

# WLAN



die hier vorgestellten Konfig-Dateien enthalten noch keine Sicherheitsparameter (z.B. WLAN-Verschlüsselung/Authentifikation bei HostAPd), sie dienen lediglich zum schnellen Test

## intern

in Kernel 4.4.70 ist der Wlan-Code vorhanden, muss aber separat aktiviert werden

[GitHub Forum](#)

[Patch Patch #2](#)



wpa\_supplicant muss deinstalliert, hostapd+dnsmasq installiert werden:

```
apt-get remove wpa_supplicant
apt-get install hostapd dnsmasq
```

hostapd.conf:

```
hw_mode=g
interface=ap0
driver=nl80211
channel=1
auth_algs=1
ssid=test
```

cfg nach /system/etc/firmware/

Hilfsprogramme für die nächsten Schritte (entpacken nach /usr/bin)

und

Firmware (entpacken /etc/firmware/)

von [hier](#)

1. wmt\_loader
2. stp\_uart\_launcher -p /etc/firmware &
3. Treibermodul laden (wenn als Modul 5.4+ compiliert): modprobe wlan\_gen2
4. ( echo 0 >/dev/wmtWifi (zurücksetzen/initialisieren) )
5. echo A >/dev/wmtWifi (AP-Modus aktivieren)

beim letzten Schritt wird das AP-Gerät (Accesspoint) erzeugt, welches dann von hostapd genutzt wird

```
[14:14] root@bpi-r2:~# ifconfig -a|grep Link
```

```
ap0      Link encap:Ethernet  HWaddr 02:08:22:68:39:ff
bond0    Link encap:Ethernet  HWaddr e2:7c:e0:71:31:c1
eth0     Link encap:Ethernet  HWaddr 08:00:00:00:00:00
         inet6 addr: fe80::a00:ff:fe00:0/64 Scope:Link
eth1     Link encap:Ethernet  HWaddr 08:00:00:00:00:01
         inet6 addr: fe80::a00:ff:fe00:1/64 Scope:Link
lo       Link encap:Local Loopback
sit0     Link encap:IPv6-in-IPv4
tunl0    Link encap:IPIP Tunnel  HWaddr
wlan1    Link encap:Ethernet  HWaddr 00:08:22:68:39:ff
```

hostapd starten

```
hostapd -dd /etc/hostapd/hostapd.conf
```

nun kann mit der Schnittstelle weitergearbeitet werden:

```
ip addr add 192.168.10.1/24 dev ap0
#ip link set dev ap0 up
service dnsmasq start
```

hostapd.conf

altes Shell-Script für das Starten  
wifi.sh

## Kernel 4.14

Code vom Kernel 4.4.70 habe ich in [mein github-repo](#) eingebunden.

Diskussion hier (EN): [forum](#)

## bekannte Probleme

### Zufallszahlen

direkt nach dem (Re-)boot ist der Zufallszahlengenerator noch nicht ausreichend gefüllt, so dass Verbindungsversuche abgelehnt werden, bis dieser gefüllt ist.

in der hostapd-log sieht das so aus:

```
random: Cannot read from /dev/random: Resource temporarily unavailable
random: Got 0/14 bytes from /dev/random
random: Only 6/20 bytes of strong random data available from /dev/random
random: Not enough entropy pool available for secure operations
WPA: Not enough entropy in random pool to proceed - reject first 4-way
handshake
```

```
...
WPA: Reject 4-way handshake to collect more entropy for random number
generation
random: Mark internal entropy pool to be ready (count=1/2)
...
random: Cannot read from /dev/random: Resource temporarily unavailable
random: Got 0/14 bytes from /dev/random
random: Only 6/20 bytes of strong random data available from /dev/random
random: Allow operation to proceed based on internal entropy
```

<http://forum.banana-pi.org/t/bpi-r2-new-image-release-ubuntu-16-04-v1-3-2018-3-30/5293/25>

```
apt-get install rng-tools
echo 'HRNGDEVICE=/dev/urandom' >> /etc/default/rng-tools
```

## extern

### MT76



[mt7612e auf AliExpress](#)

### 4.4.70

[forum](#)

```
git clone https://github.com/BPI-SINOVOIP/BPI-R2-bsp.git bpi_r2_mt76
cd bpi_r2_mt76/
cd linux-mt/drivers/net/wireless/mediatek
git clone https://github.com/dfiloni/mt76.git
cd ../../../../ #bpi_r2_mt76/linux-mt/
patch -p1 < drivers/net/wireless/mediatek/mt76/kernel-patches/0001-add-
basic-register-field-manipulation-macros.patch
nano drivers/net/wireless/mediatek/Makefile
#add: obj-$(CONFIG_MT76) += mt76/
nano drivers/net/wireless/mediatek/Kconfig
#add before endif: before endif # WL_MEDIATEK: source
"drivers/net/wireless/mediatek/mt76/Kconfig"
cd ..
./build.sh => 4
#networking support => wireless => <M>   Generic IEEE 802.11 Networking
Stack (mac80211)
#Device Drivers => Network device support => Wireless LAN => [*]   Mediatek
Wireless LAN support => <M>   MediaTek MT76x2 802.11ac chips support
./build.sh => 1
```

```
cp SD/BPI-B00T/bananapi/bpi-r2/linux/uImage /media/$USER/BPI-B00T/bananapi/bpi-r2/linux/uImage
sudo cp -r SD/BPI-R00T/lib/modules /media/$USER/BPI-R00T/lib/
sudo cp linux-mt/drivers/net/wireless/mediatek/mt76/firmware/*
/media/$USER/BPI-R00T/lib/firmware/
#scp linux-mt/drivers/net/wireless/mediatek/mt76/firmware/*
root@192.168.0.10:/lib/firmware/
sync
```

## 4.14

Kernel 4.14+ (in Arbeit...):

bei folgendem Fehler (dmesg):

```
mt76x2e: probe of 0000:01:00.0 failed with error -2
```

muss Firmware-Paket installiert werden:

```
apt-get install firmware-misc-nonfree
```

falls das abbricht muss in der /etc/apt/sources.list „non-free“ hinter „main“ ergänzt werden und „apt update“ ausgeführt werden

### PCle-patch

importieren, wenn noch nicht geschehen

```
patch -p1 < pcie.patch
cd drivers/net/wireless/mediatek/
git clone https://github.com/openwrt/mt76.git
```

- in der mt76/mt7603.h fehlt „#include <linux/interrupt.h>“
- in der mt76/mac80211.c fehlt „#include <linux/of.h>“
- im Makefile fehlen „CFLAGS\_trace.o := -I\$(src)“ und „CFLAGS\_mt76x2\_trace.o := -I\$(src)“
- und halt in drivers/net/wireless/mediatek/Makefile

```
obj-$(CONFIG_MT76) += mt76/
```

und drivers/net/wireless/mediatek/Kconfig

```
source "drivers/net/wireless/mediatek/mt76/Kconfig"
```

einbinden

fertiger Treiber für mt76x2 + mt76x3

nach drivers/net/wireless/mediatek/ entpacken

folgende module im kernel aktivieren:

```
CONFIG_MAC80211=m
CONFIG_CFG80211=m
CONFIG_MT76=m

#pcie
CONFIG_PCIEPORTBUS=y
CONFIG_PCIE_MEDIATEK=y
CONFIG_PHY_MTK_TPHY=y
```

die firmware kopieren

```
sudo cp drivers/net/wireless/mediatek/mt76/firmware/* /media/$USER/BPI-
ROOT/lib/firmware/
```

## einrichten

```
[10:50] root@bpi-r2:~# ifconfig -a |grep wlan
wlan1      Link encap:Ethernet  HWaddr f8:62:aa:50:12:1d  <<<
```

Wenn die wlan-Nummer größer 1, entsprechend anpassen

```
nano /etc/udev/rules.d/70-persistent-net.rules
```

/etc/hostapd/hostapd\_wlan1.conf (ggf. interface ändern):

```
interface=wlan1
#interface=ap0
driver=nl80211

ssid=r2_AP

hw_mode=g
channel=1
#macaddr_acl=0
auth_algs=1
#ignore_broadcast_ssid=0
#wpa=2
#wmm_enabled=1
#wpa_passphrase=12345678
#wpa_key_mgmt=WPA-PSK
#wpa_pairwise=TKIP
#rsn_pairwise=CCMP
```

/etc/hostapd/hostapd\_wlan1.conf

hostapd starten (Debugmode):

```
hostapd -dd /etc/hostapd/hostapd.conf
```

IP-Adresse setzen:

```
ip addr add 192.168.11.1/24 dev wlan1
```

/etc/dnsmasq.conf (zeile aktivieren = # am Anfang entfernen)

```
conf-dir=/etc/dnsmasq.d
```

/etc/dnsmasq.d/interfaces.conf

```
#interface=eth0
interface=wlan0
#interface=eth1
interface=ap0

# DHCP-Server nicht aktiv für Interface
#no-dhcp-interface=ppp0
no-dhcp-interface=eth0
no-dhcp-interface=eth1

#dhcp-authoritative
dhcp-range=ap0,192.168.10.100,192.168.10.150,255.255.255.0,48h
dhcp-option=ap0,3,192.168.10.1
dhcp-range=wlan1,192.168.11.100,192.168.11.150,255.255.255.0,48h
dhcp-option=wlan1,3,192.168.11.1
```

/etc/dnsmasq.d/interfaces.conf

## HostAPd

/etc/hostapd/hostapd.conf

/etc/hostapd/hostapd\_wlan1.conf

Vif: <https://github.com/openwrt/mt76/issues/433>

## 5GHz

5GHz

```
apt-get install iw wireless-regdb crda
```

## country code

den Country-Code (regularly domain) zu setzen kann bisschen tricky sein

```
iw reg set ISO_3166-1_alpha-2  
iw reg set DE  
iw reg get
```

falsche Ausgabe:

```
global  
country 00: DFS-UNSET
```

richtig:

```
global  
country DE: DFS-ETSI
```

cfg80211 als Modul (5.4,5.10,5.12+ wegen mt6625 treiber):

```
$ sudo nano /etc/modprobe.d/cfg80211.conf  
options cfg80211 ieee80211_regdom=DE
```

ggf. manuell laden

```
modprobe cfg80211 ieee80211_regdom=DE
```

evtl. probieren:

```
COUNTRY=DE crda
```

bricht bei mir aber mit „Failed to set regulatory domain: -7“ ab, kann aber das 5GHz Band mit hostapd nutzen. Der letzte Schritt ist also nicht nötig

## mögliche Frequenzen

```
iw list | grep MHz
```

## Hostapd-Konfiguration

```
$ sudo nano /etc/hostapd/hostapd.conf  
[...]  
country_code=DE  
ieee80211n=1  
ieee80211d=1  
hw_mode=a  
channel=48
```

[ ... ]

- <https://raspberrypi.stackexchange.com/a/112048>
- <https://askubuntu.com/a/1276635>
- <https://github.com/openwrt/mt76/issues/433>

## IP-Konfiguration

IP-Adresse setzen:

```
ip addr add 192.168.10.1/24 dev ap0
ip addr add 192.168.11.1/24 dev wlan1
```

/etc/dnsmasq.conf (zeile aktivieren = # am Anfang entfernen)

```
conf-dir=/etc/dnsmasq.d
```

/etc/dnsmasq.d/interfaces.conf

```
#interface=eth0
interface=wlan0
#interface=eth1
interface=ap0

# DHCP-Server nicht aktiv für Interface
#no-dhcp-interface=ppp0
no-dhcp-interface=eth0
no-dhcp-interface=eth1

#dhcp-authoritative
dhcp-range=ap0,192.168.10.100,192.168.10.150,255.255.255.0,48h
dhcp-option=ap0,3,192.168.10.1
dhcp-range=wlan1,192.168.11.100,192.168.11.150,255.255.255.0,48h
dhcp-option=wlan1,3,192.168.11.1
```

/etc/dnsmasq.d/interfaces.conf

## Routing

```
nano /etc/sysctl.conf
#net.ipv4.ip_forward=1 und net.ipv6.conf.all.forwarding=1 aktivieren durch
entfernen der Raute am Zeilenanfang
sysctl -p /etc/sysctl.conf
```

Voraussetzung ist, dass der Router (wenn es nicht der BPI-R2 selbst ist) das/die WLAN-Netz(e) auch kennt und weiß wohin er die Pakete schicken muss.

Die folgenden beiden Befehle müssen im (Debian-)Router ausgeführt werden, um die Routing-Tabelle



entsprechend zu erweitern (gehen beim reboot verloren, wenn nicht beim Systemstart ausgeführt):

```
route add -net 192.168.10.0 netmask 255.255.255.0 gw 192.168.0.10  
route add -net 192.168.11.0 netmask 255.255.255.0 gw 192.168.0.10
```

dabei ist 192.168.10.0 das Netz des 1. WLAN, 192.168.11.0 das Netz des 2. WLAN und 192.168.0.10 die LAN-IP des BPI-R2 (gleiches netz wie die LAN-IP des Routers)

From:

<http://fw-web.de/dokuwiki/> - **FW-WEB Wiki**

Permanent link:

<http://fw-web.de/dokuwiki/doku.php?id=bpi-r2:wlan>

Last update: **2023/06/08 17:06**

