**2020年新工科联盟-Xilinx暑期学校团队项目设计文档**

**设计文稿提交格式**

**(Project Paper Submission Template)**

|  |  |
| --- | --- |
| **设计作品名称** | ESP32语音识别 |
| **参赛队员姓名、学号、学校及所在院系** | 王瑞燕  2018112758  西南交通大学  信息科学与技术学院  通信工程专业 |
| **Github链接** | https://github.com/wry2018112758/wry2018112758.git |

**第一部分**

**设计概述 /Design Introduction**

自动语音识别 Automatic Speech Recognition（ASR），其目标是将人类的语音中的词汇内容转换为计算机可读的输入，例如按键、二进制编码或者字符序列。这里我们采用百度的语音识别技术 。百度语音识别通过 REST API 的方式给开发者提供一个通用的 HTTP 接口，基于该接口，开发者可以轻松的获取语音识别能力。

**第二部分**

**系统组成及功能说明 /System Construction & Function Description**

这里使用语音识别服务 REST API 的方法。总的来说呢，就是一个http的post请求，里面传json的格式数据。下面是部分验证代码：

**import** json

**import** httplib

**import** base64

f**=**open('8k.amr','rb')

data**=**f**.**read(6340)

access\_token**=**"24.44810154581d4b7e8cc3554c90b949f0.2592000.1505980562.282335-10037482"

speech**=**base64**.**b64encode(data)

length**=**6340

params**=**{'format':"amr","rate":8000,"channel":1,"cuid":"eps32\_frankie","token":access\_token,"speech":speech,"len":length}

a**=**json**.**dumps(params)

**print** a

conn **=** httplib**.**HTTPConnection("vop.baidu.com",80)

conn**.**request("GET","/server\_api",a)

response**=**conn**.**getresponse()

**print** response**.**status,response**.**reason

**print** response**.**read()

返回的结果是：

200 OK

{"corpus\_no":"6457400763920513577","err\_msg":"success.","err\_no":0,"result":["一二三四五六七八九十，"],"sn":"455838530761503480775"}

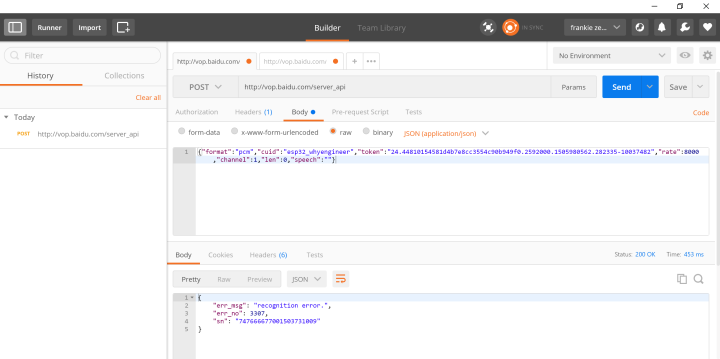
目前支持的传输的音频格式为8k或者16k，单声道。传输格式有wav格式或者amr压缩两种。这个token需要到百度的开发者平台去申请。具体的api信息可以到这里看：[百度语音识别-开发文档](https://link.zhihu.com/?target=http%3A//yuyin.baidu.com/docs/asr/57" \t "_blank)

下面开始进入重头戏，如何让这个简单的功能能在ESP32这个只有520kRAM的MCU上运行，主要的难点有两个：

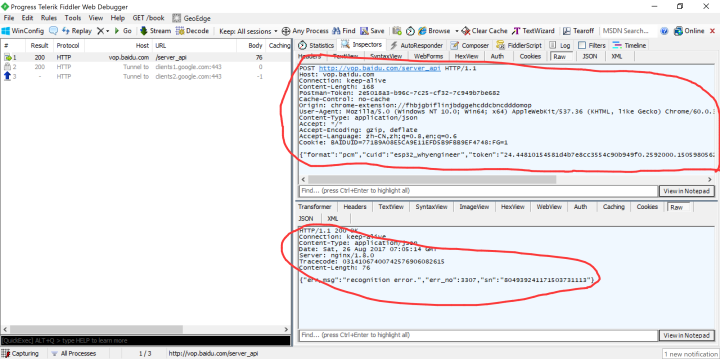
1.只有520kRAM,我们8k的采样率，16bits。录1s需要存的ram是8000\*2B；然后数据还需要base64编码，封装成json格式，如何尽量减小ram的使用。（最后实现了只用4Kram可以完成，但速度会低一些）

2.没有很好轻量级的嵌入式c的http库。

首先解决第二个问题，由于需要的功能很简单，决定采用直接socket操作，模拟http请求。http本身是居于TCP的，这里不会去叫http的帧协议，我们只需要用到两个工具，postman和fiddler，postman是一个很方便的模拟http请求的工具，fiddler则是一个抓包工具。



这边我发了一个post请求，然后看一下抓包工具的结果：



红色两个框里面就是收到的raw数据，也就是一个HTTP请求就是往tcp里写这些东西。

那么我们开始模拟下这个http协议

sprintf(p\_request, "POST %s HTTP/1.0\r\nHost: %s:%d\r\nContent-Length: %d\r\nUser-Agent: ESP32\r\nAccept: \*/\*\r\n\r\n", url**->**path, url**->**host, url**->**port,length)

这个就是我们首先要写的东西。这个里面有一个很关机的参数是Content-Length：length，代表的是payload段的长度，因为我们没有足够大的ram，能等到语音采集完后再计算长度，我们只能采集一段语言，发送一段语言，所以这里只能写一个比较大的长度，然后如果实际小于这个长度，后面补0即可。

我们需要发送json格式的数据，刚开始用cJSON这样写的：

**static** **char\*** **baidu\_rest\_input**(**char\*** speech,uint32\_t len){

cJSON**\*** root**=**NULL;

root**=**cJSON\_CreateObject();

**if**(root**==**NULL){

ESP\_LOGI(TAG,"cjson root create failed\n");

**return** NULL;

}

cJSON\_AddStringToObject(root,"format","wav");

cJSON\_AddStringToObject(root,"cuid","esp32\_whyengineer");

cJSON\_AddStringToObject(root,"token",access\_token);

cJSON\_AddNumberToObject(root, "rate", 8000);

cJSON\_AddNumberToObject(root, "channel", 1);

cJSON\_AddNumberToObject(root, "len", len);

cJSON\_AddStringToObject(root,"speech",speech);

*// if(!strncmp(item,"vbus",4)){*

*// CALIB\_DEBUG("%s\n",info);*

*// cJSON\_AddStringToObject(root,"content",info);*

*// }*

**char\*** out **=** cJSON\_PrintUnformatted(root);

*//send(client,out,strlen(out),MSG\_WAITALL);*

*//printf("handle\_return: %s\n", out);*

cJSON\_Delete(root);

**return** out;

}

由于没有足够大的ram，上面的函数也是用不了的，也只能讲json格式拆开来发

**const** **char\*** stream\_head**=**"{\"format\":\"wav\",\"cuid\":\"esp32\_whyengineer\",\"token\":\"24.44810154581d4b7e8cc3554c90b949f0.2592000.1505980562.282335-10037482\",\"rate\":8000,\"channel\":1,\"speech\":\"";

**const** **char\*** stream\_len**=**"\",\"len\":";

**const** **char\*** stream\_tail**=**"}";

然后再将真是的数据插在中间发就可以了。

到这里，就已经可以和百度的服务器交互了，然后http的返回用了这个库来parser [nodejs/http-parser](https://link.zhihu.com/?target=https%3A//github.com/nodejs/http-parser" \t "_blank)。

接下来是音频数据的采集，esp32的i2s接口采用dma+循环buf，可以更大的提高cpu的效率。首先是语音端点检测（VAD)，用来判断什么时候语言信号开始和结束。传统简单的方法有短时能力检测和过零率检测。

**短时能量：**一段时间的信号的绝对值的累计，当然平方相加更符合能量这个意义。

**过零率：**正负的变化次数。

**static** uint8\_t **vad\_check**(int16\_t**\*** data,uint32\_t lenght){

**int** sum**=**0;

**int** delta\_sum**=**0;

**for**(**int** i**=**0;i**<**lenght**-**1;i**++**){

**if**(sign(data[i])**^**sign(data[i**+**1]))

delta\_sum**++**;

*//printf("%d\n",value[i] );*

**if**(data[i]**<**0)

sum**-=**data[i];

**else**

sum**+=**data[i];

*//value=(((int16\_t)sample\_data[i\*2])<<8)&sample\_data[i\*2+1];*

}

**if**(sum**>**150000**&&**delta\_sum**<**150){

*//gpio\_set\_level(GPIO\_OUTPUT\_IO\_0,1);*

**return** 1;

}

**else**{

*//gpio\_set\_level(GPIO\_OUTPUT\_IO\_0,0);*

**return** 0;

}

}

具体逻辑是：一旦检测到语言开始输入，将音频文件以wav文件的格式文件写入sd卡保存，检测到语音声音输入完成，开始发送http请求，分段发送base64编码的数据。

**第三部分**

**完成情况及性能参数 /Final Design & Performance Parameters**

Vivado2018.3

ESp32 SEA

Spartan7xcs15

**第四部分**

**总结 /Conclusions**

经过几天的实习，今日到了结尾的时候，在这一次实习中我学到很多，也感悟到很多，回顾实习生活，感触很深，收获颇丰。经过实习我更加认识到实践是检验真理的唯一标准，只学不实践，那么所学的就等于零，理论应当与实践相结合。另一方面，实践可为以后找工作打基础。我感觉实践是大学生活的第二课堂，是知识常新和发展的源泉，是检验真理的试金石，也是大学生锻炼成长的有效途径。一个人的知识和本事仅有在实践中才能发挥作用，才能得到丰富、完善和发展。大学生成长，就要勤于实践，将所学的理论知识与实践相结合一齐，在实践中继续学习，不断总结，逐步完善，有所创新，并在实践中提高自我的各方面知识、本事、技术等因素融合成的综合素质和本事，为自我事业的成功打下良好的基础。

这次的实习也让我更加清楚的了解了ESP32的各个功能。ESP32是Espressif乐鑫信息科技推出的一块WiFi芯片。拥有40nm工艺、双核32位MCU、2.4GHz双模Wi-Fi和蓝牙芯片、主频高达230MHz,计算能力可达600DMIPS。涵盖精细分辨时钟门控、省电模式和动态电压调整等特征。它集成了天线和射频巴伦，功率放大器，低噪声放大器，滤波器和电源管理模块等元器件，性能稳定，易于制造，工作温度范围从-40℃到125℃。支持多种通信协议，如：I2C. I2S. SPI. UART. CAN.多种调节管理模式：Active模式、Modem-sleep模式、Light-sleep模式、Deep-sleep模式、Hibernation模式。可根据不同需求，调节所需方案。

原本我认为FPGA和单片机一样，它是由一个超级经典的硬核架构起来的IC，通过编写程序来实现FPG A对外围的控制。但是通过这次实习，我对FPGA有了重新的认识：首先，FPGA是纯硬件，现在我所写的‘程序’也就不能称之为程序，如果非要给它起一个名字，我感觉‘代码’也许是比较恰当的；其次，单片机和FPGA实现信息处理发出控制信号的方式不同，单片机和嵌入式是通过主函数或者操作系统实现任务调度的途径来响应各种外部条件触发，并通过‘软件’输出相应的状态来实现电子系统的正常工作，然而FPGA是通过纯硬件来实现各种激励的相应的，与单片机实现控制的根本方法不同。最后，单片机和嵌入式的软件设计会有盲点，再快的软件相应在现实应用中也会有出现盲点的时候，简单的说就是系统不能及时响应或者漏掉某些激励的现象，然而硬件不会，它以强大的并行方式来响应所有的信号激励，但硬件也不是万能的，FPGA这类的纯硬件最大的克星就是亚稳态以及竞争冒险等等，通俗的讲就是输出及其短暂的不稳定现象，虽然不稳定状态出的时间及其短，甚至到了皮秒级别（1秒等于10的十二次方皮秒）但也就是这么短暂的不稳定也可能会给设计带来致命的危险。这是有硬件的特性决定的，是在设计中永远无法避免的，对于冒险和亚稳态，我们只能想尽一切办法去解决他们。一个信号处理逻辑机制优劣将决定一个项目的好坏。