育碧土豆服务器搭建指南

氦，今天我来教大家做一个土豆。
MicroPython 官方指南：http://docs.micropython.org/en/latest/esp32/tutorial/intro.html
固件下载地址：http://micropython.org/download
使用到的 Web 库：https://github.com/jczic/MicroWebSrv
web 库官方使用指南：https://www.youtube.com/watch?v=xscBwC1SrF4
部分工具集合：https://pan.baidu.com/s/1cV5s4yI2FfGGoz-flRhYEA

目录

准备

硬件材料

用到的材料有：

~~• 土豆~~

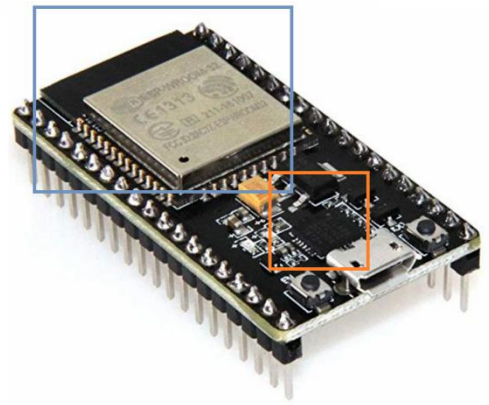
~~• 花盆~~

~~• 水~~

~~• 肥沃的泥土~~

• ESP32 开发板（○宝 20 块一个）

大概长这样：



[ESP32 开发板]

通常来说，ESP32 单片机开发板由 ESP32 单片机（蓝色框框）和 USB 转 TTL 芯片（橙色框框）组成。

当我们把 USB 线与开发板连接后，USB 转 TTL 芯片将 USB 电平转换为 TTL 电平，电脑将可以与 ESP32 单片机进行通讯。

此外，在 MicroUSB 口旁我们能看到两个按钮，它们分别是 EN 和 BOOT 按钮。

EN 按钮的功能是复位 ESP32，BOOT 的按钮是使 ESP32 进入 BOOTLOADER 模式。在 BOOTLOADER 模式下我们可以进行固件刷入操作。但本教程使用的固件刷入软件将不会使用到这个按钮。

ESP32 可以同时创建热点与连接 WIFI，支持 BLE。还有什么其他功能你们 google 一下就是了~~（才不是因为我不知道）~~。

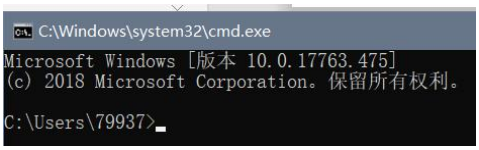
软件材料

ESPTOOL 的安装

众锁周知，我们使用的是 Micropython 的固件，既然看到了 Python，所以我们要安装 Python 和 Ptyhon 的包管理程序 Pip。~~因为我懒得重新安装一遍所以~~具体流程请大家网上搜索。

贵庚结底，我们安装 Ptyhon 和 Pip 的原因是为了获取一个叫 Esptool 的软件来刷入 Micropython 固件，所以当然有其他方法来刷固件。但是我不会。所以我只介绍这个方法。（其他方法也是拿软件刷嘛）

好了相信大家已经安装好了 Python 和 Pip，那么请打开命令行（Windows 下按 win+r 键，在弹出的框框里输入“cmd”后按回车。）



那么大家将会看到这样的界面。

Linux 用户可能是这样：

让后我们输入：pip install esptool 获取 esptool 工具。

（这个过程不赘述，详情网上搜索：如何使用 pip 安装模块和包）

Windows 下如果出现错误提示，将按提示信息操作（一般好像提示版本过低）。

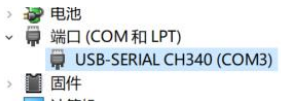
若提示权限不足，请右键以管理员身份运行。

USB 转 TTL 芯片驱动程序安装

Esptool 安装完成后，需要安装 USB 转 TTL 芯片的驱动程序。

目前主流的 USB 转 TTL 芯片为 CP2102 与 CH340。驱动程序开发板卖家应该有提供，如果不提供的话那我也莫得办法，网上搜索一下 8。

驱动安装成功后，将开发板接入电脑，打开设备管理器



应该可以看到这样的一个设备的。记住设备名称后括号的内容，即端口号。
至此软件已布置完成。

刷入 MicroPython 固件

清空 ESP32 原有固件

进入命令行，输入 esptool.py，若返回如下则说明安装正常：

```
C:\Users\79937>esptool.py
esptool.py v2.5.1
usage: esptool [-h] [--chip {auto,esp8266,esp32}] [--port PORT] [--baud BAUD]
               [--before {default_reset,no_reset,no_reset_no_sync}]
               [--after {hard_reset,soft_reset,no_reset}] [--no-stub]
               [--trace] [--override-vddsdio [{1.8V,1.9V,OFF}]]
               {load_ram,dump_mem,read_mem,write_mem,write_flash,run,image_inf
ash_id,read_flash_status,write_flash_status,read_flash,verify_flash,erase_flas
               ...

esptool.py v2.5.1 - ESP8266 ROM Bootloader Utility

positional arguments:
  {load_ram,dump_mem,read_mem,write_mem,write_flash,run,image_info,make_image,
lash_status,write_flash_status,read_flash,verify_flash,erase_flash,erase_regio
                        Run esptool {command} -h for additional help
    load_ram            Download an image to RAM and execute
    dump_mem            Dump arbitrary memory to disk
    read_mem            Read arbitrary memory location
    write_mem           Read-modify-write to arbitrary memory location
    write_flash         Write a binary blob to flash
    run                 Run application code in flash
    image_info          Dump headers from an application image
    make_image          Create an application image from binary files
    elf2image           Create an application image from ELF file
    read_mac            Read MAC address from OTP ROM
    chip_id             Read Chip ID from OTP ROM
    flash_id            Read SPI flash manufacturer and device ID
    read_flash_status   Read SPI flash status register
    write_flash_status  Write SPI flash status register
```

将开发板与电脑连接，在命令行输入：

esptool.py --port 端口号 erase_flash

以清空 ESP32 原有固件。

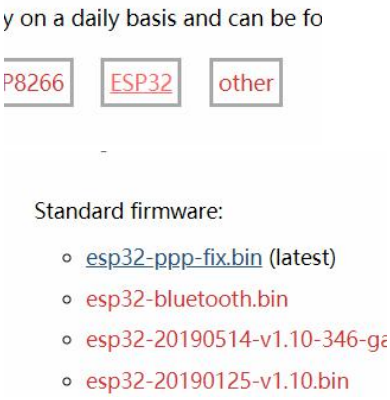
```
C:\Users\79937>esptool.py --port COM3 erase_flash
esptool.py v2.5.1
Serial port COM3
Connecting.....
Detecting chip type... ESP32
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core
MAC: 24:0a:c4:09:3a:7c
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 3.7s
Hard resetting via RTS pin...
```

出现上面这个画面表示清除成功。

如果连接不成功请检查端口号是否正确。
如果使用其他程序刷入可能需要按一下 BOOT 键。~~（反正我没用过其他程序~~

刷入 Micropython 固件

在 micropython 官网下载 ESP32 固件：http://micropython.org/download

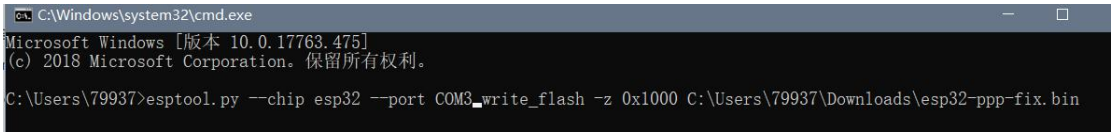


在标准固件（Standard firmware）下任选一个（如果只是搭 web 的话都能用）

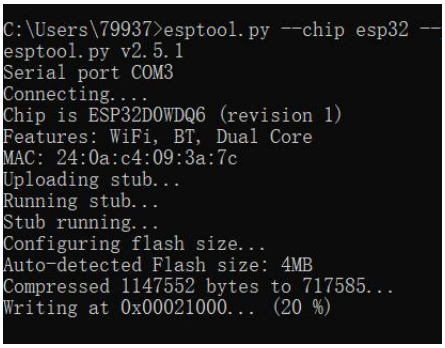
打开命令行输入：

esptool.py --chip esp32 --port 端口号 write_flash -z 0x1000 固件的绝对地址

如图



按下回车刷入：



刷入时间约为一分钟左右。

```
C:\Users\79937>esptool.py --chip esp32 --port C
esptool.py v2.5.1
Serial port COM3
Connecting....
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core
MAC: 24:0a:c4:09:3a:7c
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 1147552 bytes to 717585...
Wrote 1147552 bytes (717585 compressed) at 0x000
Hash of data verified.

Leaving...
Hard resetting via RTS pin...

C:\Users\79937>_
```

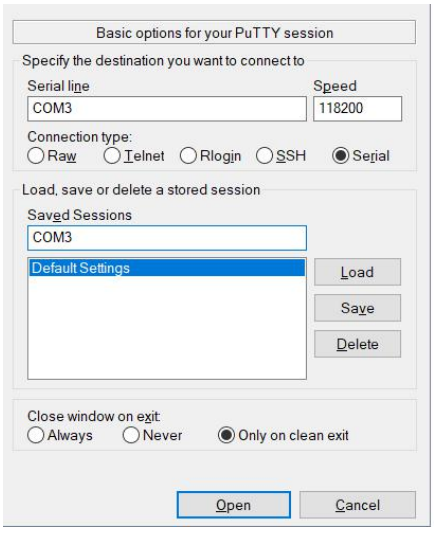
刷入完成。

连接 ESP32

连接 ESP32 我用到的软件是 Putty

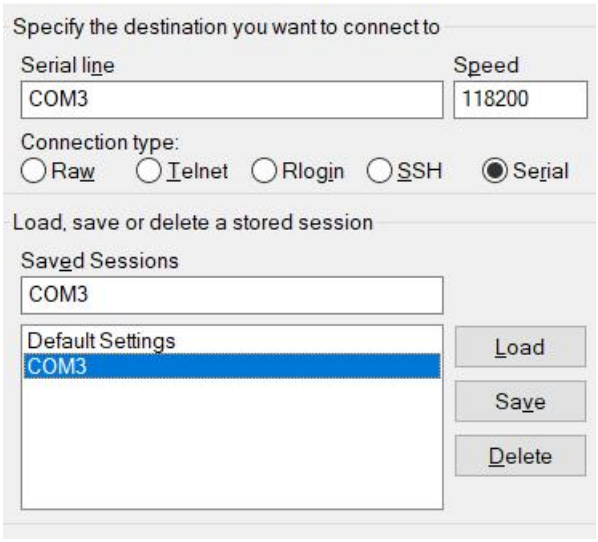
PUTTY

打开 Putty：

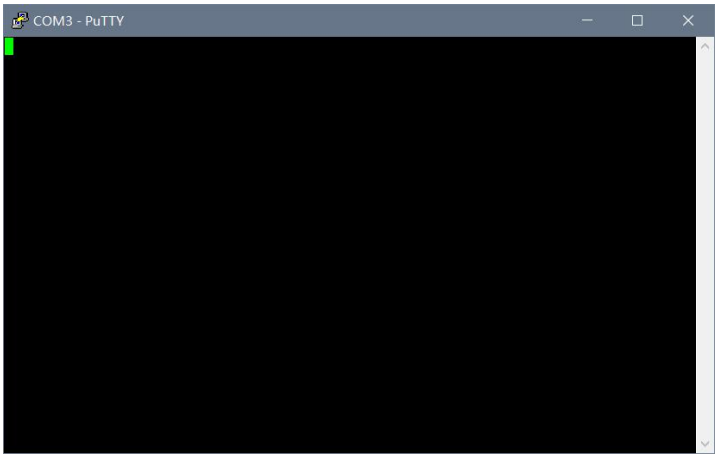


在 Connection type 中选择 Serial 即串口。
在 Serial line 中填写端口号，Speed 即波特率填写 118200。
在 Saved Sessions 填写保存的名字。
按下 Save 保存。

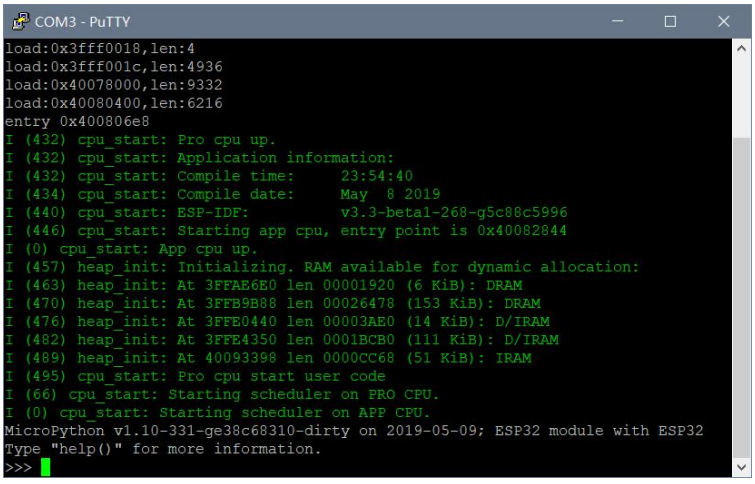
双击刚才保存的 Sessions：



连接上后可能啥也没有：



按一下 ESP32 开发板上的 EN 按钮重置一下：



至此已成功连接 ESP32。

值得一提的是，我们可以像 python 一样在这里进行逐句输入：

```
>>> print("Dimsmary niubi")
Dimsmary niubi
>>>
```

（打印“Dimsmary niubi”）

```
>>> for i in range(10):
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
```

（创建一个 0-9 的列表并打印）

```
>>> c = "dimsmary"
>>> if c == "dimsmary":
...     print("handsome")
...
handsome
```

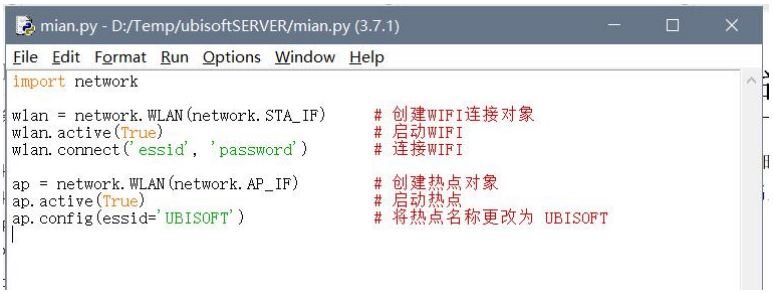
（我觉得这个不用解释）

写入初始化脚本程序

ESP32 可以同时创建热点和连接 wifi，接下来我们编写一个脚本让它连接上我们的连路由器并创建一个名为 UBISFOT 的热点。

程序编写

你可以用记事本编写后缀名为.py 的文件，我这里使用的是 Ptyhon 的 IDLE。



输入以下代码：

```
import network                          # 导入 network 模块
wlan = network.WLAN(network.STA_IF)     # 创建 WIFI 连接对象
wlan.active(True)                       # 启动 WIFI
wlan.connect('essid', 'password')       # 连接 WIFI 引号内输入 ssid 和密码
```

```
ap = network.WLAN(network.AP_IF)          # 创建热点对象
ap.active(True)                           # 启动热点
ap.config(essid='UBISOFT')                # 将热点名称更改为 UBISOFT
```

并保存。

这样我们就得到了一个 main.py 文件。

ESP32 启动时，将执行 boot.py 后执行 main.py

程序写入 ESP32

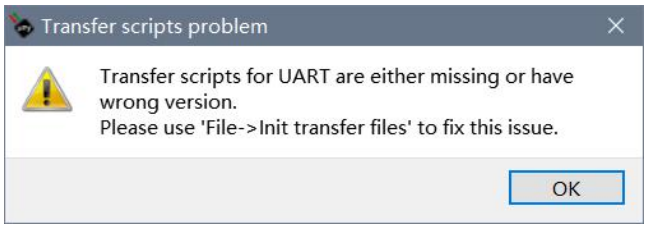
程序的写入我们用到工具：uPyLoader

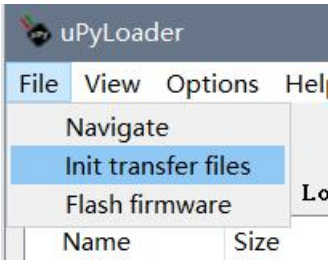
打开 uPyLoader：

在 Connection 中选择正确的端口号后点击 Connect

如果没有端口号的话，点击右边的刷新按钮试试，或重新插拔 USB。

首次连接将会提示缺少传输脚本，点击 OK。



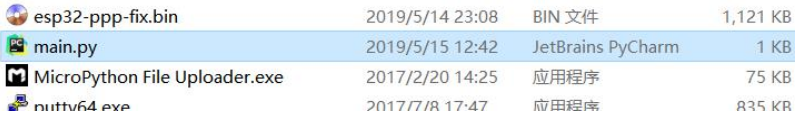


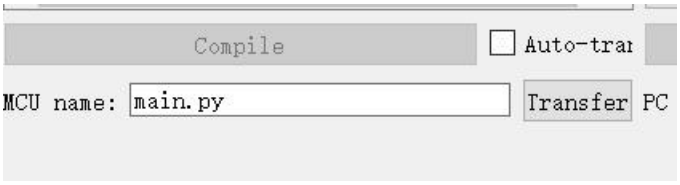
在 File->Init transfer 中进行脚本初始化。

接下来将我们的 mian.py 传输到 ESP32：

（要将 main.py 与 uPyLoader 放到同一目录）

选中 main.py 后

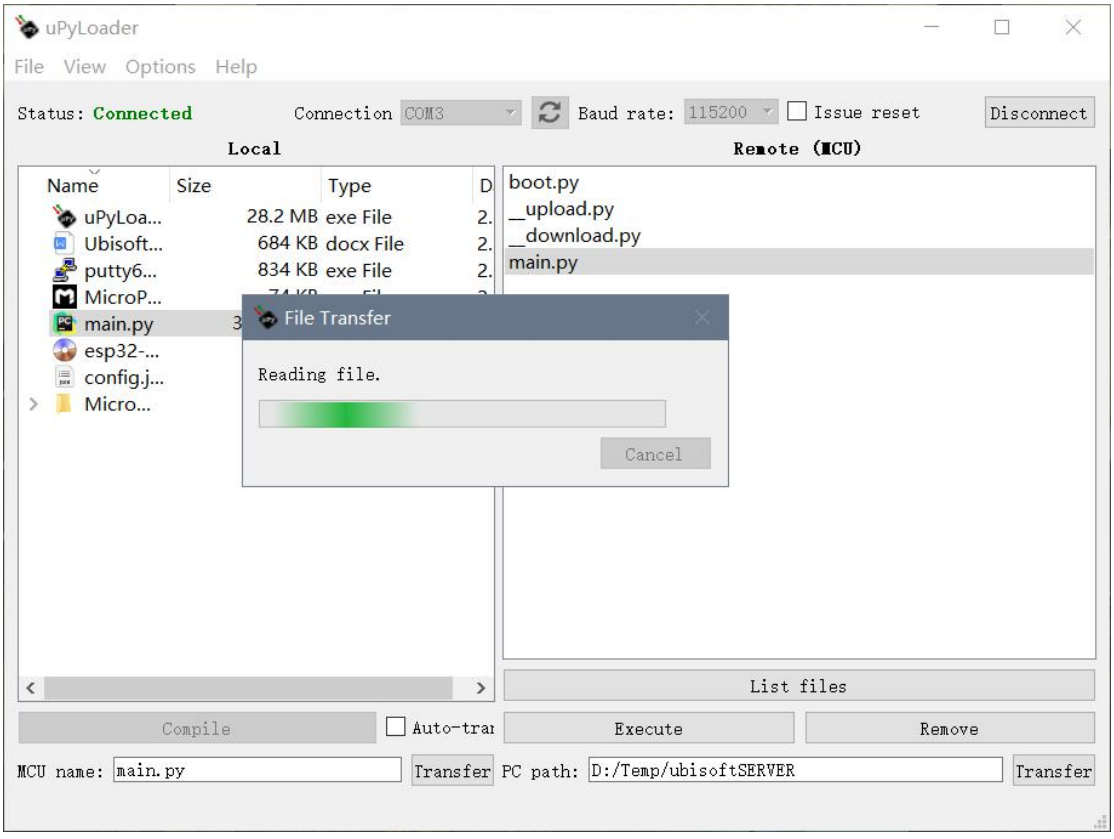




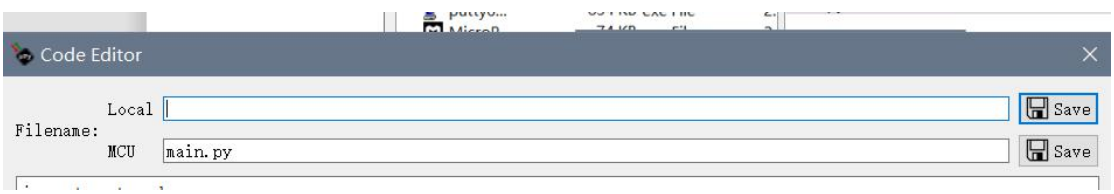
点击下方的 Transfer

传输完成。

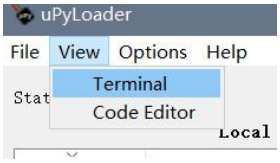
双击文件可以进行修改：

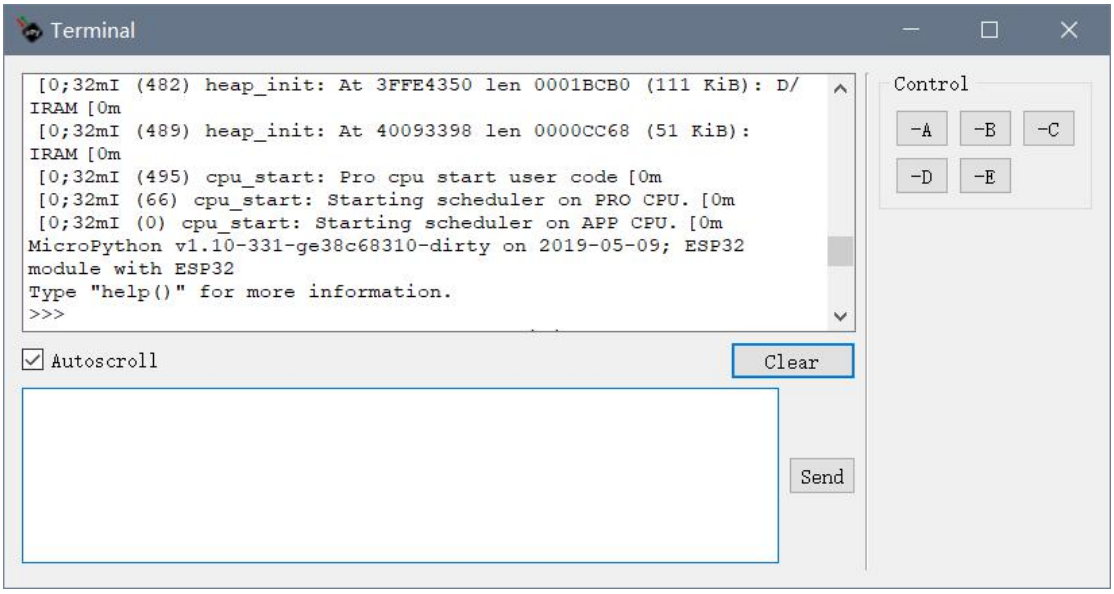


点击第二个 Save 将实时更新该文件到 ESP32:



在 VIEW 中也可打开 PUTTY 一样的终端：





main.py 传输完成后，且 Terminal 打开时按下重启按键：

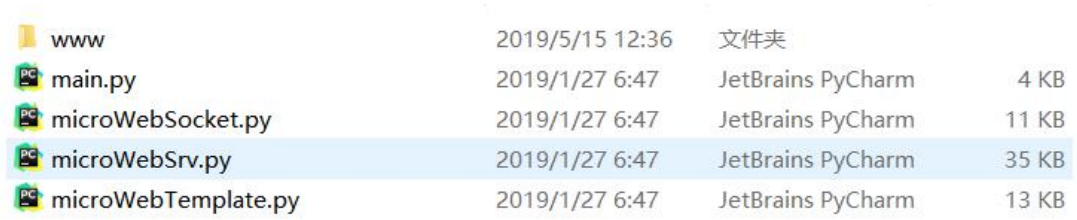
```
Type "help()" for more information.
>>> [0;32mI (468) network: event 13 [0m
 [0;32mI (478) network: event 14 [0m
 [0;32mI (478) network: event 13 [0m
I (568) wifi: new:<1,1>, old:<1,0>, ap:<1,1>, sta:<1,0>, prof:1
I (1138) wifi: state: init -> auth (b0)
I (1138) wifi: state: auth -> assoc (0)
I (1178) wifi: state: assoc -> run (10)
I (1388) wifi: connected with CtOS, channel 1, bssid =
20:76:93:20:c7:68
I (1388) wifi: pm start, type: 1

 [0;32mI (1398) network: CONNECTED [0m
 [0;32mI (2218) event: sta ip: 192.168.99.235, mask:
255.255.255.0, gw: 192.168.99.1 [0m
 [0;32mI (2218) network: GOT_IP [0m
```

连接成功后将返回连接的 IP
同样我们也可以看到它创建的热点

写入 WEB 服务程序

文件准备

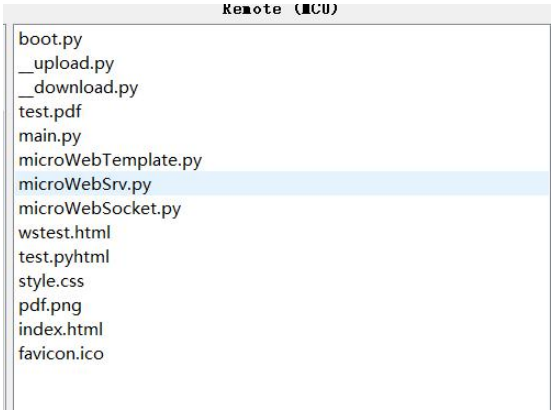
在 https://github.com/jczic/MicroWebSrv 下载源代码，我们需要的文件有：



www 文件夹内是 web 服务端的根目录，如果不理解下面我们再作解释。
将下载的源码中的 main.py 重命名为 start.py

写入 ESP32

上图的所有 py 文件传输至 ESP32。
再将 www 内文件传输至 ESP32（可选），传输完成后如图：



现在所有文件都在 ESP32 的根目录，且没有 www 文件夹。
因为辣鸡 uPyLoader 不支持文件夹操作，所以得手动打代码淦
打开终端（Terminal），将下列代码逐句回车输入：

```
Import os
os.listdir()          # 列出当前目录的所有文件
os.mkdir('www')       # 创建名为 www 的目录
os.rename('test.pdf', 'www/test.pdf')  # 将 test.pdf 移动到 www 目录下
# 重复以上 rename 操作将属于 www 文件夹的文件移动
```

移动完成后应该为：

```
>>> os.listdir()
['boot.py', '__upload.py', '__download.py', 'start.py',
'main.py', 'microWebTemplate.py', 'microWebSrv.py',
'microWebSocket.py', 'www']
```

我们可以测试一下程序工作情况：

输入：import start 后服务器程序将执行。

此时输入 ESP32 在局域网中的地址：

返回的将是 www 文件夹内的 index.html 文件。

至此 web 服务器搭建完成。

传输一个小游戏

通用的文件操作指令：

```
import os            # 导入 os 模块以操作文件
os.chdir(path)       # 更改当前操作目录
os.getcwd()          # 获取当前操作目录
os.listdir()         # 列出当前目录下的所有文件 可加参数获取其他目录的文件列表
os.mkdir(path)       # 创建一个目录
os.remove(path)      # 移除一个文件
os.rmdir(path)       # 移除一个目录
os.rename(old_path, new_path)        # 移动或重命名文件
```

将网页文件上传到 ESP32 的 www 文件夹内，在局域网内的其他设备将可以通过浏览器输入：单片机 IP/文件名 的方式访问。

在工具集合包内有一些网页小游戏（GitHub 下载的源项目地址找不着了 0.0），如果你通过浏览器运行没问题的话可以上传到 ESP32 上通过浏览器访问。这样 ESP32 就变成了一个游戏服务器（伪