

NodeMCU连接万维网

董峦 新疆农业大学 2018

一、WWW

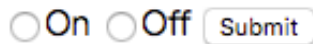
WWW（World Wide Web），中文叫做万维网，就是我们通过浏览器上网时感受到的网络。该网络上有两种角色，分别是Web客户端（即浏览器）和Web服务器程序（比如利用超文本传输协议提供服务的程序，如下图中我们在域名前加 `http://` 所访问的服务器程序）。WWW让人们可以通过浏览器访问互联网上的页面/资源。

当我们将NodeMCU的固件改为MicroPython时，便为该硬件赋予了极大地能力，比如可以将其变成Web服务器，向外部提供Web服务。它可以把硬件访问方式暴露给外部，使人们能在网络的另一端控制NodeMCU。下面我们看一个具体例子。



二、TCP socket

Socket 是在本机或不同机器间的进程间通信的技术。TCP socket 是互联网上机器间传输消息的手段。MicroPython含有 socket 模块，使我们能利用最底层的技术手段进行网络通信。在下面的示例中，NodeMCU向外提供Web服务，当我们访问NodeMCU在网络中的IP地址时，从浏览器上可以看到的页面如下：



The image shows a simple web form. It contains two radio buttons. The first radio button is selected and is followed by the text 'On'. The second radio button is not selected and is followed by the text 'Off'. To the right of these two options is a button labeled 'Submit'.

该页面非常简单，其HTML代码是：

```
<!DOCTYPE html>
<html>
<body>
    <form action="/">
        <input type="radio" name="led" value="on"> On
        <input type="radio" name="led" value="off"> Off
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

其主要是一个表单，用于提交单选按钮的值。我们如果点击Submit按钮，则单选按钮控件的值将传送给NodeMCU，我们在NodeMCU一侧监测该值得变化，当检测到特定值时便控制NodeMCU板上的LED亮或者灭。

下面是NodeMCU中运行的具体代码

```
import machine, socket

led = machine.Pin(16, machine.Pin.OUT)

html = """
<!DOCTYPE html>
<html>
<body>
<form action="/">
<input type="radio" name="led" value="on"> On
<input type="radio" name="led" value="off"> Off
<input type="submit" value="Submit">
</form>
</body>
</html>
"""

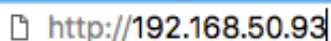
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1] # 指定80端口
s = socket.socket()
s.bind(addr)
s.listen(1) # 监听该端口(80端口)
```

```

while True:    # accept - send 循环
    cl, addr = s.accept()
    cl_file = cl.makefile('rwb', 0) # socket 以类似操作文件的方式操作网络数据
    while True:
        line = cl_file.readline()
        print(line)
        # 分析HTTP头部的数据, 如果找到特定内容, 则做相应操作
        if line.find(b'GET /?led=on') != -1:
            led.off()
        elif line.find(b'GET /?led=off') != -1:
            led.on()
        # 当浏览器的请求数据结束时, 跳出该循环
        if not line or line == b'\r\n':
            break
        # 把HTML发送给客户端的浏览器
    response = html
    cl.send(response)
    cl.close()

```

在NodeMCU中运行该段程序后, 我们从浏览器访问NodeMCU所在的局域网网址, 比如



可以看到上述包含单选按钮组的页面, 此时Putty上显示

```

b'GET / HTTP/1.1\r\n'
b'Host: 192.168.50.93\r\n'
b'Connection: keep-alive\r\n'
b'Upgrade-Insecure-Requests: 1\r\n'
b'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_3) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/64.0.3282.167 Safari/537.36\r\n'
b'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image
/apng,*/*;q=0.8\r\n'
b'Accept-Encoding: gzip, deflate\r\n'
b'Accept-Language: zh,zh-CN;q=0.9,zh-TW;q=0.8,en;q=0.7\r\n'
b'\r\n'
202

```

上图是浏览器向NodeMCU请求页面时发送的消息(HTTP的头信息)。现在我们点击单选按钮中的“On”按钮, 再点击“Submit”, 可以发现NodeMCU上的蓝色LED亮了, 同时在Putty上看到如下内容

```

b'GET /?led=on HTTP/1.1\r\n'
b'Host: 192.168.50.93\r\n'
b'Connection: keep-alive\r\n'
b'Upgrade-Insecure-Requests: 1\r\n'
b'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_3) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/64.0.3282.167 Safari/537.36\r\n'
b'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image
/apng,*/*;q=0.8\r\n'
b'Referer: http://192.168.50.93/\r\n'
b'Accept-Encoding: gzip, deflate\r\n'
b'Accept-Language: zh,zh-CN;q=0.9,zh-TW;q=0.8,en;q=0.7\r\n'
b'\r\n'

```

其中第一行就是我们在代码中监测的内容: “GET /?led=on”。当代码检测到该内容的存在时, 就认为浏览器前的用户想要点亮LED, 则我们用代码把LED点亮。

上述示例演示了如何让NodeMCU通过网络暴露其硬件资源和控制方式。该代码十分原始并面向底层，更好的Web框架有picoweb (<https://github.com/pfalcon/picoweb>)，同学们如果要在嵌入式设备比如NodeMCU上开发功能丰富的Web程序时，则需要借助这类框架程序以便提高开发效率和代码的鲁棒性。

三、NodeMCU向Web服务器传送消息

NodeMCU不仅可以扮演Web服务器，还可以作为Web客户端。当NodeMCU配置好WiFi热点或与PC连接至同一个网络，我们可以让自己的PC在网络上提供Web服务，然后让NodeMCU访问该服务。以下用Flask框架（如有Python开发环境，可通过 `pip install flask` 安装）搭建了一个简易的Web服务器

```
from flask import Flask
from flask import request, render_template
from time import strftime, localtime

DEBUG = True
SECRET_KEY = 'i93ejy89610XKPyQ128dxu00dPLMK386' # 可任意指定
messages = []

app = Flask(__name__)
app.config.from_object(__name__)

@app.route("/", methods=['GET', 'POST'])
def index():
    global messages # 用该全局变量保存数据

    if request.method == 'POST':
        msg = request.form['msg']
    else:
        msg = request.args.get('msg', '')

    if len(msg) > 0:
        msg = msg + ' received at: ' + strftime("%Y-%m-%d %H:%M:%S", \
                                                localtime())

        messages.append(msg)

    return render_template('index.html', messages=messages)

if __name__ == "__main__":
    app.run('0.0.0.0')
```

将上述代码保存成 `index.py` 文件，在命令行中执行：`python index.py` 以启动Web服务。假设NodeMCU和PC已连接到同一个网络上，PC获得的IP地址是 172.20.10.2，则在NodeMCU中运行以下代码，将向PC上的Web服务发送一则消息。

```
import usocket as socket
```

```
def main(message='', use_stream=False):
    s = socket.socket()
    addr = socket.getaddrinfo('172.20.10.2', 5000)[0][-1]
    print("Connect address:", addr)
    s.connect(addr)

    if use_stream:
        # MicroPython socket objects support stream (aka file) interface
        # directly, but the line below is needed for CPython.
        s = s.makefile("rwb", 0)
        s.write(b"GET /?msg=" + message + " HTTP/1.0\r\n\r\n")
        print(s.read())
    else:
        s.send(b"GET /?msg=" + message + " HTTP/1.0\r\n\r\n")
        print(s.recv(4096))

    s.close()

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

在PC的浏览器中访问 <http://172.20.10.2:5000> 可以看到



- HELLO received at: 2018-04-02 11:31:51

说明PC上的Web服务收到了NodeMCU发来的消息。需要补充的是，由于我们用到了Flask的模板功能，所以要正常显示该页面，你需要在 index.py 所在位置创建名为 templates 的文件夹，然后把下面HTML代码保存成 index.html 文件放入该文件夹。

```
<!DOCTYPE html>
<html>
<body>
  <ul>
    {% for msg in messages %}
      <li>{{ msg }}</li>
    {% endfor %}
  </ul>
</body>
</html>
```

四、小结

以上我演示了如何通过万维网与NodeMCU传递消息。同学们可以感受到，ESP8266这类集成WiFi功能的芯片辅以MicroPython这种功能丰富的固件使物联网应用的开发难度显著下降。Web开发对同学们来说还比较陌生，上述实例主要为同学们演示一种通信的模式以及必要的技术组成，同学们需仔细揣摩上述代码，然后在自己今后的应用中选择最合适的技术来实现相应功能。