# 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: <u>http://docs.micropython.org/en/latest/esp32/tutorial/intro.html</u> 固件下载地址: <u>http://micropython.org/download</u> 使用到的 Web 库: <u>https://github.com/jczic/MicroWebSrv</u> web 库官方使用指南: <u>https://www.youtube.com/watch?v=xscBwC1SrF4</u>

部分工具集合: <u>https://pan.baidu.com/s/1cV5s4yI2FfGGoz-flRhYEA</u>

<del>育碧</del> 土豆服务器搭建指南
准 \$ 备
硬件材料
软件材料
ESPTOOL 的安装
USB 转 TTL 芯片驱动程序安装
刷入 MicroPython 固件
清空 ESP32 原有固件
刷入 Micropython 固件
连接 ESP32
PUTTY
写入初始化脚本程序
程序编写
程序写入 ESP32
写入 WEB 服务程序
文件准备
写入 ESP32

# 准台备

### 硬件材料

用到的材料有:

<u>- 土豆</u>

● 花盆

<del>• 水</del>

- 肥沃的泥土
- ESP32 开发板(O 宝 20 块一个)





[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<del>(才不是因为我不知道)</del>。

### 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<mark>安装</mark> 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<mark>安装好了 Python 和 Pip</mark>,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:

				pi@
文件(E)	编辑(E)	标签(工)	帮助( <u>H</u> )	
pi@raspb	errypi:~	S		

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。

最佳匹配	
■ 命令提示符 桌面应用	G 以管理品身份远行
<b>応田</b>	□ 打开文件位置
194 CD	

## 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。

<mark>驱动安装成功后</mark>,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。<mark>记住设备名称后括号的内容,即端口号。</mark> 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, <mark>输入 esptool.py</mark>	, 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, esp826 [before {default_reset, [after {hard_reset, soft [trace] [override-vdd {load_ram, dump_mem, read_m ash_id, read_flash_status, write_flash_stat 	6, esp32}] [port PORT] [baud BAUD] no_reset, no_reset_no_sync}] _reset, no_reset}] [no-stub] sdio [41.88,1.99, OFF]]] em, write_mem, write_flash, run, image_inf tus, read_flash, verify_flash, erase_flash
<pre>espt001.py v2.5.1 Est3200 KOW B001104u positional arguments:     {load_ram, dump_mem, read_mem, write_mem, lash_status, write_flash_status, read_flas     Run espt001 {com     load_ram Download an imag     dump_mem Dump arbitrary m     read_mem Read arbitrary m     reite_mem Read-modify-writ     write_flash Write a binary b     run Run application     image_info Dump headers fro     make_image Create an applic     elf2image Create an applic     read_mac Read MAC address     chip_id Read SPI flash m     read_flash_status Write SPI flash</pre>	write_flash,run,image_info,make_image, h,verify_flash,erase_flash,erase_regio mand) -h for additional help e to RAM and execute emory to disk emory location lob to flash code in flash m an application image ation image from ELF file from OTP ROM m OTP ROM m OTP ROM anufacturer and device ID tatus register status register
将开发板与电脑连接,在命令	∻行输入:
esptool.pyport 端目号	erase_flash
以相工 ESP32 原有 回任。 C:\Users\79937>esptool.py 一port COM3 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 Hard resetting via RTS pin	3 erase_flash ) 3.7s

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。<del>(反正我没用过其他程序</del>

## 刷入 Micropython 固件

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

y on a daily basis and can be fo

ESP32	other
	ESP32

Standard firmware:

- esp32-ppp-fix.bin (latest)
- esp32-bluetooth.bin
- esp32-20190514-v1.10-346-ga
- esp32-20190125-v1.10.bin

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

C:\Windows\system32\cmd.ex licrosoft Windows [版本 10.0.17763.475] (c) 2018 Microsoft Corporation。保留所有权利。 :\Users\79937>esptool.py --chip esp32 --port COM3\_write\_flash -z 0x1000 C:\Users\79937\Downloads\esp32-ppp-fix.bin

按下回车刷入:

```
C:\Users\79937>esptool.py --chip esp32 -
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...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 1147552 bytes to 717585...
Writing at 0x00021000... (20%)
```

刷入时间约为一分钟左右。

C:\Users\79937>esptool.py --chip esp32 --port CO 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:

Serial line	Speed
COM3	118200
Connection type:	⊖ <u>S</u> SH
.oad, save or delete a stored sessior Saved Sessions	ı
СОМЗ	
Default Settings	Load
	122
	Sa <u>v</u> e
	Sa <u>v</u> e Delete

- 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。

#### <mark>双击刚才保存的 Sessions</mark>:

Serial line			Speed
СОМЗ			118200
Connection type: ORaw OIe	Inet ORlog <u>i</u> r	⊖ <u>s</u> s⊦	l
oad, save or dele	te a stored sess	ion	
₋oad, save or dele Sav <u>e</u> d Sessions	te a stored sess	ion	
Load, save or dele Sav <u>e</u> d Sessions COM3	te a stored sess	ion	]
Load, save or dele Sav <u>e</u> d Sessions COM3 Default Settings	te a stored sess	ion	Load
Load, save or dele Sav <u>e</u> d Sessions COM3 Default Settings COM3	te a stored sess	ion	Load
Load, save or dele Sav <u>e</u> d Sessions COM3 Default Settings COM3	te a stored sess	ion	Load Sa <u>v</u> e

#### 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:

ල් COM3 - PuTTY			×
load:0x3fff0018,len:4			~
load:0x3fff001c,len:4936			
load:0x40078000,len:9332			
load:0x40080400,len:6216			
entry 0x400806e8			
I (432) cpu start: Pro cpu up.			
I (432) cpu start: Application information:			
I (432) cpu start: Compile time: 23:54:40			
I (434) cpu start: Compile date: May 8 2019			
I (440) cpu start: ESP-IDF: v3.3-beta1-268-g5c88c5996			
I (446) cpu start: Starting app cpu, entry point is 0x40082844			
I (0) cpu start: App cpu up.			
I (457) heap init: Initializing. RAM available for dynamic alloca	tion:		
I (463) heap init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM			
I (470) heap init: At 3FFB9B88 len 00026478 (153 KiB): DRAM			
I (476) heap init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM			
I (482) heap init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM			
I (489) heap init: At 40093398 len 0000CC68 (51 KiB): IRAM			
I (495) cpu start: Pro cpu start user code			
I (66) cpu start: Starting scheduler on PRO CPU.			
I (0) cpu start: Starting scheduler on APP CPU.			
MicroPython v1.10-331-ge38c68310-dirty on 2019-05-09; ESP32 modul	e with	ESP32	
Type "help()" for more information.			
>>>			~

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:



(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<mark>连接上我们的连路</mark> 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。

🚴 mian.py - D:/Temp/ubisoftSERVER/mian.py (3	.7.1)				
<u>File Edit Format Run Options Window He</u>	elp				
import network					$\sim$
wlan = network. WLAN (network. STA_IF)	# 创建WIFI连	接对象			
wlan. active([rue)  wlan. connect('essid', 'password')	# 后切WIFI # 连接WIFI				
ap = network.WLAN(network.AP IF)	# 创建热点对	象			F
ap. active (True)	# 启动热点	· 西· · · · ·	OFF		1.
ap. config(essid= UBISOFI )	# 1月28日有朴	更以/J UBIS(	JF I		
1					
输入以下代码:					
import network		# 导入	netw	/ork	寺块
		ル 人口7曲			いた
wian = network.wLAN(network.	SIA_IF)	# 刨娃	VVIFI	连按)	<b>刈</b> 家
wlan.active(True)		# 启动	j WIFI		
wian connect('essid' 'nassword'	1	# 连控		리문	内站
wian.connect( essid, password	)	# 建按	, VVIFI	17	r 1 48

```
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。ジ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

<mark>打开 uPyLoader</mark>:

🗞 uPyLoader	-		×
File View Options Help			
Status: Disconnected Connection COM3	<ul> <li>Baud rate: 115200</li> <li>Issue reset</li> </ul>	Conne	ect
Local	Remote (MCU)		
在 Connection 中选择正确的端口号			

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。

💩 Trai	nsfer scri	pts problem		×
	Transf wrong Please	er scripts for y version. e use 'File->In	UART are eithe it transfer files'	er missing or have to fix this issue.
				ОК
🏷 U	iPyLoad	der		
File	View	Options	Hel	
ſ	Navigat	te		
1	nit tran	sfer files		
ſ	Flash fir	mware	Lo	
1	Name	Size	9	

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

esp32-ppp-fix.bin	2019/5/14 23:08	BIN 文件	1,121 KB
😫 main.py	2019/5/15 12:42	JetBrains PyCharm	1 KB
MicroPython File Uploader.exe	2017/2/20 14:25	应用程序	75 KB
Pouttv64 exe	2017/7/8 17.47	应田程序	835 KB

	Compile	🗌 Auto-tran
MCU name:	main.py	Transfer PC

点击下方的 Transfer

	Remote (MCU)
boot.py _upload.py _download.py	
main.py	

传输完成。

双击文件可以进行修改:

le View Options	Help					
tatus: Connected	Connecti	on COM3	Baud rate: 1	115200 - Issue	reset	Disconnec
Name Size vuPyLoa Ubisoft putty6 MicroP main.py sesp32 config.j Micro	Type 28.2 MB exe F 684 KB docx 834 KB exe F 3 File Trans Reading file	File 2. File 2. File 2. File 2.	boot.py _upload.py _download.py main.py Cancel			
c		> [		List files		
Compi	le	Auto-trai	Execute		Remove	
CU name: main.py		Iransfer P	C path: D:/Temp/ubi	isoftSERVER		Transf

#### 点击第二个 Save 将实时更新该文件到 ESP32:

		MicroD	74.40 61	2	_
🐞 Code Editor					
Loca	al				Save
Filename: MCU	main.py				🔚 Save
· · · · · · · · · · · · · · · · · · ·					

#### 在 VIEW 中也可打开 PUTTY 一样的终端:



🐱 Terminal			×
[0;32mI (482) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/ IRAM [0m [0;32mI (489) heap_init: At 40093398 len 0000CC68 (51 KiB): IRAM [0m [0;32mI (495) cpu_start: Pro cpu start user code [0m [0;32mI (66) cpu_start: Starting scheduler on PRO CPU. [0m [0;32mI (0) cpu_start: Starting scheduler on APP CPU. [0m MicroPython v1.10-331-ge38c68310-dirty on 2019-05-09; ESF32 module with ESF32 Type "help()" for more information. >>> ✓ Autoscroll Clear	Contr —A —D	ol ──B	-C

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



## 文件准备

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

1	www	2019/5/15 12:36	文件夹	
PC	main.py	2019/1/27 6:47	JetBrains PyCharm	4 KB
PC	microWebSocket.py	2019/1/27 6:47	JetBrains PyCharm	11 KB
PC	microWebSrv.py	2019/1/27 6:47	JetBrains PyCharm	35 KB
PC	microWebTemplate.py	2019/1/27 6:47	JetBrains PyCharm	13 KB

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

## 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:

	Kemote	(IICU)		
boot.py				
_upload.py				
download.py				
test.pdf				
main.py				
microWebTemplate.py				
microWebSrv.py				
microWebSocket.py				
wstest.html				
test.pyhtml				
style.css				
pdf.png				
index.html				
favicon.ico				

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<del>淦</del> 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:



传输一个小游戏

通用的文件操作指令:

#	导入 os 模块以操作文件
#	更改当前操作目录
#	获取当前操作目录
#	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
# ′	创建一个目录
# 🗦	移除一个文件
# 3	移除一个目录
th, n	new_path) # 移动或重命名文件
	# # # # # th, r

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