智能系统与控制

树莓派网络控制：上传乐谱演奏

于泓
鲁东大学
信息与电气工程学院
2022.5.1

任务： 利用前面学过的PWM蜂鸣器启动实验，实现通过网页上传乐谱，驱动蜂鸣器演奏的功能

music-1.txt - 记事本

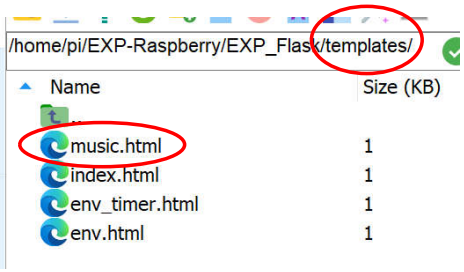
文件(F) 编辑(E) 格式(O)

```
M 1 1
M 1 1
M 1 2
L 5 2
M 3 1
M 3 1
M 3 2
M 1 2
M 1 1
M 3 1
M 5 2
M 5 2
M 4 1
M 3 1
M 2 3
M 2 1
M 3 1
M 4 2
M 4 2
```

乐谱格式： 分为三列
音高 音符 节拍

基本页面

前端代码:




```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>乐曲演奏</title>
</head>
<body>
    <h1>上传乐谱并演奏</h1>

    <form action="" method='POST' enctype="multipart/form-data" >
        <input type="file" name="file" value ={{file_name}}/>
        <br>

        <input type="submit" value="演奏" name= "music_play"/>
        <br>

        <input type="submit" value="停止" name = "music_stop"/>
        <br>
        <span>{{message_file_error}}</span>
    </form>
     <a href="/">灯光控制</a>
     <a href="/env">环境显示</a>
</body>
</html>
```

buzzer.py

```
import threading
import RPi.GPIO as GPIO
from pin_dic import pin_dic
import numpy as np
import time
class Runing_Song(threading.Thread):
    def __init__(self,pin):
        super(Runing_Song, self).__init__()

        # 设置蜂鸣器引脚模式
        self.pin_buzzer = pin
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(self.pin_buzzer,GPIO.OUT)

        self.delay_beat = 0.2

        self.freqs = []
        self.beats = []

        self.flag_stop = False
```

```python
def file_load(self,file_music):

    # 从文件中加载乐谱数据
    data = np.loadtxt(file_music,dtype = 'str')
    [n,d] = np.shape(data)

    # 必须是3列
    if not d==3:
        return False,0,0
    # 预先定义好的音符频率
    CL = [0, 131, 147, 165, 175, 196, 211, 248]
    CM = [0, 262, 294, 330, 349, 392, 440, 494]
    CH = [0, 525, 589, 661, 700, 786, 882, 990]


    # 第一列音高 第二列音频 第三列音长
    levels = data[:,0]
    beats = data[:,2]
    beats = beats.astype('int32')
    labs =  data[:,1]
    labs = labs.astype('int32')


    # 生成乐谱
    self.freqs  = []
    for i in range(n):
        if levels[i]=='H':
            self.freqs.append(CH[labs[i]])
        elif levels[i] == 'M':
            self.freqs.append(CM[labs[i]])
        elif levels[i] == 'L':
            self.freqs.append(CL[labs[i]])
```

```python
if not len(self.freqs)==len(self.beats):
    return False
else:
    return True
```

```
def dostop(self):
    self.flag_stop = True


def run(self):

    self.flag_stop = False

    # 定义PWM对象
    Buzzer = GPIO.PWM( self.pin_buzzer , 440)
    Buzzer.start(50)
    while True:
        if self.flag_stop:
            break
        for freq,beat in zip(self.freqs,self.beats):
            if self.flag_stop:
                break
            Buzzer.ChangeFrequency(freq)
            time.sleep(self.delay_beat*beat)

    Buzzer.stop()
    GPIO.output(self.pin_buzzer, GPIO.LOW)

    self.flag_stop = False
```

线程运行

main.py

```
#  蜂鸣器
global m_runing_song
m_runing_song = Runing_Song(pin_dic['G18'])
```

创建全局对象

```
@app.route('/music',methods=['GET', 'POST'])
def music():
    global m_runing_song
    if request.method == 'GET':
        return render_template('music.html')

    if request.form.get('music_stop', None) == "停止":

        print('music off')
        if m_runing_song.isAlive() == True:
            m_runing_song.dostop()
            time.sleep(0.1)
            m_runing_song.join()
        return redirect(url_for('music'))
```

添加路由

判断线程是否启动

```
# 进行文件验证
# 如果文件存在
if  request.files:
    # 验证文件类型
    f = request.files['file']
    f_name = f.filename
    ext = f_name.rsplit('.', 1)[1]
    # 报文件类型错误信息
    if not (f and ext=='txt'):
        return render_template('music.html',message_file_error="文件类型错误，只支持txt")

    # 没有问题进行文件存储
    basepath = os.path.dirname(__file__)  # 当前文件所在路径
    upload_path = os.path.join(basepath, 'static', secure_filename(f.filename))
    f.save(upload_path)

else:
    return render_template('music.html',message_file_error="文件为空")
```

有错误 页面返回

```
# 点击演奏按钮
if request.form.get('music_play', None) == "演奏":

    print('music on ')
    if m_runing_song.isAlive() == False:
        # 没有线程 创建线程并启动
        m_runing_song = Runing_Song(pin_dic['G18'])
        m_runing_song.file_load(upload_path)

        m_runing_song.setDaemon(True)
        m_runing_song.start()

    else:

        # 如果正在演奏，先停止
        m_runing_song.dostop()
        time.sleep(0.1)
        m_runing_song.join()

        # 重新加载
        m_runing_song = Runing_Song(pin_dic['G18'])
        flag = m_runing_song.file_load(upload_path)
        m_runing_song.setDaemon(True)
        m_runing_song.start()

    return render_template('music.html',message_file_error = "正在演奏"+request.files['file'].filename)
return render_template('music.html')
```