**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**

[1. 课题来源和意义 3](#_Toc56462362)

[1.1 课题来源 3](#_Toc56462363)

[1.2 课题背景 3](#_Toc56462364)

[1.3 选题意义 4](#_Toc56462365)

[2. 国内外相关研究现状 4](#_Toc56462366)

[2.1 国外研究现状 4](#_Toc56462367)

[2.2 国内研究现状 4](#_Toc56462368)

[3. 研究目标及内容 5](#_Toc56462369)

[3.1 研究目标 5](#_Toc56462370)

[3.2 研究内容 5](#_Toc56462371)

[3.2.1 嵌入式芯片和系统的选型 5](#_Toc56462372)

[3.2.2 语音唤醒功能 6](#_Toc56462373)

[3.2.3 语音识别 6](#_Toc56462374)

[3.2.4 控制系统 6](#_Toc56462375)

[3.3 研究方法与技术线路 6](#_Toc56462376)

[4. 关键技术难点和技术解决方案 7](#_Toc56462377)

[4.1 系统需求分析 8](#_Toc56462378)

[4.1.1功能需求分析 8](#_Toc56462379)

[4.1.2 非功能需求分析 10](#_Toc56462380)

[4.2 系统架构初步方案设计 11](#_Toc56462381)

[4.3 系统功能结构初步方案设计 12](#_Toc56462382)

[4.4 关键技术难点和解决方案 13](#_Toc56462383)

[5. 预期成果及可能的创新点 14](#_Toc56462384)

[6. 论文工作计划 14](#_Toc56462385)

[7. 参考文献 15](#_Toc56462386)

# 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

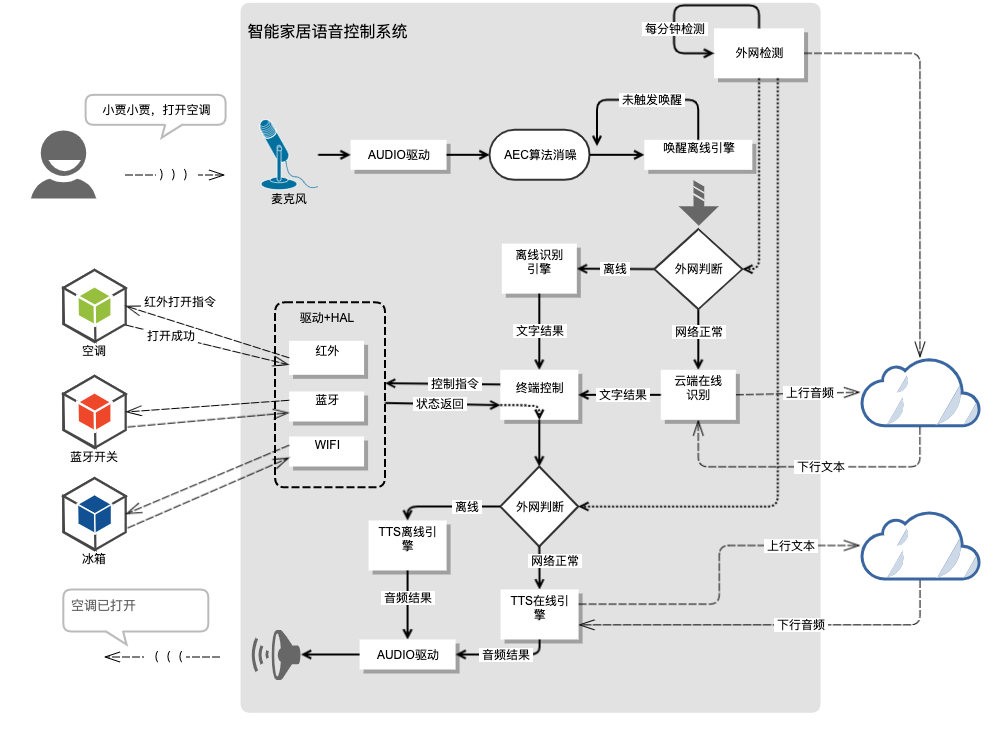
目前市面上的音箱具有重云轻端的特点，往往丢失了嵌入式系统本身灵活和控制的特点。目前家居虽然向着智能的方向发展，支持移动互联网的远程控制。但仍有大量的家居设备是按键控制，好一点的是红外或蓝牙，但相对功能单一，无法通过一个智能的语音中控进行统一控制。另一方面，智能语音的代表智能音箱，都是在有网络的环境下才能使用，在弱网或没有网的情况下基本罢工。

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

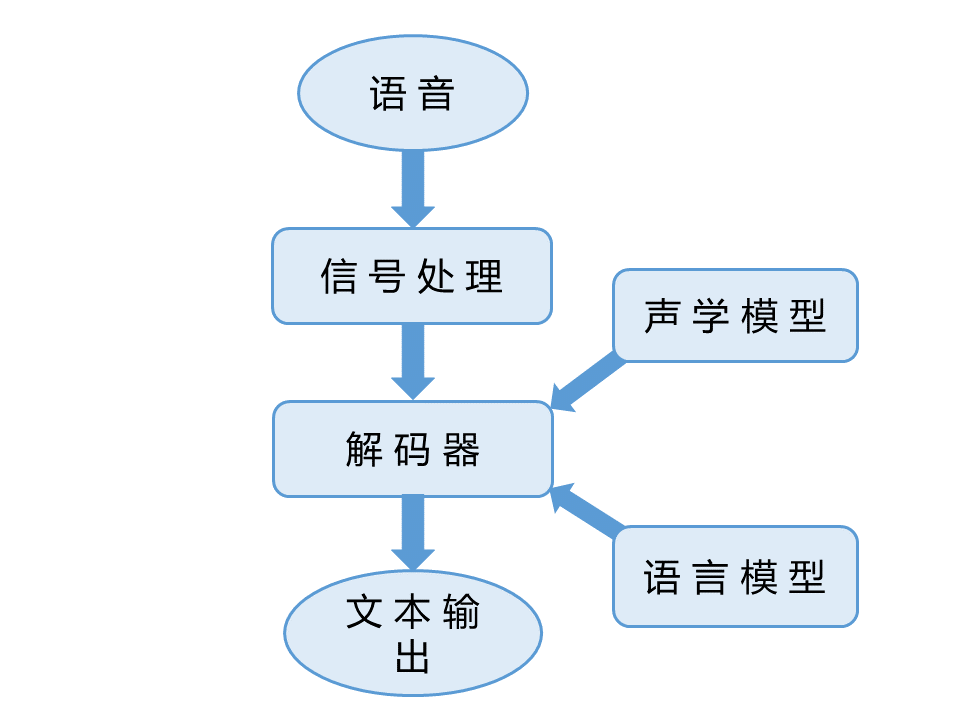
该功能可以是一个独立任务，每隔一分钟进行网络状态判断并设置网络状态标志位。在用户发起请求时能立即拿到相应状态进行在线离线模式切换。

* 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

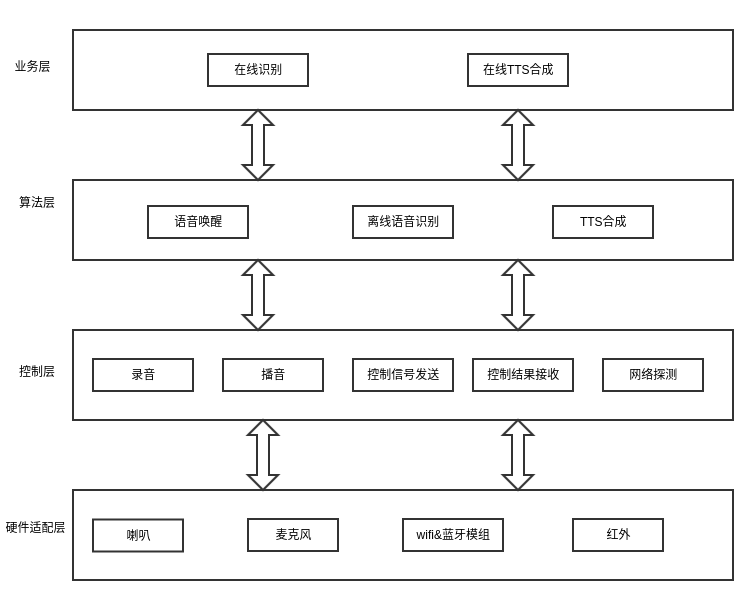
本章首先对市面上的一些智能家居语音控制系统进行了现状分析，针对分析出的痛点进行需求分析，将本系统的需求分为功能性需求和非功能性需求系统。从功能性需求角度将系统分为几个功能模块，从非功能性需求角度分析，本系统应具有一定性能和使用性。

# 第三章 系统总体设计

## 3.1 系统设计原则

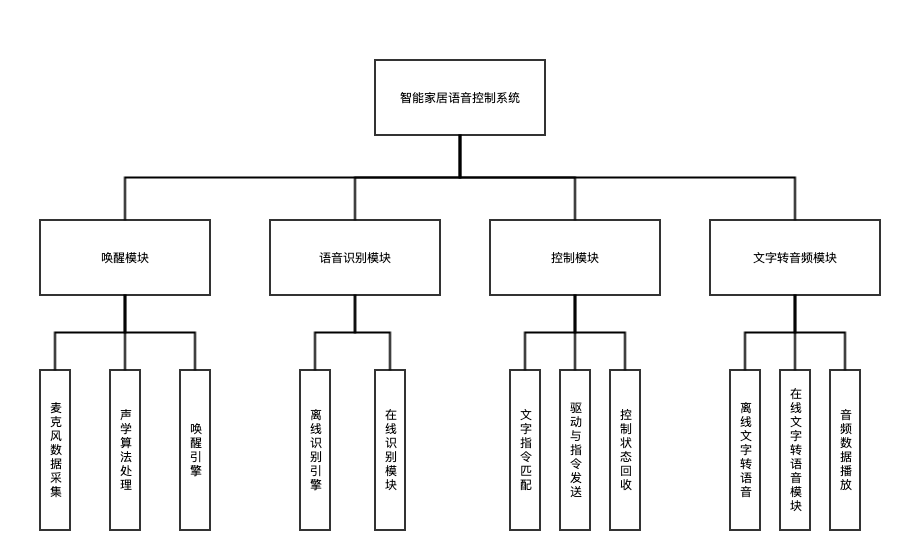
（1）系统性原则。在本控制系统设计中，遵循系统性的原则，从整个控制系统的角度进行考虑，各个模块要相互配合，保证系统的完整性一致性等。  
（2）灵活性及可变性原则。本控制系统要有一定的灵活性，随着可控制外设的增加，要有一定的灵活性和扩展性，能够对外界环境变化的适应能力，比如在网络可用的情况下支持在线功能，在网络不稳定或断连的情况下支持离线模式。  
（3）可靠性原则。本控制系统要有一定的抗干扰能力，语音唤醒和语音识别对于外部环境具有一定的要求，在系统的设计过程中为了加强噪音情况下的唤醒率和语音识别准确度，要考虑增加一些降噪功能，在硬件和软件算法上都可以采取必要的降噪或语音增强方法能够抵御外界干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。

## 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生效算法模块进行处理，最终将数据送入唤醒引擎

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常的测试地址。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，使实验评估更加准确⾼效。

### 芯片开发板选型

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 | 生态完善程度 | 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 | 一般 | 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 | 一般 | 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 | 差 | 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 | 完善 | 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 | 完善 | 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

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

a. 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。

b. 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。

c. 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。

5. Nvida Jetson Xavier NX环境搭建&使⽤

a. 材料准备

i. NX开发板+19V电源 （

1A就够）

ii. TF卡，建议32G以上

iii. 路由器、⽹线X2（可选，电脑和开发板组⽹，开发板也⽀持WIFI）

iv. ⿏标键盘（开发板⽤）

b. 烧录固件

i. ⽅式⼀：制作SD卡

1. 格式化SD卡。

2. 下载镜像，下载地址https://developer.nvidia.com/zh-cn/embedded/downloads 。

3. 把镜像刷⼊SD卡，可⽤Win32DiskImager、balenaEtcher等软件。

4. SD卡插⼊开发板，等待开机配⽹。

ii. ⽅式⼆：NVIDIA SDK Manager 刷⼊系统 。

1. 使⽤USB连接开发板和电脑，因开发板不带EMMC需要插⼊SD卡 。

2. 在Linux电脑上下载并安装Nvidia SDK Manager并运⾏ 。

3. 按照软件提⽰进⾏烧录。

c. 安装所需软件和环境，与Ubuntu类似 。

i. 安装Pytorch。

d. 像使⽤Linux⼀样，正常使⽤开发板进⾏开发测试 。

这是一颗高性能的八核心智能硬件处理器，其具有以太网接口，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。

全志R58内存方面支持DDR3/DDR3L/LPDDR3/LPDDR2多种规格，满足不同成本定位应用。存储支持EMMC4.5，全志R58采用FCBGA封装，345pin，14\*14mm面积。

开发板套件基本参数介绍

**CPU：** Amlogic A113X A53x4 up to 1.5Ghz

**Memory：** 256MB DDR3 + 512MB Nand Flash

**Mic Array：** PDM mic x 4

**Audio：**line out 3.5mm

**Network：**802.11 a/b/g/n/ac

**Bluetooth:**BT 4.1/BLE

**LED:**RGB LED x 12

**DC in：**USB-Type C 5V

### 系统

代码获取

repo 是Google官方版本

repo init -u ssh://your-account@openai.rokid.com:29418/amlogic\_a113\_audio/manifest -m rokidbase.xml

repo 是Rokid 修正版本

repo init -u ssh://your-account@openai.rokid.com:29418/amlogic\_a113\_audio/manifest -m rokidbase.xml --repo-url=ssh://your-account@openai.rokid.com:29418/tools/repo --no-repo-verify

下载代码树

repo sync

U-Boot

修改了厂商代码，支持Rokid的板级配置目录

Kernel

修改了厂商代码，支持Rokid多型号板子的DST配置目录

编译指令

目前仅支持64位版本。

编译A113请选择'banban\_m\_a113\_release'

source rokid\_br\_external/build/setenv.sh

输出

Environment setting is OK!

Just type 'lunch' and you will get a list of choices, or you can type 'lunch [choice]' to lunch directly.

lunch

输出

You are building on Linux

echo Lunch menu... pick a combo:

1. nana\_t\_s905d\_release

2. nana\_l\_a112\_release

3. rm101\_s905d\_release

4. rp102\_s905d\_release

5. banban\_m\_a113\_release

6. nana\_t2\_s905d\_release

7. banban\_m2\_a113\_release

8. banban\_m3\_a113\_release

Which would you like?

其中

banban\_m\_a113\_release:为 A113 6mic v1.0 麦克风板，版本号 v1.0 在麦克风板正中央丝印上

banban\_m2\_a113\_release: 为 A113 4mic 麦克风板

banban\_m3\_a113\_release: 为 A113 6mic v1.1 麦克风板, 版本号 v1.1 在麦克风板正中央丝印上

编译代码

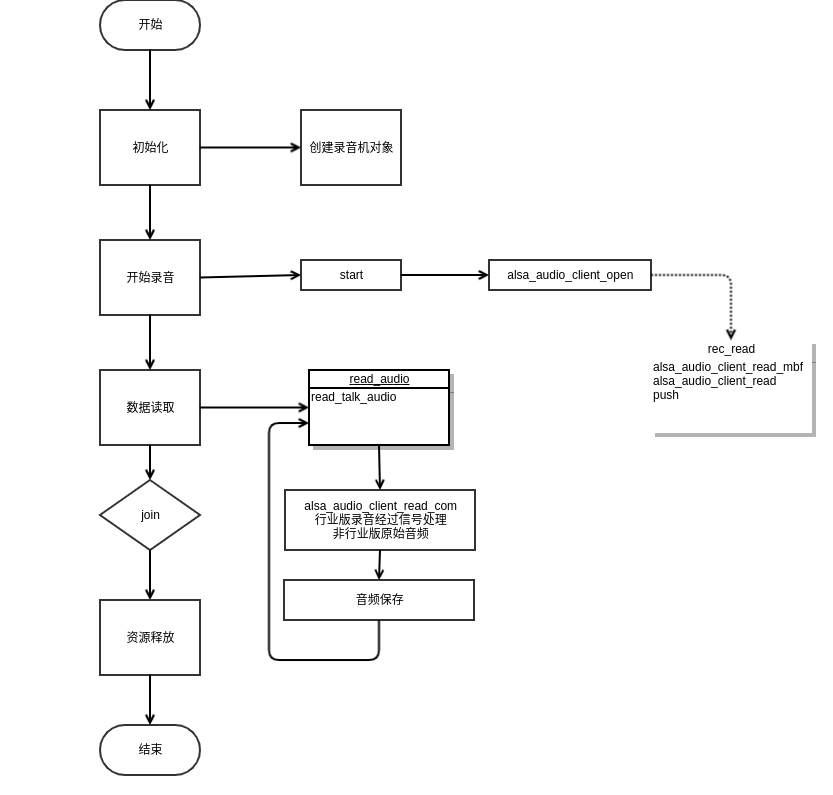
make

### 外设和驱动

### 网络

### 控制协议

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下编写本地录音和播放demo测试代码：

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，具体见下文：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

> File Name: local\_player.c

> Desc. : Read from file and playback, or capture and write to file.

> Author : horistudio@163.com

> Date : 2020/05/26

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Use the newer ALSA API \*/

#define ALSA\_PCM\_NEW\_HW\_PARAMS\_API

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <alsa/asoundlib.h>

int main(int argc, char \*argv[]) {

int rc;

int size;

snd\_pcm\_t \*handle;

snd\_pcm\_hw\_params\_t \*params;

unsigned int val;

int dir;

snd\_pcm\_uframes\_t frames;

char \*buffer;

/\* Open PCM device for playback. \*/

rc = snd\_pcm\_open(&handle, "default",

SND\_PCM\_STREAM\_PLAYBACK, 0);

/\* Open PCM device for capture. \*/

rc = snd\_pcm\_open(&handle, "default",

SND\_PCM\_STREAM\_CAPTURE, 0);

if (rc < 0) {

fprintf(stderr,

"unable to open pcm device: %s/n",

snd\_strerror(rc));

exit(1);

}

/\* Allocate a hardware parameters object. \*/

snd\_pcm\_hw\_params\_alloca(&params);

/\* Fill it in with default values. \*/

snd\_pcm\_hw\_params\_any(handle, params);

/\* Set the desired hardware parameters. \*/

/\* Interleaved mode \*/

snd\_pcm\_hw\_params\_set\_access(handle, params,

SND\_PCM\_ACCESS\_RW\_INTERLEAVED);

/\* Signed 16-bit little-endian format \*/

snd\_pcm\_hw\_params\_set\_format(handle, params,

SND\_PCM\_FORMAT\_S16\_LE);

/\* Two channels (stereo) \*/

snd\_pcm\_hw\_params\_set\_channels(handle, params, 8);

/\* 44100 bits/second sampling rate (CD quality) \*/

val = 48000;

snd\_pcm\_hw\_params\_set\_rate\_near(handle, params,

&val, &dir);

/\* Set period size to 32 frames. \*/

frames = 32;

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

params, &frames, &dir);

/\* Write the parameters to the driver \*/

rc = snd\_pcm\_hw\_params(handle, params);

if (rc < 0) {

fprintf(stderr,

"unable to set hw parameters: %s/n",

snd\_strerror(rc));

exit(1);

}

/\* Use a buffer large enough to hold one period \*/

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

&dir);

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

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

printf("------>frames:%d, size:%d\n",frames,size);

#if 1

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*[3.文件操作]\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*通过主函数传参，传入.wav格式文件名\*/

FILE \*fp\_r = NULL;

int ret = 0;

fp\_r = fopen(argv[1], "rb");

if(fp\_r == NULL) {

printf("open file fail!\n");

exit(1);

}

/\* 打开一个文件，将录音音频流写入文件 \*/

FILE \*fp = NULL;

int file\_len = 0;

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

if(fp == NULL) {

printf("open file fail!\n");

exit(1);

}

printf("Configurations are finished......\n");

while (1) {

/\* 1.read from file and playback \*/

//read audio file

memset(buffer,0,size);

rc = fread(buffer, 1, size, fp\_r);

if (rc == 0) {

fprintf(stderr, "end of file on input/n");

break;

} else if (rc != size) {

fprintf(stderr,

"short read: read %d bytes/n", rc);

}

rc = snd\_pcm\_writei(handle, buffer, frames);

if (rc == -EPIPE) {

/\* EPIPE means underrun \*/

fprintf(stderr, "underrun occurred/n");

snd\_pcm\_prepare(handle);

} else if (rc < 0) {

fprintf(stderr,

"error from writei: %s/n",

snd\_strerror(rc));

} else if (rc != (int)frames) {

fprintf(stderr,

"short write, write %d frames/n", rc);

}

/\* 2.capture and write to file \*/

/\* capture \*/

memset(buffer, 0, size);

rc = snd\_pcm\_readi(handle, buffer, frames);

printf("snd\_pcm\_readi,frames:%d, buffer:%s\n",frames, buffer);

if (rc == -EPIPE)

{

/\* EPIPE means overrun \*/

fprintf(stderr, "overrun occurred\n");

snd\_pcm\_prepare(handle);

} else if (rc < 0) {

fprintf(stderr,

"error from read: %s\n",

snd\_strerror(rc));

} else if (rc != (int)frames) {

fprintf(stderr,

"short read, read %d frames\n", rc);

}

if (fp) {

file\_len = fwrite(buffer, 1, size, fp);

printf("fwrite:%d bytes \n", file\_len);

} else {

printf("%s[%d] file open fail\n",\_\_func\_\_,\_\_LINE\_\_);

}

}

fclose(fp);

//close(fd\_f);

snd\_pcm\_drain(handle);

snd\_pcm\_close(handle);

free(buffer);

return 0;

}

\*上述代码，可以根据需要解开注释或者打上注释，分别测试录音或者播放

编译时，需要引入alsa库，-L指定具体的alsa-utils-x.x.x类库的asound等，编译方法方法：

gcc -o local\_player local\_player.c -L XXX/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

## 4.1 音频处理模块设计与实现

### 降噪

Aec

### 增强

Wpe

## 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，具体的场景如 Android 手机里的 “OK, Google” 或者苹果设备里的 “Hey, Siri”。  
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服务（语音识别组件），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（即唤醒词或“热词”），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吧，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（打招呼）。

[snowboy](https://links.jianshu.com/go?to=https://snowboy.kitt.ai/" \t "_blank)是一个开源的、轻量级语音唤醒引擎，可以通过它很轻松地创建属于自己的类似“hey, Siri” 的唤醒词。它的主要特性如下：

* 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

详细看了官网提供的安装配置教程（已经很久没更新，有点过于繁琐了），几番尝试之后，整理成下面的使用方法，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

编译源代码

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**）

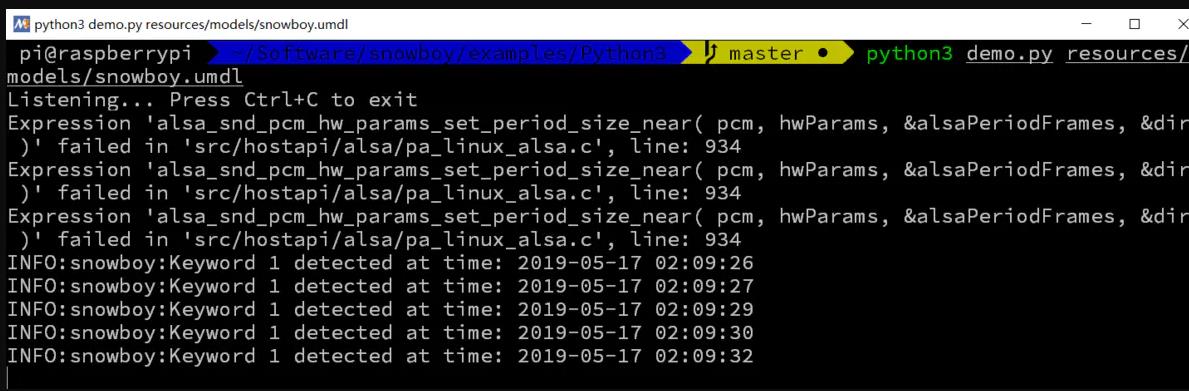
然后对着麦克风清晰地讲出**“snowboy”**，如果可以听到“滴”的声音，则安装配置成功。命令行输出如下：

snowboy test

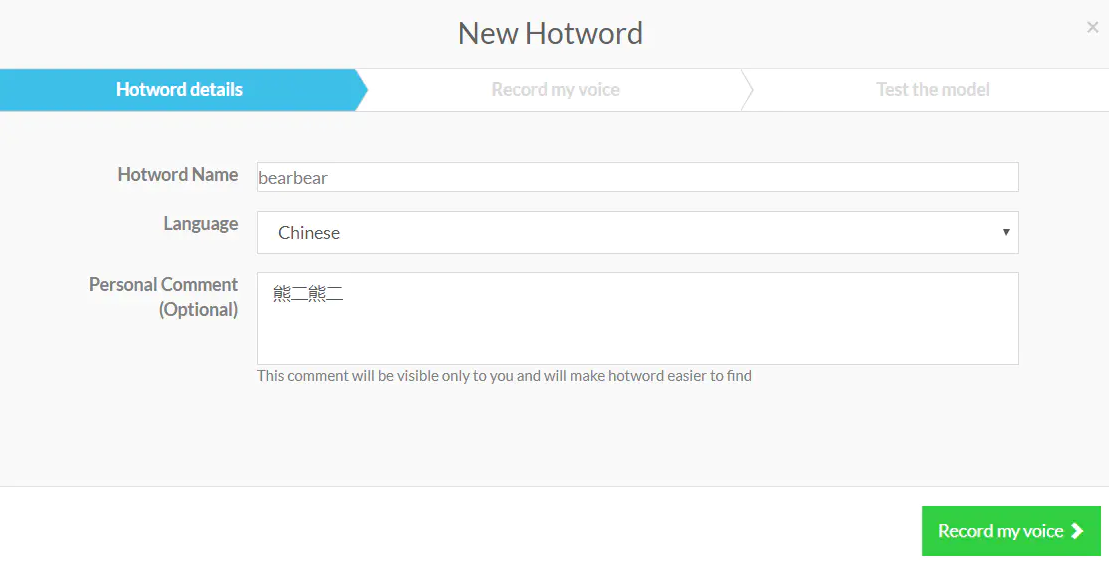
**PS**：官方源代码使用 Python3 测试有报错，经测试需修改 snowboy/examples/Python3 目录下的 snowboydecoder.py 文件。  
将第 5 行代码 from \* import snowboydetect 改为 import snowboydetect 即可直接运行。

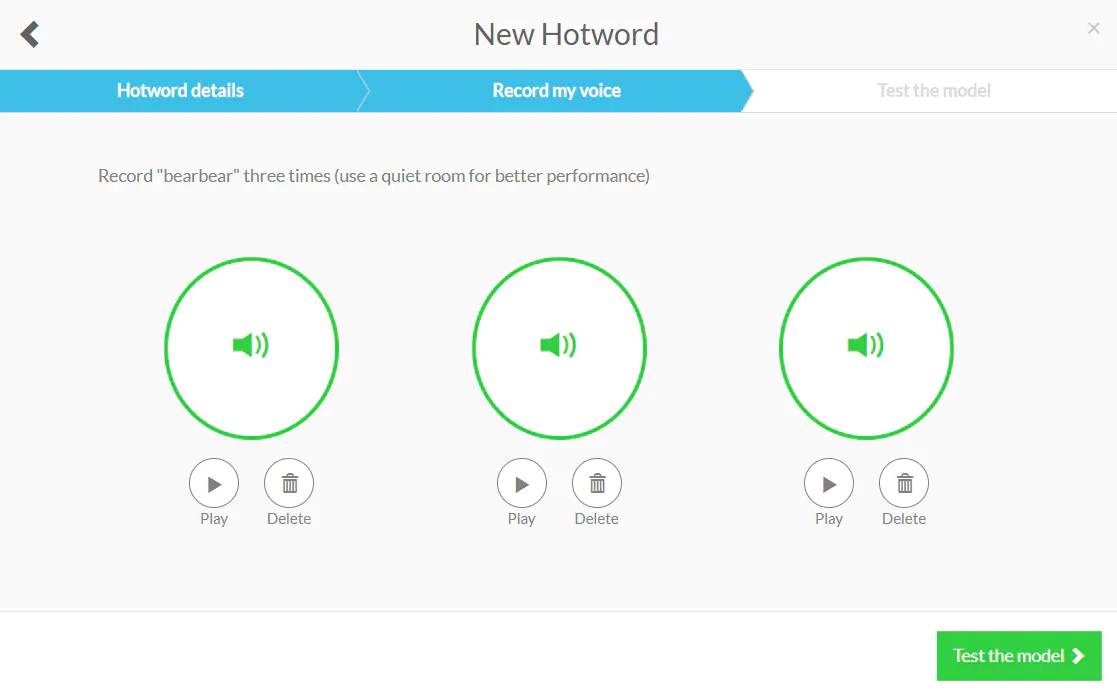
### 2、设置自己的唤醒词

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**。我试过直接在线录制，有一些问题



record





upload

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：

import snowboydecoder

import sysimport signal

interrupted = False

def interrupt\_callback():

global interrupted

return interrupted

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)

model = sys.argv[1]

# capture SIGINT signal, e.g., Ctrl+C

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

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

# main loop

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

interrupt\_check=interrupt\_callback,

sleep\_time=0.03)

detector.terminate()

通过阅读代码，可以看出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，可改变程序最终的响应。如：

import snowboydecoder

import sysimport signal

interrupted = False

def signal\_handler(signal, frame):

global interrupted

interrupted = True

def interrupt\_callback():

global interrupted

return interrupted

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)

model = sys.argv[1]

# capture SIGINT signal, e.g., Ctrl+C

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

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

# main loop

detector.start(detected\_callback=detected,

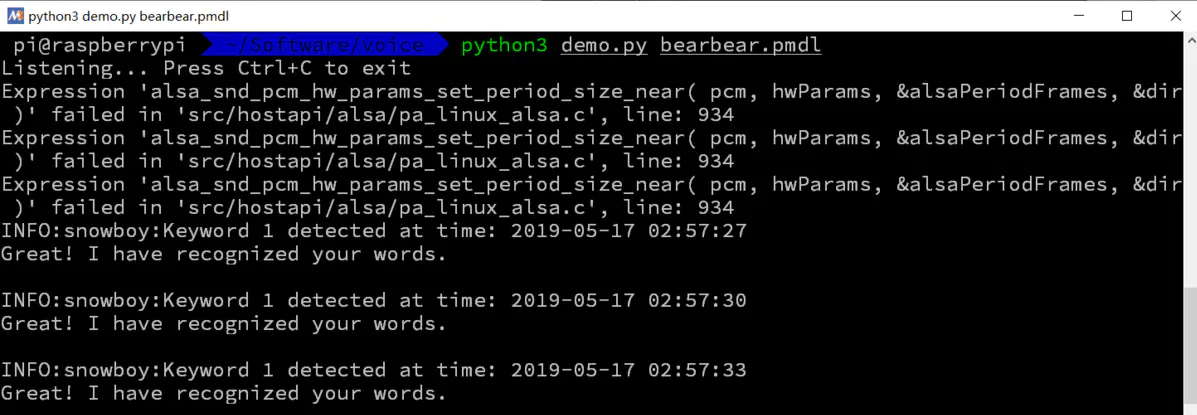
interrupt\_check=interrupt\_callback,

sleep\_time=0.03)

detector.terminate()

**注意**添加的 detected 函数。

效果如下：



personal behavior

更复杂的应用形式（如控制 LED 小灯等）也是基本上一样的思路。

## 4.2 语音识别模块设计与实现

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**

“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。

2）cmake（提示该方式未完整测试，不建议，故采用1）

## 4.3网络检测模块设计与实现

4.4 离在线识别

## 4.4 关键技术和解决方案

## 4.5 本章小结

# 第五章 系统测试分析

## 5.1 测试概述

## 5.2 测试工具及测试环境

## 5.3 测试方法及流程

## 5.4 系统功能测试

## 5.5 系统运行效果评估

## 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

## 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:

### 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。

### 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。

### 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。

### 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。

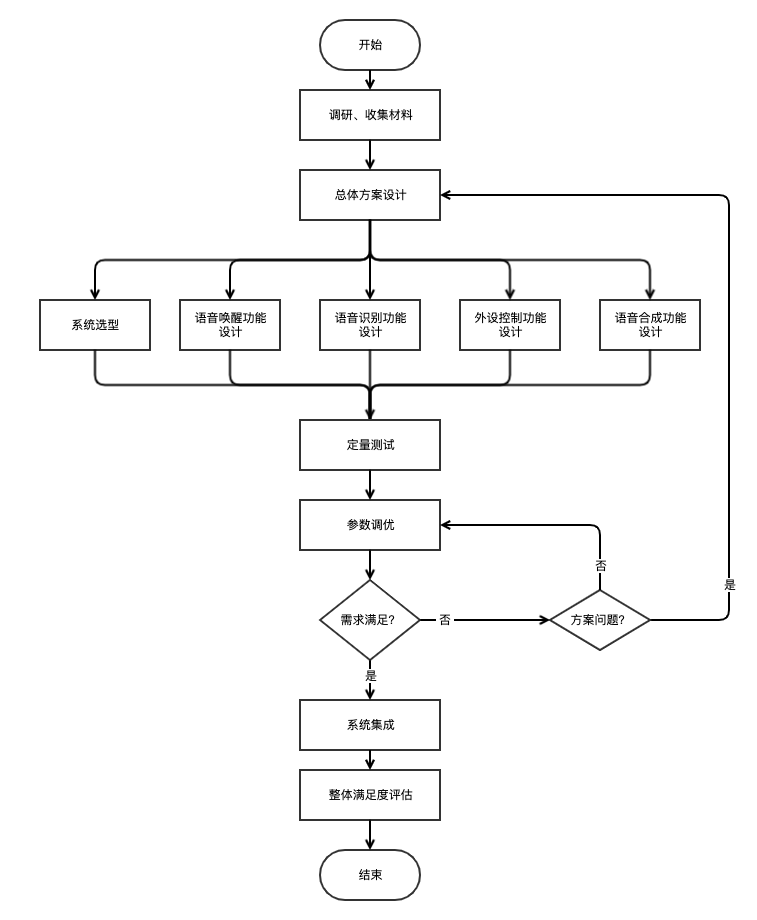
## 研究方法与技术线路

基于百度现有音箱方案对比市场主流音箱，如百度的小度音箱，小米的小爱同学，阿里的天猫精灵等选择可支撑在嵌入式系统上实现唤醒和识别算法的能力芯片。由于当前主流音箱普遍性能偏低，以上现有方案只能作为参考，需要在彻底了解后进行性能评估。

论文前期主要通过文献调查法通过大量阅读人工智能相关材料和文献，结合比较研究法进行初期系统结构构造，根据研究目标形成理论基础，从而全面地、正确地了解掌握智能语音控制系统各个模块所需知识点并形成一个统一的整体，为后续高效准确完成目标打下基础。

论文中期主要通过统计分析法对唤醒，识别和语音合成模块进行量化调优，并通过模拟法在功能和性能上对比市面已有方案，进行对比研究不断改善升级

技术路线如下图3-2:



**图3-2** **研究方法与技术路线**

# 关键技术难点和技术解决方案

# 预期成果及可能的创新点

通过本系统的研究，预期达到的目标如下：

1. 完整系统可通过语音控制智能设备，如开关，家电，窗帘，汽车空调等。
2. 支持离线语音唤醒、离线语音识别和离线TTS功能
3. 可根据实际需求出具具体的系统解决方案等。

# 芯片选型和系统搭建

Wifi 蓝牙 红外

性能 存储

1. 系统选型

驱动 开发环境

1. 开发板

# 降噪

# 语音唤醒

# 语音识别

# 控制

# 测试与效果评估

# 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