树莓派服务器搭建

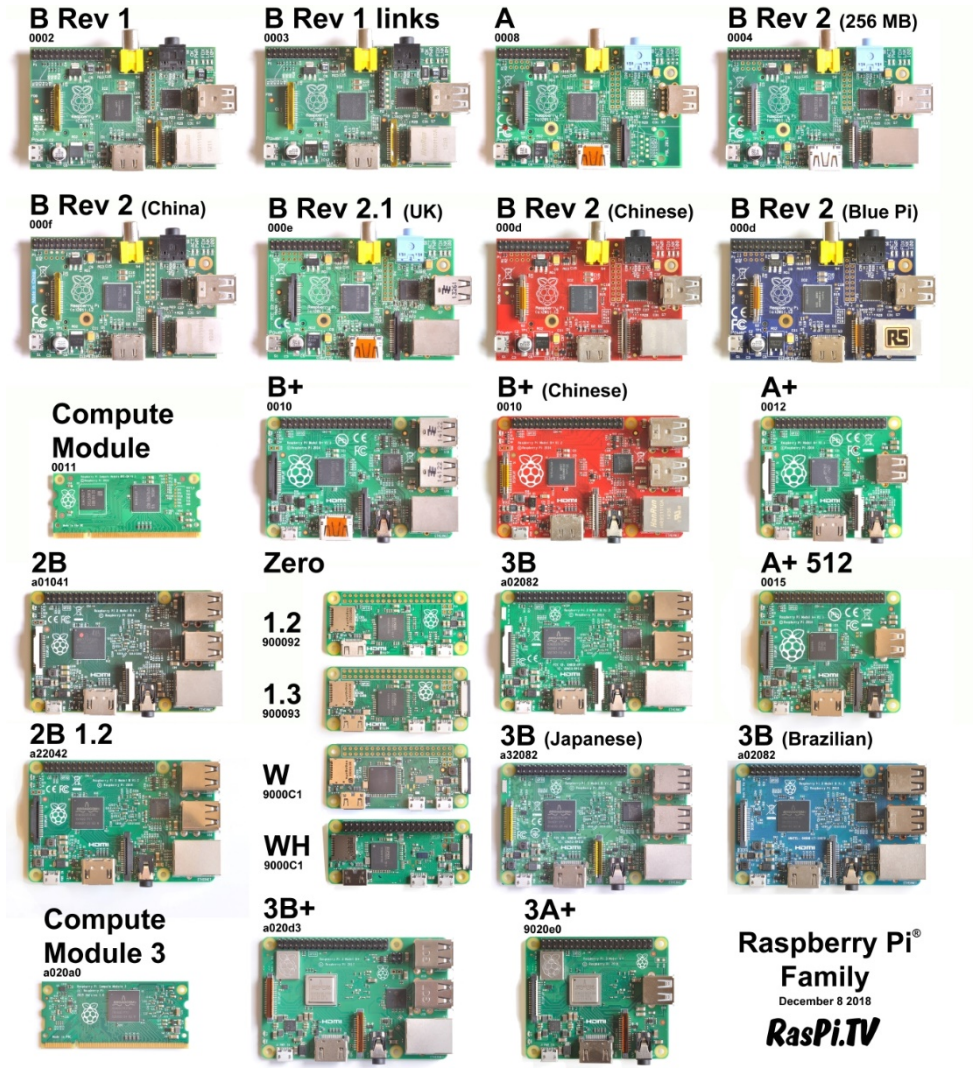
主讲人：曹家英

第一部分：树莓派

- 1. 树莓派介绍
- 2. 树莓派环境搭建
- 3. 树莓派与命令行
- 4. 相关知识扩展
- 5. 树莓派基础外设操作
- 6. 树莓派服务器

课程概览

- Web Server搭建
- 实战12——Wordpress
- Python Flask Web Server
- MQTT简单物联网实现
- NFS Server网络文件系统搭建



Web Server搭建

WEB SERVER

简单介绍：

1. 什么是HTML？
2. HTTP SERVER做什么用途？
3. 常见的HTTP Server，APACHE，NGINX，LIGHTTPD

LAMP环境中A就是APACHE，使用最多就是APACHE。

server	Apache	Nginx	Lighttpd
Proxy代理	非常好	非常好	一般
Rewriter	好	非常好	一般
Fcgi	不好	好	非常好
热部署	不支持	支持	不支持
系统压力比较	很大	很小	比较小
稳定性	好	非常好	不好
安全性	好	一般	一般
技术支持	非常好	很少	一般
静态文件处理	一般	非常好	好
Vhosts虚拟主机	支持	不支持	支持
反向代理	一般	非常好	一般
Session sticky	支持	不支持	不支持

WEB SERVER 搭建

我们看见LAMP这一套WEB SERVER工具使用非常广泛，对于开发者来说，并不是越广泛越好。合适才是最关键的。

树莓派可以安装这个 LAMP 系列，但 Apache 和 MySql 对于树莓派的性能来说，太重了。

这里推荐在树莓派上使用nginx + php + sqlite这样的组合会比较方便使用。

Web Servers

Most popular web servers

© W3Techs.com	usage	change since 1 December 2018
1. Apache	44.4%	-0.4%
2. Nginx	41.0%	+0.6%
3. Microsoft-IIS	8.9%	-0.2%
4. LiteSpeed	3.8%	+0.2%
5. Google Servers	0.9%	

percentages of sites

数据来自：https://w3techs.com/

WEB SERVER 软件安装

安装Nginx，php，及sqlite软件：

sudo apt-get install nginx php-fpm php-sqlite3

软件的简单说明：

PHP：PHP是一种开源的通用计算机脚本语言，尤其适用于网络开发并可嵌入HTML中使用。

nginx ：会安装相关的web server及服务。

php-fpm：FastCGI 进程管理器，有一个守护进程，实现快速响应。

php-sqlite3：对 SQLite v3 数据库的支持

WEB SERVER NGINX

1. 开启服务：
   sudo /etc/init.d/nginx start

```
[  9:20PM ]  [ pi@raspberrypi:~ ]
 $ sudo /etc/init.d/nginx start
[ ok ] Starting nginx (via systemctl): nginx.service.
[  9:30PM ]  [ pi@raspberrypi:~ ]
 $
```

2. 通过浏览器访问树莓派的IP地址（注意使用的网卡）

```
[  9:30PM ]  [ pi@raspberrypi:~ ]
 $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether b8:27:eb:b8:6d:d1  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 9  bytes 524 (524.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9  bytes 524 (524.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.30  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::fcac:109b:3ef0:631c  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:ed:38:84  txqueuelen 1000  (Ethernet)
        RX packets 1680  bytes 223224 (217.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 740  bytes 130067 (127.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

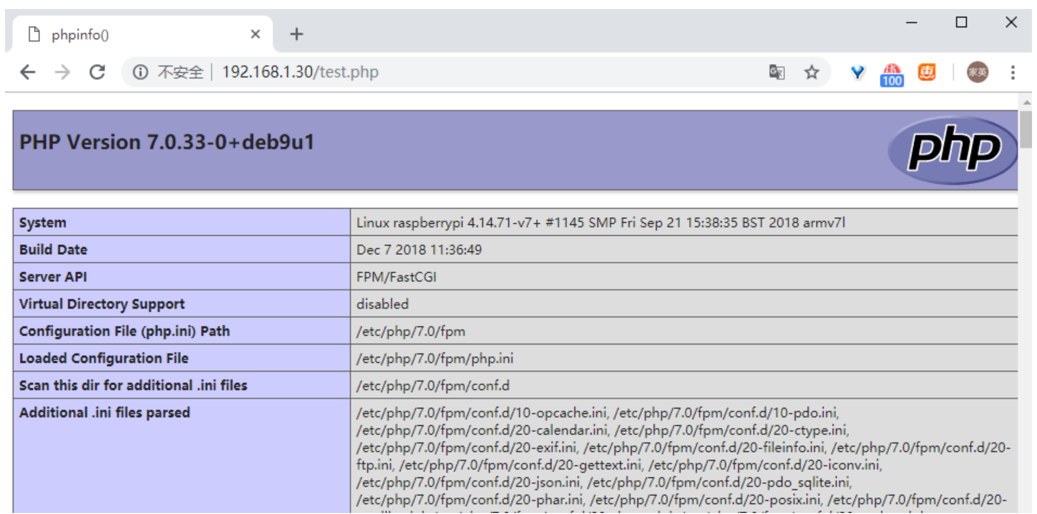
3. 访问后会发现我们打开了个一个网页：

这样就可以说明Nginx Web Server已经安装完成。

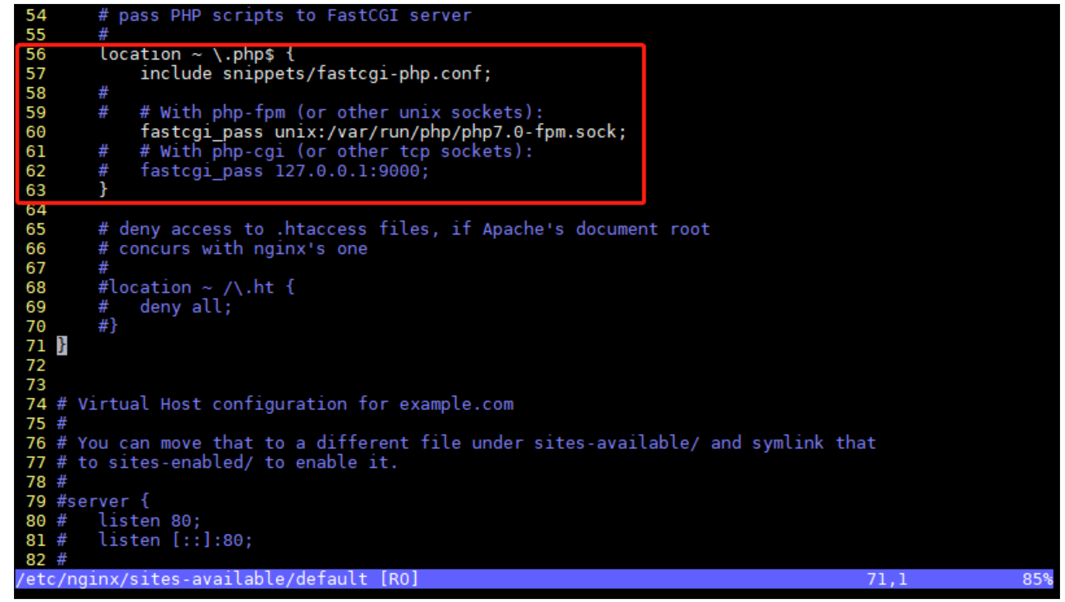
WEB SERVER PHP

1. 进入/var/www/html目录，创建一个test.php的文件：
   输入<?php phpinfo(); ?>

2. 通过网页访问：ip/test.php,如果出现如下界面，既安装成功



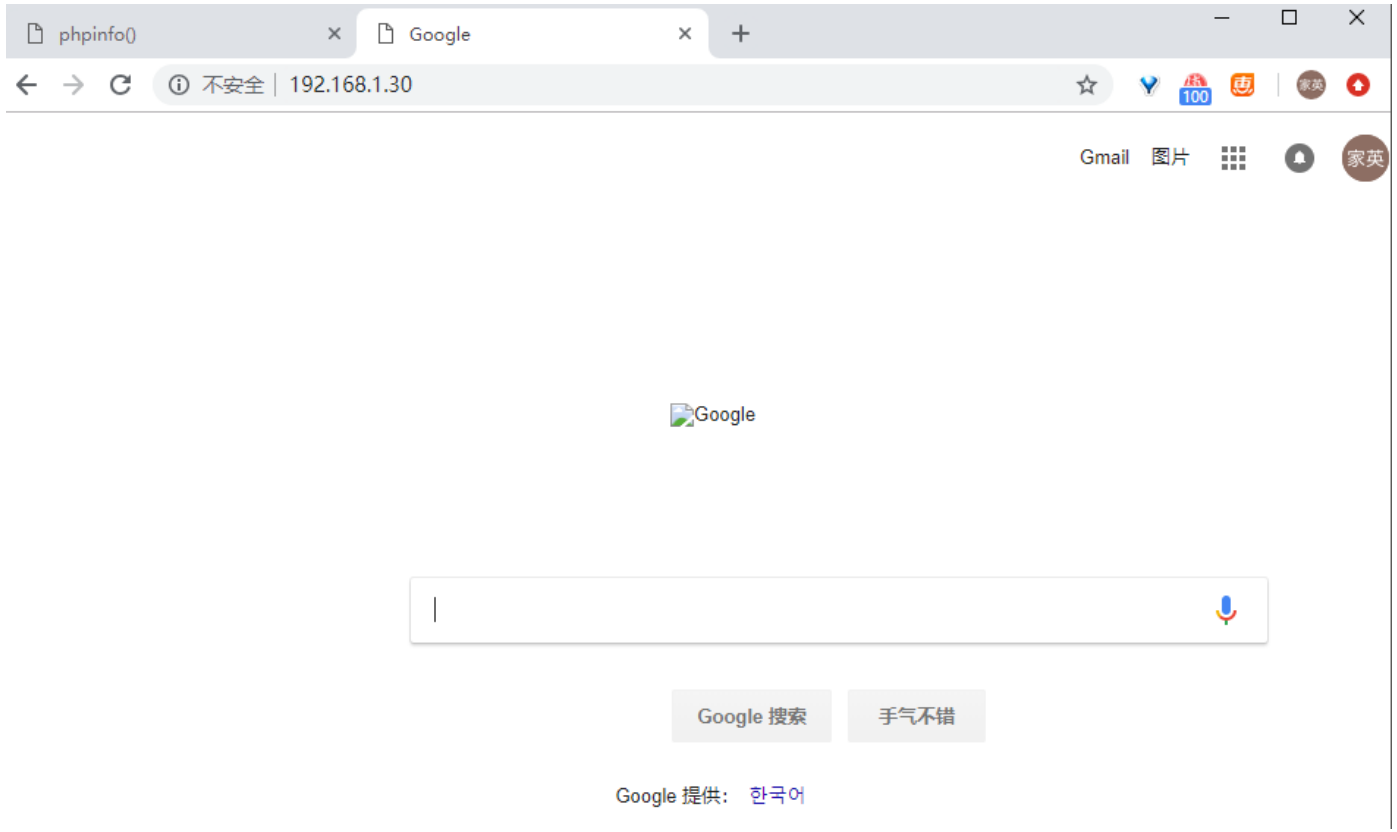
3. 打开NGINX的配置文件：
   sudo vi /etc/nginx/sites-available/default



4. 尝试打开test.php文件,确定php运行正常。
   php /var/www/html/test.php

WEB SERVER PHP

这样我们就可是使用自己的HTML页面的替换
/var/www/html/index.nginx-debian.html



实战12——Wordpress

Wordpress

WordPress：是使用PHP语言开发的博客平台，用户可以在支持PHP和MySQL数据库的服务器上架设属于自己的网站。

也可以把 WordPress当作一个内容管理系统（CMS）来使用。

Content Management Systems

Most popular content management systems

© W3Techs.com	usage	change since 1 December 2018	market share	change since 1 December 2018
1. WordPress	33.0%	+0.6%	60.0%	+0.5%
2. Joomla	3.0%		5.4%	-0.1%
3. Drupal	1.9%		3.5%	-0.1%
4. Squarespace	1.5%	+0.1%	2.7%	+0.1%
5. Shopify	1.5%	+0.1%	2.7%	+0.2%

percentages of sites

Fastest growing content management systems since 1 December 2018

相关资料下载：

wordpress获取地址：https://cn.wordpress.org/download/

SQLITE扩展插件地址：https://wordpress.org/plugins/sqlite-integration/

Wordpress 安装与搭建

1. 通过wget下载wordpress sourcecode的压缩包。

2. 将下后的wordpress的压缩文件解压到/var/www/html/目录下

```
[ 10:39PM ]  [ pi@raspberrypi:/var/www/html ]
 $ ls
index.nginx-debian.html      test.php   wordpress-5.0.3-zh_CN.tar.gz
index.nginx-debian.html.bak  wordpress
[ 10:39PM ]  [ pi@raspberrypi:/var/www/html ]
 $
```

3. 可以给wordpress降低权限，或者修改所属用户。

   sudo chmod –R 777 wordpress

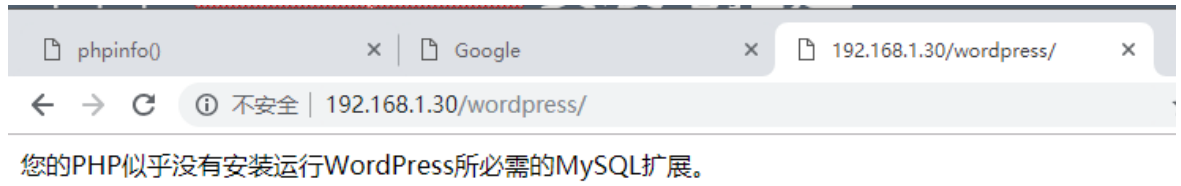
   sudo chown -R www-data:www-data wordpress

```
[ 10:44PM ]  [ pi@raspberrypi:/var/www/html ]
 $ ll
total 11M
-rw-r--r-- 1 pi   pi   243K Jan 30 21:59 index.nginx-debian.html
-rwxrwxrwx 1 root root  781 Jan 29 22:56 index.nginx-debian.html.bak
-rwxrwxrwx 1 root root   20 Jan 30 01:45 test.php
drwxrwxrwx 5 pi   pi   4.0K Jan 30 22:19 wordpress
-rw-r--r-- 1 pi   pi    11M Jan 11 18:02 wordpress-5.0.3-zh_CN.tar.gz
```

4. 修改NGINX的index文件，增加检查index.php。

```
23      listen [::]:80 default_server;
24
25      # SSL configuration
26      #
27      # listen 443 ssl default_server;
28      # listen [::]:443 ssl default_server;
29      #
30      # Note: You should disable gzip for SSL traffic.
31      # See: https://bugs.debian.org/773332
32      #
33      # Read up on ssl_ciphers to ensure a secure configuration.
34      # See: https://bugs.debian.org/765782
35      #
36      # Self signed certs generated by the ssl-cert package
37      # Don't use them in a production server!
38      #
39      # include snippets/snakeoil.conf;
40
41      root /var/www/html;
42
43      # Add index.php to the list if you are using PHP
44      index index.html index.htm index.nginx-debian.html index.php;
45
46      server_name _;
47
48      location / {
49          # First attempt to serve request as file, then
50          # as directory, then fall back to displaying a 404.
51          try_files $uri $uri/ =404;
/etc/nginx/sites-available/default [RO]                    44,1-4        35%
```

5. 尝在浏览器试访问下ip/wordpress/



Wordpress安装SQLite插件及配置

wordpress中SQLITE插件安装方法：

1. 将得到的sqlite-integration解压至
   <PATH>/ www/wordpress/wp-content/plugins/目录下。

2. 再将sqlite-integration文件夹下的db.php复制到
   <PATH>/ www/wordpress/wp-content/目录下。

复制配置文件：

sudo cp wp-config-sample.php wp-config.php
sudo vi wp-config.php
在最后加入如下语句，使用中文：
define('WPLANG', 'zh_CN');

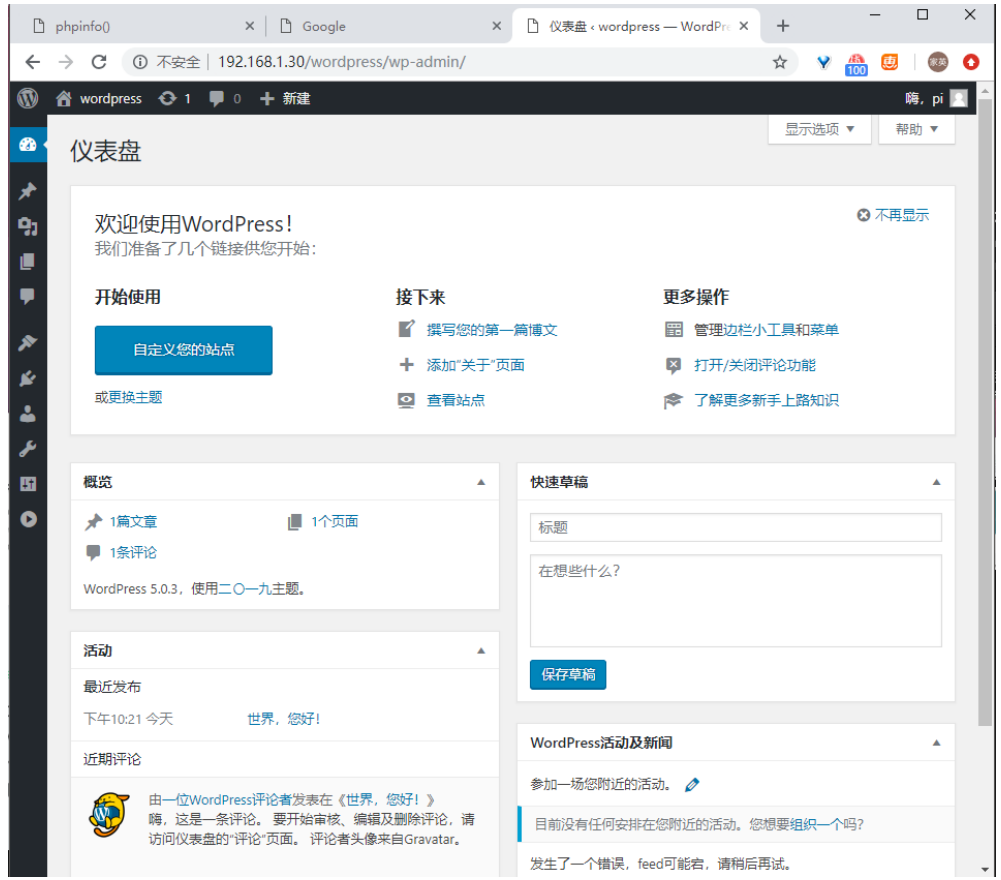
安装其他插件：

比如：Disable Google Fonts
地址：https://cn.wordpress.org/plugins/disable-google-fonts/
解压至<PATH>/www/wordpress/wp-content/plugins/
在后台启用插件，如右图。

进行安装后就可以使用wordpress

完成了上面的SQLite的支持后就可以访问如下网站进行安装：
ip/wordpress/wp-admin/install.php
后台地址： ip/wordpress/wp-admin

安装插件：Disable google fonts
在此之后就可以自己的网页上为所欲为了。



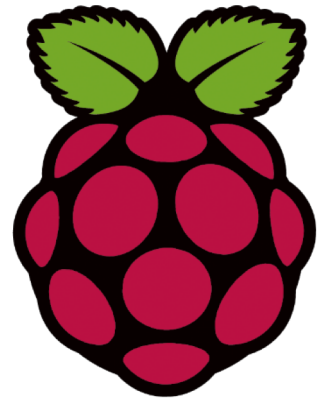
Python Flask Web Server

Python Flask 简单介绍

什么是Flask？

简而言之，Flask是一个使用Python编写的轻量级Web应用框架。用来支持动态网站、网络应用程序及网络服务的开发。

Flask基于WSGI（Python Web Server Gateway Interface）和Jinja2 模板引擎，使用BSD授权。



为什么用Flask？

相对比来说Flask的学习曲线比较平缓，不像Django覆盖面比较广，新手入门略有难度。

但这里并不是Django不好，相反Django也是非常优秀的Web应用框架，只是Django和Flask的设计理念相反，Django是完整的解决方案，相比较下，Flask只给出了核心功能，可以自由扩展。相当于开箱即用，容易上手。

Python Flask 准备工作

安装flask：

一般来说树莓派自己的带有flask的，版本可能老一些，也不排除使用Lite版本的Rasbian，还是重新安装一下：

pip3 install flask

安装后我们可以通过在终端中输入，python3进入Py Shell

>>> import flask

>>> flask.__version__

来查看当前的版本号。

```
>>> import flask
>>> flask.__version__
'0.12.1'
>>> print flask.__version__
0.12.1
>>> exit()
```

在有需求的情况下可以使用虚拟环境进行搭建：

安装python venv：

sudo apt-get install python3-dev python3-venv

创建虚拟环境：

python3 -m venv test

使能虚拟环境：

source test/bin/activate

关闭虚拟环境：

deactivate

更新基本工具：

pip install --upgrade pip setuptools wheel

```
pi@raspberrypi:~ $ pip3 install flask
Collecting flask
  Using cached https://files.pythonhoste
386634607af824ea997202cd0edb4b/Flask-1.0
Collecting Werkzeug>=0.14 (from flask)
  Downloading https://files.pythonhosted
```

```
pi@raspberrypi:~ $ python3
Python 3.5.3 (default, Jan 19 2017, 14:1
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "
>>> import flask
>>> flask.__version__
'1.0.2'
>>> exit()
```

Flask Hello World !

第一个Flask程序：

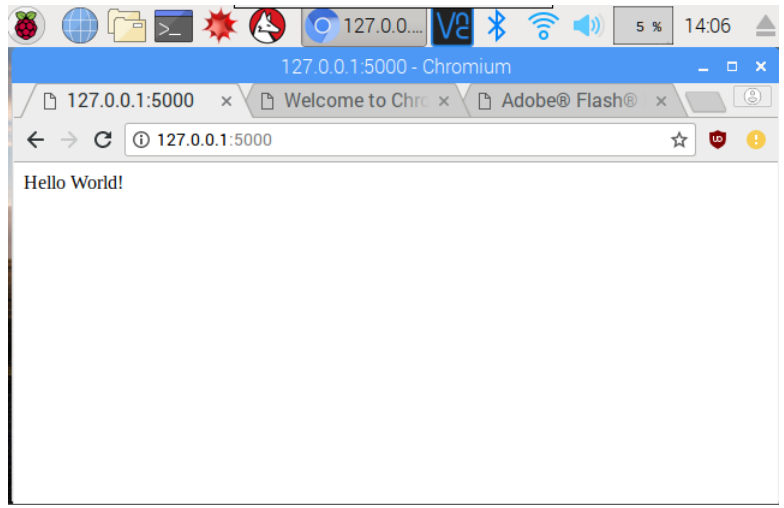
代码如下：

app = Flask（__name__）
传递__name__进入Flask，方便操作。

@app.router(“/”)
def hello_world():
return ‘Hello World!’
@为装饰器，表明url 为 “/” 的情况下执行hello_world()函数

app.run()
开始运行，内部死循环用来监听5000 port。

```python

from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world() :
    return 'Hello World!'

def main():
    app.run()

if(__name__ == __main__) :
    main()
```



Flask 运行参数

app.run() 可以加入参数：

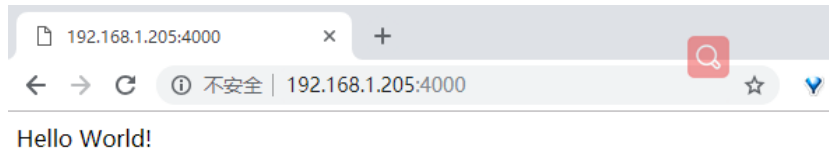
debug=True　　# 方便调试。
host=　‘0.0.0.0’　　# 设置访问地址范围
port=4000　　# 监听端口

加入配置文件：

新建一个，输入DEBUG = True
在flask程序中，使config.py用下面函数引入设定。
app.config.from_pyfile(‘config.py’)

app.config：实际就是一个字典。

```
1
2 from flask import Flask
3
4 app = Flask(__name__)
5 app.config.from_pyfile('config.py')
6
7 @app.route("/")
8 def hello_world() :
9     return 'Hello World!'
10
11 def main():
12     app.run(host='0.0.0.0',port=4000,debug=True)
13
14 if( __name__ == '__main__' ) :
15     main()
```



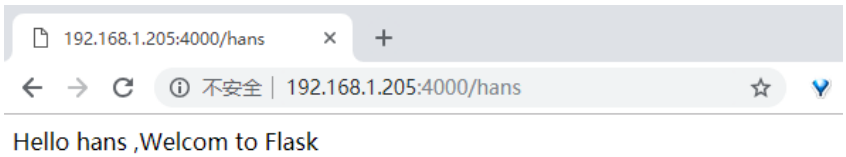
Flask url 传参

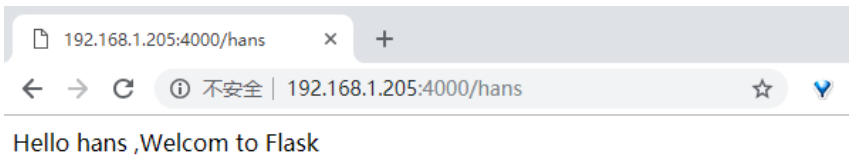
使用URL进行传递参数：

在装饰器中调整URL的格式，进行参数的传入。

同样的我们可以在不同的装饰函数里进行更多的操作比如控制一个灯。

```python

from flask import Flask

app = Flask(__name__)
app.config.from_pyfile('config.py')

@app.route('/')
def hello_world() :
    return 'Hello World!'

@app.route('/<user>')
def user(user):
    return 'Hello %s ,Welcom to Flask' %user

def main():
    app.run(host='0.0.0.0',port=4000,debug=True)

if( __name__ == '__main__' ) :
    main()

```





Flask 使用HTML模板

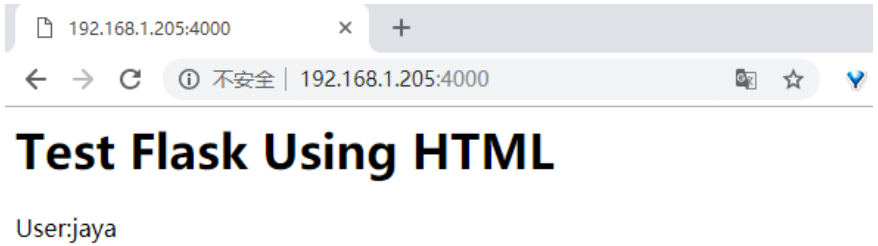
调用HTML模板：

加载render_template模块，用来加载HTML。

HTML文件放在templates 目录下，render_template 模块进行加载时会去templates目录下找相对应的HTML文件。

```
(test) pi@raspberrypi:~/flask $ tree
.
├── 1.py
├── 2.py
├── config.py
├── static
└── templates
    └── index.html
```

在HTML模板中，“{{}}”这样的表记Flask会处理，将相应的参数，或者方法执行进入。如右图方法，将user参数传入HTML中。



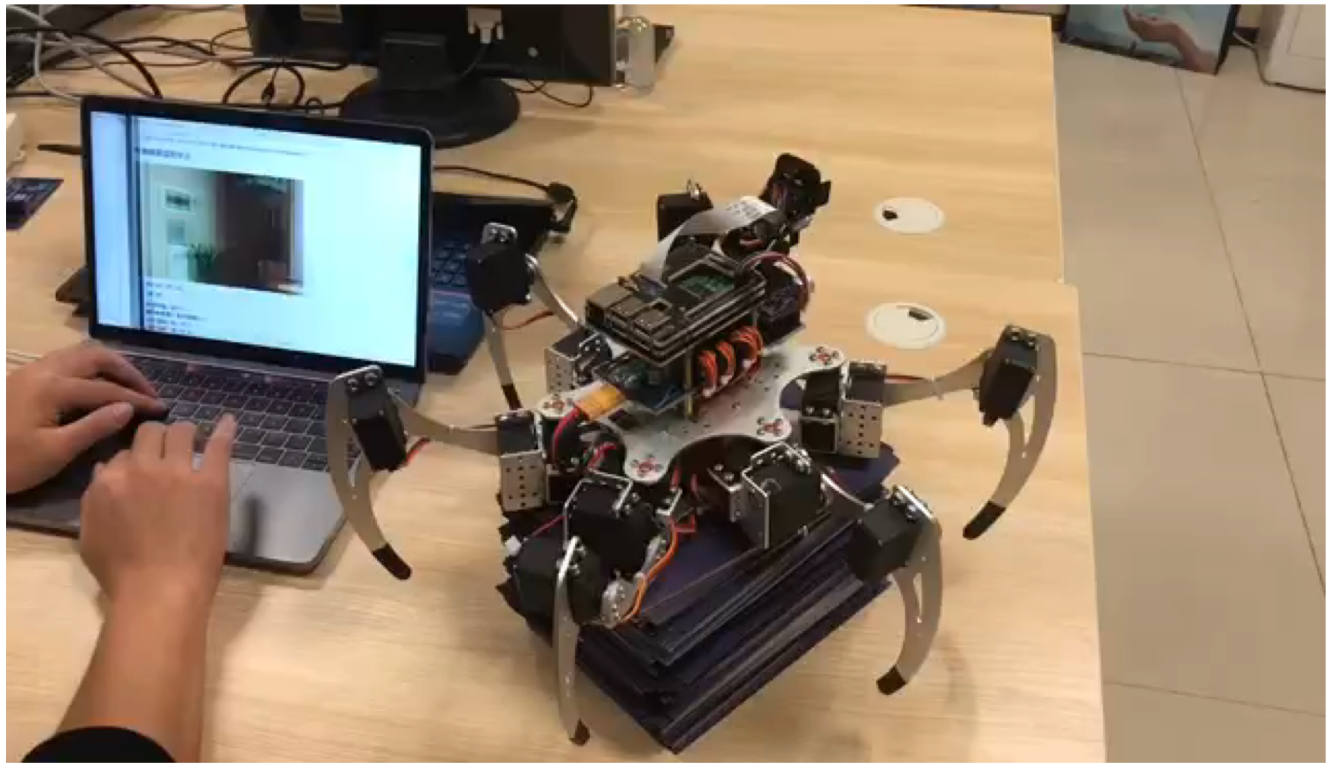

```python
from flask import Flask,render_template

app = Flask(__name__)
app.config.from_pyfile('config.py')

@app.route('/')
def hello_world() :
    return render_template('index.html',user='jaya')

@app.route('/<user>')
def user(user):
    return 'Hello %s ,Welcom to Flask' %user

def main():
    app.run(host='0.0.0.0',port=4000,debug=True)

if( __name__ == '__main__' ) :
    main()
```

```html
<!DOCTYEP html>
<html lang = "en">
<head>
        <meta charset="UTF-8">
        <title></title>
</head>
<body>
        <H1>Test Flask Using HTML</H1>
        <p>User:{{user}}</p>
</body>
</html>
```

放在后面

URL点灯：

```
1
2  from flask import Flask
3  import RPi.GPIO as GPIO
4
5  app = Flask(__name__)
6  app.config.from_pyfile('config.py')
7  GPIO.setwarnings(False)
8  GPIO.setmode(GPIO.BCM)
9  GPIO.setup(17,GPIO.OUT)
10 GPIO.output(17,GPIO.HIGH)
11
12 @app.route('/')
13 def hello_world() :
14     return 'Hello World!'
15
16 @app.route('/<user>')
17 def user(user):
18     return 'Hello %s ,Welcom to Flask' %user
19
20 @app.route('/off')
21 def led_off():
22     GPIO.output(17,GPIO.HIGH)
23     return 'The red LED is OFF'
24
25 @app.route('/on')
26 def led_on():
27     GPIO.output(17,GPIO.LOW)
28     return 'The red LED is ON'
29
30 def main():
31     app.run(host='0.0.0.0',port=4000,debug=True)
32
33 if( __name__ == '__main__' ) :
34     main()
```



MQTT简单物联网实现

MQTT 简单介绍

MQTT是什么：

消息队列遥测传输，工作在 TCP/IP协议上。

目前经常被使用的IoT协议，用于传输少量数据，和命令。

对比HTTP的优点：

低协议开销，它的每消息标题可以短至 2 个字节。

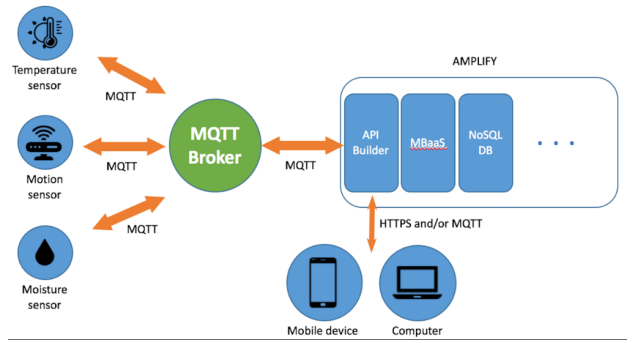
对不稳定网络的容忍，能够从断开等故障中恢复。

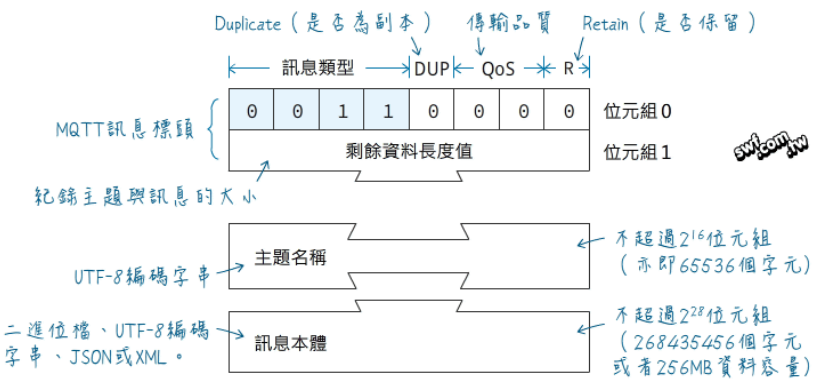
低功耗，MQTT 是专门针对低功耗目标而设计的。

推送通知，用HTTP维持长时间连接的成本很高。

数百万个连接的客户端，数百万这个量级对于HTTP来说可以实现，但不像MQTT这样简单。

客户端平台差异，由于结构简单，方便移植，可以在多重平台架构上运行。





MQTT 架构

MQTT架构：

在一个MQTT系统中存在3种角色：

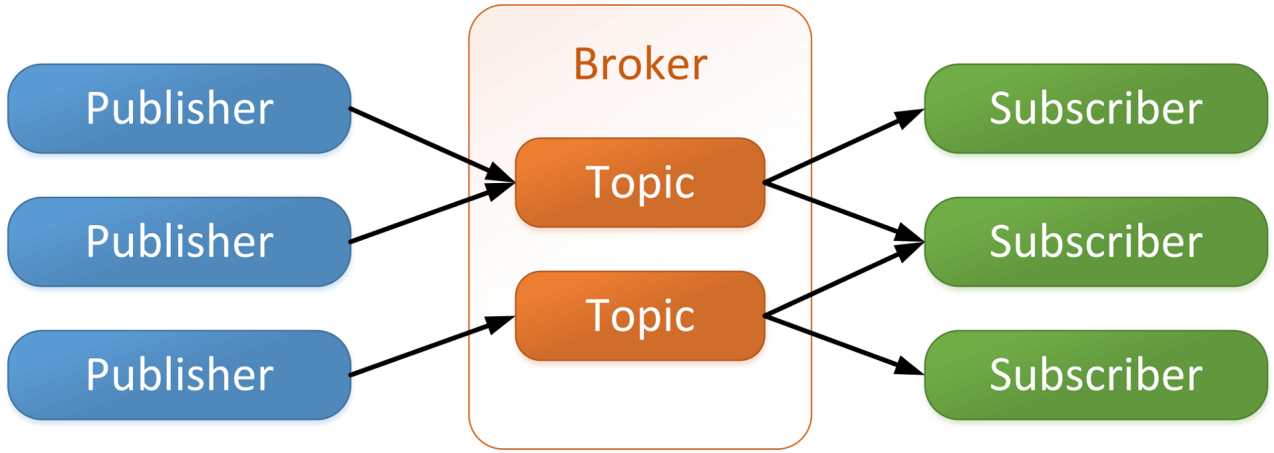
Publisher：发布者

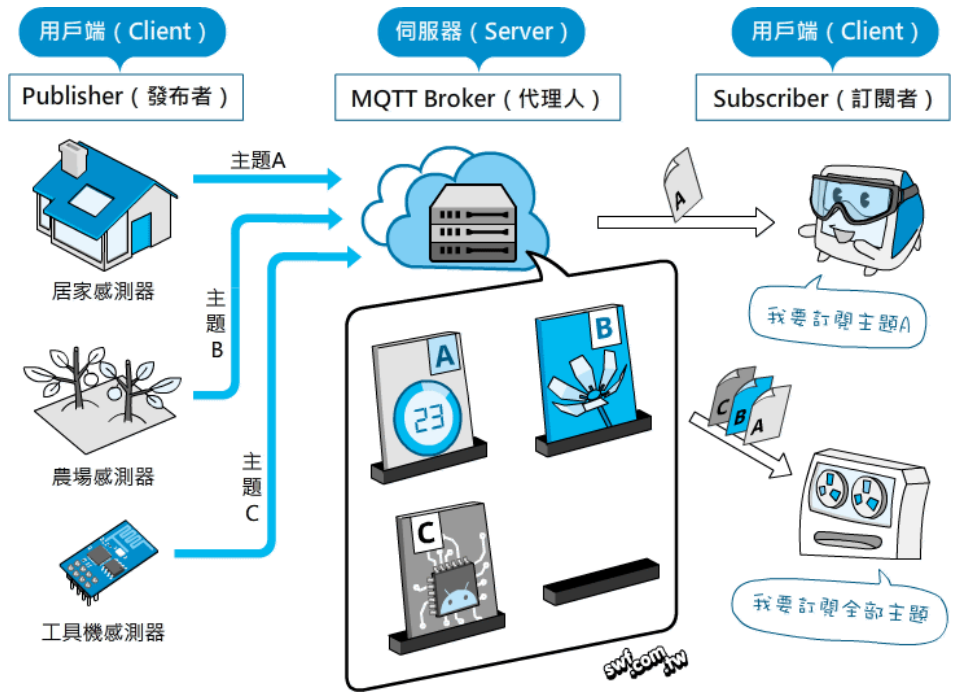
Broker：网关，代理

Subscriber：订阅者

联系途径：

Topic：主题





MQTT 简单实战

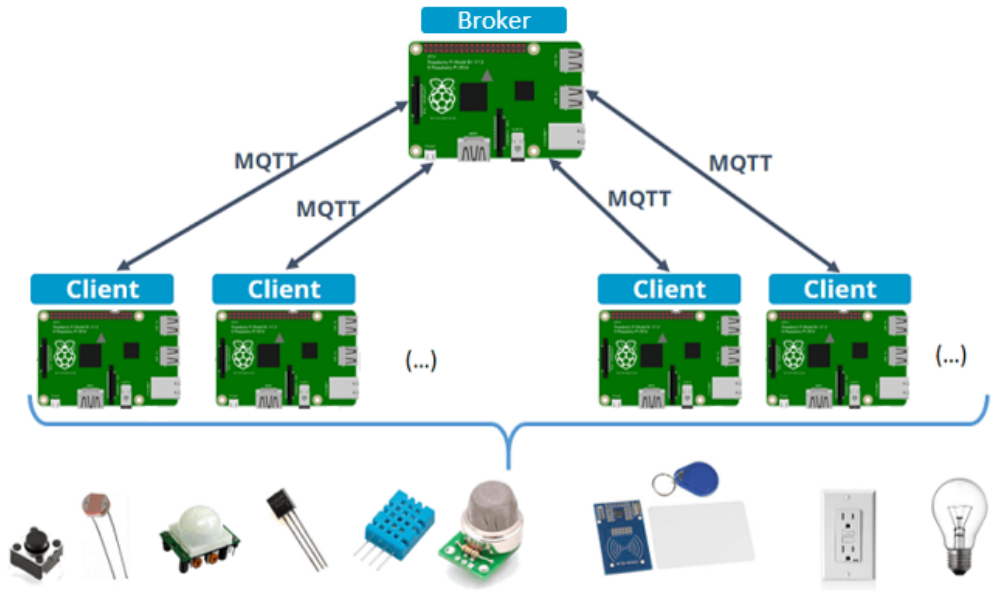
MQTT简单实战：

使用树莓派实现MQTT物联网功能。

树莓派1号： 实现Broker，代理功能

树莓派2号： 实现Publisher，发布者功能

使用MAC or NootBook or手机作为订阅者



MQTT Broker Using RPi

这里是我们的1号树莓派：

安装用于MQTT Broker的相关的软件

sudo apt-get install mosquito

sudo apt-get install mosquitto-clients

查看Mosquitto代理服务状态：

service mosquitto status

测试当前Mosquitto代理时候可用：

mosquitto_sub -t broker_test &

mosquitto_pub -t broker_test -m "Hello, world"

完整的发布者：

mosquitto_pub -h 192.168.1.21 -t gpio -m

“Hello“

PS：-h：broker地址，-t：主题，-m：消息


```
[ 10:24PM ]  [ pi@raspberrypi:~ ]
 $ service mosquitto status
● mosquitto.service - LSB: mosquitto MQTT v3.1 message broker
   Loaded: loaded (/etc/init.d/mosquitto; generated; vendor preset: enabled)
   Active: active (running) since Wed 2019-03-27 22:11:28 CST; 13min ago
     Docs: man:systemd-sysv-generator(8)
   CGroup: /system.slice/mosquitto.service
           └─1766 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

Mar 27 22:11:27 raspberrypi systemd[1]: Starting LSB: mosquitto MQTT v3.1 message broker...
Mar 27 22:11:28 raspberrypi mosquitto[1760]: Starting network daemon:: mosquitto.
Mar 27 22:11:28 raspberrypi systemd[1]: Started LSB: mosquitto MQTT v3.1 message broker.
```

```
[ 10:28PM ]  [ pi@raspberrypi:~ ]
 $ mosquitto_pub -t broker_test -m "Hello, world"
[ 10:28PM ]  [ pi@raspberrypi:~ ]
 $ mosquitto_sub -t broker_test &
[1] 23657
[ 10:28PM ]  [ pi@raspberrypi:~ ]
 $ mosquitto_pub -t broker_test -m "Hello, world"
Hello, world
[ 10:28PM ]  [ pi@raspberrypi:~ ]
 $ mosquitto_pub -t broker_test -m "Hello, world"
Hello, world
```

一号树莓派完成使命

MQTT Publisher

这里是我们的2号树莓派：

2号树莓派是发布者，我们希望发布者可以灵活的控制，Python将是不错的选择。

切换虚拟环境（可以没有心理负担尽情折腾）：

source pyenv/test/bin/activate

安装Python paho-mqtt 库：

pip install paho-mqtt

调用paho.mqtt中的client库：

import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc)：

client：是调用回调的客户端实例

userdata：是任何类型的用户数据

flags：是一个包含Broker响应参数的字典

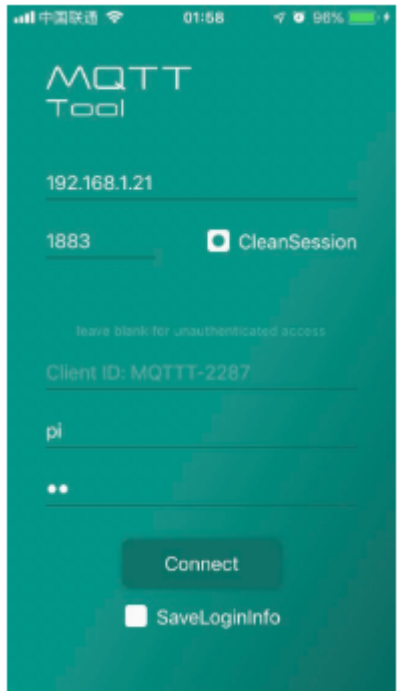
rc：用于判断是否连接成功（0连接成功）

```python
import paho.mqtt.client as mqtt
import LED


led = LED.LED(17)
led.LED_Off()


def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    client.subscribe("led")

def on_message(client, userdata, msg):
    print(msg.topic + " : " + str(msg.payload))
    if(str(msg.payload) == "b'On'") :
        led.LED_On()
    elif(str(msg.payload) == "b'Off'") :
        led.LED_Off()
    else :
        print(str(msg.payload) +  " \r\n msg is false!")

def main():
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    try:
        client.connect("192.168.1.84", 1883, 60)
        client.loop_forever()
    except KeyboardInterrupt:
        client.disconnect()

if __name__ == '__main__':
    main()
```

最后实验

paho.mqtt中的client库主要包括：

connect() / connect_async() ：连接Broker
loop() ：保持与Broker网络连接
loop_start() ：调用一个loop()进程
loop_forever() ：保持 loop()调用
subscribe() ：订阅主题并接收消息
disconnect() ：与Broker断开连接



1号树莓派订阅主题led：
mosquitto_sub -t led &

2号树莓派执行mqtt测试程序：
python test_mqtt.py

1号树莓派向2号树莓派传送信息：
mosquitto_pub -t led -m On
mosquitto_pub -t led -m Off

```
(test) pi@raspberrypi:~ $ python test_mqtt.py
Connected with result code 0
led : b''
b''
 msg is false!
led : b'On'
led : b'Off'
led : b'On'
led : b'On'
led : b'On'
led : b'Off'
led : b'On'
```

NFS Server 网络文件系统

Samba 服务搭建

Samba，是种用来让UNIX系列的操作系统与微软Windows操作系统的SMB/CIFS（Server Message Block/Common Internet File System）网络协议做链接的自由软件。第三版不仅可访问及分享SMB的文件夹及打印机，本身还可以集成在Windows Server的网域，与Windows的兼容性很好。

安装Samba服务软件：

```
sudo apt-get install samba
```

查看运行状态：

```
sudo /etc/init.d/samba status
```

添加配置文件：

```
sudo vi /etc/samba/smb.conf
```

```
256 [share]
257         path = /home/pi
258         public = yes
259         writable = yes
260         read only = no
261         browseable = yes
262         read list = pi
263         write list = pi
```

添加linux用户并激活：

```
sudo smbpasswd -a pi
sudo smbpasswd -e pi
```

重新启动服务：

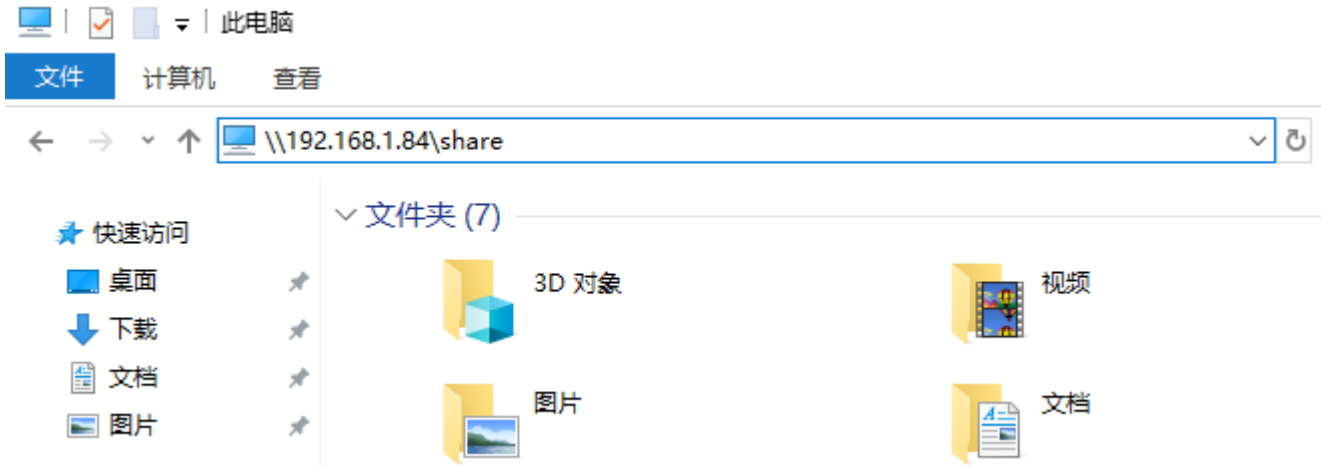
```
sudo /etc/init.d/samba restart
```

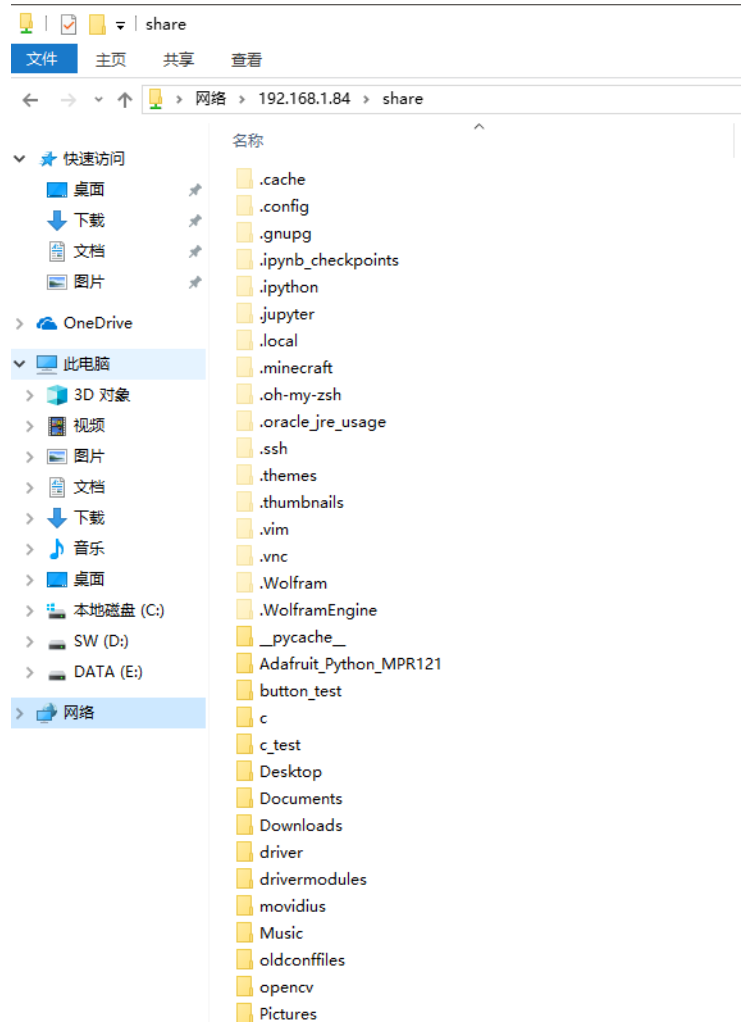
samba 服务搭建

Windows访问：

在资源管理器中输入\\<IP>\share

在对话框中输入用户名及密码即可访问





Linux中访问：

在shell中输入如下语句，将SMB挂载到/mnt/share/目录下。

sudo mount -t cifs -o username="pi",password="123456" //192.168.1.84/share /mnt/share

NFS 服务搭建

NFS，网络文件系统（英语：Network File System）是一种分布式文件系统协议，最初由Sun Microsystems公司开发。其功能旨在允许客户端主机可以像访问本地存储一样通过网络访问服务器端文件。

在 Unix 操作系统和Linux操作系统以及类Unix操作系统上，NFS 通常被频繁使用。

在Microsoft Windows操作系统上 SMB 和 NetWare核心协议（NCP）的使用比 NFS 更广泛。

在Apple Macintosh 操作系统上则 AFP 的使用更广泛。

开始搭建：

搭建NFS需要安装如下软件：

```
sudo apt-get install nfs-kernel-server
```

修改配置文件：

```
sudo vi /etc/exports
    /home/pi *(rw,sync,no_root_squash,no_subtree_check)
```

```
sudo vi /etc/exports
# 在最后添加如下配置:
<path> *(rw,sync,no_root_squash,no_subtree_check)
# <path>            :nfs客户端加载目录
# *                 :允许所有的网段访问，也可以使用具体的IP
# rw                :挂接此目录的客户端对该共享目录具有读写权限
# sync              :资料同步写入内存和硬盘
# no_root_squash    :root用户具有对根目录的完全管理访问权限
#no_subtree_check   :不检查父目录的权限

# 实例
/home/jaya/Workstation *(rw,sync,no_root_squash,no_subtree_check)
```

NFS 服务搭建，连接

重启NFS服务：

sudo /etc/init.d/rpcbind restart

sudo /etc/init.d/nfs-kernel-server restart

```
[ 12:57AM ]  [ pi@raspberrypi:~ ]
 $ sudo /etc/init.d/rpcbind restart
[ ok ] Restarting rpcbind (via systemctl): rpcbind.service.
[ 12:58AM ]  [ pi@raspberrypi:~ ]
 $ sudo /etc/init.d/nfs-kernel-server restart
[ ok ] Restarting nfs-kernel-server (via systemctl): nfs-kernel-server.service.
```

使用2号树莓派来连接一下：

sudo mount -o nolock -t nfs 192.168.1.84:/home/pi /mnt/share/

```
(test) pi@raspberrypi:/mnt $ sudo mount -o nolock -t nfs 192.168.1.84:/home/pi /
mnt/share/
(test) pi@raspberrypi:/mnt $ cd share/
(test) pi@raspberrypi:/mnt/share $ ls
01_SPI_NiXieTube_Blink.py        Downloads        Pictures
01_SPI_OLED_Full.py              driver           Public
02_SPI_NiXieTube_Static.py       drivermodules    __pycache__
03_SPI_NiXieTube_Dynamic.py      end.bmp          python_games
04_SPI_NiXieTube_Dynamic_Def.py  G57ASCII.py      python_test
05_SPI_NiXieTube_Dynamic_time.py I2C-OLED.py      python_test1
```

NFS比较适用于Unix和Linux，Widows连接需要启动NFS服务，再用mount来挂载，蛮麻烦的还不如SMB来的快。

图片来源：百度

谢谢您的观看

www.moore8.com

微信公众平台：
moore_8

QQ群： 327350729