语音信号处理课程设计论文

课程设计

论文题目: 基于 Python 的树莓派

天气闹钟



课程名称 语音信号处理

任课教师 彭醇陵

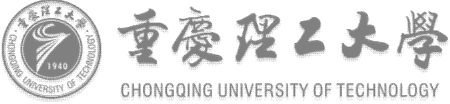
所在学院 电气与电子工程学院

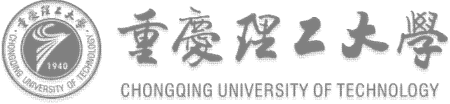
姓 名

学 号

同 组 者

提交日期 2022/06/30



目录

[一、引言 5](#_Toc21098)

[二、技术支撑 5](#_Toc21099)

[1、SYN6288芯片 5](#_Toc21100)

[1.1SYN6288芯片系统构成框图 5](#_Toc21101)

[1.2 SYN6288可实现的主要功能 6](#_Toc21102)

[1.2.1文本合成功能 6](#_Toc21103)

[1.2.2文本智能分析处理 6](#_Toc21104)

[1.2.3数字音量16级控制和6级词语语速控制 7](#_Toc21105)

[1.2.4支持多种控制命令 7](#_Toc21106)

[1.2.5支持低功耗模式 7](#_Toc21107)

[1.2.6支持三种通讯波特率 7](#_Toc21108)

[2.Raspberry Operating System 8](#_Toc21109)

[3.Linux 发行系统及其命令集 8](#_Toc21110)

[三、设计思路 9](#_Toc21111)

[四、效果演示(见附件视频) 9](#_Toc21112)

[五、测试环境与硬件设备 9](#_Toc21113)

[六、实物展示 10](#_Toc21114)

[参考文献 11](#_Toc21115)

[附件一：main.py 12](#_Toc21116)

[附件二：SNY6288.py 13](#_Toc21117)

[附件三：Get.py 15](#_Toc21118)

基于Python的树莓派天气闹钟

摘要：在当前市场下，一个纯粹的天气语音闹钟是很难见到的，市场上的闹钟趋于商业利益化，捆绑了一大堆用不到的功能；因此，本着熟练专业知识、设计一个纯粹的天气语音闹钟(可实现语音交互的实时语音播报当前天气信息和定时闹钟功能)的目的，设计了基于树莓派3B的语音天气闹钟系统。实际应用时，利用传感器采集语音数据或是Linux定时触发程序，播放天气预报。实验结果表明，定时触发预报成功率高达99%。

关键词：树莓派；天气预报；定时；SYN8266；

# 一、引言

随着大学生活走到最后一年，专业知识也已经基本学完，本着融会贯通专业知识、锻炼专业素养的目的，着手设计一款只能语音播报闹钟。

目前市场上普遍的是内置的语音信息，不具备可编程性，并且也很少做到联网获取实时天气预报信息。针对现状，设计了一款基于树莓派的，既可以语音主动唤醒又可以定时定点唤醒播放天气信息的可编程闹钟。

# 二、技术支撑

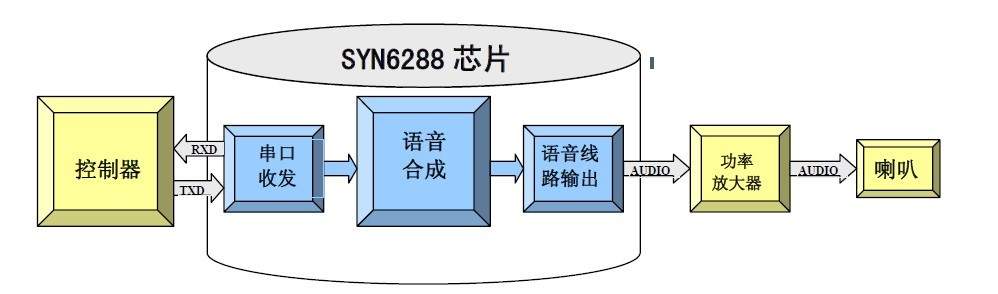
## 1、SYN6288芯片

SYN6288 中文语音合成芯片是北京宇音天下科技有限公司于2010 年初推出的一款性/价比更高，效果更自然的一款中高端语音合成芯片。SYN6288通过异步串口（UART）通讯方式，接收待合成的文本数据，实现文本到语音（或TTS 语音）的转换。

### 1.1SYN6288芯片系统构成框图

最小系统包括:控制器模块、SYN6288语音合成芯片、功放模块和喇叭。

主控制器和 SYN6288 语音合成芯片之间通过 UART 接口连接，控制器可通过通讯接口向SYN6288语音合成芯片发送控制命令和文本，SYN6288语音合成芯片把接收到的文本合成为语音信号输出，输出的信号经功率放大器进行放大后连接到喇叭进行播放。



SYN6288芯片系统构成框图（图2）

### 1.2 SYN6288可实现的主要功能

#### 1.2.1文本合成功能

芯片支持任意中文文本的合成，可以采用GB2312、GBK、BIG5 和Unicode 四种编码方式。芯片支持英文字母的合成，遇到英文单词时按字母方式发音。

每次合成的文本量可达200个字节。

#### 1.2.2文本智能分析处理

芯片具有文本智能分析处理功能，对常见的数值、电话号码、时间日期、度量衡符号等格式的文本，芯片能够根据内置的文本匹配规则进行正确的识别和处理。

例如：“2008-12-21”读作 “二零零八年十二月二十一日”，“10:36:28” 读作 “十点三十六分二十八秒”，“28℃”读作“二十八摄氏度”，等等。

表1.2.2-1 文字处理控制标记

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 设置数字处理策略 | 全局 | [n?] | ? 为0，自动判断  ? 为1，数字作号码处理  ? 为2，数字作数值处理  ? 为其它无符号整数时，将视为整数0处理 | [n0] |
| 设置号码中“1”的读法 | 全局 | [y?] | ? 为0，合成号码时“1”读成“幺”  ? 为1，合成号码时“1”读成“一”  ? 为其它无符号整数时，将视为整数0处理 | [y0] |
| 设置标点是否读出 | 全局 | [b?] | ? 为0，不读标点  ? 为1，读标点  ? 为其它无符号整数时，将视为整数0处理 | [b0] |
| 设置文本朗读方式 | 全局 | [o?] | ? 为0，设置为自然朗读方式  ? 为1，设置为Word-By-Word方式  ? 为其它无符号整数时，将视为整数0处理 | [o0] |

#### 1.2.3数字音量16级控制和6级词语语速控制

芯片可实现 16 级数字音量控制，音量更大，更广。播放文本的前景音量和播放背景音乐的背景音量可分开控制，更加自由。

表1.2.3-1 文字处理控制标记

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 设置背景音乐音量 | 全局 | [m?] | ? 为音量值，取值：0～16 (其中0为静音)  ? 为其它无符号整数时，将视为最大音量值 16 处理 | [m4] |
| 设置词语语速（针对自然朗读方式） | 全局 | [t?] | ? 为词语语速值，取值：0～5  ? 为其它无符号整数时，将视为最高词语语速值5 处理  0 级为最慢词语语速，5 级为最快词语语速。注意：对于 Word-by-Word 朗读方式不支持词语语速调节 | [t4] |
| 设置前景文本播放音量（含提示音效） | 全局 | [v?] | ? 为音量值，取值：0～16 (其中0为静音)  ? 为其它无符号整数时，将视为最大音量值16处理 | [v10] |



#### 1.2.4支持多种控制命令

控制命令包括：合成文本、停止合成、暂停合成、恢复合成、状态查询、进入 Power Down 模式、改通讯波特率等控制命令。控制器通过通讯接口发送控制命令实现对芯片的控制。

#### 1.2.5支持低功耗模式

芯片支持 Power Down 模式。使用控制命令可以使芯片进入 Power Down

模式。复位芯片可以使芯片从Power Down 模式恢复到正常工作模式。

#### 1.2.6支持三种通讯波特率

芯片支持的通讯波特率：9600bps，19200bps、38400bps 。

## 2.Raspberry Operating System

2.1 Raspberry Pi 3B 简介

树莓派（英语：Raspberry Pi）英国树莓派基金会开发的微型单板计算机，目的是以低价硬体及自由软体促进学校的基本电脑科学教育。

树莓派系列计算机每一代均使用博通

（Broadcom）出产的ARM架构处理器，如今生产的机型（树莓派4B）内存在2GB和8GB之间，主要TF卡作为系统存储媒体（初代使用SD卡），配备USB接口和HDMI的视频输出（支持声音输出），内置 Ethernet/WLAN/Bluetooth 网络链接的方式（依据型号决定），并且可使用多种操作系统。产品线型号分为A型、B型、Zero型和ComputeModule计算卡。

表1树莓派3B硬件资源

|  |  |  |  |
| --- | --- | --- | --- |
| Soc | CPU | GPU | 内存 |
| Broadcom BCM2837 | ARM Cortex A53(64) | BVCoreIV[27]… | 1 GB (LPDDR2) |
| USB接口个数 | 影像输入 | 影像输出 | 音源输入 |
| USB 2.0（\*4） | 15-针头 MIPI 相机 | HDMI、MIPI DSI | I²S |
| 板载存储 | 网络接口 | GPIO引脚数 | 外设 |
| MicroSD 卡插槽 | 以太网/WLAN | 40 | GPIO、HAT |
| 额定功率 | 电源输入 | 总体尺寸 | 音源输出 |
| (5V/2.1A) | 5V | 85.60 × 53.98 mm | 3.5mm插孔 |

## 3.Linux 发行系统及其命令集

Linux 是一个基于 Linux 的开源类 Unix 操作系统家族内核。由 Linus

Torvalds 于 1991 年 9 月 17 日首次发布的操作系统内核。Linux 可以在嵌入式系统上运行，即操作系统通常内置在固件中并针对系统高度定制的设备。

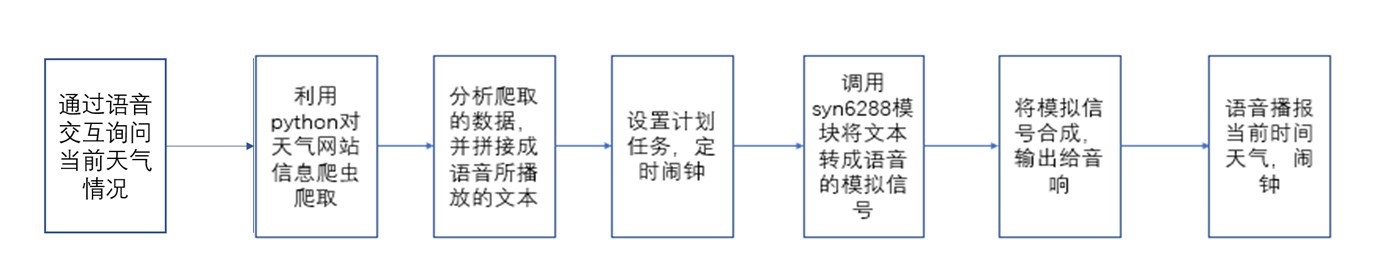
Linux 是免费和开源软件协作最突出的例子之一。任何人都可以根据其各自的许可条款。本次使用的 Raspberry 3B 使用的系统就是 Linux 发行系统之

一。

表3-1 常用的命令集

|  |  |
| --- | --- |
| 命令名 | 作用 |
| ls | List-列出该目录下的所有文件(夹) |
| cd dirName | ChangeDirectory-切换目录到dirName |
| mkdir folderName | MakeDirectory-新建文件夹folderName |
| rm file/folderName | Remove-删除文件(夹) file/folderName |
| mv src dst | Move-移动文件src到dst |

# 三、设计思路

 系统逻辑框图（图1）

有两种激活系统的方式：

方法一：系统定时自动触发。配置系统定时任务，到了时间后由系统自动触发程序；然后通过python程序将网站天气信息爬虫下来，同时将爬取的零散的信息转化成一段文本内容，调用sy5n6288模块将文本转成语音的模拟信号，将模拟信号合成，输出给音响，完成播报。

方法二：语音信号被动触发。首先询问当前天气信息，然后树莓派通过 python将网站天气信息爬虫下来，同时将爬取的零散的信息转化成一段文本内容。提前设置好计划任务，定时闹钟。调用sy5n6288模块将文本转成语音的模拟信号，将模拟信号合成，输出给音响。通过音响语音播报当前时间天气，闹钟。

# 四、效果演示(见附件视频)

场景一(演示人)：问询天气，例如：今天天气怎么样？场景二(Raspberry)：正在为您查询…

>>树莓派开始抓取网上的天气数据，并将数据交由Syn6288处理场景三(Raspberry)：上午/下午好，今天是…

>>Syn6288将处理得到的语音数据传输至音响设备完成播报

场景四(演示人)：设置定时器

>>树莓派等待时间到来…

>>时间到点了！树莓派开始抓取网上的天气数据，并将数据交由Syn6288 处理

场景五(Raspberry)：上午/下午好，今天是…

>> Syn6288将处理得到的语音数据传输至音响设备完成播报

# 五、测试环境与硬件设备

测试环境：

Raspberry 3B系统环境：Debian GUN/Linux 11(bullseye)

Python环境：Python 3.9.2

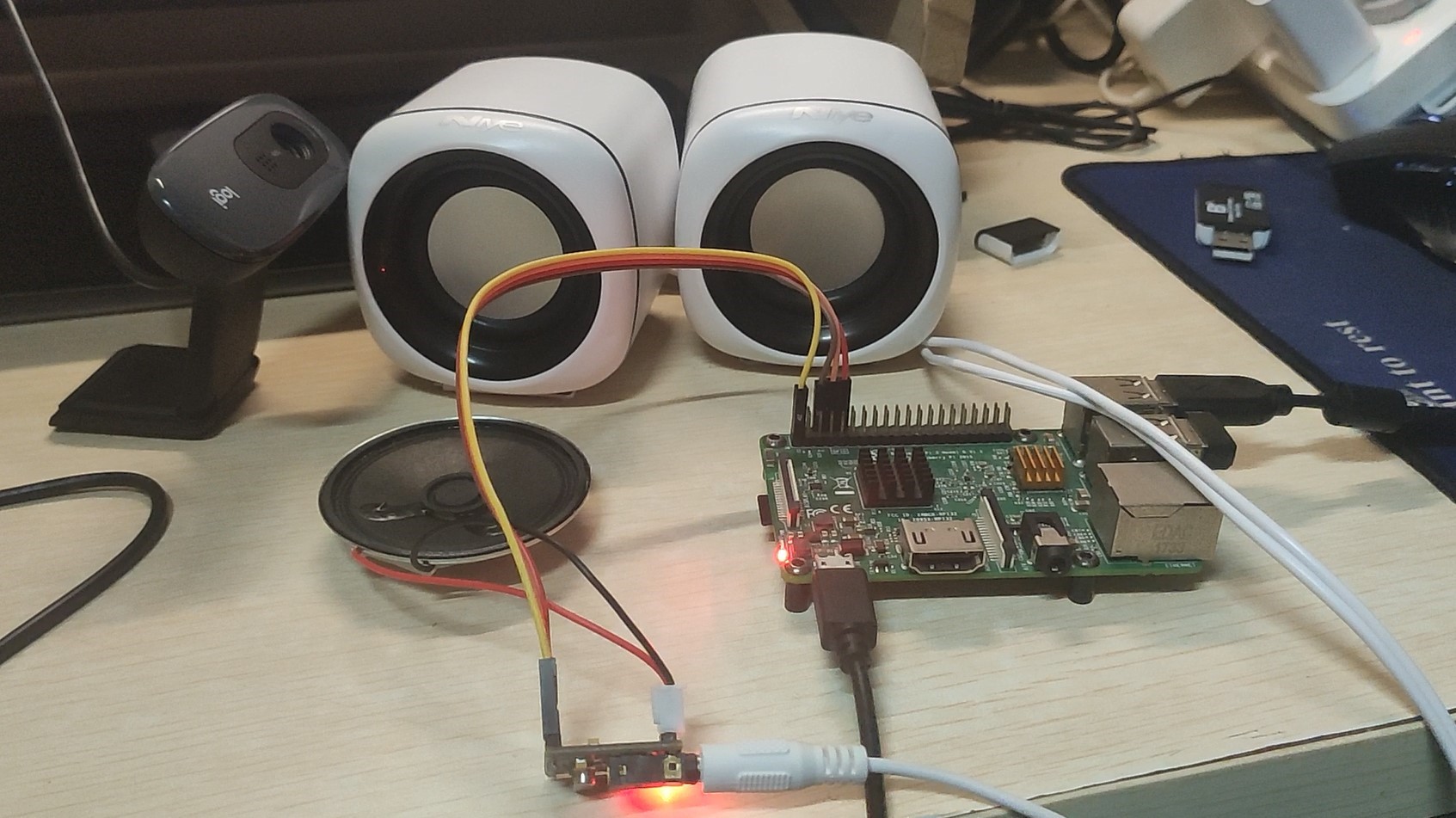
Windows Operating System：Windows FEP 1000.22632.1000.0

硬件设备：

Raspberry Pi 3B\*1、 音箱(可接 3.5mm 音频线)\*1、 3.5mm 音频线\*1、

SAMSUNG SD Card（32GB-USB3.0）\*2、 TF 卡/MicroSD 卡读卡器\*1、 Raspberry Pi 3B 电源适配器 5V@2.5A、 SYN6288 语音合成模块\*1、 杜邦线\*20

# 六、实物展示



# 参考文献

[1]陈丽媚,张学娜,易向东.基于 Arduino 的 AI 语音识别智能音箱设计[J].科学技术创新,2020(19):57-58.

[2]李静波,邹黎明,李茂,付硕,李硕.智能语音客服的响应时间研究[J].人类工效学,2020,26(04):29-34.DOI:10.13837/j.issn.1006-8309.2020.04.0006.

[3]成童,张志杰.人机交互中的系统反应时间研究[J].人类工效学,2017,23(05):69-

74.DOI:10.13837/j.issn.1006-8309.2017.05.0013.

[4]牛芳,吾守尔·斯拉木.韵律增强型汉语语音合成系统[J].现代电子技术,2022,45(13):87-92.DOI:10.16652/j.issn.1004-373x.2022.13.017.

[5]范会敏,何鑫.中文语音合成系统的设计与实现[J].计算机系统应用,2017,26(02):73-

77.DOI:10.15888/j.cnki.csa.005616.

[6]王朝民,谢湘,匡镜明. 一种嵌入式中文语音合成系统非周期成分音节层建模方法 [C]//.第十二届全国人机语音通讯学术会议（NCMMSC2013）论文集.[出版者不详],2013:195-198.

[7]周开来.中文语音合成系统过程分析及实现初探[J].现代计算机(专业

版),2010(04):73-77.

# 附件一：main.py

|  |
| --- |
| import Get import time import syn6288  Info=str(Get.WeatherInfo()) Info\_split=Info.split("：") print(Info) syn = syn6288.SYN6288() for character in Info\_split:  time.sleep(5) try:  syn.play\_text(character) while not syn.query\_state():  time.sleep(0.01) print(character) except:  print("Failed on play...") time.sleep(1) |

# 附件二：SNY6288.py

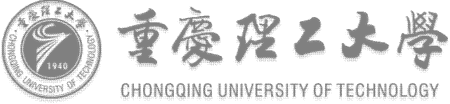


|  |
| --- |
| # -\*- coding: GBK -\*- import serial  class SYN6288: def \_\_init\_\_(self) -> None:  # self.\_\_ser = serial.Serial('/dev/ttyAMA0', 9600, timeout=0.5) self.\_\_ser = serial.Serial('/dev/ttyS0', 9600, timeout=0.5) def \_\_del\_\_(self) -> None:  self.\_\_ser.close()  def play\_text(self, text: str) -> bool: gbk = text.encode('GBK') if len(gbk) > 200: print(len(gbk)) gbk = gbk[:200]  frame = self.\_\_gen\_frame(b'\x01\x01', data=gbk) print("encode(GBK): starting...")  self.\_\_ser.write(frame) print("Frame\_Write: starting...") while True: state = self.\_\_ser.read(1) if state == b'O' or state == b'N': continue  return state == b'A'  def query\_state(self) -> bool: # ¿ÕÏÐÊ±·µ»ØTrue£¬Ã¦Ê±·µ»ØFalse frame = self.\_\_gen\_frame(b'\x21')  self.\_\_ser.write(frame) while True: state = self.\_\_ser.read(1) if state == b'A' or state == b'E': continue  return state == b'O'  def stop\_synthesis(self): frame = self.\_\_gen\_frame(b'\02') self.\_\_ser.write(frame) self.\_\_ser.read(1)  def pause\_synthesis(self): frame = self.\_\_gen\_frame(b'\03') self.\_\_ser.write(frame) self.\_\_ser.read(1)  def continue\_synthesis(self): frame = self.\_\_gen\_frame(b'\04') self.\_\_ser.write(frame) self.\_\_ser.read(1)    @staticmethod def \_\_calc\_bcc(bytes: bytes) -> bytes: |



|  |
| --- |
| result = 0 for byte in bytes: result = result ^ byte return result.to\_bytes(1, 'big')    @staticmethod def \_\_gen\_frame(cmd\_word: bytes, data: bytes = b'') -> bytes: result = b'\xfd' limited\_data = data if len(data) > 200: limited\_data = data[:200]  data\_len = (len(cmd\_word) + len(limited\_data) + 1).to\_bytes(2,  'big') result += data\_len + cmd\_word + data bcc = SYN6288.\_\_calc\_bcc(result) return result + bcc |

# 附件三：Get.py



|  |
| --- |
| # -\*- coding: GBK -\*- import os import re import time import requests  from datetime import datetime, timedelta  from bs4 import BeautifulSoup import sys,importlib importlib.reload(sys)    headers = {  'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64)  AppleWebKit'  '/537.36 (KHTML, like Gecko)  Chrome/53.0.2785.143 Safar'  'i/537.36',  } def numtozh(num): num\_dict = {1: '一', 2: '二', 3: '三', 4: '四', 5: '五', 6: '六', 7 '七',  8: '八', 9: '九', 0: '零'} num = int(num) if 100 <= num < 1000: b\_num = num // 100  s\_num = (num-b\_num\*100) // 10 g\_num = (num-b\_num\*100) % 10 if g\_num == 0 and s\_num == 0: num = '%s百' % (num\_dict[b\_num]) elif s\_num == 0: num = '%s百%s%s' % (num\_dict[b\_num], num\_dict.get(s\_num,  ''), num\_dict.get(g\_num, '')) elif g\_num == 0: num = '%s百%s十' % (num\_dict[b\_num], num\_dict.get(s\_num,  '')) else: num = '%s百%s十%s' % (num\_dict[b\_num], num\_dict.get(s\_num,  ''), num\_dict.get(g\_num, '')) elif 10 <= num < 100: s\_num = num // 10 g\_num = (num-s\_num\*10) % 10 if g\_num == 0: g\_num = '' num = '%s十%s' % (num\_dict[s\_num], num\_dict.get(g\_num, '')) elif 0 <= num < 10:  g\_num = num  num = '%s' % (num\_dict[g\_num]) elif -10 < num < 0: g\_num = -num num = '零下%s' % (num\_dict[g\_num]) elif -100 < num <= -10: num = -num s\_num = num // 10 g\_num = (num-s\_num\*10) % 10 |

:

|  |
| --- |
| if g\_num == 0: g\_num = ''  num = '零下%s 十%s' % (num\_dict[s\_num], num\_dict.get(g\_num, '')) return num    def get\_weather():  # 下载墨迹天气主页源码  res = requests.get('http://tianqi.moji.com/', headers=headers)  # 用BeautifulSoup获取所需信息  soup = BeautifulSoup(res.text, "html.parser") temp = soup.find('div', attrs={'class': 'wea\_weather clearfix'}).em.getText() temp = numtozh(int(temp))  weather = soup.find('div', attrs={'class': 'wea\_weather clearfix'}).b.getText()  sd = soup.find('div', attrs={'class': 'wea\_about clearfix'}).span.getText()  sd\_num = re.search(r'\d+', sd).group() sd\_num\_zh = numtozh(int(sd\_num)) sd = sd.replace(sd\_num, sd\_num\_zh)  wind = soup.find('div', attrs={'class': 'wea\_about clearfix'}).em.getText()  aqi = soup.find('div', attrs={'class': 'wea\_alert clearfix'}).em.getText()  aqi\_num = re.search(r'\d+', aqi).group() aqi\_num\_zh = numtozh(int(aqi\_num))  aqi = aqi.replace(aqi\_num, aqi\_num\_zh).replace(' ', ',空气质量') info = soup.find('div', attrs={'class': 'wea\_tips clearfix'}).em.getText()  sd = sd.replace(' ', '百分之').replace('%', '')  aqi = '实时空气质量指数' + aqi  info = info.replace('，', ',')  # 获取今天的日期  today = datetime.now().strftime('%p %Y年%m月%d日 %I点%M分%S秒') today = today.replace('PM', '下午好！今天是').replace('AM', '上午好！今天是')  # 将获取的信息拼接成一句话  text = '%s, 实时天气是%s, 温度%s 摄氏度 ,%s ,%s ,%s ,%s' % \  (today, weather, temp, sd, wind, aqi, info) return text def text2voice(text): url =  'http://tts.baidu.com/text2audio?idx=1&tex={0}&cuid=baidu\_speech\_' \ 'demo&cod=2&lan=zh&ctp=1&pdt=1&spd=4&per=4&vol=5&pit=5'.form at(text) def WeatherInfo():  # 获取需要转换语音的文字  text = str(get\_weather()) return text text2voice(text) |





|  |
| --- |
| 批阅教师意见 |
| 经综合评价，论文得分为：          批阅教师签名：    批阅日期： |