

use [esp8266](#) module over sdio on [cubieboard](#) as wifi card

~~https://github.com/0x72a0a0~~

Contents: [Dobrica Pavlinu's random unstructured stuff]

- [Dobrica Pavlinu's random unstructured stuff \(links\)](#)
- [Dobrica Pavlinu's random unstructured stuff \(setup\)](#)
 - ◆ [Dobrica Pavlinu's random unstructured stuff \(cubieboard pinout\)](#)
 - ◆ [Dobrica Pavlinu's random unstructured stuff \(ESP-201 pinout from top\)](#)
 - ◆ [Dobrica Pavlinu's random unstructured stuff \(CH_EN mapping to GPIO pin\)](#)
- [Dobrica Pavlinu's random unstructured stuff \(additional components\)](#)
- [Dobrica Pavlinu's random unstructured stuff \(clear the QE \(quad enable\) bit in the sreg2 of a W25Q40BV or similar\)](#)
- [Dobrica Pavlinu's random unstructured stuff \(bootloader\)](#)
 - ◆ [Dobrica Pavlinu's random unstructured stuff \(setup baudrate to 74880\)](#)
 - ◆ [Dobrica Pavlinu's random unstructured stuff \(boot modes\)](#)
 - ◇ [Dobrica Pavlinu's random unstructured stuff \(IO15 -> 3V3\)](#)
 - ◇ [Dobrica Pavlinu's random unstructured stuff \(IO15 -> GND\)](#)

links

<https://hackaday.io/project/8678-rpi-wifi-hat>

<https://github.com/al177/esp8089>

setup

cubieboard pinout

```
root@cubieboard:/home/dpavlin# cat mmc3-sdio.txt
22 PI4 2 1 2 [mmc mmc3]          SCD3_CMD
24 PI5 2 1 2 [mmc mmc3]          SCD3_CLK
26 PI6 2 1 2 [mmc mmc3]          SCD3_D0
28 PI7 2 1 2 [mmc mmc3]          SCD3_D1
30 PI8 2 1 2 [mmc mmc3]          SCD3_D2
32 PI9 2 1 2 [mmc mmc3]          SCD4_D3
```

ESP-201 pinout from top

15 IO0	13 IO15
14 IO2	12 IO13
18 D2 GPIO9	10 IO12
21 CLK GPIO6	9 IO14
20 CMD GPIO11	8 XPD GPIO16
22 D0 GPIO7	7 CHIP_EN
23 D1 GPIO8	32 RST
19 D3 GPIO10	6 TOUT ADC
16 IO4	24 IO5
3.3V	GND
3.3V	GND

CH_EN mapping to GPIO pin

```
PE4 "132" => "PE4", -> CHIP_EN
```

```
root@cubieboard:/home/dpavlin/linux-gpio-pinout# modprobe esp8089 esp_reset_gpio=132 # PE4
[Sun Jun 30 19:19:24 2019]
          ***** EAGLE DRIVER VER:bdf5087c3deb*****

[Sun Jun 30 19:19:24 2019] ESP8089 reset via GPIO 132
[Sun Jun 30 19:19:35 2019] esp_sdio_init ----- RETRY -----
[Sun Jun 30 19:19:35 2019] ESP8089 reset via GPIO 132
[Sun Jun 30 19:19:35 2019] ESP8089 reset via GPIO 132
[Sun Jun 30 19:19:46 2019] esp_sdio_init ----- RETRY -----
[Sun Jun 30 19:19:46 2019] ESP8089 reset via GPIO 132
[Sun Jun 30 19:19:47 2019] ESP8089 reset via GPIO 132
[Sun Jun 30 19:19:58 2019] esp_sdio_init ----- RETRY -----
[Sun Jun 30 19:19:58 2019] ESP8089 reset via GPIO 132
[Sun Jun 30 19:19:58 2019] ESP8089 reset via GPIO 132
[Sun Jun 30 19:20:09 2019] esp_sdio_init ----- RETRY -----
[Sun Jun 30 19:20:09 2019] ESP8089 reset via GPIO 132
modprobe: ERROR: could not insert 'esp8089': No such device
[Sun Jun 30 19:20:10 2019] eagle sdio can not power up!
```

additional components

<https://hackaday.io/project/8678-rpi-wifi/details>

missing 200 ohm resistors on sdio lines

4.7k pullup on CH_PD <https://github.com/al177/esp8089/issues/4>

rpi needs dtoverlay=sdio,poll_once=false

it's still not clear to me do I have to remove spi flash or not

newer project with simpler overview

- <https://hackaday.io/project/12980-pi2wifi>
- <https://sites.google.com/site/mincepi/pi2wifi>

it suggests 33 ohm resistors on sdio, includes dts overlay)

<https://hackaday.io/project/8678/instructions> also suggest 33 ohm resistors...

clear the QE (quad enable) bit in the sreg2 of a W25Q40BV or similar

<https://github.com/jacksonliam/rpi-bitbang-spiflash>

bootloader

setup boudrate to 74880

[anyboud.c](#)

<https://gist.githubusercontent.com/sentinel/3f1a984533556cf890d9/raw/8a35958138b1167fce5c2301a73e2>

You have to first open terminal in some valid baud rate

```
root@cubieboard:~# microcom -p /dev/ttyS1
```

and then in another terminal force new speed for port with:

```
dpavlin@cubieboard:~/anybaud$ ./anybaud /dev/ttyS1 74880  
Changed successfully.
```

boot modes

<https://github.com/esp8266/esp8266-wiki/wiki/Boot-Process#esp-boot-modes>

The Espressif code can boot in different modes, selected on power-up based on GPIO pin levels. (MTDO is equivalent to GPIO15).

MTDO	GPIO0	GPIO2	Mode	Description
L	L	H	UART	Download code from UART
L	H	H	Flash	Boot from SPI Flash
H	x	x	SDIO	Boot from SD-card

In the bootup message 'boot mode:(x,y)' three low bits of x are {MTDO, GPIO0, GPIO2}.

IO15 -> 3V3

My guess from espressif powerpoint this is sdio boot mode (and it doesn't work for me right now)

```
ets Jan 8 2013,rst cause:1, boot mode:(7,7)  
  
waiting for host
```

IO15 -> GND

According to hackaday thread, this is correct way to force SDIO mode, serial output is:

```
ets Jan 8 2013,rst cause:1, boot mode:(3,7)
```

```
load 0x40100000, len 31020, room 16
```

```
ets Jan 8 2013,rst cause:1, boot mode:(3,7)
```

This seems like loading from spi flash to me. If I don't touch kernel module, but just toggle CH_EN pin

```
root@cubieboard:/home/dpavlin# cat esp-reset.sh  
#!/bin/sh -xe
```

```
cd /sys/class/gpio/  
echo 132 > export  
echo out > gpio132/direction  
echo 0 > gpio132/value  
echo 1 > gpio132/value  
echo 132 > unexport  
root@cubieboard:/home/dpavlin#
```

I get full bootloader output:

```
ets Jan 8 2013,rst cause:1, boot mode:(3,7)
```

```
load 0x40100000, len 31020, room 16
```

```
tail 12
```

```
chksum 0x79
```

```
ho 0 tail 12 room 4
```

```
load 0x3ffe8000, len 2888, room 12
```

```
tail 12
```

```
chksum 0x6a
```

```
ho 0 tail 12 room 4
```

```
load 0x3ffe8b50, len 14864, room 12
```

```
tail 4
```

```
chksum 0x45
```

```
csum 0x45
```

It seems that esp tries to load first part, but fails and kernel module toggle CH_PD and resets it. Time to try soldering resistors between 200 and 33 ohms I guess...