ESP8266 unter Linux

- Doku-Repo: https://github.com/esp8266/esp8266-wiki/wiki

- Meist (halbwegs) aktuell, enthält aber kleinere Fehler und Ungenauigkeiten

- Auf Ubuntu-Systeme ausgerichtet

ESP8266 unter Linux

- Cross-Compiler: „XTensa Crosstools-NG“
  - Compiler, Linker, Debugger, Basis-Libraries
- SDK: Espressif SDK
  - Libraries, Beispiele, Tools
- Weitere Libraries (binary only)
- Flash-Tool: esptool.py

ESP2866 unter Linux

- Doku im Wiki
- Shell-Skript für Ubuntu: toolchain.sh

  https://github.com/esp8266/esp8266-wiki/blob/master/toolchain.sh

- Oder manuell....
- Man sollte aber bei den Default-Pfaden unterhalb von `/opt/Espressif` bleiben.

Abhängigkeiten installieren

Unter Ubuntu:

```
apt-get install git autoconf build-essential
gperf bison flex texinfo libtool libncurses5-
dev wget gawk libc6-dev-amd64 python-serial
libexpat-dev
```

Unter anderen Distributionen heißen diese ähnlich..

XTensa Crosstools-NG

```
cd /opt/Espressif

git clone -b lx106
git://github.com/jcmvbkbc/crosstool-NG.git

cd crosstool-NG

./bootstrap && ./configure --prefix=`pwd` &&
make && make install

./ct-ng xtensa-lx106-elf

./ct-ng build

PATH=$PWD/builds/xtensa-lx106-elf/bin:$PATH
```

Espressif SDK

```
cd /opt/Espressif
# mkdir ESP8266_SDK  <- Unfug
wget -O esp_iot_sdk_v0.9.3_14_11_21.zip
https://github.com/esp8266/esp8266-
wiki/raw/master/sdk/esp_iot_sdk_v0.9.3_14_11_21.zip
wget -O esp_iot_sdk_v0.9.3_14_11_21_patch1.zip
https://github.com/esp8266/esp8266-
wiki/raw/master/sdk/esp_iot_sdk_v0.9.3_14_11_21_patch1.
zip
unzip esp_iot_sdk_v0.9.3_14_11_21.zip
unzip esp_iot_sdk_v0.9.3_14_11_21_patch1.zip
mv esp_iot_sdk_v0.9.3 ESP8266_SDK
mv License ESP8266_SDK/
```

Dies installiert Version 0.9.3. Aktuell ist 0.9.5...

Patchen des SDK

```
cd /opt/Espressif/ESP8266_SDK

sed -i -e 's/xt-ar/xtensa-lx106-elf-ar/' -e
's/xt-xcc/xtensa-lx106-elf-gcc/' -e 's/xt-
objcopy/xtensa-lx106-elf-objcopy/' Makefile

mv examples/IoT_Demo .
```

Installieren der XTensa Libraries

Binary-Libraries für C und Hardware-Abstraktion (es gibt aber auch Sourcen).

- `cd /opt/Espressif/ESP8266_SDK`
- `wget -O lib/libc.a https://github.com/esp8266/esp8266-wiki/raw/master/libs/libc.a`
- `wget -O lib/libhal.a https://github.com/esp8266/esp8266-wiki/raw/master/libs/libhal.a`
- `wget -O include.tgz https://github.com/esp8266/esp8266-wiki/raw/master/include.tgz`
- `tar -xvzf include.tgz`

ESP image tool

```
cd /opt/Espressif
```

```
wget -O esptool_0.0.2-1_i386.deb
https://github.com/esp8266/esp8266-
wiki/raw/master/deb/esptool_0.0.2-1_i386.deb
```

```
dpkg -i esptool_0.0.2-1_i386.deb
```

Umwandeln nach RPM:

```
alien -r esptool_0.0.2-1_i386.deb
```

Auspacken:

```
ar x esptool_0.0.2-1_i386.deb
```

```
tar xzvf data.tar.gz
```

ESP Upload Tool

```
cd /opt/Espressif

git clone
https://github.com/themadinventor/es
ptool esptool-py

ln -s $PWD/esptool-py/esptool.py
crosstool-NG/builds/xtensa-lx106-
elf/bin/
```

Das Blinky-Beispiel

```
cd /opt/Espressif
```

```
git clone https://github.com/esp8266/source-code-examples.git
```

```
cd source-code-examples/blinky
```

```
make
```

produziert:

```
firmware/0x00000.bin firmware/0x40000.bin
```

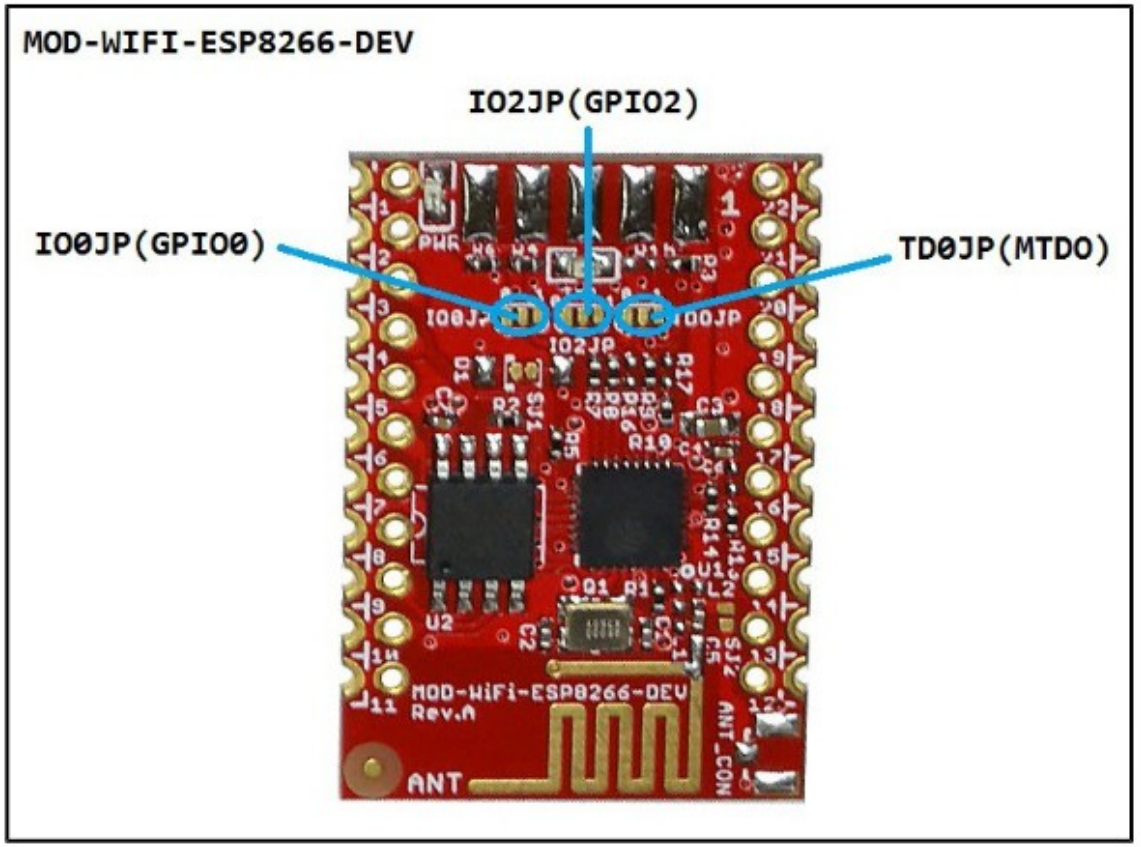
Andere Beispiele müssen noch mit `esptool` vorbereitet werden.

Siehe:

```
https://github.com/esp8266/esp8266-wiki/wiki/Building#preparing-the-firmware-image
```

MOD-WIFI-ESP8266-DEV: IoT für 5,50€

- Webshop: `https://www.olimex.com/Products/IoT/MOD-WIFI-ESP8266-DEV/`
- Etwas teurer, dafür aber mehr Pins
- Open Hardware: Schaltplan, Eagle-Dateien



- 3.3 Volt (nicht 5V-fest)
- Ein 5V/3V3-UART ist hilfreich
- Löt-Jumper für Bootloader-Config (auch auf pins herausgeführt)
- Reset-Pin
- Ein Prototyping-Adapter wäre praktisch....

The positions for the all the modes are printed on the board itself. The table looks like this:

MODE / JUMPER	TD0JP(MTDO)	IO0JP(GPIO0)	IO2JP(GPIO2)
SDIO	1	x	x
UART	0	0	1
FLASH (DEFAULT)	0	1	1

Kommunikation mit dem Bootloader

- Bootloader verwendet 74880 bps. Sehr ungewöhnlinche Geschwindigkeit.
- Eigentlich sollte das ROM Auto-Bauden.....aber wir sind neugierig (und lernen was dabei...)

Serielle Kommunikation mit 74880bps

- Man kann mit `setserial(1)` nicht-Standard-Geschwindigkeiten setzen. Aber nicht mit jedem Controller. (FT232RL geht, PL2303 nicht..)

```
# setserial -a /dev/ttyUSB0
/dev/ttyUSB0, Line 0, UART: unknown, Port: 0x0000, IRQ: 0
        Baud_base: 24000000, close_delay: 0, divisor: 0
        closing_wait: infinte
        Flags: spd_normal low_latency
```

Taktfrequenz (baud_base) geteilt durch gewuenschte BPS ergibt einen Divisor:

24.000.000 / 74880 = 320 (circa...)

Mit `spd_cust` kann man den divisor setzen, der an Stelle(!) von 38400bps verwendet wird..

Serielle Kommunikation mit 74880bps

```
# setserial  /dev/ttyUSB0 spd_cust divisor 320
# setserial -a /dev/ttyUSB0
/dev/ttyUSB0, Line 0, UART: unknown, Port: 0x0000, IRQ: 0
        Baud_base: 24000000, close_delay: 0, divisor: 320
        closing_wait: infinte
        Flags: spd_cust low_latency
# screen /dev/ttyUSB0 38400          (minicom geht auch)
```

... und dann das Board resetten...

UART-Boot-Modus (zum Flashen):

```
ets Jan  8 2013,rst cause:2, boot mode:(1,7)
```

Flash-Boot-Modus (Normalbetrieb):

```
 ets Jan  8 2013,rst cause:2, boot mode:(3,6)

load 0x40100000, len 23616, room 16
tail 0
chksum 0x0d
load 0x3ffe8000, len 2636, room 8
tail 4
chksum 0x4a
load 0x3ffe8a50, len 3328, room 4
tail 12
chksum 0x41
csum 0x41

SDK ver: 0.9.3 compiled @ Nov 21 2014 11:15:48
phy ver: 273, pp ver: 5

mode : softAP(1a:fe:34:9c:68:05)
dhcp server start:(ip:192.168.4.1,mask:255.255.255.0,gw:192.168.4.1)
add if1
bcn 100
```

Flashen der Firmware

- Setzen von UART-Boot. (`IO0JP/GPIO0` auf 0/GND).
- Reset des Board (Pin 13 auf GND)
- Flashen des Blinky-Beispiels:

```
cd /opt/Espressif/source-code-examples/blinky
make ESPPORT=/dev/ttyUSB0 flash
```

Flashen der Firmware

```
Connecting...
Erasing flash...
Writing at 0x00007000... (100 %)
Erasing flash...
Writing at 0x00061000... (100 %)

Leaving...
```

Flashen der Firmware

Das IoT-Example:

```
cd /opt/Espressif/ESP8266_SDK/IoT_Demo

cd .output/eagle/debug/image

/opt/Espressif/esptool-py/esptool.py --port /dev/ttyUSB0
write_flash 0x00000 eagle.app.v6.flash.bin 0x40000
eagle.app.v6.irom0text.bin
```