无符号值，类型、取值及描述如下：

5.1.2 标识位

位置：Byte 1中bits 3-0。

在不使用标识位的消息类型中，标识位被作为保留位。如果收到无效的标志时，接收端必须关闭网络连接：

（1）DUP：发布消息的副本。用来在保证消息的可靠传输，如果设置为1，则在下面的变长中增加MessageId，并且需要回复确认，以保证消息传输完成，但不能用于检测消息重复发送。

（2）QoS：发布消息的服务质量，即：保证消息传递的次数

Ø00：最多一次，即：<=1

Ø01：至少一次，即：>=1

Ø10：一次，即：=1

Ø11：预留

（3）RETAIN： 发布保留标识，表示服务器要保留这次推送的信息，如果有新的订阅者出现，就把这消息推送给它，如果设有那么推送至当前订阅者后释放。 5.1.3 剩余长度（Remaining Length）

地址：Byte 2。

固定头的第二字节用来保存变长头部和消息体的总大小的，但不是直接保存的。这一字节是可以扩展，其保存机制，前7位用于保存长度，后一部用做标识。当最后一位为1时，表示长度不足，需要使用二个字节继续保存。例如：计算出后面的大小为0

5.2 MQTT可变头

MQTT数据包中包含一个可变头，它驻位于固定的头和负载之间。可变头的内容因数据包类型而不同，较常的应用是作为包的标识：

很多类型数据包中都包括一个2字节的数据包标识字段，这些类型的包有：PUBLISH (QoS > 0)、PUBACK、PUBREC、PUBREL、PUBCOMP、SUBSCRIBE、SUBACK、UNSUBSCRIBE、UNSUBACK。

5.3 Payload消息体

Payload消息体位MQTT数据包的第三部分，包含CONNECT、SUBSCRIBE、SUBACK、UNSUBSCRIBE四种类型的消息：

（1）CONNECT，消息体内容主要是：客户端的ClientID、订阅的Topic、Message以及用户名和密码。

（2）SUBSCRIBE，消息体内容是一系列的要订阅的主题以及QoS。

（3）SUBACK，消息体内容是服务器对于SUBSCRIBE所申请的主题及QoS进行确认和回复。

（4）UNSUBSCRIBE，消息体内容是要订阅的主题。

## 4.4 网络检测模块设计与实现

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。

网络初始化

开始

网络探活

curl ai.baidu.com

curl sina.com

Sleep 1min

结果收集

标志文件写入

图1.1 网络探测流程图

关键代码实现：

#目标网站

local target=www.sina.com

local target=www.ai.baidu.com

local ret\_code=`curl -I -s --connect-timeout ${timeout} ${target} -w %{http\_code} | tail -n1`

if [ "x$ret\_code" = "x200" ]; then

#网络畅通

return 1

else

#网络不畅通

return 0

fi

## 4.5 TTS模块

TTS模块直接使用百度AI开发平台提供的TTS能力，简要步骤如下：

1)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。免费额度说明详见语音合成免费额度文档。

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TTS文本设计如下：

“空调已打开”“空调已关闭”

“冰箱已打开”“冰箱已关闭”

“电视已打开”“电视已关闭”

4）编写程序

安装语音合成 C++ SDK

语音合成 C++ SDK目录结构

├── base

│ ├── base.h // 请求客户端基类

│ ├── base64.h // base64加密相关类

│ ├── http.h // http请求封装类

│ └── utils.h // 工具类

└── speech.h // 语音合成 交互类

最低支持 C++11+

直接使用开发包步骤如下：

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。

6.调用示例可以参考 https://github.com/Baidu-AIP/sdk-demo

新建client

client是语音合成的C++客户端，为使用语音合成的开发人员提供了一系列的交互方法。当您引入了相应头文件后就可以新建一个client对象

用户可以参考如下代码新建一个client：

//设置APPID/AK/SK

std::string app\_id = "myAppID";

std::string api\_key = "myApikey";

std::string secret\_key = "mySecretKey";

aip::Speech client(app\_id, api\_key, secret\_key);

在上面代码中，常量APP\_ID在百度云控制台中创建，常量API\_KEY与SECRET\_KEY是在创建完毕应用后，系统分配给用户的，均为字符串，用于标识用户，为访问做签名验证，可在AI服务控制台中的应用列表中查看。

请求说明

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。文本长度不可超过限制

要把一段文字合成为语音文件：

void tts(aip::Speech client)

{

std::ofstream ofile;

std::string file\_ret;

std::map<std::string, std::string> options;

options["spd"] = "9";

// 合成成功的二进制数据写入文件中

ofile.open("./test.mp3", std::ios::out | std::ios::binary);

// 不带可选参数调用

Json::Value result = client.text2audio("百度语音合成测试", aip::null, file\_ret);

// 带可选参数调用, 参数参考参数列表中的可选参数

// Json::Value result = client.text2audio("百度语音合成测试", options, file\_ret);

// 如果file\_ret为不为空则说明合成成功，返回mp3文件内容回结果

if (!file\_ret.empty())

{

ofile << file\_ret;

} else {

// 服务端合成错误

std::cout << result.toStyledString();

}

}

参数 类型 描述 是否必须

tex String 合成的文本，使用UTF-8编码，

请注意文本长度必须小于1024字节 是

cuid String 用户唯一标识，用来区分用户，

填写机器 MAC 地址或 IMEI 码，长度为60以内 否

spd String 语速，取值0-9，默认为5中语速 否

pit String 音调，取值0-9，默认为5中语调 否

vol String 音量，取值0-15，默认为5中音量 否

per String 发音人选择, 0为女声，1为男声，

3为情感合成-度逍遥，4为情感合成-度丫丫，默认为普通女 否

返回样例：

返回Json::value对象。 如果合成成功，error\_code为Json::nullValue, 下行数据为二进制语音文件数据。 如果合成出现错误，则会填充除data属性外的其它属性。

返回失败：

// 失败返回

{

"err\_no":500,

"err\_msg":"notsupport.",

"sn":"abcdefgh",

"idx":1

}

## 4.6 关键技术和解决方案

## 4.7 本章小结

# 第五章 系统测试分析

## 5.1 测试概述

系统测试指的是将已确认的软件、计算机硬件、网络、外设等元素结合在一起，进而系统的组装测试和确认测试，目的是为了与系统的需求进行比较，从而找出所开发的系统是否与用户的需求有不符或者是矛盾的情况，从而提出更加完善的方案.。它的任务是尽可能彻底地检查出程序中的错误，提高软件系统的可靠性，其目的是检验系统做的效果。系统测试是作为计算机系统的一部分，为保证系统正常运行，在实际运行环境下对计算机系统进行的一系列严格而有效的测试。系统测试的目的是验证最终的软件系统是否满足用户规定的要求。

主要内容包括：

功能测试。 即测试软件系统的功能是否正确，依据是需求文档。 由于正确性是软件最重要的质量因素，因此功能测试必不可少。

稳健性测试。 即测试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智能家居语音控制系统，软件部分即使满足功能要求，也未必能够满足性能要求，虽然从单元测试起，每一测试步骤都包含性能测试，领测认为只有当系统真正集成之后，在真实环境中才能全面、可靠地测试运行性能系统性能测试是为了完成这一任务。

## 5.2 测试工具及测试环境

测试工具包括：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。

|  |  |  |  |
| --- | --- | --- | --- |
| 安静 | 一米 | 三米 | 五米 |
| 噪音 | 一米 | 三米 | 五米 |

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。另外，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，unixbench是一个用于测试unix系统性能的工具，也是一个比较通用的benchmark， 此测试的目的是对我们的只能家居语音控制系统提供一个基本的性能指示，很多测试用于系统性能的不同方面，这些测试的结果是一个指数值，这个值是测试系统的测试结果与一个基线系统测试结果比较得到的指数值，这样比原始值更容易得到参考价值，测试集合里面所有的测试得到的指数值结合起来得到整个系统的指数值。

* 测试系统的单任务性能
* 测试系统的多任务性能
* 测试系统并行处理的能力

另一个Netperf是一种网络性能测量工具，主要基于TCP或UDP的传输。Netperf根据应用的不同，可以进行不同模式的网络性能测试，即批量数据传输（bulk data transfer）模式和请求/应答（request/reponse）模式。Netperf 工具以 client/server 方式工作。server 端是 netserver，用来侦听来自 client 端的连接，client 端是 netperf ，用来向 server 发起网络测试。

## 5.3 测试方法及流程

系统测试要包括基本的功能测试、性能测试，可靠性测试，安全性测试，兼容性测试、安装卸载测试、安全性测试、访问控制测试、故障转移测试、故障恢复测试、分布测试、性能测试的几个方面。

1. 功能测试：测试软件各个功能模块是否能够正确运行，业务逻辑是否符合预期，数据流转、呈现、处理和存储和预期一致。

**测试方法**

1、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）

2、将录制的音频通过工具灌入进行识别测试

3、根据识别结果利用wer工具计算字准和句准

测试流程  
   播测，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。

二、性能测试：测试系统整体的性能，是否能达到基本的使用指标，系统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。

三、可靠性测试：根据产品可靠性结构、寿命类型和各单元的可靠性试验信息，利用概率统计方法，评估出产品的可靠性特征量。软件可靠性是软件系统在规定的时间内以及规定的环境条件下，完成规定功能的能力。一般情况下，只能通过对软件系统进行测试来度量其可靠性。测试可靠性是指运行应用程序，以便在部署系统之前发现并移除失败。因为通过应用程序的可选路径的不同组合非常多，所以在一个复杂应用程序中不可能找到所有的潜在失败。但是，可测试在正常使用情况下最可能的方案，然后验证该应用程序是否提供预期的服务。如果时间允许，可采用更复杂的测试以揭示更微小的缺陷。可靠性测试的方法包括组件压力测试，针对智能家居语音控制系统的压测测试，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，TTS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，对智能家居语音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。

四、安全性测试：

1.功能验证

功能验证是采用软件测试当中的黑盒测试方法,对涉及安全的软件功能,如：用户管理模块，权限管理，加密系统，认证系统等进行测试，主要验证上述功能是否有效。

2.漏洞扫描

安全漏洞扫描主要是借助于特定的漏洞扫描器完成的。通过使用漏洞扫描器，系统管理员能够发现系统存在的安全漏洞，从而在系统安全中及时修补漏洞的措施。一般漏洞扫描分为两种类型：主机漏洞扫描器是指在系统本地运行检测系统漏洞的程序。网络漏洞扫描器是指基于网络远程检测目标网络和主机系统漏洞的程序。

3.模拟攻击

对于安全测试来说，模拟攻击测试是一组特殊的极端的测试方法，我们以模拟攻击来验证软件系统的安全防护能力。系统安全测试的内容：它主要包括

* 应用程序安全测试
* 操作系统安全测试
* 数据库安全测试
* 网络环境安全测试

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完善。

5.4 系统功能测试

软件环境：

{

"voice\_hi\_pid": "hi\_211",

"voice\_hi\_package": "com.hi.intell",

"voice\_hi\_server\_url": " [https://hi.baidu.com/intell](https://email.baidu.com/owa/redir.aspx?C=ACnnrt6SLT8twSH5uUk2IirbtaPcbNml2qG0399JdwN-H51xnDXVCA..&URL=https://vse.baidu.com/v2) ",

"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](https://email.baidu.com/owa/redir.aspx?C=RldHZcR-YMvRzVdDVOD2DAZL6_raAJd3zSYDAE8NY5x-H51xnDXVCA..&URL=https://xiaodu.baidu.com:443/saiya/ws) ",

"outputlog\_hi\_console":"0",

"clicklog\_hi\_url":" [https://hi.baidu.com/saiya/log?](https://email.baidu.com/owa/redir.aspx?C=HZzFikHn-OBdYBgq2HDU5nSzsMXvDzAesooG3u7jIaF-H51xnDXVCA..&URL=https://xiaodu.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](https://email.baidu.com/owa/redir.aspx?C=r5qeRkhaS_R6hNXR1_xYIbaQTWntoJ6wWq_atzOdbQp-H51xnDXVCA..&URL=https://xiaodu.baidu.com/ws) ",

"debug\_hi\_ws\_url":"",

"debug\_hi\_token":"",

"request\_hi\_login\_token\_url":" [https://hi.baidu.com:443/saiya/device/token](https://email.baidu.com/owa/redir.aspx?C=1doA1O9xAXumBWhV70ajwWcweR_Mqr_IwImSOLZK2hR-H51xnDXVCA..&URL=https://xiaodu.baidu.com:443/saiya/device/token) "

}

识别测试数据：

播测识别测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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 |

性能测试：

Unixbench数据统计，运行三次取平均值

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |  |  |  |  |  |  |  |

几个指标含义如下，

Score：整体跑分评估，是一个综合指标，直接标识该系统的整体计算性能。

1x\_score:单核评估分数，是一个进行单位

Double:测试聚焦在浮点运算操作。这个测试用于测试链接器编译、代码优化、内存缓存、等待状态、整数数据类型等，硬件和软件设计都会非常大的影响测试结果。

FileCp\_1024:这项测试衡量文件数据从一个文件被传输到另外一个，使用大量的缓存。包括文件的读、写、复制测试，测试指标是一定时间内（默认是10秒）被重写、读、复制的字符数量，文件大小按1024字节操作。

FileCp\_256:文件大小按256字节操作。

PipeCtxSw:这项测试衡量两个进程通过管道交换和整数倍的增加吞吐的次数。基于管道的上下文切换和真实程序很类似。测试程序产生一个双向管道通讯的子线程。

SysCall:这项测试衡量进入和离开系统内核的消耗，例如，系统调用的消耗。程序简单重复的执行getpid调用（返回调用的进程id）。消耗的指标是调用进入和离开内核的执行时间。

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数据统计

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 |  |  |  |  |  |  |

各项指标意义如下，

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

# 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]陈孝良.语音交互为何迭代如此之快[J].科技导报,2017, 35(3): 92-92.

[2]王智国.嵌入式人机语音交互系统关键技术研究[D].中国科学技术大学,2014.

[3]陈振锋,吴蔚澜,刘加,等.基于Mel倒谱特征顺序统计滤波的语音端点检测算法 [J]. 中国科学院大学学报, 2014, 31(4): 524-529.

[4]Li X, Li G, Li X. Improved voice activity detection based on iterative spectral subtraction and double thresholds for CVR[C]//Power Electronics and Intelligent Transportation System, 2008. PEITS'08. Workshop on. IEEE, 2008: 153-156.

[5]薛胜尧.基于改进型双门限语音端点检测算法的研究[J].电子设计工程, 2015 (4): 78-81