南昌大学2018-2019学年夏季学期

**项目总结报告**

|  |  |  |
| --- | --- | --- |
| **项目名称** | **：** | Xilinx PYNQ Z2上基于语音识别的标签打印机 |
| **项目类别** |  | 专业实习 |
| **学院名称** | **：** | 信息工程学院 |
| **项目负责人** | **：** | \*\* |
| **导师姓名** | **：** | \*\*\* |

**教务处制**

**二〇 年 月**

## 一 项目背景

该项目旨在检验学生编程的实际操作能力和软硬件结合的实际系统的搭建能力。首先，由指导老师选定本项目的主题为办公用标签打印机。目前，市面上常见的标签打印机都配有键盘，需要用户手动输入、编辑信息，然后打印，如图1所示。在本次项目中，我们希望就标签打印机的输入方式上进行创新，设计并实现一个基于语音识别的标签打印机原型，并计划添加对emoji表情的支持。



图1. 市面上常见的标签打印机

## 二 项目成员

项目指导老师：南昌大学GOOD实验室 \*\*\*教授

项目负责人：计算机177班 \*\*

项目组成员：（按照班级序号排序）

际銮书院17级综合实验班 \*\*\*

计算机171班 \*\*

计算机171班 \*\*

计算机175班 \*\*\*

## 三 项目分析

在向老师请教过后，结合组内成员共同探讨分析，我们将项目分为以下四个部分：

1. 语音识别
2. 文字处理与渲染
3. 预览显示
4. 目标打印

其中各个部分的功能具体分析如下：

**语音识别**

**这是本项目中的主要创新部分。语音识别作为一项有着数十年发展历史的技术，其目标在于提供不用人力实现的语音转文字功能。以往的语音识别主要是基于人工手动优化的复杂函数，可以在特定的环境、语言条件下产生较好的效果，但是由于开发者无法完整而全面地捕获并有效利用所有的特征信息，导致在预料之外的环境下有着不尽人意的识别率。近年来，随着深度学习的飞速发展，神经网络的出现为语音识别提供了一种新的思路：开发者们只需要设计特征识别处理结构，然后将数据交由神经网络处理，最后选出有效的结构，进行合理组合即可。并且，由于神经网络基于数据的可学习性，可以通过使用特定场景下的样例数据进行训练，从而获得特定场景下的优化性能。目前，已有不少基于语音识别的成功产品，大者如微软小娜(Cortana)，QQ小冰等语音聊天助手，小者则有目前部分输入法自带的语音输入功能。在这个部分中，我们需要实现从录音设备（麦克风）的语音读入，然后使用一定的语音识别程序，识别出使用者所录入的声音，然后交由下一步操作。**

**文字处理与渲染**

**这是本项目中的次要创新部分。这一部分负责将语音识别生成的结果进行再处理，并且将处理结果转化为以下两步（预览显示和目标打印）可以使用的格式。传统的标签打印机仅支持文字和少量自行设计的图像作为打印内容。为了提高打印图像的实用程度，我们计划使用识别度更高的emoji表情，并将其自动插入打印内容中，以提升打印内容的丰富度。**

**预览显示**

**考虑到实际使用时存在的资源节约问题，我们和传统标签打印机一样，提供打印预览功能。在这一部分中，我们将把上一步处理过的文字按照打印时的排版进行显示，以向用户确认打印内容无误。预期的显示设备是LED背光的液晶点阵显示屏，由于开发板上有众多硬件交互接口，对于点阵屏的操作，我们计划使用PYNQ Z2开发板上的串口实现。此外，我们还计划添加确认和取消两个按钮，用于提供用户与打印机程序间的操作指令交互。**

**目标打印**

**这是本项目中的关键部分，也是用户使用该打印机完成一次操作时的最后一步。在用户对预览显示确认无误后，打印程序将按照收到的渲染结果，重新按照打印机指令格式编码，交付打印机打印用户所需求的标签。根据目标产品的需求，我们在互联网上进行了简单的搜索，发现最佳的组合是热敏打印机和热敏标签纸。相对于喷墨和激光打印机来说，热敏打印机对于环境的要求更低，功率更小，体积也更小，与我们制作小型便携设备的目标更加契合。**

**根据初步了解，Xilinx PYNQ Z2开发板使用Python语言作为其板上开发的推荐语言，并已经编写好了大量Python库以提供包括常用软件与硬件控制在内的使用高级语言的操控，这为我们进行开发提供了许多便捷之处。我们计划使用Python作为我们的主要开发语言，必要时结合C/C++语言开发，并通过调用库的方式对两者进行结合。**

## 四 项目规划

### 项目目标

在规定期限内，进行相关知识的学习，并完成一个可以正常使用的语音识别标签打印机的设计、组装与测试。

### 任务分配

**由于在项目开始时，项目组的各位成员仍在理论课程的学习中，没有相关方向的实践经验，故任务通过按照兴趣方向进行自主分配。分配结果如下：**

语音识别部分由 \*\* 负责完成；

文字处理与渲染、模块整合由 \*\*\* 负责完成；

预览显示部分由 \*\* 负责完成；

目标打印部分由 \*\*\* 负责完成。

项目的进度追踪、模块整合、资源管理，以及综合性的技术设计等由 易晟 完成。

## ****时间规划****

2019.7.1-2019.7.14 项目相关的资料查找与基础技术学习，包括Python编程语言的基本掌握与Xilinx PYNQ Z2开发板基本使用方法；

2019.7.15-2019.8.15 基于已经学习的知识进行开发，各个模块完成自己的功能，并进行相关的测试；

2019.8.15-2019.8.31 各个模块进行对接，完成完整流程上的测试，解决模块间可能出现的不匹配问题；完成硬件上的组装；总结开发过程及其中遇到的问题，完成项目报告。

## 五 开发过程记录

**语音识别部分**

项目初期时，我们在网络上寻找开源的语音识别项目。经过长时间的查找后，我们在主流开源网站上收集到了以下项目：GitHub平台上的masr [2], ASRT\_SpeechRecognition [3], speech\_recognition [4], DeepSpeech [5]和PyPI平台上的SpeechRecognition [6]。

在其他平台的官方网站上，我们同样找到了科大讯飞、百度等语音识别服务，但是这些服务需要连接网络。由于我们的目标是制作离线设备，无法连接网络以使用这些平台，故只能放弃这些方案。科大讯飞有提供离线识别的语音工具，但是只能在安卓平台上使用。在接下来的时间里，我们将逐个对收集到的项目进行评测，以选择出性能最佳的方案。

下表就两个准确率较高和可以使用的模型进行简单介绍：

|  |  |  |
| --- | --- | --- |
|  | ASRT\_SpeechRecognition | MASR |
| 是否提供预训练模型 | 是 | 是 |
| 预训练模型大小 | 20MB | 120MB |
| 识别效果 | 拼音准确率80% | 汉字准确率92% |
| 基于框架 | keras | pytorch |
| 是否端到端 | 是 | 是 |
| 支持自行录音识别 | 是 | 是 |

MASR中文语音识别是一个基于端到端的深度神经网络的中文普通话识别工具，可以直接使用该项目进行语音识别，识别过程不需要gpu。MASR提供的语音模型的识别效果是个人开源项目中最好的。将MASR与Github 上面Star数较高的python中文语音识别项目进行对比。以下对比凡是提到识别效果，指的都是测试集识别效果，实际使用效果都会比测试集效果差。

ASRT\_SpeechRecognition的预训练模型很小下载方便，是语音转拼音再转汉字。但是开启API服务器之后，需要使用项目对应的客户端软件来进行语音识别较为复杂，硬件条件不足。

|  |  |  |
| --- | --- | --- |
|  | speech\_recognition | MASR |
| 是否提供预训练模型 | 是 | 是 |
| 预训练模型大小 | 110MB | 120MB |
| 识别效果 | 无 | 汉字准确率92% |

speech\_recognition使用thchs-20的测试集当作训练集，只有六个小时的录音时长，且完全用于训练，容易出现过拟合的情况，项目主页展示的是已经被训练过的的测试集上的识别效果，数据十分好看，但是毫无意义。

|  |  |  |
| --- | --- | --- |
|  | DeepSpeech | MASR |
| 基于框架 | paddlepaddle | pytorch |
| 是否提供预训练模型 | 是 | 是 |
| 模型结构 | CNN+RNN(gpu) | CNN |
| 预训练模型大小 | 750MB | 120MB |
| 预训练数据集大小 | 1200小时 | 150小时 |
| 外部模型 | 有 | 无 |
| 识别效果 | 87%（内部数据集） | 92%（AISHELL-1测试集） |

DeepSpeech是百度官方的项目。分别提供了中英文语音识别的预训练模型。其中中文语音识别模型提供了2个，一个基于AISHELL-1数据集训练（和MASR一样），一个基于百度内部的1200小时的中文数据集训练。百度给出了后者在内部测试集上的表现，大约是87%的准确率。这个项目使用RNN构建模型，要求使用GPU进行加速，且仅支持Python2.7。

SpeechRecognition

一些软件包（如 wit 和 apiai ）提供了一些超出基本语音识别的内置功能，如识别讲话者意图的自然语言处理功能。其他软件包，如谷歌云语音，则专注于语音向文本的转换。其中，SpeechRecognition 就因便于使用脱颖而出。识别语音需要输入音频，而在 SpeechRecognition 中检索音频输入是非常简单的，它无需构建访问麦克风和从头开始处理音频文件的脚本，只需几分钟即可自动完成检索并运行。SpeechRecognition 库可满足几种主流语音 API ，因此灵活性极高。其中 Google Web Speech API 支持硬编码到 SpeechRecognition 库中的默认 API 密钥，无需注册就可使用。SpeechRecognition 以其灵活性和易用性成为编写 Python 程序的最佳选择。本项目采用的是SpeechRecognition离线版本，用到的包为sphinx，sphinx是由美国卡内基梅隆大学开发的大词汇量、非特定人、连续语音识别系统，但是识别正确率不尽人意。

我们使用苹果耳机的麦克风作为录音设备，接入到开发板上，借助pynq的库函数录音生成.wav文件，用作STT的输入。在经过小规模的实际测试后，我们得出如下结果：

SpeechRecognition+Sphinx离线识别：

|  |  |  |
| --- | --- | --- |
| 原语句 | 识别结果 | 识别用时 |
| 很高兴认识你 | 南二 高 是 要 原 市人 因 无 | 29秒 |
| 床前明月光 | 中国 西安 里 民 联 国 | 38秒 |
| 我想找一个漂亮的女朋友 | 不过 小 着 一 颗 务 必要 让 那 名 日本 活跃 区 | 34秒 |
| 你来自哪里 | 施 以 来自 哪里 出 | 32秒 |
| 他今天没有来上学 | 他 今天 没有 来 稍 事 了 | 30秒 |
| 你需要帮忙吗 | 您 需要 帮忙 漫步 | 21秒 |
| 今天去哪里吃饭 | 今天 其余 拉丁 情人 法 | 23秒 |
| 祝贺南昌大学获得好成绩 | 祝贺 南昌 当事人 和 台湾 后 成绩 | 31秒 |
| 南昌的天气很热 | 南昌市 但 天气 很 热 | 23秒 |
| 我爱学习 | 我爱 献 爱心 | 24秒 |
| 你了解中国传统文化吗 | 以 了解 状况 传统 文化 忙 | 29秒 |

根据以上小规模的测试，语音识别结果的相似度（即发音上大致相似）为61.25%，准确度为46.25%。

在x86兼容计算机上进行测试的时候，带有自然语言模型的MASR模型可以有效提高识别的准确率，但是在移植到开发板上的时候，遇到了内存不足的问题，如图2所示。起初我们认为可以通过添加虚拟内存的方法缓解该问题，但在实践后发现并没有任何帮助。在资源管理器中观察程序行为后，发现程序将开发板的512MB内存占满后随即自动退出，并不占用任何虚拟内存。我们猜测该行为可能与相关Python库的底层实现有关，无法解决，因而决定放弃该方案。

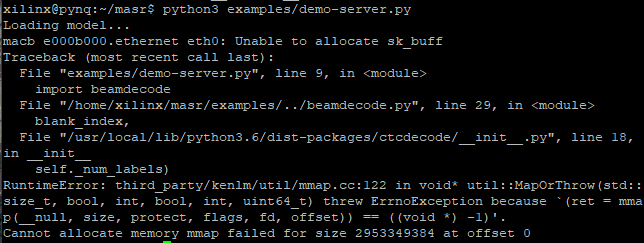


图2. 在使用带有自然语言模型的MASR样例在PYNQ-Z2开发板上测试的时候，出现了内存不足的问题，导致程序直接崩溃

我们在不使用语言模型的MASR项目上进行了测试，结果如下：

|  |  |  |
| --- | --- | --- |
| 原语句 | 识别结果 | 识别用时 |
| 他今天没有来上学 | 好··五五五五五五五五国五五五五 | 3分30秒 |
| 你需要帮忙吗 | 某宝宝五五无母母 | 3分42秒 |
| 很高兴认识你 | 五五毛五五五五五八某往宝宝 | 3分43秒 |
| 床前明月光 | 过五宝宝宝五五路 | 7分39秒 |
| 我想找一个漂亮的女朋友 | 某无王五国五五五五母五 | 4分08秒 |
| 你来自哪里 | 欧某宝宝宝宝五五无五无无八宝宝包 | 3分44秒 |
| 今天去哪里吃饭 | 华宝五五五万宝宝王五 | 5分02秒 |
| 祝贺南昌大学获得好成绩 | 往五五王五奥五络 | 3分54秒 |
| 南昌的天气很热 | 无五毛物五无五宝母五 | 4分09秒 |
| 我爱学习 | 鲁五五五五五八马宝 | 3分37秒 |
| 你了解中国传统文化吗 | 无五奥国某 | 3分51秒 |

结果表明配套的语言模型可能针对识别结果做了特殊优化，在PYNQ-Z2开发板的实际情景下有着远差于SpeechRecognition的效果。我们决定采用SpeechRecognition用于离线环境下的语音识别工具。

但是，结合本项目的最终目的考虑，过差的语音识别效果将会对下一步的表情插入工作带来很大的困难，我们在有网络连接的环境下引入了在线识别功能，可以在条件较佳的时候借助云服务提升我们的识别效果。在本项目中，我们使用了百度AI的在线语音识别，可以在开发板连接到网络后自动使用在线服务。在不涉及人名地名等语法难以推测的词语时，正确率在95%以上。

**文字处理与渲染部分**

1. Emoji 支持：在github找到键为中文或英文，值为unicode编码的emoji的字典，通过字典索引实现文字转emoji。

2. Emoji转位图：借助python的Image库，然后下载支持unicode编码的emoji字体Symbola.ttf实现emoji转位图。

**预览显示部分**

**项目初始阶段**

在项目初始阶段，主要学习了Python的基础知识，并在控制台和IDE环境下练习Python代码。了解了点阵屏幕显示文字的原理，并在GitHub上寻找到相关LED点阵屏幕显示的项目源代码，接着学习Linux系统常用命令，为项目开发阶段控制PYNQ-Z2开发板做了准备，然后经过讨论后在淘宝上选择并购买了第一种显示屏模块16032ZWA，如图2所示。该显示屏模块属于液晶屏，通过控制其中液晶材料的透光与否区分黑白两色，从而实现文字的显示。根据我们对该显示屏模块说明书的初步阅读，认为可以通过PYNQ开发板上的GPIO针脚进行信号传输，从而实现对显示输出的控制。



图3. 点阵显示屏模块16032ZWA

总的来说在项目初始阶段，主要工作就是学习基础技术和相关资料的查找，遇到的问题主要是在点阵屏幕原理方面，在学习基础技术方面都可以在网上查阅资料解决。

**项目开发阶段**

在项目开发阶段，首先是了解PYNQ-Z2开发板官网的各种输出接口，先使用了兼容树莓派的RPi GPIO针口。由于我们计划先使用提供的C语言样例程序先实现对显示屏的简单控制，接着仔细研究了16032ZWA液晶屏模块显示文字的源代码。由于该模块基于89S32单片机，reg52.h和intrins.h等头文件未包含在Linux中广泛使用的GCC编译器中，因此下载并学习了Keil软件，可以编译运行模块说明书中的源代码，但不能与PYNQ-Z2开发板直接相连。

在实现显示屏模块与开发板的连接时遇到了一定的问题。由于开发板上的GPIO针脚与杜邦线兼容，一开始我们直接将杜邦线焊接在显示屏上，但在使用时发现由于杜邦线内部的导线韧性不足，在模块的频繁移动下会出现断裂现象，而后改用韧性更好的铜导线用于连接。在经过一定的资料查找后，我们发现正确的焊接方法为使用特定的排针，然后再使用对应的杜邦线连接模块和开发板。

而后我们在控制PYNQ Z2开发板控制GPIO针脚的问题上遇到了瓶颈：官方的介绍书上对于RPi接口的描述使用寥寥几笔带过，提供的Python模块里也没有对基于RPi接口的GPIO使用做出详细的介绍。我们没有办法定位到确切的GPIO针脚。然后我们在Xilinx官网论坛上向开发人员寻求帮助，被告知PYNQ Z2开发板现有的Python库中没有可以直接用于控制RPi接口的支持，需要自行使用配套的Vivado软件进行RPi接口控制器的设计，然后再在开发板上加载控制器以实现目标需求。我们决定先学习控制Arduino针脚的Arduino\_IO模块。与此同时，根据自带教程中的显示输出样例，找到一款AdaFruit 1.8寸LCD显示屏模块可以接入Arduino接口工作，重新购买了同样适合Arduino的ST7735S TFT显示屏模块进行实验。在实验过程中发现模块的八个引脚和PYNQ-Z2板上接口并不是连续对应的，经过多次连接尝试，显示屏模块只会亮却不能实现其他控制操作。接着使用最先购买的sx显示屏模块实验尝试了8位并口输入方式和串口输入方式，分别按照并口和串口的方法输入，仍然没有反应。

然后尝试了另一种方案，在Xilinx官网上下载了Vivado软件，初步学习了工程建立的方法，将一个加法程序的C++代码通过Vivado HLS软件导出RTL后生成IP核，然后在Vivado软件添加PYNQ-Z2板后添加IP核，最后在jupyter软件中用python成功测试加法程序。由于项目时间有限，开发RPi控制器驱动难度较大，在老师的建议下，考虑用USB接口连接的显示屏实验。

在搜寻USB显示屏模块过程中发现了另一种HDMI接口控制方式，后来研究了HDMI的控制方法并成功显示了单行文本、多行文本以及颜文字。在此过程中也遇到了不少问题，主要是屏幕的分辨率问题和文字转图片的换行问题，通过Image模块中的revise函数将图片进行变化成最接近显示屏的800\*600进行显示，如图4所示：

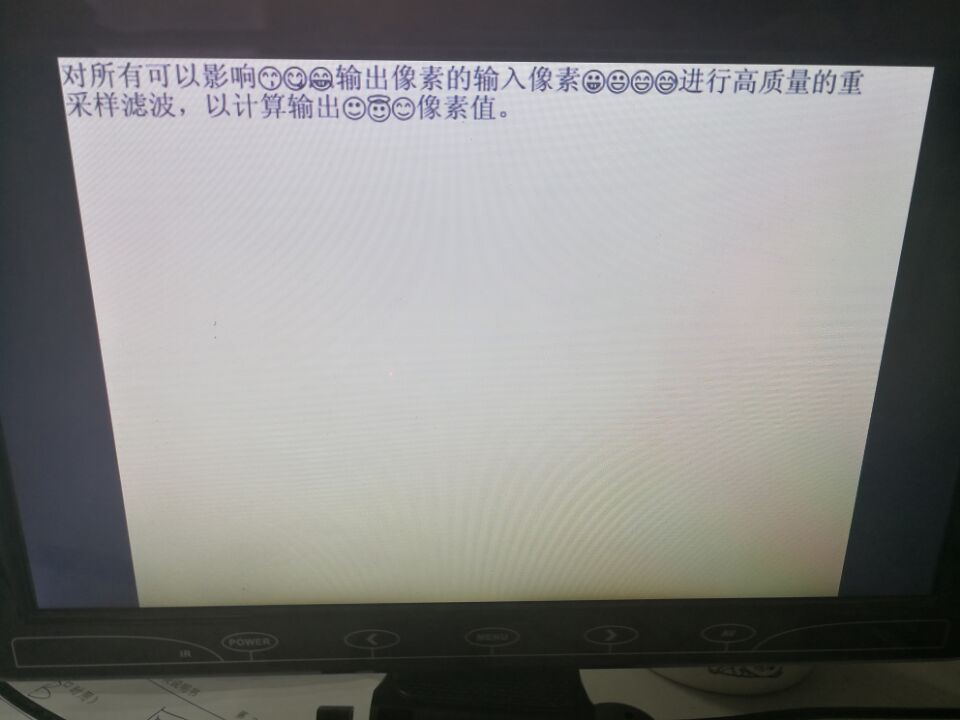


图4. 使用HDMI显示屏进行预览

由于之前对于GPIO的认识不够清楚，网络上也没有寻找到清晰易懂的解释，我们认为GPIO针脚如同其名字一样，可以进行广泛而通用的信号输入输出控制。在项目接近尾声的时候，无意中了解到了单片机这一概念，根据其文档的叙述，对GPIO这一接口概念有了不同的认识，才发现自己之前在这一方面浪费了不少时间。GPIO针脚提供的是二进制电平的发送与读取，而不是像串口一样，通过有规律的电平周期在两端设备之间建立通信通道，进而进行高层次（如系统）的信息传输。也就是说，理论上GPIO不是串口通信的替代品，而是通过GPIO的电平控制功能，我们可以模拟实现串口通信。但是由于Linux系统本身不属于实时系统，也就是说为了提升系统性能，Linux不会对硬件信号做出实时的相应，导致GPIO针脚可能无法按照预定的时钟周期发送信号，与目标设备的时钟周期错乱可能导致不正确或者无效的信息发送，因而我们不能使用这种方法进行硬件的控制。

**项目对接阶段**

在项目对接阶段，形成HDMI显示屏接口的过程中，经过多次对比实验，重新修改了显示屏显示图片的分辨率为640\*480，用reset方式解决了显示屏图片随着代码多次执行自动右移的问题，将中文字库和颜文字字库组成一起解决中文字符和颜文字同时出现在一句话中的显示问题，放弃了Image模块中的revise函数将图片进行变化的方式，采用了背景-图片的方式，将文字产生的图片放在640\*480的背景中显示，避免图片经过裁剪可能出现的清晰度问题。另外，也发现了使用的HDMI显示屏屏幕的接受图片时间是有限制的，因此，需要先生成几张用户提示信息的图片，既解决了显示屏的待机问题，又提高了显示屏的人机交互功能。在与打印机模块拼接的过程中，需要达到的要求是屏幕上显示的图像为打印机预览状态，通过对文字长度进行了分类，分别测出对应在显示屏上的显示偏移量和打印纸上的偏移量，形成一个字典解决了这个问题。

**目标打印部分**

1. 了解目前市面上常见微型打印机

不同于普通的大中型喷墨式/激光打印机，微型打印机一般都是热敏式的。热敏打印机的工作原理是打印头上安装有半导体加热元件，打印头加热并接触热敏打印纸后就可以打印出需要的图案。所以热敏打印机需要使用热敏纸（标签打印机需要使用标签纸），我们日常生活中常见的票据就是热敏纸，超市收银台用的正是微型热敏打印机。

1. 系统整体框架

微型热敏打印机由主控芯片、步进电机驱动模块、热敏打印头过热保护模块、热敏打印头缺纸检测模块、RS-232通信模块、供电模块等部分组成。其中步进电机驱动模块负责控制打印纸走纸及走纸速度；热敏打印头过热保护模块防止热敏打印头温度过高损坏；热敏打印头缺纸检测电路完成热敏打印头是否有纸检测；RS-232通信模块实现打印机与上位机之间的通信；供电模块给控制电路及热敏打印头供电。

1. 学习python基础语法和相关模块

开发板PYNQ-Z2支持python开发，需掌握一定的python语法基础，除此之外，也学习了开发过程中需要用到Image，serial，numpy，textwrap等模块。

1. 学习微型热敏打印机指令并测试

学习打印机厂家提供的《DP-EH300/L规格书》，并对规格书的相关小票指令和标签指令进行测试，着重对位图指令进行学习和测试。标签指令包含有两个层次：标签操作指令和标签内容定义指令，

1. 利用打印机指令打印

本款打印机没有自动换行功能，当打印内容超出标签纸宽度和高度时，超出部分的内容将被自动忽略，当打印内容超过一行所能打印的长度时，需要手动换行。由于我们的打印目标语言为中文，每个字块的宽度相同，因而只需要在间隔一定的字符处插入换行符即可。需要打印的字符可以直接通过发送GBK编码下的二进制内容，由打印机内置的字库进行对应的字符转换并打印出来。

我们根据手动测量，根据识别出的字符的数目（设定识别出的字符数目最大不超过48），判断合适的打印行数和字体大小。再把文字和emoji表情生成图片时设置对应的行数，当超过下表中对应的“单行最大字符数”时，生成图片时就要换行。

1. 支持emoji表情和文字同时打印

根据说明书介绍，打印机所使用的字库为GB2312，是由中国国家标准总局在1980年制定的一套汉字信息交换与处理的编码标准。显然，在当时制定的字符集没有也无法为现在所流行的emoji这种符号文字进行编码规定，因而我们无法使用打印机内置的字库进行打印。为此，我们设想了两种解决方案：

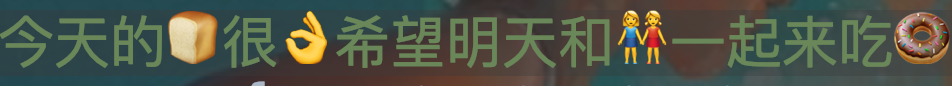
1. 我们所使用的打印机支持特定的标签指令集，其中包含有图片指令和文字指令，两者的工作方式都是先声明一块打印区域，再在区域内进行各自的绘图指令。因而，我们可以先准备emoji表情所对应的图片文件，然后将emoji作为图片，与文字一同放在标签纸上打印。为了同时实现对emoji的普遍性支持，加上我们希望能够在黑白打印机上也能够有较好的打印效果，我们在网络上找到了支持黑白显示emoji的字体Symbola，通过字体渲染的方式将字体中的字模转换为将emoji表情转化为图片，然后执行标签打印机的位图指令将内容打印出来。
2. 在上一方案的基础上，由于打印机可以支持图片打印，我们更进一步地计划将包括文字和emoji在内的所有内容渲染为一个图像，然后将该图像作为在该标签页上的唯一打印内容，直接进行打印。我们通过直接打印分辨率较高的图片，发现在打印区域内的部分可以直接打印，这更进一步支持了我们使用图片方式完成一次完整打印的可能性。



图5. 我们成功地使用打印机的绘图指令打印出了一张“高分辨率”（500\*665，部分内容已超出打印范围）的图像。

在经过实际的尝试后，我们选择了上述的第二种方案，将所有的内容渲染为一个图像，并且成功地达到了预期中的效果。

如字符串“今天的面包很OK，希望明天和朋友一起来吃甜甜圈”



转换成图像，并按照标签大小进行换行排版后的渲染图像：



还提供了添加下划线的功能：



1. 打印样式调整

在将生成的图像发送给打印机打印时，我们遇到了以下问题：

1）如果不进行特殊设定的话，将默认从同一位置开始打印，即页面的左上角。由于输入的文字长度可变，为了达成较好的显示效果，即让文字居中显示，我们针对特定规格的标签纸进行了打印位置的精确设定，经过测试，得出以下结论：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字符数 | 字体大小 | 行数 | 单行最大字符数 | page左上角坐标（x,y） |
| 1 | 5 | 1 | 2 | (120,70) |
| 2 | 5 | 1 | 2 | (40,70) |
| 3 | 4 | 1 | 3 | (13,90) |
| 4 | 3 | 1 | 4 | (12,105) |
| 5 | 2 | 1 | 6 | (40,120) |
| 6 | 2 | 1 | 6 | (13,120) |
| [7, 12] | 2 | 2 | 6 | (15,93) |
| [13, 18] | 2 | 3 | 6 | (15,66) |
| [19, 24] | 2 | 4 | 6 | (17,38) |
| [25, 36] | 1 | 3 | 12 | (15,95) |
| [37, 48] | 1 | 4 | 12 | (15,90) |

将该规则应用于我们的Python程序之中，得出了我们预料之中的结果。

2）我们一开始生成图片的方式为使用相同的字体和字号进行渲染，然后根据字数重新调整图片的大小。这导致在字数较少时，打印出来的图像分辨率十分低像素块聚集非常明显，打印效果不佳。之后，我们不将图片放大打印，而在生成图片时将图片内的文字按照相应的大小生成，同时图片的宽度和高度也要做出对应的调整。打印图片时就不需要通过打印放大的图片来实现不同字体大小，从而解决了该问题，打印效果非常清晰。

**整体逻辑**

对于与用户交互部分的操作流程，我们使用PYNQ-Z2自带的Python库，通过板上带有的4个按钮和4个LED灯与用户进行交互，并表达相应的信息。整体逻辑表达如图5。

我们让程序以一种服务的方式运行，在对应的状态下在显示屏上展示出对应的提示信息。该程序可以指导用户完成录音、预览、打印的基本流程，同时提供样式修改、步骤撤销等额外功能，以提升产品的用户友好程度。

为了使产品具有开箱即用的特性，我们需要让程序开机启动。我们首先采用的方法是使用系统提供的、用于执行用户指定的开机指令的rc.local脚本。但是，在设置程序为开机启动的时候，我们遇到了问题：使用rc.local运行程序时，由于我们所写的程序只能在前台运行，这将阻塞系统的正常启动流程。而rc.local在系统启动过程中执行较早，许多相应的服务没有执行完成，我们的程序在开始执行后不到一分钟直接退出。我们通过尝试了定时执行脚本、自动登陆执行等多种方法，都没有奏效。最终，我们决定以服务的方式，使用systemd提供的开机启动功能后台执行我们的主程序。事实证明，我们的方法是具有可行性的。再按照我们预期的规范输入与操作的情况下，程序可以正常运行，当遇到意料之外的情况的话，程序会意外中止，在该情况下无法继续工作。这种问题可以通过重启设备解决，再重新通电开机后，程序会重新自动执行，用户可以继续正常使用。



图6. 使用Python脚本构建的程序的基本逻辑

## 六 开发结果展示

**打印效果展示**

经过我们反复测试、修改出现的问题之后，我们选择了合适的标签大小参数、字体大小和位置参数，获得了一个相对较为稳定的结果，在联网情况下能够获得较高准确度的识别结果，emoji替换可以有效进行，而离线识别情况下由于本地识别准确度不高，可能产生意料之外的结果。以下两张图片分别展示了在不同长度下打印相似内容的效果和在不同环境下的打印结果。



图7. 经过特定的适配后，在不同字数下都有较为美观的排版

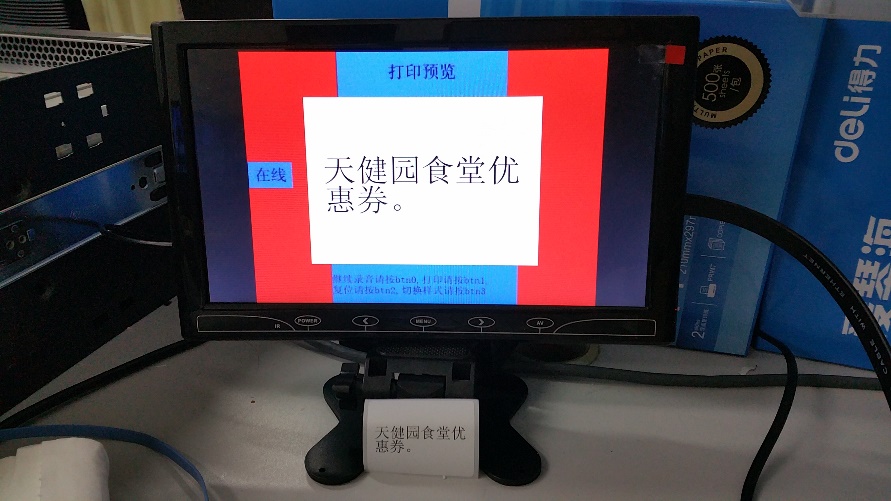
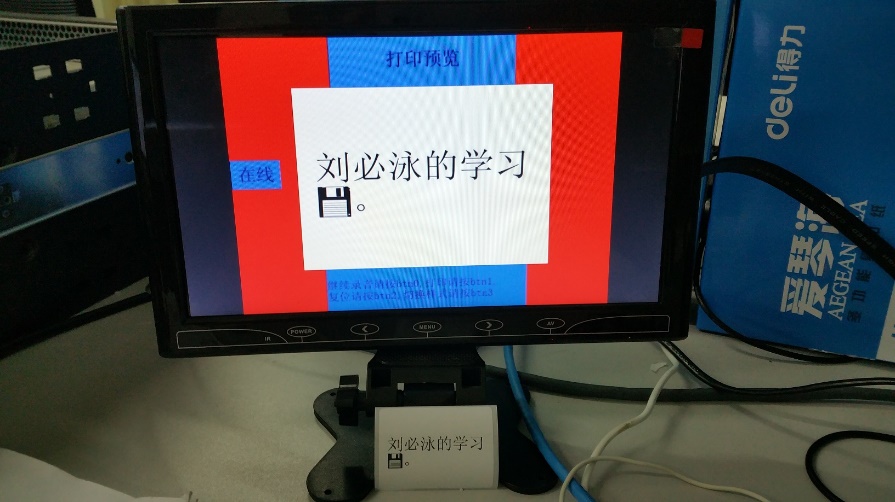
 

图9. 在不同语境下，在线识别的结果。百度的在线实别API能够对常用的抽象语句产生较好的识别结果；但是对于具体有具体意义的特殊名词，往往因为不了解语境而出现识别失误

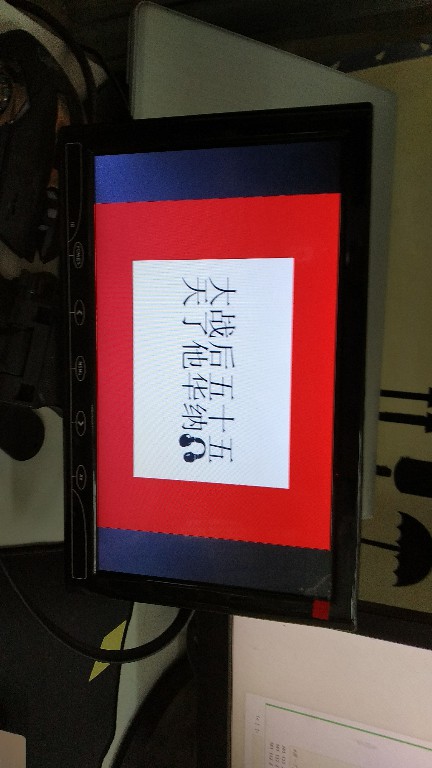
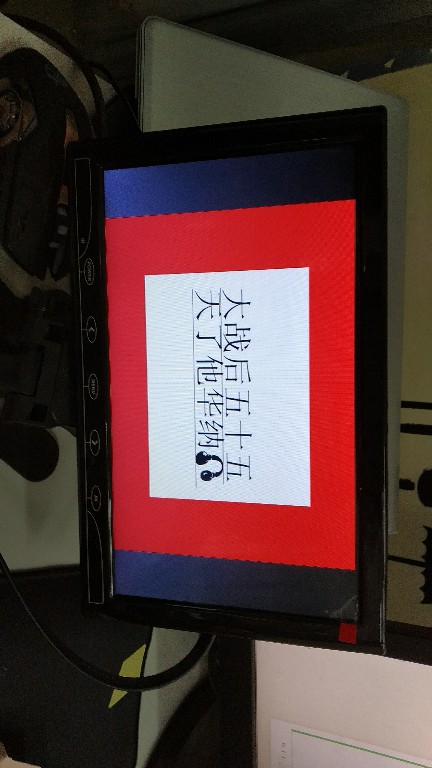
 

图10. 离线识别情况下，输入语音内容为“大家好，我系古天乐，贪玩蓝月”情况下的识别结果；左右侧分别为没有和有添加下划线情况下的打印预览

**外壳设计展示**

对于外壳的设计，我们计划使用亚克力板作为主要部分，加上螺丝等辅助部件进行固定。由于PYNQ Z2开发板的生产厂家没有提供现有的外壳产品或是有效的外壳设计图纸（图11-13），我们通过网购途径购买了一些亚克力板和螺丝的替代品，但是没有寻找到合适的产品用于替代螺帽，因而使用螺丝固定的硬纸壳用作中间部分的支撑。我们计划将PYNQ Z2开发板固定在亚克力板的边缘，由于我们使用的热敏打印机高度有约5cm，这个高度将足够用户伸手去按开发板上的操作按钮以与程序交互。

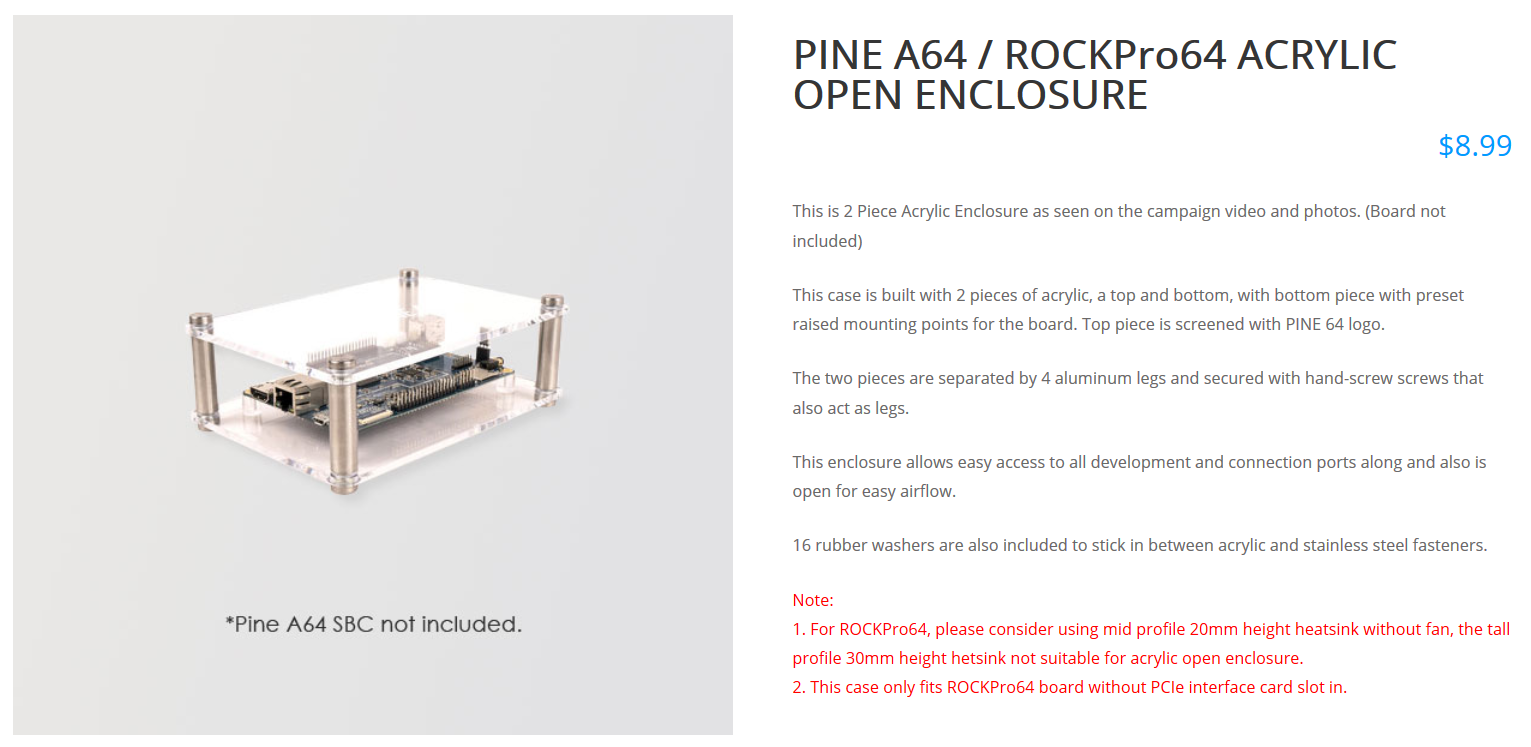


图11. 其他厂商在售卖开发板的同时，也售卖对应的外壳成品

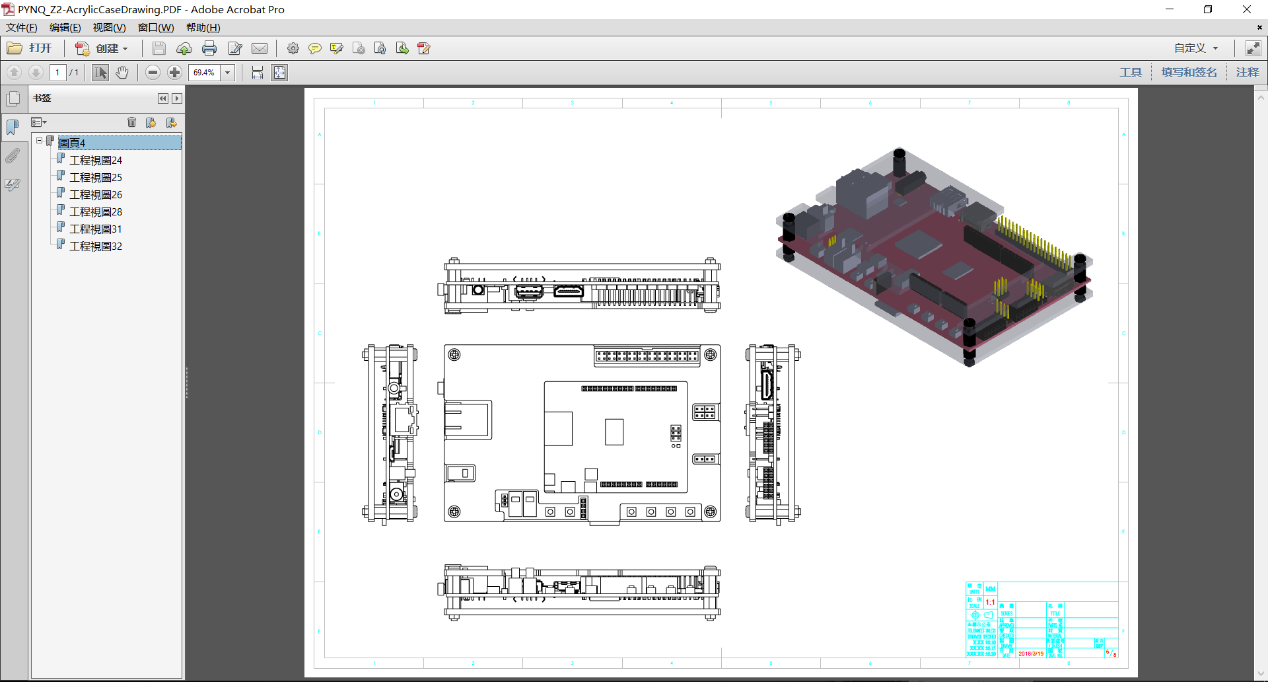


图12. PYNQ Z2官方网站所提供的外壳的全部信息

图13. PYNQ-Z2的生产厂商只提供了设计图纸；在向淘宝提供定制服务的店家提供了我们获得的所有相关信息后，仍然被告知信息不足，无法制作

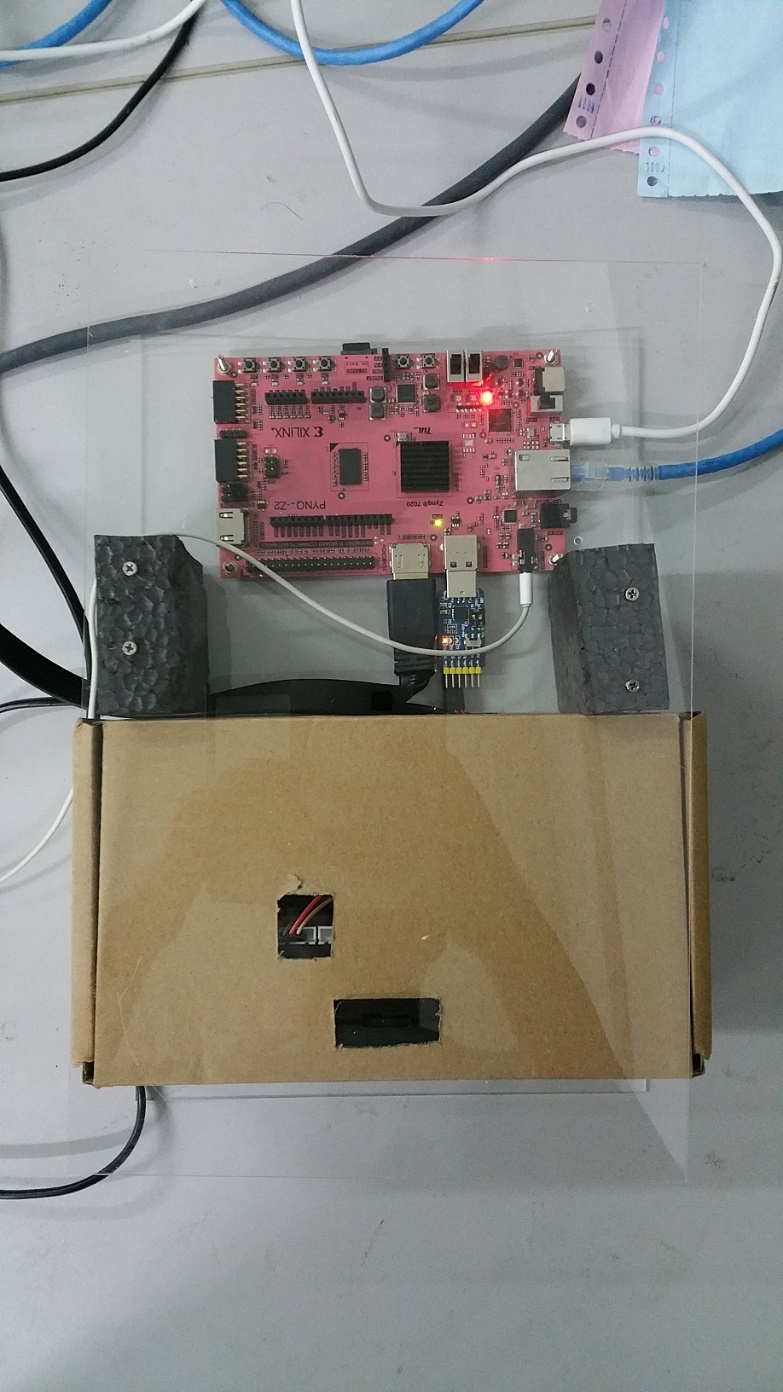


图14. 我们手工制作外壳的最终效果

**完整的打印过程展示**



图15. 开机时的LED闪烁，之后程序自动启动；

在此之前，屏幕会没有任何显示

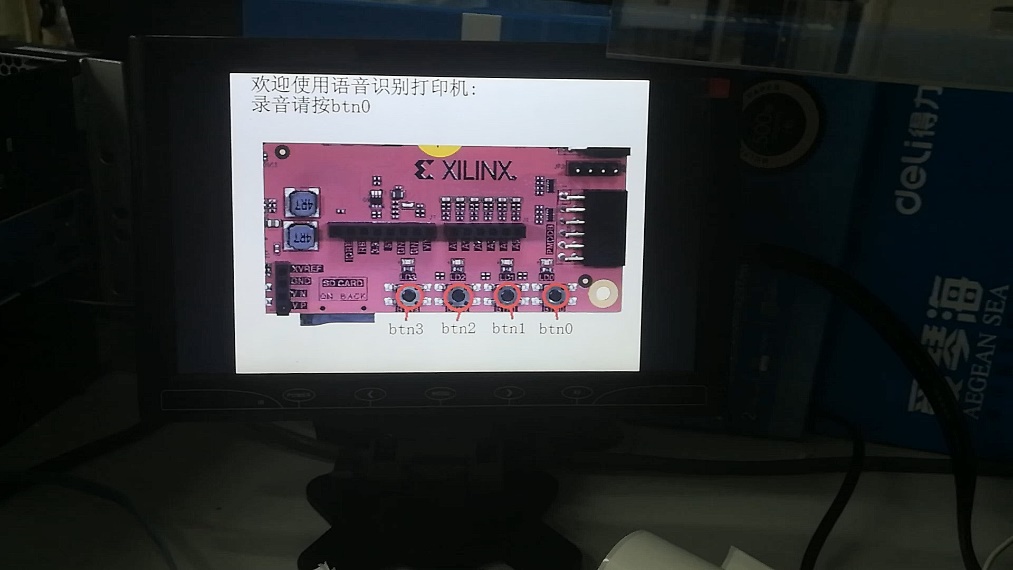


图16. 初始界面，包含简要的使用方法说明



图17. 根据提示按下btn1，录音提示灯亮，进行定时为5秒的录音

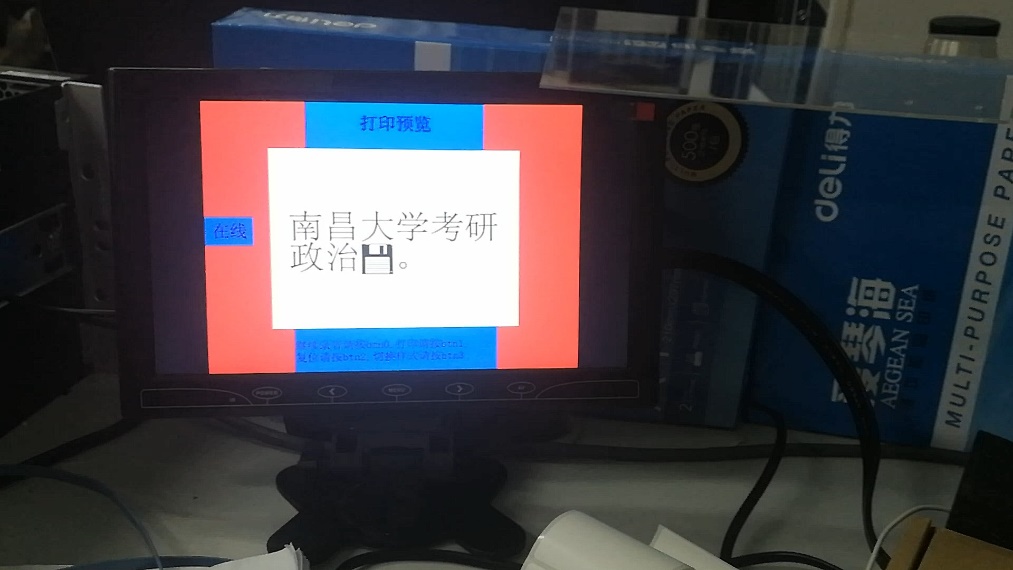


图18. 识别结果展示：左侧显示最后一次识别使用的方式，

下侧为下一步的可用操作提示

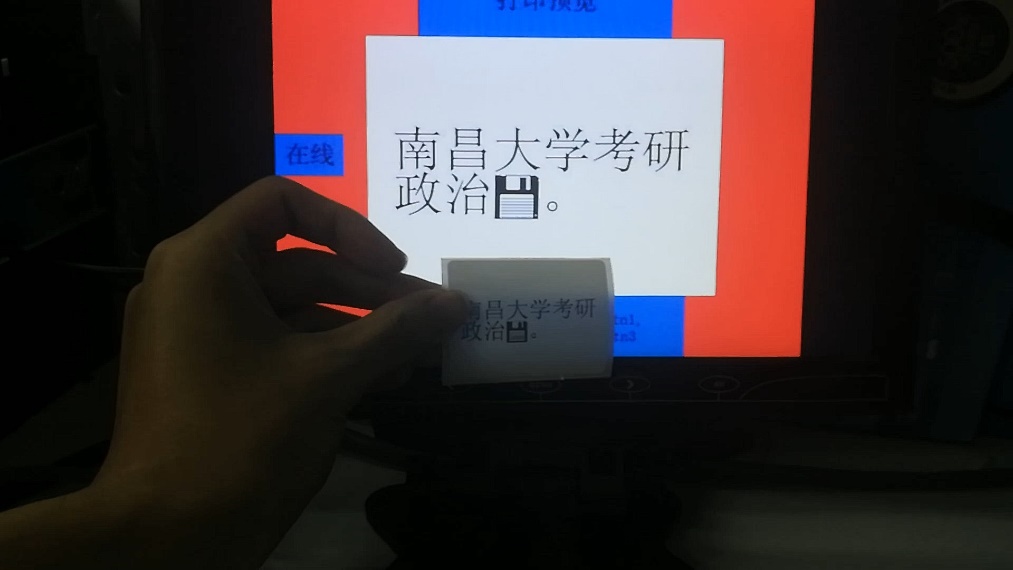


图19. 按下btn2进行打印，打印实物与预览对比

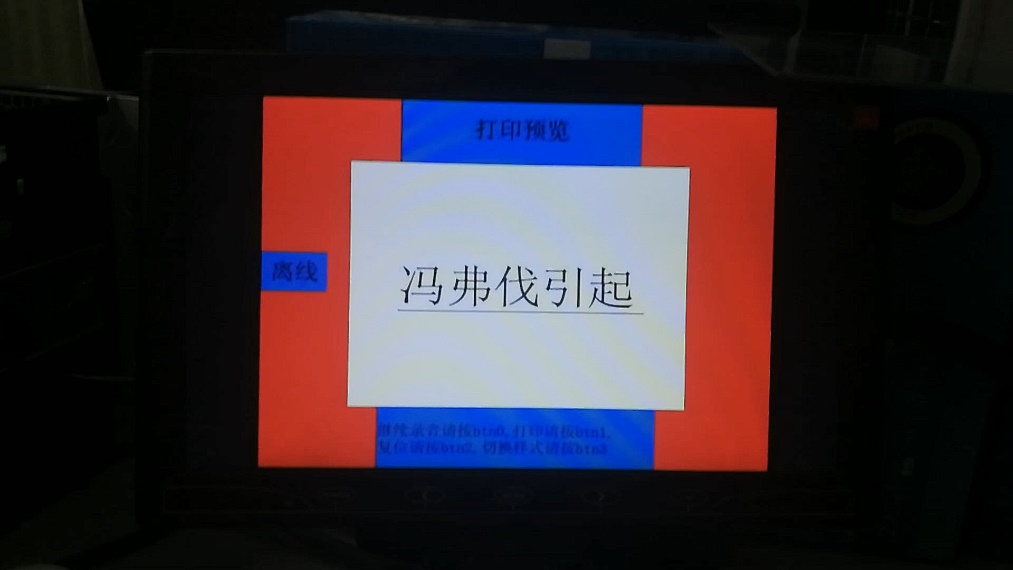


图20. 离线识别：作为没有网络连接时的一种替代方案，识别效率较低。

本图的输入内容为“公共打印机”，添加了可选的下划线样式

## 七 项目评估

经过两个月的小学期，项目组的各位成员都奋斗于其中，从全新知识的学习，到目标的设计与实现，获益匪浅。根据项目组中成员的交流来看，这也是所有人第一次参与一个从硬件到软件的完整项目，从系统构思，到软硬件相关信息的了解，再到根据需求在大量的候选中做出选择；而后，又需要再根据所选择的具体产品，进行文档的阅读，将初步的构思用对应的操作进行实现。在整个过程中，我们也遇到了不少意料之外的情况，但是都经过我们的尝试、在网络上的搜索，以及询问相关产品的技术人员，最终还是解决了问题，基本完成了任务。

虽然在制作的过程中我们收获颇丰，但也从中发现了自己的一些不足之处。由于在学校学习的大多数课程都是理论课程，我们很难在日常学习中将其与实践结合起来，因此在实际开发的过程中遇到了很多基础性的问题。在硬件开发过程中，应该在一开始就先研究PYNQ-Z2的各种接口的控制方式，将其研究透彻后再购买相关的模块进行实验，这样能缩小在遇到问题时查找帮助的关键词范围，能减少开发过程中可能遇到的未知因素，减小出现购买的模块无法控制的可能性。我们对于网络博客的认识也存在一定问题，网络博客并非学习新知识的最佳去处，而应该是一个补充知识的平台。根据我们这段时间来看，大部分博客都其实并没有什么用处，甚至有的还存在错误的操作，与作者联系往往还无法得到回答，时间浪费十分严重。相反，在论坛上与他人交流能有较为有效地进行反馈和互动，可以获得更多有实际意义的解决方案，有了问题提出后也能及时解决。在软件开发中，没有注意到开发环境与生产环境的统一，打印机部分在经过了几天才发现输出结果不对是使用了Python 2而不是目标环境的Python 3的原因，使用在Python 3上编写无误的代码在个人电脑上出现了异常结果，使得打印机部分的进度一度停滞。

在团队合作方面，我们存在着一定的不足，由于一开始对于项目的重视程度不高，对于基础知识的学习都花费了很多时间，以至于当七月完全过去之后，项目组的各位成员才完成了基础知识的学习，开始进入状态。但是在意识到了问题之后，我们加强了联系，经过协商决定放下各自手头上的其他任务，优先处理这个项目，最终才赶在结束之前将任务基本完成。这给我们的经验教训是，作为项目负责人，应该多与成员交流，积极组织，调动大家的积极性；作为项目成员，应该尽可能高效地完成分配到的任务，以大局为重，不因为自己的安排而耽误整体项目的进度。

就项目成果来看，我们从功能上基本达成了项目开始时所提出的需求：语音输入转化为文字，加入emoji的替换，预览并调整样式，最后根据用户的选择进行打印，有外壳包装，满足了作为产品原型的基本要求。在用户界面上，我们只实现了最基本的用户交互，没有对界面就美学上进行大量的设计，因而在面向用户时的第一观感可能不是很好；在外壳上，由于我们对于工业设计不了解，设计时仅仅考虑了方便制作和想到了的必要功能；由于使用的打印机是通用功能型的，在大小和成本上都显然偏高，等等。因而，当我们接下来若是以制作一个产品为目标的时候，还有不少能够提升的地方。

## 八 参考资料与文献

1. PYNQ Introduction — Python productivity for Zynq (Pynq) v1.0, <https://pynq.readthedocs.io/en/v2.3/>
2. libai3/masr: 中文语音识别，高识别率预训练模型，支持docker快速安装 Chinese Speech Recognition; Mandarin Automatic Speech Recognition;， <https://github.com/libai3/masr>
3. nl8590687/ASRT\_SpeechRecognition: A Deep-Learning-Based Chinese Speech Recognition System 基于深度学习的中文语音识别系统, <https://github.com/nl8590687/ASRT_SpeechRecognition>
4. xxbb1234021/speech\_recognition: 中文语音识别, <https://github.com/xxbb1234021/speech_recognition>
5. PaddlePaddle/DeepSpeech: A PaddlePaddle implementation of DeepSpeech2 architecture for ASR., <https://github.com/PaddlePaddle/DeepSpeech>
6. [SpeechRecognition · PyPI, https://pypi.org/project/SpeechRecognition](SpeechRecognition%20·%20PyPI,%20https://pypi.org/project/SpeechRecognition)
7. 586628502DP-EH300L规格书（标签）.docx
8. pyserial · PyPI, <https://pypi.org/project/pyserial/>
9. 定制PYNQ的overlay - bramblewalls的博客 - CSDN博客, <https://blog.csdn.net/bramblewalls/article/details/80045922>