BananaPi R2

- Hardware
  - Debug-UART
  - Anschalten (10 Sekunden Taster drücken)
  - zusätzliche Hardware:
    - Netzteil: 12V/2A empfohlen,Stecker Koax: 2,1 x 5,5 mm Reichelt
    - microSD-Karte (>=8GB): Sandisk Ultra microSDXC 32GB, Samsung microSDHC 8GB
    - WLAN-Antennen: 5dBi mit rp-SMA auf IPEX/u-FL Kabel
    - PCIe Wlan-Karte: Forum ⇒ MT76
  - emmc: Forum
- Software
  - Betriebssystem-Abbilder:
    - offiziell:GoogleDrive dev.banana-pi.org.cn
    - meine (Debian/Ubuntu): GoogleDrive
    - Debian / Ubuntu
    - LEDE / OpenWRT
    - Ubuntu forum
    - CentOS Forum bootstrap
  - Kernel/Uboot (debian):
    - 4.4.70 auf offiziellem GitHub
    - 4.14+ auf meinem GitHub
    - offizieller 4.14
    - kompilierte Kernel auf meinem GDrive oder auf github (Releases)
    - aktuelles uboot auch auf Github
  - Netzwerkeinstellungen
  - GPIO
  - WLAN / Bluetooth
  - HW-NAT 4.9/4.14 / HW-NAT in NF-Tables 5.12+
  - VLAN-Support
  - CryptoDev
  - HDMI
- Hilfe / Dokumentation:
  - Gitbook: BPI-R2 GitBook
  - Schaltungs-Schema: GoogleDrive
  - Datenblatt v1.0 v1.1 (GoogleDrive)
  - Forum EN DE
- Links

bekannte Probleme

- on-board-WLAN aktuell beta (AP-Modus funktioniert soweit bis 5.1)
- Bluetooth funktioniert soweit (4.14,4.19) als beta
- BPi-R2 schaltet nach shutdown nicht ab: GitHub,funktioniert mit 4.14,4.19 und 5.0
- Switch 7530 Porttrennung/vlan wird nicht unterstützt von 4.4.x GitHub Forum, in 4.14+ Porttrennung/vlan funktionsfähig
- 2.GMAC funktioniert in 4.14 + 4.19

Bluetooth

R2-Bluetooth

Bluetooth scheint im 4.4er Kernel zu funktionieren

http://forum.banana-pi.org/t/bpi-r2-new-image-ubuntu-16-04-v1-2-1-bt-and-wifi-ap-mode-are-working-fine-2017-11-27/4291

http://forum.banana-pi.org/t/bpi-r2-kernel-bluetouh-module/4592

im 4.19+ funktioniert es auch. vorausgesetzt, es wird erst der combo-chip initialisiert und danach erst das bluetooth-modul geladen:

http://forum.banana-pi.org/t/bpi-r2-kernel-bluetooth-module/4592/63

```
apt install bluez
root@bpi-r2:~# bluetoothctl
Agent registered
[bluetooth]# agent on
Agent is already registered
[bluetooth]# power on
Changing power on succeeded
[bluetooth]# discoverable yes
Changing discoverable on succeeded
[CHG] Controller 00:00:46:85:90:01 Discoverable: yes
[bluetooth]# scan on
Discovery started
...
[CHG] Controller 00:00:46:85:90:01 Discoverable: yes
[CHG] Device 04:B4:XX:XX:XX:XX RSSI: -29
[CHG] Device 04:B4:XX:XX:XX:XX RSSI: -48
[CHG] Device 04:B4:XX:XX:XX:XX Connected: yes
```

serielle Console über BT

habe dazu ein HC05-Modul (DSD-Tech), welches ich erst einstellen musste (name,pin, baudrate)

dieser Teil war bisschen tricky und man musste bei dem Modul aufpassen, dass nur 1 Zeilenendezeichen geschickt wird.

Das Modul muss über die Pins an einen Usb2Serial angeschlossen werden, habe dazu den genommen, den ich für die serielle Console schon verwende (einfach nur RX/TX vom bpi-r2 auf den BT-Adapter zusätzlich VCC+GND). Das Modul musste erst in den Command-Mode gebracht werden. Dazu have ich den 3v3-pin meines usb2Serial auf den en-Pin des Modules gelegt. Hier ist es wichtig, dass erst die 5V VCC anliegen (und das Modul hochfährt) und danach erst der EN-Pin verkabelt wird . dann kann man die serielle Console auf das BT-Modul mit 9600 Baud starten.

Pin habe ich noch alt gelassen...

```
AT+NAME=Name
AT+UART=115200,0,0
#AT+PIN=1234
#oder
#AT+PSWD=1234
```

danach verkabelt (VCC an gpio pin 4, gnd an gpio pin 6), tx auf rx und rx auf tx des debug-Ports

```
$ bluetoothctl
[NEW] Controller DC:85:DE:91:3A:42 frank-N56VZ [default]
[NEW] Device 72:DC:5C:46:62:60 Primo 413 by Doro
[NEW] Device 7A:32:F8:33:AB:43 7A-32-F8-33-AB-43
[NEW] Device 00:14:03:05:08:AE BPI-R2
[NEW] Device 9C:8C:6E:4A:F8:23 [TV] Samsung 8 Series (65)
Agent registered
[bluetooth]# pair 00:14:03:05:08:AE
Attempting to pair with 00:14:03:05:08:AE
[CHG] Device 00:14:03:05:08:AE Connected: yes
Request PIN code
[BPI-1m[agent] Enter PIN code: 1234
[CHG] Device 00:14:03:05:08:AE UUIDs: 00001101-0000-1000-8000-00805f9b34fb
[CHG] Device 00:14:03:05:08:AE ServicesResolved: yes
[CHG] Device 00:14:03:05:08:AE Paired: yes
Pairing successful
```

zweiter Teil, serielles Gerät anlegen:

```
sudo rfcomm connect /dev/hci0 00:14:03:05:08:AE
[sudo] Passwort für frank:
Connected /dev/rfcomm0 to 00:14:03:05:08:AE on channel 1
Press CTRL-C for hangup
```

dritter teil, zugriff mit minicom

```
sudo minicom -D /dev/rfcomm0
```

http://stefanfrings.de/bluetooth/

Cryptodev

http://forum.banana-pi.org/t/is-it-possible-to-have-the-crypto-extensions-working/4034/11

in Main-Branch des 4.14-Kernel integriert: https://github.com/frank-w/BPI-R2-4.14/tree/main

Debian

debootstrap

https://blog.night-shade.org.uk/2013/12/building-a-pure-debian-armhf-rootfs/

in jessie/Ubuntu 14.x (64bit) gibt es Probleme in der second-stage: link, hier muss ggf. debootstrap geupdated werden (erste Stufe muss erneut durchgeführt werden): download

```
sudo apt-get install qemu-user-static debootstrap binfmt-support
distro=buster
arch=armhf
#r64: arch=arm64
targetdir=$(pwd)/debian_${distro}_${arch}
mkdir $targetdir
sudo debootstrap --arch=$arch --foreign $distro $targetdir
sudo cp /usr/bin/qemu-arm-static $targetdir/usr/bin/
#r64:sudo cp /usr/bin/qemu-aarch64-static $targetdir/usr/bin/
sudo cp /etc/resolv.conf $targetdir/etc
sudo distro=$distro chroot $targetdir
export LANG=C
/debootstrap/debootstrap --second-stage
```

ggf. gleich root-pw setzen und/oder neue User anlegen (sudo nicht vergessen). sonst kein login (seriell/ssh) in das System möglich.

Man kann auch vom hostsystem Befehle in der chroot ausführen.

```
chroot rootordner/ ./chroot.sh
```

chroot verlassen (exit oder strg+D)

ggf. Packen:

```
#sudo umount $targetdir/proc
sudo tar -czf ${distro}_${arch}.tar.gz -C $targetdir $targetdir
```

wie in der Quelle noch hostname,apt (ggf. De-Server),... einrichten

- hostname (/etc/hostname)

```
bpi-r2
```

- /etc/apt/sources.list

```
deb http://ftp.de.debian.org/debian $distro main contrib non-free
deb-src http://ftp.de.debian.org/debian $distro main contrib non-free
```

```
deb http://ftp.de.debian.org/debian $distro-updates main contrib non-free
deb-src http://ftp.de.debian.org/debian $distro-updates main contrib non-
free
deb http://security.debian.org/debian-security $distro/updates main contrib
non-free
deb-src http://security.debian.org/debian-security $distro/updates main
contrib non-free
```

- fstab (boot,root)

```
# <file system>     <dir>       <type>  <options>       <dump>  <pass>
/dev/mmcblk0p2      /       ext4    errors=remount-ro   0       1
/dev/mmcblk0p1      /boot       vfat    defaults        0       0
```

- Dns-server in /etc/resolv.conf eintragen

```
nameserver 192.168.0.5
```

- Netzwerk-Konfiguration

```
auto eth0
iface eth0 inet manual
  pre-up ip link set $IFACE up
  post-down ip link set $IFACE down

auto lan0
iface lan0 inet static
  hwaddress ether 08:00:00:00:00:00 # if you want to set MAC manually
  address 192.168.0.10
  netmask 255.255.255.0
  gateway 192.168.0.5
  pre-up ip link set $IFACE up
  post-down ip link set $IFACE down
...
```

im bestehenden Jessie-Image die root-Partition (außer lib/modules/<kernelversion>) leeren und den Inhalt des Bootstrap-ordners (debian_stretch) dorthin kopieren

fertiges bootstrapped debian stretch (sd+emmc in separaten img's): gdrive

damit der root-login via SSH funktioniert muss noch in der /etc/ssh/sshd_config folgendes hinzugefügt werden:

```
PermitRootLogin yes
```

sowie der ssh-server neu gestartet werden mit

```
service sshd restart
```

lauffähiges Image anpassen

- logs leeren (echo -n „“>logdatei),
- backup-dateien löschen,
- cache (z.B. APT) löschen,
- nicht benötigte Kernel-Module (/lib/modules/)
- ggf. eigene User löschen
- PW für root zurücksetzen

freien Speicher im Image mit null-datei füllen (besseres packen):

```
loopdev=$(losetup -f) #erstes freies loopdevice
losetup ${loopdev} /path/to/file
partprobe ${loopdev}
mount ${loopdev}p2 /mnt

#wenn man noch Sachen installieren will bzw. Updates machen
cp /usr/bin/qemu-arm-static /mnt/usr/bin/
chroot /mnt

#bei apt-update-fehler "Couldn't create temporary file /tmp/apt.conf.xxxxx
for passing config to apt-key"
mount -t tmpfs none /tmp

#freien Speicherplatz überschreiben
dd if=/dev/zero of=/mnt/null.dat
rm /mnt/null.dat

#uboot-update r2
dd of=${loopdev} if=u-boot.bin bs=1k seek=320;

losetup -d ${loopdev}
```

https://softwarebakery.com/shrinking-images-on-linux

```
myimage=myimage.img
size=$(fdisk -l $myimage | tail -1 | awk '{print $3}')
truncate --size=$[($size+1)*512] $myimage
```

alternative https://wiki.debian.org/DebianInstaller/Arm/OtherPlatforms

NTP

```
apt-get install ntpdate
```

```
service ntp stop
ntpdate -s ptbtime1.ptb.de
```

```
service ntp start
```

Netzwerk

alles unter 4.14 in der /etc/network/interfaces:

normale Konfiguration

```
auto eth0
iface eth0 inet manual
  pre-up ip link set $IFACE up
  post-down ip link set $IFACE down
auto lan0
iface lan0 inet static
  hwaddress ether 08:00:00:00:00:00 # if you want to set MAC manually
  address 192.168.0.10
  netmask 255.255.255.0
  gateway 192.168.0.5
  pre-up ip link set $IFACE up
  post-down ip link set $IFACE down

auto lan1
iface lan1 inet static
  hwaddress ether 08:00:00:00:00:01 # if you want to set MAC manually
  address 192.168.1.10
  netmask 255.255.255.0
  pre-up ip link set $IFACE up
  post-down ip link set $IFACE down

auto lan2
iface lan2 inet static
  hwaddress ether 08:00:00:00:00:02 # if you want to set MAC manually
  #...

auto lan3
iface lan3 inet static
  hwaddress ether 08:00:00:00:00:03 # if you want to set MAC manually
  #...

auto wan
iface wan inet static
  hwaddress ether 09:00:00:00:00:01 # if you want to set MAC manually
  #...
```

unter debian 9 funktioniert hwaddress nicht mehr, hier lässt sich das setzen der MAC so erreichen:

```
iface lan0 inet static
  address 192.168.0.10
```

```
  netmask 255.255.255.0
  gateway 192.168.0.5
#  pre-up ip link set $IFACE up
  pre-up ip link set $IFACE address 02:01:02:03:04:08 up
  post-down ip link set $IFACE down
```

Netzwerkbrücke

```
apt-get install bridge-utils
```

```
iface br0 inet static
  address 192.168.40.1
  netmask 255.255.255.0
  bridge_ports lan1 lan2
  bridge_fd 5
  bridge_stp no
```

vlan

```
auto lan3
iface lan3 inet manual

auto lan3.60
iface lan3.60 inet static
  address 192.168.60.10
  netmask 255.255.255.0
#  gateway 192.168.0.5
  pre-up ip link set $IFACE address 02:01:02:03:04:03 up #setting mac does
not work currently
```

BananaPi R2 - Debug-UART

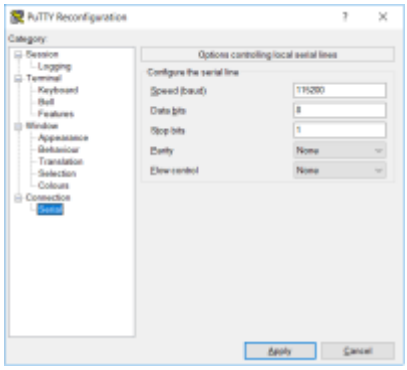
- USB2Serial-Adapter (z.B. CP2102 oder FTDI, Probleme mit Profilic- und ch340g-Chipsätzen bekannt)
- separate Uart-Buchsen
- jeweils TX auf RX
- Programm für PC:
    - Linux: minicom
    - Windows: Putty
- Einstellungen: 115200 8N1 FlowControl: aus
- es kommt nur eine Ausgabe, wenn mindestens der Preloader geladen wird (Schiebeschalter auf SD, wenn EMMC noch leer ist)

micro-usb-cp2102-Adapter:
https://www.ebay.de/itm/CP2102-MICRO-USB-to-UART-TTL-Module-6-Pin-Serial-Converter-STC-Replace-FT232/401269171476

Putty einrichten

Windows-Treiber für CP2102:
https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

cp210x_universal_windows_driver.zip



reg-Datei für die Putty-Sitzung:

bpi-r2-serial-putty-reg.zip

Minicom einrichten

Gerätename ermitteln:

```
grep ttyUSB /var/log/syslog
Oct 15 12:44:59 Frank-Laptop kernel: [ 5113.456306] usb 3-1: cp210x
converter now attached to ttyUSB0
```

```
ls -l /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 0 Mai 10 15:12 /dev/ttyUSB0
#aktuellen Benutzer in die Gruppe dialout (für Zugriff auf das tty-Gerät)
aufnehmen
sudo adduser $USER dialout
#login/logout nötig, damit die Gruppe übernommen wird
#Gruppen anzeigen, in denen der aktuelle Benutzer Mitglied ist
groups
frank adm dialout cdrom sudo dip plugdev lpadmin sambashare
```

Minicom installieren und einrichten:

```
sudo apt-get install minicom
sudo minicom -s
```

```
          +------------[Konfiguration]-------------+
          | Dateinamen und Pfade                   |
          | Protokolle zur Dateiübertragung        |
          | Einstellungen zum seriellen Anschluss  |  <<<<<<<<<<<
          | Modem und Wählverhalten                |
```

```
          | Bildschirm und Tastatur                |
          | Speichern als »dfl«                    |
          | Einstellungen speichern als ...        |
          | Verlassen                              |
          | Minicom beenden                        |
          +----------------------------------------+
```

```
  +---------------------------------------------------------------------+
  | A - Serieller Anschluss      : /dev/ttyUSB0                         |
  | B - Pfad zur Lockdatei       : /var/lock                            |
  | C - Programm zur Rufannahme  :                                      |
  | D - Programm zum Wählen      :                                      |
  | E - Bps/Par/Bits             : 115200 8N1                           |
  | F - Hardware Flow Control    : Nein                                 |
  | G - Software Flow Control    : Nein                                 |
  |                                                                     |
  |    Welchen Parameter möchten Sie ändern?                            |
  +---------------------------------------------------------------------+
```

```
speichern als .dfl
minicom beenden
```

nun minicom als user starten (ohne sudo), ggf. minicom -C boot.log

sollte nun „keine Berechtigung“ oder „Permission denied“ kommen:

```
ls -l /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 0 Apr 27 14:18 /dev/ttyUSB0
sudo adduser $USER dialout
```

dies wird aber erst nach erneutem Anmelden aktiv

Beenden mit Strg+A,q oder Strg+A,x

Farben

```
minicom -c on
```

oder

```
MINICOM='-c on'
export MINICOM
```

Logging

```
minicom -C boot.log
```

GPIO

Grundlagen

https://wiki.openwrt.org/doc/hardware/port.gpio

Pin-Belegung

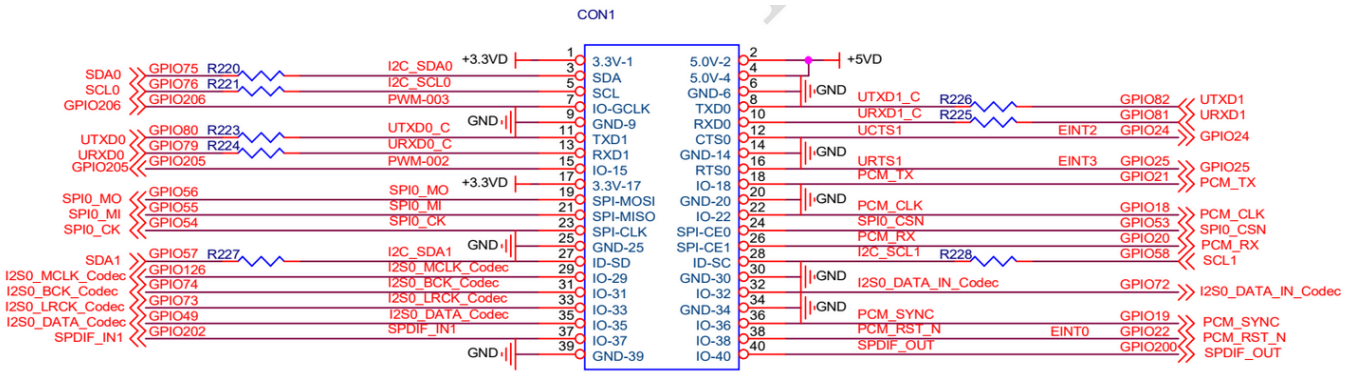
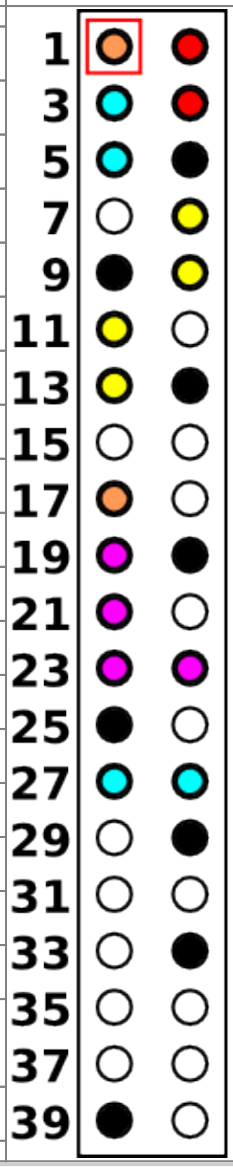


Bild aus den BPI-R2 Schematics

Nebenfunktion	Hauptfunktion	pin#	xxxxxxxxxxxxxxxxxxxx	pin#	Hauptfunktion	Nebenfunktion
-	3V3	1		2	5V	-
I2C_SDA0	GPIO 75	3		4	5V	-
I2C_SCL0	GPIO 76	5		6	GND	-
PWM3	GPIO 206	7		8	GPIO 82	UART1 TX
-	GND	9		10	GPIO 81	UART1 RX
UART0 TX	GPIO 80	11		12	UCTS1 / INT2	GPIO 24 (*)
UART0 RX	GPIO 79	13		14	GND	-
PWM2	GPIO 205	15		16	GPIO 25 / INT3	URTS1
-	3V3	17		18	GPIO 21	PCM_TX
SPI0_MO	GPIO 56	19		20	GND	-
SPI0_MI	GPIO 55	21		22	GPIO 18	PCM_CLK
SPI0_CK	GPIO 54	23		24	GPIO 53	SPI0_CSN
-	GND	25		26	GPIO 20	PCM_RX
I2C_SDA1	GPIO 57	27		28	GPIO 58	I2C_SCL1
	GPIO 126	29		30	GND	-
I2S0_BCK	GPIO 74	31		32	GPIO 72	I2S0_DATA_IN
I2S0_LRCK	GPIO 73 (?)	33		34	GND	-
I2S0_DATA	GPIO 49 (M)	35		36	GPIO 19	PCM_SYNC
SPDIF_IN1	GPIO 202	37		38	INT0	GPIO 22 (*) / PCM_RST_IN
-	GND	39		40	GPIO 200	SPDIF_OUT



	main	spare
Pin 1	3V3	-
Pin 2	5V	-
Pin 3	GPIO 75	I2C_SDA0
Pin 4	5V	-
Pin 5	GPIO 76	I2C SCL0
Pin 6	GND	-
Pin 7	GPIO 206	PWM3
Pin 8	GPIO 82	UART1 TX
Pin 9	GND	-
Pin 10	GPIO 81	UART1 RX
Pin 11	GPIO 80	UART0 TX
Pin 12	Int2	GPIO 24 (*) / UCTS1
Pin 13	GPIO 79	UART0 RX
Pin 14	GND	-
Pin 15	GPIO 205	PWM2
Pin 16	GPIO 25 / Int3	URTS1
Pin 17	3V3	-

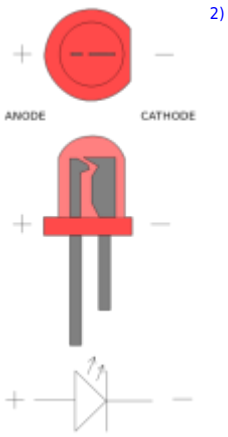
	main	spare
Pin 18	GPIO 21	PCM_TX
Pin 19	GPIO 56	SPI0_MO
Pin 20	GND	-
Pin 21	GPIO 55	SPI0_MI
Pin 22	GPIO 18	PCM_CLK
Pin 23	GPIO 54	SPI0_CK
Pin 24	GPIO 53	SPI0_CSN
Pin 25	GND	-
Pin 26	GPIO 20	PCM_RX
Pin 27	GPIO 57	I2C_SDA1
Pin 28	GPIO 58	I2C_SCL1
Pin 29	GPIO 126	
Pin 30	GND	-
Pin 31	GPIO 74	I2S0_BCK
Pin 32	GPIO 72	I2S0_DATA_IN
Pin 33	GPIO 73 (?)	I2S0_LRCK
Pin 34	GND	-
Pin 35	GPIO 49 (M)	I2S0_DATA
Pin 36	GPIO 19	PCM_SYNC
Pin 37	GPIO 202	SPDIF_IN1
Pin 38	INT0	GPIO 22 (*) / PCM_RST_IN
Pin 39	GND	-
Pin 40	GPIO 200	SPDIF_OUT

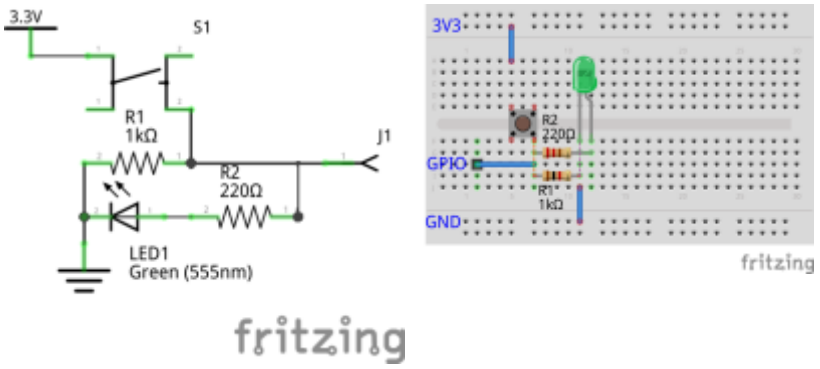
```
(*) Spezial-GPIO benötigen Speicher-Patch und Mode-Einstellung (4.4.70)
(?) GPIO getested, noch nicht funktionsfähig
(M) Mode=0 erforderlich
```

Schaltungen

1)

2)





Fritzing-Datei

https://docs.labs.mediatek.com/resource/linkit7697-arduino/en/tutorial/smd-buttons

Kernel 4.4.70

Standard GPIO

```
root@bpi-iot-ros-ai:~# GPIO=/sys/devices/platform/1000b000.pinctrl/mt_gpio
root@bpi-r2:~# echo "mode 25 0" >$GPIO #nicht immer nötig
root@bpi-r2:~# echo "dir 25 1" >$GPIO
root@bpi-r2:~# echo "out 25 1" >$GPIO
```

funktioniert mit LED an Pin 14 (-) und Pin 16 (+), inkl. Vorwiderstand (220 Ohm)

Spezial GPIO

für die GPIOs 22/(23??)/24 ist es nötig, vorher ein Register zu setzen (siehe issue#17 Kommentar #15)

mwrite

```
root@bpi-iot-ros-ai:~# ./mwrite /dev/mem 0x10005b10 0x00000038
./mwrite offset : 10005b10, val : 00000038
b6f03b10
root@bpi-r2:~# GPIO=/sys/devices/platform/1000b000.pinctrl/mt_gpio
root@bpi-r2:~# echo "mode 24 0" >$GPIO
root@bpi-r2:~# echo "dir 24 1" >$GPIO
root@bpi-r2:~# echo "out 24 1" >$GPIO
```

zusätzlich musste ich beim GPIO24 (pin 12) den mode noch auf 0 setzen

Kernel 4.14

GPIO_SYSFS und CONFIG_DEBUG_GPIO müssen in Kernel-Config gesetzt sein (.config)

Standard GPIO

```
root@bpi-r2# mount -t debugfs none /sys/kernel/debug
root@bpi-r2# cat /sys/kernel/debug/pinctrl/1000b000.pinctrl/gpio-ranges
GPIO ranges handled:
0: 1000b000.pinctrl GPIOS [232 - 511] PINS [0 - 279]
root@bpi-r2# GPIO_NO=$((232+25))
root@bpi-r2# echo $GPIO_NO
257
root@bpi-r2# echo $GPIO_NO > /sys/class/gpio/export
```

Pin 14=GND/16=GPIO25

GPIO als Ausgang

```
root@bpi-r2# echo out > /sys/class/gpio/gpio${GPIO_NO}/direction
root@bpi-r2# echo 1 > /sys/class/gpio/gpio${GPIO_NO}/value
root@bpi-r2# echo 0 > /sys/class/gpio/gpio${GPIO_NO}/value
```

schaltet LED (inkl. Vorwiderstand) an Pin 14=GND/16=GPIO25 ein (1) und wieder aus (0)

GPIO als Eingang

high-active Taster-Schaltung an GPIO 200 (Pin 40 zwischen Schalter und Widerstand, Pin 39 als GND [Widerstand] und Pin 17 als 3v3-vcc)

```
[10:54] root@bpi-r2:~# echo in > /sys/class/gpio/gpio${GPIO_NO}/direction
[10:56] root@bpi-r2:~# cat /sys/class/gpio/gpio${GPIO_NO}/value
0 #Taster nicht gedrückt
[10:56] root@bpi-r2:~# cat /sys/class/gpio/gpio${GPIO_NO}/value
1 #Taster gedrückt
[10:56] root@bpi-r2:~# cat /sys/class/gpio/gpio${GPIO_NO}/value
0 #Taster nicht gedrückt

#dauerhaft alle 1/4s abfragen
watch -n 0.25 cat /sys/class/gpio/gpio${GPIO_NO}/value
```

Special GPIO

Speicher-hack (wie in 4.4.70) nicht notwendig

Beispiel für GPIO24 (pin12):

```
root@bpi-r2# GPIO_NO=$((232+24))
root@bpi-r2# echo $GPIO_NO > /sys/class/gpio/export
root@bpi-r2# echo out > /sys/class/gpio/gpio${GPIO_NO}/direction
```

```
root@bpi-r2# echo 1 > /sys/class/gpio/gpio${GPIO_NO}/value
```

LED geht an :)

on-board LEDs

Die On-Board-LEDs welche hier angesteuert werden befinden sich nahe der Netzteil-Buchse (nicht neben der GPIO-Leiste)

http://forum.banana-pi.org/t/control-on-board-leds/4287/13

an ⇒

```
echo 1 > /sys/class/leds/bpi-r2:isink:green/brightness
```

aus ⇒

```
echo 0 > /sys/class/leds/bpi-r2:isink:green/brightness
```

blinken (erstellt delay_on/off-knoten zur Frequenz-Kontrolle) ⇒

```
echo timer > /sys/class/leds/bpi-r2:isink:green/trigger
```

ändern der Blink Frequenz (an/aus-Zeit in ms) ⇒

```
echo 100 > /sys/class/leds/bpi-r2:isink:green/delay_on
echo 100 > /sys/class/leds/bpi-r2:isink:green/delay_off
```

in meinen Tests, grün blinkt beim anschalten (rot+blau gehen einfach an/aus), bisher weis ich noch nicht, wie man das Blinken der grünen LED deaktivieren kann

```
L=/sys/class/leds/bpi-r2\:isink
echo 0 > $L:red/brightness          #rot aus
echo 1 > $L:red/brightness          #rot an
echo 0 > $L:green/brightness        #grün aus
echo 1 > $L:green/brightness        #grün blinkt
```

```
[16:08] root@bpi-r2:~# L=/sys/class/leds/bpi-r2\:isink
[17:41] root@bpi-r2:~# L2=/sys/class/leds/bpi-r2\:pio
[17:42] root@bpi-r2:~# echo 1 > $L2:green/brightness
[17:42] root@bpi-r2:~# echo 1 > $L2:blue/brightness
[17:42] root@bpi-r2:~# echo 0 > $L2:green/brightness
[17:42] root@bpi-r2:~# echo 0 > $L2:blue/brightness
```

UART

DTS(i) anpassen

unter Kernel 4.4.x fehlen die DeviceTree-Abschnitte, diese kann man man aber einfach aus einem höheren Kernel nachtragen (dtsi). in der *bpi*.dts bzw. *bananapi*.dts dann auf enabled setzen

definition in der mt7623.dtsi:
http://elixir.free-electrons.com/linux/v4.13-rc7/source/arch/arm/boot/dts/mt7623.dtsi

nun in der bananapi.dts den uart noch auf „status=okay“ setzen

bei uart muss darauf geachtet werden, dass in der mt7623.dtsi erst uart2 und dann die anderen kommen, da sonst nach der uboot-Meldung „Starting Kernel“ keine Ausgabe mehr auf dem Terminal kommt

Uart3 kann auf UCTS2/URTS2 gerouted werden. Diese befinden sich neben dem Anschluss für Debug-UART (hier)

Einstellungen des Ports

```
#Einstellungen des seriellen Ports anzeigen (ersetze ttyS2 mit ttyUSB0 wenn ein USB2serial-Adapter verwendet wird):
stty -F /dev/ttyS2 -a

#Das setzt die Baudrate auf 9600, 8 bits, 1 stop bit, keine parität:
stty -F /dev/ttyS2 9600 cs8 -cstopb -parenb

#verarbeitung deaktivieren (Zeichenkonvertierung,Zeilenumbrüche,...)
stty -F /dev/ttyS2 -opost

#raw Modus
stty -F /dev/ttyS2 raw
```

Nutzung

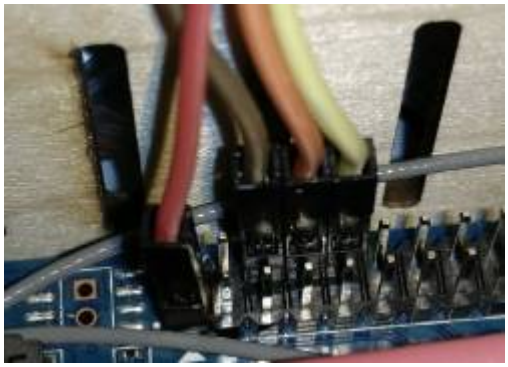
```
pin 8/10 = uart1 (tx/rx) = 11003000
pin 11/13 = uart0 (tx/rx) = 11002000
```

```
#!/bin/bash
DEV=/dev/ttyS2
#stty -F ${DEV} sane
#stty -F ${DEV} 9600 cs8 -cstopb -parenb -crtscts -echo
stty -F ${DEV} 9600 cs8 -cstopb -parenb raw -echo

dmesg | grep "ttyS.*MMIO" | sed 's/^\[.*\] \(\d*.*\) at.*$/\1/'

echo "11002000 = uart0 (tx/rx) = pin 11/13"
echo "11003000 = uart1 (tx/rx) = pin 8/10"
```

```
echo "using $DEV"
echo "send data using \"echo \"AT\" >$DEV\""

while read line; do
#  if [[ -n "$line" ]]; then
    echo "["$(date "+%Y-%m-%d %H:%M:%S")"] received: "$line
#  fi
done < <(cat $DEV)
```

```
echo "AT" >/dev/ttyS2
```

einfaches Beispiel für Arduino (Nano)



```
PI             Levelshifter       Arduino
1 (3V3)  -------  LV   HV -------- 5V
6 (GND)  ---------  GND  --------- GND
8 (TX)   ------- LV2  HV2 -------- RX
10 (RX)  ------- LV1  HV1 -------- TX
```

PWM

Kernel-Option PWM_MEDIATEK muss gesetzt sein (Modul möglich), benötigt PWM(=y)

gpio 206 (pin 7) als pwm3 verwenden

```
echo 3 >/sys/class/pwm/pwmchip0/export
echo 200000 >/sys/class/pwm/pwmchip0/pwm3/period
echo 100000 >/sys/class/pwm/pwmchip0/pwm3/duty_cycle
echo 1 >/sys/class/pwm/pwmchip0/pwm3/enable
```

https://www.kernel.org/doc/Documentation/pwm.txt

```
period The total period of the PWM signal (read/write). Value is in
nanoseconds and is the sum of the active and inactive time of the PWM.
```

```
duty_cycle The active time of the PWM signal (read/write). Value is in
nanoseconds and must be less than the period.
```

```
period=200000ns=200ms=5Hz
duty_cycle=100000ns=1/2 period=50% high + 50% low Signal
```

aktuell ist aber die Ausgangsfrequenz nicht korrekt (statt 5kHz kommt 1kHz raus) siehe Forum und Fehlerreport

seit 2.3.2018 ist die Frequenz richtig: Commit in 4.14-main

SPI

http://forum.banana-pi.org/t/bpi-r2-spi-communication/4779/27

I2C

Echtzeituhr via i2c

https://tutorials-raspberrypi.de/raspberry-pi-rtc-modul-i2c-echtzeituhr/

ggf.

```
apt-get install i2c-tools
```

(benötigt unter ubuntu 18.4 universe in der /etc/apt/sources.list)

```
[17:13] root@bpi-r2:~# modprobe i2c-dev
[17:14] root@bpi-r2:~# i2cdetect -y 0
```

Realtime-Clock DS1307 (mit entfernten pullup-Widerständen) an i2c0 (I2C_SDA0=pin3, I2C_SCL0=pin5, 5V=pin4, GND=pin6)

```
#!/bin/bash
modprobe i2c-dev
modprobe rtc-ds1307
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-0/new_device
#cat /sys/class/i2c-dev/i2c-0/device/0-0068/rtc/rtc0/time
#read rtc
hwclock -r
#set system-clock to rtc-value
#hwclock -s
#set rtc to system-time
#hwclock -w
```

1)

Quelle: wiki.openwrt.org

2)

Quelle: commons.wikimedia.org

Hardware

- Debug-UART
- Anschalten: 10s Power-Taste drücken, ohne Power-Taste: Forum
    - Lötbrücke neben Power-Taster (führt zu boot-loop mit 4.14 durch Poweroff-Patch)
    - 5V auf den OTG-Port anlegen
- emmc: Forum
- WLAN-Chip onBoard: MT6625 Forum (dev) Forum (usr)
- Switch-Chip onBoard: MT7530 Forum
- GPIO

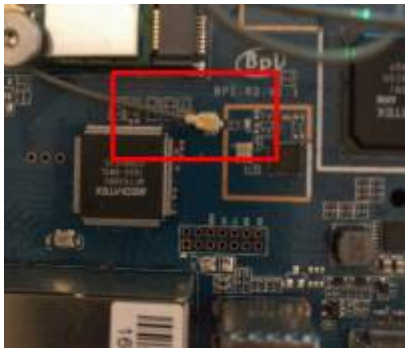
zusätzliche Hardware

- Netzteil: 12V/2A empfohlen,Stecker Koax: 2,1 x 5,5 mm Reichelt
- microSD-Karte (>=8GB): Sandisk Ultra 32GB
- WLAN-Antennen: 5dBi mit rp-SMA auf IPEX/u-FL Kabel
- PCIe Wlan-Karte: Forum ⇒ MT76
- Gehäuse:
    - offiziell: Aluminium-Gehäuse,Acryl Gehäuse
    - inoffiziell: Dateien für 3D Printer Wandmontage
- Kühlkörper:
    - mt7623: 20x20mm
    - mt7530: 14x14mm (wie für rpi soc)

Kabel

WLAN

- rp-SMA auf IPEX/u-FL Kabel für Wlan-Antennen



SATA

- 2,54mm XH 2pin/4pin für SATA Stromversorgung oder 2-Pin kombiniert mit SATA

es funktioniert immer nur der benachbarte GND (1+2,3+4). 12V wäre das gelbe Kabel, 5V das rote

die beiden Buchsen an den SATA-Buchsen führen auch Strom, wenn der R2 „halted“ ist, der Lüfter-Anschluss (beim IR-Empfänger) hat bei meinem Test (4.14) keine Spannung rausgegeben. vermutlich wird ein PWM-Signal ausgegeben (habe leider kein Oszilloskop hier)

Batterie

- ~~1,25mm SH 6pin für Batterie~~ AliExpress~~, noch keine Schaltung bekannt~~ Batterie Anschluss funktioniert nicht Info hier

Befestigungen

- Wlan-mPCIe-Karte festgeschraubt mit M2x10 Schrauben inkl. Rädelmutter + 2xU-Scheibe aus Polyamid (Isolation+Schutz des Boards)

HDMI

HDMI funktioniert mit Kernel 4.16 in den meisten Fällen (1080p,1280×1024). Der Treiber inkl. fbdev wurde nach 4.14 und 4.19+ portiert.

Diskussion im Forum.

Auflösung lässt sich in der BPI-BOOT/bananapi/bpi-r2/linux/uEnv.txt setzen:

```
bootopts=vmalloc=496M debug=7 initcall_debug=0 video=1280x1024-32
```

diese Auflösung wird dann für die virtuelle Konsole und den X-Server verwendet. Der x-server wird installiert, wenn z.B. lxde installiert wird

damit wird die Ausgabe auf einen bestimmten Anschluss festgelegt (Hotplug, Anzeige Muss während des bootens nicht angeschlossen sein):

```
video=HDMI-A-1:1280x1024D
```

leider gibt es aber manchmal Darstellungsprobleme, wenn das Anzeigegerät nachträglich angeschlossen wird

mehr infos zu dieser Einstellung: https://nouveau.freedesktop.org/wiki/KernelModeSetting/

um nur die Schrift auf der Konsole zu vergrößern kann man folgendes tun:

```
setfont Uni3-TerminusBold32x16.psf.gz
```

die Schriften liegen in /usr/share/consolefonts/

für die Framebuffer-console werden zusätzlich diese optionen benötigt:

```
console=tty1 fbcon=map:0
```

das drm-debug kann mit dieser cmdline aktiviert werden:

```
drm.debug=0x7
```

testing

```
for p in /sys/class/drm/*/status; do con=${p%/status}; echo -n
"${con#*/card?-}: "; cat $p; done
HDMI-A-1: connected

cat /sys/class/graphics/fb0/modes
U:1024x768p-0
```

Vielen Dank an Forum-User Alex R. aka „DeadMeat“

Normalerweise wird hdmi deaktiviert wenn kein Monitor angeschlossen (oder ausgeschaltet) ist während des Bootvorgangs. Dies lässt sich mit folgender video-option ändern:

video=HDMI-A-1:1280×1024-32@60D

damit bleibt hdmi aktiviert und man kann hotplug nutzen, aber in meinem Fall gibt es Auflösungsprobleme

http://forum.banana-pi.org/t/bpi-r2-hdmi-in-uboot-and-linux/4651/123

X-Server

Autostart deaktivieren (systemd)

```
systemctl set-default multi-user.target
```

aktivieren:

```
systemctl set-default graphical.target
```

Hardware-NAT

- für LEDE verfügbar forum
- scheint mit 4.14 funktionieren, Test: https://github.com/frank-w/BPI-R2-4.14/tree/4.14-hnat forum
- keine Unterstützung durch 4.4.70 forum

Die aktuelle Implementation arbeitet nur zwischen lan und wan (nicht wlan und anderen Schnittstellen)

NAT auf wan-Schnittstelle aufsetzen:

```
ipt=/sbin/iptables
if_wan=wan
${ipt} -t nat -F
${ipt} -t nat -A POSTROUTING -o ${if_wan} -j MASQUERADE
```

hnat-module laden:

```
modprobe mtkhnat
```

Funktionstests

```
#via debugfs (bei Funktion BIND-status)
cat /sys/kernel/debug/hnat/all_entry
#via interrupts (Zähler erhöht sich langsamer nach einer Weile (download))
cat /proc/interrupts | grep 'ethernet'
```

original-Patch für 4.9 (lede)

Kernel

Übersicht

welchen Kernel nehmen?

Überblick der Kernel-features auf github

offiziellen Kernel kompilieren

4.4.70 auf GitHub

```
sudo apt-get install git make gcc-arm-linux-gnueabihf u-boot-tools
git clone https://github.com/BPI-SINOVOIP/BPI-R2-bsp.git bpi-r2
cd bpi-r2
./build.sh
```

siehe auch Patch für build.sh

- Option 4 für Einstellungen am Kernel,
- 1 um alles zu kompilieren,
- 3 um nur den Kernel zu kompilieren, Option 6 danach, um die kompilierten Daten in den SD-Ordner zu kopieren

wenn alles durchlaufen ist, ist der Kernel im Ordner SD/BPI-BOOT und die Module in SD/BPI-ROOT

der Kernel kann direkt auf die Boot-Partition geschrieben werden (Pfad beachten, vorher Backup!) die Module müssen als Root rüberkopiert werden

```
cp SD/BPI-BOOT/bananapi/bpi-r2/linux/uImage /media/$USER/BPI-
BOOT/bananapi/bpi-r2/linux/
sudo cp -r SD/BPI-ROOT/lib/modules/4.4.70-BPI-R2-Kernel /media/$USER/BPI-
ROOT/lib/modules/
```

meinen 4.4er Kernel kompilieren

4.4.x auf GitHub

```
sudo apt-get install git make gcc-arm-linux-gnueabihf u-boot-tools
git clone https://github.com/frank-w/BPI-R2-4.4.git bpi-r2
cd bpi-r2
./build.sh
```

Installation wie beim offiziellen Kernel

kompilierter Kernel 4.4.x

v1.2.1 (23. Nov 2017),incl. mt76x2/x3 WLAN-Treiber

Bootloader (u-boot):
https://drive.google.com/file/d/11XQ4n6WYSj7gGPtMwug4SITtYn0GZxOb/view?usp=sharing

https://drive.google.com/drive/folders/1kkFyxeHoskszI7CpSsL6Wi6ROjDrBLOB?usp=sharing

das 4.4-Archiv (Kernelversion-BPI-R2-Kernel_boot+root.tgz) enthält 2 Ordner, deren Inhalt auf die SD/MMC mit dem selben Namen kopiert werden müssen.

- BPI-BOOT ⇒ boot-Partition, uImage=Kernel-file, backup first your existing uImage before overwriting it
- BPI-ROOT ⇒ partition mit dem Betriebssystem, hier liegen die Kernel-Module (/lib/modules/kernelname). Mit jedem neuen Kernel wird ein neuer Ordner angelegt, so dass ein Backup nicht unbedingt nötig ist, außer es ist die selbe Kernelversion.

eigenen (neueren) Kernel verwenden

http://forum.banana-pi.org/t/what-s-the-best-practice-to-build-own-kernel/3937

Patches, die noch nicht im Mainline-Kernel sind:
https://patchwork.kernel.org/project/linux-mediatek/list/

mein github-repo verwenden

https://github.com/frank-w/BPI-R2-4.14.git

standard-kernel-Repo verwenden + patches

defconfig von GitHub nach arch/arm/configs/ kopieren

defconfig von GitHub importiert und angepasst

Dateindung .txt entfernen

```
  sudo apt-get install git make gcc-arm-linux-gnueabihf u-boot-tools
  git clone
https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git bpi-r2
  cd bpi-r2

  export CROSS_COMPILE=arm-linux-gnueabihf-
  export ARCH=arm
  export INSTALL_MOD_PATH=$(pwd)/mod/
  mkdir $INSTALL_MOD_PATH
```

```
  #git reset --hard v4.14-rc5
  cp pfad/zu/mt7623n_evb_bpi_defconfig arch/arm/configs/
  make mt7623n_evb_bpi_defconfig
  make

  ...

  #cat arch/arm/boot/zImage arch/arm/boot/dts/mt7623n-bananapi-bpi-r2.dtb
>zImage_dtb
  cat arch/arm/boot/zImage arch/arm/boot/dts/mt7623n-bananapi-bpi-r2.dtb >
arch/arm/boot/zImage-dtb
  #mkimage -A arm -O linux -C none -T kernel -a 0x80008000 -e 0x80008000 -d
zImage_dtb uImage
  mkimage -A arm -O linux -T kernel -C none -a 80008000 -e 80008000 -n
"Linux Kernel 4.14" -d arch/arm/boot/zImage-dtb ./uImage

  rm -r $INSTALL_MOD_PATH/lib
  make modules_install

  cp uImage /media/$USER/BPI-BOOT/bananapi/bpi-r2/linux/  #<< mountpunkt
anpassen
  sudo cp -r $INSTALL_MOD_PATH/lib /media/$USER/BPI-ROOT/ #<< mountpunkt
anpassen
```

für Ramdisk-version:

config from user RyderLee
Ramdisk

pcie.patch
damit der PCIe-Slot (wenn CONFIG_PCIEPORTBUS,CONFIG_PCIEPORTBUS,CONFIG_PCIEPORTBUS aktiviert)

Script zum kompilieren/kopieren
angepasste defconfig

da beim 4.4.70 mmc0=sd-karte und mmc1=emmc habe ich dieses auch beim 4.14er Kernel durchgeführt (sonst muss man u-boot und ggf. die Mount-Points im System permanent ändern)

einfach in der arch/arm/boot/dts/mt7623n-bananapi-bpi-r2.dts und der arch/arm/boot/dts/mt7623.dtsi den mmc1-Block über den mmc0-Block setzen

mmc-swap.diff

wie build.sh verwenden

1. cd in das Kernel-Verzeichnis
    1. git clone https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git/
    2. oder git pull/git fetch

2. letztes „tag“ von 4.14: „git tag|grep 4.14“ #evtl. aktuelle Änderung sichern: “git diff > file.diff” (auch aktuelle Revision merken)
3. „git reset –hard v4.14“ #« tag ändern wenn neueres existiert (löscht alle Änderungen!)
    1. alternativ git stash,git checkout tags/4.14.x,git stash apply
4. die
   defconfig
   und Patch (
   mmc_pcie.patch
) in das Kernel-Verzeichnis kopieren (über „linux“)
5. das
   Script zum kompilieren/kopieren
   in das Verzeichnis „linux“ innerhalb des Kernel-Verzeichnisses kopieren.
6. dann in „linux“ reingehen und folgendes ausführen

```
patch -p1 < ../mmc_pcie.patch
./build.sh importconfig #mt7623n_evb_fwu_defconfig (.txt entfernt) muss in übergeordnetem Verzeichnis existieren
./build.sh config #starte menuconfig
./build.sh #starte make & Kopiervorgang
```

kompilierter 4.14.x

https://drive.google.com/drive/folders/1EGN1TvqCpDHdOAS-mjRg9ipi0kahnOUV?usp=sharing

das 4.14-Archiv (bpi-r2_kernelversion_gitbranch.tar.gz) enthält 2 Ordner, deren Inhalt auf die SD/MMC mit dem selben Namen kopiert werden müssen.

- BPI-BOOT ⇒ boot-Partition, uImage=Kernel-file, backup first your existing uImage before overwriting it
- BPI-ROOT ⇒ partition mit dem Betriebssystem, hier liegen die Kernel-Module (/lib/modules/kernelname). Mit jedem neuen Kernel wird ein neuer Ordner angelegt, so dass ein Backup nicht unbedingt nötig ist, außer es ist die selbe Kernelversion. Die Dateien in den anderen Ordnern sind großteils für den internen WLAN-Chip (wmt-tools, firmware, config) und müssen nicht jedesmal kopiert werden (ändern sich kaum bis gar nicht). Diese sind nur mit vorhanden für Personen, welche sich ein Image selbst bauen.

DTS ändern

dts(i)-Definition aus neuerem Kernel (arch/arm/boot/dts/) in älteren einbauen

- mt7623.dtsi
- mt7623n-bananapi-bpi-r2.dts ⇒ mt7623n-bpi-r2.dts

debugging

- https://kernelnewbies.org/FAQ/LinuxKernelDebug101
- https://elinux.org/Debugging_by_printing

- https://www.kernel.org/doc/Documentation/printk-formats.txt

Links

- Linux Linux-Stable
- Elixir Cross Referencer
- MTK-Patchwork MTK-Archive
- Net Net-Next Netdev-ML-Archive

LEDE (OpenWRT)

- Garys github
- Forum (img)
- Forum (kompilieren)

```
git clone https://github.com/garywangcn/bpi-r2_lede.git
cd bpi-r2_lede/
git checkout bpi-r2-on-lede-v1
make menuconfig
```

hier muss bei „Target System“ der Eintrag „MediaTek Ralink ARM“ ausgewählt werden und unter „Boot Loaders“ der Eintrag „u-boot-bpi_r2“ mit einem ∗

kompilieren mit

```
make -j1 V=s
```

die img-Dateien liegen auf meinem gdrive

Aktuell unterstützt lede keine HDMI-Ausgabe (es kommt nur ein Lila Bildschirm)

SD

```
  dd if=mtk-bpi-r2-SD.img of=/dev/sdx
```

EMMC

1. EMMC-image auf eine SD-karte mit einem lauffähigen System kopieren
2. System starten, vorher schauen, welches mmcblk das emmc ist: cat /proc/partitions (dort wo es eine boot0 gibt ist der emmc)
3. emmc-Abbild auf den EMMC-Benutzerblock kopieren: dd if=mtk-bpi-r2-EMMC.img of=/dev/mmcblkX
4. EMMC boot0 Block entsperren: echo 0 > /sys/block/mmcblkXboot0/force_ro
5. Preloader auf EMMC boot0 block schreiben: dd if=mtk-bpi-r2-EMMC.img of=/dev/mmcblkXboot0 bs=1M count=1
6. Ändern der Partitions-Konfiguration des EMMC auf 48h: System-Neustart mit SD-Karte und in die Uboot-Konsole, Befehl „emmc pconf 48“ ausführen
7. Ausschalten, SD-Karte entfernen und R2 wieder hochfahren.

Netzwerk-Konfiguration

Am Anfang sind die LAN-Ports zusammen gebrückt mit IP-Adresse 192.168.1.1

Weitergehende Konfiguration siehe hier: https://wiki.openwrt.org/doc/uci/network

Links

eigene Repos

- https://github.com/frank-w/BPI-R2-4.4
- https://github.com/frank-w/BPI-R2-4.14

offizielle Repos

- https://github.com/BPI-SINOVOIP/BPI-R2-bsp
- https://github.com/BPI-SINOVOIP/BPI-R2-bsp-4.14
- https://github.com/garywangcn/bpi-r2_lede

andere Repos

- https://github.com/abbradar/mt6625l-wlan-gen2 (separated wifi-driver)
- https://github.com/abbradar/openwrt/tree/bpi-upstream/target/linux/mediatek/patches-4.14 (patches)
- https://github.com/d3adme4t/BPI-R2-4.14/tree/4.14-hdmi
- https://github.com/wtolkien/meta-mediatek (uboot and other patches)

Mainline-Kernel

- https://www.kernel.org/
- https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
- https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable-rc.git/?h=linux-4.14.y
- https://patchwork.kernel.org/project/linux-mediatek/list/

LXC

🔧 Fix Me!

LXC-create

```
lxc-create -n name -t debian -- -r stretch -a armhf
```

manuelle Installation

```
distro=stretch
targetdir=$(pwd)/debian_stretch
#arch=amd64
arch=armhf
mkdir -p $targetdir
sudo debootstrap --include=lxc --arch=$arch --foreign $distro
$targetdir/rootfs
sudo chroot $targetdir/rootfs
```

Netzwerk

/etc/network/interfaces (host)

```
auto lxcbr0
iface lxcbr0 inet static
        bridge_ports none
        bridge_fd 0
        bridge_maxwait 0
        address 10.0.3.1
        netmask 255.255.255.0
#       broadcast 10.0.3.255
```

dnsmasq für DHCP-Server

Konfiguration

/var/lib/lxc/containername/config

```
lxc.start.auto = 1
```

```
#lxc.start.delay = 0 (in seconds)
lxc.start.delay = 5
#lxc.start.order = 0 (higher means earlier)
#lxc.start.order = 0

lxc.network.type = veth
lxc.network.link = lxcbr0
lxc.network.flags = up

#optional bei fester IP-Adresse
lxc.network.ipv4 = 10.0.3.10/24
lxc.network.ipv4.gateway = auto

lxc.mount.entry = /var/www /var/lib/lxc/stretch-web/rootfs/var/www/ none
bind 0 0
```

start/stop Container

```
lxc-start -n name
lxc-stop -n name
```

anzeigen des Status:

```
lxc-ls --fancy
```

in Container einsteigen

```
lxc-console -n name
```

beenden mit Strg+A,q

Befehl im Container ausführen

```
lxc-attach -n name -- /usr/local/sbin/script.sh
```

Netzwerk-Konfiguration

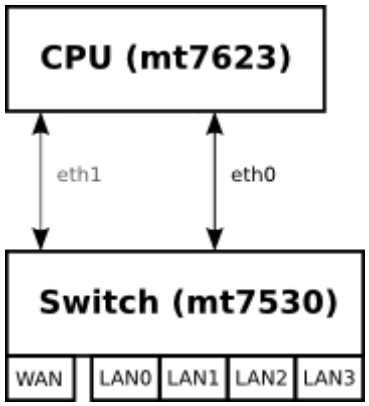
IP-Befehl benötigt iproute2-Paket

sonstige Pakete:

```
apt install telnet traceroute
apt install nftables
apt install net-tools #netstat
apt install nmap
apt install tcpdump
apt install dnsutils --no-install-recommends #dig
apt install iperf3
apt install ethtool
```

die Konfiguration auf dieser Seite basiert auf debian stretch, sollte genauso unter Ubuntu funktionieren setzt Kernel 4.14 oder höher voraus (DSA-Treiber für Port-Trennung)

im Kernel 4.14 existiert eth0 als Verbindung zwischen CPU und dem Switch-Chip (mt7530), an denen die Ports wan und lan0-4 angeschlossen sind. diese Verbindung muss zuerst „up“ genommen werden.



Die Zuordnung der Ports auf eine GMAC ist definiert in der dts-Datei und kann mit „ip a“ angezeigt werden

Mit Kernel 4.14 >.52 wurde in meinem Repo die GMAC #2 (eth1) hinzugefügt und der Wan-Port darauf verbunden.

ohne weitere Konfiguration sind die LAN-Ports getrennt und benötigen eine eigene IP-Konfiguration in verschiedenen Subnetzen

viele Nutzer möchten sicherlich alle LAN-Ports im gleichen Netzwerk-Segment haben, so empfiehlt sich sie mit einer Netzwerkbrücke zu verbinden, um nur 1 IP-Konfiguration für “LAN” zu haben

MAC-Adresse

Die MAC-Adresse lässt sich nur für die GMAC (Verbindung Switch-CPU) setzen. Aktuell wird unter

Kernel 4.14 nur eine erkannt (eth0). Hardwareseitig sind 2 GMACs vorhanden.

Möglichkeit via UDEV von hier

```
$ cat /etc/udev/rules.d/00-static-mac-address.rules
ACTION=="add", SUBSYSTEM=="net", KERNELS=="1b100000.ethernet",
RUN+="/sbin/ip link set dev %k address ae:fc:de:ad:be:ef"
```

/etc/network/interfaces

```
iface lan0 inet static
  address 192.168.0.10
  netmask 255.255.255.0
  gateway 192.168.0.5
#  pre-up ip link set $IFACE up
  pre-up ip link set $IFACE address 02:01:02:03:04:08 up
```

systemd nutzen:

/etc/systemd/network/10-wan.link

```
[Match]
OriginalName=wan

[Link]
MACAddress=XX:XX:XX:XX:XX:XX
```

http://forum.banana-pi.org/t/set-mac-address-on-boot/7224/7

Gerätename

Ubuntu 18.4 (und vermutlich kommende Debian-Varianten) nutzen neue Gerätenamen. Wlan-Geräte heißen nicht mehr wlanX sondern wlpXsY

um das zu verhindern, kann man „net.ifnames=0“ zur Kernel-Cmdline hinzufügen (uEnv.txt bei uboot) und dann wie gewohnt via udev ubenennen

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

```
SUBSYSTEM=="net", ACTION=="add", ATTR{address}=="f8:62:aa:50:15:c8",
NAME="wlan1"
```

an die Attribute kommt man mit

```
udevadm info --attribute-walk /sys/class/net/<interface-name>
```

damit die Änderungen übernommen werden soll folgendes funktionieren (hat es bei mir aber nicht, obwohl ifnames schon aktiv und mein Interface schon wlanx hieß, auch den udev service neu starten hat nicht geklappt):

```
udevadm control --reload-rules && udevadm trigger
```

nach einem Reboot hat es aber gepasst

alternativ kann man das entsprechende Treibermodul neu laden (nachdem man die udev-rules neu geladen hat)

```
modprobe -r mt76x2e
modprobe mt76x2e
```

IP

dauerhaft

/etc/network/interfaces:

```
#upstream-Port (NIC zwischen CPU und MT7530-Switch) auf up
auto eth0
iface eth0 inet manual
  pre-up ip link set $IFACE up
  post-down ip link set $IFACE down

#upstream-Port #2 4.14.x >52 (NIC zwischen CPU und MT7530-Switch) auf up
auto eth1
iface eth1 inet manual
  pre-up ip link set $IFACE up
  post-down ip link set $IFACE down

#dann die lan-ports konfigurieren
auto lan0
iface lan0 inet static
  hwaddress ether 08:00:00:00:00:00 # if you want to set MAC manually
  address 192.168.0.10
  netmask 255.255.255.0
  gateway 192.168.0.5
  pre-up ip link set $IFACE up
  post-down ip link set $IFACE down
```

systemd

/etc/systemd/network/eth0.network:

```
[Match]
Name=eth0

[Network]
DHCP=no
```

```
LinkLocalAddressing=no
ConfigureWithoutCarrier=true
```

/etc/systemd/network/wan.network

```
[Match]
Name=wan

[Network]
BindCarrier=eth0
#ConfigureWithoutCarrier=true

#IPForward=yes
#IPMasquerade=yes
Address=192.168.0.18/24
DNS=192.168.0.10
Gateway=192.168.0.10
```

Wird ConfigureWithoutCarrier beim wan-port gesetzt wird die default-route nicht gesetzt,weil die Adresse ungültig ist (da Netzwerk down zum Zeitpunkt der Konfiguration). Das darf also nur gesetzt werden,wenn keine default-route gesetzt werden muss.

systemd

zweite Ethernet-Anbindung (gmac)

benötigt kernel-patch für eth1 + aux interfaces (aktuell nur in 5.15)

```
- create a bridge for use for wan

    /etc/systemd/network/11-wanbr.netdev
    [NetDev]
    Name=wanbr
    Kind=bridge

    [Bridge]
    DefaultPVID=0 # should be different to other vlan-aware bridges (like
lanbr)
    VLANFiltering=1

- map aux and wan to vlan-aware bridge
- traffic will be tagged inside with vlan-id 99

    /etc/systemd/network/12-wanbr-bind.network
    [Match]
    Name=wan aux

    [Link]
    RequiredForOnline=no
```

```
    [Network]
    BindCarrier=eth0
    Bridge=wanbr

    [BridgeVLAN]
    VLAN=99
    PVID=99
    EgressUntagged=99

- put wanbr up by default

    /etc/systemd/network/13-wanbr.network
    [Match]
    Name=wanbr

    [Network]
    BindCarrier=eth0
    ConfigureWithoutCarrier=true

- configure eth1 as wan

    /etc/systemd/network/15-wan.network
    [Match]
    Name=eth1

    [Network]
    BindCarrier=eth0

    Address=192.168.0.18/24
    Gateway=192.168.0.10
    DNS=192.168.0.10

    IPForward=yes
```

temporär

```
ifconfig eth0 192.168.0.10 netmask 255.255.255.0 broadcast 192.168.0.255
```

```
ip addr set 192.168.0.10/24 broadcast 192.168.0.255 dev eth0
```

DHCP

Client

/etc/network/interfaces:

```
auto lan3
```

```
allow-hotplug lan3
iface lan3 inet dhcp
```

Server

/etc/dnsmasq.conf (aktivieren durch entfernen des # am Zeilenanfang)

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

/etc/dnsmasq.d/interfaces.conf

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

# DHCP-Server nicht aktiv für Schnittstelle
#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

```
service dnsmasq restart
```

mehrere IP pro MAC mit dnsmasq: https://stackoverflow.com/a/26964151

mehr Inforationen hier: dnsmasq

IPv6

```
ip -6 addr add fd00:a::12/64 dev wan
ping -6 -I wan fe80::a02:ff:fe00:10 #für ping auf fe80 muss Interface
angegeben werden
ip -6 route
ip -6 route add default dev he-ipv6 #if you use he-ipv6-tunnel and no
default-route is set
ping6 2001:4860:4860::8844 #google-dns
```

sysctl

```
sysctl -w net.ipv6.conf.all.autoconf=1
sysctl -w net.ipv6.conf.all.accept_ra=1
```

do not set forwarding if you want RAs

```
# Uncomment the next line to enable packet forwarding for IPv6
#  Enabling this option disables Stateless Address Autoconfiguration
#  based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1
```

if RAs were not received/sent after config-change (e.g. sysctl), try to restart radvd

interfaces

```
auto wan
...
iface wan inet6 auto
```

or

```
iface lanbr0 inet6 static
  address fd00:A::10
  netmask 64
```

radvd

```
interface lanbr0 {
  AdvSendAdvert on;
  prefix fd00:A::0/64
  {
    AdvOnLink on;
    AdvAutonomous on;
    AdvRouterAddr on;
  };
  prefix 2001:470:xxxx:0::/64 #he
  {
    AdvOnLink on;
    AdvAutonomous on;
    AdvRouterAddr on;
  };
  route ::/0
  {
    AdvRoutePreference high;
  };
};
```

NAT/Routing

NAT

zum aktivieren von Network Adress Translation (private IPs hinter einer öffentlichen IP)

```
ipt=/sbin/iptables
if_wan=wan
${ipt} -t nat -A POSTROUTING -o ${if_wan} -j MASQUERADE
```

Routing

Routing für IPv4 aktivieren

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Alternative:

```
nano /etc/sysctl.conf
#activate net.ipv4.ip_forward=1 and net.ipv6.conf.all.forwarding=1 by
removing # at beginning of line
sysctl -p /etc/sysctl.conf
```

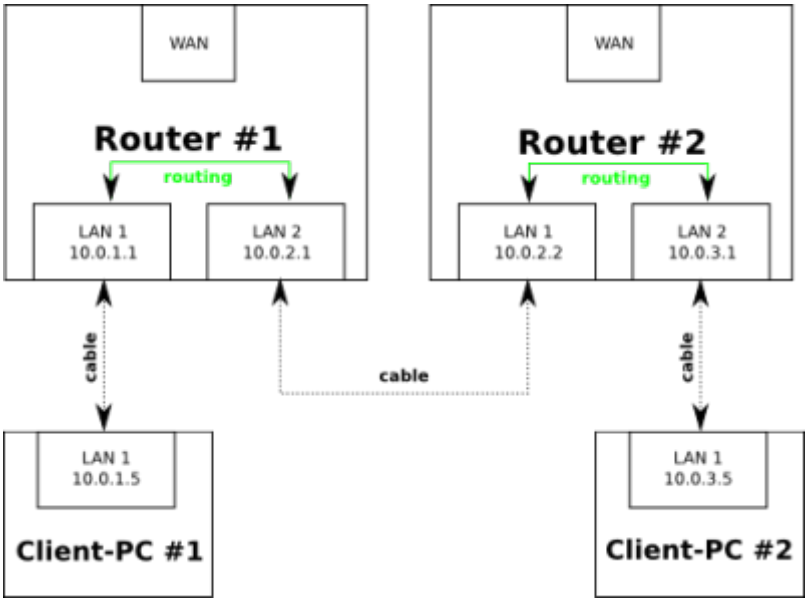
Standard-Gateway ändern:

```
ip route del default
ip route add default via 192.168.50.2
```

statische Routen zu anderen Netzwerken

Pakete werden an das Default-Gateway geschickt, wenn das Netz dem Router nicht bekannt ist. In klassischen Heimnetzen existiert meist nur ein Router und dort ist das default-Gateway die Internet-Schnittstelle und an den Client-Rechnern ist das Default-Gateway der Router.

statische Routen werden benötigt, wenn ein Netz nicht direkt am Router hängt und nicht über sein Default-Gateway (hier WAN) erreichbar ist



- im Router #1 muss eine statische Route für das Netz 10.0.3.0/24 mit einem Next-Hop 10.0.2.2 (schickt Pakete über lan#2) eingerichtet werden
  - ```
    route add -net 10.0.3.0 netmask 255.255.255.0 gw 10.0.2.2
    ip route add 10.0.3.0/24 via 10.0.2.2
    ```
- im Router #2 muss eine statische Route für das Netz 10.0.1.0/24 mit einem Next-Hop 10.0.2.1 (schickt Pakete über lan#1) eingerichtet werden
  - ```
    route add -net 10.0.1.0 netmask 255.255.255.0 gw 10.0.2.1
    ip route add 10.0.1.0/24 via 10.0.2.1
    ```

Beispiel für Netz 192.168.50.x hinter Router mit IP 192.168.0.10

```
route add -net 192.168.50.0 netmask 255.255.255.0 gw 192.168.0.10
ip route add 192.168.50.0/24 via 192.168.0.10
```

DNS

/etc/resolv.conf

enthält die IP-ADresse des Nameservers, z.B.

```
nameserver 192.168.0.10
```

systemctl-resolvd:

```
ln -fs /var/run/systemd/resolve/resolv.conf /etc
systemctl restart systemd-resolved
```

aktuell verwendeten Server ermitteln

```
dig | grep SERVER
```

Netzwerkbrücke

```
apt-get install bridge-utils
```

/etc/network/interfaces:

```
auto lan1
iface lan1 inet manual
auto lan2
iface lan2 inet manual

auto br0
iface br0 inet static
    address 192.168.40.1
    netmask 255.255.255.0
    bridge_ports lan1 lan2
    bridge_fd 5
    bridge_stp no
```

temporär

```
brctl addbr br0
brctl addif br0 lan2
ip addr add 192.168.0.18/24 dev br0
ip link set br0 up

brctl show br0
brctl delif br0 lan1
ip link set br0 down
brctl delbr br0
```

VLAN

temporär:

```
ip link add link wan name wan.140 type vlan id 140
#löschen
ip link del wan.140
```

/etc/network/interfaces:

```
auto lan3.60
iface lan3.60 inet static
```

```
  address 192.168.60.10
  netmask 255.255.255.0
```

vlan-aware bridge

```
# ip link add name br0 type bridge
# ip link set dev br0 up
# ip link set dev lan3 master br0
# ip link set br0 type bridge vlan_filtering 1
# ip a a 192.168.50.1/24 dev br0
# ip link set lan3 up
# bridge vlan add vid 100 dev br0 self
# bridge vlan add vid 100 dev lan3
# bridge vlan
port              vlan-id
lan3              1 PVID Egress Untagged
                  100
br0               1 PVID Egress Untagged
                  100
# ip link add link br0 name br0.100 type vlan id 100
# ip a a 192.168.51.1/24 dev br0.100
# ip link set br0.100 up
```

on the other side create normal vlan

```
# ip link add link enx00133b9302ee name eth2.100 type vlan id 100
# ip a a 192.168.51.2/24 dev eth2.100
# ip link set eth2.100 up
```

Firewall (iptables)

iptables nftables

Monitoring

```
#alles außer ssh
sudo tcpdump -i eth0 port not 22 > tcpdump.log
#alles von oder nach 192.168.0.11
tcpdump -nni lan0 host 192.168.0.11
#alles von 192.168.0.11
tcpdump -nni lan0 src host 192.168.0.11
#alles nach 192.168.0.11
tcpdump -nni lan0 dst host 192.168.0.11
#nur icmp
tcpdump -nni lan0 icmp
#pppoe discovery (PADI-PADT), ppp/lcp wird mit "pppoes" angezeigt
```

```
tcpdump -i wan pppoed
#wireshark-compatibles mitloggen (volle Pakete), es geht auch "-s0"
tcpdump -i <interface> -s 65535 -w tcpdump.cap
```

https://www.rationallyparanoid.com/articles/tcpdump.html https://danielmiessler.com/study/tcpdump/
https://www.cyberciti.biz/faq/network-statistics-tools-rhel-centos-debian-linux/
https://serverfault.com/questions/533513/how-to-get-tx-rx-bytes-without-ifconfig

```
ip -s link
netstat -i
netstat -s
ethtool -S eth0
cat /proc/net/dev
nmap <IP> -Pn -p T:<PORT>
```

erweiterte Netzwerk-Konfiguration

- https://lartc.org/howto/
- https://wiki.archlinux.org/index.php/advanced_traffic_control
- https://wiki.gentoo.org/wiki/Traffic_shaping
- https://wiki.ubuntuusers.de/Skripte/Traffic-Shaping/

Policy Routing

policyrouting

QoS

spezielle konfiguration

Bambit

- vlan 140 ⇒ internet
- vlan 110 ⇒ voip

https://www.stadtwerke-bamberg.de/de/Meta/Service/Downloadportal/__attic__20180503_103612__Downloadportal/STWB-Internet-Technische-Information-Einsatz-eigenes-Endgeraet.pdf

```
ip link add link wan name wan.140 type vlan id 140
ip link add link wan name wan.110 type vlan id 110
# 2. pppoe-verbindung muss von anderer MAC-Adresse kommen
ip link set wan.110 down
ip link set wan.110 address 02:01:02:03:04:08 up
pppoeconf wan.140
plog
#config sichern
```

```
cp /etc/ppp/peers/dsl-provider /etc/ppp/peers/bambit-internet
pppoeconf wan.110
plog

cp /etc/ppp/peers/dsl-provider /etc/ppp/peers/bambit-voip
#diese datei bearbeiten und diese beiden Optionen auskommentieren:
#defaultroute
#replacedefaultroute

über die option unit <nummer> oder ifname <name> lässt sich der Name des
PPP-Interfaces festlegen (um dieses im ppp-ip-up-script unterscheiden zu
können)

unit 8 #nennt das Interface ppp8
#ifname ppp-bambit #nennt das Interface ppp-bambit
```

anpassen der /etc/network/interfaces:

```
#bambit-inet
auto wan.140
iface wan.140 inet manual

auto bambit-internet
iface bambit-internet inet ppp
pre-up /bin/ip link set wan.140 address 02:11:02:03:04:05 up # line
maintained by pppoeconf
provider bambit-internet

#bambit-voip
auto wan.110
iface wan.110 inet manual

auto bambit-voip
iface bambit-voip inet ppp
pre-up /bin/ip link set wan.110 address 02:12:02:03:04:06 up # line
maintained by pppoeconf
provider bambit-voip
```

sind beide Verbindungen aufgebaut, sollte „ip a“ so aussehen:

```
13: ppp0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1492 qdisc
pfifo_fast state UNKNOWN group default qlen 3
    link/ppp
    inet 217.61.x.y peer 217.61.x.1/32 scope global ppp0
16: ppp1: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1492 qdisc
pfifo_fast state UNKNOWN group default qlen 3
    link/ppp
    inet 172.20.x.y peer 172.20.x.1/32 scope global ppp1
```

DNSMASQ

Konfigurationsorder freischalten

damit eigene Konfigurationsdateien geladen werden muss das Verzeichnis /etc/dnsmasq.d aktiviert werden

dazu in der /etc/dnsmasq.conf bei folgender Zeile die Raute (#) am Zeilenanfang löschen

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

Schnittstellen

die Schnittstellenkonfiguration habe ich bei mir in die /etc/dnsmasq.d/interfaces.conf ausgelagert. hier wird festgelegt, auf welchen Interfaces welche Einstellungen gelten.

welche Interfaces sollen DHCP-Requests beantworten (binding)?

```
interface=lan0
interface=wlan1
#interface=tun0
interface=lxcbr0
interface=ap0
```

welche nicht

```
no-dhcp-interface=eth0
no-dhcp-interface=eth1
```

welche EInstellungen:

```
dhcp-range=lan0,192.168.0.100,192.168.0.150,255.255.255.0,48h
dhcp-option=lan0,3,192.168.0.10
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
dhcp-range=lxcbr0,10.0.3.100,10.0.3.150,255.255.255.0,48h
dhcp-option=lxcbr0,3,10.0.3.1
```

dhcp-range=interface,ip-range-start,ip-range-end,netmask,lease-time dhcp-option=interface,3,Default-Gateway

statische IP für MAC

man kann mit DNSMASQ via DHCP bestimmten MAC-ADressen immer die gleiche IP-Adresse geben. Für diesen zweck habe ich mir eine eigene Konfig-Datei (/etc/dnsmasq.d/mac.conf) angelegt.

ein Eintrag sieht da z.b. so aus:

```
dhcp-host=b8:27:eb:90:e6:06,raspberry,192.168.0.2,12h
```

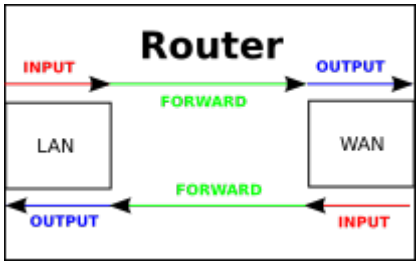
die MAC-Adresse des jeweiligen Clients, ein Name, die IP und als letztes die Lease-time (wie lange ist die DHCP-Antwort gültig ⇒ Erneuerung der IP notwendig)

bei WLAN hatte ich nun das Problem, dass ich 2 WLAN-APs (mit unterschiedlichem IPv4-Subnet) am R2 habe und auf beiden die gleichen MAC_Adressen ankamen. LÖsung ist hier, die Schnittstelle am Anfang hinzuzunehmen:

```
dhcp-host=ap0,00:25:d3:f5:32:4b,media,192.168.10.11,12h
dhcp-host=wlan1,00:25:d3:f5:32:4b,media,192.168.11.11,12h
```

IPTables

🔧 Fix Me!



IPv4

```
#alle vorherigen Regeln löschen
${ipt} -F
${ipt} -X
${ipt} -t nat -F
${ipt} -t nat -X
${ipt} -t mangle -F
${ipt} -t mangle -X

# standard-Regel für IPv4: alles droppen
${ipt} -P INPUT   DROP
${ipt} -P OUTPUT  DROP
${ipt} -P FORWARD DROP
```

```
# policy für TCP-Reset/UDP-Reject als Alternative für "-j DROP"
${ipt} -N ABGELEHNT
if [[ ! "${LOG}" = "" ]];
then
  echo "enable IPv4-Firewall-Logging (all)...";
  ${ipt} -A ABGELEHNT -m limit --limit 10/min -j LOG --log-prefix
"NETFILTER4-ABGELEHNT: " --log-level 4
fi
${ipt} -A ABGELEHNT -p tcp -j REJECT --reject-with tcp-reset
${ipt} -A ABGELEHNT -p udp -j REJECT --reject-with icmp-port-unreachable
${ipt} -A ABGELEHNT -j DROP
```

```
# localhost
${ipt} -A INPUT  -i lo -j ACCEPT
${ipt} -A OUTPUT -o lo -j ACCEPT

${ipt} -A OUTPUT -j ACCEPT
${ipt} -A INPUT  -m state --state RELATED,ESTABLISHED -j ACCEPT #
angeforderte, bestehende Verbindungen eingehend
${ipt} -A INPUT  -p icmp -m limit --limit 5/s --icmp-type echo-request -j
```

```
ACCEPT # ICMP eingehen, max 5/s
```

```
#Block Teredo-Stuff
#${ipt} -I FORWARD -p udp --dport 3544 -j ABGELEHNT
#${ipt} -I FORWARD -p udp --sport 3544 -j ABGELEHNT
#http://en.wikipedia.org/wiki/List_of_IP_protocol_numbers
${ipt} -A FORWARD -p 41 -j ABGELEHNT #IPv6 Encapsulation
${ipt} -A FORWARD -p 43 -j ABGELEHNT #Routing Header for IPv6
${ipt} -A FORWARD -p 44 -j ABGELEHNT #Fragment Header for IPv6
${ipt} -A FORWARD -p 58 -j ABGELEHNT #ICMP for IPv6
${ipt} -A FORWARD -p 59 -j ABGELEHNT #No Next Header for IPv6
${ipt} -A FORWARD -p 60 -j ABGELEHNT #Destination Options for IPv6
```

```
#ssh mit rate-limit
${ipt} -I INPUT -p tcp --dport 22 -i ${if_ext} -m state --state NEW -m
recent --set
${ipt} -I INPUT -p tcp --dport 22 -i ${if_ext} -m state --state NEW -m
recent --update --seconds 60 --hitcount 4 -j ABGELEHNT #4 verbindungen in 1
Minute
${ipt} -A INPUT  -p tcp --dport  22 -j ACCEPT #SSH eingehend
```

```
${ipt} -A FORWARD -i ${if_int} -o ${if_ext} -j ACCEPT #Forwarding Int->Ext
${ipt} -A FORWARD -i ${if_ext} -o ${if_int} -m state --state
ESTABLISHED,RELATED -j ACCEPT #Forwarding Ext->Int (nur
bestehende/angeforderte Verbindg.)

${ipt} -A INPUT -i ${if_int} -j ACCEPT #erlaubt alle Anfragen von Intern
```

port-forwardings

```
# REJECT/RESET fuer alles andere
${ipt} -A INPUT   -j ABGELEHNT
${ipt} -A OUTPUT  -j ABGELEHNT
${ipt} -A FORWARD -j ABGELEHNT
```

zusätzliche Optionen:

```
#Kernel-Option fuer SYN-Cookies
echo 1 > /proc/sys/net/ipv4/tcp_syncookies  #enable syn cookies (prevent
against 'syn flood attack')

if [ -f /proc/sys/net/ipv4/conf/all/accept_redirects ]; then
    echo "   Kernel ignores all ICMP redirects"
    echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects
fi

if [ -f /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts ]; then
   echo "   Kernel ignores ICMP Echo requests sent to broadcast/multicast
addresses"
   echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

```
fi
```

Port-Forwardings

einrichten

port 522 auf Client 192.168.0.5 port 22 weiterleiten

```
${ipt} -t nat -A PREROUTING -p tcp --dport 522 -j DNAT --to-destination
192.168.0.5:22
```

anzeigen

```
iptables -L -t nat

Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DNAT       tcp  --  anywhere             anywhere             tcp dpt:522
to:192.168.0.5:22
```

active-FTP

damit Clients FTP im ACTIVE-Modus nutzen können müssen 2 Module geladen und eine 1 iptables-Regel angewandt werden

```
modprobe ip_conntrack_ftp
modprobe ip_nat_ftp ports=21
```

```
${ipt} -A INPUT -p tcp --sport 20 -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

IPv6

NFTables

```
apt install nftables
echo 1 > /proc/sys/net/ipv4/ip_forward

nft list ruleset
nft add table nat
nft add chain ip nat prerouting { type nat hook prerouting priority 100 \; }
nft add chain ip nat postrouting { type nat hook postrouting priority 100 \;
}
nft add rule nat postrouting masquerade

#portforwarding
nft add rule nat prerouting iif lan1 tcp dport 443 dnat 192.168.0.10:443 #
ip needs to be routed to other interface then in-interface (here lan1)
```

einfache Befehle

```
nft flush ruleset #alles löschen
nft -f flowoffload.nft #importieren
nft show ruleset #alles anzeigen
```

einfache struktur

```
table ip filter {
        chain input {
                type filter hook input priority 0; policy accept;
        }

        chain output {
                type filter hook output priority 0; policy accept;
        }

        chain forward {
                type filter hook forward priority 0; policy accept;
        }
}
table ip nat {
        chain post {
                type nat hook postrouting priority 0; policy accept;
                oifname "wan" masquerade
        }

        chain pre {
                type nat hook prerouting priority 0; policy accept;
```

```
        }
}
```

links

- https://developers.redhat.com/blog/2017/01/10/migrating-my-iptables-setup-to-nftables/
- https://wiki.gentoo.org/wiki/Nftables/Examples#Basic_routing_firewall
- https://wiki.nftables.org/wiki-nftables/index.php
- https://wiki.nftables.org/wiki-nftables/index.php/Quick_reference-nftables_in_10_minutes
- https://github.com/alpinelinux/aports/blob/master/main/nftables/nftables.nft

hwnat

https://github.com/frank-w/BPI-R2-4.14/commits/5.12-hnat

ipv6 mangle crasht noch

für hwnat wird eine neuere version der nftables benötigt als in debian buster angeboten wird

https://github.com/frank-w/nftables-bpi

kompiliert: https://drive.google.com/drive/folders/1hajKvqQa96WRrAy52fQX90i59I1s0h-i?usp=sharing

basic IPv4 Ruleset:

```
flush ruleset
table ip filter {
        flowtable f {
                hook ingress priority filter + 1
                devices = { lan3, lan0, wan }
                flags offload;
        }
        chain input {
                type filter hook input priority filter; policy accept;
        }

        chain output {
                type filter hook output priority filter; policy accept;
        }

        chain forward {
                type filter hook forward priority filter; policy accept;
                ip protocol { tcp, udp } flow add @f
        }
}
table ip nat {
        chain post {
                type nat hook postrouting priority filter; policy accept;
```

```
                oifname "wan" masquerade
        }

        chain pre {
                type nat hook prerouting priority filter; policy accept;
        }
}
```

basic v6 Ruleset (crash!):

```
flush ruleset
table ip6 filter {
        flowtable f {
                hook ingress priority 1
                devices = { lan3, lan0, wan }
                flags offload;
        }
        chain input {
                type filter hook input priority 0; policy accept;
        }

        chain output {
                type filter hook output priority 0; policy accept;
        }

        chain forward {
                type filter hook forward priority 0; policy accept;
                ip6 nexthdr { tcp, udp } flow add @f
        }
}
table ip6 nat {
        chain post {
                type nat hook postrouting priority 0; policy accept;
                oifname "wan" masquerade
        }

        chain pre {
                type nat hook prerouting priority 0; policy accept;
        }
}
```

testen:

```
nft -f nft-nat-flowoffload.nft
#vom client traffic generieren
cat /sys/kernel/debug/mtk_ppe/entries
```

IPV6-Setup

```
#!/bin/bash
#on main-router:
```

```
#ip -6 route add fd00:a2::/64 via fd00:a::12
#ip -6 route add 2001:470:xxxx:a2::/64 via 2001:470:xxxx::12

ip -6 addr add fd00:a::12/64 dev wan
ip -6 addr add fd00:a2::12/64 dev lan3

ip -6 addr add 2001:470:xxxx::12/64 dev wan
ip -6 addr add 2001:470:xxxx:a2::12/64 dev lan3

sysctl -w net.ipv6.conf.all.forwarding=1
```

Policy Routing

hat man mehrere Wege kann man über rules definieren, welchen Weg ein Paket nehmen soll. In meinem Fall habe ich mit 2 Internet-Anschlüssen (telekom+bambit) rumprobiert

separate Routingtabelle in /etc/iproute2/rt_tables anlegen (am Ende hinzufügen)

```
1 telekom
2 bambit
```

PPP-UP-Script (erstellt default-Routen in separater Routing-Tabelle für ppp-Verbindung)

```
ip route flush table bambit

ip route add default via 192.168.178.1 dev wan table bambit
ip rule add from 192.168.178.10 lookup bambit

#route specific devices
ip rule add from 192.168.0.80/32 table bambit
ip rule add from 192.168.0.26/32 table bambit

#exit on local adresses
ip route add throw 192.168.10.0/24 table bambit
ip route add throw 192.168.11.0/24 table bambit
ip route add throw 192.168.0.0/24 table bambit

ip route flush cache
```

auch sollte man nicht den DNS des Providers nehmen, da dieser ggf. vom anderen Provider keine DNS-Auflösung macht. Diese Erfahrung habe ich gemacht...anpingbar war er, aber es gab keine DNS-Auflösung. Dazu in der peers-Datei (/etc/ppp/peers/dateiname) die option usepeerdns ausschalten.

Load-Balancing

Loadbalancing mit NAT hat ein paar Probleme besonders im Zusammenhang mit Captchas, da diese oft an eine IP-Adresse gebunden sind und es sein kann, dass die nachfolgende Verbindung nicht den gleichen Weg nimmt (=andere IP-Quell-Adresse). Der nachfolgende Teil dokumentiert nur meinen bisherigen Ansatz und ist nicht für den Produktiveinsatz bestimmt.

der zweite step ist, dass ankommender Traffic markiert wird um die Antwort zum gleichen interface wieder raus zu schicken. Das ist wichtig, da der client die Antwort von der Adresse erwartet an die er die Anfrage geschickt hat. Weiterhin existiert bei vielen Providern ein sog. Reverse Path Filtering, es wird also geschaut, ob die Quelladresse zum sendenden Host passt. Das ist besonders bei lokalem Traffic wichtig, da dieser nicht in der Prerouting-Chain landet sondern erst das NAT greift (default route) und dann nur in der Output Chain greifbar ist.

```
wan1=ppp8
wan2=ppp0

iptables -A PREROUTING -t mangle -j CONNMARK --restore-mark
#iptables -A PREROUTING -t mangle --match mark --mark 1 -j ACCEPT
iptables -A PREROUTING -t mangle -i $wan1 -j MARK --set-mark 1
#iptables -A PREROUTING -t mangle --match mark --mark 2 -j ACCEPT
iptables -A PREROUTING -t mangle -i $wan2 -j MARK --set-mark 2

#ggf. weitere markings

iptables -A PREROUTING -t mangle -j CONNMARK --save-mark
```

danach kommt der interne Traffic, damit auch dieser durchgängig den gleichen Uplink verwendet (solange die TCP-Session besteht)

```
iptables -t mangle -N MARKING

iptables -A PREROUTING -t mangle -m mark --mark 0x0 -j MARKING #without
mark move to new chain

#for local packets to get in prerouting-chain
#needs sysctl -w net.ipv4.conf.lan0.rp_filter=0

iptables -t mangle -A MARKING -j MARK --set-mark 3 #bambit
iptables -t mangle -A MARKING -m statistic --mode random --probability 0.3
-j MARK --set-mark 4 #telekom
#iptables -t mangle -A MARKING -m mark --mark 3 -j LOG --log-prefix "fwmark
3: "
#iptables -t mangle -A MARKING -m mark --mark 4 -j LOG --log-prefix "fwmark
4: "

iptables -A PREROUTING -t mangle -j CONNMARK --save-mark
```

hier wird jdes 3. unmarkierte Paket mit 4 markiert, der Rest (die anderen 2) bleibt bei der vorher gesetzten Markierung 3

zum Schluss muss man sich noch um den lokal generierten Traffic kümmern (kein forwarded ⇒ kein prerouting). Diesen bekommt man nur in der OUTPUT-Chain (des ausgehenden Interfaces...hier meine 2 ppp) zu packen

```
iptables -t mangle -N MARKING_LOCAL

iptables -t mangle -A OUTPUT -j CONNMARK --restore-mark
iptables -t mangle -A OUTPUT -o ppp0 -m mark --mark 0x0 -j MARKING_LOCAL
#without mark move to new chain
iptables -t mangle -A OUTPUT -o ppp8 -m mark --mark 0x0 -j MARKING_LOCAL
#without mark move to new chain
iptables -t mangle -A MARKING_LOCAL -j HMARK --hmark-offset 3 --hmark-tuple
sport,dport --hmark-mod 2 --hmark-rnd 0xdeb1a4f0
#iptables -t mangle -A MARKING_LOCAL -m mark --mark 3 -j LOG --log-prefix
```

```
"fwmark 3 (l): "
#iptables -t mangle -A MARKING_LOCAL -m mark --mark 4 -j LOG --log-prefix
"fwmark 4 (l): "
iptables -t mangle -A MARKING_LOCAL -j CONNMARK --save-mark

iptables --table nat --append POSTROUTING --out-interface ppp8 -j
MASQUERADE
iptables --table nat --append POSTROUTING --out-interface ppp0 -j
MASQUERADE
```

nun kann man über „ip rule“ festlegen welcher traffic (nach Markierung) über welche Routing-Tabelle geschickt wird:

```
ip rule add fwmark 1 table telekom #incoming from telekom
ip rule add fwmark 2 table bambit #incoming from bambit
ip rule add fwmark 3 table telekom #outgoing
ip rule add fwmark 4 table bambit #outgoing
```

OpenVPN

Installation und dann in der /etc/openvpn/server.conf folgendes hinzufügen/ändern:

```
port 1194
proto udp
dev tun

#user+group müssen ggf. vorher angelegt werden
user openvpn
group openvpn

#Schlüssel
ca /etc/openvpn/easy-rsa2/keys/ca.crt
cert /etc/openvpn/easy-rsa2/keys/bpi-r2.crt
key /etc/openvpn/easy-rsa2/keys/bpi-r2.key    # Diese Datei geheim halten.

dh /etc/openvpn/easy-rsa2/keys/dh2048.pem     # Diffie-Hellman-Parameter


persist-key
persist-tun

#tun-mtu ganz wichtig!! per default nimmt OpenVPN 1500, was u.a. bei
Hotspots zu Problemen führt
tun-mtu 1300

#if you want to access other networks
push "route 192.168.0.0 255.255.255.0"
#sämtlicher Verkehr durch den VPN-Tunnel
push "redirect-gateway def1"
#own DNS-Server
push "dhcp-option DNS 192.168.0.10"
```

Software

erste Schritte

Betriebssystem herunterladen: Google Drive oder dev.banana-pi.org.cn

Image auf SD-Karte bringen (Linux):

```
dd if=pfad/zum/image.img of=/dev/sdx bs=1M
```

Debian

erstes Booten (am besten mit Debug-UART)

Login: root **Passwort:** bananapi

System aktualisieren & Uhrzeit einstellen

```
apt-get update && apt-get upgrade
#hostname bpi-r2 # and
#sysctl kernel.hostname=bpi-r2 #does not work
echo "bpi-r2">/etc/hostname
dpkg-reconfigure tzdata
#echo "export PS1='[\A] \u@\h:\W# '">>~/.bashrc
```

Umgebungsvariablen (dauerhaft: echo „...“»~/.bashrc, „ vom Befehl selbst mit \ maskieren)

```
#fix für nano probleme auf der debug-console
if [[ "$(tty)" =~ "ttyS" ]]; then export TERM=vt100;fi
#prompt mit Zeitstempel
export PS1='[\A] \u@\h:\W# '
#leichterer Zugriff auf die GPIO mit $GPIO
export GPIO=/sys/devices/platform/1000b000.pinctrl/mt_gpio
```

Netzwerkeinstellungen

Netzwerkeinstellungen

temporär

```
#4.4.70:
ifconfig eth0 192.168.0.10/24
```

```
route add default gw 192.168.0.5
echo "nameserver 192.168.0.5" > /etc/resolv.conf
```

```
#4.14:
#ifconfig eth0 up
ip link set eth0 up
#ifconfig lan0 192.168.0.10/24
ip addr add 192.168.0.10/24 dev lan0
#ip -6 addr add 2001:0db8:0:xxxx::1/64 dev lan0
#ifconfig lan0 up
ip link set lan0 up
#route add default gw 192.168.0.5
ip route add 0.0.0.0/0 via 192.168.0.5
echo "nameserver 192.168.0.5" > /etc/resolv.conf
```

dauerhaft (auch nach reboot)

4.4.70:

```
nano /etc/network/interfaces
```

```
auto eth0
  iface eth0 inet static
  hwaddress ether 08:00:00:00:00:01
  address 192.168.0.10
  netmask 255.255.255.0
  gateway 192.168.0.5

auto eth1
  iface eth1 inet static
```

4.14:

```
auto eth0
iface eth0 inet manual
  pre-up ip link set $IFACE up
  post-down ip link set $IFACE down

auto lan0
iface lan0 inet static
  hwaddress ether 08:00:00:00:00:00 # if you want to set MAC manually
  address 192.168.0.10
  netmask 255.255.255.0
  gateway 192.168.0.5
  pre-up ip link set $IFACE up
  post-down ip link set $IFACE down

auto lan1
iface lan1 inet static
```

```
  hwaddress ether 08:00:00:00:00:01 # if you want to set MAC manually
  address 192.168.1.10
  netmask 255.255.255.0
  pre-up ip link set $IFACE up
  post-down ip link set $IFACE down

auto lan2
iface lan2 inet static
  hwaddress ether 08:00:00:00:00:02 # if you want to set MAC manually
  #...

auto lan3
iface lan3 inet static
  hwaddress ether 08:00:00:00:00:03 # if you want to set MAC manually
  #...

auto wan
iface wan inet static
  hwaddress ether 09:00:00:00:00:01 # if you want to set MAC manually
  #...
```

unter debian 9 funktioniert hwaddress nicht mehr, hier lässt sich das setzen der MAC so erreichen:

```
iface lan0 inet static
  address 192.168.0.10
  netmask 255.255.255.0
  gateway 192.168.0.5
#  pre-up ip link set $IFACE up
  pre-up ip link set $IFACE address 02:01:02:03:04:08 up
  post-down ip link set $IFACE down
```

Möglichkeit via UDEV von hier

```
$ cat /etc/udev/rules.d/00-static-mac-address.rules
ACTION=="add", SUBSYSTEM=="net", KERNELS=="1b100000.ethernet",
RUN+="/sbin/ip link set dev %k address ae:fc:de:ad:be:ef"
```

DHCP

```
allow-hotplug lan3
iface lan3 inet dhcp
```

Netzwerkbrücke (4.14)

Fix Me! ungetestet !

```
apt-get install bridge-utils
```

```
brctl addbr br0
brctl addif br0 lan1 lan2 lan3 #bridging lan1-lan3 (lan0 separat lassen für
vlan-tagging o.ä.)

root@bpi-r2:~# brctl show br0
bridge name     bridge id               STP enabled     interfaces
br0             8000.6acba7512bc1       no              lan1
                                                        lan2
                                                        lan3
```

/etc/network/interfaces:

```
iface br0 inet static
    address 192.168.40.1
    netmask 255.255.255.0
    bridge_ports lan1 lan2
    bridge_fd 5
    bridge_stp no
```

vlan

4.14:

/etc/network/interfaces:

```
auto lan3
iface lan3 inet manual

auto lan3.60
iface lan3.60 inet static
  address 192.168.60.10
  netmask 255.255.255.0
#  gateway 192.168.0.5
  pre-up ip link set $IFACE address 02:01:02:03:04:03 up #setting mac does
not work currently
```

Speicher

Partition		SD card	eMMC	Update(BPI-Tool)	Update(RAW command)
BootLoader	Header	0-2KB	Boot0: 0-2KB	**SD (On HOST PC or target board) :** *bpi-bootsel BPI-R2-720P-2k.img.gz /dev/mmcblkX* (for updating both uboot and preloader) **eMMC(Only on target board):** *echo 0 > /sys/block/mmcblkYboot0/force_ro* *bpi-bootsel BPI-R2-EMMC-boot0-DDR1600-0k-0905.img.gz /dev/mmcblkyboot0* (for updating bootloader to EMMC boot0) *bpi-bootsel BPI-R2-720P-2k.img.gz /dev/mmcblkY* (for updating Uboot to eMMC user block)	We just have RAW command to update EMMC preloader under uboot: *emmc write 1 84000000 0 200*
	Preloader	2-320KB	Boot0: 0-320KB		
	Uboot	320KB - 1MB	User Data Area: 320KB - 1MB		Write Uboot to SD or EMMC under uboot: mmc dev Z mmc write 84000000 280 200 Where Z is the number of disk, 0-emmc, 1-SD
	Reserved	1MB - 100MB	User Data Area: 1MB - 100MB		
File System Partition1(FAT32)		100MB - 356MB	User Data Area:100MB - 356MB		
File System Partition2(EXT4)		356MB - 7456MB	User Data Area:356MB - 7456MB	cd SD;bpi-update -c bpi-r2.conf -d /dev/<device>	

Quelle: http://forum.banana-pi.org/t/how-to-update-uboot-without-bpi-update/4023/2

es sieht so aus, als wenn /dev/mmcblk1, /dev/mmcblk1boot0 und /dev/mmcblk1boot1 unabhängige Geräte sind (bootx nicht Partitionen in /dev/mmcblk1)

preloader

for SD

muss am 2k-offset (0x800) geschrieben werden

```
sudo dd if=BPI-R2-EMMC-boot0-DDR1600-20190722-2k.img of=/dev/sdc bs=1k seek=2
```

(SD-Card benötigt zusätzlich MMC_BOOT & BRLYT header, siehe weiter unten)

for EMMC

muss am 0-offset (0x0) der boot0-Partition geschrieben werden

```
sudo dd if=BPI-R2-EMMC-boot0-DDR1600-20190722-0k.img of=/dev/mmcblk1boot0
```

Dateien von hier: https://github.com/BPI-SINOVOIP/BPI-files/tree/master/SD/100MB

SD

sdcard-bootsektor reverse-engineering
http://forum.banana-pi.org/t/boot-fails-with-self-build-u-boot/5460/20

http://forum.banana-pi.org/t/how-to-build-an-ubuntu-debian-sd-image-from-scratch/6805/8

SD-Headers

bpi-r2-head440-0k.img
bpi-r2-head1-512b.img

- SDMMC_BOOT-Signatur + Adresse des 2. Headers (0x00000200) - erste 440 byte (vor Partitionstabelle):

```
gunzip -c BPI-R2-HEAD440-0k.img.gz | dd of=/dev/loop8 bs=1024 seek=0
```

- BRLYT-Signatur + Preloader-Adresse (0x00000800):

```
gunzip -c BPI-R2-HEAD1-512b.img.gz | dd of=/dev/loop8 bs=512 seek=1
```

komplett

```
dd if=/dev/zero of=../bpi-r2-buster.img bs=1M count=7168
loopdev=$(losetup -f)
sudo losetup ${loopdev} ../bpi-r2-buster.img
echo $loopdev
sudo dd if=~/Downloads/BPI-R2-preloader-DDR1600-20190722-2k.img
of=${loopdev} bs=1k seek=2
sudo dd if=~/Downloads/BPI-R2-HEAD440-0k.img of=${loopdev} bs=1024 seek=0
sudo dd if=~/Downloads/BPI-R2-HEAD1-512b.img  bs=512 seek=1
sudo dd if=/path/to/u-boot/u-boot.bin of=${loopdev} bs=1k seek=320
sudo sfdisk ${loopdev} < ~/Downloads/parttable.dat
sudo partprobe ${loopdev}
ls ${loopdev}*
sudo mkfs -t vfat ${loopdev}p1
sudo mkfs -t ext4 ${loopdev}p2
sudo fatlabel ${loopdev}p1 BPI-BOOT
sudo e2label ${loopdev}p2 BPI-ROOT
```

install debian (from bootstrapped rootfs)

```
sudo mount ${loopdev}p2 /mnt/
sudo cp -r debian_buster_armhf/. /mnt/
#install kernel-modules to same partition
kernelpack=/path/to/bpi-r2_<version>_main.tar.gz
sudo tar -xzf ${kernelpack} -C /mnt/ --strip-components=1 BPI-ROOT
#install kernel to boot-partition
sudo umount /mnt
sudo mount ${loopdev}p1 /mnt/
sudo tar -xzf ${kernelpack} -C /mnt/ --strip-components=1 BPI-BOOT
#maybe create a uEnv.txt
sudo umount /mnt
```

```
sudo losetup -d ${loopdev}
#now write the image to card (make sure /dev/sdc is your sdcard-device and
no partition is mounted)
sudo dd if=../bpi-r2-buster.img of=/dev/sdc
sync
```

MMC-Utils

über die mmc-utils kann man aus einem laufenden System testen, ob die EMMC-Partitionierung stimmt (sollte 0x48 sein siehe partition-konfiguration_des_emmc_aendern).

```
./mmc extcsd read /dev/mmcblk1
....
Boot configuration bytes [PARTITION_CONFIG: 0x48]
....
```

ich habe die mmc-utils auch in mein Kernel-Repo übernommen (mit angepasstem Makefile für Cross-Compile)

laut einem Forum-Nutzer (siehe hier) lässt sich die partition config mit den mmc-utils auch schreiben

```
./mmc bootpart enable 1 1 /dev/mmcblk1
```

```
[18:02] root@bpi-r2:~# ./mmc extcsd read /dev/mmcblk1 | grep
PARTITION_CONFIG
Boot configuration bytes [PARTITION_CONFIG: 0x00]
[18:02] root@bpi-r2:~# ./mmc bootpart enable 1 1 /dev/mmcblk1
[18:03] root@bpi-r2:~# ./mmc extcsd read /dev/mmcblk1 | grep
PARTITION_CONFIG
Boot configuration bytes [PARTITION_CONFIG: 0x48]
```

Betriebssystem auf EMMC installieren

http://forum.banana-pi.org/t/bpi-r2-new-image-ubuntu-16-04-v1-2-1-bt-and-wifi-ap-mode-are-working-fine-2017-11-27/4291

1. partition-konfiguration_des_emmc_aendern
2. Schreibmodus für /dev/mmcblk1boot0 aktivieren:

   ```
   echo 0 > /sys/block/mmcblk1boot0/force_ro
   ```

3. Preloader von hier auf das boot-device schreiben:

   - ```
     gunzip -c BPI-R2-EMMC-boot0-DDR1600-0k-0905.img.gz | sudo dd
     of=/dev/mmcblk1boot0 bs=1024 seek=0
     ```
   - mit bpi-tools:

     ```
     bpi-bootsel BPI-R2-EMMC-boot0-DDR1600-0k-0905.img.gz
     /dev/mmcblk1boot0
     ```

4. kopieren des OS-Abbildes auf EMMC (device=/dev/mmcblk1):

   - ```
     unzip -p <XXX.img.zip> | pv | dd of=<device> bs=10M status=noxfer
     ```

- Alternative (mit bpi-tools):

```
bpi-copy  <XXX.img.zip> <device>
```

5. Ausschalten, SD entfernen und neu hochfahren

wenn sd-Karten-Abbild nicht auf emmc passt: Abbild verkleinern

manuelles kopieren des Betriebssystems

- für ein neues SD-Card-Image wird der Bootblock eines vorhandenen Images benötigt
  - erste 2k ohne preloader/uboot
    erstes MB mit preloader/uboot

```
gunzip bpi-r2-sd-boot*.img.gz
dd if=bpi-r2-sd-boot1m.img of=/dev/sdx
#ggf. partitonstabelle neu einlesen:
sfdisk -R /dev/sdx
#alternativ aus Paket parted
partprobe /dev/sdx
```

- uboot installieren:

```
dd if=BPI-R2-720P-2k.img of=/dev/mmcblk1 bs=1k seek=2 count=1022
```

- Partitionstabelle auf der SD exportieren
  parttable.dat
  und auf emmc einspielen:

```
root@bpi-r2:~# sfdisk -d /dev/mmcblk0 > parttable.dat
root@bpi-r2:~# sfdisk /dev/mmcblk1 < parttable.dat
```

- ggf. checken/vergrößern
- Dateisysteme anlegen (mkfs) für p1=vfat (apt-get install dosfstools) und p2=ext4

```
mkfs -t vfat /dev/mmcblk1p1
mkfs -t ext4 /dev/mmcblk1p2
```

- im bestehenden System mount-Punkte anlegen/konfigurieren

```
mkdir -p /mnt/emmc/boot
mkdir -p /mnt/emmc/root
nano /etc/fstab
# <file system>         <dir>           <type>  <options>
<dump>  <pass>
/dev/mmcblk0p2          /               ext4    errors=remount-ro
0       1
/dev/mmcblk0p1          /boot           vfat    defaults
0       0
/dev/mmcblk1p2          /mnt/emmc/root  ext4    errors=remount-
ro,noauto   0       1
```

```
/dev/mmcblk1p1          /mnt/emmc/boot  vfat    defaults,noauto
0       0
```

- mounten:

```
mount /mnt/emmc/root
mount /mnt/emmc/boot
```

- rootfs entpacken/rüberkopieren

```
rsync -aAXv --
exclude={"/dev/*","/proc/*","/sys/*","/tmp/*","/run/*","/mnt/*","/media
/*","/lost+found","/boot/*"} / /mnt/emmc/root/
```

- kernel (p1) und Module (p2) rüberkopieren

```
mkdir -p /mnt/emmc/boot/bananapi/bpi-r2/linux
cp /boot/bananapi/bpi-r2/linux/uImage /mnt/emmc/boot/bananapi/bpi-
r2/linux
mkdir -p /mnt/emmc/root/lib/modules/
cp -r /lib/modules/$(uname -r) /mnt/emmc/root/lib/modules/
```

- uboot auf die richtige Partition konfigurieren

```
sed 's/mmcblk0/mmcblk1/' /boot/bananapi/bpi-r2/linux/uEnv.txt >
/mnt/emmc/boot/bananapi/bpi-r2/linux/uEnv.txt
```

UBoot

erreichbar über debug-uart

```
    *** U-Boot Boot Menu ***
       1. System Load Linux to SDRAM via TFTP.
       2. System Load Linux Kernel then write to Flash via TFTP.
       3. Boot Linux from SD.
       4. System Load Boot Loader then write to Flash via TFTP.
       5. System Load Linux Kernel then write to Flash via Serial.
       6. System Load Boot Loader then write to Flash via Serial.
       7. Boot system code via Flash.
       U-Boot console       <<<<<<<<
    Press UP/DOWN to move, ENTER to select
```

Uboot erneuern

Der emmc-Befehl ist erst seit dem 29.September 2017 im uboot (Version: „U-Boot 2014.04-rc1 (Oct 16 2017 - 19:33:23)“)

U-Boot von GitHub kompilieren

```
sudo dd if=/dev/sdx of=bpi-r2-first10M.img bs=1M count=10 #Backup der ersten 10MB

SD/100MB$ gunzip BPI-R2-720P-2k.img.gz
SD/100MB$ sudo dd if=BPI-R2-720P-2k.img of=/dev/sdx bs=1k seek=2 count=1022 #unzipped img!

sudo dd of=/dev/sdx if=bpi-r2-first10M.img bs=1M count=10 #die ersten 10MB wiederherstellen (bei Fehler)
```

alternativ nur uboot (nach option 2 in build.sh, backup nicht vergessen):

```
sudo dd of=/dev/sdb if=u-boot-mt/u-boot.bin bs=1k seek=320
```

vorcompiliertes uboot-image kann auf meinem gDrive herunter geladen werden. oder die bin-Datei

hier

Quelle für Position des BPI-R2-720p-Images:
https://github.com/BPI-SINOVOIP/bpi-tools/blob/beb36af51a4b455a2a09ec9348a6efca1fe390cc/bpi-bootsel#L245

Zusammensetzung des Images:
https://github.com/BPI-SINOVOIP/BPI-R2-bsp/blob/d94f55022a9192cb181d380b1a6699949a36f30c/scripts/bootloader.sh#L19

```
TMP_FILE=${U}/${BOARD}.tmp
IMG_FILE=${U}/${BOARD}-2k.img
PRELOADER=$TOPDIR/mt-
pack/mtk/${TARGET_PRODUCT}/bin/preloader_iotg7623Np1_emmc.bin
UBOOT=$TOPDIR/u-boot-mt/u-boot.bin

(sudo dd if=$PRELOADER  of=${LOOP_DEV} bs=1k seek=2) >/dev/null 2>&1
(sudo dd if=$UBOOT  of=${LOOP_DEV} bs=1k seek=320) >/dev/null 2>&1
(dd if=${TMP_FILE} of=${IMG_FILE} bs=1k skip=2 count=1022 status=noxfer)
>/dev/null 2>&1
```

es wird also die compilierte uboot.bin (u-boot-mt/u-boot.bin) verwendet und diese liegt auf der SD-Karte an position 0x50000 (320k), der Preloader (mt-pack/mtk/bpi-r2/bin/preloader_iotg7623Np1_emmc.bin) liegt an position 0x800 (2k) der SD-Karte

uboot 2018-11

Mediatek hat patches für den BPI-R2 geposted…diese habe ich einem uboot-fork angewendet und eingerichtet (build.sh, config, default-Environment, …): https://github.com/frank-w/u-boot

Kernel von der SD-Karte lassen sich bereits starten (emmc sollte auch funktionieren), mittlerweile gibt es auch einen Ethernet-Treiber

falls nicht das default environment geladen wurde (buildargs):

```
env default -a
printenv
#saveenv
```

Liste der Befehle

zurück zum Menü mit dem Befehl „bootmenu“

```
BPI-IoT> help
```

```
?       - alias for 'help'
backup_message- print backup message.
base    - print or set address offset
bdinfo  - print Board Info structure
boot    - boot default, i.e., run 'bootcmd'
bootd   - boot default, i.e., run 'bootcmd'
bootm   - boot application image from memory
bootmenu- ANSI terminal bootmenu
bootp   - boot image via network using BOOTP/TFTP protocol
cmp     - memory compare
coninfo - print console devices and information
cp      - memory copy
```

```
crc32   - checksum calculation
echo    - echo args to console
editenv - edit environment variable
emmc    - eMMC sub system
env     - environment handling commands
esw_read- esw_read   - Dump external switch/GMAC status !!

exit    - exit script
false   - do nothing, unsuccessfully
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls   - list files in a directory (default /)
go      - start application at address 'addr'
help    - print command description/usage
image_blks- read image size from img_size or image header if no specifying
img_.
image_check- check if image in load_addr is normal.
iminfo  - print header information for application image
imxtract- extract a part of a multi-image
itest   - return true/false on integer compare
loadb   - load binary file over serial line (kermit mode)
loads   - load S-Record file over serial line
loadx   - load binary file over serial line (xmodem mode)
loady   - load binary file over serial line (ymodem mode)
loop    - infinite loop on address range
md      - memory display
mdio    - mdio   - Ralink PHY register R/W command !!

mm      - memory modify (auto-incrementing address)
mmc     - MMC sub-system
mmc2    - MMC sub system
mmcinfo - display MMC info
mtk_image_blks- read image size from image header (MTK format) located at
load_.
mw      - memory write (fill)
nm      - memory modify (constant address)
nor     - nor   - nor flash command

ping    - send ICMP ECHO_REQUEST to network host
printenv- print environment variables
reco_message- print recovery message.
reg     - reg   - Ralink PHY register R/W command !!

reset   - Perform RESET of the CPU
run     - run commands in an environment variable
saveenv - save environment variables to persistent storage
serious_image_check- seriously check if image in load_addr is normal.
setenv  - set environment variables
showvar - print local hushshell variables
sleep   - delay execution for some time
snor    - snor   - spi-nor flash command
```

```
source  - run script from memory
test    - minimal test like /bin/sh
tftpboot- boot image via network using TFTP protocol
true    - do nothing, successfully
uboot_check- check if uboot in load_addr is normal.
version - print monitor, compiler and linker version
```

Partition-Konfiguration des EMMC ändern

```
BPI-IoT> emmc --help
emmc - eMMC sub system
Usage:
emmc read part addr blk# cnt
emmc write part addr blk# cnt
emmc ecsd       - Dump ext csd
emmc pconf val  - Set Part Config val
BPI-IoT> emmc ecsd
```

emmc ecsd

```
============================================================
[EXT_CSD] EXT_CSD rev.              : v1.7 (MMCv5.0)
[EXT_CSD] CSD struct rev.           : v1.2
[EXT_CSD] Supported command sets    : 1h
[EXT_CSD] HPI features              : 1h
[EXT_CSD] BG operations support     : 1h
[EXT_CSD] BG operations status      : 0h
[EXT_CSD] Correct prg. sectors      : 0h
[EXT_CSD] 1st init time after part. : 3000 ms
[EXT_CSD] Min. write perf.(DDR,52MH,8b): 0h
[EXT_CSD] Min. read perf. (DDR,52MH,8b): 0h
[EXT_CSD] TRIM timeout: 0 ms
[EXT_CSD] Secure feature support: 55h
[EXT_CSD] Secure erase timeout  : 8100 ms
[EXT_CSD] Secure trim timeout   : 5100 ms
[EXT_CSD] Access size           : 3072 bytes
[EXT_CSD] HC erase unit size    : 512 kbytes
[EXT_CSD] HC erase timeout      : 300 ms
[EXT_CSD] HC write prot grp size: 8192 kbytes
[EXT_CSD] HC erase grp def.     : 0h
[EXT_CSD] Reliable write sect count: 1h
[EXT_CSD] Sleep current (VCC) : 7h
[EXT_CSD] Sleep current (VCCQ): 7h
[EXT_CSD] Sleep/awake timeout : 26214400 ns
[EXT_CSD] Sector count : e90000h
[EXT_CSD] Min. WR Perf.  (52MH,8b): 0h
[EXT_CSD] Min. Read Perf.(52MH,8b): 0h
[EXT_CSD] Min. WR Perf.  (26MH,8b,52MH,4b): 0h
```

```
[EXT_CSD] Min. Read Perf.(26MH,8b,52MH,4b): 0h
[EXT_CSD] Min. WR Perf.  (26MH,4b): 0h
[EXT_CSD] Min. Read Perf.(26MH,4b): 0h
[EXT_CSD] Power class: 0
[EXT_CSD] Power class(DDR,52MH,3.6V): 0h
[EXT_CSD] Power class(DDR,52MH,1.9V): 0h
[EXT_CSD] Power class(26MH,3.6V)    : 0h
[EXT_CSD] Power class(52MH,3.6V)    : 0h
[EXT_CSD] Power class(26MH,1.9V)    : 0h
[EXT_CSD] Power class(52MH,1.9V)    : 0h
[EXT_CSD] Part. switch timing    : 1h
[EXT_CSD] Out-of-INTR busy timing: 5h
[EXT_CSD] Card type       : 57h
[EXT_CSD] Command set     : 0h
[EXT_CSD] Command set rev.: 0h
[EXT_CSD] HS timing       : 0h
[EXT_CSD] Bus width       : 0h
[EXT_CSD] Erase memory content : 0h
[EXT_CSD] Partition config      : 0h      <<<<<<<<<<<<<<<<<<<<<<<<<<<<
falsche Partitionskonfiguration
[EXT_CSD] Boot partition size   : 4096 kbytes
[EXT_CSD] Boot information      : 7h
[EXT_CSD] Boot config protection: 0h
[EXT_CSD] Boot bus width        : 0h
[EXT_CSD] Boot area write prot  : 0h
[EXT_CSD] User area write prot  : 0h
[EXT_CSD] FW configuration      : 0h
[EXT_CSD] RPMB size : 512 kbytes
[EXT_CSD] Write rel. setting  : 1fh
[EXT_CSD] Write rel. parameter: 4h
[EXT_CSD] Start background ops : 0h
[EXT_CSD] Enable background ops: 0h
[EXT_CSD] H/W reset function   : 0h
[EXT_CSD] HPI management       : 0h
[EXT_CSD] Max. enhanced area size : 136h (2539520 kbytes)
[EXT_CSD] Part. support  : 7h
[EXT_CSD] Part. attribute: 0h
[EXT_CSD] Part. setting  : 0h
[EXT_CSD] General purpose 1 size : 0h (0 kbytes)
[EXT_CSD] General purpose 2 size : 0h (0 kbytes)
[EXT_CSD] General purpose 3 size : 0h (0 kbytes)
[EXT_CSD] General purpose 4 size : 0h (0 kbytes)
[EXT_CSD] Enh. user area size : 0h (0 kbytes)
[EXT_CSD] Enh. user area start: 0h
[EXT_CSD] Bad block mgmt mode: 0h
```

```
  ============================================================
  BPI-IoT> emmc pconf 0x48      <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
partition config ändern
  BPI-IoT> emmc ecsd            <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

```
verifizieren
  ============================================================
```

emmc ecsd (nachher)

```
[EXT_CSD] EXT_CSD rev.              : v1.7 (MMCv5.0)
[EXT_CSD] CSD struct rev.           : v1.2
[EXT_CSD] Supported command sets    : 1h
[EXT_CSD] HPI features              : 1h
[EXT_CSD] BG operations support     : 1h
[EXT_CSD] BG operations status      : 0h
[EXT_CSD] Correct prg. sectors      : 0h
[EXT_CSD] 1st init time after part. : 3000 ms
[EXT_CSD] Min. write perf.(DDR,52MH,8b): 0h
[EXT_CSD] Min. read perf. (DDR,52MH,8b): 0h
[EXT_CSD] TRIM timeout: 0 ms
[EXT_CSD] Secure feature support: 55h
[EXT_CSD] Secure erase timeout  : 8100 ms
[EXT_CSD] Secure trim timeout   : 5100 ms
[EXT_CSD] Access size           : 3072 bytes
[EXT_CSD] HC erase unit size    : 512 kbytes
[EXT_CSD] HC erase timeout      : 300 ms
[EXT_CSD] HC write prot grp size: 8192 kbytes
[EXT_CSD] HC erase grp def.     : 0h
[EXT_CSD] Reliable write sect count: 1h
[EXT_CSD] Sleep current (VCC) : 7h
[EXT_CSD] Sleep current (VCCQ): 7h
[EXT_CSD] Sleep/awake timeout : 26214400 ns
[EXT_CSD] Sector count : e90000h
[EXT_CSD] Min. WR Perf.  (52MH,8b): 0h
[EXT_CSD] Min. Read Perf.(52MH,8b): 0h
[EXT_CSD] Min. WR Perf.  (26MH,8b,52MH,4b): 0h
[EXT_CSD] Min. Read Perf.(26MH,8b,52MH,4b): 0h
[EXT_CSD] Min. WR Perf.  (26MH,4b): 0h
[EXT_CSD] Min. Read Perf.(26MH,4b): 0h
[EXT_CSD] Power class: 0
[EXT_CSD] Power class(DDR,52MH,3.6V): 0h
[EXT_CSD] Power class(DDR,52MH,1.9V): 0h
[EXT_CSD] Power class(26MH,3.6V)    : 0h
[EXT_CSD] Power class(52MH,3.6V)    : 0h
[EXT_CSD] Power class(26MH,1.9V)    : 0h
[EXT_CSD] Power class(52MH,1.9V)    : 0h
[EXT_CSD] Part. switch timing    : 1h
[EXT_CSD] Out-of-INTR busy timing: 5h
[EXT_CSD] Card type       : 57h
[EXT_CSD] Command set     : 0h
[EXT_CSD] Command set rev.: 0h
[EXT_CSD] HS timing       : 1h
[EXT_CSD] Bus width       : 0h
[EXT_CSD] Erase memory content : 0h
```

```
[EXT_CSD] Partition config      : 48h     <<<<<<<<<<<<<<<<<<<<<<<<<<<<
[EXT_CSD] Boot partition size   : 4096 kbytes
[EXT_CSD] Boot information      : 7h
[EXT_CSD] Boot config protection: 0h
[EXT_CSD] Boot bus width        : 0h
[EXT_CSD] Boot area write prot  : 0h
[EXT_CSD] User area write prot  : 0h
[EXT_CSD] FW configuration      : 0h
[EXT_CSD] RPMB size : 512 kbytes
[EXT_CSD] Write rel. setting  : 1fh
[EXT_CSD] Write rel. parameter: 4h
[EXT_CSD] Start background ops : 0h
[EXT_CSD] Enable background ops: 0h
[EXT_CSD] H/W reset function   : 0h
[EXT_CSD] HPI management       : 0h
[EXT_CSD] Max. enhanced area size : 136h (2539520 kbytes)
[EXT_CSD] Part. support  : 7h
[EXT_CSD] Part. attribute: 0h
[EXT_CSD] Part. setting  : 0h
[EXT_CSD] General purpose 1 size : 0h (0 kbytes)
[EXT_CSD] General purpose 2 size : 0h (0 kbytes)
[EXT_CSD] General purpose 3 size : 0h (0 kbytes)
[EXT_CSD] General purpose 4 size : 0h (0 kbytes)
[EXT_CSD] Enh. user area size : 0h (0 kbytes)
[EXT_CSD] Enh. user area start: 0h
[EXT_CSD] Bad block mgmt mode: 0h
============================================================
```

in neueren uboot-Versionen (2018):

http://forum.banana-pi.org/t/add-latest-u-boot-support-for-bpi-r2-bpi-r64-not-yet/6938/26

```
mmc partconf 0 1 1 0
```

System von Console starten

```
BPI-IoT> printenv
...
boot10=mmc init; run boot_normal; bootm
...
bootmenu_2=3. Boot Linux from SD.=run boot10
...
```

```
run boot10
```

Kernel angeben

in der BPI-BOOT/bananapi/bpi-r2/linux/uEnv.txt den parameter kernel anpassen:

```
#kernel=uImage
#kernel=uImage_4.14.33
kernel=uImage_4.9.92
```

dies hat den Vorteil, dass man einen neuen Kernel testen kann und notfalls auf den alten leicht wieder zurück kann (wenn diese 2 verschiedene Namen haben). Für Multiboot muss der uboot-code angepasst werden, da die uEnv.txt erst mit dem Menüpunkt „Boot Linux from SD“ geladen wird…vorher sieht man seine eigenen Variablen nicht.

uEnv.txt laden

Standardmäßig wird die uEnv.txt erst geladen wenn der Menüpunkt „Boot from SD“ ausgewählt wurde.

```
#Boot from emmc
enter to uboot-console
execute “mmc init 0”
execute “setenv partition 0:1”
execute “run loadbootenv”
execute “env import -t ${scriptaddr} ${filesize} ”

#Boot from SD:
enter to uboot-console
execute “mmc init 1”
execute “setenv partition 1:1”
execute “run loadbootenv”
execute “env import -t ${scriptaddr} ${filesize} ”
```

Quelle: http://forum.banana-pi.org/t/how-to-extend-the-uboot-menu/5415/7

da loadbootenv eine Variable ist, die nur im offiziellen bpi-r2-uboot definiert ist und u.a. im U-Boot-Upstream-repo nicht existiert müssen folgende Variablen definiert werden um die uEnv.txt + kernel zu laden

```
setenv scriptaddr 0x83000000
setenv bpi bananapi
setenv board bpi-r2
setenv service linux
setenv device mmc
setenv partition 1:1
setenv bootenv uEnv.txt
setenv loadbootenv fatload ${device} ${partition} ${scriptaddr}
${bpi}/${board}/${service}/${bootenv}
setenv importenv env import -t ${scriptaddr} ${filesize}
```

```
run loadbootenv
run importenv

printenv

setenv newboot "fatload ${device} ${partition} ${loadaddr}
${bpi}/${board}/${service}/${kernel}; bootm"
run newboot

#check for boot-device (emmc/sd)
setenv checksd fatinfo ${device} 1:1
setenv selectmmc "if run checksd; then echo Boot from SD ; setenv partition
1:1;else echo Boot from eMMC; setenv partition 0:1 ; fi;"

run selectmmc
```

nützliche Befehle

MMC

```
U-Boot> mmc list
mmc@11230000: 0 (eMMC)
mmc@11240000: 1 (SD)

#set mmc-device (1=sd,0=emmc)
U-Boot> mmc dev 1

#read current device
U-Boot> mmc dev
switch to partitions #0, OK
mmc1 is current device

U-Boot> mmcinfo
Device: mmc@11240000
Manufacturer ID: 1b
OEM: 534d
Name: 00000
Bus Speed: 50000000
Mode : SD High Speed (50MHz)
Rd Block Len: 512
SD version 2.0
High Capacity: Yes
Capacity: 7.6 GiB
Bus Width: 4-bit
Erase Group Size: 512 Bytes

sd-card (1) has 4-bit bus width, emmc (0) has 8-bit bus width
```

```
#partitionconfig
#mmc partconf dev [boot_ack boot_partition partition_access]
# - Show or change the bits of the PARTITION_CONFIG field of the specified
device
#example for mode 0x48 (needed for emmc-boot on bpi-r2)
U-Boot> mmc partconf 0
EXT_CSD[179], PARTITION_CONFIG:
BOOT_ACK: 0x1
BOOT_PARTITION_ENABLE: 0x1
PARTITION_ACCESS: 0x0

#setzen mit
U-Boot> mmc partconf 0 1 1 0
```

environment löschen/neu schreiben

```
#zur Sicherheit kann man sich das geschriebene env anschauen (vorher auf sd
wechseln,wenn davon gebootet):
BPI-R2> mmc dev 1
BPI-R2> mmc read ${scriptaddr} 800 10

MMC read: dev # 1, block # 2048, count 16 ... 16 blocks read: OK
BPI-R2> strings ${scriptaddr}
p��saskkernel=askenv kernelinput "enter uImage-name:";
...
#die ersten 4 bytes sind die CRC-Prüfsumme, danach geht das env los...

#environment löschen (ab block #800h=1MB/512b 16 Blöcke á 512b=8k => 10h)
verwendet eigenen mmc erase befehl
#BPI-R2> mmc erase 800 10
#wird mit eraseenv (patchwork) abgelöst
eraseenv

#default-environment laden
BPI-R2> env default -a;

#environment neu schreiben
BPI-R2> saveenv
Saving Environment to MMC... Boot From SD(id:1)

Writing to MMC(1)... OK
```

mehr zu den mmc-Kommandos hier

Verzeichnisauflistung

```
ls mmc 1:1 bananapi/bpi-r2/linux
#mit den Variablen aus meiner U-boot-Umgebung:
```

```
ls ${device} ${partition} ${bpi}/${board}/${service}
```

kernelabfrage

```
lskernel=ls ${device} ${partition} ${bpi}/${board}/${service};
askkernel=askenv kernelinput "enter uImage-name:";
boot0=run lskernel;run askkernel;if printenv kernelinput ;then setenv kernel
${kernelinput}; run newboot; fi
bootmenu_0=1. Enter kernel-name to boot from SD/EMMC.=run boot0
```

prüfen, ob datei existiert

```
checkenv=test -e ${device} ${partition}
${bpi}/${board}/${service}/${bootenv}
#will be evaluated to check if bananapi/bpi-r2/linux/uEnv.txt (device=mmc,
partition=1:1 for sdcard)
if run checkenv; then ...; else echo file not found; fi;
```

anderes uboot via TFTP nachladen

(benötigt CONFIG_CMD_CACHE)

```
BPI-R2> tftp 0x81E00000 ${serverip}:u-boot_2019.07-rc4-bpi-r2-dbg.bin
BPI-R2> icache off;dcache off
BPI-R2> go 0x81E00000
```

PCIe

uboot vor 2020-10 (meine version) hat einen Bug welcher beim „pci enum“ hängen bleibt, wenn keine Karte im pcie-slot gesteckt ist

```
BPI-R2> pci enum
BPI-R2> pci 0
Scanning PCI devices on bus 0
BusDevFun  VendorId   DeviceId   Device Class       Sub-Class
_____________________________________________________________
00.00.00   0x14c3     0x0801     Bridge device           0x04
00.01.00   0x14c3     0x0801     Bridge device           0x04
BPI-R2> pci 1
Scanning PCI devices on bus 1
BusDevFun  VendorId   DeviceId   Device Class       Sub-Class
_____________________________________________________________
01.00.00   0x14c3     0x7612     Network controller      0x80
BPI-R2> pci 2
```

```
Scanning PCI devices on bus 2
BusDevFun  VendorId   DeviceId   Device Class       Sub-Class
_____________________________________________________________
02.00.00   0x1b21     0x0611     Mass storage controller 0x01
BPI-R2> scsi scan
scanning bus for devices...
SATA link 0 timeout.
Target spinup took 0 ms.
AHCI 0001.0200 32 slots 2 ports 6 Gbps 0x3 impl SATA mode
flags: 64bit ncq stag led clo pmp pio slum part ccc sxs
  Device 0: (1:0) Vendor: ATA Prod.: ST750LM022 HN-M7 Rev: 2AR1
            Type: Hard Disk
            Capacity: 715404.8 MB = 698.6 GB (1465149168 x 512)
BPI-R2>
```

SATA

siehe pcie (pci enum + scsi scan) und dann via

```
ls scsi 0:1
```

auf die HDD zugreifen

USB

```
BPI-R2> usb start
starting USB...
Bus usb@1a1c0000: hcd: 0x1a1c0000, ippc: 0x1a1c4700
u2p:1, u3p:1
Register 200010f NbrPorts 2
Starting the controller
USB XHCI 0.96
Bus usb@1a240000: hcd: 0x1a240000, ippc: 0x1a244700
u2p:1, u3p:1
Register 200010f NbrPorts 2
Starting the controller
USB XHCI 0.96
scanning bus usb@1a1c0000 for devices... 1 USB Device(s) found
scanning bus usb@1a240000 for devices... 2 USB Device(s) found
       scanning usb for storage devices... 1 Storage Device(s) found
BPI-R2> usb tree
USB device tree:
  1  Hub (5 Gb/s, 0mA)
     U-Boot XHCI Host Controller

  1  Hub (5 Gb/s, 0mA)
  |  U-Boot XHCI Host Controller
  |
```

```
  +-2  Mass Storage (480 Mb/s, 200mA)
       USB      Flash Disk       906B030002F4

BPI-R2> ls usb 0:1
            efi/
  4767728   kernel

1 file(s), 1 dir(s)
```

dt overlays

https://forum.banana-pi.org/t/set-mac-address-on-boot/7224/4

Overlay muss mit -@ compiliert werden

```
dtc -@ -I dts -O dtb -o bpi-r2-mac.dtb bpi-r2-mac.dts
```

Sonst kommt die Meldung beim laden:

```
failed on fdt_overlay_apply(): FDT_ERR_NOTFOUND base fdt does did not have a
/symbols node make sure you’ve compiled with -@
```

Die Haupt-DTB muss auch mit -@ compiliert werden...dazu kann man beim lernel kompilieren die Variable DTC_FLAGS setzen:

```
export DTC_FLAGS=-@
```

Verfizieren lässt sich das mit fdtdump

```
fdtdump arch/.../boot/dts/.../board.dtb | grep -C3 __symbols__
```

Hier sollten die Namen der dts nodes auftauchen

Beim laden in uboot muss erst die haupt-dtb geladen werden und dann das overlay. Dafür habe ich in meinem uboot folgende Variablen definiert (ausführen via run $varname)

```
loadfdt=fatload ${device} ${partition} ${dtaddr}
${bpi}/${board}/${service}/dtb/${fdt}
loaddto=echo "loaddto:${dto}";fdt addr ${dtaddr};fdt resize 8192; setexpr
fdtovaddr ${dtaddr} + F000;fatload ${device} ${partition} ${fdtovaddr}
${bpi}/${board}/${service}/dtb/${dto} && fdt apply ${fdtovaddr}
```

Links

Patchwork Archiv

git

Ubuntu

debootstrap

Schritte um ein Ubuntu-system anzulegen (armhf als Architektur):
https://help.ubuntu.com/lts/installation-guide/armhf/apds04.html

```
sudo apt-get install qemu-user-static debootstrap binfmt-support

targetdir=$(pwd)/debootstrap_ubuntu_18.4
distro=bionic
arch=armhf

sudo debootstrap --arch=$arch --foreign $distro $targetdir

#wenn folgender Fehler kommt: E: Cannot install into target '...' mounted
with noexec or nodev
sudo mount -i -o remount,exec,dev /mounted_dir

sudo cp /usr/bin/qemu-arm-static $targetdir/usr/bin/
sudo cp /etc/resolv.conf $targetdir/etc
sudo distro=$distro chroot $targetdir
export LANG=C
/debootstrap/debootstrap --second-stage

#echo "deb-src http://archive.ubuntu.com/ubuntu $distro
main">>/etc/apt/sources.list
#echo "deb http://security.ubuntu.com/ubuntu $distro-security
main">>/etc/apt/sources.list
#echo "deb-src http://security.ubuntu.com/ubuntu $distro-security
main">>/etc/apt/sources.list
echo "deb http://ports.ubuntu.com/ubuntu-ports/ $distro
main">>/etc/apt/sources.list
echo "deb-src http://ports.ubuntu.com/ubuntu-ports/ $distro
main">>/etc/apt/sources.list
echo "deb http://ports.ubuntu.com/ubuntu-ports/ $distro-updates
main">>/etc/apt/sources.list
echo "deb-src http://ports.ubuntu.com/ubuntu-ports/ $distro-updates
main">>/etc/apt/sources.list
echo "deb http://ports.ubuntu.com/ubuntu-ports/ $distro-security
main">>/etc/apt/sources.list
echo "deb-src http://ports.ubuntu.com/ubuntu-ports/ $distro-security
main">>/etc/apt/sources.list

echo "bpi-r2-ubuntu" >/etc/hostname
#root-password setzen! sonst kein Login möglich
passwd
```

fstab/... konfigurieren wie bei debian  ubuntu 18.4 verwendet netplan.io als Standard Netzwerk-Framework bionic releasenotes

```
#chroot verlassen
exit
```

packen:

```
cd $targetdir
sudo tar cvpzf ../debootstrap_$distro.tar.gz .
```

System Auf SD-Karte installieren

SD-Karte vorbereiten

```
#in root-partition entpacken with
sudo tar -xpzf /path/to/debootstrap_$distro.tar.gz
sudo mkdir lib/modules/
cd lib/modules/
#kernel-module hier entpacken
```

temporäre Netzwerk-Konfiguration:

```
ip a
ip link set eth0 up
ip addr add 192.168.0.11/24 dev lan0 #freie ip/prefix deines Lan-Segments
ip link set lan0 up
ip route add default via 192.168.0.10 #ip deines Routers
echo "nameserver 192.168.0.10" >>/etc/resolv.conf #ip deines Router für DNS-Auflösung
```

Installieren von Paket “ifupdown” und Hinzufügen von “netcfg/do_not_use_netplan=true” zu den bootopts in der /boot/bananapi/bpi-r2/linux/uEnv.txt Nach einem Reboot wird das „alte“ System mit /etc/network/interfaces verwendet. nun wird die /etc/resolv.conf bei jedem Neustart zurückgesetzt

```
root@bpi-r2-ubuntu:~# ls -l /etc/resolv.conf
lrwxrwxrwx 1 root root 39 Jun 13 10:27 /etc/resolv.conf ->
../run/systemd/resolve/stub-resolv.conf
#löschen des Symlinks und ersetzen durch eine "normale" Datei mit den Einstellungen wie bei Debian
rm /etc/resolv.conf
echo "nameserver 192.168.0.10" >>/etc/resolv.conf
```

unter ubuntu 18.4 läuft bereits ein eigener dns-dienst, welcher deaktiviert werden muss (gefolgt von einem reboot oder via stop beenden), um z.B. DNSMasq laufen zu lassen (wie in meinem wifi.sh-script)

```
systemctl disable systemd-resolved
systemctl stop systemd-resolved
```

image erstellen

```
imgfile=/path/to/ubuntu-18.04-bpi-r2-preview.img
sudo dd if=/dev/sdx of=$imgfile
#Status von DD über anderes Terminal mit "sudo kill -SIGUSR1 $(pidof dd)" ermitteln
#image beschränken auf das Ende der Letzten Partition...Pfad zur img-Datei darf keine Leereichen enthalten
IFS=$'\t' #zur Sicherheit (ignoriere Leerzeichen im Pfad)
ENDOFDATA=$(fdisk -l "$imgfile" |tail -1|awk '{print $3}')
echo $ENDOFDATA
truncate --size=$[($ENDOFDATA+1)*512] $imgfile
#check size
ls -lh "$imgfile"

#Image weiter manipulieren
loopdev=$(losetup -f)
sudo losetup $loopdev $imgfile
sudo partprobe $loopdev
sudo mount ${loopdev}p2 /mnt
ls /mnt
#...
#Freien Speicher mit 0 füllen für besseres Packen
sudo sh -c 'cat /dev/zero >/mnt/null.dat'
sudo rm /mnt/null.dat
sudo umount /mnt

#image packen
gzip $imgfile
md5sum $imgfile.gz > $imgfile.gz.md5
```

Ich habe mein Image auf mein gdrive hochgeladen zum testen (ubuntu-18.04-bpi-r2-preview.img.gz)

SSH

ssh-server ist in meinem image installiert ubuntu-18.04-bpi-r2-preview.img.gz from my gdrive, aber root-login muss aktiviert werden

```
echo "PermitRootLogin yes" >> /etc/ssh/sshd_config
service sshd restart
```

es sollten neue host-keys für ssh erzeugt werden...

```
#alte Schlüssel löschen
```

```
rm /etc/ssh/ssh_host_*
#sshserver-paket neu konfigurieren
dpkg-reconfigure openssh-server
```

VLAN-Unterstützung

- Port-Trennung (4x LAN) funktioniert mit 4.14, vlan-unterstützung kann über mein github-repo getestet werden
- keine Unterstützung in 4.4.70 forum

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

wpasupplicant muss deinstalliert, hostapd+dnsmasq installiert werden:

```
apt-get remove wpasupplicant

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 Verbindungversuche 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-BOOT/bananapi/bpi-r2/linux/uImage /media/$USER/BPI-
BOOT/bananapi/bpi-r2/linux/uImage
sudo cp -r SD/BPI-ROOT/lib/modules /media/$USER/BPI-ROOT/lib/
sudo cp linux-mt/drivers/net/wireless/mediatek/mt76/firmware/*
/media/$USER/BPI-ROOT/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

PCIe-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 (regulary 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)

X-Server

```
sudo apt-get install xserver-xorg lxde xfonts-base (policykit-1) (xinit)
```

installiert einen einfachen xserver

log:

```
less /var/log/Xorg.0.log
```

auflösungen auflisten (fbdev):

```
cat /sys/class/graphics/fb0/modes
```

„no session for pid“ kommt nur, wenn der xserver mit startx gestartet wird. Nach dem ersten Reboot wid die lightdm-Anmeldemaske geladen und nach der Anmeldung kommt der Fehler nicht mehr