语音英文播报

1. 学习目标

本次课程我们主要学习使用树莓派和语音合成播报模块实现语音英文播报。

2. 课前准备

语音合成播报模块采用的是 I2C 通讯，将模块的 SDA，SCL 分别连接树莓派板子的 SDA 和 SCL 引脚。VCC 和 GND 分别连接树莓派的 5V 和 GND。树莓派需要开启 I2C 服务。

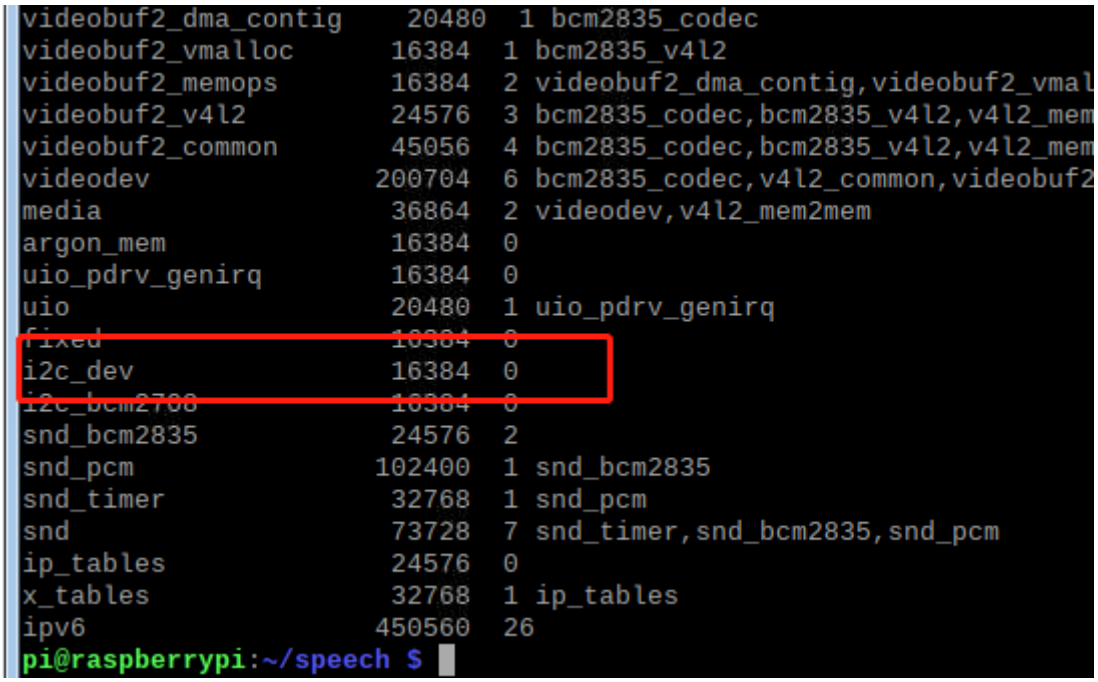
引脚定义

Raspberry Pi GPIO Header + PoE Header

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

01	TR01		TR00	02
03	TR03		TR02	04

树莓派 I2C 开启后终端输入 lmusb，查看 I2C 是否成功启动



下载 i2c-tools，该软件可以对硬件设备的使用情况、故障进行监控

终端输入 sudo apt-get install i2c-tools

通过终端输入 i2cdetect -y -a 1 查看是否有检测到模块的地址：0x30

注意新旧版语音播报模块 IIC 地址不同，新版 IIC 地址为:0x30,旧版为 0x50。如果使用不正常可以尝试修改另一个 IIC 地址测试。

3.语音模块协议

语音播报是通过 I2C 通讯间按照规定的数据帧进行配置和使用的，具体请查看语音合成播报模块通讯协议文档教程。

4. 程序

模块操作用定义相应的函数和枚举。设置单词的发音方式，由模块协议文档中的协议可知，通过写入相应文本符号可以设置相应的功能，如[h?]为设置单词的发音方式，通过设置枚举变量，定义相应的设置函数。

设置单词的发音方式	[h?]	? 为 0，自动判断单词发音方式	[h1]
		? 为 1，字母发音方式	
		? 为 2，单词发音方式	

```
Articulation_Type = {
                    'Articulation_Auto':0,#为 0，自动判断单词发音方式
                    'Articulation_Letter':1,#为 1，字母发音方式
                    'Articulation_Word':2#为 2，单词发音方式
                    }#设置单词的发音方式 [h?]

def SetArticulation(num):
    TextCtrl('h',num)
    while GetChipStatus() != ChipStatus_Type['ChipStatus_Idle']:
        time.sleep(0.002)
```

设置单词的发音方式函数中字符串 h 和选择单词发音方式的枚举变量通过调用文本控制函数将字符串和数值变量合成新的符合文本控制格式的字符串，再调用帧发送函数完成设置。

```
def TextCtrl(ch,num):
    if num != -1:
        str_T = '[' + ch + str(num) + ']'
        SetBase(str_T)
    else:
        str_T = '[' + ch + ']'
        SetBase(str_T)
```

帧发送函数，将字符串按照命令帧的格式，计算对应字符串的高低直接数值，通过列表将所有的设置值写入列表中。

命令帧格式结构	帧头	数据区长度		数据区		
	0xFD	高字节	低字节	命令字	文本编码格式	待合成文本
		0xHH	0xLL	0x01	0x00~0x03	…… ……

```
def SetBase(str_):
    str_ = str_.encode('gb2312')
    size = len(str_)+2

    DataHead = date_head
    Length_HH = size>>8
    Length_LL = size & 0x00ff
    Commond = 0x01
    EncodingFormat = 0x00

    Date_Pack = [DataHead,Length_HH,Length_LL,Commond,EncodingFormat]

    I2C_WriteBytes(Date_Pack)

    I2C_WriteBytes(str_)
```

当语音合成模块正在合成文本的时候，如果又收到一帧有效合成命令帧，芯片会立刻停止当前正在合成的文本，转而合成新收到的文本，这里就需要利用查询芯片状态的命令，只有当检测到芯片空闲的时候再发送新的命令帧。通过查看模块通讯协议文本可知模块对应状态的返回值，相应的也在类中定义了相应的变量。

回传数据类型	回传数据	触发条件
初始化成功	0x4A	芯片初始化成功
收到正确命令帧	0x41	收到正确的命令帧
收到错误命令帧	0x45	收到错误的命令帧
芯片忙碌	0x4E	收到“状态查询命令”，芯片处于合成文本状态回传 0x4E
芯片空闲	0x4F	当一帧数据合成完以后，芯片进入空闲状态回传 0x4F；当芯片收到“状态查询命令”，芯片处于空闲状态回传 0x4F

```
ChipStatus_Type = {
                    'ChipStatus_InitSuccessful':0x4A,#初始化成功回传
                    'ChipStatus_CorrectCommand':0x41,#收到正确的命令帧回传
                    'ChipStatus_ErrorCommand':0x45,#收到不能识别命令帧回传
                    'ChipStatus_Busy':0x4E,#芯片忙碌状态回传
                    'ChipStatus_Idle':0x4F #芯片空闲状态回传
                  }
```

根据模块通讯协议可知获取模块的帧协议规则，并定义相应的获取模块状态的函数。

名称	发送的数据	说明
命令字	0x21	通过该命令获取相应参数，来判断 TTS 芯片是否处在合成状态，返回 0x4E 表明芯片仍在合成中，返回 0x4F 表明芯片处于空闲状态
参数列表	无	
命令帧格式结构	帧头 / 数据区长度 / 数据区 0xFD / 高字节 / 低字节 / 命令字 0x00 / 0x01 / 0x21	

```
def GetChipStatus():
    global i2c_addr
    AskState = [0xfd,0x00,0x01,0x21]
    try:
        I2C_WriteBytes(AskState)
        time.sleep(0.05)
    except:
        print("I2CRead_Write error")


    try:
        Read_result = bus.read_byte(i2c_addr)
        return Read_result
    except:
        print("I2CRead error")
```

设置模块发音人和播报音量，然后设置播报的文本和芯片状态的检测，当芯片空闲才可以进行新的设置和文本播报。

```
'''
程序执行部分，以上为配置相关的函数和字典定义
'''
SetArticulation(Articulation_Type["Articulation_Word"])

SetReader(Reader_Type["Reader_XiaoPing"])
SetVolume(10)
Speech_text("hello yahboom intelligent Technology",EncodingFormat_Type["GB2312"])
while GetChipStatus() != ChipStatus_Type['ChipStatus_Idle']:
    time.sleep(0.1)


SetReader(Reader_Type["Reader_XuDuo"])
Speech_text("Welcome to use yahboom intelligent technology voice broadcast module",EncodingFormat_Type["GB2312"])
while GetChipStatus() != ChipStatus_Type['ChipStatus_Idle']:
    time.sleep(0.1)
```

5. 运行程序

在树莓派终端输入 python3 English_speech.py 即可运行程序。

6. 实验现象

程序运行成功后，会依次播报，由晓萍播报“hello yahboom intelligent Technology”，由许多播报“Welcome to use yahboom intelligent technology voice broadcast module”。