

Banana Pi Router - BPi-R1

Manual for HW setup and basic router
functionalities

Table of contents

[Information](#)
[Instruction about how to use this manual](#)
[Prerequisite for beginners](#)
 [with LINUX](#)
 [with WINDOWS](#)
[Prerequisite](#)
[nano text-editor tuning](#)
[Backup your Image](#)
[Configuration files](#)
 [swconfig](#)
 [wpa supplicant](#)
 [isc-dhcp](#)
 [Checks:](#)
 [interfaces](#)
 [Checks:](#)
[IPtables](#)
 [To secure a little bit the router:](#)
[hostapd](#)
 [Checks:](#)
[before you go for compilation](#)
 [compile from Git](#)
[hostapd compilation](#)
[Compile RealTek WiFi driver](#)
[Internet resources](#)

version	date	change
0.1	14. May	initial document release
0.2	08. June	extended warning overheating
0.3	09. Aug	updated Information & Internet resources
0.4	06. Oct	isc-dhcp & interfaces updated

Information

The BPi-R1 is a special version in the family of Banana's and to make use of its router functionality you have to configure this first.

The A20's uses the BCM53125 (7 port ethernet chip) as PHY.

So physically the SoC (A20) and all 5 external ethernet-ports are interconnected through a simple switch, 6 of them connected (A20 + 5 GBit Ethernet ports).

Switch port 0, 1, 2, 4 will be the LAN, port 3 the WAN. [To see this with some graphic](#)

If you want to use it like a classical router with WAN and LAN you must separate it in the configuration. Therefore you have to create a virtual-LAN (VLAN) and a bridge (br0) to connect the VLAN and WLAN. No! firewall configuration with these steps.

Let's think about if we have everything:

Software:

1. driver wireless RealTek
2. driver BCM53125
3. 8021q = VLAN module
4. swconfig = to control BCM53125 switch (from OpenWrt) /usr/sbin
5. bridge-utils = bring WiFi & Ethernet together
6. isc-dhcp = dhcp server
7. hostapd = WiFi security /usr/sbin/
8. hostapd_cli
9. IPtables = the Kernel needs to know how to route the traffic

Configuration:

1. swconfig
2. isc-dhcp
3. hostapd
4. interfaces
5. IPtables

Goal of this document:

1. Router functionalities, all ports LAN & WLAN shall be connected to each other.

I read somewhere that the MALI GPU does not play nice with other Kernel than 3.4 (Feb. 2015)

About the manual, unless I highlight the text, you have to copy & paste line after line.

Instruction about how to use this manual

By accident you might insert or delete something, then simply press: ctrl + z

Prerequisite for beginners

You can omit these lines and jump to the next page, if you already know how to write the data on the SDcard.

We will Install a fresh [Igor's 2.7 k3.4.107.zip](#) on your SDcard (based on Debian 7 wheezy), my old SDcard from 2010 wouldn't boot. Make sure you have two to test in case one doesn't work.

with LINUX

After you have downloaded the .zip file from the link above, you will create a folder on your PC and unzip /unrar the files with your distributions archive program.

After this, which will take some time as it is quite a large zip, you go to the folder with the extracted files, you can try in your file-manager a 'right-click' to open a terminal straight from this location.

Or simply open a terminal and go to the directory in which you keep the extracted files.

your extract: Lamobo-r1_Debian_2.7_wheezy_3.4.107.raw

Insert your SD card into the card reader

write the image to the SDcard;

As you can see red highlighted, you have to tell the PC to which device /SDcard it shall write to, the number of the partition is not necessary. To find this information you can try the command: `df -h`

```
sudo dd if=Lamobo-r1_Debian_2.7_wheezy_3.4.107.raw of=/dev/sdj bs=1M && sync
```

after you entered the password, you will not see a lot. It takes ca. 2min to write. When it has finished, it will write more information in the Terminal.

So you have the time to get some water, the following steps are a complete walk-through and I recommend to work through without interruption. At the end you have a working system.

When `dd` has finished you can remove the card, the `sync` command made sure that everything from RAM has been written to the devices. Now put the SDcard into your BPI-R1 and turn on the power.

with WINDOWS

After you have downloaded the .zip file from the link above, you will create a folder on your PC and unzip with 'right-click' extract.

your extract:

Lamobo-r1_Debian_2.7_wheezy_3.4.107.raw and the imagewriter.exe

Insert your SD card into the card reader, start the imagewriter.exe

Prerequisite

Install a fresh [Igor's 2.7 k3.4.107.zip](#) on your SDcard (based on Debian 7 wheezy), put the SDcard into your BPi-R1 (my old SDcard from 2010 wouldn't boot. Make sure you have two to test in case one doesn't work) and connect the single WAN port to your current Router.

Go to your Router's Webinterface and find out the IP-Address of BPi-R1. Based on the MAC-Address I then fixed the IP-Address so it is always the same.

In Linux open a terminal and login with: `ssh root@IP-Address - password: 1234`

In Windows you need [PuTTY](#) if you don't know PuTTY, look here [YouTube](#).

Check internet connection: `ping www.google.com` (stop with `Ctrl + c`).

```
apt-get update
```

```
apt-get upgrade
```

```
dpkg -l hostapd isc-* dnsmasq vlan iptables-persistent
```

```
dpkg -l bridge-utils (un = install it)
```

```
apt-get install nethogs bridge-utils isc-dhcp-server
```

To change some standard settings, see the [FAQ on Igor's Website](#)

```
shutdown -r now (reboot) login with new password ;)
```

```
ifconfig (LAN)
```

```
iwconfig (WLAN)
```

```
swconfig dev switch0 show (see what you have)
```

```
route -n (do this on the router and on the PC see if it fits – troubleshooting)
```

```
dmesg | grep 8021 (VLAN Support)
```

```
modinfo 8021q
```

```
lsmod (Show the status of modules in the Linux Kernel)
```

```
ps -e (to see every process the system is running)
```

```
service networking restart restart Network-Services
```

To copy files use an FTP-client (i.e. gFTP) on your PC and SSH2 or SFTP on Banana Pi

Check your log files:

```
less /var/log/syslog (type capital G to jump to last entry - q to exit)
```

```
less /var/log/messages
```

```
tail -f /var/log/syslog (open an additional ssh and let it run to see the current logfile)
```

You can check the dpkg log files to see which packages you installed recently:

```
grep 'install ' /var/log/dpkg.log
```

list all partitions

```
lsblk -o NAME,FSTYPE,SIZE,MOUNTPOINT,LABEL
```

nano text-editor tuning

We will use the nano text editor often and it helps to have different colors in the config files.

You can turn it **on** or **off** at anytime with: Alt + y

create nano /usr/share/nano/conf.nanorc and add these lines:

```
# config file highlighting

syntax "conf"
"(\.(conf|config|cfg|cnf|rc|lst|list|defs|ini|desktop|mime|types|preset|
cache|seat|service|htaccess)$|(^|/)(\w*crontab|mirrorlist|group|hosts|pa
sswd|rpc|netconfig|shadow|fstab|inittab|inputrc|protocols|sudoers|interf
aces|swconfig|hostapd)$|conf.d/|.config/)"

# default text
color magenta "^.*$"
# special values
icolor brightblue "^(|s|=)(default|true|false|on|off|yes|no)(|s|$)"
# keys
icolor cyan "^s*(set\s+)?[A-Z0-9_\/\.\%\@+-]+\s*([:|>|<|>)"
# commands
color blue "^s*set\s+<"
# punctuation
color blue "[.]"
# numbers
color red "^(|s|[[/:|<>(){}=,|\\])[-+]?[0-9](\\.?[0-9])%?(|$|>)"
# keys
icolor cyan "^s*(\\$if )?([A-Z0-9_\/\.\%\@+-]|s)+="
# punctuation
color blue "/"
color green "(\\|[(|<>[{};,:=])"
color brightwhite "^(|s|\\[|\\(|\\:|\\s)*-(|s|$)"
# section headings
icolor brightyellow "^s*(\\([([A-Z0-9_\\.\\-]|s)+\\))+s*$"
color brightcyan "^s*((Sub)?Section\\s*(=|>)|End(Sub)?Section\\s*$)"
color brightcyan "^s*\\$(end)?if(|s|$)"
# URLs
icolor green "\\b(([A-Z]+://|www[.])[A-Z0-9/:#?&$=\\.\\-]+)(\\b|$| )"
# XML-like tags
icolor brightcyan
"</?\\w+((\\s*\\w+\\s*=)?\\s*("[^"]*"|'['']*'|!)?[A-Z0-9_:/]))*(\\s*/)?>"
# strings
color yellow "\\(\\.|[\\^"])*\\\"" "'(\\.|[\\^'])*'"
# comments
color white "#.*$"
color blue "^s*##.*$"
color white "^;.*$"
color white start="<!--" end="-->"
```

Ctrl + o to save the file, Ctrl + x to close the application.

Then you include this configuration to nano. Go to the last line of `nano /etc/nanorc` file and add:

```
## Configuration files (catch-all syntax)
include "/usr/share/nano/conf.nanorc"
```

In the first codeline, at the end you can add more filename if you miss some.

And now it is much easier to read configuration files, as long as you have no inabilities to see colors.

Backup your Image

with Linux

`shutdown -h now` to turn off the R1 - place the SDcard in your PCs CardReader.

`dd` will not make an image of the size of the sum of the partition.

It is fed input data, it delivers output data. It does no interpretation at all.

```
fdisk -l
```

i.e. Igor's on 16GB MicroSD

Gerät	boot.	Begin	End	Block	Id	System
/dev/sdj1		2048	4443007	20480	83	Linux

sector size is given, 512 bytes

go to the directory where you want to store the backup, then:

```
dd if=/dev/sdj of=igors_2.7_wheezy_3.4.107_bkup_01.img bs=512
```

a good chance to grab a bite, it will take around 10min

When finished remove the MicroSD from your PCs CardReader, put it in the R1 and turn on power to boot the device.

Configuration files

swconfig

SWITCH configuration - by standard Igor keeps the swconfig in the 'interfaces-file', you get a better overview by keeping it in a separate file. So you create this one:

```
nano /etc/network/if-pre-up.d/swconfig
```

```
#!/bin/sh
#-----#
# BPI-R1 VLAN configuration #
#-----#
#
# This will create the following ethernet ports:
# - eth0.101 = WAN (single port)
# - eth0.102 = LAN (4 port switch)
#
# You have to adjust your /etc/network/interfaces
#
# Comment out the next line to enable the VLAN configuration:
# exit 0
```

```
ifconfig eth0 up
```

```
swconfig dev eth0 set reset 1
swconfig dev eth0 set enable_vlan 1
swconfig dev eth0 vlan 101 set ports '3 8t'
swconfig dev eth0 vlan 102 set ports '4 0 1 2 8t'
swconfig dev eth0 set apply 1
```

note: (wildcat_paris armbian forum)

with my 3t

Make the file executable: `chmod +x /etc/network/if-pre-up.d/swconfig`

Remove all rights from other VLAN files, so it is read-only:

```
chmod 0444 /etc/network/if-post-down.d/vlan
```

```
chmod 0444 /etc/network/if-pre-up.d/vlan
```

wpa supplicant

It is suitable for both desktop/laptop computers and embedded systems.

Supplicant is the IEEE 802.1X/WPA component that is used **in the client stations**.

security mechanism - wpa_supplicant is designed to be a "daemon" program that runs in the background and acts as the backend component controlling the wireless connection.

isc-dhcp

This is the server from the Internet Software Consortium's implementation of DHCP.

configuration file `nano /etc/default/isc-dhcp-server`

```
INTERFACES="br0"
```

configuration file `nano /etc/dhcp/dhcpd.conf`

option definitions common to all supported networks...

```
#option domain-name "example.org";
```

```
#option domain-name-servers ns1.example.org, ns2.example.org;
```

the numbers represent seconds

```
default-lease-time 3601;
```

```
max-lease-time 44000;
```

Use this to send dhcp log messages to a different log file (you also
have to hack syslog.conf to complete the redirection).

```
log-facility local7;
```

This is a very basic subnet declaration, with Google for DNS (8.8.8.8)

```
subnet 192.168.9.0 netmask 255.255.255.0 {  
    range 192.168.9.150 192.168.9.250;  
    option routers 192.168.9.2;  
    option domain-name-servers 8.8.8.8;  
}
```

I use this instead of above, 192.168.10.1 is my NETGEAR router connected to the modem.

```
subnet 192.168.9.0 netmask 255.255.255.0 {  
    range 192.168.9.150 192.168.9.250;  
    option routers 192.168.9.2;  
    option domain-name-servers 192.168.10.1;  
}
```

Checks:

```
service isc-dhcp-server status
```

Location of the IP-Address lease file:

```
cd /var/lib/dhcp/
```

```
less dhclient.eth0.101.leases
```

```
less dhcpd.leases (WLAN, actually br0)
```

interfaces

by standard Igor keeps several interface files for different devices. We will now remove all these files and create a new one:

```
cd /etc/network/
rmdir interfa*
rm interfa*
```

Bridge, will be used to bridge the Ethernet and Wireless interfaces.

Lines starting with `#` are ignored. Note that end-of-line comments are NOT supported, comments must be on a line of their own.

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

```
# interfaces(5) file used by ifup(8) and ifdown(8)
# auto = Start the interface(s) at boot.
# iface = interface
auto lo
iface lo inet loopback
# allow-hotplug <interface> - Start the interface when a "hotplug" event is detected.
# In real world, this is used on the same situations of auto but the difference is that it will wait an
event like "plug the cable" on ethernet interfaces.
allow-hotplug eth0
allow-hotplug wlan0
```

```
# ** LAN config VLAN 1** #####
# receive IP-Address from your DSL- or cablemodem
auto eth0.101
    iface eth0.101 inet dhcp
```

```
# * LAN config VLAN 2, for the 4 ports * #####
# generate IP-Address for connected devices
auto eth0.102
    iface eth0.102 inet manual
```

```
# ** WLAN config ** #####
# generate IP-Address for connected devices
auto wlan0
    iface wlan0 inet manual
# post-down, actions taken right after the iface is down
    post-down rm -rf /run/hostapd/wlan0
```

```
# ** Bridge config ** #####
auto br0
    iface br0 inet static
    bridge_ports eth0.102 wlan0
    bridge_waitport 0
    address 192.168.9.2
    network 192.168.9.0
    netmask 255.255.255.0
# adjust the gateway to your configuration, my gateway is my NETGEAR router. Comment (#) if your
R1 is directly connected to the DSL- or cablemodem.
    gateway 192.168.10.1
```

Make the file executable: `chmod +x /etc/network/interfaces`

Checks:

```
netstat -uap    Active Internet connections
ip r
ip link (successor of ifconfig)
ifconfig -a
ifconfig eth0.101
brctl show
brctl showmacs br0
less /var/log/syslog (type capital G to jump to last entry - q to exit)
less /var/log/messages
```

IPtables

iptables are used to set up, maintain, and inspect the tables of IPv4 packet filter rules in the Linux kernel. Several different tables may be defined.

Each table contains a number of built-in chains and may also contain user-defined chains.

Each chain is a list of rules which can match a set of packets.

Each rule specifies what to do with a packet that matches. This is called a 'target', which may be a jump to a user-defined chain in the same table. `man iptables`

<http://ubuntuforums.org/showthread.php?t=716192>

```
iptables --list-rules (see what you have)
iptables --list --line-numbers
```

For bridged interfaces you need these:

```
iptables -A FORWARD -i br0 -s 192.168.9.0/255.255.255.0 -j ACCEPT
iptables -A FORWARD -i eth0.101 -d 192.168.9.0/255.255.255.0 -j ACCEPT
iptables -A POSTROUTING -o eth0.101 -t nat -j MASQUERADE
mkdir -m 755 /etc/iptables/
iptables-save > /etc/iptables/rules.v4
```

```
nano /etc/network/if-pre-up.d/iptablesload
```

```
#!/bin/sh
iptables-restore --counters < /etc/iptables/rules.v4
exit 0
```

Make the file executable: `chmod +x /etc/network/if-pre-up.d/iptablesload`

And we need to turn on IP forwarding in the Kernel:

```
less /proc/sys/net/ipv4/ip_forward (0)
echo 1 > /proc/sys/net/ipv4/ip_forward
less /proc/sys/net/ipv4/ip_forward (1)
```

to make this change permanent edit `nano /etc/sysctl.conf` and uncomment line `net.ipv4.ip_forward=1`

To secure a little bit the router:

- these steps are not necessary, but you can if you want to -

Be careful to understand what a rule does, before you apply it. If you are connected to your Banana via SSH you might lose your connection and have to edit files directly with HDMI monitor and USB-Keybord or put the SDcard in your PC to correct a wrong rule.

These lines allow incoming connection to `loopback` and `br0` (the 4 LAN and WLAN if you followed this tutorial) :

```
iptables -I INPUT 1 -i lo -j ACCEPT
iptables -I INPUT 1 -i br0 -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables-save > /etc/iptables/rules.v4
```

These lines defines the default policy.

In this case to DROP **every** packet that is not explicitly told to be accepted.

So before you apply these rules, read below: open a specific port

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
```

If you need, you can open a specific port:

```
iptables --append INPUT -i eth0.101 -p 1 --dport 2 -j ACCEPT
```

allow incoming SSH connection to WAN port and give the rule priority (no WiFi).

```
iptables --append INPUT -i eth0.101 -p tcp --dport 22 -j ACCEPT
```

1: tcp or udp

2: the port to open

to check your configuration:

```
iptables --list-rules
iptables --list --line-numbers
iptables -L -t nat
```

hostapd

Host AccessPoint Daemon

Nice information: <http://wiki.gentoo.org/wiki/Hostapd>

hostapd; is a daemon that handles access point management and authentication

Which hostapd do you have installed:

`hostapd -v` (returns the version)

by standard Igor keeps two hostapd files for different devices. We will now remove all these files and create a new one:

```
cd /etc/
```

```
rm hostap* it will only remove the .conf file, not the directory.
```

```
nano /etc/default/hostapd
```

activate the daemon, set the absolute path to your hostapd configuration

```
DAEMON_CONF="/etc/hostapd/hostapd.conf" # point to configuration file
```

```
nano /etc/init.d/hostapd
```

```
DAEMON_SBIN=/usr/sbin/hostapd-rt
```

```
DAEMON_CONF=/etc/hostapd/hostapd.conf
```

hostapd.conf

```
nano /etc/hostapd/hostapd.conf
```

```
interface=wlan0
```

```
bridge=br0
```

```
driver=rtl871xdrv
```

```
# This advertises the country_code and the set of allowed channels and transmit power
```

```
# country_code=DE
```

```
# ieee80211d=1
```

```
# Maximum data rate 54Mbps in 802.11g and 300Mbps in 802.11n
```

```
# The RTL8192CU provides simple legacy and 20MHz/40MHz co-existence mechanisms to ensure backward and network compatibility. Maximum PHY data rate up to 144.4Mbps using 20MHz bandwidth, 300Mbps using 40MHz bandwidth
```

```
# if you don't have a wireless N capable device = 0
```

```
ieee80211n=1
```

```
# 2,4GHz bandwidth
```

```
channel=4
```

```
hw_mode=g
```

```
wmm_enabled=1
```

```
ht_capab=[SHORT-GI-20][SHORT-GI-40][HT40]
```

```
# Identification and Login
```

```
ssid=bpir1
```

```
wpa_passphrase=supersecret
```

```
wpa_key_mgmt=WPA-PSK
```

```
wpa_pairwise=TKIP CCMP
#rsn_pairwise=CCMP
auth_algs=1
wpa=2

macaddr_acl=0
beacon_int=100
max_num_sta=8
ignore_broadcast_ssid=0
ctrl_interface=/run/hostapd
noscans=1
```

Make the file executable: `chmod +x /etc/hostapd/hostapd.conf`

Checks:

check your config file: `hostapd-rt -dd /etc/hostapd/hostapd.conf`

is the service running? `service hostapd status`

`ps ax | grep [h]ostapd | grep -v grep`

`less /var/log/daemon.log`

`find / -name hostapd` (to find all files with this name)

scan networks: `iwlist wlan0 scan`

show available channels: `iwlist wlan0 channel`

find least used channel: `cat /proc/net/rtl819xC/wlan0/best_channel`

How much traffic do you have per process?

`nethogs eth0.101 q to quit`

`shutdown -h now ->` it's time for a **Backup** now, before you really start to gamble ;)

i.e.:

`dd if=/dev/sd{j of=igors_2.7_wheezy_3.4.107_bkup_02.img bs=512`

!! You now have a working BPi-R1 !! and you have a backup.

One last word, the housing is tight - make sure the temperature will not rise to high. The board should stand vertically, instead of horizontally (with the smallest face at the bottom), to allow convection.

The acrylic case takes away any little ventilation possible and makes things even worse. Out of the 4 lateral sides of the acrylic case, I just kept the 2 smallest faces, used to mount the antennas, but removed others to allow for more ventilation.

before you go for compilation

```
apt-get update
apt-get install build-essential git
```

Install the Kernel source:

```
apt-get install linux-headers-$(uname -r)
```

symlink, copy both line as one (it is a long command)

```
ln -s /usr/src/linux-headers-3.4.107-lamobo-r1/arch/arm/
/usr/src/linux-headers-3.4.107-lamobo-r1/arch/armv7l
```

```
cd /usr/src/linux-headers-3.4.107-lamobo-r1/
make modules_prepare (may return an error)
make headers_check
make headers_install
make scripts/kallsyms
```

compile from Git

```
create a directory: mkdir -m 755 /home/git
cd /home/git
git clone https:  the git you want to get
```

```
make
make dkms
```

hostapd compilation

It is written in several posts, that the DEBIAN RealTek driver 8192cu v3.3.2 doesn't perform well. Neither does the DEBIAN 1.0 hostapd play nice - it won't let you connect (a daemon that handles access point management and authentication (wpa2)).

So you would get the original hostapd & driver from RealTek, compile it and you are done.

Unfortunately, according to several posts - this driver v4.0.2 does may be work in combination with a modified hostapd well, but not with 0.8 included.

So what we need is a working hostapd for WiFi!

You can find several source files on Git. The following line's suppose you've done the preparation above: 'before you go for compilation'

jekader - just hostapd-2.4 for the RTL driver v4.0.2

[jekader's mixture](#)

Jens hostapd (and the RealTek driver is enough?)

<http://www.jenssegers.be/blog/43/realtek-rtl8188-based-access-point-on-raspberry-pi>

<https://github.com/jenssegers/RTL8188-hostapd>

create the driver with this `git` and use the hostapd from RealTek

[Wannabe Nerd Randomness](#)

<https://github.com/dz0ny/rt8192cu> v4.0.2 may be modified

kernel.org about rtl819x

<https://wireless.wiki.kernel.org/en/users/drivers/rtl819x>

[driver rtl8188C_8192C_8192D_usb_linux_v3.4.2](#)

Compile RealTek WiFi driver

While you are testing, do a reboot from now and then `shutdown -r now` sometimes it helps ;)

To get hostapd running with rtl871xdrv you need the proprietary kernel module 8192cu (without rtl*)

For the proprietary 8192cu you have to set driver=rtl871xdrv

```
nano /etc/hostapd/hostapd.conf
```

```
modinfo 8192cu          (... v3.3.2, better v3.4 or v4.0)
lsmod | grep 8192       (8192cu          453340 0)
dmesg | grep rtl81      (usbcore: registered new interface driver rtl8192cu)
```

```
cd /etc/modprobe.d/
nano blacklist.conf
blacklist rtl8192cu (add this line)
```

The binary from Debian's repository doesn't include the latest driver so we have to download it from Realtek's website and compile it ourselves. I downloaded it on my PC, unzipped it and with gFTP I copied it to the 'home' directory on the SDcard - or simply put the SDcard into your PC to copy.
<http://www.realtek.com/downloads/downloadsView.aspx?Langid=1&PNid=48&PFid=48&Level=5&Conn=4&DownTypeID=3&GetDown=false&Downloads=true#2772> (10MB)

The following line's suppose you've done the preparation above: 'before you go for compilation'

```
mkdir -m 755 /home/RTL81xxxxx
cd /home/RTL81xxxxx
ls -la
chmod 775 install.sh
./install.sh (get a tea or coffee, takes about 2min)
```

The driver is compiled and can be found:

```
/lib/modules/3.4.107-lamobo-r1/kernel/drivers/net/wireless/
```

To rename the existing, old driver, follow these steps:

```
cd /lib/modules/3.4.107-lamobo-r1/kernel/drivers/net/wireless/rtl8192cu/
```

copy three lines as one (it is a long command, rename the current (old) driver)

```
mv
/lib/modules/3.4.107-lamobo-r1/kernel/drivers/net/wireless/rtl8192cu/8192cu.ko 8192cu.ko.old
```

depmod creates a list of module dependencies by reading each module under /lib/modules/version and determining what symbols it exports and what symbols it needs.

```
/sbin/depmod -a 3.4.104-bananian
```

of course, you don't have to, but consider how many hours I have invested into this document.
So if you have login for lemaker.org - I would appreciate it

if this manual was helpful for you, please leave a 'I like' at
<http://forum.lemaker.org/forum.php?mod=viewthread&tid=15458&fromuid=72927>

Internet resources

BananaPi as a fileserver -- some personal thoughts and experiences.

<http://forum.lemaker.org/thread-7102-1-1-.html>

Banana Pi R1 as firewall /router - short but well explained in each detail

<http://rolfebozier.com/archives/51>

Configuring a Linux home internet gateway - well explained what you need each part for

http://www.freesoftwaremagazine.com/articles/home_internet_gateway

Realtek RTL8188 based access point on Raspberry Pi

<http://www.jenssegers.be/blog/43/realtek-rtl8188-based-access-point-on-raspberry-pi>

Banana Pro – WLan-AccessPoint (german)

<http://frank-mankel.de/?p=401>

Netzwerk unter Linux (german)

<http://www.stefanux.de/wiki/doku.php/linux/netzwerke>

Housing for BPi - R1

on [Google+](#) add files to the print and diagrams in one month (approx end of April '15)

Files for 3D printing available on <http://www.thingiverse.com/thing:735741> this is an early version.

Access Point - Bash Script

<http://crunchbang.org/forums/viewtopic.php?id=35427>