IoTWeb

大四终于接触到直接跟物联网有关的实验啦！所以给这个项目起了 叫 ***IoTWeb***，即基于车联网的远程控制 web 项目。本课程设计用 Arduino 以及相关配件模拟车门开关，用树莓派相机作为远程监控，并且开发了远程操控树莓派拍摄照片，并将照片保存在本地。相应的技术有：python3、Flask Web 服务器、HTML5、Jinja、css美化、FRP 内网穿透、SimpleCV，并利搬瓦工服务器和阿里域名实现了外网访问树莓派。

实验准备

器材	型号
树莓派	Raspberry Pi 1
树莓派相机	Raspberry Pi Camera Rev 1.3
Ardunio	UNO R3
Ardunio 拓展版	IO Expansion Shield for Arduino V7.1
蜂鸣器	(SKU:DFR0032)数字蜂鸣器模块
数字大按钮	(SKU:DFR0029)数字大按钮模块 V2
LED x 2	数字LED发光模块 (SKU: DFR0021)
网线	
SD卡	金士顿16G

实验部署

所需要安装好python3、Ardunio IDE(可以在自己电脑上安装进行 Ardunio 程序烧写)

基本的网络配置和Ardunio IDE的使用可以参考好友凡骐的博客，我们重点来讲讲树莓派是怎么实现视频流以及拍照功能。

Ardunio IDE 烧写程序

我们需要模拟一个车门开关的情况，需要用到数字大按钮和 LED 共同表示车门的状态：

- LED 灯亮，车门未关，蜂鸣器报警
- LED 灯灭，车门已关，蜂鸣器关闭

Ardunio 语言采用 C/C++ 的语法，主要是由两个固定函数构成的：***setup()*** 和 ***loop()*** 。在程序运行的时候先设置函数 setup()，然后再无线循环 loop() 函数。

主要代码如下（详细代码参考Git 仓库/ardunio.txt）：

```
void setup() {
  Serial.begin(9600);           // 打开串口，设置数据通信速率为 9600 bps，这里要与python数值一致
  pinMode(tonepin,OUTPUT);      // 定义蜂鸣器的引脚为输出引脚
  pinMode(ledPin, OUTPUT);      // 定义灯的引脚为输出引脚
  pinMode(ledPin2, OUTPUT);     // 定义灯的引脚为输出引脚
  pinMode(inputPin, INPUT);     // 定义按键引脚为输入引脚
  length=sizeof(tune)/sizeof(tune[0]);   //计算长度
}
void loop(){
  val = digitalRead(inputPin);     //读取输入值
  if (val == HIGH) {               // 检查输入是否为高，这里高为按下
     digitalWrite(ledPin, LOW);    // 灯关闭状态
     digitalWrite(ledPin2, LOW);   // 灯关闭状态
  } else {
     digitalWrite(ledPin, HIGH);   // 灯开启状态
     digitalWrite(ledPin2, HIGH);  // 灯开启状态
for(int x=0;x<length;x++)
  {
    tone(tonepin,tune[x]);
    delay(50*durt[x]);    //这里用来根据节拍调节延时，500这个指数可以自己调整，在该音乐中，我发现用500比较合适。
    noTone(tonepin);
    if (val == HIGH){break;}
  }
  delay(500);
  }
}
```

此程序可以实现：

- 按下开关，小灯熄灭，蜂鸣器关闭
- 不按开关，小灯亮灯，蜂鸣器开启

Flask 视频流

基本原理

流的定义

流是一种让服务器在响应请求时将响应数据分块的技术。流的有点在于以下两点：

- **超级巨大的响应数据。**对于超大的响应数据来说，先把响应数据装载到内存中，再返回给客户端是非常低效的。另一种方法是将响应数据写入到磁盘中，然后用 `flask.send_file()` 将文件返回给客户端，但这样将会增加 I/O 操作。如果响应数据较小，这就是个好得多的方法，因为数据能够按块进行存储。
- **实时数据。**对于某些应用来说，也许需要向某个请求返回来自实时数据源的数据。一个很贴切的例子是实时视频或音频传送。很多安全摄像头用该技术将视频以流的形式发送到服务器。

用 Flask 实现视频流

Flask 是一个微框架（Micro framework），官网上对“微”做了详细解释

> “微”(micro) 并不意味着你要把整个web应用放到一个python文件里（虽然确实可以），也不意味着Flask 在功能上有所欠缺。微框架中的“微”意味着 Flask 旨在保持核心功能的简单而易于扩展。Flask 不会替你做出太多决策，比如使用何种数据库。而那些 Flask 帮你做好的决策（比如使用哪种模板引擎），都是很容易替换。除此之外的一切都由可由你掌握。
>
> 默认情况下，Flask 不包含数据库抽象层、表单验证，或是任何已在其它已库中处理的很好的功能。相反，Flask 支持通过扩展来给应用添加这些功能，如同是 Flask 本身实现的一样。众多的扩展提供了数据库集成、表单验证、上传处理及各种各样的开放认证技术等功能。Flask 也许是“微小”的，但它已准备好在复杂的生产环境中投入使用。

我们可以使用 Flask 框架作为 Web 服务器，通过使用生成器响应流。

构建实时视频流服务器

有很多种流式传输视频到浏览器的方式，每一种方法各有优劣。与 Flask 的流式特性结合得非常好的一种方法是流式输出一系列单独的 JPEG 图片。这被称为 移动的 JPEG（Motion JPEG），这种方法正被一些 IP 安全摄像头使用。这种方法的延迟低，但是质量并不是最好，因为对于移动视频来说，JPEG 的压缩并不高效。

下面你将看到一个特别简单但又十分完善的 web 应用，可以提供移动的 JPEG 流：

```
#!/usr/bin/env python
from flask import Flask, render_template, Response
from camera import Camera

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(gen(Camera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == '__main__':
```

```
    app.run(host='0.0.0.0', debug=True)
```
复制代码

这个应用导入了 `Camera` 类，该类负责提供帧序列。当前情形将摄像头控制部分放在单独的模块中是很好的主意，这样 web 应用就能保持代码的整洁、简单和通用性。

该应用有两个路由。路由 `/` 提供定义在 `index.html` 模版中的主页面。你能从下面的代码中看到模版文件的内容：

```
<html>
……
  <body>
    <h1>实时监控</h1>
    <hr>
    <h3><img src="{{ url_for('video_feed') }}" width="70%"></h3>
    <hr>
    <h3><a href="/index" class="button">返回菜单</a></h3>
    <p id="logInfo"></p>
  </body>
</html>
```

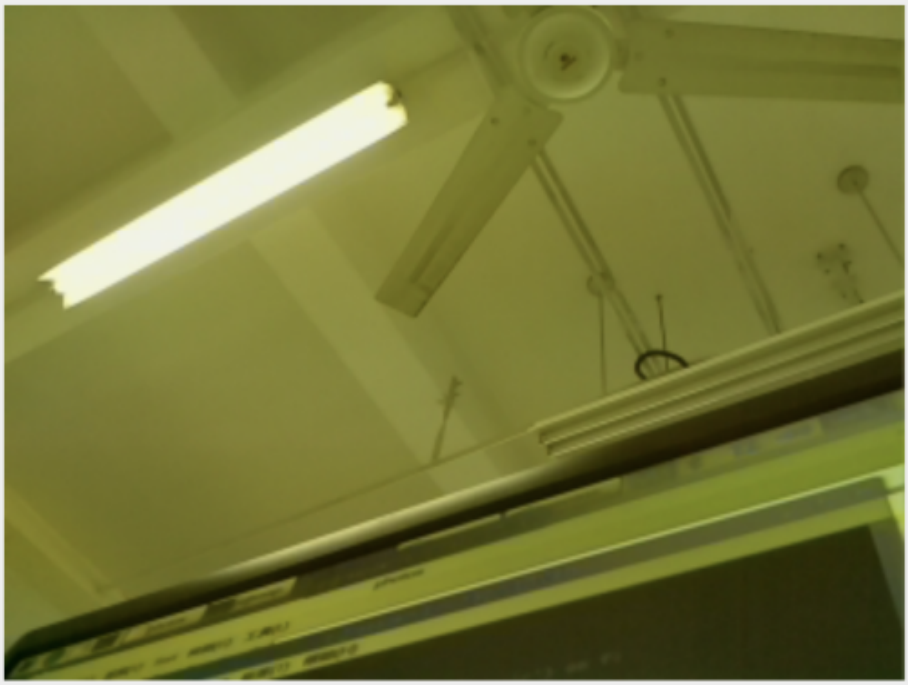
html 具体参考 /camera.html 。在html中最重要的一行代码就是 `<img src="{{ url_for('video_feed') }}" width="70%">` 。路由 `/video_feed` 返回的是流式响应。因为流返回的是可以显示在网页中的图片，到该路由的 URL 就放在图片标签的 `src` 属性中。浏览器会自动显示流中的 JPEG 图片，从而保持更新图片元素，由于分部响应受大多数（甚至所有）浏览器的支持（如果你找到一款浏览器没有这种功能，请务必告诉我）。

在 `/video_feed` 路由中用到的生成器函数叫做 `gen()` ，它接收 `Camera` 类的实例作为参数。 `mimetype` 参数的设置和上面一样，是 `multipart/x-mixed-replace` 类型，边界字符串设置为 `frame` 。

`gen()` 函数进入循环，从而持续地将摄像头中获取的帧数据作为响应块返回。该函数通过调用 `camera.get_frame()` 方法从摄像头中获取一帧数据，然后它将这一帧以内容类型为 `image/jpeg` 的响应块形式产出（yield），如上所述。

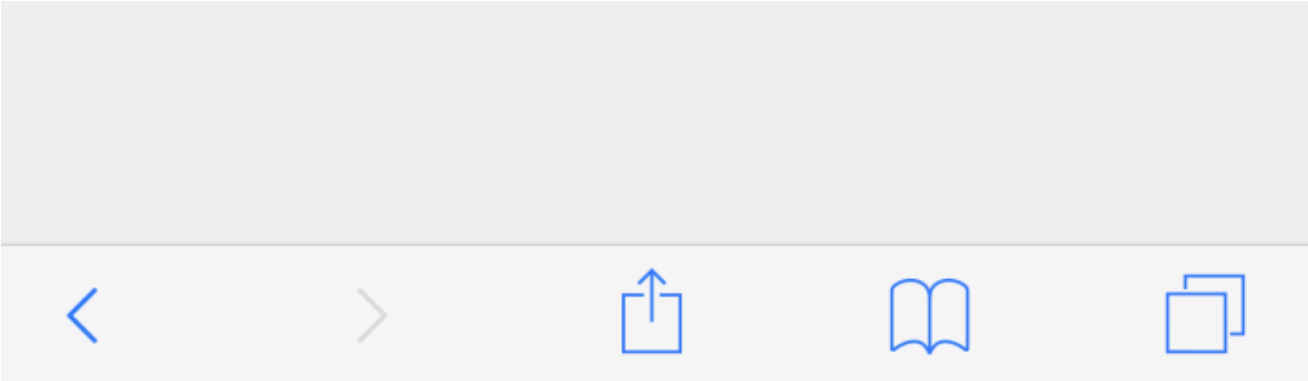
最终实现的效果：

实时监控



返回菜单

2019年01月11日 16:12:44 星期五



到现在我们的直播间就创建成功啦!

用户登录

远程操控直播间是一个很隐私的操作，加上认证模块，会让我们的摄像头更安全。

首先创建一个用户和用户名:

```
SECRET_KEY = 'development key'
USERNAME = 'admin'
PASSWORD = '123456'
```

设置路由响应:

```
@app.route('/', methods=['GET', 'POST'])
def login():
    error = None
    if request.method == 'POST':
        if request.form['username'] != app.config['USERNAME']:
            error = '帐号不存在! '
        elif request.form['password'] != app.config['PASSWORD']:
            error = '密码不匹配! '
        else:
            session['logged_in'] = True
            flash('登录成功! ')
            return redirect(url_for('index'))
    return render_template('login.html', error=error)
```

登录成功即可跳转到 /index.html 界面

远程拍照

我们采用了树莓派自带的相机模块拍摄静态照片工具`raspistill`，基本语法是:

```
raspistill -o cam.jpg  # 拍摄一个名叫cam.jpg的照片，保存到当前目录下
```

想要在 python里运行 debian 的命令，就需要用到`subprocess.call()`这个函数，主要代码如下:

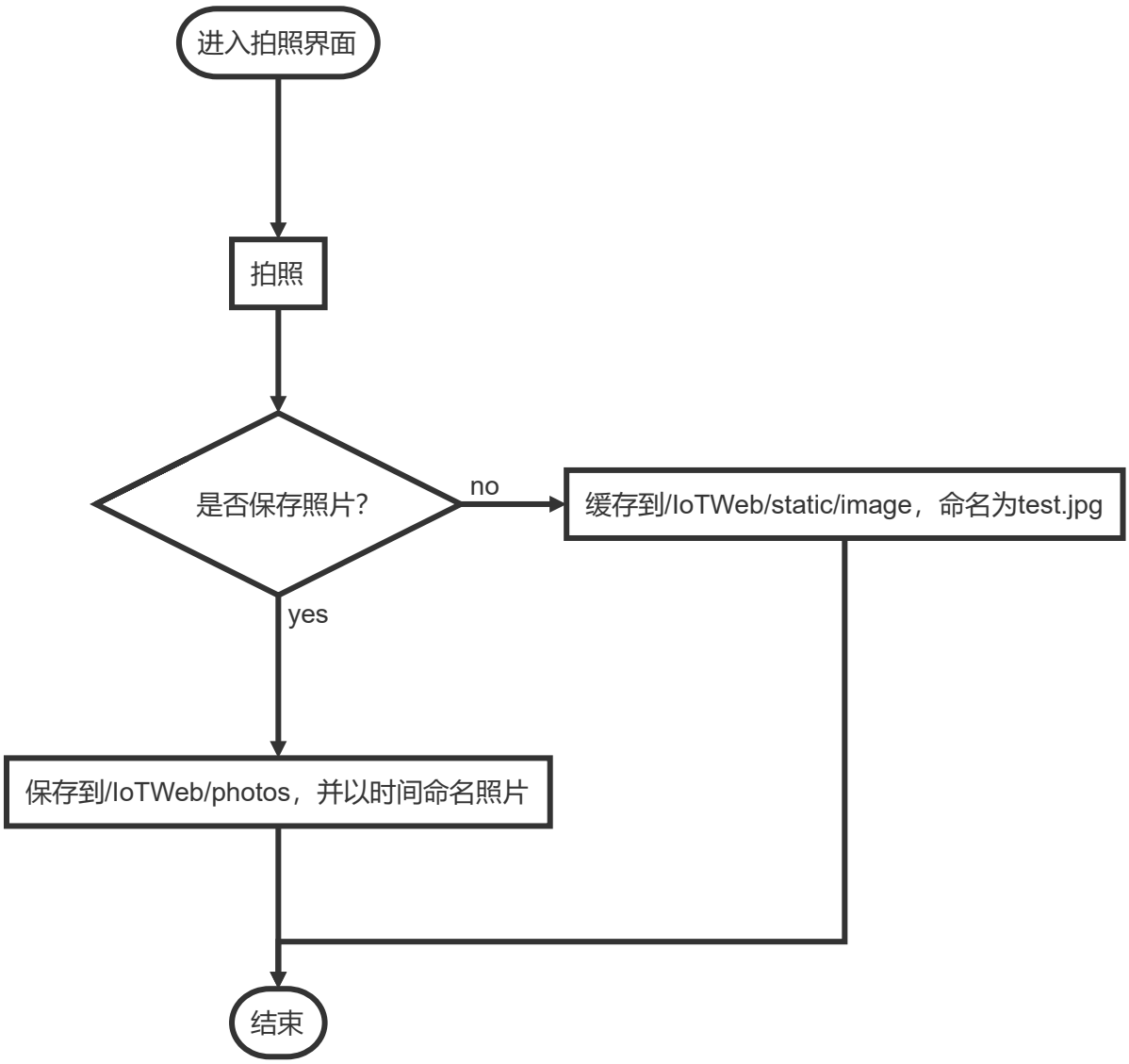
```
import subprocess

@app.route('/photo')
def photo():

    subprocess.call("raspistill -o %s -t 100" %
("/home/pi/flask/IoTWeb/static/image/test.jpg"), shell=True)

    return render_template('view.html', **templateData)
```

照片拍下来存到 image 目录下，如果要保存照片，则将照片存在 photos 里。具体流程参考下图：

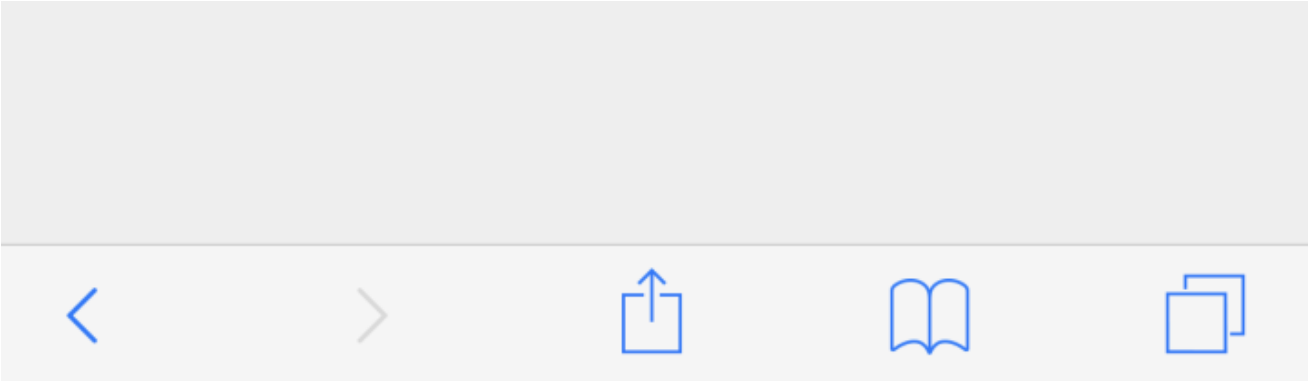


具体实现效果如下：

照片预览

保存 删除

2019年01月11日 16:11:56 星期五



遇到的问题

- 脚本启动缓慢

  原因：由于课设用到的树莓派为1代，内存等各个方面性能不足，项目 import 的包越多，python 脚本启动越慢。

- xshell 无法远程连接树莓派

  原因：未开启 ssh 服务

  解决方案：进入到树莓派界面点击左上角的树莓派，找到【设置】-【远程设置】，开启 ssh 和 VNC connect

- 每次登陆树莓派，发现树莓派的 IP 地址可能会改变，如何设置静态 IP 呢?

  原因：树莓派网络默认开启 DHCP 模式

  解决方案：树莓派采用的是 Debian 操作系统，需要修改 `/etc/dhcpcd.conf` 里添加：

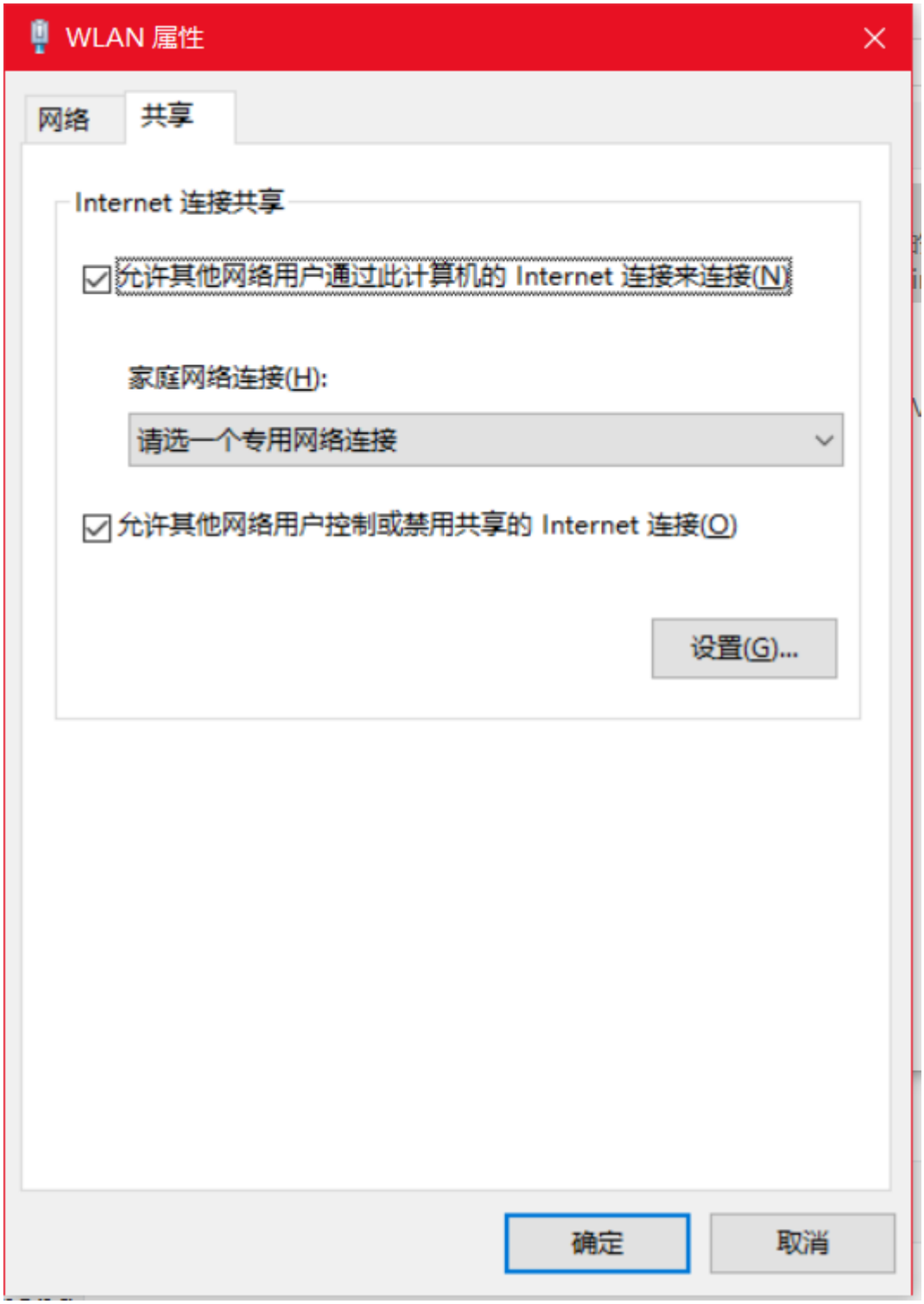
  ```
  vim /etc/dhcpcd.conf
  # 树莓派自带的vi很不好用，通过sudo apt-get install vim

  # 指定接口
  interface eth0
  # 固定IP，别忘了子网掩码
  static ip_address=192.168.137.238/24
  # 设置网关
  static routers=192.168.137.1
  # 手动自定义DNS服务器
  static domain_name_servers=114.114.114.114

  sudo reboot # 修改完重启生效（debian好像没有systemctl restart network命令）
  # 或者采用以下命令
  sudo ifconfig eth0 down
  sudo ifconfig eth0 up
  ```

- `ping www.baidu.com` 域名解析暂时失败

  1. 若连接的是手机热点（win 10系统），右键右下角WiFi【网络和Internet设置】-【网络连接】，右键WLAN属性 - 共享，将 Internet 连接共享以下的两个打勾，然后再 ping 百度。



2. 修改回 DHCP 模式，reboot 重启获取新的网络 IP

- 安装好 SimpleCV 之后会出现两个错误：

```
lsof: status error on /dev/video*: No such file or directory
WARNING: caught exception: SystemError("Cannot identify '/dev/video0': 2, No such file or directory",)
WARNING: SimpleCV can't seem to find a camera on your system, or the drivers do not work with SimpleCV.
```

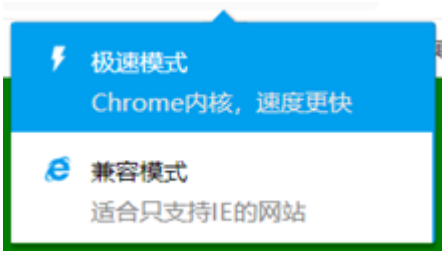
一个原因是因为 lsof 未安装，直接 `sudo apt-get install lsof` 即可

另一个原因是 SimpleCV 找不到相机，需要在`/etc/modules`添加模块：

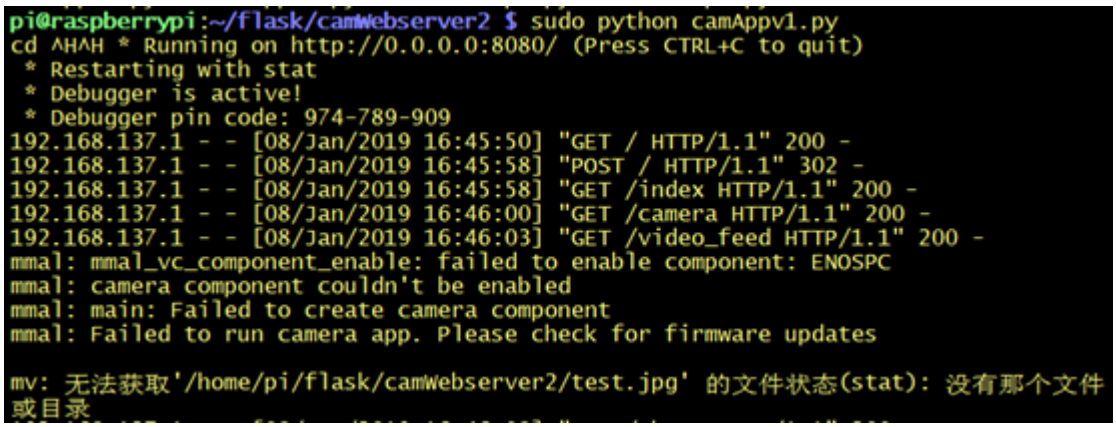
```
vim /etc/modules
bcm2835-v4l2
```

- Flask 视屏流做好之后，发现 QQ 浏览器上没有出现实时画面：

  有些浏览器没有开启***内核***模式，导致页面显示不出来



- 实现了 Flask 视频流之后，想要实现远程拍照的功能，要利用到摄像头。如果视屏流占用摄像头，会报以下错误：



  mmal 是因为摄像头被占用了，需要等待视屏流关闭摄像头才可以启动拍照功能。

参考文档

Arduino 官网

购买相应器材→DF 创客社区

Linux 开源社区|树莓派

Arduino 配件查询