**编 号：**

**审定成绩：**

|  |  |  |  |
| --- | --- | --- | --- |
| **重庆邮电大学**  **毕业设计（论文）** | | | |
|  | | | |
|  | **中文题目** | **基于智能语音识别与分析的情绪管理系统** |  |
|  |  |
| **英文题目** | **Emotion Management System Based on** |
|  | **Intelligent Speech Recognition and** |
|  | **Analysis** |
| **学院名称** | **自动化学院/工业互联网学院** |
| **学生姓名** | **朱孝虎** |
| **专 业** | **物联网工程** |
| **班 级** | **08051701** |
| **学 号** | **2017212965** |
| **指导教师** | **邓钦元 讲师** |
| **答 辩 组**  **负 责 人** | **张开碧 副教授** |
|  | |

**2021年6月**

**重庆邮电大学教务处制**

自动化学院/工业互联网学院本科毕业设计(论文)诚信承诺书

本人郑重承诺：

我向学院呈交的论文《基于智能语音识别与分析的情绪管理系统》，是本人在指导教师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明并致谢。本人完全意识到本声明的法律结果由本人承担。

年级

专业

班级

承诺人签名

年 月 日

学位论文版权使用授权书

本人完全了解重庆邮电大学有权保留、使用学位论文纸质版和电子版的规定，即学校有权向国家有关部门或机构送交论文，允许论文被查阅和借阅等。本人授权重庆邮电大学可以公布本学位论文的全部或部分内容，可编入有关数据库或信息系统进行检索、分析或评价，可以采用影印、缩印、扫描或拷贝等复制手段保存、汇编本学位论文。

（注：保密的学位论文在解密后适用本授权书。）

|  |  |
| --- | --- |
| 学生签名： | 指导老师签名： |
| 日期： 年 月 日 | 日期： 年 月 日 |

摘要

话语作为人们交流表达最直接的方式，其中蕴藏了大量的信息，能体现出你实时情感，当前任务情况以及未来要做的事等等。当前社会上流通的只有功能单一的录音机，只能单机的进行语音的收集和简单的存储，并没有一个能对这些数据进行合理的存储和分析并得到有价值东西的系统。如果能把这些数据收集起来并做好相应的分析，既能增强对自己的了解，也能对未来做出更好的选择。同时这个系统也算是在收集分析记录数据做出的尝试，为了收集随时随地产生的语音信息，就需要建立一个高效，准确的语音分析系统。

本系统主要实现的功能是：能对用户的语音数据进行采集，并采用一定的方式方法分析语音数据得到对用户有用的信息，并根据这些信息做出相应的行为，使用户能增强对自身的理解，及时的调整情绪。经过对本系统的需求分析，最终将系统分为三个部分，分别为：语音采集传输感知层，数据分析通信服务端，数据展示客户端。语音采集传输感知层基于STM32F103ZET6芯片，利用VS1053进行语音数据的采集，并将文件保存在板载SD卡上，通过串口控制ESP8266WIFI无线通信模块将语音数据上传到服务端，及其它通信操作。数据分析通信服务端主要完成多线程接收请求，和感知层的通信:语音文件的接收，WAV语音重构。数据的分析:在完成参数自适应后利用讯飞API完成语音到字符串的转换，和关键词采集，情感分析等分析,并将数据按照本系统文件管理管理体系存储。客户端请求的处理：按照定义好的本应用通信协议响应客户端的请求。数据展示客户端主要实现关键词词云，心情图表，智能推荐，今日总结四大功能。

在完成设计之后对成品的功能进行了测试，实现了预定功能，达到了预期的效果。

**关键词：**WAV语音重构，参数自适应，词云，多线程

**Abstract**

Discourse, as the most direct way for people to communicate and express, contains a lot of information, which can reflect your real-time emotions, current task situation and things to do in the future, etc.At present, there is only a single-function recorder in circulation in the society, which can only collect and simply store voice alone, and there is no system that can reasonably store and analyze these data and get valuable things.If you can collect these data and do the corresponding analysis, you can not only enhance your understanding of yourself, but also make better choices for the future.At the same time, this system can be regarded as an attempt to collect and analyze recorded data. In order to collect voice information generated anytime and anywhere, it is necessary to establish an efficient and accurate voice analysis system.

The main function of this system is: it can collect the user's voice data, and use a certain method to analyze the voice data to obtain useful information for the user, and make corresponding behaviors based on this information, so that the user can enhance their own Understand and adjust emotions in time.After analyzing the needs of this system, the system is finally divided into three parts: voice collection and transmission perception layer, data analysis communication server, and data display client.The voice acquisition and transmission perception layer is based on the STM32F103ZET6 chip, uses VS1053 to collect voice data, and saves the files on the onboard SD card, controls the ESP8266WIFI wireless communication module through the serial port to upload voice data to the server, and other communication operations.The data analysis communication server mainly completes the multi-threaded receiving request, the communication with the perception layer: the receiving of voice files, and the reconstruction of WAV voice.Data analysis: After completing the parameter adaptation, use the Xunfei API to complete the conversion of speech to string, and analyze the keyword collection, sentiment analysis, etc., and store the data according to the file management system of this system.Client request processing: Respond to client requests according to the defined communication protocol of this application.The data display client mainly implements keyword word cloud, mood chart, intelligent recommendation, and four functions of today's summary.

After completing the design, the function of the finished product was tested, the predetermined function was realized, and the expected effect was achieved.

**Keywords**: Wav Speech Reconstruction,Parameter Adaptation,Word Cloud, Multithreading

目录

[第1章 引言 1](#_Toc19533)

[1.1 研究背景和意义 1](#_Toc3917)

[1.2 国内外研究现状 2](#_Toc21635)

[1.2.1 国外研究现状 2](#_Toc17522)

[1.2.2 国内研究现状 3](#_Toc14176)

[1.3 主要内容和工作安排 4](#_Toc21014)

[第2章 系统方案论证 5](#_Toc13416)

[2.1 主控芯片选择 5](#_Toc3908)

[2.1.1 单片机技术简介 5](#_Toc28074)

[2.1.2 单片机性能对比及选择 5](#_Toc26685)

[2.2 网络通信技术选择 6](#_Toc20258)

[2.3 音频采集芯片介绍 7](#_Toc3778)

[2.4 显示屏技术论证 7](#_Toc19046)

[2.5 本章小结 8](#_Toc24994)

[第3章 系统总体设计 9](#_Toc20583)

[3.1 功能需求分析 9](#_Toc28128)

[3.2 总体设计 10](#_Toc652)

[3.2.1 感知层总体设计 10](#_Toc15856)

[3.2.2 服务端总体设计 12](#_Toc7550)

[3.2.3 客户端总体设计 14](#_Toc4050)

[3.3 本章小结 15](#_Toc29520)

[第4章 硬件设计 16](#_Toc4376)

[4.1 核心电路设计 16](#_Toc14315)

[4.1.1 主控芯片电路设计 16](#_Toc30751)

[4.1.2 LCD电路设计 17](#_Toc14827)

[4.1.3 复位电路设计 17](#_Toc19454)

[4.1.4 按键电路设计 18](#_Toc13134)

[4.2 VS1053模块 18](#_Toc30967)

[4.3 ESP8266模块 19](#_Toc30879)

[4.4 本章小结 20](#_Toc32113)

[第5章 软件设计 21](#_Toc17033)

[5.1 感知层软件设计 21](#_Toc7110)

[5.1.1 单片机主函数程序设计 21](#_Toc27893)

[5.1.2 ESP8266连接功能程序设计 22](#_Toc8033)

[5.1.3 WAV定长音频录制功能程序设计 23](#_Toc8225)

[5.1.4 音频文件上传更新功能程序设计 23](#_Toc30729)

[5.2 服务端程序设计 25](#_Toc15178)

[5.2.1 服务端整体程序设计 25](#_Toc17484)

[5.2.2 WAV音频文件的重构 27](#_Toc21991)

[5.3 客户端程序设计 29](#_Toc27745)

[5.4 本章小结 33](#_Toc5001)

[第6章 系统功能实现及测试 34](#_Toc14586)

[6.1 感知层功能实现与测试 34](#_Toc10199)

[6.2 服务端功能实现与测试 36](#_Toc28133)

[6.3 客户端功能实现与测试 39](#_Toc4922)

[6.4 PCM转字符的性能分析及参数自适应 43](#_Toc28266)

[6.5 系统整体测试 46](#_Toc9792)

[6.6 本章小结 49](#_Toc25021)

[第7章 总结与展望 50](#_Toc2236)

[7.1 主要工作 50](#_Toc16396)

[7.2 后续研究工作展望 52](#_Toc11045)

[参考文献 54](#_Toc13549)

[致谢 56](#_Toc31694)

[附录 57](#_Toc27185)

[一、英语原文 57](#_Toc18147)

[二、英文翻译 61](#_Toc9838)

[三、源程序 65](#_Toc8021)

# 

# 第1章 引言

本章主要介绍基于智能语音识别与分析的情绪管理系统设计的研究背景和意义。进行对国外和国内语音识别学术研究和商业应用的现状和分析比对，并进行了总结概括。并且大体总结了本系统的主要内容，采用对功能进行分模块实现的策略，最后对后期的工作进行了相应的安排。

## 1.1 研究背景和意义

语音是人们进行信息表达的重要方式之一，这个功能是我们作为人类最基本的特点之一，语言是人类区别于其它动植物的重要标准之一。人类获取信息的方法多种多样，人们可以通过视觉，听觉，还有文本来进行信息的传递。但最重要,最精细的信息源只有语言,图像和文字三种[1]。在这三种途径的对比中，图像和文字传输的效率比通过语音传播信息的效率要低不少，因为语音中除了说话者表达的表面内容外，还含有声音强度，音色，甚至情绪状态等信息，语音是人类最重要,最有效,最常用和最方便的交换信息的形式[2]。它蕴含了大量可以被挖掘处理的信息。

另一方面，语音和人类的社会活动，文化发展息息相关，它蕴含了巨大的信息潜力，从语音中能很好分析出说话者当时的情感。人类是一种感性动物，丰富的情感是我们重要的特点之一。情绪是我们“生命的指挥棒”“健康的寒暑表”[3]。情感是我们非常客观的一种感受，能体现我们最真实的状态，但情绪就像大染缸一样，有黑有白，有好有坏，好的情绪能让我们非常开心，利于人类的发展，但消极的情绪如果不及时发现和控制，将会对人们的生产和生活带来灾难性的打击，这时我们应该对情绪反应进行评估和具体动作的挖掘和管理，情绪管理可以帮我们做到这一点，所以情绪管理对我们来说是非常重要的[4]。

情绪管理善于自我控制，善于调整情绪，解决生活中的矛盾和事件，随着社会主义经济的快速发展和我国金融实力的不断发展，生产力水平将呈现上升趋势现代信息技术已进入信息化建设时代[5-6]。在满足人们物质需求的同时，人们也应该着重精神方面的追求。在互联网高速发展的背景下，新媒体，社交软件等信息层出不穷，人们在日常生活中接触到的信息量呈爆炸式增长，在信息时代，人们陷入了使用信息的陷阱，相反，他们迷失在海量的信息中，对自己的认识变得低下，在某些时候，我们感到焦虑和困惑，我们的身体开始出现问题，我们的睡眠质量开始下降，最终我们的身心在当前环境变得更糟，但我们不能任其恶化而不做任何改变，精神是人类生活的支柱，只有人们有了积极的精神王国，才能保持良好的状态，更好地促进经济发展。经济的快速发展促进了人们对精神的向往，积极乐观的态度也促进了经济的快速发展，因此对人们来说是极其重要的，提高对自己的理解，调整自己的情绪，所以使人增强对自己的了解，调整自己的情绪变得极其重要。

## 1.2 国内外研究现状

### 1.2.1 国外研究现状

语言识别研究始于20世纪50年代，当Hidden Markov模型中成功地用于语音识别时标志着语音识别走向进一步深入[7-8]。最近几十年的语音识别发展过程中，分为三个阶段，英语语言的语音识别技术已经有不少的经验较为成熟，能够在连续语音和音素识别方面达到高水平。随着隐马尔可夫模型的多次迭代更新，它已经变成可以建立高性能系统的有力工具[9]。

在商用语音识别中，Nuance公司是这个领域的开拓者，目前Nuance公司识别引擎技术被全球大约五分之四的语音识别技术使用，能对五十多种语言进行识别，因为语言支持多，覆盖了航空，金融等各个领域，用户量也达到了巨大的20亿。在专利数上面，它也是全球拥有语音识别专利最多的公司，连顶级的银行和大部分的航空公司都以Nuance公司的语音识别引擎作为呼叫系统的底层实现。此外苹果公司也在语音识别及应用方面拥有一定技术成果，如ios系统中的Siri，2010年苹果收购siriinc公司后，在2011年发布的iphone4s搭载了此技术。但刚开始用户都觉得这个产品不好，用户体验极差，但随着苹果公司收购Novaris technologies语音识别公司，vocaliq公司之后。苹果拥有了对整个句子进行识别，并且能用AI与用户自然的交流技术，并且该公司还使用了机器学习开发虚拟助理，将其技术应用于便携式设备和家庭网络中。

总体来说，国外在语言识别方面的研究还没有一款即有语音识别和分析又有根据分析结果来做出智能行为，用来帮助用户了解自己本身并帮助用户来调节其情绪的系统。

### 1.2.2 国内研究现状

近几年来，语音技术发展迅猛，国内的语音技术研究基本同步。科大讯飞，捷通华声，中科信利等语音企业相继成立[10]。

在商业发展中，国内在语音识别上的研究与国外发达国家相比相对滞后[11]。也有许多企业注意到了这方面的滞后情况，进而投入研究。其中的科大讯飞就是目前语音识别国内的龙头，在国内拥有最大的语音识别影响力。科大讯飞依靠着中国科技大学的语音处理技术于1999年成立，在国家的大力支持和自身不断的努力下，在2008年以500多亿的市值上市。并且各省的口试系统都使用讯飞语音识别引擎作为基础，在2014年的市场调查报告中显示，讯飞市场份额达到60%，是中国语音识别领域无可争辩的霸主。此外百度也注意到了语言识别发展的重要性，在2010年，它选择通过和中国科学院声学研究所合作的方式研发语音识别技术，而后在2014年，著名的人工智能大师吴恩达加入团队后，其语音识别技术才得到一定的发展，市场份额达到了13%左右[12]。其语音识别技术也在不断向科大讯飞靠近。

在智能推荐方面，许多手机厂商做了不少的尝试，包括华为，小米等手机都会有“负一屏”这个功能[13]。这个功能主要是以用户在手机上的行为作为判断基础，猜测用户可能需要的东西。

总体来说，中国企业的科学家在这一技术领域保留了大量的专利。无论是核技术还是下游技术，我国申请人的专利申请数量都比较丰富，其次，我国语音识别技术的应用已经渗透和延伸到各个行业的应用范围。与早期发展的语音识别技术不同，现在的语音识别技术通过与不同领域的应用工具相结合，加快不同行业的发展速度，推动自动化和智能化进程，增强了这一领域的用户体验和智能化[14]。不管是理论学术的研究还是具体的商用项目，国内的情况都已经很接近国外先进水平。

## 1.3 主要内容和工作安排

本文需解决的实际问题为对基于智能语音识别与分析的情绪管理系统进行设计，将基于功能要求，主要包括感知层语音采集转存与发送、服务端数据的接收与分析，客户终端展示与设置三个部分。

结合单片机，无线通信，音频采集，语音识别相关原理，对三个部分进行可行的技术设计方案，分开模块设计，择取当中最优的组合方案，总体设计完毕后，开始硬件选型及搭建，进行软件开发，并说明相关功能实现逻辑，调试情况，目标功能，最终完成设计。本文包括7个章节，内容主体结构安排如下：

第1章为引言，通过查阅参考文献以及考察调研，来对本设计相关技术的国内外研究背景、研究目的和意义进行详细说明解释。

第2章是系统方案论证，根据课题需要以及考虑实际情况如成本等，对通信技术、硬件选型等进行论证，最终确定采用何种方法来完成本次设计。

第3章是系统总体方案设计，根据考核标准，在整理课题需求基础上，选择相应的技术、完成拓扑结构设计，阐述各模块之间逻辑，完成总体方案。

第4章是硬件设计，选择或选择硬件部件之后，通过查阅相关使用手册，参考文献资料或上网了解相关模块的应用方式，以此为基础设计硬件电路。

第5章是软件设计，查阅网上相关学习资料，结合基础程序的应用进行软件的编写。

第6章是系统功能实现及性能测试，在完成了硬件搭建与电路设计与制作、之后，对整个基于智能语音识别与分析的情绪管理系统进行功能功能实现方案详解与性能分析测试。

第7章作为论文的结束语，总结毕业设计工作，提出可以在今后继续深入研究的方向和改进，并对以后研究进展进行了畅想。

# 第2章 系统方案论证

本章在研究并确定系统的整体设计方案后，根据每个功能模块进行硬件的选型和分析，主要完成了主控芯片的选择，通信技术的选择，音频采集芯片的选择，显示屏技术的选择，以及每个选择的优缺点分析。

## 2.1 主控芯片选择

### 2.1.1 单片机技术简介

单片机是精简版的电脑，它主要包括CPU，内部存储有些还能拓展外部储存，并且芯片使用超大集成电路(VLSI)[15]。一般情况拥有多种功能，包括I/O的输入输出，中断行为的监听和相应，同时计数器也是单片机不可缺少的部分，主要是用来进行计数，产生定时终端等，单片机的的功能还包括AD数据的转换，PWM波的输出。目前通用的单片机速度已经高达300M，能满足大部分的需求，除了单片机核心部分，它还有很好的拓展性，使用者可以根据自己的需求增加相应的模块，包括音频采集，摄像头等模块。正因为单片机体积小，功能多，拓展性强等特点，广泛的应用与生活的方方面面[16]。

### 2.1.2 单片机性能对比及选择

目前市场上单片机的位数主要是8位，16位，32位，每款芯片都占有不少的市场份额[17]，具体数据如图2.1所示。

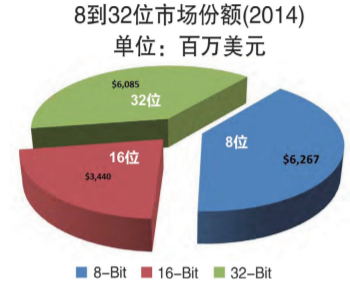


图2.1 单片机市场份额饼状图

在市场上广泛应用的微型控制器中，51系列和STM32系列单片机是目前流行的产品，其中51单片机性价比很高，51单片机从内部硬件到软件都有一套完整的位操作，位是51单片机的处理对象，它可以通过操控一些特殊的寄存器来实现特定的功能，比如定时寄存器TMOD可以用于设置定时器的模式，一般也有5个中断，内存也能达到256KB，同时在内存上开辟了双功能地址，再价格方面，常用的89C51价格才10人名币左右。虽然51单片机价格很低性能也不错，但它仍然有不少缺点给，它的引脚个数不是很多，想使用高级一点的功能需要扩展相应的外设，这不仅会增加硬件的负担，还会增加芯片的运算负担，51芯片保护能力弱是容易烧坏芯片的重要原因[18]。

相比51单片机，与51单片机不同的是，STM32虽然在价格上会比51单片机贵，但是在性能和功能上有了质的提升，STM32是基于ARM的Cortex-M内核，它性能提高了很多但能耗消耗却控制的很好，外设相比51单片机丰富，内部资源也有巨大优势，与计算机在很大程度上相似，能适用与当下众多的智能设备开发需求。

综合以上STM32和51单片机的优缺点，因为要进行大量的数据采集和上传操作，51单片机的速度和引脚已经无法满足功能需求，而STM32的高性能，低功耗，刚好符合这次设计的需求，再由于STM32103ZET6已经有很多成熟的设计结果，网上的资料也很丰富，所以最终选择STM32103ZET6作为主控芯片

## 2.2 网络通信技术选择

总的来说，有线通信和无线通信是两个主要的网络通信技术，有线通信必须有实物的连接，电缆的开通时必要条件，如果不安装线路就不能实现有线通信，反观无线通信，不需要实体线路的连接，少了很多约束，通信建立就比有线通信快。在经济投入方面，有线通信需要大量的埋线投入，并且还要花费金钱来维护这一些线路，遇到地形不合理，还会投入更多的资金来建立有线连接。而无线通信不需要实体线路的连接，就不用投入线路建立的资金，也不用花费资金维护线路[19]。

本次设计需要和服务端通信，而服务端程序放在服务器，所以不可能使用有线的方式进行通信，进而排除有线通信策略，现在分析无线的通信协议的选择，当前常用的无线协议有蓝牙，WIFI，Zigbee，UWB，NFC等。下面例出了几个关键参数的比较，如图2.1所示。



图2.1 常用无线通信协议比较

大范围Wi-Fi的普及，使得无线通信市场成为了Wi-Fi技术的主场，再考虑到本系统用户就是普通的每个人，手机也能释放WIFI信号，使得WIFI通信十分满足此系统的需求因此选择WIFI技术作为设计基于智能语音识别与分析的情绪管理系统的无线通信方式十分符合预期。

## 2.3 音频采集芯片介绍

本系统的音频采集基于VS1053芯片，VS1053芯片是市面上占有比比较大的音频解码芯片，它的解码性能不仅高，而且支持的音频格式也非常多，包括MP3格式，OGG格式，WMA格式，FLAC格式，WAV格式，MIDI格式，AAC格式等。VS1053芯片的DSP处理器内核性能十分卓越，不仅拥有0.5K的数据RAM而且也还有16K的指令RAM。在控制方面VS1053芯片由SPI（Serial Peripheral Interface）通信控制[20]。

## 2.4 显示屏技术论证

LCD（Liquid Crystal Display）和OLED（OrganicElectroluminesence Display）是两类不同的显示屏技术，它们的工作方式有着根本的区别。OLED是主动发光的原理。OLED又称为有机发光半导体。OLED是电流型的有机发光器件，它可以看作是一个OLED屏幕，每个像素是一个灯[21]。

与OLED不同，LCD通过背光LED/CCFL发光，也就是说，LCD发光相当于在快门后面放置几个大的灯。虽然OLED比LCD有很多优点，但由于成本高，大多数OLED被应用于先进的器件中。LCD不仅是目前最完善、最成熟的显示技术，而且由于价格、资质水平、市场价值、需求满足度等因素，也是商业市场上使用最多的屏幕显示技术，十分适合用于此设计中。

## 2.5 本章小结

本章首先说明了为了实现基于智能语音识别与分析的情绪管理系统设计的最终目标和设计需求所需要的相关技术。根据实际的需要和现实多方面的综合考量，选择最适合的技术或者硬件设备。阐述了无线通信技术当前的发展，对不同技术分析各自优缺点，最终确定以STM32103ZET6作为主控芯片，通过串口控制ESP8266WIFI模块实现无线通信，控制这以VS1053为核心的音频采集芯片，接着简述了显示屏技术中OLED和LCD的差异，选择适用于本次设计的方案。

# 系统总体设计

在总体方案确定的情况下，对系统具体的功能和需求进行分析，并确定每个功能模块的功能设计和交互方式整体分为三个模块，分别是感知层数据的采集和上传，服务端请求的接收与相应数据处理，最后是客户端数据的展示。总体设计框图如图3.1所示。

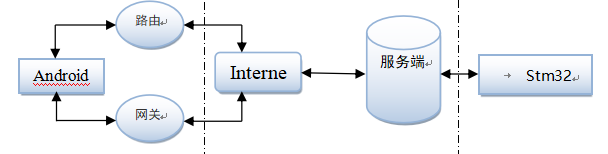


图 3.1 系统总体框图

## 3.1 功能需求分析

本系统分为三个大的模块：

1）感知层语音采集转存与发送处理：利用VS1053芯片采集WAV文件格式，

然后将wav文件存入SD卡，再利用esp8266连入服务端，将数据源源不断的传给服务器。

2）服务端数据的接收与分析：服务端数据的接收与分析是系统一个至关重要的环节，对整个系统的性能有着至关重要的影响。利用多线程监听单片机的连接，设置一套当前应用的专属传输层协议，方便数据的接收与传输，将接收到的wav格式进行语音识别（讯飞在线识别的相应API），识别出相应的文字，然后存入数据库或直接以文件形式存入服务器，调试相应的自然语言分析分析算法（关键字提取、情感分析、语义角色标注），得到当前情绪状况，当日说话量或更多信息，并且记录下来，发给客户终端展示。

3）客户终端展示与设置：

(1)心情图：通过提供的请求得到想要的时间内（日，周，月）的数据，然后通过好看line控件展示出来。

(2)词云：主要是通过请求得到词云，词云内容和图形可由客户端设置。生成部分有服务端处理。

(3)智能推荐：请求得到未来可能会做的关键字，然后关联到相关内容展示给用户。

(4)今日总结：将得的所有数据进行合理的展示包括但不限于（词频分析数据，词性分析数据，最值数据）展现形式为向上滑动的页面，背景为柔和的图片。

## 3.2 总体设计

### 3.2.1 感知层总体设计

1）声音采集模块：

(1)声音采集：音频数据采集采用VS1053芯片。主控芯片通过SPI控制VS1053芯片，用七条线将主控芯片与VS1053相连。主要是控制VS1053的复位，接收VS1053目前是否能接受数据的状态，最重要的是进行音频数据的采集。

(2)声音存储：芯片采集到的数据都是 PCM（Pulse Code Modulation），这是最基本的WAVE文件格式，要实现录波，首先需要了解波形文件格式，在熟悉WAV文件格式的情况下，严格按照相应的格式分割块，创建了一个wav文件，本系统用8G存储卡保存，在文件名方面本系统用时间戳来命名文件，适合以后处理。

2）WIFI通信模块：

(1)通信模块：本系统采用的时ESP8266作为通信芯片， 模块ATK-ESP8266利用串口与主控芯片通信，需要实现指定WIFI的连接，指定IP和端口的连接，字符串的发送这三个主要功能。本设计采用UART3连接WiFi模块进行指令配置和无线通信，实现TCP的安全传输。

3）核心母板：

(1)整体策略：声音模块，通信模块，sd卡模块，串口模块等都以一片stm32f10 为核心，开始时进行各个模块的初始化，包括串口，屏幕，延时和中断等模块，然后这个芯片进入主循环，等待按键的按下，然后检测哪个按键被按下，根据按键的不同分别执行音频文件的录制和音频文件的上传两个功能。负责所有的计算和控制具体流程如图3.2所示。

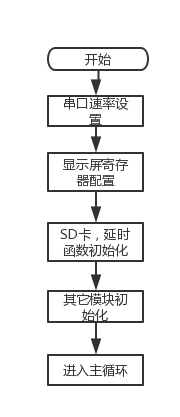


图 3.2 核心母板控制流程图

(2)无线通信的建立：通过串口三和8266进行通信，利用at指令实现服务器的连接和状态检查，并把连接信息输出在显示屏上。

(3)进入主要循环：录音模块开始工作，不断记录语音信息并存入sd卡，生成与时间相关的wav文件用户可通过按钮来直接控制文件的上传，上传的起始时间又存储在一个单独文件里，这样就不会造成文件的反复传输同时芯片还在不断检测各个模块的工作状态，如有问题及时自动处理并显示问题。

### 3.2.2 服务端总体设计

根据所分析的需求，单片机一直进行音频的采集，当需要上传数据时直接和服务端通过WIFI无线通信，此时服务端一直监听来自单片机或者客户端的请求，同时也另有线程执行关键词提取，智能捕捉，并按照本系统的应用协议进行通信。

大体结构都是，先发json字符串的命令码，然后再发具体的通信内容，字段及说明如图3.3所示。

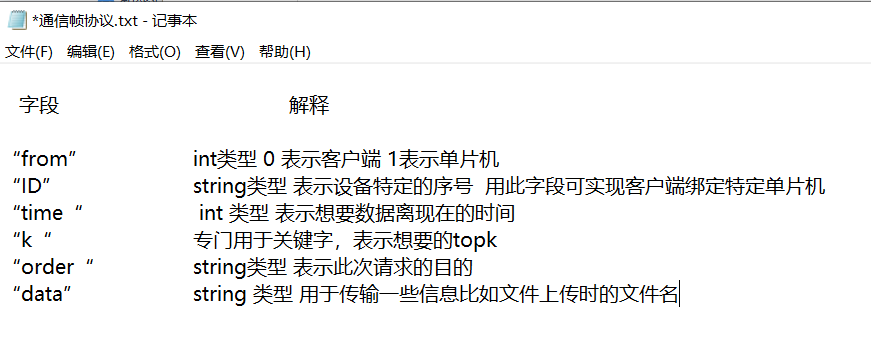


图 3.3 通信帧协议图

具体应用层协议如下：

1）{"from":0,"ID":"123","time":1614738576,"k":3,"order":"210"} 表示客户端向云端访问关键词，并返回json格式的字符串 {"文件":2.075,"录制":2.1,"音频":2.162}

2）{"from":0,"ID":"123","time":1614740484,"k":0,"order":"211"} 表示客户端向云端访问心情值，并返回json嵌套json格式的字符串{"0":{"zhi":"0","time":1614655520},"1":{"zhi":"-2115","time":1614655537},"2":{"zhi":"0","time":1614655554},"3":{"zhi":"0","time":1614655617},"linecount":4}，其中licount代表心情值条数。

3）{"from":0,"ID":"123","time":1614746611,"k":0,"order":"212"} 表示客户端向云端请求捕获的预测事情，返回帧格式（json字符串）：{"0":"电影","linecount":1}。

4）{"from":0,"ID":"123","time":1614751767,"k":0,"order":"213"} 表示客户端向云端请求其他统计数据（今日总结）返回帧格式：{"11":{"nd":1},"Freword":"电影","MaxJiange":120,"totalnu":112,"Frequence":6,"0":{"v":25},"1":{"n":13},"2":{"r":9},"3":{"wp":6},"linecount":12,"4":{"m":6},"5":{"nt":5},"6":{"q":4},"MaxDuohuaStart":111,"7":{"c":2},"8":{"nh":2},"9":{"p":1},"MaxJiangeStart":114,"MaxStr":"这是第1个测试文件，现在录制测试的音频。","10":{"u":1}}，Linecount键 代表词性总数 整型代表词性出现频率的排序，MaxJiange 代表最大的沉默时间，MaxJiangeStart 代表最长沉默时间的起始时间，Totalnu 代表总共说的词语数量，MaxDuohuaStart 代表最长句子的开始时间，MaxStr 代表最长话的内容 ，Freword 表示频率最高的词语，Frequence 表示最高频率的次数。

5）{"from":0,"ID":"123","time":1614738576,"k":100,"order":"214"} 表示客户端向云端访问词云，并返回图片流。

6）{"data":"电影","from":0,"ID":"123","time":0,"k":0,"order":"215"} 表示客户端向云端访问指定事情的推荐描述，服务端返回 ：百度电影吧，身边的影迷发烧友聚集地！??https://tieba.baidu.com/f?kw=%B5%E7%D3%B0&fr=ala0&tpl=5??百度电影吧。

7）{"data":"电影","from":0,"ID":"123","time":0,"k":0,"order":"216"} 表示客户端向云端访问指定事情的推荐图片，服务端返回相应图片。

8）{"data":"","from":0,"ID":"123","time":0,"k":0,"order":"217"} 表示客户端向云端访问指定句子，服务端返回:以前，总是以为自己丑，矮，衰，穷。而现在，慢慢的都朝着好的方向发展了:好丑，好矮，好衰，好穷！。

9）{"data":"","from":0,"ID":"123","time":0,"k":0,"order":"218"}表示客户端向云端访问小说类的名言，服务端返回： 有时你飞到了头，却发现还不如中间掉下来《球状闪电》。

10）{"data":"","from":0,"ID":"123","time":0,"k":0,"order":"219"}表示客户端向云端访问心情分析服务端返回： 相应的心情分析。

11）{"data":"","from":0,"ID":"123","time":0,"k":0,"order":"21A"} 表示客户端向云端访问最近词云背景图，服务端返回： 相应图片。

12）{“from”:1,order:”010”} 代表单片机向服务端请求时间戳，然后服务端直接返回一个数字字符串 “1620709417”。

13）{“from”:1,order:”011”} 代表单片机向服务端发送wav串口文件，服务端开始准备接收文件，并进行wav音频文件重构。

14）{“from”:1,order:”012”} 代表单片机向服务端询问当前的发送文件名，服务端返回应当发送的文件名。

### 3.2.3 客户端总体设计

客户端分为四大界面对数据进行展示。具体如下:

四大界面：

1）心情图表

要求：

(1)线尽量平滑连续柔和

(2)能够用图标显示这段时间的平均心情

(3)提供选项可以查看（日，周，月）视图 （利用下拉菜单切换）

2）词云

要求：

(1)提供选项可以查看（日，周，月）视图 （利用下拉菜单切换）

(2)最好能让词云的形状和当时心情挂钩

3）智能推荐

要求：

(1)根据待做的事情关键字进行联想推荐

(2)最好还能提供网址，比如购物网，百科之类的

(3)用户黑名单，用户可以左划选择这条消息的准确性进行反馈，然后更新推荐策略。

4）今日总结

要求：

(1)对所有的词语进行统计，输出各种图表（比如词性的统计饼状图 词汇量）输出像哪些软件的年度报告（最多的词汇，最密集的时段，最沉默的时段）根据位置数据（获取本机的定位看使用者轨迹推荐 运动之类的话术）输出相应的文字用图片配文字并采用优美的下滑效果。

## 3.3 本章小结

本章首先说明了基于智能语音识别与分析的情绪管理系统的功能需求，将感知层，服务端，客户端分别的作用进行解释说明，此后详细的列举了通信的应用层协议，最完成对系统的总体设计。

# 第4章 硬件设计

在确定整体设计和各个功能的需求之后，需要进行下一步的硬件设计和搭建，分别设计各个功能模块的电路原理图，分别为核心电路设计，音频采集电路，无线通信电路等电路的设计。完成硬件搭建才能进行软件开发，功能的实现。

## 4.1 核心电路设计

### 4.1.1 主控芯片电路设计

本设计所采用的芯片STM32F103ZET6，所需要用到有：USART3串口用于通信模块，SPI接口用于控制VS1053芯片采集音频数据[22]，用于存储音频文件的SD外部存储接口，LCD显示接口，按键控制接口等。

本设计所使用的单片机原理图如图4.1所示。

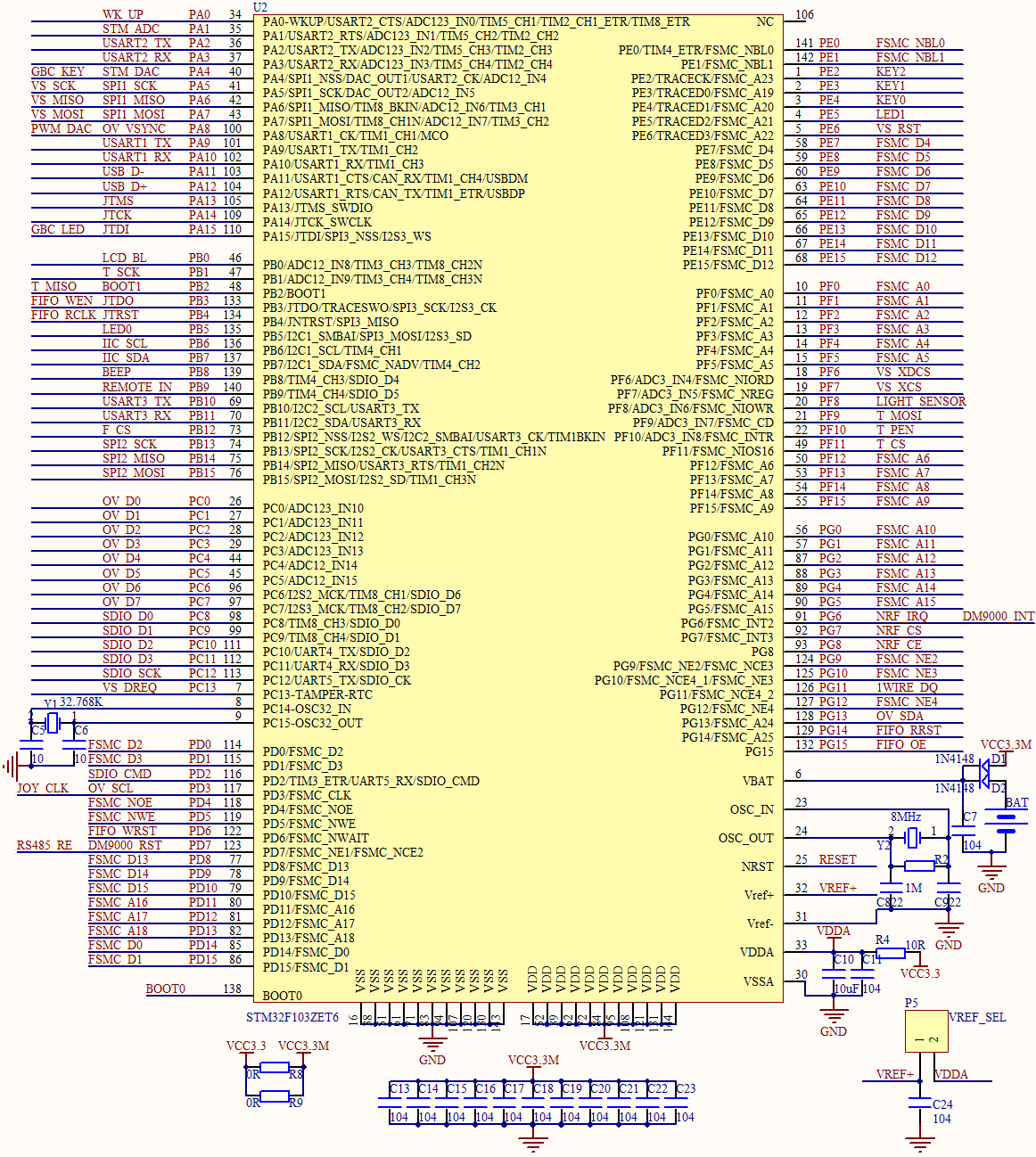


图4.1 STM32F103ZET6部分原理图

### 4.1.2 LCD电路设计

LCD最终用于显示提示信息，能够提示使用者当前单片机是处在空闲状态，还是工作状态，如果是工作状态还应该显示是在采集数据还是上传数据，并且操作的文件名也应该显示在屏幕上。在开发过程中LCD屏幕还要用于显示调试信息，方便代码正确实现。根据这些需求，最终确定的硬件原理图如图4.2所示。

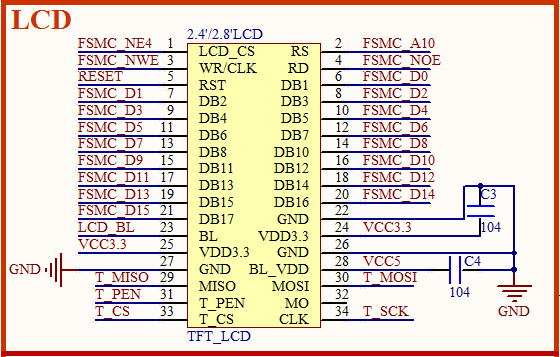


图4.2 LCD电路原理图

### 4.1.3 复位电路设计

因为主控芯片是低电平复位，所以本系统也以主控芯片为标准，采用低电平复位，R3和C12构成上电复位电路。在根据需求分析，在主控芯片复位的时候，LCD也应该进行复位，所以将LCD的复位触发也连接到MCU的复位线路上，使LCD和MCU能产生关联同时复位。电路原理图如图 4.3所示。

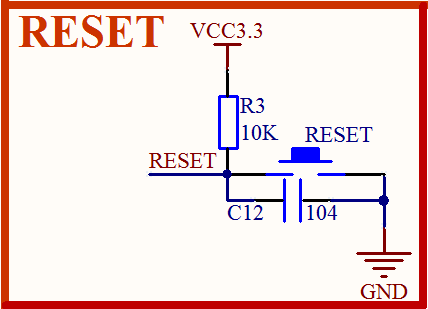


图4.3 复位电路原理图

### 4.1.4 按键电路设计

KEY0和KEY\_UP用于检测用户的按键操作分别控制音频的录制和音频文件的上传，KEY1和KEY2用作保留按键，用于开发时的调试与后期功能的拓展按键。这四个按键分别连在PE4，PA0，PE3，PE2即可。具体电路原理图如图 4.4所示。

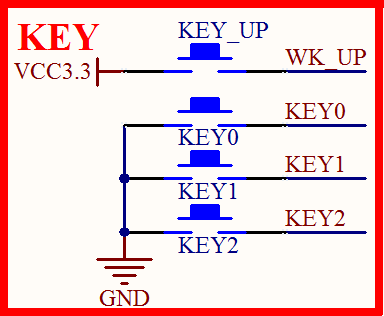


图4.4 按键电路原理图

## 4.2 VS1053模块

VS1053芯片是市面上占有比比较大的音频解码芯片，它的解码性能不仅高，而且支持的音频格式也非常多，包括MP3格式，OGG格式，WMA格式，FLAC格式，WAV格式，MIDI格式，AAC格式等。VS1053芯片的DSP处理器内核性能十分卓越，不仅拥有0.5K的数据RAM而且也还有16K的指令RAM。在控制方面VS1053芯片由SPI（Serial Peripheral Interface）通信控制。主控芯片通过SPI控制VS1053芯片，用七条线将主控芯片与VS1053相连[23]，本系统中VS1053模块原理图如图 4.5所示。

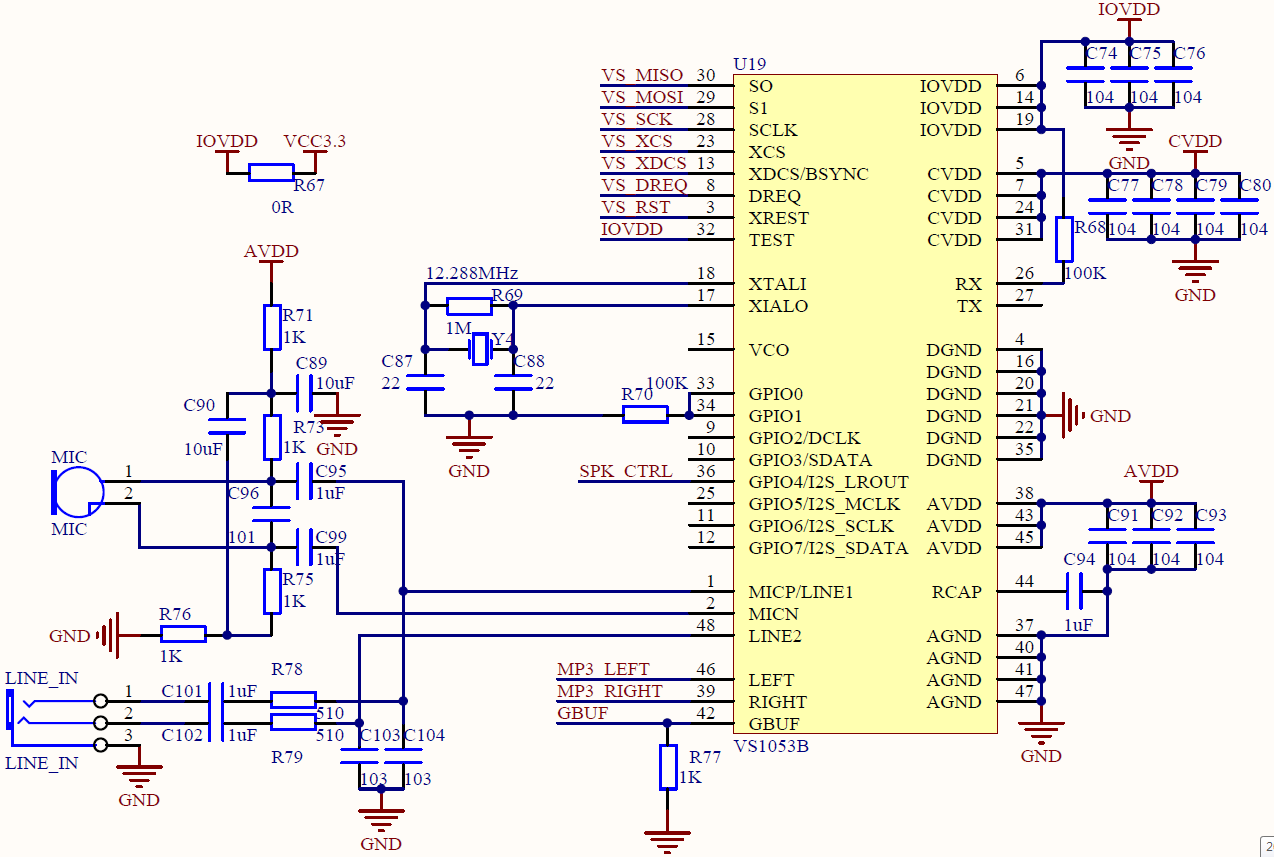


图4.5 音频解码模块原理图

## 4.3 ESP8266模块

ESP8266模块拥有六个引脚，分别是VCC，GND，RXD，TXD，KEY，LED。VCC和GND主要用于为整个模块供电，本系统将VCC连接到主板的5V接口，GND连接到主板的GND，RXD和TXD用于数据的通信[24]，采用串口通信，将它们和主板的USART3进行连接，KEY和LED分别用于按键检测和模块LED灯控制。接口原理图如图 4.6所示。

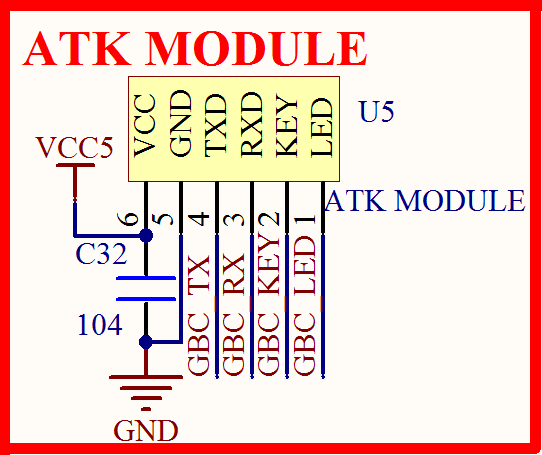


图 4.6 ESP8266接口原理图

整体实物图如图 4.7所示。

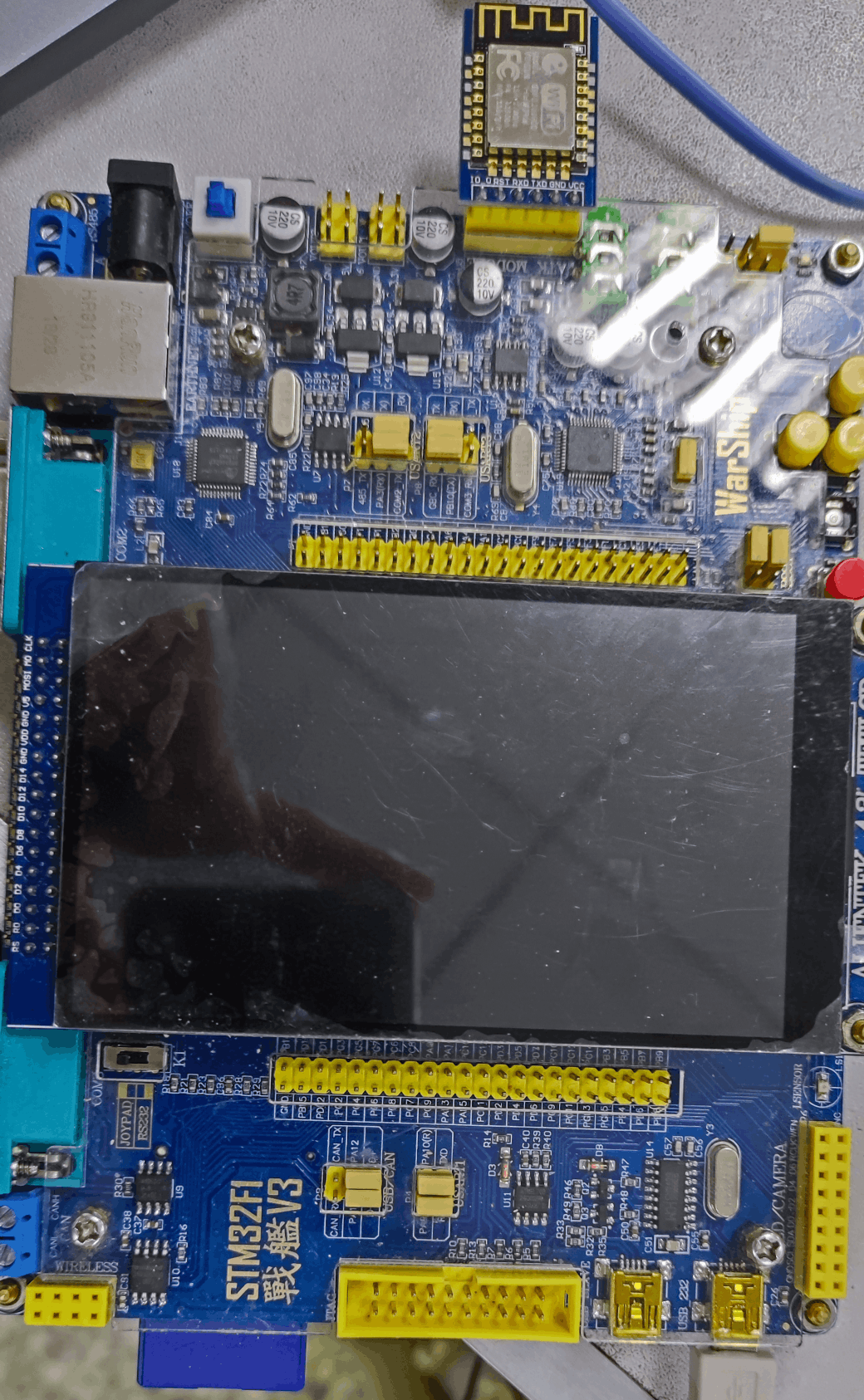


图 4.7 系统硬件实物图

## 4.4 本章小结

本章是本设计的硬件部分，主要分为三个部分，第一部分主要介绍了核心电路所使用的STM32的基本信息和内部结构，以及在本设计中控制芯片模块的电路设计。第二部分介绍了VS1035的基本结构及功能，VS1035的电路原理图。第三部分介绍了本设计所使用的无线通信模块的基本信息，以及整体的实物连接。

# 第5章 软件设计

本系统的软件设计由三部分组成，分别是感知层的音频采集与发送，服务端的通信与数据处理，客户端数据的请求与展示，以三个部分为主，分别完成每个部分的各个功能的设计，三个部分之间以特定的应用层协议进行通信。

## 5.1 感知层软件设计

### 5.1.1 单片机主函数程序设计

当系统通电之后，STM32F103ZET6开始工作，首先要对其进行初始化，包括延时函数初始化，串口初始化，LCD初始化，按键初始化，中断初始化，VS1052初始化，还有内部存储池初始化，接着进入WIFI的连接及服务端的连接，连接成功后进入一个死循环，一直检测按键按下情况，当按键0按下时，开始录制一个指定时长的声音wav文件，当检测到按键4按下时，开始进行wav文件上传，直到所有wav文件全部上传完毕。整体程序流程图如 图 5.1 所示。

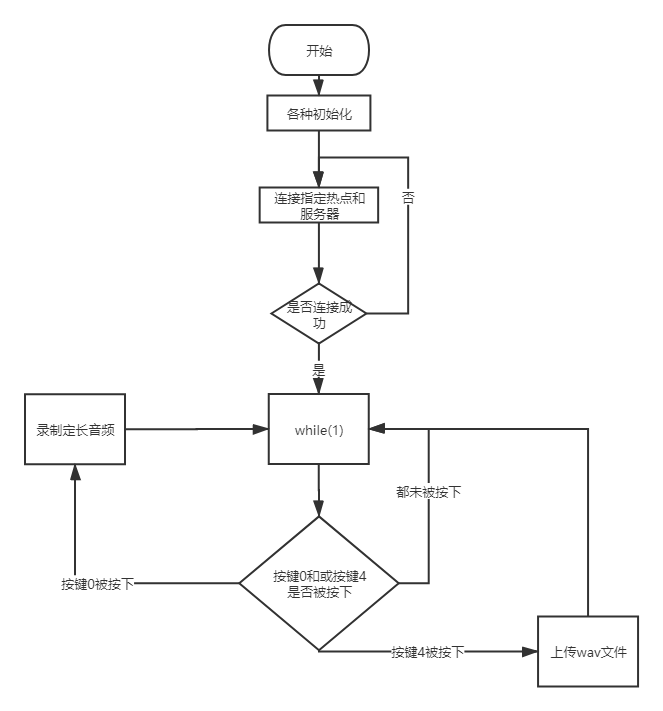


图5.1 感知层整体流程图

### 5.1.2 ESP8266连接功能程序设计

ESP8266WIFI主要实现指定WIFI的连接，并在此基础上连接到服务器建立同行，具体流程如下：先判断WIFI模块是否在线，不在线就一直循环检测，在检测到模块在线后，开始进行模式配置，将模块设置成sta模式并且重启，等待三秒重启完成后，连接指定的WIFI热点，并一直检测连接状态，直至连接成功，当连接成功后，又连接指定的IP地址和端口，并一直检测是否连接成功，直至连接成功。WIFI模块连接流程图如 图5.2 所示。

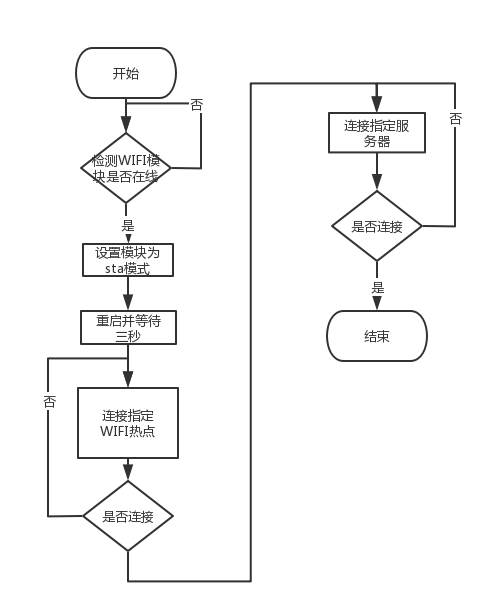


图 5.2 WIFI模块连接服务器流程图

### 5.1.3 WAV定长音频录制功能程序设计

先进行增益值的初始设置，然后打开指定的录音文件夹，如果打开失败就爆出错误，并将标志位设置成为 1，如果打开成功就进行各个数据结构的内存申请包括 WAV头文件变量，文件名变量，还有日志输出变量，这里出错也将标志位设置成1，进行标志位判断，如果过标志位为 1，那就退出程序，当标志位不为1时，检测流程控制位status是否为0，如果是向服务端申请当前时间戳，并将此时间戳作为要录制音频的文件名，然后将流程控制位status设置为 -1，继续检测status的值，当status等于-1时，模块都在进行音频的录制并写入相应文件，当时间超过设置的时间后，将status设置为1，当status为1时，更新WAV头文件的值，写入文件，完成音频的录制。程序流程图如 图 5.3 所示。

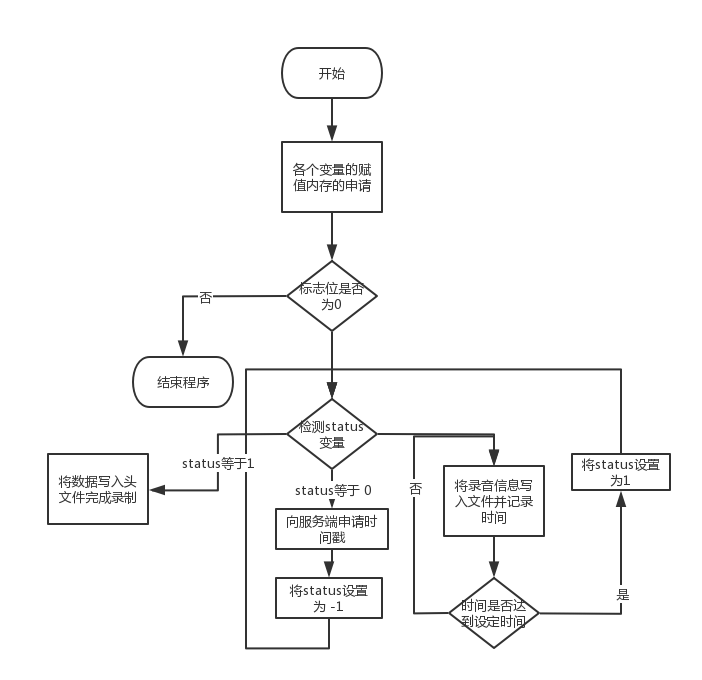


图 5.3 WAV定长音频录制程序流程图

### 5.1.4 音频文件上传更新功能程序设计

本系统设计让用户自动控制自己音频的上传时间点，而不强制自动上传用户的音频数据，当用户点击相应的按键，上传行为被触发，开始执行上传操作，具体流程如下：

程序先申请内存空间，用于后面数据的存储，然后向服务器申请待发送的文件名称，检测收到的文件名，如果文件名为 -1 ，表示文件已经全部上传完毕，没有待发送的文件，当文件名不为 -1时，发送相应的文件。流程图如 图 5.4所示。

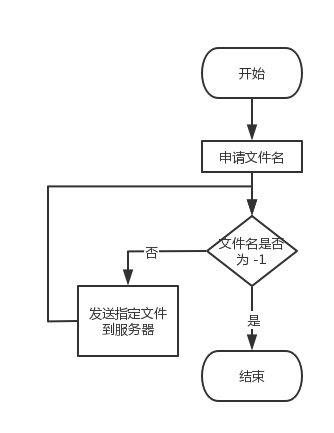


图 5.4 音频文件上传流程图

## 5.2 服务端程序设计

### 5.2.1 服务端整体程序设计

服务端的功能就是接收来自感知层和客户端的请求，并根据请求来进行数据的处理，并返回相应的值，整体功能如下。

服务端程序使用三个线程，用于并行处理三个不同的功能，包括请求的接收和处理，pcm文件到字符文件的转换，以及字符文件的智能处理。请求的接收和处理就是一个死循环监听来自客户端和感知层的所有连接请求，当收到连接请求后，开启一个新的线程专门处理这个请求，这样就可以实现同时处理多个请求，当收到请求信息时，得到解析出请求帧，并按照应用层协议进行相应的处理操作，具体请求帧和操作如下：

1）{"from":0,"ID":"123","time":1614738576,"k":3,"order":"210"} 表示客户端向云端访问关键词，并返回json格式的字符串 {"文件":2.075,"录制":2.1,"音频":2.162}

2）{"from":0,"ID":"123","time":1614740484,"k":0,"order":"211"} 表示客户端向云端访问心情值，并返回json嵌套json格式的字符串{"0":{"zhi":"0","time":1614655520},"1":{"zhi":"-2115","time":1614655537},"2":{"zhi":"0","time":1614655554},"3":{"zhi":"0","time":1614655617},"linecount":4}，其中licount代表心情值条数。

3）{"from":0,"ID":"123","time":1614746611,"k":0,"order":"212"} 表示客户端向云端请求捕获的预测事情，返回帧格式（json字符串）：{"0":"电影","linecount":1}。

4）{"from":0,"ID":"123","time":1614751767,"k":0,"order":"213"} 表示客户端向云端请求其他统计数据（今日总结）返回帧格式：{"11":{"nd":1},"Freword":"电影","MaxJiange":120,"totalnu":112,"Frequence":6,"0":{"v":25},"1":{"n":13},"2":{"r":9},"3":{"wp":6},"linecount":12,"4":{"m":6},"5":{"nt":5},"6":{"q":4},"MaxDuohuaStart":111,"7":{"c":2},"8":{"nh":2},"9":{"p":1},"MaxJiangeStart":114,"MaxStr":"这是第1个测试文件，现在录制测试的音频。","10":{"u":1}}，Linecount键 代表词性总数 整型代表词性出现频率的排序，MaxJiange 代表最大的沉默时间，MaxJiangeStart 代表最长沉默时间的起始时间，Totalnu 代表总共说的词语数量，MaxDuohuaStart 代表最长句子的开始时间，MaxStr 代表最长话的内容 ，Freword 表示频率最高的词语，Frequence 表示最高频率的次数。

5）{"from":0,"ID":"123","time":1614738576,"k":100,"order":"214"} 表示客户端向云端访问词云，并返回图片流。

6）{"data":"电影","from":0,"ID":"123","time":0,"k":0,"order":"215"} 表示客户端向云端访问指定事情的推荐描述，服务端返回 ：百度电影吧，身边的影迷发烧友聚集地！??https://tieba.baidu.com/f?kw=%B5%E7%D3%B0&fr=ala0&tpl=5??百度电影吧。

7）{"data":"电影","from":0,"ID":"123","time":0,"k":0,"order":"216"} 表示客户端向云端访问指定事情的推荐图片，服务端返回相应图片。

8）{"data":"","from":0,"ID":"123","time":0,"k":0,"order":"217"} 表示客户端向云端访问指定句子，服务端返回:以前，总是以为自己丑，矮，衰，穷。而现在，慢慢的都朝着好的方向发展了:好丑，好矮，好衰，好穷！。

9）{"data":"","from":0,"ID":"123","time":0,"k":0,"order":"218"}表示客户端向云端访问小说类的名言，服务端返回： 有时你飞到了头，却发现还不如中间掉下来《球状闪电》。

10）{"data":"","from":0,"ID":"123","time":0,"k":0,"order":"219"}表示客户端向云端访问心情分析服务端返回： 相应的心情分析。

11）{"data":"","from":0,"ID":"123","time":0,"k":0,"order":"21A"} 表示客户端向云端访问最近词云背景图，服务端返回： 相应图片。

12）{“from”:1,order:”010”} 代表单片机向服务端请求时间戳，然后服务端直接返回一个数字字符串 “1620709417”。

13）{“from”:1,order:”011”} 代表单片机向服务端发送wav串口文件，服务端开始准备接收文件，并进行wav音频文件重构。

14）{“from”:1,order:”012”} 代表单片机向服务端询问当前的发送文件名，服务端返回应当发送的文件名。

pcm转字符串线程就是不断地检测有没有新的pcm文件生成，当检测到新pcm文件生成，就将它转换成字符文件。

字符处理线程也时在不断检测有无新的字符文件生成，当检测到有新文件生成时，对新的字符文件进行智能事件捕捉，情感分析，关键字提取等操作，并把结果存入相应的文件，方便后续使用。服务端整体流程图如图 5.5所示。

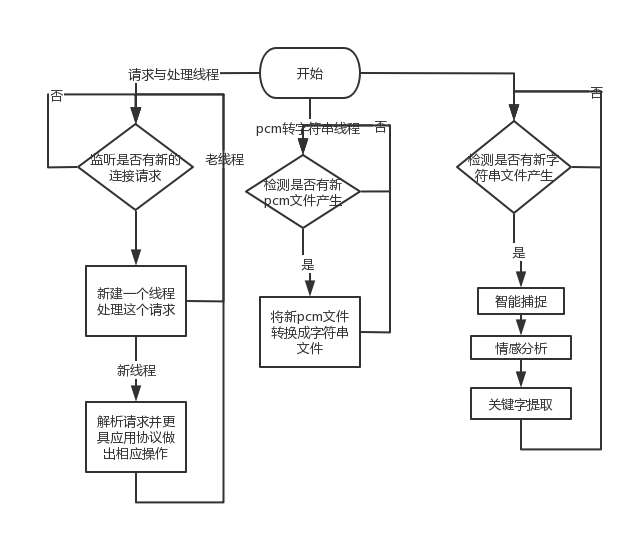


图 5.5 服务端整体程序流程图

### 5.2.2 WAV音频文件的重构

音频数据都是来源于感知层的上传，而感知层不支持 .wav 文件的传输，WIFI模块只支持字符串的发送，所以就必须将wav文件以字节的方式读出，再将字节以字符格式发送给服务端，服务端接收完毕后，文件只是一个字符串文件，应该将它重新构建成为 .wav 文件，按照发送时的处理，现在服务端应该将每个字符重新读出，转化为10进制的值，再将这个值以字节形式写入 .wav文件，完成wav音频文件的重构，程序流程图如图 5.6 所示。

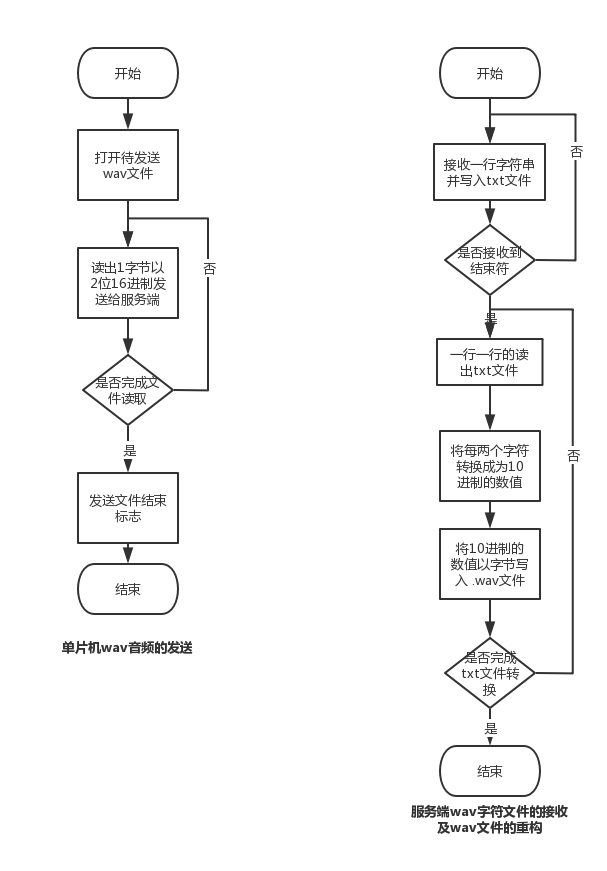


图 5.6 WAV文件重构流程图

## 5.3 客户端程序设计

客户端展示处理是移动客户终端完成的主要工作之一，是移动客户端的主体部分。其主要完成分析好的数据的接收、并且生成相应的报告和图表展示给用户。客户端一共有四个界面，分别是关键词词云，心情图表，智能推荐，今日总结。当进入客户端后，默认在关键词词云界面，导航按钮在屏幕底部，当点击心情图表功能时，跳转到心情图表界面，并绘制当日的心情曲线。当点击智能推荐时，跳转到智能推荐界面，获取当前关键词，并根据关键词得到相应的推荐图片，推荐描述和推荐网址，将这三条信息对应的显示在界面上，同时也更新笑话并显示在界面上。当点击今日总结功能时，跳转到今日总结界面，并开始获取分析数据和励志话语，并将这些数据显示在指定位置，具体流程图如图 5.7所示。

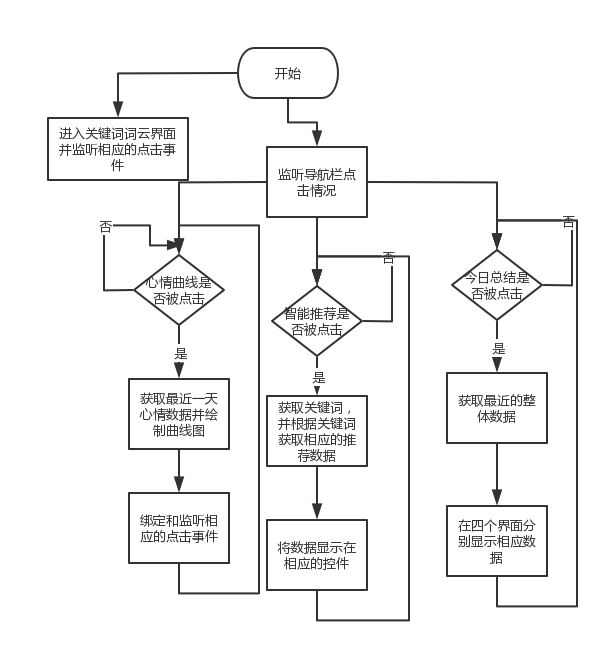


图 5.7 客户端整体程序流程图

关键词词云，主要是一张关键字词云图片的显示及一张原本轮廓的显示，在加上一个更新按钮，方便用户更新关键词词云图片。界面设计如图 5.8所示。

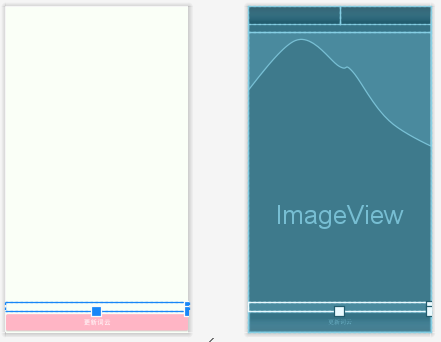


图 5.8 关键词词云界面设计图

心情图表提供 日，周，月三个维度的数据，并利用图表进行显示，在收到数据后还有一个情绪分析按钮供和一个Text框，用户点击按钮并可以在Text框查看最近的情绪状态和一些建议，以便及时调整自己的状态。心情图表界面设计如图 5.9所示。

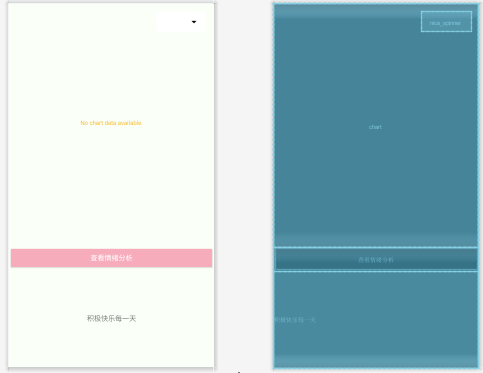


图 5.9 心情图表界面设计图

智能推荐功能是根据捕捉到的关键词，来进行活动项目的推荐，这个界面，有三个自定义控件，用于显示推荐的图片和描述，点击相应控件还可以跳转到相应的网页，除此之外还有一个按钮和一个Text框，当用户点击按钮时，在Text框更行笑话并显示，智能推荐界面如图 5.10所示。



图 5.10 智能推荐界面设置

今日总结功能是展示用户最近的整体数据，包括四个可滑动的页面，第一个页面上面有一个Text控件主要用于展示用户最近所说的词汇总数，界面二上面也有一个Text控件，用于展示用户最近说的最长的句子，界面三上面也有一个Text控件，用于展示用户最近提及最多的词语，界面四有一个词性分析饼状图，和一个Text控件，饼状图用于展示最近话语的词性构成，Text控件用于展示励志话语，今日总结整体界面设计如图 5.11，5.12，5.13，5.14所示。

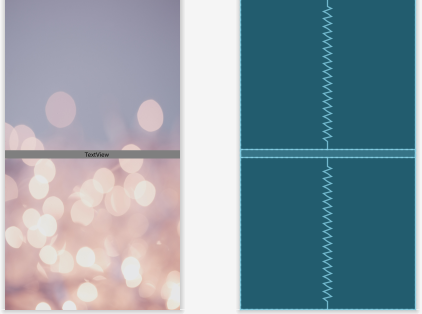


图 5.11 今日总结界面一

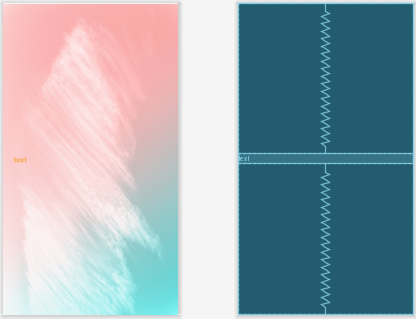


图 5.12 今日总结界面二

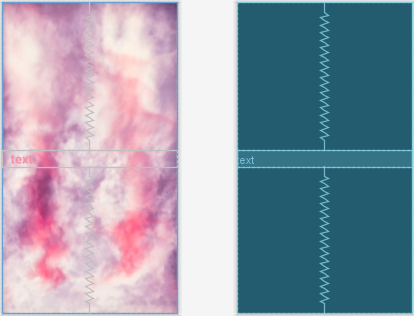


图 5.13 今日总结界面三

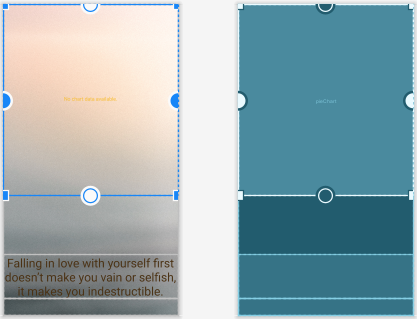


图 5.14 今日总结界面四

## 5.4 本章小结

本章主要介绍了软件部分，软件部分主要分为三个部分，第一部分是单片机音频采集与发送确定了采集和发送分离的结构，并由主板按键控制的设计。第二部分是服务端的通信与数据处理，主要包括和感知层，客户端的通信，WAV文件的接收和数据的智能分析。第三部分是客户端数据的展示，分为了关键词词云，心情图表，智能推荐，今日总结四大功能。

# 第6章 系统功能实现及测试

在整体软件设计完成的基础上，应该按照流程图来进行软件的实现，并在实现过程中记录问题及解决方案，对功能进行模块化测试和整体测试。对关键的功能还应该进行性能分析及优化想法的实现。

## 6.1 感知层功能实现与测试

在按照感知层硬件设计和软件设计过程中，硬件采用STM32战舰版，满足了硬件方面的需求，所用到的引脚和功能都能正常的进行工作。在软件功能实现和测试方面，存在的问题很多。这些问题和解决方式如下：

1）问题:esp8266 连接服务器有时候连得上，有时候连不上 十分不稳定，解决方案:AT命令的下达和执行需要时间，要做好延时操作。

2）问题:eclipse print 打印的值最后消失，解决方案:因为一行太长就自动消失了,所以要做好长度判断。

3）问题:传输数据有些字节会出错导致声音质量很差，解决方案:将数据在录制的时候就只录制很少的一部分，减少对整体影响。

4）问题:莫名奇妙的程序卡住，而且进程卡在一个地方出不来了，解决方案:函数指针p未申请地址，导致访问读写到未知地址去了,做好内存的申请与释放

5）问题:服务端收到的wav数据不正确，解决方案:感知层和服务端未同步，应该先让单片机在发送玩文件上传请求命令后，延时一会再发送真正的文件，留给服务端准备接收的时间。

6）问题:显示屏显示的信息不能体现当前单片机运行状态，导致单片机出问题不知道具体问题在哪，解决方案:在单片机代码中，增加显示屏显示函数，在每一步操作中打印日志信息在屏幕上。后续就能通过显示屏上显示的日志信息了解单片机当前运行状态，方便调试。

7）问题:不能连续发送wav串口文件，每次发送一个就没后续了，解决方案:两个上传操作间隔太短，服务端还未准备好接收，导致请求帧丢失

最终完成了感知层原本的设计。测试结果如图 6.1，6.2所示。

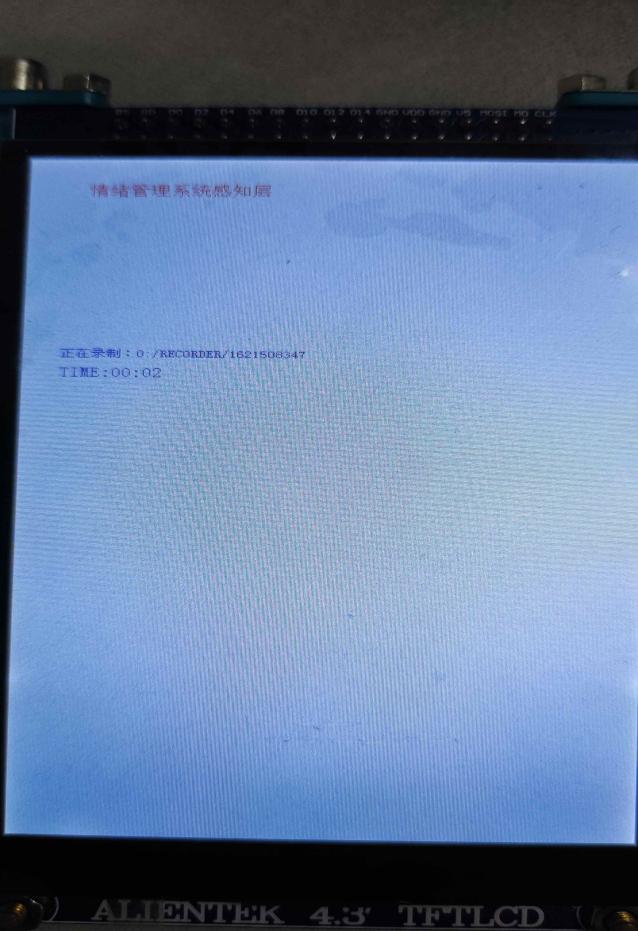


图 6.1 音频录制图

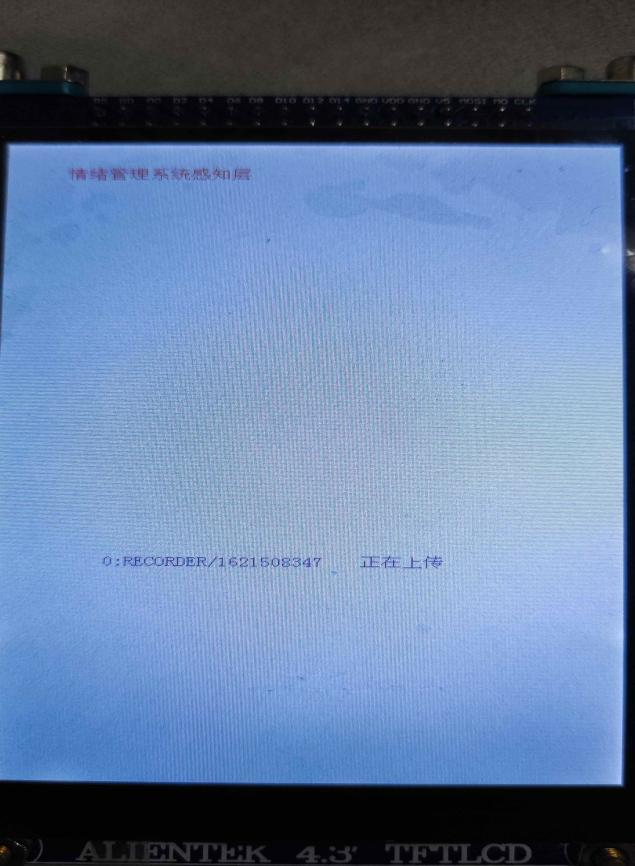


图 6.2 音频上传图

## 6.2 服务端功能实现与测试

服务端是工作量最大的部分，不仅需要完成和感知层的交互，还需要支持客户端的请求，还需要进行大量的数据处理，本系统严格按照设计需求来进行功能开发，测试的结果及改进方案如下：

1）问题:讯飞语音在线识别语音只能识别第一个词语，解决方案:项目文件的问题，需要在下载的SDK包的MscInvisibleDemo上进修改开发。

2）问题:在有不同音色时可能会出现只能识别一个人说话的内容，原因:是因为一段音频的结果提前返回，判定是最后一句话，判定结束后就会不接受writeAudio传来的东西，导致只能识别一句话。

3）问题:网络状态会影响返回结果，解决方案:使用延时来确保分片的有效性，因为一直快速上传分片会导致，识别返回的时候，剩下分片丢失,所以要做好分片和分片之间的延时操作。

4）问题:关键字提取的demo报错，有两个包无法import，解决方案:官网

<https://commons.apache.org/proper/commons-codec/download_codec.cgi下载相应的jar>包，然后添加jar就行

5）问题:Java没有自带json包，所以需要自己下载jar包并导入，解决方案:下载地址<https://github.com/alibaba/fastjson>

6）问题:读取文本文件乱码，解决方案:指定读取流的编码格式BufferedReader br=newBufferedReader(newInputStreamReader(newFileInputStream(fileName),"UTF-8"));

7）问题:关键字提取返回结果乱码，它的文件就行，我自己的新建类就不行，解决方案:连个文件的默认编码不一样，应该把自己的新建类编码格式设置为utf-8

8）问题:调用split进行文件名切割时出错，解决方案:.号需要进行转义 加上右双斜杠才能成功调用。

9）问题:词云不能按照图片形状生成，解决方案:必须使用抠图，把其他地方扣取，只保留想要的地方。

10）问题:在图片传输时 接收出问题，解决方案:这个收发不能传输png文件，需要jpg格式。

11）问题:在进行提取数据转换时，pcm会提前转换，但此时pcm数据都还没有，解决方案:在转换时进行判断如果正在进行最新pcm数据的转换要等待一段时间，避免转换的是空值。

最终完成了服务端所设计的功能，测试结果图如 图 6.3，6.4，6.5，6.6，6.7所示。

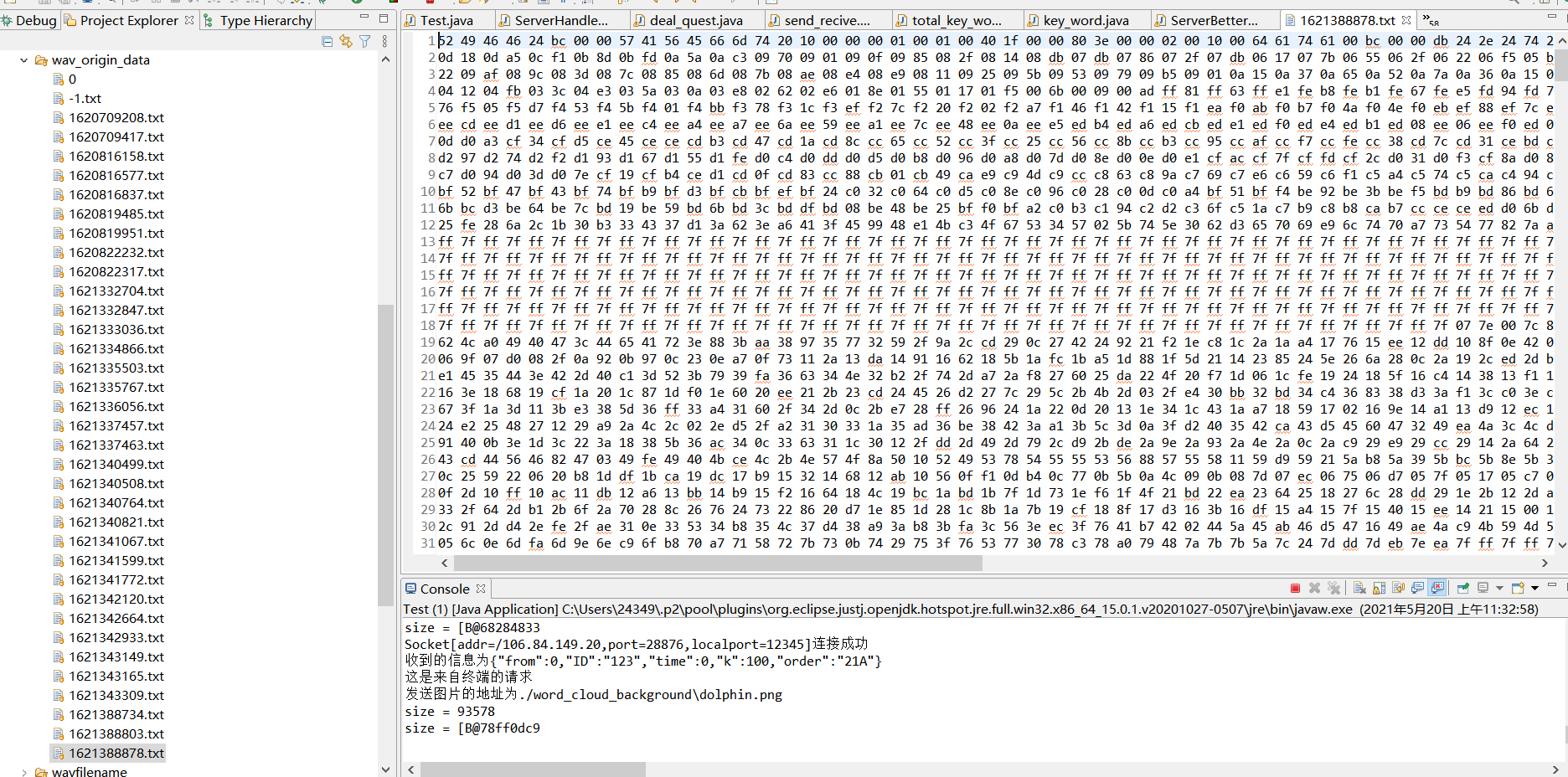


图 6.3 wav串口文件的接收

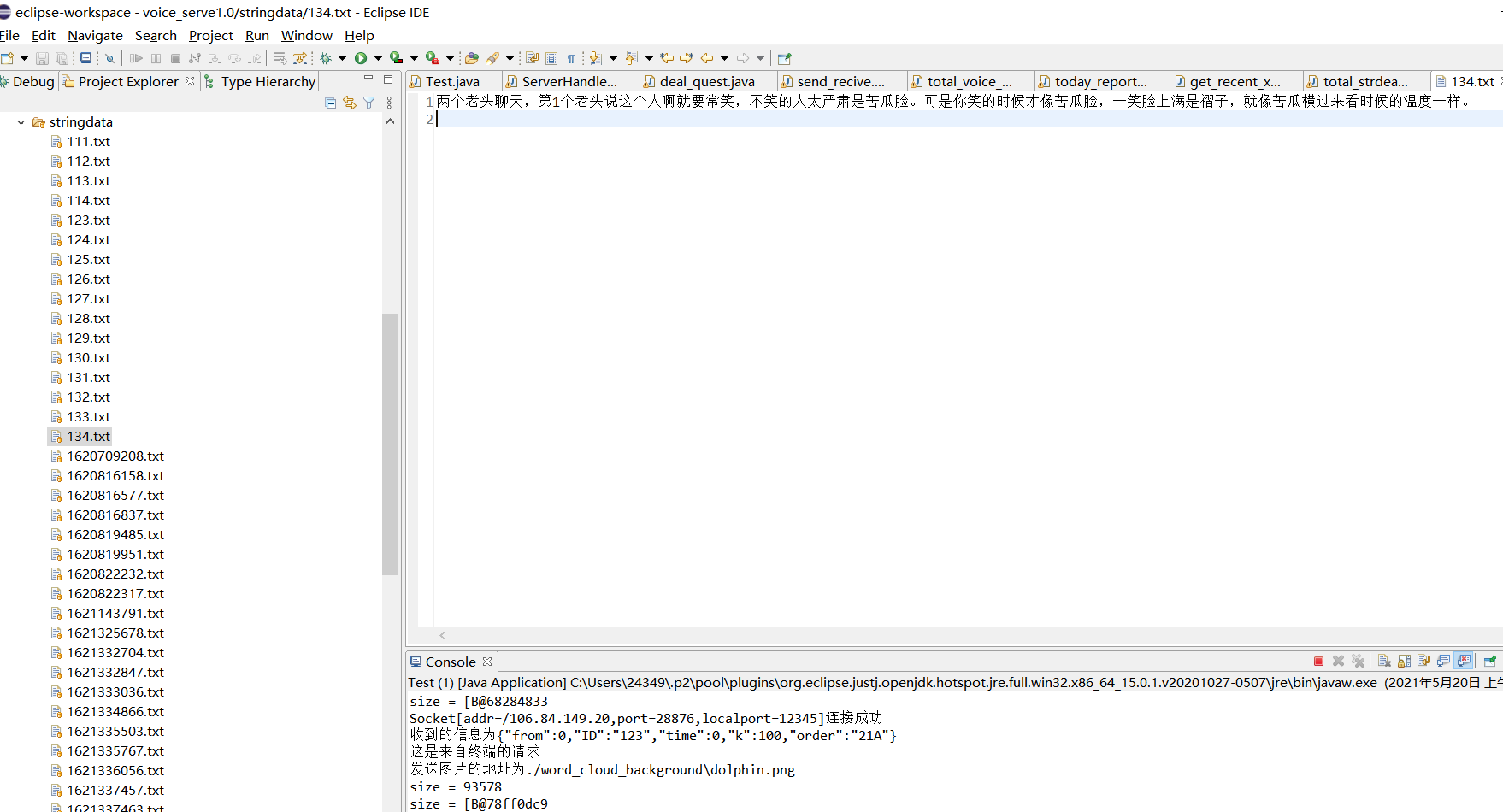


图 6.4 pcm转字符串结果

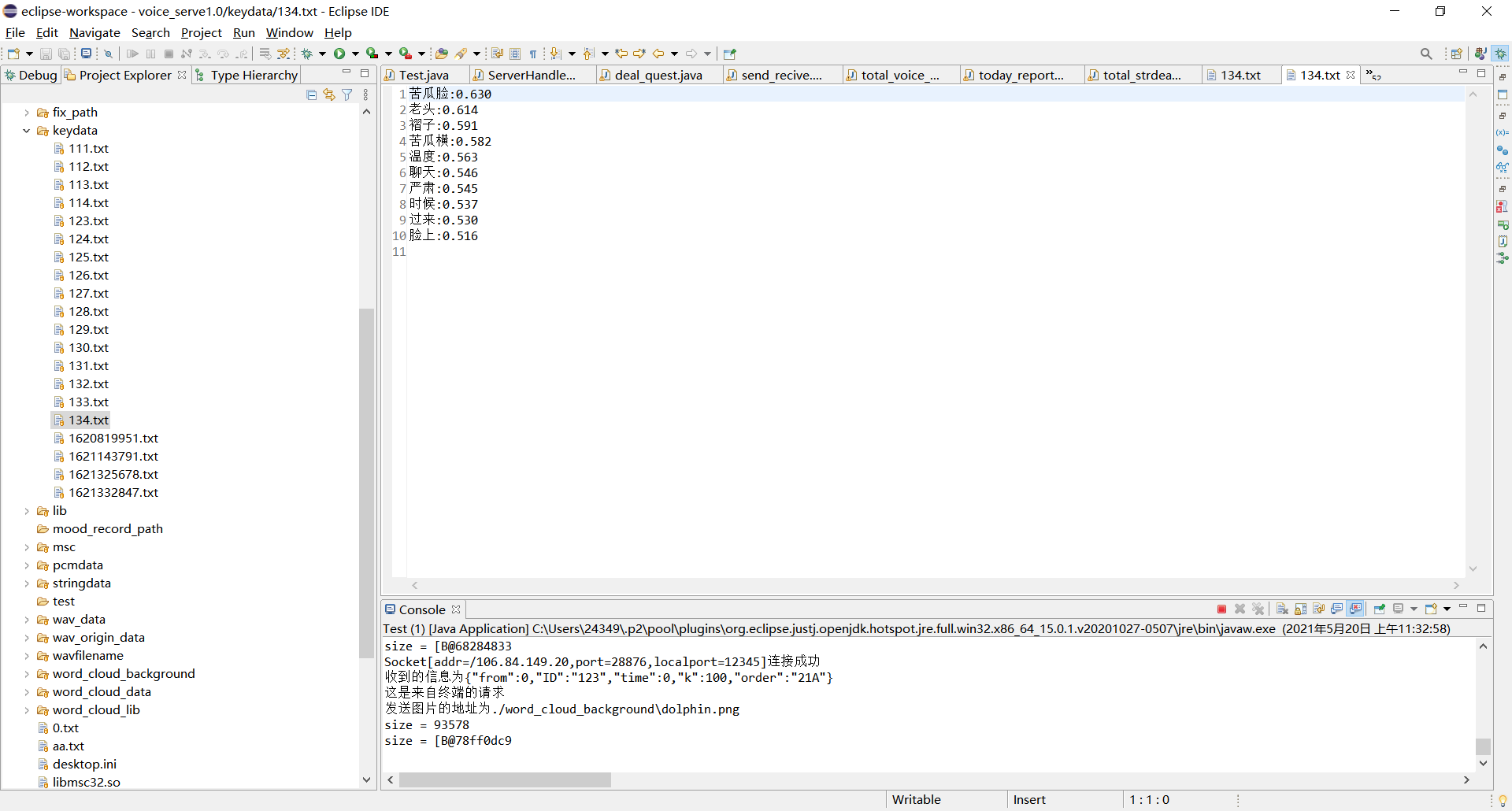


图 6.5 关键词提取结果

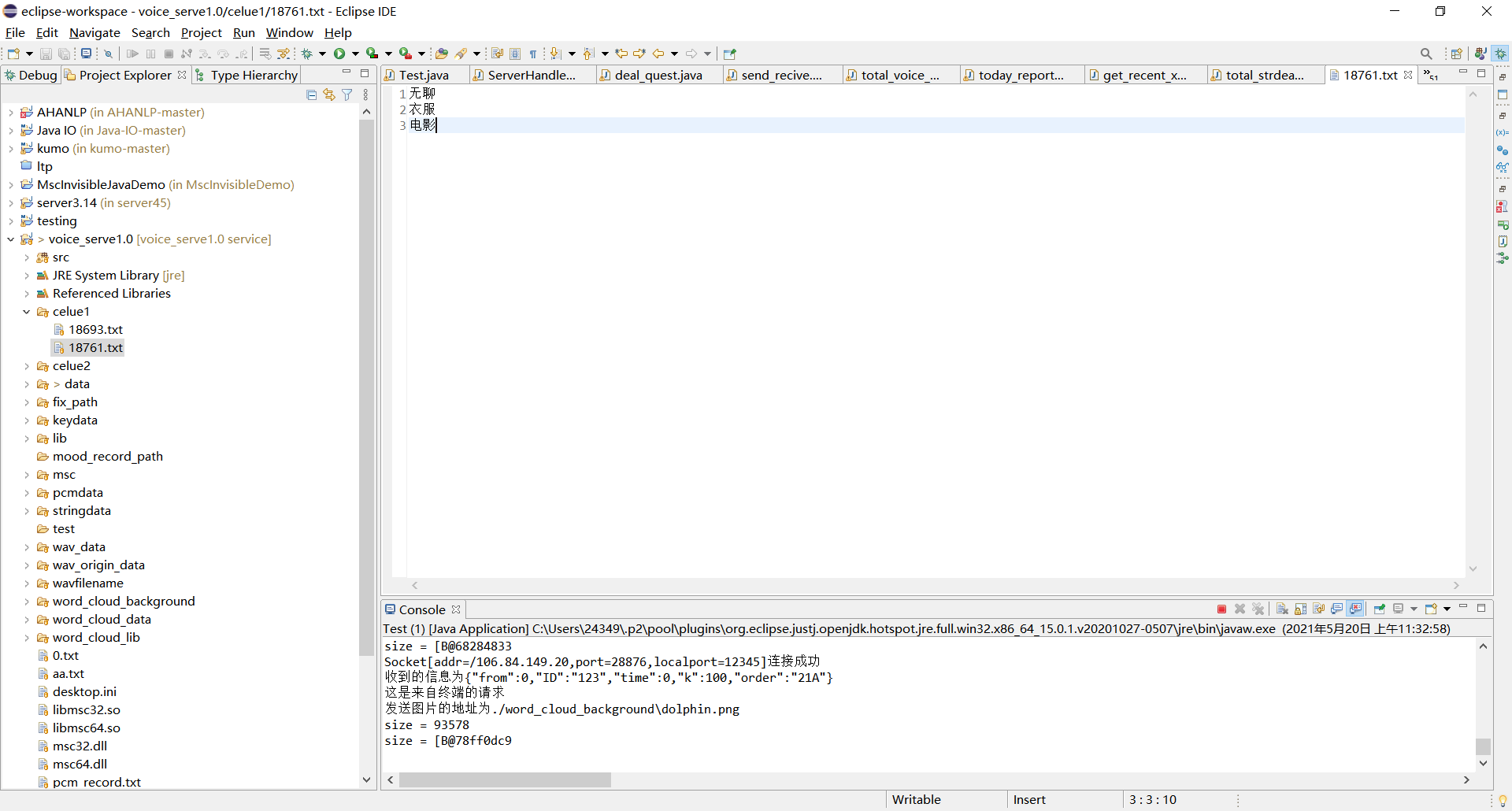


图 6.6 智能捕捉结果

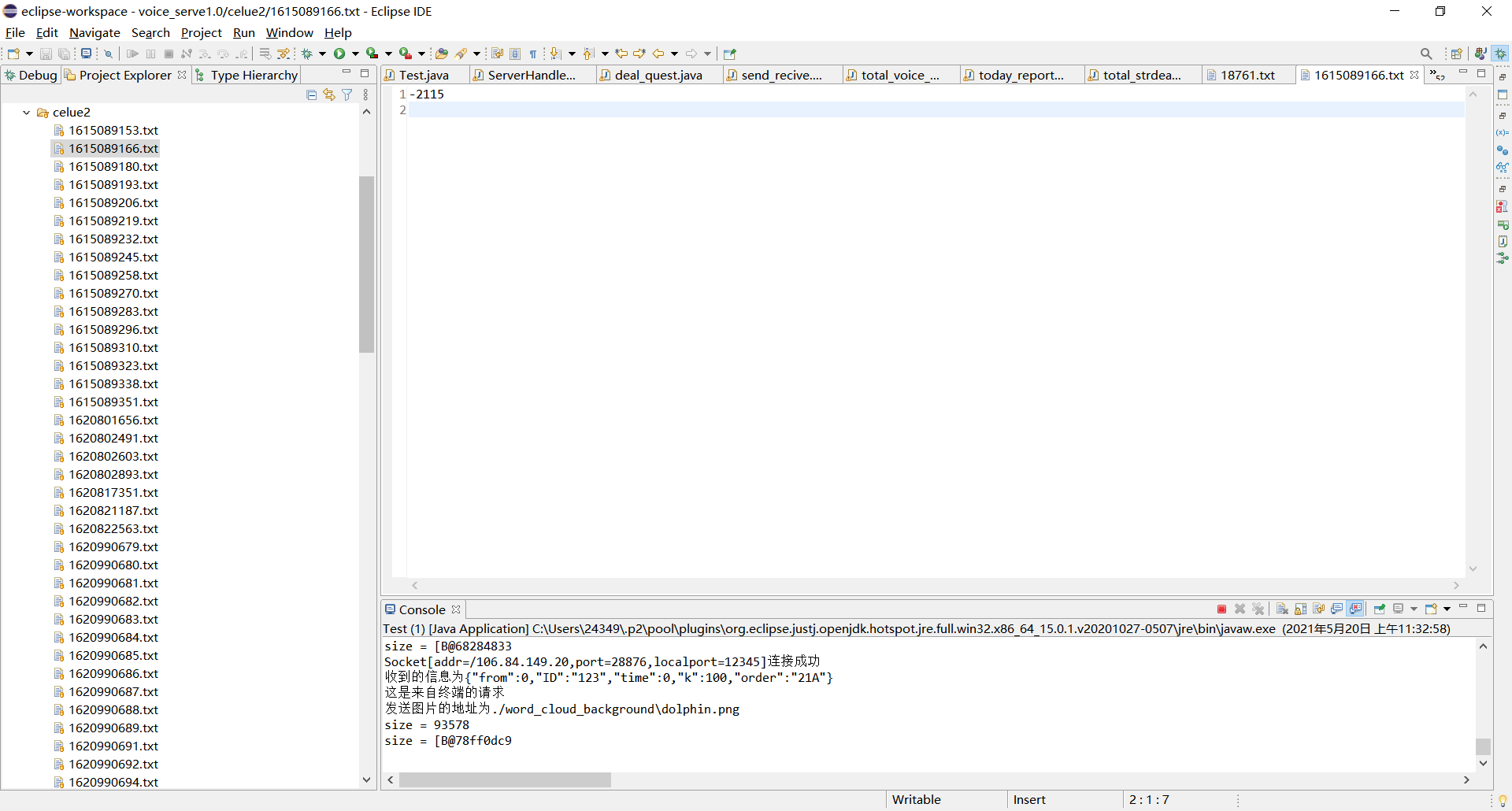


图 6.7 情绪分析结果

## 6.3 客户端功能实现与测试

客户端严格按照四大功能来进行开发和实现，具体bug和解决措施如下:

1）问题:测试通信时连接本地socket接口报错，解决方案:安卓终端不能用127.0.0.1代替本机地址，必须通过ipconfig去找到自己的地址。

2）问题:接收句子时只能收到一行，不能收到其它行，解决方案:增加接收的结束判断，不只是收到一行就返回。

3）问题；显示句子时报错闪退，解决方案:不能再工作线程中操作ui元素<https://blog.csdn.net/u013164293/article/details/51361588>。

4）问题:在使用bitmap时 BitmapFactory.decodeByteArray 返回值为null，解决方案:原图片太大了，在发送之前应该进行压缩（只针对虚拟机）实体机并不会为空。

5）问题:在report功能时，突然返回的分析值就为空了，解决方案:自然语言分析api每天有次数限制，当日的次数用完了，只有等到明天才行了。

具体功能测试结果如图6.8，6.9，6.10，6.11，6.12，6.13，6.14所示。



图 6.8 关键词词云结果展示

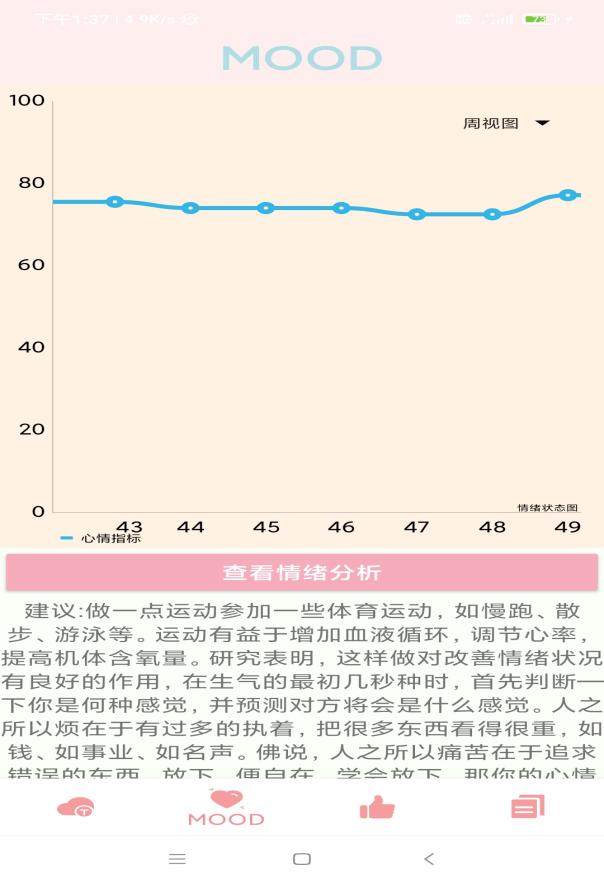


图 6.9 心情图表结果展示



图 6.10 智能推荐结果展示



图 6.11 今日总结界面一结果展示

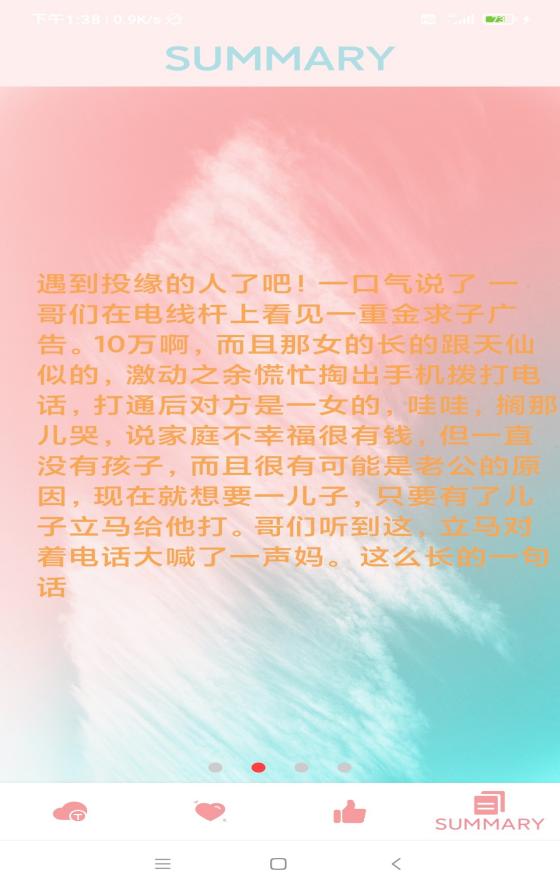


图 6.12 今日总结界面二结果展示



图 6.13 今日总结界面三结果展示

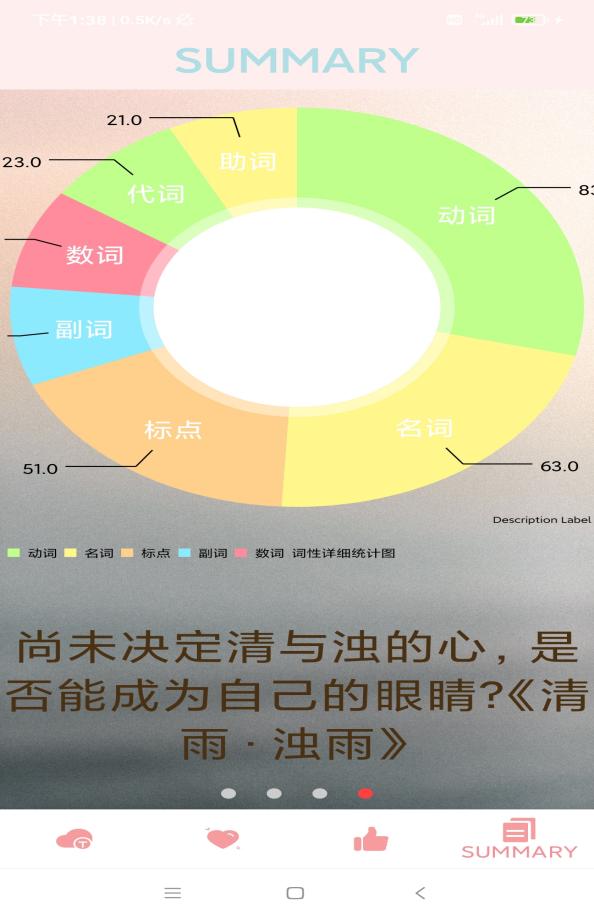


图 6.14 今日总结界面四结果展示

每个界面都实现了设计的功能，达到了目标。

## 6.4 PCM转字符的性能分析及参数自适应

服务端的PCM转字符串是通过调用科大讯飞相应的API来实现的，在使用时将待转换文件分片上传到相应的地址，然后等待返回上传结果，经过多次尝试发现，同一个pcm文件的分析质量会与分片大小，分片间隔时间，当前网络状态有关，现在利用控制变量法来分析这些参数对最终转换结果的分析，详细操作步骤如下：

先确定下来一条语音数据，并且知道这一条语音数据的内容。接着通过改变间隔时间这个参数来对语音进行转换，先设置时间步长为150，并得到转换结果，测试结果如图 6.15所示。

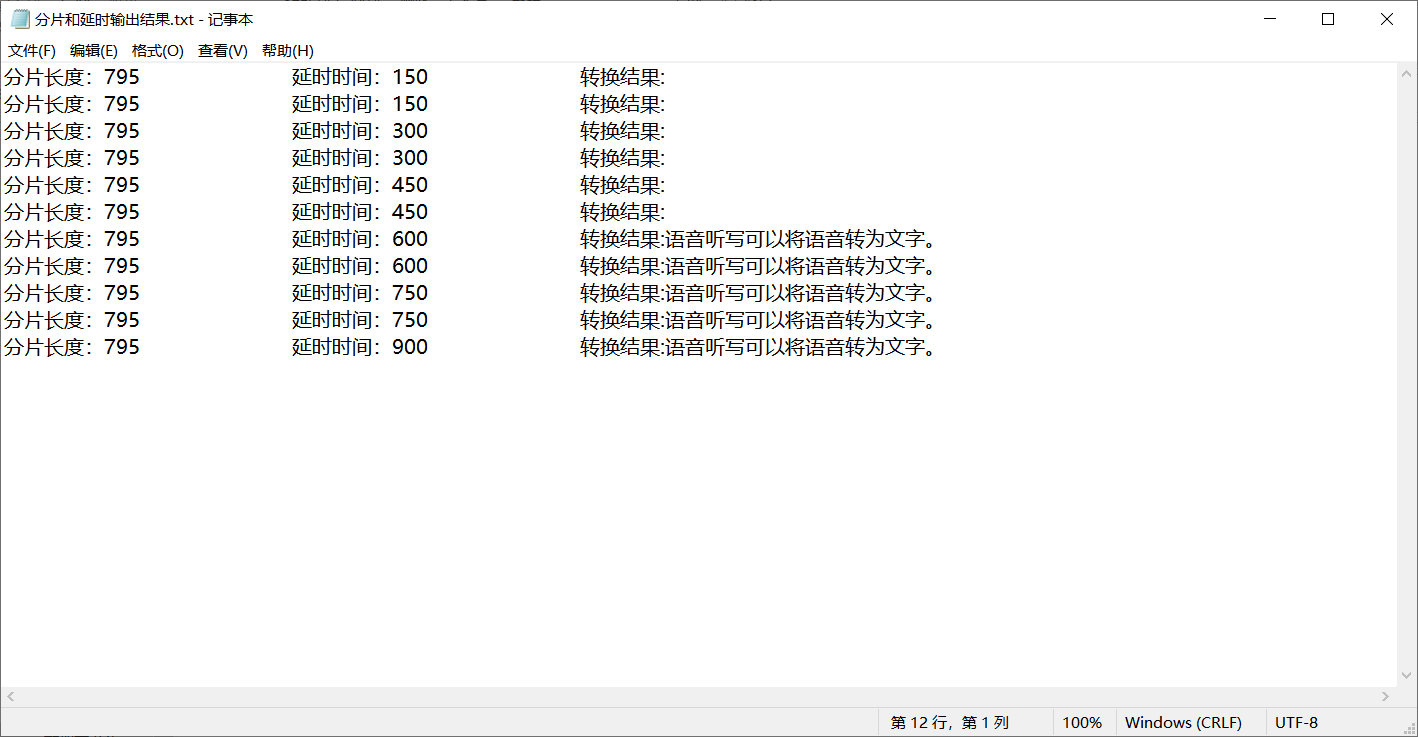


图 6.15 时间步长为150 的转换结果图

可以看出最优延时时间在450之后，现在调整步长为15，在450 之后进行转换，转换结果如下图 6.16所示。

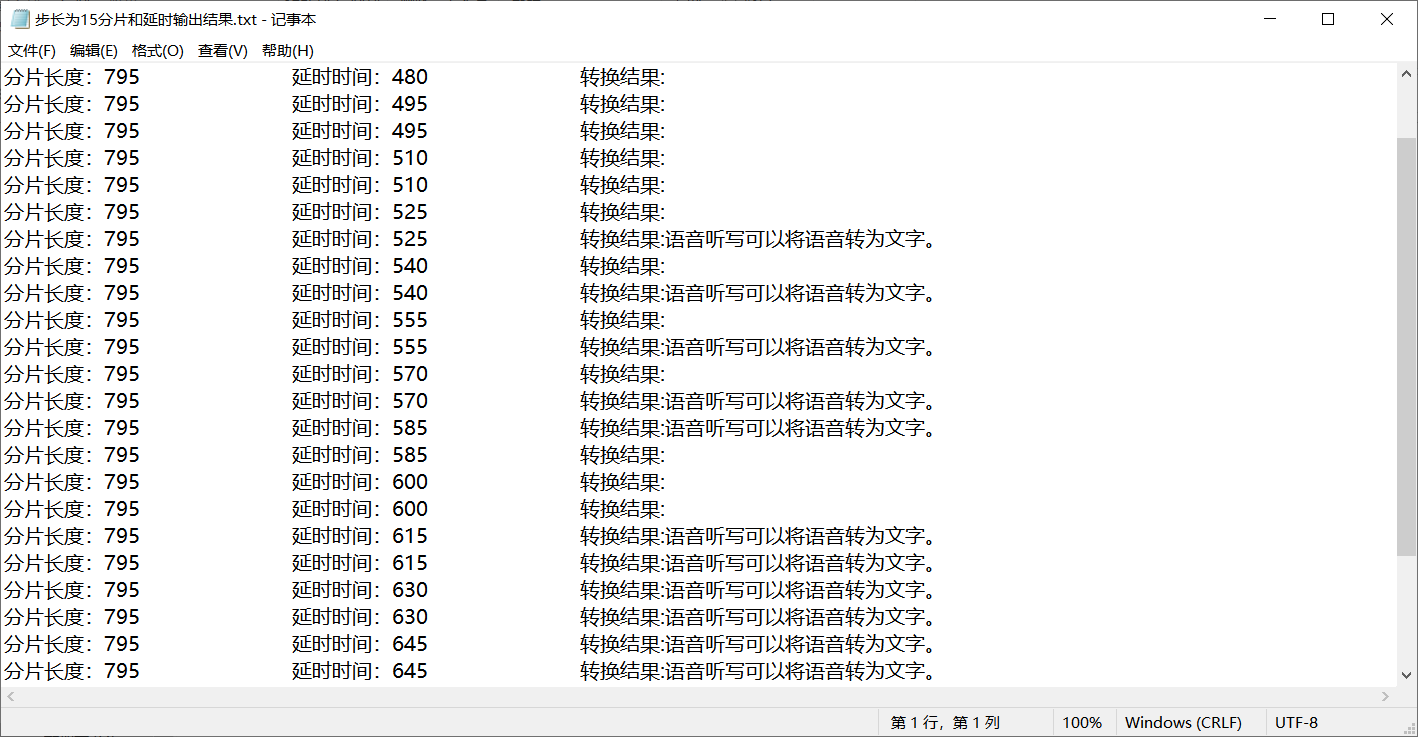


图 6.16 时间步长为15的转换结果图

可以看出间隔时间在615 处达到稳定，所以此时最优的间隔时间大概为615，但是由于这些测试都集中在相对较短的时间，所处在的网络环境可以看作一样，当网络环境不一样时，最优的间隔时间可能又会和615相差较大，如果此时系统还采用615作为间隔时间可能就会会造成没有转化结果的情况。综上本系统决定采用参数自适应的方式，具体步骤如下：

当系统启动时，首先以15的步长对已知的文件进行转换，并将转换结果和标准结果进行比较，如果相同则说明此时间隔时间比较合适，但为了避免偶然情况的发生，设置成为连续三次都和标准结果一样才采用此间隔时间作为后续的间隔时间，程序流程图如下图 6.17所示。

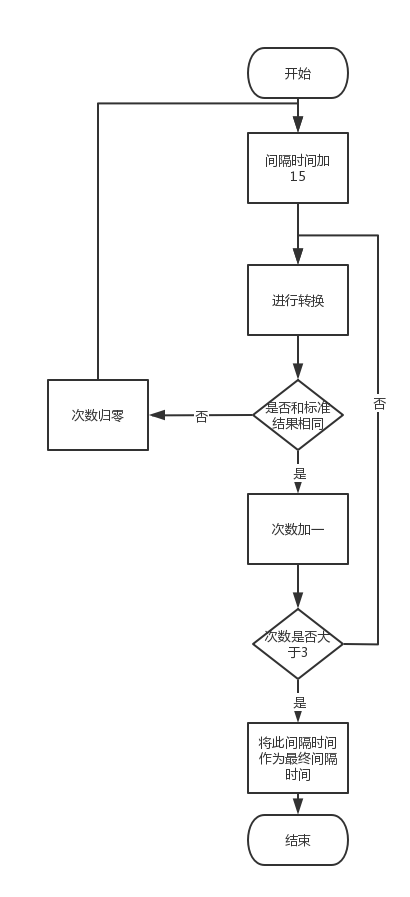


图 6.17 间隔时间自适应流程图

最终服务端使用结果如下图 6.18所示。

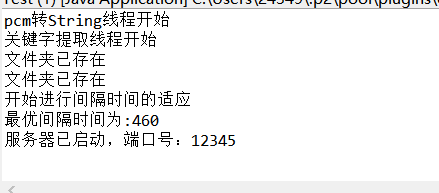


图 6.18 自适应间隔时间结果

现在将固定间隔时间和自适应间隔时间这两种方式用于同样的音频文件的转换，观察所耗费时间的区别，结果如下图 6.19，6.20所示。

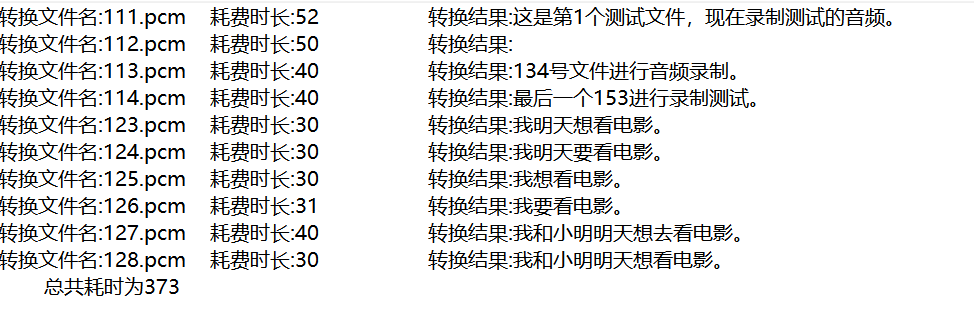


图 6.19 固定时间耗时

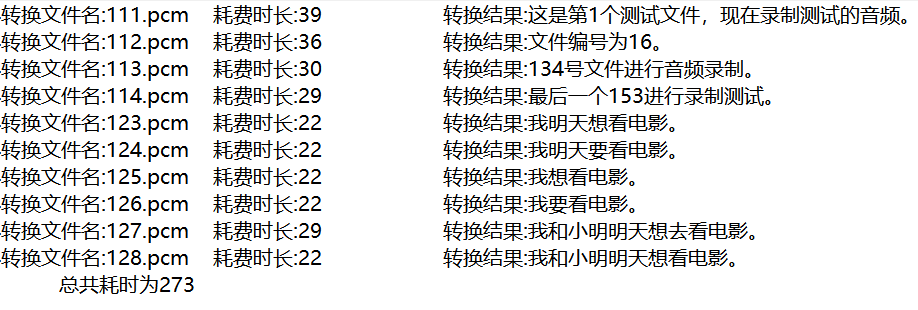


图 6.20 自适应时间耗时

相比之前设置的间隔时间为1000来说，此时自适应的间隔时间，文件转换质量也上去了，十个文件全都转换成功，而固定时间只进行了九个成功转换，在时间方面，固定时间耗费的时间为373，而自适应时间只耗费了273，提高了36.6%以上的转换效率，可以看出加入自适应，能很好的找到当前网络情况下的最优间隔时间，增加本系统的转换效率。

## 6.5 系统整体测试

本系统的主要功能是采集音频文件，并分析数据，最后在客户端展示和智能推荐，所以将从 数据的准确性，时间的及时性，和推荐的智能性三个方面进行判断，根据之前设计需求，数据的准确性应当在10个音频文件中，至少有九个文件识别成功，识别成功率要达到90%及其以上才不会对推荐产生太大的影响。时间的及时性要求在用户上传录音开始计时到数据分析出来展示给用户停止计时，时间耗时不得超过1分钟，推荐的智能性要求在用户话语说出表达意愿很强的词后，能正确的捕捉，并做出能满足用户的需求。

数据准确性测试结果如图 6.21所示。

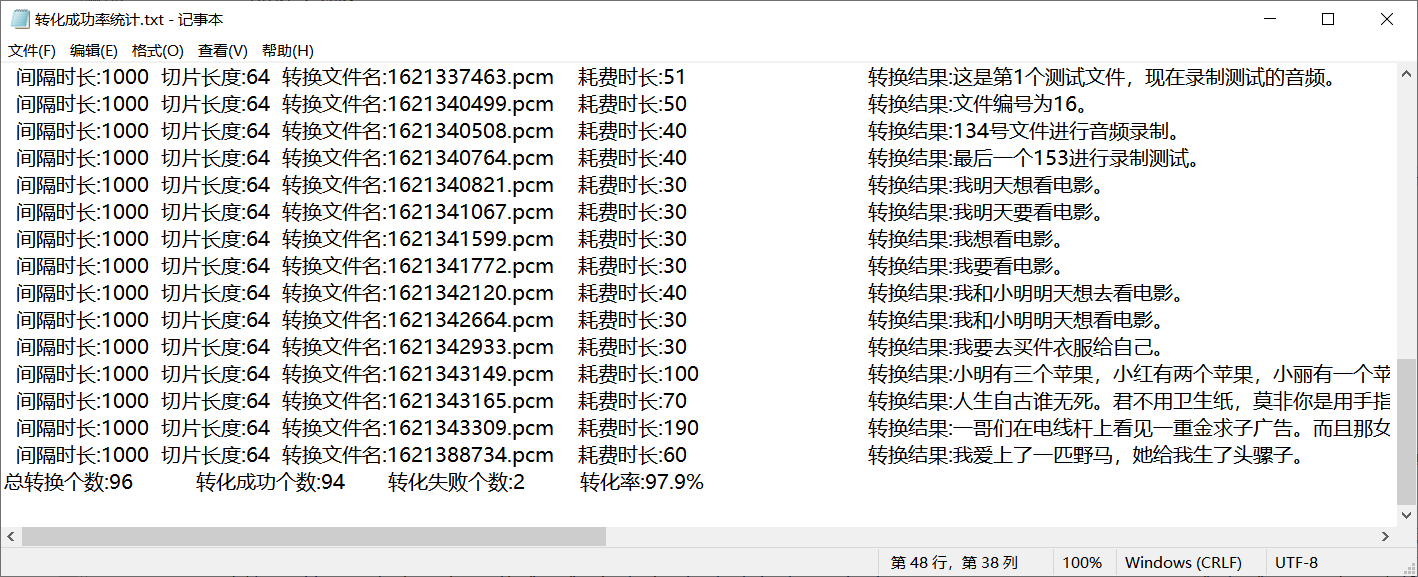


图 6.21 音频转字符成功率图

可以看出，在96个转换记录中，成功的个数高达94个，转换率高达97.9%，远高于目标90%，所以数据准确性满足要求。

在及时性方面，本测试采用心情分析响应时间作为结果，先在客户端查询心情分析，然后在上传数据后再进行心情分析查询。结果如图6.22，6.23所示。



图 6.22 上传数据前客户端心情分析截图



图 6.23 上传数据后客户端心情分析截图

由截图可以看到在8:18时候进行了心情分析的查询，当时没有上传任何数据，分析显示也没有数据，在第二张截图时间还是8:18，此时数据已经上传和分析完毕，得到了相应的分析结果，从两张截图的时间可以看出，整体消耗的时间小于一分钟，满足设计要求。

在智能推荐方面，之前测试的时候文本中出现了多次电影这个词语，表明当前用户十分想看电影，此时查看客户端的推荐界面，如下图6.24所示。



图 6.24 客户端推荐界面图

从截图可以看到，第一个推荐就是猫眼电影的图片，及其描述的文字，满足设计需求，推荐的智能性也达到了目标。

## 6.6 本章小结

本章是将本设计的软件、硬件模块进行结合之后，对需求的实现以及问题的解决方法，先将三个模块的功能实现的所遇问题及解决方案列出，分别是感知层的音频录制和音频发送，服务端的与感知层，客户端的通信还有字符的智能处理，然后是客户端展示的相应功能。接着还描述了PCM转字符串的性能分析及自适应的逻辑和前后性能对比，最后进行了一个整体测试，主要从数据的准确性，时间的及时性和推荐的智能性对整个系统进行了判断，经判断，本系统满足设计要求，达到预期目标。

# 第7章 总结与展望

## 7.1 主要工作

在信息爆炸的当下，一方面人们一边在享受信息爆爆炸带来的福利中，同时也慢慢的迷失在广袤的信息中，自己对自己的了解也变得不清楚，自己的情绪也没关注，导致经常情绪失控。另一方面大家没有意识到语音的重要性，语音中除包含实际发音内容的话言信息外,还包括发音者是谁及喜怒哀乐等各种信息。所以,语音是人类最重要,最有效,最常用和最方便的交换信息的形式。本系统就从这两点出发，通过语音数据的采集和分析，将用户的最近情绪状态和隐藏的需求挖掘出来，并展现给用户，帮助用户能增强对自己的了解，控制自己的情绪，做出更好的抉择。本文的工作主要包含以下几个方面：

1）分析了当前语音的重要性及人们情绪健康的重要性，明确了基于智能语音识别与分析的情绪管理系统设备实现的可行性和应用的意义。

2）讲述了目前国内外研究的情况，并确定系统整体采用C/S架构，将系统分为感知层，服务端，客户端三个部分，以及本系统各个功能模块的选择及理由，主要分析了无线通信的ESP8266模块以及音频采集的VS1053芯片。

3）介绍了整体的功能需求分析和感知层，服务端，客户端的总体设计方案，紧接着完成了各个功能模块的硬件模块的介绍和整体硬件的设计。

4）详细描述了软件设计的细节，包括ESP8266从WIFI热点的连接到指定服务器的连接最后实现双方的通信，并给每个逻辑都绘制了相应的程序流程便于理解。

5）介绍了WAV音频文件的重构，完成了单片机通过串口控制ESP8266，传输WAV的字符串文件，最后在服务端进行进行格式转换和重新字节写入，最终实现WAV文件的无线传输。

6）按照系统分层分别对感知层，服务端，客户端三个部分功能实现过程中出现的问题和相应的解决方案进行了详细介绍，接着单独分析了PCM转字符串的性能，和自适应参数的实现逻辑，最后在三个维度对整体系统进行了分析判断，最终判定本系统满足设计需求，达到预期目标。

本设计的创新点如下：

1）想法独特新颖，当下结合语音收集进行数据分析挖掘的产品很少，本设计课题有很高的应用价值，紧密结合现在的生活，在物质越来越丰富精神追求被重视的情况下，能帮人了解自己，控制情绪，做出抉择的智能系统流行将是大势所趋。

2）本系统是在对预测未来，智能抉择尝试性的迈出了一步，理论上只要采集到详细的数据就能通过，数据分析处理，最终达到预测未来智能抉择的效果。

3）实现了通过ESP8266WIFI模块的WAV文件透明传输，没有直接传输WAV文件，而是通过传输WAV文件的字符数据，再将数据在服务端重构成为WAV文件。

4）在PCM转字符串上面，提出了自适应的想法，并将此想法运用于本系统，不仅提高了转换正确率，而且还提高了转换效率。

本设计完成了预设的需求功能，但在一些用户体验上还有提高的空间。而且数据采集还远远不够，分析处理也停留在初级的阶段，智能的推荐和情绪管控也不能挖掘到深处需求，只停留在表面，所以后续还需要花费更多的时间用在这系统的优化和更新，具体的待改进需求如下：

1）感知层音频采集策略的优化，当前感知层采用单线程处理，音频的采集和发送不能同时进行，所以不能一天24小时都在采集音频，后面采用多线程处理这两个功能实现同时录制和发送。

2）服务端程序结构变得冗余，并且耦合性太高，后面要对服务端程序重构一遍，方便以后的开发。

3）本系统的硬件实物较大，不适合随身携带，后期重新设计PCB板，将没用的模块删除，精简实物。

4）采集数据的拓展，目前只能采集音频数据，这显然远远不够，后期增加其它数据的收集和保存，包括但不限于心跳数据，环境温湿度数据，甚至实时的视频数据，一步一步收集足够的数据，最终实现“预测未来”。

## 7.2 后续研究工作展望

虽然本系统只完成了基于音频采集的分析处理，但从单音频数据中也获得了大量信息，进行的推荐也比较合理。这足以体现基于数据采集的未来预测的正确性，在物质条件越来越好的当下，能帮人做好分析和决策的功能必将流行，这也能极大的提高社会整体生产力，所以这个研究方向会越来越好，将会是大势所趋。

# 

# 参考文献

1. 吴进. 语音信号处理实用教程[M].人民邮电出版社:, 201502.349.
2. 程锐. 基于语音相位估计和声源空间特征的语音增强方法研究[D].北京工业大学, 2020.
3. 刘志宏,顾建.不良情绪的自我调节[J].解放军健康,2012(05):23.
4. 梅笑.情感劳动中的积极体验：深层表演、象征性秩序与劳动自主性[J].社会,2020,40(02):111-136.
5. 任保平,邹起浩.新经济背景下我国高质量发展的新增长体系重塑研究[J].经济纵横,2021(05):74-84+2.
6. Ding Yibing,Zhang Hongyuan,Tang Sitong. How Does the Digital Economy Affect the Domestic Value-Added Rate of Chinese Exports?[J]. Journal of Global Information Management (JGIM),2021,29(5).
7. 邵娜,李晓坤,刘磊,陈虹旭,郑永亮,杨磊.基于深度学习的语音识别方法研究[J].智能计算机与应用,2019,9(02):135-142.
8. Han Sapphire Yu,Liefbroer Aart C.,Elzinga Cees H.. Mechanisms of family formation: an application of Hidden Markov Models to a life course process[J]. Advances in Life Course Research,2020,43(C).
9. 蒲新和. 基于深度神经网络声学模型的说话人自适应技术研究[D].南京邮电大学,2020.
10. 安晖,冯晓辉,王哲.国内外智能语音产业的格局与趋势[J].人工智能,2018(01):5-18.
11. 徐兆星.国内近10年英语语音习得研究综述[J].重庆与世界(学术版),2015,32(01):109-114.
12. 焦潇雅. 基于语音识别的手语翻译算法研究与实现[D].电子科技大学,2020.
13. 轩中.中国有代表性的人工智能手机应用排行榜[J].互联网周刊,2018(18):64-65.
14. 石慧.我国语音识别专利申请技术构成变化趋势分析[J].产业科技创新,2020,2(17):97-98.
15. Ma Zhaoqian,Zhi Yan. Design and Research of Health Tester System Based on STM32[J]. Open Access Library Journal,2020,07(12).
16. Yucun Wang,Hongxiang Yang. Research on Ideological and Political Reform of Single Chip Microcomputer Course in Colleges and Universities[A]. Science and Engineering Research Center.Proceedings of 2021 International Conference on Modern Education and Humanities Science (ICMEHS2021)[C].Science and Engineering Research Center:Science and Engineering Research Center,2021:3.
17. 迎九.8位单片机市场不会消亡[J].电子产品世界,2015,22(10):9-11+41.
18. 俞阿龙,李正,孙红兵,孙华军. 传感器原理及其应用[M].南京大学出版社:, 201706.331.
19. 欧阳大亮. 无线通信中基于强化学习的天线选择研究[D].华侨大学,2020.
20. 张丽梅,甘伯青,易施光.基于STM32的多功能MP3设计[J].电子技术与软件工程,2019(06):72-74.
21. 常爱珍,李宝辉,张凯.TFT-LCD显示原理及评判参数[J].汽车电器,2021(04):31-33.
22. 李亚儒. 基于STM32的无线语音传输系统[D].北京邮电大学,2013.
23. 张丽梅,甘伯青,易施光.基于STM32的多功能MP3设计[J].电子技术与软件工程,2019(06):72-74.
24. Holovatyy Andriy. Development of IoT Weather Monitoring System Based on Arduino and ESP8266 Wi-Fi Module[J]. IOP Conference Series: Materials Science and Engineering,2021,1016(1).

# 致谢

这次毕业设计的顺利完成，不仅得益于我自己的努力，共重要的是在于多方面的共同帮助。首先，我要感谢本学校自动化学院四年来的真诚教学，感谢导师、老师和同学们的帮助。

其中，我要特别感谢邓钦元老师。在本次毕业设计中，他时刻关注我的研究进展。每次我遇到问题，他都能给我指导，拓展我的思路。当我没有完全理解的时候，他也能耐心热情地提出建议。他严谨的学术态度和敏锐的思维使我认识到他作为教师的模范作用。感受到他对事业的奉献和热爱，感受到他崇高的使命感和责任感，他的一言一行无一例外地影响着我的思想，他不仅在研究上给了我悉心的指导，而且在思想和生活上给了我关怀和帮助，在此我向邓钦元老师表示衷心的感谢。

经过四年的学习，在老师和朋友的全力支持下，我非常努力也很不容易，但是收获了很多。回顾工作中的学习经历，太多的情感难以表达。虽然我做了很多艰辛的努力，但我深深地感到所有的努力都是值得的。经过这段时间的学习，我在工作和生活中都学到了很多东西，也从身边的老师和同学身上学到了很多宝贵的知识和经验。在这里同样向帮助过我的老师和同学致以同样真诚的谢意！

# 附录

一、英语原文

**Robust Speech Processing Using Multi-sensor Multi-source**

**Information Fusion--an Overview of The State of The Art**

Parham Aarabi a,\*, Belur V. Dasarathy b a Electrical and Computer Engineering, University of Toronto, 10 Kings College Road, Toronto, ON, Canada M5S3G4 b Huntsville, AL, USA Received 3 December 2003; received in revised form 3 February 2004; accepted 3 February 2004

**Abstract**

This article offers an overview of the state of the art in robust speech processing and delineates the role of information fusion in furthering its objectives. In addition, it also serves the function of the traditional guest editorial of a special issue in terms of presenting a brief introduction to its contents.

@2004 Elsevier B.V. All rights reserved.

Keywords: Robust speech processing; Multisensor information fusion; Speech recognition

**1. Introduction**

For decades, extensive research has been conducted on enabling computers to process (i.e. localize, enhance, and/or recognize) speech [1–4,16–20,33]. During this time, numerous advancements in automatic speech recognition and processing have been made, resulting in various commercial speech recognition systems. These state-of-the-art speech recognition systems often have a high recognition accuracy rate obtained through extensive training and computations. The high accuracy rate, however, substantially decreases in the presence of noise, such as secondary conversations. As a consequence, robust speech recognition in noisy environments continues to be a popular research topic as a yet unresolved challenge [16,18,20,21]. Recent advancements in single-microphone speech recognition (including probabilistic speech enhancements and noise modeling techniques such as [16,21]) have gone a long way in addressing some of the robustness issues with current speech recognition systems. There is, however, a limit as to what can be achieved with a single microphone. Systems that employ multiple sensors (including cameras and microphones) have a potential for greater robustness and higher recognition accuracy rates. The fusion of multiple sensors or multiple information sources for robust speech processing is the focus of this special issue.

**2. Literature review**

The application of information fusion methodologies for speech processing applications has been explored in the past [4,6,8]. For example, in [4], an audio-visual vowel recognition system was explored and analyzed using an elucidative fusion paradigm. This work not only illustrated the benefits of the fused multi-modal system, but also illustrated the relative influence of the individual modalities on the fused result. For example, at a signal-to-noise ratio of )24 dB, the visual modality had a much larger relative influence over the acoustic modality for the correct decisions than it did when there was no noise present. This clearly illustrates the benefits of information fusion: in different situations, some information sources may be unreliable, and as long as a proper method for the fusion of these sources exists, the relative influence of the unreliable sources will be smaller than those of the more reliable sources. Similar work in audio-visual supports this claim [4–8,23,34–39]. Audio-visual information fusion has for many years been an area of extensive research. In [25], a hybrid audio-visual feature fusion and audio-visual decision fusion methodology is employed for robust speech recognition. In [26], a coupled hidden Markov model (which is a special case of Dynamic Belief Networks) is employed for audio-visual information fusion for speech recognition. In [27], a weighted decision fusion approach is employed for audio-visual speech recognition. A variety of other audio-visual speech recognition systems, which fuse the signal of a microphone with the images recorded by a camera, have also been proposed [28–32]. In all of these systems, the fused audio-visual system results in a substantial reduction in the word-error-rate as compared to either the visual or the acoustic modality by itself.

In spite of all of this research, fusing the information obtained by multiple sensors, some of which could be of different modalities, is still an open-ended and by its very nature a vaguely defined problem [2,8,22]. First, there is \the question as to exactly what type of sensors should be used. Combining cameras and microphones, for example, can result in greater robustness since cameras are unaffected by acoustic noise and microphones are unaffected by events (such as objects, obstacles, lighting, etc.) that affect cameras. However, fusing different modalities is often more difficult and complex, which can lead to greater computational requirements. Nevertheless, even with the existence of this complexity-effectiveness trade-off, numerous systems have attempted to fuse multi-modal information for a variety of applications. Examples include audio-visual speech recognition systems employing a single camera and a microphone, resulting in a higher speech recognition accuracy rate, and a greater robustness to noise [5–15]. Other applications include audio-visual sound localization [2,3], where a speaker is localized visually using multiple cameras and acoustically using multiple microphones. As expected, these systems illustrated improved performance over the audio-only or the video-only localization system. The next issue that arises is the point at which the information is fused. For example, the signal values that are obtained by a set of microphones could be integrated at the signal level (using techniques such as beamforming), or, they could be integrated after speech recognition is performed on each channel. In fact, this integration or fusion can occur anywhere in the speech recognition or localization process. Most of the prior work in this area has focused on data fusion prior to the recognition process (i.e. the signals are integrated to result in a cleaner signal which is then used for recognition). Such a pre-recognition fusion has the advantage of computational simplicity, although its optimality is not very clear. Pre-recognition data fusion can be effective when an array of microphones is used. When cameras and microphones are used jointly, it is far more difficult to combine their data at the signal level. Any such fusion would have to involve a projection of the visual results into the space of acoustic results, or vice versa. Clearly, this projection can be a very complex and computationally demanding task. Nevertheless, there have been attempts at early-stage audio-visual information fusion [1,11,12] as well as late-stage audio-visual information fusion [14,15]. These examples explore the relative tradeoff between increased complexity and increased speech recognition accuracy.

So, the questions at hand are (1) what do we fuse, (2) when (i.e. at what stage of the recognition process) do we perform this fusion, and (3) how do we actually perform the fusion. These are three inter-related questions whose final answers may be years away. Nevertheless, the six papers in this special issue provide some unique and innovative solutions to these questions.

**3.Special issue overview**

The first paper, which introduces asynchronous hidden Markov models (HMMs) for audio-visual speech recognition, proposes an asynchronous HMM structure, which is jointly applied to acoustic and visual data obtained from a microphone and camera, respectively. It is shown that the proposed fusion model outperforms the standard HMM architecture for either audio-visual speech recognition or for acoustic-only recognition, especially at lower signal-to-noise ratios (SNRs).

The second paper also considers the audio-visual information fusion problem by performing decision fusion for continuous audio-visual digit recognition. Their approach involves a post-recognition fusion (decision fusion) of the recognition results obtained from the visual data and the results obtained from the acoustic data. It is shown that at very low SNRs (15 dB), their system performs similar to the audio-only system and far better than the visual-only system. In the mid-SNR range (0–15 dB), their system outperforms both the audio-only and the video-only systems.

The third paper in this special issue attempts to fuse two microphones and two cameras at the signal level for speech enhancement. Here, the visual data that are related to speech (i.e. the lip movements) are initially translated into two corresponding acoustic signals. These visually derived or virtual acoustic signals are then used as a basis for speech separation using the signals obtained from the two microphones

The fourth paper deals with the very interesting topic of quantizing the worth of a sensor in a multi-sensor setting. In a setting were multiple sensors such as microphones are present, the utilization of all sensors for speech processing applications may not be needed. However, in order to determine what sensors should be utilized, the amount of unique information provided by that sensor must be measured. This study proposes an information theoretic measure for exactly this purpose.

The fifth paper in this special issue employs an array of 24 microphones for the control and navigation of a robotic tour-guide. Here, the data collected from the 24 microphones are processed in order to estimate the location of the speech source, which is assumed to be the robot. This multi-sensor acoustic localization is used to correct the position of the robot and to steer it towards its next destination. It is interesting to compare this fifth paper with the system proposed by Enzo et al. [24]. Unlike the technique of [24], where a microphone array onboard the robot was used for human operator localization and for improved human–robot interactions, this study utilizes a fixed array placed in the environment. It is interesting to note that the proposed acoustic robot localization systems are in fact complementary, making the fusion of the two an interesting and possibly fruitful direction of future work.

The sixth paper, which focuses on feature fusion for robust speech recognition, proposes a novel method of improving speech recognition accuracy rates. Instead of relying on multiple microphones or audio-visual information fusion, they rely upon two information sources consisting of the acoustic features (the Mel-frequency cepstral coefficients) and the articulatory features.

**4. Concluding comments**

The ultimate goal of realizing a robust and accurate speech recognition system requires extensive research into computationally efficient methods to localize, enhance, and recognize speech. Just as we humans use our two ears and two eyes for speech localization and recognition, artificial speech recognition systems will also require multi-modal and multi-sensor information fusion in order to achieve accurate results in practical situations. As such, the realization of such systems requires extensive research on fusing the information obtained from multiple microphones and cameras, as well as other sensors/sources. The six papers in this special issue provide some rather unique and positive steps towards this ultimate goal; an ultimate goal which may one day enable robust speech recognition to be used in cars, appliances, tablet computers, portable computers, personal digital assistants, and for the disabled.

二、英文翻译

**基于多传感器多源的鲁棒语音处理信息融合——最新技术概述**

Parham Aarabi a，\*，Belur V。

2003年12月3日，加拿大安大略省多伦多市国王学院路10号多伦多大学电气与计算机工程学士学位，

地址：M5S3G4 b Huntsville，AL，USA；2004年2月3日收到订正表格；2004年2月3日接受

摘要

本文概述了鲁棒语音处理的最新进展，并描述了信息融合在进一步实现其目标中的作用。此外，它还具有传统特刊客座社论的功能，对特刊内容进行简要介绍。

@2004爱思唯尔公司版权所有。

关键词：稳健语音处理；多传感器信息融合；语音识别

1.引言

几十年来，人们对使计算机能够处理（即定位、增强和/或识别）语音进行了广泛的研究[1–4,16–20,33]。在此期间，在自动语音识别和处理方面取得了许多进展，产生了各种商业语音识别系统。这些先进的语音识别系统通常通过大量的训练和计算获得较高的识别准确率。然而，高准确率在存在噪声（例如二次对话）的情况下显著降低。因此，噪声环境下的鲁棒语音识别一直是一个热门的研究课题，也是一个尚未解决的挑战[16,18,20,21]。单麦克风语音识别的最新进展（包括概率语音增强和噪声建模技术，如[16,21]）在解决当前语音识别系统的一些鲁棒性问题方面取得了很大进展。但是，使用一个麦克风可以实现的功能是有限的。采用多传感器（包括摄像机和麦克风）的系统具有更强的鲁棒性和更高的识别准确率。融合多个传感器或多个信息源进行鲁棒语音处理是本专题研究的重点。

信息融合方法在语音处理中的应用已经在过去进行了探索[4,6,8]。例如，在文献[4]中，我们使用解释性融合范式探索和分析了一个视听元音识别系统。这项工作不仅说明了融合多模态系统的优点，而且还说明了各个模态对融合结果的相对影响。例如，当信噪比为24分贝时，视觉模态对声学模态的相对影响要大得多，以便做出正确的判断。这清楚地说明了信息融合的好处：在不同的情况下，某些信息源可能是不可靠的，只要存在一种适当的方法对这些信息源进行融合，不可靠的信息源的相对影响将小于那些更可靠的信息源。类似的视听作品支持这项权利要求[4-8,23,34-39]。

2.论文回顾

多年来，视听信息融合一直是一个广泛研究的领域。在[25]中，采用了一种混合的视听特征融合和视听决策融合方法来实现鲁棒的语音识别。在[26]中，一个耦合的隐马尔可夫模型（这是动态信念网络的特例）被用于语音识别的视听信息融合。在[27]中，一种加权判决融合方法被用于视听语音识别。人们还提出了多种其他视听语音识别系统，将麦克风的信号与摄像机记录的图像融合在一起[28–32]。在所有这些系统中，与视觉或声学模态本身相比，融合的视听系统导致字错误率的显著降低。

尽管进行了所有这些研究，但融合多个传感器获得的信息（其中一些传感器可能具有不同的模式）仍然是一个开放的问题，其本质是一个定义模糊的问题[2,8,22]。首先，有一个问题是究竟应该使用哪种类型的传感器。例如，将相机和麦克风结合起来可以产生更大的鲁棒性，因为相机不受噪声的影响，而麦克风不受影响相机的事件（如物体、障碍物、照明等）的影响。然而，融合不同的模式往往更加困难和复杂，这可能导致更大的计算需求。然而，即使存在这种复杂性-有效性的权衡，许多系统都试图融合多模态信息以用于各种应用。示例包括采用单个摄像机和麦克风的视听语音识别系统，从而获得更高的语音识别准确率和更强的噪声鲁棒性[5–15]。其他应用包括视听声音定位[2,3]，其中使用多个摄像头和多个麦克风对扬声器进行视觉定位。正如预期的那样，这些系统说明了与仅音频或仅视频定位系统相比的改进的性能。出现的下一个问题是信息融合的点。例如，由一组麦克风获得的信号值可以在信号电平上被集成（使用诸如波束形成之类的技术），或者，它们可以在每个信道上执行语音识别之后被集成。事实上，这种集成或融合可以发生在语音识别或定位过程中的任何地方。该领域的大部分前期工作都集中在识别过程之前的数据融合上（即，将信号进行整合，得到更清晰的信号，然后用于识别）。这种预识别融合具有计算简单的优点，尽管其最优性不是很清楚。

当使用麦克风阵列时，预识别数据融合是有效的。当相机和麦克风联合使用时，在信号水平上组合它们的数据要困难得多。任何这样的融合都必须涉及到视觉结果到声学结果空间的投影，反之亦然。显然，这个投影可能是一个非常复杂和计算要求很高的任务。然而，早期的视听信息融合[1,11,12]和后期的视听信息融合[14,15]都有尝试。这些例子探讨了复杂性增加和语音识别准确率提高之间的相对折衷。

因此，我们手头的问题是（1）我们融合什么，（2）什么时候（即在识别过程的哪个阶段）我们进行融合，以及（3）我们实际如何进行融合。这是三个相互关联的问题，它们的最终答案可能还需要几年的时间。尽管如此，本期特刊的六篇论文为这些问题提供了一些独特而创新的解决方案。

3.特刊概述

第一篇论文介绍了用于视听语音识别的异步隐马尔可夫模型（HMMs），提出了一种异步HMM结构，它分别应用于从麦克风和摄像机获取的声音和视觉数据。结果表明，该融合模型无论是对视听语音识别还是对纯声语音识别都优于标准的HMM结构，特别是在较低的信噪比下。

第二篇论文还考虑了音视频信息融合问题，对连续音视频数字识别进行决策融合。他们的方法涉及从视觉数据获得的识别结果和从声学数据获得的结果的识别后融合（决策融合）。结果表明，在很低的信噪比（15db）下，它们的系统性能与纯音频系统相似，远优于纯视频系统。在中等信噪比范围（0–15 dB）内，他们的系统性能优于纯音频和纯视频系统。

本文的第三篇论文试图将两个麦克风和两个摄像机融合在信号级，以实现语音增强。这里，与语音相关的视觉数据（即唇部运动）最初被转换为两个相应的声学信号。然后，这些视觉衍生或虚拟声学信号被用作使用从两个麦克风获得的信号进行语音分离的基础.

第四篇论文讨论了一个非常有趣的话题，即在多传感器环境中量化传感器的价值。如果存在多个传感器（如麦克风），则可能不需要将所有传感器用于语音处理应用。但是，为了确定应该使用哪些传感器，必须测量该传感器提供的唯一信息量。为此，本研究提出了一种信息论的方法。

本期特刊的第五篇论文使用了24个麦克风阵列来控制和导航机器人导游。这里，对从24个麦克风收集的数据进行处理，以估计语音源的位置，假设语音源是机器人。这种多传感器的声学定位是用来纠正机器人的位置，并引导它走向下一个目的地。有趣的是，将第五篇论文与Enzo等人[24]提出的系统进行比较。与[24]中的技术不同，机器人上的麦克风阵列用于人-操作员定位和改进人-机器人交互，本研究使用放置在环境中的固定阵列。值得注意的是，所提出的声学机器人定位系统实际上是互补的，使得两者的融合成为未来工作中一个有趣且可能富有成效的方向。

第六篇论文针对鲁棒语音识别中的特征融合问题，提出了一种提高语音识别准确率的新方法。它们不依赖于多个麦克风或视听信息融合，而是依赖于由声学特征（Mel频率倒谱系数）和发音特征组成的两个信息源。

4.总结

实现一个健壮而准确的语音识别系统的最终目标需要对定位、增强和识别语音的有效计算方法进行广泛的研究。就像我们人类用两只耳朵和两只眼睛进行语音定位和识别一样，人工语音识别系统也需要多模态和多传感器的信息融合，才能在实际情况下获得准确的结果。因此，实现这样的系统需要对融合从多个麦克风和照相机以及其他传感器/源获得的信息进行广泛的研究。本期特刊的六篇论文为实现这一最终目标提供了一些相当独特和积极的步骤；最终的目标是有一天使强大的语音识别能够应用于汽车、电器、平板电脑、便携式电脑、个人数字助理和残疾人。

三、源程序

感知层主要代码:

int main(void)

{

u8 key,fontok=0;

u8 \*p;

delay\_init(); //延时函数初始化

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);//设置中断优先级分组为组2：2位抢占优先级，2位响应优先级

uart\_init(115200); //串口初始化为115200

usmart\_dev.init(72); //初始化USMART

LED\_Init(); //初始化与LED连接的硬件接口

KEY\_Init(); //初始化按键

LCD\_Init(); //初始化LCD

W25QXX\_Init(); //初始化W25Q128

tp\_dev.init(); //初始化触摸屏

usart3\_init(115200); //初始化串口3

my\_mem\_init(SRAMIN); //初始化内部内存池

exfuns\_init(); //为fatfs相关变量申请内存

delay\_init(); //延时函数初始化

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);//设置中断优先级分组为组2：2位抢占优先级，2位响应优先级

uart\_init(115200); //串口初始化为115200

LED\_Init(); //初始化与LED连接的硬件接口

KEY\_Init(); //初始化按键

TPAD\_Init(6); //初始化触摸按键

LCD\_Init(); //初始化LCD

W25QXX\_Init(); //初始化W25Q128

VS\_Init(); //初始化VS1053

my\_mem\_init(SRAMIN); //初始化内部内存池

exfuns\_init(); //为fatfs相关变量申请内存

f\_mount(fs[0],"0:",1); //挂载SD卡

f\_mount(fs[1],"1:",1); //挂载FLASH.

POINT\_COLOR=RED;

while(font\_init()) //检查字库

{

LCD\_ShowString(30,50,200,16,16,"Font Error!");

delay\_ms(200);

LCD\_Fill(30,50,240,66,WHITE);//清除显示

}

//fileTest();

qiuqiuni("zxh", "12345678", "125.81.59.212", "12345");

while(1)

{

LED1=0;

printf("Ram Test:0X%04X\r\n",VS\_Ram\_Test());//打印RAM测试结果

VS\_Sine\_Test();

LED1=1;

recoder\_play();

}

}

//VS1053的WAV录音有bug,这个plugin可以修正这个问题

const u16 wav\_plugin[40]=/\* Compressed plugin \*/

{

0x0007, 0x0001, 0x8010, 0x0006, 0x001c, 0x3e12, 0xb817, 0x3e14, /\* 0 \*/

0xf812, 0x3e01, 0xb811, 0x0007, 0x9717, 0x0020, 0xffd2, 0x0030, /\* 8 \*/

0x11d1, 0x3111, 0x8024, 0x3704, 0xc024, 0x3b81, 0x8024, 0x3101, /\* 10 \*/

0x8024, 0x3b81, 0x8024, 0x3f04, 0xc024, 0x2808, 0x4800, 0x36f1, /\* 18 \*/

0x9811, 0x0007, 0x0001, 0x8028, 0x0006, 0x0002, 0x2a00, 0x040e,

};

//激活PCM 录音模式

//agc:0,自动增益.1024相当于1倍,512相当于0.5倍,最大值65535=64倍

void recoder\_enter\_rec\_mode(u16 agc)

{

//如果是IMA ADPCM,采样率计算公式如下:

//采样率=CLKI/256\*d;

//假设d=0,并2倍频,外部晶振为12.288M.那么Fc=(2\*12288000)/256\*6=16Khz

//如果是线性PCM,采样率直接就写采样值

VS\_WR\_Cmd(SPI\_BASS,0x0000);

VS\_WR\_Cmd(SPI\_AICTRL0,8000); //设置采样率,设置为8Khz

VS\_WR\_Cmd(SPI\_AICTRL1,agc); //设置增益,0,自动增益.1024相当于1倍,512相当于0.5倍,最大值65535=64倍

VS\_WR\_Cmd(SPI\_AICTRL2,0); //设置增益最大值,0,代表最大值65536=64X

VS\_WR\_Cmd(SPI\_AICTRL3,6); //左通道(MIC单声道输入)

VS\_WR\_Cmd(SPI\_CLOCKF,0X2000); //设置VS10XX的时钟,MULT:2倍频;ADD:不允许;CLK:12.288Mhz

VS\_WR\_Cmd(SPI\_MODE,0x1804); //MIC,录音激活

delay\_ms(5); //等待至少1.35ms

VS\_Load\_Patch((u16\*)wav\_plugin,40);//VS1053的WAV录音需要patch

}

//初始化WAV头.

void recoder\_wav\_init(\_\_WaveHeader\* wavhead) //初始化WAV头

{

wavhead->riff.ChunkID=0X46464952; //"RIFF"

wavhead->riff.ChunkSize=0; //还未确定,最后需要计算

wavhead->riff.Format=0X45564157; //"WAVE"

wavhead->fmt.ChunkID=0X20746D66; //"fmt "

wavhead->fmt.ChunkSize=16; //大小为16个字节

wavhead->fmt.AudioFormat=0X01; //0X01,表示PCM;0X01,表示IMA ADPCM

wavhead->fmt.NumOfChannels=1; //单声道

wavhead->fmt.SampleRate=8000; //8Khz采样率 采样速率

wavhead->fmt.ByteRate=wavhead->fmt.SampleRate\*2;//16位,即2个字节

wavhead->fmt.BlockAlign=2; //块大小,2个字节为一个块

wavhead->fmt.BitsPerSample=16; //16位PCM

wavhead->data.ChunkID=0X61746164; //"data"

wavhead->data.ChunkSize=0; //数据大小,还需要计算

}

//显示录音时长

//x,y:地址

//tsec:秒钟数.

void recoder\_show\_time(u32 tsec)

{

//显示录音时间

LCD\_ShowString(30,250,200,16,16,"TIME:");

LCD\_ShowxNum(30+40,250,tsec/60,2,16,0X80); //分钟

LCD\_ShowChar(30+56,250,':',16,0);

LCD\_ShowxNum(30+64,250,tsec%60,2,16,0X80); //秒钟

}

//通过时间获取文件名

//仅限在SD卡保存,不支持FLASH DISK保存

//组合成:形如"0:RECORDER/REC20120321210633.wav"的文件名

char \* get\_now\_time(void)

{

int rlen = 0;

u8 \*p;

p=mymalloc(SRAMIN,30);

sendStr((char\*)("{\"from\":1,\"order\":\"010\"}"));

//Show\_Str(30+34,115,156,12,"go into get\_now\_time",12,0); //显示接收到的数据长度

while(!(USART3\_RX\_STA&0X8000));

USART3\_RX\_STA = 0;

while(!(USART3\_RX\_STA&0X8000));

if(USART3\_RX\_STA&0X8000) //接收到一次数据了

{

rlen=USART3\_RX\_STA&0X7FFF; //得到本次接收到的数据长度

USART3\_RX\_BUF[rlen]=0; //添加结束符

printf("%s",USART3\_RX\_BUF); //发送到串口

sprintf((char\*)p,"收到%d字节,内容如下",rlen);//接收到的字节数

LCD\_Fill(30+54,400,239,130,WHITE);

POINT\_COLOR=BRED;

POINT\_COLOR=BLUE;

LCD\_Fill(0,430,239,319,WHITE);

USART3\_RX\_STA=0;

}

sprintf(p,"正在录制：0:/RECORDER/%s",(char\*)USART3\_RX\_BUF);

Show\_Str(30,230,500,12,p,12,0); //显示接收到的数据长?

myfree(SRAMIN,p);

return (char\*)USART3\_RX\_BUF;

}

char \* get\_wav\_time(void)

{

int rlen = 0;

u8 \*p;

p=mymalloc(SRAMIN,30);

sendStr((char\*)("{\"from\":1,\"order\":\"012\"}"));

//Show\_Str(30+34,115,156,12,"go into get\_now\_time",12,0); //显示接收到的数据长度

while(!(USART3\_RX\_STA&0X8000));

USART3\_RX\_STA = 0;

while(!(USART3\_RX\_STA&0X8000));

if(USART3\_RX\_STA&0X8000) //接收到一次数据了

{

rlen=USART3\_RX\_STA&0X7FFF; //得到本次接收到的数据长度

USART3\_RX\_BUF[rlen]='\0'; //添加结束符

printf("%s",USART3\_RX\_BUF); //发送到串口

sprintf((char\*)p,"收到%d字节,内容如下",rlen);//接收到的字节数

LCD\_Fill(30+54,400,239,130,WHITE);

POINT\_COLOR=BRED;

POINT\_COLOR=BLUE;

LCD\_Fill(0,430,239,319,WHITE);

USART3\_RX\_STA=0;

}

myfree(SRAMIN,p);

return (char\*)USART3\_RX\_BUF;

}

// send newest wav file

int send\_wav()

{

char\* time;

int flag1 =0;

u8 \* pname1;

pname1 = mymalloc(SRAMIN,40);

time = mymalloc(SRAMIN,20);

sprintf((char\*)time,"%s",get\_wav\_time());

//printf("get\_wav\_time:%s", time);

//Show\_Str(30+54,500,156,12,(u8\*)time,12,0);

flag1 = atoi(time);

if (flag1 == -1)

return 0;

sprintf((char\*)pname1,"0:RECORDER/%s",time);

Show\_Str(30+54,520,200,16,(u8\*)pname1,16,0);

Show\_Str(30+250,520,156,16,(u8\*)"正在上传",16,0);

//printf("TcpSendWav:%s", pname);

TcpSendWav(pname1);

//Show\_Str(30+54,640,156,12,(u8\*)"--go out sendwav",12,0);

myfree(SRAMIN,pname1);

//Show\_Str(30+54,620,156,12,(u8\*)"--pname1",12,0);

myfree(SRAMIN,time);

//Show\_Str(30+54,660,156,12,(u8\*)"--time",12,0);

return 1;

}

u8\* get\_send\_filename(u8 \* path, u8 \* FileName)

{

FRESULT res;

char \*fn; /\* This function is assuming non-Unicode cfg. \*/

#if \_USE\_LFN

fileinfo.lfsize = \_MAX\_LFN \* 2 + 1;

fileinfo.lfname = mymalloc(SRAMIN,fileinfo.lfsize);

#endif

res = f\_opendir(&dir,(const TCHAR\*)path); //打开一个目录

if (res == FR\_OK)

{

printf("\r\n");

while(1)

{

res = f\_readdir(&dir, &fileinfo); //读取目录下的一个文件

if (res != FR\_OK || fileinfo.fname[0] == 0) break; //错误了/到末尾了,退出

//if (fileinfo.fname[0] == '.') continue; //忽略上级目录

#if \_USE\_LFN

fn = \*fileinfo.lfname ? fileinfo.lfname : fileinfo.fname;

#else

fn = fileinfo.fname;

#endif /\* It is a file. \*/

printf("%s/", path);//打印路径

printf("%s\r\n", fn);//打印文件名

if (strcmp(fn,(char \*)FileName)>0)

{

return (u8 \*)fn;

}

}

}

myfree(SRAMIN,fileinfo.lfname);

return (u8 \*)"";

}

char \* GetWavName()

{

u8\* path = (u8\*)"0:/RECORDER";

char \* time;

char \* p ,\*name;

u8 \* xianshi;

xianshi = mymalloc(SRAMIN,50);

p = mymalloc(SRAMIN,50);

time = get\_wav\_time();

sprintf((char\*)xianshi,"wav time:%s",time);

Show\_Str(30,230,440,16,xianshi,16,0);

sprintf((char\*)p,"0:RECORDER/%s",time);

name = (char \*)get\_send\_filename(path,(u8 \*)p);

myfree(SRAMIN,p);

printf("out:%s",name);

myfree(SRAMIN,xianshi);

return name;

}

//通过时间获取文件名

//仅限在SD卡保存,不支持FLASH DISK保存

//组合成:形如"0:RECORDER/REC20120321210633.wav"的文件名

void recoder\_new\_pathname(u8 \*pname)

{

u8 res;

u16 index=0;

while(index<0XFFFF)

{

sprintf((char\*)pname,"0:RECORDER/REC%05d.wav",index);

res=f\_open(ftemp,(const TCHAR\*)pname,FA\_READ);//尝试打开这个文件

if(res==FR\_NO\_FILE)break; //该文件名不存在=正是我们需要的.

index++;

}

}

//显示AGC大小

//x,y:坐标

//agc:增益值 1~15,表示1~15倍;0,表示自动增益

void recoder\_show\_agc(u8 agc)

{

LCD\_ShowString(30+110,250,200,16,16,"AGC: "); //显示名称,同时清楚上次的显示

if(agc==0)LCD\_ShowString(30+142,250,200,16,16,"AUTO"); //自动agc

else LCD\_ShowxNum(30+142,250,agc,2,16,0X80); //显示AGC值

}

void sendwav(u8 \*pname)

{

FIL\* fmp3;

u16 br;

int j;

u8 res,rval=0;

u8 \*databuf;

u16 i=0;

int count = 0;

fmp3=(FIL\*)mymalloc(SRAMIN,sizeof(FIL));//申请内存

databuf=(u8\*)mymalloc(SRAMIN,1); //开辟512字节的内存区域

if(databuf==NULL||fmp3==NULL)rval=0XFF ;//内存申请失败.

if(rval==0)

{

res=f\_open(fmp3,(const TCHAR\*)pname,FA\_READ);//打开文件

if(res==0)//打开成功.

{

VS\_SPI\_SpeedHigh(); //高速

while(rval==0)

{

res=f\_read(fmp3,databuf,1,(UINT\*)&br);//读出4096个字节

printf("%02x ",databuf[0]);

count ++;

if(count>100)

{

count = 0;

printf("\r\n");

}

if(br!=1||res!=0)

{

rval=0;

break;//读完了.

}

}

f\_close(fmp3);

}else rval=0XFF;//出现错误

}

myfree(SRAMIN,fmp3);

myfree(SRAMIN,databuf);

}

//播放pname这个wav文件（也可以是MP3等）

u8 rec\_play\_wav(u8 \*pname)

{

FIL\* fmp3;

u16 br;

u8 res,rval=0;

u8 \*databuf;

u16 i=0;

fmp3=(FIL\*)mymalloc(SRAMIN,sizeof(FIL));//申请内存

databuf=(u8\*)mymalloc(SRAMIN,512); //开辟512字节的内存区域

if(databuf==NULL||fmp3==NULL)rval=0XFF ;//内存申请失败.

if(rval==0)

{

VS\_HD\_Reset(); //硬复位

VS\_Soft\_Reset(); //软复位

VS\_Set\_All(); //设置音量等参数

VS\_Reset\_DecodeTime(); //复位解码时间

res=f\_open(fmp3,(const TCHAR\*)pname,FA\_READ);//打开文件

if(res==0)//打开成功.

{

VS\_SPI\_SpeedHigh(); //高速

while(rval==0)

{

res=f\_read(fmp3,databuf,512,(UINT\*)&br);//读出4096个字节

i=0;

do//主播放循环

{

if(VS\_Send\_MusicData(databuf+i)==0)i+=32;//给VS10XX发送音频数据

else recoder\_show\_time(VS\_Get\_DecodeTime());//显示播放时间

}while(i<512);//循环发送4096个字节

if(br!=512||res!=0)

{

rval=0;

break;//读完了.

}

}

f\_close(fmp3);

}else rval=0XFF;//出现错误

VS\_SPK\_Set(0); //关闭板载喇叭

}

myfree(SRAMIN,fmp3);

myfree(SRAMIN,databuf);

return rval;

}

char\* record\_voice(int time)

{

u8 res;

int status = 0;

char \* reciveTemp;

int rlen = 0;

u8 \*p;

u8 key;

u8 rval=0;

\_\_WaveHeader \*wavhead=0;

u32 sectorsize=0;

FIL\* f\_rec=0; //文件

DIR recdir; //目录

u8 \*recbuf; //数据内存

u16 w;

u16 idx=0;

u8 rec\_sta=0; //录音状态

//[7]:0,没有录音;1,有录音;

//[6:1]:保留

//[0]:0,正在录音;1,暂停录音;

u8 \*pname=0;

u8 timecnt=0; //计时器

u32 recsec=0; //录音时间

u8 recagc=4; //默认增益为4

while(f\_opendir(&recdir,"0:/RECORDER"))//打开录音文件夹

{

Show\_Str(30,230,240,16,"RECORDER文件夹错误!",16,0);

delay\_ms(200);

LCD\_Fill(30,230,240,246,WHITE); //清除显示

delay\_ms(200);

f\_mkdir("0:/RECORDER"); //创建该目录

}

f\_rec=(FIL \*)mymalloc(SRAMIN,sizeof(FIL)); //开辟FIL字节的内存区域

if(f\_rec==NULL)rval=1; //申请失败

wavhead=(\_\_WaveHeader\*)mymalloc(SRAMIN,sizeof(\_\_WaveHeader));//开辟\_\_WaveHeader字节的内存区域

if(wavhead==NULL)rval=1;

recbuf=mymalloc(SRAMIN,512);

if(recbuf==NULL)rval=1;

pname=mymalloc(SRAMIN,30); //申请30个字节内存,类似"0:RECORDER/REC00001.wav"

p=mymalloc(SRAMIN,30);

if(pname==NULL)rval=1;

if(rval==0) //内存申请OK

{

recoder\_enter\_rec\_mode(1024\*recagc);

while(VS\_RD\_Reg(SPI\_HDAT1)>>8); //等到buf 较为空闲再开始

recoder\_show\_time(recsec); //显示时间

pname[0]=0; //pname没有任何文件名

while(rval==0)

{

switch(status)

{

case 1: //STOP&SAVE

if(rec\_sta&0X80)//有录音

{

wavhead->riff.ChunkSize=sectorsize\*512+36; //整个文件的大小-8;

wavhead->data.ChunkSize=sectorsize\*512; //数据大小

f\_lseek(f\_rec,0); //偏移到文件头.

f\_write(f\_rec,(const void\*)wavhead,sizeof(\_\_WaveHeader),&bw);//写入头数据

f\_close(f\_rec);

sectorsize=0;

}

rec\_sta=0;

recsec=0;

LED1=1; //关闭DS1

LCD\_Fill(30,230,240,246,WHITE); //清除显示,清除之前显示的录音文件名

recoder\_show\_time(recsec); //显示时间

rval = 1;

break;

case 0: //REC/PAUSE

{

rec\_sta|=0X80; //开始录音

reciveTemp = get\_now\_time();

sprintf((char\*)pname,"0:RECORDER/%s.wav",reciveTemp);

recoder\_wav\_init(wavhead); //初始化wav数据

res=f\_open(f\_rec,(const TCHAR\*)pname, FA\_CREATE\_ALWAYS | FA\_WRITE);

if(res) //文件创建失败

{

rec\_sta=0; //创建文件失败,不能录音

rval=0XFE; //提示是否存在SD卡

}else

{

res=f\_write(f\_rec,(const void\*)wavhead,sizeof(\_\_WaveHeader),&bw);//写入头数据

}

}

status = -1;

break;

}

///////////////////////////////////////////////////////////

//读取数据

if(rec\_sta==0X80)//已经在录音了

{

w=VS\_RD\_Reg(SPI\_HDAT1);

if((w>=256)&&(w<896))

{

idx=0;

while(idx<512) //一次读取512字节

{

w=VS\_RD\_Reg(SPI\_HDAT0);

recbuf[idx++]=w&0XFF;

recbuf[idx++]=w>>8;

}

res=f\_write(f\_rec,recbuf,512,&bw);//写入文件

if(res)

{

printf("err:%d\r\n",res);

printf("bw:%d\r\n",bw);

break;//写入出错.

}

sectorsize++;//扇区数增加1,约为32ms

}

}

/////////////////////////////////////////////////////////////

if(recsec!=(sectorsize\*4/125))//录音时间显示

{

LED0=!LED0;//DS0闪烁

recsec=sectorsize\*4/125;

recoder\_show\_time(recsec);//显示时间

if (recsec >=time)

{

status = 1;

}

}

}

}

myfree(SRAMIN,wavhead);

myfree(SRAMIN,recbuf);

myfree(SRAMIN,f\_rec);

myfree(SRAMIN,pname);

myfree(SRAMIN,p);

return (char \*)pname;

}

//录音机

//所有录音文件,均保存在SD卡RECORDER文件夹内.

u8 recoder\_play(void)

{

u8 res;

int xunhuan =1;

int flag = 1;

char \* reciveTemp;

int rlen = 0;

u8 \*p;

u8 key;

u8 rval=0;

\_\_WaveHeader \*wavhead=0;

u32 sectorsize=0;

FIL\* f\_rec=0; //文件

DIR recdir; //目录

u8 \*recbuf; //数据内存

u16 w;

u16 idx=0;

u8 rec\_sta=0; //录音状态

//[7]:0,没有录音;1,有录音;

//[6:1]:保留

//[0]:0,正在录音;1,暂停录音;

u8 \*pname=0;

u8 timecnt=0; //计时器

u32 recsec=0; //录音时间

u8 recagc=4; //默认增益为4

while(f\_opendir(&recdir,"0:/RECORDER"))//打开录音文件夹

{

Show\_Str(30,230,240,16,"RECORDER文件夹错误!",16,0);

delay\_ms(200);

LCD\_Fill(30,230,240,246,WHITE); //清除显示

delay\_ms(200);

f\_mkdir("0:/RECORDER"); //创建该目录

}

f\_rec=(FIL \*)mymalloc(SRAMIN,sizeof(FIL)); //开辟FIL字节的内存区域

if(f\_rec==NULL)rval=1; //申请失败

wavhead=(\_\_WaveHeader\*)mymalloc(SRAMIN,sizeof(\_\_WaveHeader));//开辟\_\_WaveHeader字节的内存区域

if(wavhead==NULL)rval=1;

recbuf=mymalloc(SRAMIN,512);

if(recbuf==NULL)rval=1;

pname=mymalloc(SRAMIN,30); //申请30个字节内存,类似"0:RECORDER/REC00001.wav"

p=mymalloc(SRAMIN,30);

if(pname==NULL)rval=1;

if(rval==0) //内存申请OK

{

recoder\_enter\_rec\_mode(1024\*recagc);

while(VS\_RD\_Reg(SPI\_HDAT1)>>8); //等到buf 较为空闲再开始

pname[0]=0; //pname没有任何文件名

while(rval==0)

{

key=KEY\_Scan(0);

switch(key)

{

case WKUP\_PRES: //STOP&SAVE

LCD\_Fill(30+34,500,300,500+16,WHITE);

while(flag!=0)

{

//xunhuan++;

flag = send\_wav();

delay\_ms(5000);// 避免请求太快导致服务端没完成上一个的转换丢失此次请求

if (flag == 0)

{

Show\_Str(30+34,500,156,16,(u8\*)"已全部上传完毕",16,0);

}

}

LCD\_Fill(30+34,500,500,500+50,WHITE);

break;

case KEY0\_PRES: //REC/PAUSE

LCD\_Fill(30+34,500,300,500+16,WHITE);

record\_voice(3);

LCD\_Fill(30,250,900,400,WHITE);

flag = 1;

break;

case KEY1\_PRES: //AGC-

if(key==WKUP\_PRES)recagc++;

else if(recagc)recagc--;

if(recagc>15)recagc=15; //范围限定为0~15.0,自动AGC.其他AGC倍数

recoder\_show\_agc(recagc);

VS\_WR\_Cmd(SPI\_AICTRL1,1024\*recagc); //设置增益,0,自动增益.1024相当于1倍,512相当于0.5倍

break;

}

///////////////////////////////////////////////////////////

//读取数据

if(rec\_sta==0X80)//已经在录音了

{

w=VS\_RD\_Reg(SPI\_HDAT1);

if((w>=256)&&(w<896))

{

idx=0;

while(idx<512) //一次读取512字节

{

w=VS\_RD\_Reg(SPI\_HDAT0);

recbuf[idx++]=w&0XFF;

recbuf[idx++]=w>>8;

}

res=f\_write(f\_rec,recbuf,512,&bw);//写入文件

if(res)

{

printf("err:%d\r\n",res);

printf("bw:%d\r\n",bw);

break;//写入出错.

}

sectorsize++;//扇区数增加1,约为32ms

}

}else//没有开始录音，则检测TPAD按键

{

if(TPAD\_Scan(0))//如果触摸按键被按下,且pname不为空

{

reciveTemp = record\_voice(3);

/\*

sprintf((char\*)p,"go into tpad %d",timecnt);

Show\_Str(30+34,115,156,12,p,12,0); //显示接收到的数据长度

reciveTemp = get\_now\_time();

Show\_Str(30+34,450,156,12,(u8 \*) reciveTemp,12,0); //显示接收到的数据长度

\*/

/\*

pname = "0:/RECORDER/1620709208";

Show\_Str(30,230,240,16,"播放:",16,0);

Show\_Str(30+40,230,240,16,pname+11,16,0); //显示当播放的文件名字

TcpSendWav(pname); //播放pname

LCD\_Fill(30,230,240,246,WHITE); //清除显示,清除之前显示的录音文件名

recoder\_enter\_rec\_mode(1024\*recagc); //重新进入录音模式

while(VS\_RD\_Reg(SPI\_HDAT1)>>8); //等到buf 较为空闲再开始

recoder\_show\_time(recsec); //显示时间

recoder\_show\_agc(recagc); //显示agc

\*/

/\*

reciveTemp = GetWavName();

sprintf((char\*)p,"send name %d",(u8 \*)reciveTemp);

Show\_Str(30+34,500,156,12,p,12,0); //显示接收到的数据长度

\*/

}

sprintf((char\*)p,"当前状态:空闲 %d",timecnt);

Show\_Str(30+34,500,156,16,p,16,0);

delay\_ms(5);

timecnt++;

if((timecnt%20)==0)LED0=!LED0;//DS0闪烁

}

/////////////////////////////////////////////////////////////

if(recsec!=(sectorsize\*4/125))//录音时间显示

{

LED0=!LED0;//DS0闪烁

recsec=sectorsize\*4/125;

recoder\_show\_time(recsec);//显示时间

}

}

}

Show\_Str(30+34,200,156,12,"go out recoder\_play",12,0);

myfree(SRAMIN,wavhead);

myfree(SRAMIN,recbuf);

myfree(SRAMIN,f\_rec);

myfree(SRAMIN,pname);

myfree(SRAMIN,p);

return rval;

}

服务端主要代码:

public class Test {

//测试主方法

//线程池 懒汉式的单例

private static ExecutorService executorService = Executors.newFixedThreadPool(60);

public static void main(String[] args) throws InterruptedException {

//运行服务器

new Thread(new Runnable() { //通信 文件接收线程

@Override

public void run() {

try {

//Voice\_to\_String.AutoFindTime();

//Voice\_to\_String.JiluTime1(16\*6,"./pcmdata");

ServerBetter.start();

} catch (IOException e) {

e.printStackTrace();

}

}

}).start();

new Thread(new Runnable() { //PCM转str线程

@Override

public void run() {

try {

total\_voice\_tostring.start();

} catch (IOException e) {

e.printStackTrace();

}

}

}).start();

new Thread(new Runnable() { //str处理线程（提取关键字，策略一匹配）

@Override

public void run() {

try {

total\_key\_word.start();

} catch (IOException e) {

e.printStackTrace();

}

}

}).start();

}

}

public final class ServerBetter {

//默认的端口号

private static int DEFAULT\_PORT = 12345;

//单例的ServerSocket

private static ServerSocket server;

//线程池 懒汉式的单例

private static ExecutorService executorService = Executors.newFixedThreadPool(60);

//根据传入参数设置监听端口，如果没有参数调用以下方法并使用默认值

public static void start() throws IOException{

//使用默认值

start(DEFAULT\_PORT);

}

//这个方法不会被大量并发访问，不太需要考虑效率，直接进行方法同步就行了

public synchronized static void start(int port) throws IOException{

if(server != null) return;

try{

//通过构造函数创建ServerSocket

//如果端口合法且空闲，服务端就监听成功

server = new ServerSocket(port);

System.out.println("服务器已启动，端口号：" + port);

String curTime = System.currentTimeMillis() / 1000L + "";

int endtime = Integer.parseInt(curTime);

//get\_recent\_top\_words.get\_top\_words(endtime, 4);

Socket socket;

//通过无线循环监听客户端连接

//如果没有客户端接入，将阻塞在accept操作上。

while(true){

socket = server.accept();

System.out.println(socket+"连接成功");

//当有新的客户端接入时，会执行下面的代码

//然后创建一个新的线程处理这条Socket链路

executorService.execute(new ServerHandler(socket));

}

}finally{

//一些必要的清理工作

if(server != null){

System.out.println("服务器已关闭。");

server.close();

server = null;

}

}

}

}

public class total\_voice\_tostring {

static String APPID="5fed5ca2";

///规范文件路径的命名

static String RECENT\_PATH = "./data";//存储pcm转字符串记录和str数据处理记录的路径

static String PCMDATA\_PATH = "./pcmdata";//存储pcm文件的路径

static String PCMDATA\_JILU = "pcm\_record.txt";//存储pcm记录的文件名

static String STRDATA\_PATH = "./stringdata";//存储string字符串文件的路径

static File\_ways file\_ways = new File\_ways();//声明一个文件处理对象

public static void start() throws IOException{

//使用默认值

System.out.println("pcm转String线程开始");

String recentname ="";

boolean flag;

file\_ways.init1(RECENT\_PATH, PCMDATA\_JILU);

while(true)

{

recentname = file\_ways.get\_recent\_name(RECENT\_PATH,PCMDATA\_JILU);//获取最近的转换记录，方便下一次转换

flag = judge\_zhuanhuan(recentname);

if(flag)

System.out.println(" 等待新的pcm文件进行转换");

}

}

static boolean judge\_zhuanhuan(String recentname)//判断是否这个文件以经是最新文件

{

boolean flag = false;

String str\_re,str\_re1;

int nowtime = Integer.parseInt(recentname);

File file = new File(PCMDATA\_PATH);

String[] fileNameLists = file.list(); //这是不带绝对路径的文件名

for(int i = 0; i < fileNameLists.length; i ++){

if (i == fileNameLists.length) {// 如果正在进行最新pcm数据的转换要等待一段时间，避免转换的是空值

Thread.currentThread();

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

String[] temp = fileNameLists[i].split("\\.");// .号需要进行转义 加上右双斜杠

int x= Integer.parseInt(temp[0]);// 提取到不含后缀的文件名对应的整数

//System.out.println(nowtime + "读取文件顺序" + x);

if(nowtime < x)

{

flag =true;

System.out.println(x+" 开始从pcm转换为String");

str\_re = Voice\_to\_String.convert(APPID, fileNameLists[i],PCMDATA\_PATH);

/\*两次转换优化策略

System.out.println(x+" 第一次转换结果" + str\_re);

str\_re1 = Voice\_to\_String.convert(APPID, fileNameLists[i],PCMDATA\_PATH);//多加一次识别增加准确性

if(str\_re.length()<str\_re1.length())

{

str\_re = str\_re1;

}

System.out.println(x+" 最终转换结果" + str\_re);

\*/

if(str\_re != null)

{

file\_ways.write\_recent\_name(RECENT\_PATH,""+x,PCMDATA\_JILU);//该进行记录的回写

file\_ways.write\_string(STRDATA\_PATH,str\_re,x+".txt");

}

nowtime = x;

System.out.println(x+" 转换完成:" + str\_re);

//转换后内容的写入

}

}

return flag;

}

}

public class total\_key\_word {

static String RECENT\_PATH = "./data";//存储pcm转字符串记录和str数据处理记录的路径

static String STRDATA\_JILU = "str\_record.txt";//存储pcm转字符串记录和str数据处理记录的路径

static String STRDATA\_PATH = "./stringdata";//存储string字符串文件的路径

static String KEYDATA\_PATH = "./keydata";//存储string字符串文件的路径

static String PCMDATA\_JILU = "pcm\_record.txt";//存储pcm记录的文件名

final static String appid = "5fed5ca2";

final static String api\_key = "31c6c1836c5c2946fdf18ca5a9fddc5d";

static boolean flag;

static key\_word kword=new key\_word();//声明一个关键字提取对象

static File\_ways file\_ways = new File\_ways();//声明一个文件处理对象

public static void start() throws IOException{

//使用默认值

System.out.println("关键字提取线程开始");

String recentname ="";

file\_ways.init1(RECENT\_PATH, STRDATA\_JILU);

while(true)

{

recentname = file\_ways.get\_recent\_name(RECENT\_PATH,STRDATA\_JILU);//获取最近的转换记录，方便下一次转换

flag = str\_zhuanhuan(recentname);

if(flag)

System.out.println(" 等待新的str文件进行提取");

}

}

static boolean str\_zhuanhuan(String recentname) throws UnsupportedEncodingException//判断是否这个文件以经是最新文件

{

boolean flag = false;

String str\_re = null;

String pcmrec = file\_ways.get\_recent\_name(RECENT\_PATH,PCMDATA\_JILU);//获取最近的转换记录，方便下一次转换

int texx = Integer.parseInt(pcmrec);

int nowtime = Integer.parseInt(recentname);

File file = new File(STRDATA\_PATH);

String[] fileNameLists = file.list(); //这是不带绝对路径的文件名

if(fileNameLists==null)

return false;

for(int i = 0; i < fileNameLists.length; i ++){

String[] temp = fileNameLists[i].split("\\.");// .号需要进行转义 加上右双斜杠

int x= Integer.parseInt(temp[0]);

//System.out.println(nowtime + "另一个" + x);

//System.out.println(nowtime+" 读取的值为"+pcmrec);

if(texx<=nowtime)//等待2秒，避免pcm还未转化完成

{

Thread.currentThread();//等待是否出现一句话

try {

Thread.sleep(2000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

if(nowtime < x)

{

flag = true;

System.out.println(" 开始对"+x+"进行文字处理");

kword.get\_keyword(fileNameLists[i],appid,api\_key,STRDATA\_PATH,x+".txt",KEYDATA\_PATH);

file\_ways.write\_recent\_name(RECENT\_PATH,""+x,STRDATA\_JILU);//该进行记录的回写

nowtime = x;

//转换后内容的写入

}

}

return flag;

}

}

客户端重点代码:

public class MainActivity extends AppCompatActivity {  
  
 private Toolbar mToolBar;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
 BottomNavigationView navView = findViewById(R.id.nav\_view);  
 // Passing each menu ID as a set of Ids because each  
 // menu should be considered as top level destinations.  
 AppBarConfiguration appBarConfiguration = new AppBarConfiguration.Builder(  
 R.id.navigation\_mood, R.id.navigation\_future, R.id.navigation\_report,R.id.navigation\_cloud)  
 .build();  
 initView();  
 setSupportActionBar(mToolBar);  
 getSupportActionBar().setDisplayShowTitleEnabled(false);  
 NavController navController = Navigation.findNavController(this, R.id.nav\_host\_fragment);  
 NavigationUI.setupWithNavController(mToolBar,navController,appBarConfiguration);  
 NavigationUI.setupWithNavController(navView, navController);  
 }  
  
 private void initView() {  
 mToolBar = findViewById(R.id.toolBar);  
 }  
}

public class CloudFragment extends BaseFragment<FragmentCloudBinding, CloudViewModel> {  
 private Button button,word\_cloud,Receive,disconnect;//定义按钮变量  
 private int progress = 0;  
 Typeface typeface;  
 private Toast toast = null;  
 private ImageView imageView = null;//定义图片变量  
 private ExecutorService mThreadPool;//声明线程池  
 private Socket socket;//声明socket变量  
 private InputStreamReader isr;  
 private String response;  
 private InputStream is;  
 private TCPSocket tcpsocket;  
 private BufferedReader br;  
 private OutputStream outputStream;  
 public Bitmap bmp = null;  
 public Bitmap bmp\_background = null;  
 private Handler handler=null;//用于线程改变控件  
 TongXing tongxing = new TongXing();//声明通信的对象  
 @Override  
 protected void initData() {  
 word\_cloud.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {//接收图片数据并且改变界面图片  
  
 mThreadPool.execute(new Runnable() {  
 @Override  
 public void run() {  
  
  
 Looper.prepare();  
 Toast.makeText(getContext(), "正在生成请稍后", Toast.LENGTH\_SHORT).show();  
  
 //接收图片数据  
 String curTime = System.currentTimeMillis() / 1000L + "";  
 int Timename = Integer.parseInt(curTime);  
 bmp=tongxing.get\_word\_cloud(Timename);  
 bmp\_background = tongxing.get\_word\_cloud\_background();  
 Log.d("TCP2", String.valueOf(bmp));  
 while(bmp==null);  
 Log.d("TCP4", String.valueOf(bmp));  
  
 //显示图片  
 handler.post(runnableUi);//使用主线程改变控件imageview  
  
 Log.d("TCP5", String.valueOf(bmp));  
 Looper.loop();  
  
 }  
 });  
  
  
 }  
 });  
 }  
  
 @Override  
 protected void initView(View root) {  
 imageView = binding.imageView;  
 // 初始化线程池  
 mThreadPool = Executors.newCachedThreadPool();  
 handler=new Handler();  
  
 word\_cloud = binding.wordCloud;  
 }  
  
 @Override  
 protected int getResId() {  
 return R.layout.fragment\_cloud;  
 }  
  
 @Override  
 protected String setTitle() {  
 return getResources().getString(R.string.title\_cloud);  
 }  
 Runnable runnableUi=new Runnable(){  
 @Override  
 public void run() {  
 //更新界面  
 binding.lunkuo.setImageBitmap(bmp\_background);  
 binding.textView.setText("词云背景图");  
  
 imageView.setImageBitmap(bmp);  
 binding.textView1.setText("关键词词云");  
 bmp=null;  
 bmp\_background = null;  
 }  
  
 };  
}

public class MoodFragment extends BaseFragment<FragmentMoodBinding, MoodViewModel> {  
 LineChart chart;  
 String MoodStr = "";  
 private Handler handler=null;//用于线程改变控件  
 private ExecutorService mThreadPool;  
 private TongXing tongxing = new TongXing();//声明通信的对象  
 private List<String> dataset ;  
 @Override  
 protected void initData() {  
 handler=new Handler();  
 NiceSpinner niceSpinner = binding.niceSpinner;  
 binding.textfenxi.setMovementMethod(ScrollingMovementMethod.getInstance());  
 dataset = new LinkedList<>(Arrays.asList("日视图", "周视图", "月视图"));  
 niceSpinner.attachDataSource(dataset);  
 niceSpinner.setTextSize(10);  
 niceSpinner.setBackgroundColor(Color.rgb(255, 242, 226));  
 niceSpinner.addOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {  
  
 switch(i)  
 {  
 case 0:  
 mThreadPool.execute(new Runnable() {  
 @Override  
 public void run() {  
 binding.chart.clearValues();  
 huayitian();  
 Log.d("xinqing","功能待添加");  
 }  
 });  
 break;  
 case 1:  
 mThreadPool.execute(new Runnable() {  
 @Override  
 public void run() {  
 binding.chart.clearValues();  
 huayizhou();  
 Log.d("xinqing","功能待添加");  
 }  
 });  
 break;  
  
 case 2:  
 mThreadPool.execute(new Runnable() {  
 @Override  
 public void run() {  
 binding.chart.clearValues();  
 huayiyue();  
 Log.d("xinqing","功能待添加");  
 }  
 });  
 break;  
 }  
  
 }  
 });  
  
  
 //setTitle("RealtimeLineChartActivity");  
 chart = binding.chart;  
 //chart.setOnChartValueSelectedListener(this);  
  
 // enable description text  
 chart.getDescription().setEnabled(true);  
 mThreadPool = Executors.newCachedThreadPool();//初始化线程池  
 // enable touch gestures  
 chart.setTouchEnabled(true);  
  
 // enable scaling and dragging  
 chart.setDragEnabled(true);  
 chart.setScaleEnabled(true);  
 chart.setDrawGridBackground(false);  
  
 // if disabled, scaling can be done on x- and y-axis separately  
 chart.setPinchZoom(true);  
  
 // set an alternative background color  
 chart.setBackgroundColor(Color.rgb(255, 242, 226));//设置背景颜色  
 //描述文字的操作  
 Description de = new Description();  
 de.setText("情绪状态图");  
 chart.setDescription(de);  
  
 LineData data = new LineData();  
 data.setValueTextColor(Color.BLACK);  
  
 // add empty data  
 chart.setData(data);  
  
 // get the legend (only possible after setting data)  
 Legend l = chart.getLegend();  
  
 // modify the legend ...  
 l.setForm(Legend.LegendForm.LINE);  
 // l.setTypeface(tfLight);  
 l.setTextColor(Color.BLACK);  
  
 //上面的横坐标  
 XAxis xl = chart.getXAxis();  
 // xl.setTypeface(tfLight);  
 xl.setTextColor(Color.BLACK);  
 xl.setPosition(XAxis.XAxisPosition.BOTTOM);  
 xl.setDrawGridLines(false);  
 xl.setTextSize(15f);  
 xl.setAvoidFirstLastClipping(true);  
 xl.setEnabled(true);  
  
  
 YAxis leftAxis = chart.getAxisLeft();  
 // leftAxis.setTypeface(tfLight);  
 leftAxis.setTextColor(Color.BLACK);  
 leftAxis.setTextSize(15f);  
 leftAxis.setAxisMaximum(100f);  
 leftAxis.setAxisMinimum(0f);  
 leftAxis.setDrawGridLines(false);//不绘制Y轴网格线  
  
 YAxis rightAxis = chart.getAxisRight();  
 rightAxis.setEnabled(false);  
  
  
 binding.addone.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 mThreadPool.execute(new Runnable() {  
 @Override  
 public void run() {  
  
 MoodStr = tongxing.get\_mood\_str();  
 handler.post(MoodStrUI);  
  
 }  
 });  
  
 }  
 });  
 mThreadPool.execute(new Runnable() {  
 @Override  
 public void run() {  
  
 huayitian();  
  
 }  
 });  
  
  
 }  
  
 @Override  
 protected void initView(View root) {  
 setMenuId(R.menu.home\_menu);  
  
 }  
  
 @Override  
 protected int getResId() {  
 return R.layout.fragment\_mood;  
 }  
  
 @Override  
 protected String setTitle() {  
 return getResources().getString(R.string.title\_mood);  
 }  
  
 private LineDataSet createSet() {  
  
 LineDataSet set = new LineDataSet(null, "心情指标");  
 set.setAxisDependency(YAxis.AxisDependency.LEFT);  
 set.setColor(ColorTemplate.getHoloBlue());//设置连线颜色  
 set.setCircleColor(ColorTemplate.getHoloBlue());//设置坐标点的颜色  
 set.setLineWidth(4f);//设置连线宽度  
  
 set.setMode(LineDataSet.Mode.HORIZONTAL\_BEZIER);//设置线段是否是平滑  
 set.setCircleRadius(6f);//设置坐标点大小  
 set.setFillAlpha(65);  
 set.setFillColor(Color.BLACK);  
 set.setHighLightColor(Color.rgb(244, 117, 117));  
 set.setValueTextColor(Color.YELLOW);  
 set.setValueTextSize(40f);  
 set.setDrawValues(false);  
 return set;  
 }  
  
 private Thread thread;  
  
 private void feedMultiple() {  
  
 if (thread != null)  
 thread.interrupt();  
  
 final Runnable runnable = new Runnable() {  
  
 @Override  
 public void run() {  
 addEntry();  
 }  
 };  
  
 thread = new Thread(new Runnable() {  
  
 @Override  
 public void run() {  
 for (int i = 0; i < 100; i++) {  
  
 // Don't generate garbage runnables inside the loop.  
 getActivity().runOnUiThread(runnable);  
  
 try {  
 Thread.sleep(200);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 });  
  
 thread.start();  
 }  
 private float covert(int now,float befor,float xishu)//对收到的心情值进行映射 使它们既能体现信息也好看  
 {  
 int zhi =50;  
 float buchang = 0;  
 float xinqingzhi;  
 float huigui;  
 huigui = 50 - befor;  
 if(befor>50)  
 {  
 huigui = befor - 50;  
 if(now<0)  
 buchang =1;  
 }  
 if(now>0)  
 buchang =1;  
 huigui = 1- huigui/50;  
 xinqingzhi = now /2000;  
 xinqingzhi = befor + (huigui+buchang)\*xinqingzhi;  
 return xinqingzhi ;  
 }  
 private void huayizhou()//绘制一周的情绪图  
 {  
 int timecount = 7\*24\*60\*60;  
 float xishu = 2,beforzhi = 50;  
 String result = "";  
 result = tongxing.get\_xinqing(timecount);  
 Log.d("xinqing",result);  
 int linecount;  
 JSONObject tempjson = new JSONObject();  
 JSONObject js =JSONObject.fromObject(result); //string转json格式  
 linecount = js.getInt("linecount");  
 if(linecount == 0)  
 {  
 Looper.prepare();  
 Toast.makeText(getContext(), "暂无周数据", Toast.LENGTH\_SHORT).show();  
 Looper.loop();  
 }  
 Log.d("xinqing",linecount+"");  
 for(int i = 0;i<linecount;i++)  
 {  
 String temp = js.getString(""+i);  
 JSONObject jstemp =JSONObject.fromObject(temp); //string转json格式  
 String zhi = jstemp.getString("zhi");  
 float xinqingzhi ;  
 xinqingzhi = covert(Integer.parseInt(zhi),beforzhi,xishu);  
 beforzhi = xinqingzhi;  
 addzhi(xinqingzhi);  
 try {  
 Thread.sleep(100);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 }  
 private void huayitian()//绘制一天的情绪图  
 {  
  
 int timecount = 1\*24\*60\*60;  
 float xishu = 2,beforzhi = 50;  
 String result = "";  
 result = tongxing.get\_xinqing(timecount);  
 Log.d("xinqing",result);  
 int linecount;  
 JSONObject tempjson = new JSONObject();  
 JSONObject js =JSONObject.fromObject(result); //string转json格式  
 linecount = js.getInt("linecount");  
 if(linecount == 0)  
 {  
 Looper.prepare();  
 Toast.makeText(getContext(), "暂无日数据", Toast.LENGTH\_SHORT).show();  
 Looper.loop();  
 }  
 Log.d("xinqing",linecount+"");  
 for(int i = 0;i<linecount;i++)  
 {  
 String temp = js.getString(""+i);  
 JSONObject jstemp =JSONObject.fromObject(temp); //string转json格式  
 String zhi = jstemp.getString("zhi");  
 float xinqingzhi ;  
 xinqingzhi = covert(Integer.parseInt(zhi),beforzhi,xishu);  
 beforzhi = xinqingzhi;  
 addzhi(xinqingzhi);  
 try {  
 Thread.sleep(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 }  
 private void huayiyue()//绘制一个月的情绪图  
 {  
 int timecount = 30\*24\*60\*60;  
 float xishu = 2,beforzhi = 50;  
 String result = "";  
 result = tongxing.get\_xinqing(timecount);  
 Log.d("xinqing",result);  
 int linecount;  
 JSONObject tempjson = new JSONObject();  
 JSONObject js =JSONObject.fromObject(result); //string转json格式  
 linecount = js.getInt("linecount");  
 if(linecount == 0)  
 {  
 Looper.prepare();  
 Toast.makeText(getContext(), "暂无月数据", Toast.LENGTH\_SHORT).show();  
 Looper.loop();  
 }  
 Log.d("xinqing",linecount+"");  
 for(int i = 0;i<linecount;i++)  
 {  
 String temp = js.getString(""+i);  
 JSONObject jstemp =JSONObject.fromObject(temp); //string转json格式  
 String zhi = jstemp.getString("zhi");  
 float xinqingzhi ;  
 xinqingzhi = covert(Integer.parseInt(zhi),beforzhi,xishu);  
 beforzhi = xinqingzhi;  
 addzhi(xinqingzhi);  
 try {  
 Thread.sleep(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 }  
 private void addzhi(float zhi)  
 {  
 LineData data = chart.getData();  
  
 if (data != null) {  
  
 ILineDataSet set = data.getDataSetByIndex(0);  
 // set.addEntry(...); // can be called as well  
  
 if (set == null) {  
 set = createSet();  
 data.addDataSet(set);  
 }  
  
 data.addEntry(new Entry(set.getEntryCount(), zhi), 0);  
 data.notifyDataChanged();  
  
 // let the chart know it's data has changed  
 chart.notifyDataSetChanged();  
  
 // limit the number of visible entries  
 chart.setVisibleXRangeMaximum(7);//设置横坐标最大显示的值  
 // chart.setVisibleYRange(30, AxisDependency.LEFT);  
  
 // move to the latest entry  
 chart.moveViewToX(data.getEntryCount());  
  
 // this automatically refreshes the chart (calls invalidate())  
 // chart.moveViewTo(data.getXValCount()-7, 55f,  
 // AxisDependency.LEFT);  
 }  
  
 }  
 private void addEntry() {  
  
 LineData data = chart.getData();  
  
 if (data != null) {  
  
 ILineDataSet set = data.getDataSetByIndex(0);  
 // set.addEntry(...); // can be called as well  
  
 if (set == null) {  
 set = createSet();  
 data.addDataSet(set);  
 }  
  
 data.addEntry(new Entry(set.getEntryCount(), (float) (Math.random() \* 40) + 30f), 0);  
 data.notifyDataChanged();  
  
 // let the chart know it's data has changed  
 chart.notifyDataSetChanged();  
  
 // limit the number of visible entries  
 chart.setVisibleXRangeMaximum(15);  
 // chart.setVisibleYRange(30, AxisDependency.LEFT);  
  
 // move to the latest entry  
 chart.moveViewToX(data.getEntryCount());  
  
 // this automatically refreshes the chart (calls invalidate())  
 // chart.moveViewTo(data.getXValCount()-7, 55f,  
 // AxisDependency.LEFT);  
 }  
 }  
  
 protected void saveToGallery() {  
 saveToGallery(chart, "RealtimeLineChartActivity");  
 }  
  
 public void onValueSelected(Entry e, Highlight h) {  
 Log.i("Entry selected", e.toString());  
 }  
  
 public void onNothingSelected() {  
 Log.i("Nothing selected", "Nothing selected.");  
 }  
  
 protected void saveToGallery(Chart chart, String name) {  
 if (chart.saveToGallery(name + "\_" + System.currentTimeMillis(), 70))  
 makeToast("Saving SUCCESSFUL!");  
 else  
 makeToast("Saving FAILED!");  
 }  
  
 Runnable MoodStrUI=new Runnable(){  
 @Override  
 public void run() {  
 //更新界面  
 binding.textfenxi.setText(MoodStr);  
 }  
 };  
}

public class FutureFragment extends BaseFragment<FragmentFutureBinding, FutureViewModel> {  
 public Bitmap bmp = null;  
 private Handler handler=null;//用于线程改变控件  
 String sug\_str = "";  
 int KongjianCount = 3;  
 ArrayList<String> strArray = new ArrayList<String> ();  
 ArrayList<Bitmap> bmpArray = new ArrayList<Bitmap> ();  
 ImageWithTextView[] ImageWithTextView;  
 String juzi = "我姓黄，红绿灯的黄",novel\_juzi;  
  
 public String get\_uri(String str)// 解析并返回网址  
 {  
 String result;  
 String TempStr[] = str.split("\\?\\?");  
 if(TempStr.length<2)  
 return "把握当下";  
 result = TempStr[1];  
 return result;  
  
 }  
 public String get\_title(String str)// 解析并返回网址  
 {  
  
 String result;  
 String TempStr[] = str.split("\\?\\?");  
 if(TempStr.length<3)  
 return "把握当下";  
 result = TempStr[2];  
  
 return result;  
  
 }  
 public String get\_str(String str)// 解析并返回做了长度处理的字符串  
 {  
 int max\_len = 40; //设置最大显示的字符长度  
 String result;  
 String TempStr[] = str.split("\\?\\?");  
 result = TempStr[0];  
 if(max\_len<result.length())  
 {  
 result = result.substring(0,max\_len);  
 result += "...";  
 }  
  
 return result;  
  
 }  
 @Override  
 protected void initData() {  
 ExecutorService mThreadPool;//声明线程池  
 mThreadPool = Executors.newCachedThreadPool();  
 handler=new Handler();  
 TongXing tongxing = new TongXing();//声明通信的对象  
 binding.juzi.setMovementMethod(ScrollingMovementMethod.getInstance());  
 String tempJuzi = "";//接收到的句子  
 binding.fenge.setMovementMethod(ScrollingMovementMethod.getInstance());  
// binding.imageWithText.setTitle("Title");  
// binding.imageWithText.setContent("Content");  
 ImageWithTextView =new ImageWithTextView[]{binding.imageWithText1,binding.imageWithText2,binding.imageWithText3};  
  
 mThreadPool.execute(new Runnable() {  
 @Override  
 public void run() {  
  
 strArray.clear();  
 bmpArray.clear();  
 //接收图片数据  
 String curTime = System.currentTimeMillis() / 1000L + "";  
 int Timename = Integer.parseInt(curTime);  
 String tempJuzi=tongxing.get\_juzi("");  
 String types = tongxing.get\_future(Timename);  
 types = types.substring(1,types.length()-1);  
 String TempStr[] = types.split(",");  
 int len = 5;  
 if(TempStr.length >KongjianCount)  
 len = TempStr.length;  
 for(int i = 0;i<len;i++)  
 {  
 String str\_temp = TempStr[i%KongjianCount];  
 String final\_str[] = str\_temp.split(":");  
  
 if(final\_str[1].length() > 1)  
 {  
 String type = final\_str[1];  
 sug\_str = tongxing.get\_sug(type);  
 strArray.add(sug\_str);  
 bmp = tongxing.get\_sug\_photo(type);  
 bmpArray.add(bmp);  
 }  
  
 }  
 juzi=tongxing.get\_juzi("");  
 handler.post(JuZi); // 更新笑话  
 handler.post(runnableUi);// 更行推荐的图片和文字  
  
 // 分割的显示句子  
 //novel\_juzi=tongxing.get\_novel\_juzi();  
 //handler.post(adjust\_novel);  
  
  
  
 }  
 });  
 binding.fenge.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 mThreadPool.execute(new Runnable() {  
 @Override  
 public void run() {  
  
 juzi=tongxing.get\_juzi("");  
 handler.post(JuZi);  
  
 }  
 });  
  
 }  
 });  
 /\*  
 binding.fenge.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 mThreadPool.execute(new Runnable() {  
 @Override  
 public void run() {  
  
 novel\_juzi=tongxing.get\_novel\_juzi();  
 handler.post(adjust\_novel);  
  
 }  
 });  
  
 }  
 });  
 \*/  
 binding.imageWithText1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 String uri = "https://www.cnblogs.com/kori/p/12496197.html";  
 Intent intent=new Intent(Intent.ACTION\_VIEW, Uri.parse(uri));  
 startActivity(intent);  
  
 }  
 });  
 for(int i = 0;i<KongjianCount;i++)  
 {  
 int finalI = i;  
 ImageWithTextView[i].setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 String uri = get\_uri(strArray.get(finalI));  
 Intent intent=new Intent(Intent.ACTION\_VIEW, Uri.parse(uri));  
 startActivity(intent);  
  
 }  
 });  
 }  
 }  
  
 @Override  
 protected void initView(View root) {  
  
 }  
  
 @Override  
 protected int getResId() {  
 return R.layout.fragment\_future;  
 }  
  
 @Override  
 protected String setTitle() {  
 return getResources().getString(R.string.title\_future);  
 }  
 Runnable runnableUi=new Runnable(){  
 @Override  
 public void run() {  
 //更新界面  
 for(int i = 0;i<KongjianCount;i++)  
 {  
 ImageWithTextView[i].setContent(get\_str(strArray.get(i)));  
 if(bmpArray.get(i) != null)  
 ImageWithTextView[i].setImageResource(bmpArray.get(i));  
 ImageWithTextView[i].setTextResource(get\_title(strArray.get(i)));  
 }  
  
 }  
 };  
  
 Runnable JuZi=new Runnable(){  
 @Override  
 public void run() {  
 //更新界面  
 binding.juzi.setText(juzi);  
 }  
 };  
 Runnable adjust\_novel=new Runnable(){  
 @Override  
 public void run() {  
 //更新界面  
 binding.fenge.setText(novel\_juzi);  
 }  
 };  
}

public class ReportFragment extends BaseFragment<FragmentReportBinding, ReportViewModel> {  
  
 private ViewPagerAdapter adapter;  
 String ReportStr = "";  
 String ReportJuzi = "";  
 TextView Report\_juzi;  
 Map<String, String> cixin\_map = new HashMap<String,String>(); // 用于存储词性的英文和中文对照关系  
 int Index = 0;  
 double Min\_Zhanshi = 0.027; // 饼状图能展示的最小比例  
 int WaitTime = 3000;  
 int TextCount = 3;// 待修改text的个数  
 List<TextView> text\_views = new ArrayList<>();  
 private Handler handler=null;//用于线程改变控件  
  
 @Override  
 protected void initData() {  
  
  
 }  
  
 @Override  
 protected void initView(View root) {  
 text\_views.clear();  
 adapter = new ViewPagerAdapter();  
 binding.viewPager.setAdapter(adapter);  
 ExecutorService mThreadPool;//声明线程池  
 handler=new Handler();  
 TongXing tongxing = new TongXing();//声明通信的对象  
 mThreadPool = Executors.newCachedThreadPool();  
 List<View> views = new ArrayList<>();  
 init\_map();  
 View page1 = LayoutInflater.from(getContext()).inflate(R.layout.page\_text1,null);  
 View page2 = LayoutInflater.from(getContext()).inflate(R.layout.page\_text2,null);  
 View page3 = LayoutInflater.from(getContext()).inflate(R.layout.page\_text3,null);  
 View page4 = LayoutInflater.from(getContext()).inflate(R.layout.page\_pie,null);  
  
 TextView text1 = page1.findViewById(R.id.text);  
 TextView text2 = page2.findViewById(R.id.text);  
 TextView text3 = page3.findViewById(R.id.text);  
 text\_views.add(page1.findViewById(R.id.text));  
 text\_views.add(page2.findViewById(R.id.text));  
 text\_views.add(page3.findViewById(R.id.text));  
 PieChart pieChart = page4.findViewById(R.id.pieChart);  
 Report\_juzi = page4.findViewById(R.id.reportJuzi);  
 mThreadPool.execute(new Runnable() {  
 @Override  
 public void run() {  
  
 Looper.prepare();  
 Toast.makeText(getContext(), "正在生成请稍后", Toast.LENGTH\_SHORT).show();  
  
 String curTime = System.currentTimeMillis() / 1000L + "";  
 int Timename = Integer.parseInt(curTime);  
 ReportStr=tongxing.get\_report(Timename);  
 ReportJuzi = tongxing.get\_novel\_juzi();  
 handler.post(adjust\_text);  
 handler.post(JuZi);  
  
  
  
  
 // 进行饼状图的生成  
 String TempStr[] = ReportStr.split("\\.");  
 String pie\_data = TempStr[3];  
 double total\_num = Double.parseDouble(TempStr[4]);  
 String single\_date[] = pie\_data.split(",");  
 pieChart.setEntryLabelTextSize(20);  
 List<PieEntry> pieEntries = new ArrayList<>();  
 for(int i=0;i<single\_date.length;i++)  
 {  
 String key\_value[] = single\_date[i].substring(1,single\_date[i].length()-1).split(":");  
 String dic\_name = key\_value[0];  
 int dic\_count = Integer.parseInt(key\_value[1]);  
 if(dic\_count/total\_num > Min\_Zhanshi)  
 pieEntries.add(new PieEntry(dic\_count,GetChinese(dic\_name)));  
 }  
  
 PieDataSet dataSet = new PieDataSet(pieEntries,"词性详细统计图");  
 dataSet.setColors(ColorTemplate.VORDIPLOM\_COLORS);  
 dataSet.setValueLineColor(Color.BLACK);  
 dataSet.setYValuePosition(PieDataSet.ValuePosition.OUTSIDE\_SLICE);  
 dataSet.setValueTextSize(13);  
  
 PieData data = new PieData();  
 data.setValueTextColor(Color.WHITE);  
 data.setDataSet(dataSet);  
 pieChart.setData(data);  
 pieChart.notifyDataSetChanged();  
 pieChart.setTouchEnabled(false);  
  
 Looper.loop();  
  
 }  
 });  
  
 Report\_juzi.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 mThreadPool.execute(new Runnable() {  
 @Override  
 public void run() {  
  
 ReportJuzi=tongxing.get\_novel\_juzi();  
 handler.post(JuZi);  
  
 }  
 });  
  
 }  
 });  
 //while(ReportStr.split("\\.").length!=4);// 等待数据的接收完成 不然后面访问会出错  
  
  
 views.add(page1);  
 views.add(page2);  
 views.add(page3);  
 views.add(page4);  
  
 adapter.setViews(views);  
 adapter.notifyDataSetChanged();  
 binding.viewPager.setOnPageChangeListener(new ViewPager.OnPageChangeListener() {  
 @Override  
 public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {  
  
 }  
  
 @Override  
 public void onPageSelected(int position) {  
 clearDot();  
 String TempStr[] = ReportStr.split("\\.");  
 AnimationSet animationSet = new AnimationSet(true);  
 AlphaAnimation alphaAnimation = new AlphaAnimation(0, 1);  
 switch (position){  
 case 0:  
 startAnimation(text1);  
  
 Index =0;  
 text1.setText(TempStr[Index]);  
 alphaAnimation.setDuration(WaitTime);  
 animationSet.addAnimation(alphaAnimation);  
 text1.startAnimation(animationSet);  
  
  
 binding.dot1.setImageResource(R.drawable.shape\_circle\_black);  
 break;  
 case 1:  
 startAnimation(text2);  
  
 Index = 1;  
 text2.setText(TempStr[Index]);  
 alphaAnimation.setDuration(WaitTime);  
 animationSet.addAnimation(alphaAnimation);  
 text2.startAnimation(animationSet);  
  
  
 binding.dot2.setImageResource(R.drawable.shape\_circle\_black);  
 break;  
 case 2:  
 startAnimation(text3);  
  
 Index = 2;  
 text3.setText(TempStr[Index]);  
 alphaAnimation.setDuration(WaitTime);  
 animationSet.addAnimation(alphaAnimation);  
 text3.startAnimation(animationSet);  
  
  
 binding.dot3.setImageResource(R.drawable.shape\_circle\_black);  
 break;  
 case 3:  
 startAnimation(pieChart);  
 binding.dot4.setImageResource(R.drawable.shape\_circle\_black);  
 break;  
 }  
 }  
  
 @Override  
 public void onPageScrollStateChanged(int state) {  
  
 }  
 });  
 }  
 public String GetChinese(String str)  
 {  
 String key = str.substring(1,str.length()-1);// 删除多余的两个引号  
 String result = cixin\_map.get(key);  
 if (result == null)  
 {  
 result =str;  
 }  
 return result;  
 }  
 private void clearDot() {  
 binding.dot1.setImageResource(R.drawable.shape\_circle\_gray);  
 binding.dot2.setImageResource(R.drawable.shape\_circle\_gray);  
 binding.dot3.setImageResource(R.drawable.shape\_circle\_gray);  
 binding.dot4.setImageResource(R.drawable.shape\_circle\_gray);  
 }  
  
  
 @Override  
 protected int getResId() {  
 return R.layout.fragment\_report;  
 }  
  
 @Override  
 protected String setTitle() {  
 return getResources().getString(R.string.title\_report);  
 }  
  
 public void startAnimation(View view){  
 PropertyValuesHolder alphaHolder = PropertyValuesHolder.ofFloat("alpha",0f,1f);  
 ObjectAnimator objectAnimator = ObjectAnimator.ofPropertyValuesHolder(view,alphaHolder);  
 objectAnimator.setDuration(200);  
 objectAnimator.start();  
  
 }  
 public void init\_map()  
 {  
 cixin\_map.put("v","动词");  
 cixin\_map.put("r","代词");  
 cixin\_map.put("n","名词");  
 cixin\_map.put("ns","地名");  
 cixin\_map.put("wp","标点");  
 cixin\_map.put("k","后缀");  
 cixin\_map.put("h","前缀");  
 cixin\_map.put("u","助词");  
 cixin\_map.put("c","连词");  
 cixin\_map.put("p","介词");  
 cixin\_map.put("d","副词");  
 cixin\_map.put("q","量词");  
 cixin\_map.put("nh","人名");  
 cixin\_map.put("m","数词");  
 cixin\_map.put("e","语气词");  
 cixin\_map.put("b","状态词");  
 cixin\_map.put("a","形容词");  
 cixin\_map.put("nd","方位词");  
 cixin\_map.put("ni","处所词");  
 cixin\_map.put("o","拟声词");  
 cixin\_map.put("nt","时间词");  
 cixin\_map.put("nz","其他专名");  
 cixin\_map.put("nl","机构团体");  
 cixin\_map.put("i","成语");  
 cixin\_map.put("j","缩写词");  
 cixin\_map.put("ws","外来词");  
 cixin\_map.put("g","词素");  
 cixin\_map.put("x","非词位");  
  
 }  
  
 Runnable JuZi=new Runnable(){  
 @Override  
 public void run() {  
 //更新界面  
 Report\_juzi.setText(ReportJuzi);  
 }  
 };  
 Runnable adjust\_text=new Runnable(){  
 @Override  
 public void run() {  
 //更新界面  
 String TempStr[] = ReportStr.split("\\.");  
 for(int i =0;i< TextCount;i++)  
 {  
 text\_views.get(i).setText(TempStr[i]);  
 }  
 Report\_juzi.setText(ReportJuzi);  
 }  
 };  
 Runnable adjust\_text\_zhiding=new Runnable(){  
 @Override  
 public void run() {  
 //更新界面  
 String TempStr[] = ReportStr.split("\\.");  
 text\_views.get(Index).setText(TempStr[Index]);  
 }  
 };  
}