线性 CCD 翻译设备

2012 级 8 班黄杰

2016 年 5 月 26 日

指导老师: 杨伟超
电子信息工程学院
电子信息工程专业

目录

摘要

在日常的生活学习中，我们经常会接触到不同的语言。一个优秀的翻译工具就显得尤为重要。传统的翻译工具有字典，电子词典等，而这些翻译工具效率低下。本文介绍一种基于 Linux 操作系统的线性 CCD 翻译装置，该装置利用 OCR 技术对图像进行识别，结合翻译程序将纸质文本上面的英文字符翻译。论文详细介绍了该装置的各部分硬件组成和软件实现，其中包括利用线性 CCD 传感器获取图片信息，利用图片信息生成图片，OCR 字符识别，字符处理，词语查询发音。该装置可拓展性强，可以将获得的字符翻译成各国语言，以及识别不同的语言，具有很强的灵活性。

关键字： 语言; 词典;Linux;CCD; 扫描;OCR; 翻译

Abstract

In our daily day life,we often meet different languages.So a good translation tool is becoming more and more vital to us.There are some tools like electronic dictionary, but they are ineffective.This article introduce a translation device base on Linux,which using OCR technique combining a translation program to translate the English text on paper to Chinese.The details about the components,including using linear CCD sensor to get the data information,constructing the image by using the original data, optical character recognition,handling the text,and pronunciation,will be described in this article.This translation tool have good practical use in almost many languages,not just English.

Keywords: Language, Dictionary, Linux, CCD, Scan ,OCR,Translation

1 引言

1.1 课题背景及研究意义

随着电子科技，计算机科技，网络技术飞速发展，人们已经进入信息大爆炸的时代。在日常生活中我们或许会接触到各种语言，我们需要这样一种翻译设备，它快捷，高效。传统的翻译设备有纸质的词典，电子词典等，但这些工具要么笨拙，要么需要手动输入单词。因此在这种背景下一种扫词查词的工具应育而生。本文讨论的翻译设备基于 Linux 操作系统，词库可以根据需要调整，可拓展性强。基于本设备的设计，可以根据用户的需要将本设备修改为识别任意的语言，以及对任意的语言进行翻译，具有很强的可定制性，为外语的学习提供了极大的便利。

1.2 课题的设计要求

本课题的主要工作是利用线性 CCD 采集数据，生成图片，OCR 识别文字，翻译单词。课题的要求如下:

1. 正确生成采集的图片
2. 正确识别图片文字
3. 能够对识别文字进行翻译
4. 能够对识别单词发音

1.3 课题的研究内容

本文讨论围绕的主题是如何将打印在白纸上的英文单词翻译成中文。具体来说，论文会详细的介绍设备的硬件组成，讨论如何利用线性 CCD 传感器收集白纸上的文字信息，如何利用这些文字信息生成一张清晰的图片，如何利用 OCR 识别软件将图片上的文字，如何利用 Linux 命令行工具处理文本，以及如何将单词发音。

2 综述

2.1 功能描述

本设备实现对文本的扫描，录入，识别，翻译，发音。该设备主要由微电脑树莓派和外围数据采集电路构成，其中微电脑的型号选择的是 ARM 结构的 Linux 微控制器 Raspberry Pi B+。树莓派连接线性 CCD 传感器，将采集到的模拟量传给 ADC 芯片, 之后树莓派接受 ADC 的数字输出，保存数据。接着将接受到的数据合成图片，降噪，生成一张供识别的图片。之后利用 OCR 软件识别图片，在利用字符处理程序对得到的字符进行后续的处理生成待翻译的文本文件。利用翻译程序将生成的文本进行翻译，以及对识别的单词进行发音。[1]

2.2 主要模块

第一部分，CCD 获取图片信息。[2] 这部分利用线性 CCD 获取纸质材料上的文本的数据信息。本设备采用的是线性 CCD 传感器 TSL1401R-LR，该传感器能够将光强转换为电压值。电压值经过模数转换变为数字量，以供运算处理。本文将详细介绍如何驱动线性 CCD 传感器，如何将电压值转换成数字量，如何将数字量的光强进行二值化处理，以及如何储存数据。

第二部分，利用图片信息生成图片。这部分的目的是利用线性 CCD 获取的数据，生成一张清晰的图片，供 OCR 软件识别。这部分我们主要利用的工具是 Python 语言，以及一个 Python 的图像处理库。本文会以一个实际的例子演示如何利用 CCD 获取的原始数据生成一张图片，以及如何对图片进行降噪处理，最终得到一张清晰的图片。

第三部分，OCR 字符识别。这一部分对生成图片上的字符进行识别，得到一个包含字符的文本文件。这里是使用已经开发好的 OCR 开源引擎，本文仅仅讨论如何利用 OCR 软件来获取文本。

第四部分，词语查询与发音。这部分实现对所识别单词的翻译，发音。同样本文讨论的是如何利用 Linux 下的命令行工具，对文本数据流进行过滤。最终实现单词的翻译与发音。

2.3 技术实现

数据采集: 本设备采用的 CCD 传感器是 TSL1401R-LF, 这是一款 128*1 个像素的线性 CCD 传感器。TSL1401R-LF 一次采集 128 个像素点的电压值，之后串行输出到 ADC 的输入引脚将模拟量转化成数字量。树莓派获取 ADC 的转换结果存入缓存，根据生成的数据和光照强度对数据进行二值化处理。将处理之后的数据保存到文本文件 CCD_data.txt 中。至此，数据的采集工作就完成了。接下来需要将得到的数据进行进一步的处理。

图片生成: 利用 Python 语言，结合 Python 的图形处理库 Image 读取文本文件的数据，经过降噪处理生成一张 128*256 的图片文件 CCD_pic_v.jpg。该部分的目的就是最大限度的生成清晰的图片供下一个环节使用。

OCR 识别字符: 该部分识别 CCD_pic_6.jpg 上的字符。OCR 程序选择开源软件 Tesseract。生成包含单词的文本文件 CCD_data.txt

翻译与发音——由于 CCD_data.txt 中含有多余字符，因此该部分将 CCD_data.txt 的包含数据以数据流的形式过滤，得到只包含单词的数据流，接着将其输入给查词程序 sdcv 和 espeak。sdcv 是翻译的程序，可以根据需要安装多种词库。espeak 是一个文本转语音的程序，用来发出单词的读音。

3 系统硬件设计

3.1 微电脑 Raspberry Pi

树莓派（英语：Raspberry Pi），是一款基于 Linux 的单板机电脑。它由英国的树莓派基金会所开发，目的是以低价硬件及自由软件刺激在学校的基本电脑科学教育。[6]

Raspberry Pi B+ 采用 Broadcom BCM2835 Soc, ARM1176JZF-S 核心（ARM11 系列）700MHz (单核) CPU，512M 内存，4 个 USB 接口，3.5mm 音频输出，100Mbps 以太网接口（RJ45 接口），40 个 GPIO 接口。GPIO 功能如图:

GPIO#	2nd func	pin#		pin#	2nd func	GPIO#
N/A	+3V3	1		2	+5V	N/A
GPIO2	SDA1 (I2C)	3		4	+5V	N/A
GPIO3	SCL1 (I2C)	5		6	GND	N/A
GPIO4	GCLK	7		8	TXD0 (UART)	GPIO14
N/A	GND	9		10	RXD0 (UART)	GPIO15
GPIO17	GEN0	11		12	GEN1	GPIO18
GPIO27	GEN2	13		14	GND	N/A
GPIO22	GEN3	15		16	GEN4	GPIO23
N/A	+3V3	17		18	GEN5	GPIO24
GPIO10	MOSI (SPI)	19		20	GND	N/A
GPIO9	MISO (SPI)	21		22	GEN6	GPIO25
GPIO11	SCLK (SPI)	23		24	CE0_N (SPI)	GPIO8
N/A	GND	25		26	CE1_N (SPI)	GPIO7
EEPROM	ID_SD	27		28	ID_SC	EEPROM
GPIO5	N/A	29		30	GND	N/A
GPIO6	N/A	31		32	-	GPIO12
GPIO13	N/A	33		34	GND	N/A
GPIO19	N/A	35		36	N/A	GPIO16
GPIO26	N/A	37		38	Digital IN	GPIO20
N/A	GND	39		40	Digital OUT	GPIO21

图 1: Raspberry Pi B+ GPIO

3.2 CCD 数据采集模块设计

感光耦合元件（CCD），又称电耦合元件（英语：Charge-coupled Device）, 是一种集成电路，上有许多排列整齐的电容，能感应光线，并将影像转变成数字信号。经由外部电路的控制，每个小电容能将其所带的电荷传给它相

邻的电容。CCD 广泛应用在数码摄影，天文学，尤其是光学遥测技术，光学与频谱望远镜，和高速摄像技术。

3.2.1 线性 CCD 简介

TSL1401R-LR 线性 CCD 传感器包含 128*1 个像素点，内部自带放大电路，具有像素数据保持功能。128 个像素点同时开始，结束收集电荷。每个像素高 63.5μm 宽 55.5 μm, 每两个像素中心的间隔 8μm。内部的逻辑控制电路简化了操作，该器件只需要一个时钟信号和一个串行输入信号。[4] TSL1401R-LR 的引脚图如下图所示：

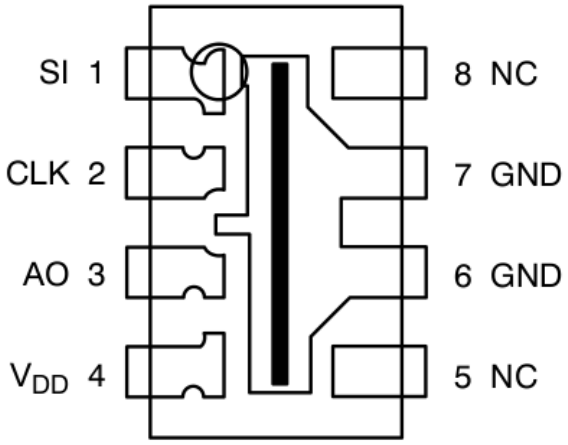

图 2: TSL1401R-LR 引脚

TSL1401R-LR 引脚介绍:

SI ： 串行输入，决定CCD何时输出数据
CLK: 时钟输入，控制像素的转移，像素输出，重置
AO ： 模拟输出
VDD: 参考电压
GND: 地
NC ： 无内部连接

3.2.2 CCD 与 Raspberry Pi B+ 的接口

由于 TSL1401R-LR 含有内部控制和放大作用，因此它和树莓派的连接很简洁。SI 连接树莓派的 GPIO22 引脚，CLK 连接树莓派的 GPIO23 引脚。树莓派的这两个引脚是用的通用输入输出的功能，而没有用第二功能。接口图如下图:

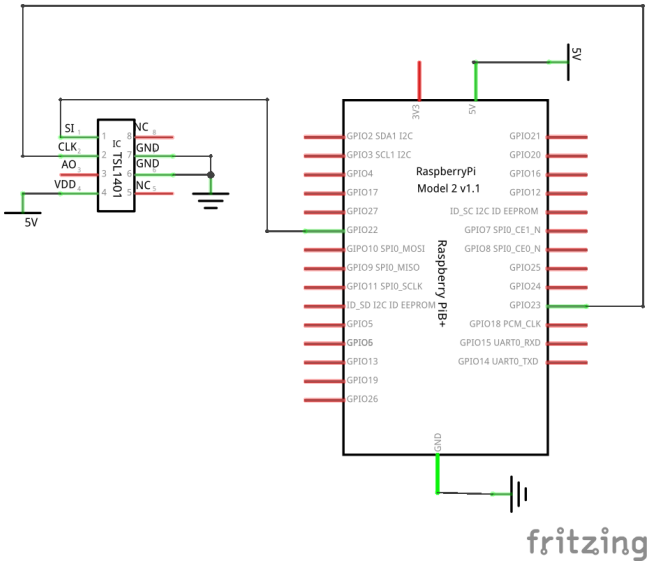

图 3: TSL1401R-LR 与 Raspberry Pi B+ 的连接

3.2.3 CCD 实物的安装

CCD 对光线很敏感，不能直接暴露在光线之下。该论文讨论的设备采用小孔成像的原理，用封闭的外壳将 CCD 隔绝大部分的光线，再在一端开启一个小孔，以便于只让 CCD 正下方的光线透过小孔投射到 CCD 的感光器件上。如下图所示，经过测试这种方法可以满足本设备的要求。[5]

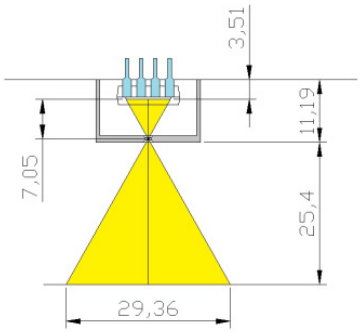

图 4: 利用小孔成像原理安装线性 CCD

3.3 ADC 模块设计

由 TSL1401R 的引脚 AO 可以知道线性 CCD 的输出是一个模拟的电压值，因此本模块的功能就是将线性 CCD 输出的像素点的电压值转换为数字量，以供下一步处理。

3.3.1 ADC0832 简介

ADC0832 是一款由美国国家半导体公司生产的 8 位分辨率 A/D 转换芯片，其最高分辨可达 256 级，可以适应一般的模拟量转换要求。其内部电源输入与参考电压的复用，使得芯片的模拟电压输入在 0 5V 之间。芯片转换时间仅为 32μS，据有双数据输出可作为数据校验，以减少数据误差，转换速度快且稳定性能强。独立的芯片使能输入，使多器件挂接和处理器控制变的更加方便。通过 DI 数据输入端，可以轻易的实现通道功能的选择。[7] ADC0832 的引脚图如下图所示:

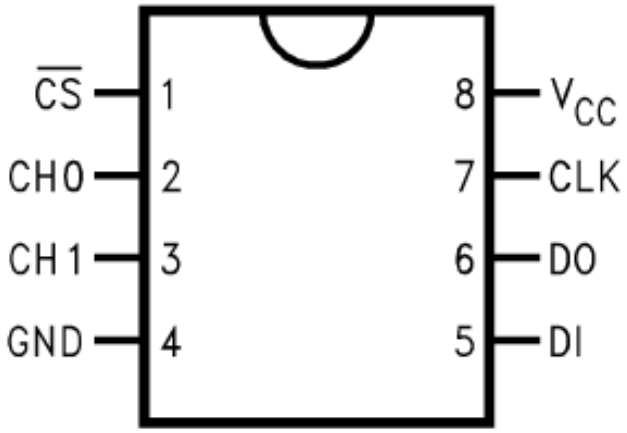

图 5: ADC0832 引脚

ADC0832 引脚介绍:

CS ： 片选信号（低电平有效）
CH0： 模拟量输入通道 0
CH1： 模拟量输入通道 1
GND： 地
DI ： 数据输入端
DO ： 数据输出端
CLK： 时钟信号
VCC： 电源

3.3.2 ADC 与 Raspberry P B+ 的接口

ADC0832 的数字输出是由 DO 串行输出，接入树莓派的 GPIO27; 片选信号 CS 连接到树莓派的 GPIO17, 如下图所示:

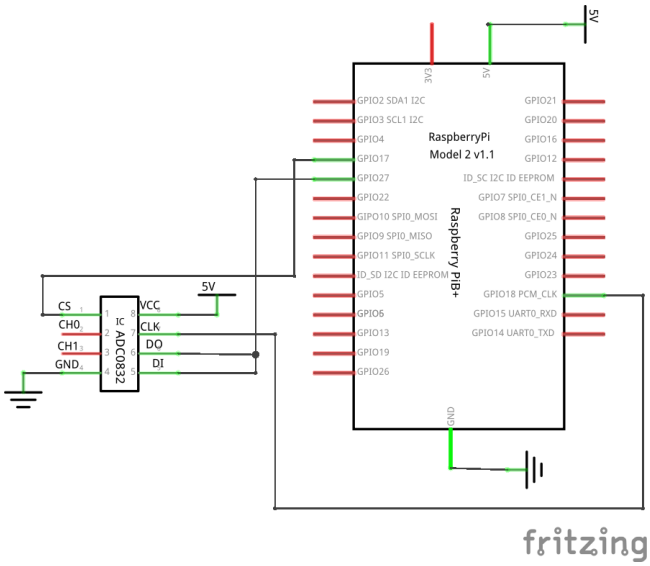

图 6: ADC0832 与 Raspberry Pi B+
的连接

3.4 CCD 数据采集模块和 ADC 模块与 Raspberry Pi B+ 的接口

该部分是是 CCD 数据采集模块和 ADC 模块的综合，将 TSL1401R-LR 采集到的模拟电压值，输入到 ADC0832 的通道 0（CH0），即将模拟的像素电压值转化为数字量。进一步通过 ADC0832 的数字量输出引脚 DO 将数字量串行的输入到树莓派存储，进而为图像处理做准备。接口图如下:

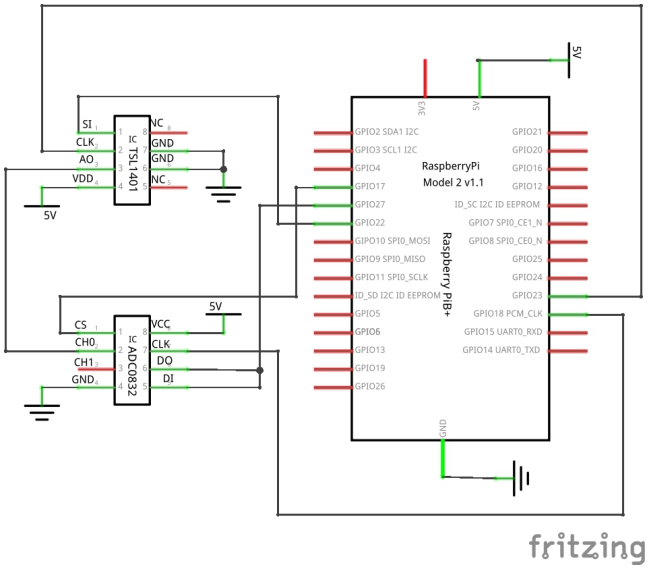

图 7: CCD 数据采集模块和 ADC 模块与 Raspberry Pi B+ 的接口

3.5 小结

通过这一部分的讨论，我们已经基本搭建好了翻译设备的硬件组成。在实际操作的时候需要适当增加光照强度，以获得质量更高的扫描图片，提高 OCR 的识别效率。接下来我们将继续讨论设备软件实现。具体的来说我们会依次讨论利用 CCD 获取图片信息，利用图片数据得到扫描图片，光学字符识别，单词翻译及发音。

4　CCD 获取图片信息

本论文讨论的翻译装置获取图片的原理和扫描仪的原理一样，一次扫描一条线，这一条线横向移动构成一张完整的图片。接下来简单介绍 CCD 的工作原理，以及如何编写程序操纵 TSL1401R-LF，收集 128 个像素点的电压值，将电压值进行 A/D 转换，根据光线强度对 A/D 转换的结果二值化处理，对二值化处理后的数据进行保存，进而获取用与显示的图片的数据。

4.1　CCD 工作原理

在一个用于感光的 CCD 中，有一个光敏区域（硅的外延层），和一个由移位寄存器制成的传感区域（狭义上的 CCD）。图像通过透镜投影在一列电容上（光敏区域），导致每一个电容都积累一定的电荷，而电荷的数量则正比于该处的入射光强。用于线扫瞄相机的一维电容阵列，每次可以扫瞄一单层的电容；而用于摄像机和一般相机的二维电容阵列，则可以扫瞄投射在焦平面上的图像。一旦电容阵列被曝光，一个控制回路将会使每个电容把自己的电荷传给相邻的下一个电容（传感区域）。而阵列中最后一个电容裡的电荷，则将被传给一个电荷放大器，并被转化为电压信号。通过重複这个过程，控制迴路可以把整个阵列中的电荷转化为一系列的电压信号。在数字电路中，这些信号将会被采样、数字化，并通常被储存起来；而在模拟电路中，它们将会被处理成一个连续的模拟信号（例如把电荷放大器的输出信号输给一个低通滤波器）。[3]

4.2　TSL1401R 操作时序

TSL1401R 具有严格的时序要求，SI 高脉冲的出现标识着一轮数据收集的开始。TSL1401R 每个像素内部光敏光电二极管经过内部的放大电路将光强转换成电压，通过采样电容对电压进行采样，再由逻辑控制电路将模拟电压值串行的输出到 AO, AO 输出的模拟量必须即时送入到 ADC0832 中，进而即时将数字化的光强储存在树莓派中。TSL1401R-LF 具体的操作时序如下:

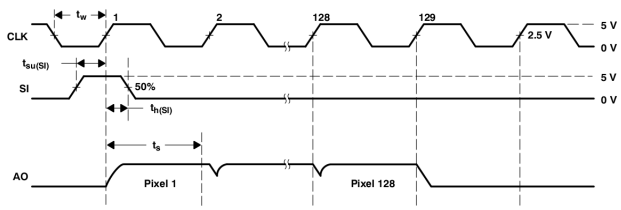

图 8: TSL1401R-LF 时序图

根据时序图我们可以写出采集 CCD 数据的函数，如下所示:

```
void caiji()      //Use 2D array to get 256*128
                  //pixels' information
{

  int k=0;
  int temp=0;
  digitalWrite(SI,HIGH);
  delayMicroseconds(1);
  digitalWrite(CLK,HIGH);
  delayMicroseconds(1);

  digitalWrite(SI,LOW);
  delayMicroseconds(1);

  delayMicroseconds(20);

  pinMode(ADC_DIO, OUTPUT);
  data[datacount++]=choose(get_ADC_Result());
  digitalWrite(CLK,LOW);
  for(k=0;k<127;k++)
  {
    delayMicroseconds(1);
    delayMicroseconds(1);
    digitalWrite(CLK,HIGH);
```

```
    delayMicroseconds(1);
    delayMicroseconds(1);
    pinMode(ADC_DIO, OUTPUT);
    data[datacount++]=choose(get_ADC_Result());
    digitalWrite(CLK,LOW);
  }
  delayMicroseconds(1);
  delayMicroseconds(1);
  digitalWrite(CLK,HIGH);
  delayMicroseconds(1);
  delayMicroseconds(1);
  digitalWrite(CLK,LOW);


}
```

该函数将采集到的 128 个像素数据经过 A/D 转换，二值化处理，暂存在数组 data 中。

4.3 ADC

由 TSL1401R-LF 的时序图可知，要操纵 CCD 获取图片信息，必须同时对每一个像素输出的模拟量进行 A/D 转换，因此我们还需要掌握如何操作 ADC0832。

4.3.1 ADC0832 时序

ADC0832 是双通道的模数转换芯片，因此在获取 ADC 转换值之前应该告诉 ADC0832 是使用哪个通道。具体的操作时序如下:

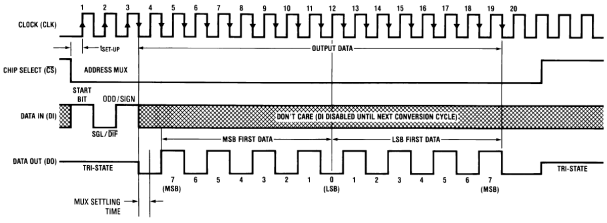

图 9: ADC0832 时序图

结合 ADC0832 的时序图我们可以写出获取 ADC 值的函数，部分程序如下 [?]:

```
#include <wiringPi.h>
#include <stdio.h>

typedef unsigned char uchar;
typedef unsigned int uint;

#define     ADC_CS     0
#define     ADC_CLK    1
#define     ADC_DIO    2

uchar get_ADC_Result(void)
{
    uchar i;
    uchar dat1=0, dat2=0;

    digitalWrite(ADC_CS, 0);
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);     delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);     delayMicroseconds(2);

    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);     delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);     delayMicroseconds(2);
```

```
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,0);      delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);
    digitalWrite(ADC_DIO,1);      delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);      delayMicroseconds(2);

    for(i=0;i<8;i++)
    {
    digitalWrite(ADC_CLK,1);      delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);      delayMicroseconds(2);

    pinMode(ADC_DIO, INPUT);
    dat1=dat1<<1 | digitalRead(ADC_DIO);
    }

    for(i=0;i<8;i++)
    {
    dat2 = dat2 | ((uchar)(digitalRead(ADC_DIO))<<i);
    digitalWrite(ADC_CLK,1);      delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);      delayMicroseconds(2);
    }

    digitalWrite(ADC_CS,1);

    return(dat1==dat2) ? dat1 : 0;
}
```

4.4 数据二值化处理

ADC0832 的转换精度是 8 位，因此理论上可以将光强分为 256 个不同的灰度值，但对于本文探讨的翻译装置，我们不需要用到 255 个灰度值，我们最后生成的一张黑白的照片，根据这个设计要求，我们还需要对 ADC0832

获的数字量 x 进行二值化处理，即根据光强的不同找到一个阈值 Y[1], 当 x>Y 时我们将结果记为 255，x<=Y 时将结果记为 0。该部分程序如下：

```
int choose( int a )
{
      if (a<=YUZHI)
      {
            return 0;   //white
      }else
      {
            return 255;//black
      }
}
```

4.5 生成数据文本

经过二值化的数据就可以利用文本保存起来了。这里使用 C 语言的系统调用将获得的一列即 128 个像素点的二值化后的数据写入到文件 CCD_data.txt 中。部分程序如下:

```
void write_one_col_to_file(int *data)
{
      FILE *f = fopen("CCD_data.txt","a+");
      if (f == NULL)
      {
            printf("Error␣openning␣file!\n");
            exit(1);
      }

      datacount=datacount-128;
      int i;
      for(i=0;i<HEIGHT;i++)
      {
            fprintf(f,"%d\n",data[datacount++]);
```

[1]这里的阈值 Y 可根据光照强度等因素进行修改

```
        }
        fclose(f);
}
```

4.6 综合各个部分

由前面的分析我们已经写好了采集 128 个像素点的函数，这个函数采集了一条线段上的像素点的二值化数字量。我们的目标是生成一张 128*256 分辨率的灰度值为 2 的黑白照片。利用扫描的原理，在采集数据的时候不断改变镜头的位置，重复采集的步骤，如此我们就可以获得 128*256 分辨率的图片数据。此外，需要补充说明的是，在每一次收集数据之前，CCD 需要进行曝光，即进行一轮和采集几乎一样的过程（但这个过程不记录数据）以获得较好的数据值。具体操作如下:

```
int main(void)
{    // Setup Pi
     if(wiringPiSetup() == -1){
          printf("setup␣wiringPi␣failed␣!");
          return 1;
     }
     init();
     baoguang();

     printf("Data␣coletction␣started.\n");
     int i;
     for(i=0;i<WIDTH;i++)
     {
      if(i==64) printf("%25␣completed\n");
      if(i==128) printf("%50␣completed\n");
      if(i==192) printf("%75␣completed\n");
      if(i==256) printf("%100␣completed\n");
      caiji();
      write_one_col_to_file(data);
```

```
        }

        printf("Data coletction finished.\n");
        return 0;
    }
```

4.7 小结

以上讨论的最终目的是获得一个包含 128*256 个像素点的二值化数字量数据文件 CCD_data.txt, 在本次讨论的翻译设备中我将使用以太网口将这个文本文件发送给电脑，而数据处理的工作将会在电脑上面操作。接下来，我们会讨论如何利用 Python 的库函数 Image 将得到的文本文件 CCD_data.txt 绘制成一张分辨率为 128*256 的图片，以及如何降噪以得到更清晰的图片。而传输和接受 CCD_data.txt 的程序我会在附录里面给出，具体不做讨论。

5 图片数据生成图片

对于数据的处理我们可以用很多选择。本文讨论的翻译装置使用 Python 语言进行图像处理。利用该语言的好处之一是它包含了一个图形处理的库，利用这个图形处理库，可以简化我们的操作。例外，接下来要讨论的一个很重要的话题是如何对得到的图片进行降噪处理，本文将通过一个实际的例子向读者展示翻译装置所使用的降噪方法，最后得到一张清晰的图片，而这也是这一部分的最终目的。

5.1 基本数据生成图片

该小结的目的是直接利用 CCD_data.txt 绘制出一张 128*256 的图片——CCD_pic.jpg

5.1.1 Python 获取像素数据

CCD_data.txt 里面文件的格式是由树莓派创建的，每一行只有一个数字，这样的储存方式 Python 处理起来会很方便，即依次提取每一行的字符串，再将字符串的数字转换成整型的数据，暂存在数组当中，具体操作如下:

```
#!/usr/bin/python
import Image,ImageDraw,sys,os

#Get the data,and save it to data[]
WIDTH = 256
HEIGHT= 128
data = [0]*(WIDTH*HEIGHT)
file = open("CCD_data.txt");
lines = file.read().split('\n')
for i in range(WIDTH*HEIGHT):
    data[i] = int(lines[i])
```

5.1.2 绘制基本图像

通过上面的分析，我们已经将 128*256 个像素的二值化数据暂存在了 Python 代码的数组 data 中，接下来我们就可以根据数组 data 的数据，描绘

图片了。具体的步骤是，先创建一张大小为 128*256 的空白图片，然后用一个二重循环根据 data 中的数据，给每一个像素的点进行赋值，保存图片，最终获得基本的图像。这部分 Python 代码如下:

```
#Create a new image
im = Image.new("L",(WIDTH,HEIGHT),255)

#Draw the picture according to the data[]
draw = ImageDraw.Draw(im)
k=0
for i in range(WIDTH):
    for j in range(HEIGHT):
        draw.point((i,j),fill=data[k])
        k+=1
del draw
#Save the picture
im.save("CCD_pic.jpg")
```

5.2 去噪声

通过上面部分的讨论我们得到一张没有经过处理的图片 CCD_pic.jpg, 如下图:

图 10: 未经处理的图片

这是扫描的单词"was", 由上图我们可以看到，除了"was" 本身以外，还

有很多我们不需要的黑点，黑线，另外在我们认为该显示黑色的地方处理了我们认为不应该出现的竖直的” 白线”。尽管这样的结果对于人来说是可以接受的但是，对于后面要用到的 OCR（光学字符识别）软件来说情况就不太乐观了，因此，我们需要对原始得到的图片进行降噪处理，以使得得到的图片更为清晰。

5.2.1 去除竖直的白线

那些被认为不该出现的白线具有这样的特征: 白线的左右两端是黑色的像素，因此，根据这个特点我们可以将满足这个特点的白色的像素点变黑，具体的 Python 代码如下:

```
k=0
for i in range(WIDTH):
    for j in range(HEIGHT):
        if data[k]==255 and i>2 and\
        i<(WIDTH-2) and j>2 and \
        j<(HEIGHT-2) and k>128*4 and\
         k< WIDTH*HEIGHT-4*128:
            if (data[k-2*HEIGHT]==0 and \
            data[k+2*HEIGHT]==0) or \
            (data[k-HEIGHT]==0 and data[k+HEIGHT]==0):
                data[k]=0
        draw.point((i,j),fill=data[k])
        k+=1
```

经过处理的图片如下所示:

图 11: 去除竖直白线的图片

5.2.2 去除横线的黑线

通过对原始图片的观察我们可以发现，对于图片上方的几条黑线，有这样一个特征，他们很”孤单”，即他们的上方或下方没有和他一样的黑色的像素，而对于”was” 字符串图像则不存在这样的现象，因此我们可以根据这样的特征去掉一些水平的黑线。具体的 Python 代码如下:

```
#Let the  herizion noise go away
draw = ImageDraw.Draw(im)
k=0
for i in range(WIDTH):
    for j in range(HEIGHT):
        if data[k]==255 and i>2 \
        and i<(WIDTH-2) and j>2 \
        and j<(HEIGHT-2) and k>128*4 \
        and k< WIDTH*HEIGHT-4*128:
            if (data[k-2*HEIGHT]==0 and \
            data[k+2*HEIGHT]==0) or \
            (data[k-HEIGHT]==0 and data[k+HEIGHT]==0):
                data[k]=0
        draw.point((i,j),fill=data[k])
        k+=1
```

经过处理的图像如下:

图 12: 去除部分水平黑线的图片

同样的操作可以重复一遍，即得到下面的图片:

图 13: 经过两次去水平黑线的图片

5.2.3 去除图片四周的黑点

由上图可以看出，对于字符串“was”本身算是想对比较清晰了，但图片的四周还有很多黑点，接下来可以强制将四周这些黑色的像素变为白色。具体的 Python 代码如下:

```
for i in range(WIDTH):
    for j in range(HEIGHT-17,HEIGHT):
        draw.point((i,j),fill=255)
for i in range(WIDTH):
    for j in range(HEIGHT):
        if (i==0 or i==1  or i==2 \
        or i==WIDTH or i==WIDTH-1 \
```

```
                or  i==WIDTH-2  or  i==WIDTH-3):
                    draw.point((i,j),fill=255)
```

经过以上步骤的处理最终得到的图片如下:

图 14: 最终的图像

5.3 小结

以上我们了如何利用 CCD 采集到的数据生成一张待识别的的图片 CCD_pic.jpg。当然降噪的处理不仅仅这一种，但本文讨论的方法经过实践证明是可行的。接下来我们会继续讨论如何将得到的图片经过 OCR 识别成文本。

6 光学字符识别与后续处理

该部分的最终目标是将 CCD_pic.jpg 上的文字通过 OCR 软件识别出来，生成一个只包含单词的文本文件 BIYESHEJI.txt, 以供查词程序的使用。该部分使用的工具主要有 tesseract 和 Linux 系统内置的命令，接下来会详细的讲述。

6.1 OCR 简介

光学字符识别（英语：Optical Character Recognition, OCR）是指对文本资料的图像文件进行分析识别处理，获取文字及版面信息的过程。[8] 本装置使用的 OCR 识别软件是 Tesseract 3.03。Tesseract 原本是由惠普公司在 1985 年至 1995 期间开发的一款商用 OCR 软件，2005 年时开源，本设备选用这款软件的原因是它的准确度高并且免费。

6.2 OCR 实现过程

本次使用的 Tesseract OCR 引擎是命令行下面的工具，这样的一个好处是便于自动化处理。通过查看 Tesseract 的用户手册可以使用如下的命令利用图片 CCD_pic.jpg 生成识别后的文本文件 BIYESHEJI.txt。

```
tesseract CCD_pic_06.jpg BIYESHEJI -l eng
```

其中 -l eng 表示待识别的语言为英语。这条语句的执行结果是生成了一个包含识别结果的文本 BIYESHIJI.txt

6.3 字符后续处理

由于 OCR 软件识别出的字符含有其他字符，例如 BIYESHEJI.txt 中包含的文字中会多出一行空行，另外在字符的边缘可能会出现其他的多余字符。因此我们在将单词送入查词软件之前需要将文本 BIYESHEJI.txt 进行过滤，过滤掉空行和多余的的字符，采用的具体方法是将多余的字符变成新行字符“\n”, 然后去掉所有的新行字符，过滤掉多余的部分。[9] 具体的命令行操作如下:

```
cat BIYESHEJI.txt | tr -cs [:alpha:] "\n"
```

cat BIYESHEJI.txt 输出文本文件的内容，”|”是管道操作符将上一级的文本输出连接到下一级的输入，tr -cs [:alpha:] ”\n” 将数据流中所有不是字母的字符转化为新行符”\n”, 之后去掉新行符，由此得到只含识别字符的数据流。具体操作如下:

```
jack@MOS ~/Temp $ ls
BIYESHEJI.txt   CCD_pic.jpg
jack@MOS ~/Temp $ cat BIYESHEJI.txt
Was)

jack@MOS ~/Temp $ cat BIYESHEJI.txt \
| tr -cs [:alpha:] "\n"
Was
jack@MOS ~/Temp $
```

7 单词翻译

经过上面的步骤我们已经可以得到过滤过的识别字符，接下来我们继续处理数据流，将数据流向命令行下的翻译软件 sdcv。最终达到翻译的目的。

7.1 sdcv 简介

sdcv 是 console version of StarDict program 的简称，即命令行版本的 StarDict 程序，简单的说就是一个离线翻译工具，词典库需要自行下载。利用 sdcv 我们就可以处理字符数据流，对单词进行翻译。安装好离线词库后，例如需要查询单词”ayatollah” 的中文意思，我们命令行操作如下:

```
sdcv ayatollah
```

通过管道操作我们可以数据流输入到 sdcv 程序中，命令行如下:

```
cat BIYESHEJI.txt | tr -cs [:alpha:] "\n" | sdcv
```

利用上述命令如果文字识别正确，我们已经可以能够得出翻译结果了，但还可以做一些显示上面的处理，我们可以让显示结果只显示前 20 行，不显示前两行。具体的操作如下:

```
cat BIYESHEJI.txt | tr -cs [:alpha:] "\n" | \
sdcv
| head -n 20 | awk 'NR>2{print}'
```

命令执行的结果：

```
jack@MOS ~/Temp $ cat BIYESHEJI.txt | tr -cs \
[:alpha:] "\n" | sdcv | head -n 20 \
| awk 'NR>2{print}'
-->stardict1.3 英汉辞典
-->was

[w□z, wəz]
prep. 是;

-->朗道英汉字典5.0
```

```
--->was

*[w□z]
be 的过去式

--->21 世纪英汉汉英双向词典
--->was

was
(( 轻读 )wEz; wEz:( 重读 )wBz; wCz)
<<动词 >>
```

7.2 词语发音

这一部分是实现词语发音的功能，本转置采用的工具是 Linux 操作系统下的一个文本转语音的软件 espeak, 通过这款软件实现单词的发音。命令行操作如下:

```
cat BIYESHEJI.txt | tr -cs [:alpha:] "\n" | espeak
```

这样便实现了单词的发音功能。最后可以将以上一系列的操作写成自动化脚本，这样就不用每次都去输入，具体的操作如下:

```
Automatic.sh
#!/bin/bash
#This a scrit to use CCD_Translation Divece
#Getting data from CCD to CCD_data.txt
./Jcaiji

#Using CCD_data.txt form CCD_pic.jpg
./display_data_v6.py

#Using OCR to get the word in the CCD_pic.jpg
#and save to BIYESHIJI.txt
tesseract CCD_pic_06.jpg BIYESHEJI -l eng
```

```
#Using espeak to speak the English word
cat BIYESHEJI.txt | tr -cs [:alpha:] "\n" | espeak

#Using sdcv to translate the word
cat BIYESHEJI.txt | tr -cs [:alpha:] "\n" | sdcv \
| head -n 20 | awk 'NR>2{print}'
```

8 结语

本文介绍了基于 Linux 操作系统的线性 CCD 翻译设备的设计，对整个系统的硬件电路和软件程序设计进行分析。建立在操作系统上，程序设计变得简单，灵活性也大大提高。通过对本设备的讨论我们可以利用已有的一些工具，结合实际的需求，设计出相应的产品。例如，我们可以 AT&T Natural Voices 项目优化我们的发音，让发音更真实。又比如我们还可以利用日本开发的语音识别引擎 Julius 对任何语言进行训练识别，通过处理以另一种语言的语音输出，利用这样的设备实现世界人民的无障碍交流等。

9 附录

9.1 源程序

9.1.1 信息获取源程序:Jcaji.c

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>

#define     WIDTH     256
#define     HEIGHT    128
#define     YUZHI     40
#define     ADC_CS     0
#define     ADC_CLK    1
#define     ADC_DIO    2
#define     SI         3
#define     CLK        4

int datacount=0;
int data[WIDTH*HEIGHT];

int get_ADC_Result(void);

void init()
{
    pinMode(SI,  OUTPUT);
    pinMode(CLK, OUTPUT);
    pinMode(ADC_CS,  OUTPUT);
    pinMode(ADC_CLK, OUTPUT);
    digitalWrite(CLK,0);
    digitalWrite(SI,0);
}
```

```
void baoguang()
{
  int i=0;
  digitalWrite(SI,HIGH);
  delayMicroseconds(1);
  digitalWrite(CLK,HIGH);
  delayMicroseconds(1);
  digitalWrite(SI,LOW);
  delayMicroseconds(1);
  digitalWrite(CLK,LOW);
  for(i=0;i<127;i++)
  {
    delayMicroseconds(1);
    delayMicroseconds(1);
    digitalWrite(CLK,HIGH);
    delayMicroseconds(1);
    delayMicroseconds(1);
    digitalWrite(CLK,LOW);
  }
}
//2值化
int choose( int a )
{
        if (a<=YUZHI)
        {
                return 0;
        }else
        {
                return 255;
        }
}
//Use 2D array to get 256*128
//pixels' information
```

```
void caiji()
{

  int k=0;
  int temp=0;
  digitalWrite(SI,HIGH);
  delayMicroseconds(1);
  digitalWrite(CLK,HIGH);
  delayMicroseconds(1);

  digitalWrite(SI,LOW);
  delayMicroseconds(1);

  //delay 25us make image
  //look better
  delayMicroseconds(20);

  pinMode(ADC_DIO, OUTPUT);
  data[datacount++]=choose(get_ADC_Result());
  digitalWrite(CLK,LOW);
  for(k=0;k<127;k++)
  {
    delayMicroseconds(1);
    delayMicroseconds(1);
    digitalWrite(CLK,HIGH);
    delayMicroseconds(1);
    delayMicroseconds(1);
    pinMode(ADC_DIO, OUTPUT);
    data[datacount++]=choose(get_ADC_Result());
    digitalWrite(CLK,LOW);
  }
  delayMicroseconds(1);
  delayMicroseconds(1);
```

```
  digitalWrite(CLK,HIGH);
  delayMicroseconds(1);
  delayMicroseconds(1);
  digitalWrite(CLK,LOW);


}

int get_ADC_Result(void)
{
    int i;
    int dat1=0, dat2=0;

    digitalWrite(ADC_CS, 0);
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);

    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);

    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,0);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);

    for(i=0;i<8;i++)
    {
        digitalWrite(ADC_CLK,1); delayMicroseconds(2);
        digitalWrite(ADC_CLK,0); delayMicroseconds(2);
```

```
        pinMode(ADC_DIO, INPUT);
        dat1=dat1<<1 | digitalRead(ADC_DIO);
    }

    for(i=0;i<8;i++)
    {
        dat2 = dat2 | ((int)(digitalRead(ADC_DIO))<<i);
        digitalWrite(ADC_CLK,1); delayMicroseconds(2);
        digitalWrite(ADC_CLK,0); delayMicroseconds(2);
    }

    digitalWrite(ADC_CS,1);

    return(dat1==dat2) ? dat1 : 0;
}



void write_one_col_to_file(int *data)
{
        FILE *f = fopen("CCD_data.txt","a+");
        if (f == NULL)
        {
                printf("Error␣openning␣file!\n");
                exit(1);
        }

        datacount=datacount-128;
        int i;
        for(i=0;i<HEIGHT;i++)
        {
                fprintf(f,"%d\n",data[datacount++]);
```

```
        }
        fclose(f);
}
int main(void)
{   // Setup Pi
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    init();
    baoguang();

    printf("Data coletction started.\n");
    int i;
    for(i=0;i<WIDTH;i++)
    {
        if(i==64) printf("%25 completed\n");
        if(i==128) printf("%50 completed\n");
        if(i==192) printf("%75 completed\n");
        if(i==256) printf("%100 completed\n");
        caiji();
        write_one_col_to_file(data);
    }

    printf("Data coletction finished.\n");
    return 0;
}
```

9.1.2 图片生成源程序:display_data.py

```
#!/usr/bin/python
import Image,ImageDraw,sys,os

#Get the data,and save it to data[]
WIDTH = 256
HEIGHT= 128
data = [0]*(WIDTH*HEIGHT)
file = open("CCD_data.txt");
lines = file.read().split('\n')
for i in range(WIDTH*HEIGHT):
    data[i] = int(lines[i])

#Create a new image
im = Image.new("L",(WIDTH,HEIGHT),255)

#Draw the picture according
#to the data[]
draw = ImageDraw.Draw(im)

#Eliminate the  vertical noise
#-- make white pixels to black
k=0
for i in range(WIDTH):
    for j in range(HEIGHT):
        if data[k]==255 and i>2 and i<(WIDTH-2)\
                and j>2 and j<(HEIGHT-2) and k>128*4 \
                and k< WIDTH*HEIGHT-4*128:
            if (data[k-2*HEIGHT]==0 and \
                    data[k+2*HEIGHT]==0) \
                    or (data[k-HEIGHT]==0 \
                    and data[k+HEIGHT]==0):
                data[k]=0
```

```
            draw.point((i,j),fill=data[k])
            k+=1

#Eliminate the horizontal noise
#-- make black pixels fade away
k=0
for i in range(WIDTH):
    for j in range(HEIGHT):
        if data[k]==0 and i>2 \
                and i<(WIDTH-2) and j>2 and \
                j<(HEIGHT-2) and k>2 and \
                k< WIDTH*HEIGHT-3*128:
            if data[k]==data[k+2] or \
                    data[k]==data[k-2]:
                data[k]=0
            else:
                data[k]=255
        draw.point((i,j),fill=data[k])
        k+=1

#Eliminate the horizontal noise
#-- make black pixels fade away
k=0
for i in range(WIDTH):
    for j in range(HEIGHT):
        if data[k]==0 and i>2 and i<(WIDTH-2) \
                and j>2 and j<(HEIGHT-2) and\
                k>2 and k< WIDTH*HEIGHT-3*128:
            if data[k]==data[k+1] or data[k]==data[k-1]:
                data[k]=0
            else:
                data[k]=255
        draw.point((i,j),fill=data[k])
```

```
        k+=1

# make the last few lines and the
#beginning lines become white
for i in range(WIDTH):
    for j in range(HEIGHT-17,HEIGHT):
        draw.point((i,j),fill=255)
for i in range(WIDTH):
    for j in range(HEIGHT):
        if (i==0 or i==1  or i==2 or\
                i==WIDTH or i==WIDTH-1\
                or i==WIDTH-2 or i==WIDTH-3):
            draw.point((i,j),fill=255)
del draw

#save to file
im.save("CCD_pic.jpg")
```

9.1.3 字符识别发音脚本:

```
#!/bin/bash
#This a scrit to use CCD_Translation Divece
#Getting data from CCD to CCD_data.txt
./Jcaiji

#Using CCD_data.txt form CCD_pic.jpg
./display_data_v6.py

#Using OCR to get the word in the CCD_pic.jpg
#and save to BIYESHIJI.txt
tesseract CCD_pic_06.jpg BIYESHEJI -l eng

#Using espeak to speak the English word
cat BIYESHEJI.txt | tr -cs [:alpha:] "\n" | espeak

#Using sdcv to translate the word
cat BIYESHEJI.txt | tr -cs [:alpha:] "\n" | sdcv \
| head -n 20 | awk 'NR>2{print}'
```

9.2 原理图

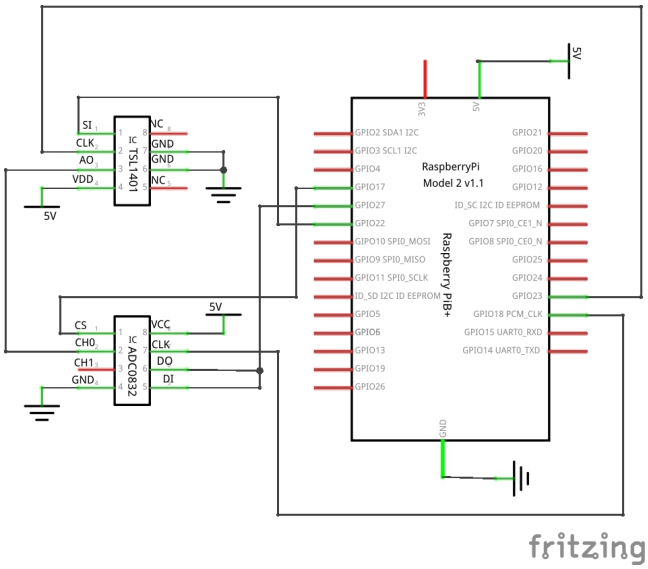

图 15: 原理图

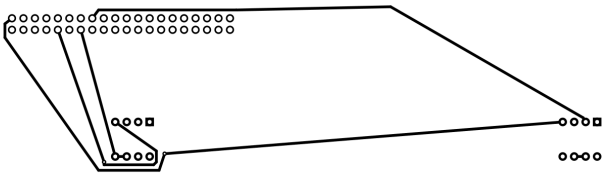

图 16: PCB 图正面

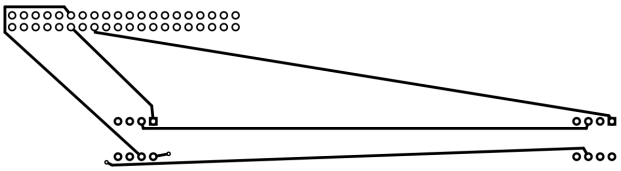

图 17: PCB 图背面

9.3 实物图

图 18:

参考文献

[1] YUE Si-cong, ZHAO Rong-chun, Implementation of Portable Device with off-line OCR and Bi-Linguist Translation Based on DSP, 2006,Vol.27,No.2,MINI-MICRO SYSTEMS,269-271

[2] YANG Shao-Peng, GAO Mei-feng, A design of high-speed linear CCD data acquisition system,2014,Vol.22,No.11,Electronic Design Engineering,127

[3] httqps://zh.wikipedia.org/wiki/感光耦合元件

[4] http://pdf1.alldatasheet.com/datasheet-pdf/view/203031/TAOS/ TSL1401R-LF.html

[5] https://arduining.com/2014/03/26/using-the-linear-sensor-array- tsl201r-with-arduino/

[6] https://zh.wikipedia.org/wiki/树莓派

[7] http://baike.baidu.com/link?url=43KrOBqTi1t4Upny0YXN4i7YbUJMTWsoDFPwBit8P968B0aH xiiWIU8jEkcdvPSXXZ fR4z0uXJRiq Dx66qgPKq

[8] https://zh.wikipedia.org/wiki/光学字符识别

[9] Harley Hahns.Harley Hahns Guide to Unix and Linux.McGraw-Hill Higher Education,2009,Chpter16-19

10 致谢

经过这段时间的努力，我的毕业论文《线性 CCD 翻译设备》终于完成了。在这里我要感谢我的同学，朋友，亲人。我的同学给我提供了很多参考资料，朋友给了我很多支持，亲人在这段时间给了我无微不至的关怀。在这里我还要特别感谢我的指导老师，杨伟超老师，在毕业设计和论文写作上都给了我很好的建议，没有他我的毕业设计也不可能在这段时间内完成。