

Projektarbeit
in der Fakultät
Informatik

Access Point und Router mit embedded Board Banana Pi R1

Referent : Dr. Jiri Spale

Vorgelegt am : ??.??.2017

Vorgelegt von : Bastian
Elias
Jakob
Jonas
Tom

Abstract

The project „Access Point und Router mit embedded Board Banana Pi R1“ is a continuation of a previous project done at the university in Furtwangen of the same name from the winter semester 2016/2017. Its main goals were to find a suitable operating system for the BananaPi and recording the network traffic. There were future objectives listed at the end of the documentation which have been worked upon in this semester. The main goals for this project are to further analyse and test applicable operating systems as well as implementing different functions such as Radius, Samba, Mailserver, display status readout and a backup solution.

Bei dem Projekt „Access Point und Router mit embedded Board Banana Pi R1“, handelt es sich um die Weiterführung des bereits im Wintersemester 2016/2017 an der Hochschule Furtwangen abgehaltenen Projekts mit demselben Name. Das damalige Vorhaben bestand daraus, ein passendes Betriebssystem für den BananaPi zu ermitteln und den Netzwerkverkehr aufzuzeichnen. Am Ende wurden zukünftige Ziele aufgelistet welche in diesem Semester erarbeiten wurden. Projektziele sind, das Analysieren und Testen passender Betriebssysteme, sowie die Implementierung verschiedener Funktionen wie Radius, Samba, einem Mailserver, einer Displaystatusanzeige und einer Backup Lösung.

Inhaltsverzeichnis

Abstract	i
Inhaltsverzeichnis	v
Abbildungsverzeichnis	vii
Abkürzungsverzeichnis	ix
1 Projektübersicht	1
1.1 Kontext	1
1.2 Ziel	1
2 Betriebssystemeübersicht	3
2.1 OpenWRT	3
2.2 IPFire	3
2.3 Bananian	4
2.4 Armbian	4
3 Implementierung	5
3.1 Backup und Wiederherstellung des Betriebssystems	5
3.1.1 dd	5
3.1.2 rsync	7
3.1.3 tar	7
3.1.4 Automatisierung mittels Bash-Skript	8
3.2 Update des Banana Pi	9
3.3 Implementierung einer Displaystatusanzeige	10

3.4	Mail-Server	11
3.5	Samba	13
3.6	DDNS	15
3.6.1	Einrichtung No-IP	15
3.6.2	Einrichtung Custom-Domain	18
3.7	Radius	20
3.7.1	Allgemein	20
3.7.2	Funktionsweise	20
3.7.3	FreeRADIUS	21
3.8	Inbetriebnahme von FreeRADIUS2 auf OpenWRT	23
3.8.1	Instalation	23
3.9	Captive Portal	27
3.9.1	Algemeines	27
3.9.2	Funktionsweise	27
3.9.3	CoovaChilli	28
4	Benutzeranleitung	31
4.1	Pi Hoch- und Herunterfahren	31
4.2	Verbindung über SSH	31
4.3	Lan, Wlan und Vlans	33
4.4	Samba	34
4.5	Mailserver	34
4.6	Domainverwaltung	35
4.7	Backup und Restore	37
4.8	Statusanzeige	37
5	Projektbewertung	39
6	Teilnehmer und Rollenverteilung	41

7	Zukünftige Ziele	43
	Literaturverzeichnis	45
	Eidesstattliche Erklärung	47

Abbildungsverzeichnis

Abbildung 1: IPFire Logo	3
Abbildung 2: Bananian Logo	4
Abbildung 3: Armbian Logo	4
Abbildung 4: Vorhandene Dateisysteme	5
Abbildung 5: Laufender Backupprozess mit Statusanzeige	6
Abbildung 6: Laufender Wiederherstellungsprozess mit Statusanzeige	6
Abbildung 7: Ausführung wöchentliches Backup	8
Abbildung 8: Verwendung von Tmux mit 'htop' und 'iftop'	10
Abbildung 9: noip Domain	15
Abbildung 10: noip Konfiguration	16
Abbildung 11: freenom Domain	18
Abbildung 12: freenom Nameserver	18
Abbildung 13: Cloudflare DNS	19
Abbildung 14: Radius Funktionsweise	20
Abbildung 15: Radius Test	22
Abbildung 16: CoovaChilli Funktionsweise	28
Abbildung 17: Eingabe des Hostnamens	32
Abbildung 18: Die SSID des AP	33
Abbildung 19: No-IP Startseite	35
Abbildung 20: Freenom Domainverwaltung	36
Abbildung 21: Cloudflare DNS	36

Abkürzungsverzeichnis

1 Projektübersicht

1.1 Kontext

Das Projekt mit dem Titel Router mit embedded Board Banana Pi R1 wird als Semesterprojekt im Sommersemester 2017 an der Hochschule Furtwangen durchgeführt. Das Projekt wurde von Dr. Jiri Spale ins Leben gerufen und wird intern, ohne die Kooperation mit einem Unternehmen durchgeführt. Die Anzahl der studentischen Projektteilnehmer beträgt fünf.

1.2 Ziel

Das primäre Ziel des Projektes ist es, das bestehende Router-Projekt weiterzuentwickeln. Folgende Funktionalitäten sollen implementiert werden:

- Implementierung eines RADIUS-Servers zur zentralen Authentifizierung von Anwendern
- Implementierung eines Voucher-Systems
- Installation des Betriebssystems IP Fire
- Mehrere WLAN Access Points auf einem WLAN-Chip
- Mail-Server zum Statusbericht der Backups
- SAMBA/NFS Server
- Automatische Aktualisierung von Programmpaketen des Systems
- Implementierung einer Display-Statusanzeige

Die Funktionen sollen auf dem Betriebssystem "Bananian", einem auf das Banana Pi zugeschnittenen Debian, implementiert werden.

2 Betriebssystemeübersicht

2.1 OpenWRT

2.2 IPFire

Das Betriebssystem IPFire ist darauf ausgelegt, als Router genutzt zu werden, wobei das Augenmerk auf der Implementierung einer Firewall oder eines VPN Gateways liegt. Das vorgehende Projekt hatte dieses Betriebssystem als Alternative vorgeschlagen, weshalb wir einen Blick darauf geworfen haben. Zwei Abbilder für den Betrieb auf ARM Prozessoren stehen auf der Webseite des Betriebssystems bereit. [2]

Bei beiden Abbildern stellte sich allerdings heraus, dass die HDMI Schnittstelle nicht unterstützt wird. Auch eine Verbindung SSH nicht möglich ist, da IPFire auf eine Konfiguration über das Webinterface ausgelegt ist. Eines der beiden Abbilder ist auf die Konfiguration über eine serielle Schnittstelle ausgelegt. Da uns aber für diese Schnittstelle keine Werkzeuge und Adapter bereitstehen, haben wir auch von dieser Option abgesehen.



Abbildung 1: IPFire Logo

Quelle: [1]

2.3 Bananian

Der Banana Pi R1 wird von dem Betriebssystem Bananian vollständig unterstützt. Dieses Betriebssystem basiert auf Debian 8 und nutzt das Debian Jessie armhf repository. [4]

Neben OpenWRT wurde das Projekt mit Bananian gestartet. Da die Weiterentwicklung von Bananian am 2.4.2017 eingestellt wurde haben wir uns entschieden auf das Betriebssystem Armbian zu wechseln. [5]



Abbildung 2: Bananian Logo
Quelle: [3]

2.4 Armbian

Wie auch Bananian basiert Armbian auf Debian Jessie, das speziell für ARM Prozessoren entwickelt wurde. Dieses Betriebssystem haben wir gewählt, da es Bananian stark ähnelt, im Gegensatz zu Bananian aber weiterentwickelt wird und damit auch in gewissem Maße für die Zukunft gewappnet ist. Außerdem unterstützt es alle Funktionen die für das Projekt benötigt werden. [7]

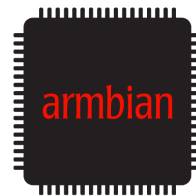


Abbildung 3: Armbian Logo
Quelle: [6]

3 Implementierung

3.1 Backup und Wiederherstellung des Betriebssystems

Um die Implementationen und Konfigurationen des Pi's abzusichern, wird eine Backup Lösung eingesetzt. Die Sicherung relevanter Daten soll hierbei einem möglichen Defekt oder Inkompatibilität durch Updates o.ä. entgegenwirken.

'FauBackup' und 'gitbac' bieten hierbei eine Lösung mittels externer Programme an. Debian bietet jedoch bereits standardmäßig einige Aufrufe welche zur Anlegung von Backups verwendet werden können. Diese wurden im Folgenden in Ubuntu anhand eines Bash Skripts zur Automatisierung getestet und implementiert.

3.1.1 dd

Als erste Ausführung wurde der Aufruf 'dd' verwendet. Hierbei wird der Inhalt der gesamten SD Karte als Image Datei abgespeichert.

Schritt 1:

Übersicht der vorhandenen Dateisysteme mittels des Terminalaufrufs 'df'

```
bastian@bastian-VirtualBox:~$ df
Dateisystem    1K-Blöcke  Benutzt Verfügbar Verw% Eingehängt auf
udev           497668      4    497664   1% /dev
tmpfs          101768     900    100868   1% /run
/dev/sda1       9156984  5217368  3451424  61% /
none            4          0         4    0% /sys/fs/cgroup
none           5120       0        5120   0% /run/lock
none          508832     76    508756   1% /run/shm
none          102400     56    102344   1% /run/user
none          455712764 373044796 82667968  82% /media/sf_OrdnerWindows
bastian@bastian-VirtualBox:~$
```

Abbildung 4: Vorhandene Dateisysteme

Schritt 2:

Terminalaufruf für den Backupprozess

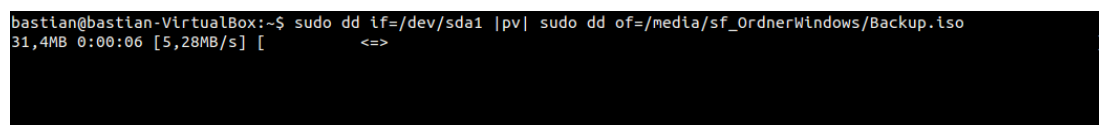
```
sudo dd if=INPUTPARTITION of=OUTPUTFILE
```

sudo -> Backupprozess benötigt root-Rechte
 dd -> bit-genaues Kopieren der Dateien
 if=FILE -> Die Datei oder Partition welche integriert wird
 of=FILE -> Die Output Datei welche angelegt wird

Schritt 3:

Optionale Nutzung von Terminalaufruf 'pv' um den Fortschritt des Backup Prozesses zu sehen. Die mögliche Restzeit lässt sich nur durch das Hinterlegen der Größe der Partition anzeigen.

```
sudo dd if=INPUTPARTITION |pv| sudo dd of=OUTPUTFILE
```



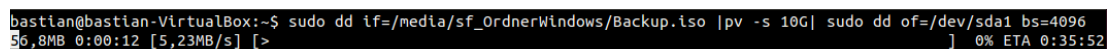
```
bastian@bastian-VirtualBox:~$ sudo dd if=/dev/sda1 |pv| sudo dd of=/media/sf_OrdnerWindows/Backup.iso
31,4MB 0:00:06 [5,28MB/s] [ <=> ]
```

Abbildung 5: Laufender Backupprozess mit Statusanzeige

Schritt 4:

Wiederherstellen eines hinterlegten Backups läuft ähnlich wie der ursprüngliche Prozess ab.

```
sudo dd if=OUTPUTFILE |pv| of=INPUTPARTITION
```



```
bastian@bastian-VirtualBox:~$ sudo dd if=/media/sf_OrdnerWindows/Backup.iso |pv -s 10G| sudo dd of=/dev/sda1 bs=4096
56,8MB 0:00:12 [5,23MB/s] [ > ] 0% ETA 0:35:52
```

Abbildung 6: Laufender Wiederherstellungsprozess mit Statusanzeige

Quelle: [8]

3.1.2 rsync

Da der Vorgang mittels 'dd' als suboptimal angesehen wird, wurde alternativ der Aufruf 'rsync' verwendet. Dieser bietet die Möglichkeit eines inkrementellen Backups wodurch die Dauer des Prozesses erheblich reduziert werden kann. Hierbei werden die Größe und die Änderungszeit der Dateien in Quelle und Ziel miteinander verglichen. Eine Aktualisierung findet demnach nur statt, wenn Unterschiede vorzufinden sind.

```
rsync -aAXv --delete --exclude={"/dev/*","/proc/*","/sys*  
/*","/tmp/*","/run/*","/mnt/*","/media/*","/lost+found*  
"} / /path/to/backup/folder
```

- rsync -> Kopieren der Dateien
- aAX -> Übertragung im Archiv Modus wodurch alle symbolischen Verweise beibehalten werden
- delete -> Dateien die im Ursprungsverzeichnis nicht mehr existieren werden im Zielverzeichnis ebenfalls gelöscht
- exclude -> Dateien werden ausgelassen

Wiederherstellen des Rsync Backups durch folgenden Befehl:

```
rsync -aAXv /path/to/backup/location/* /mount/point/of/ ↵  
new/install/ --exclude={"/dev/*,/proc/*,/sys/*,/tmp/*,/ ↵  
run/*,/mnt/*,/media/*,/lost+found ,/home/*}
```

Quelle: [9]

3.1.3 tar

Eine weitere Anwendungsmöglichkeit bietet die 'tar' Archivierung. Vorteil dieses Aufrufs ist, dass durch Angabe von Parametern die Berechtigungen aller zu sichernden Daten ebenfalls beibehalten werden und die Archivierung Speicherplatz spart.

Wechsel in das Backupverzeichnis dann:

```
tar -cpzf Backup.tar ORDNER
```

- tar -> Archivieren von Daten
- c -> Archiv wird erzeugt (create)
- p -> Berechtigungen beibehalten (privilege)
- z -> Zusätzliche Komprimierung mit gzip
- f -> Archiv in Datei schreiben (finish)

Quelle: [10]

3.1.4 Automatisierung mittels Bash-Skript

Damit der Nutzer die Aufrufe nicht händisch zu bestimmten Zeiten ausführen muss, wurden zwei Bash-Skripte zur Automatisierung geschrieben. Es gibt ein monatliches Backup mittels (tar) und wöchentliche inkrementelle Backups (rsync) auf die zurückgegangen werden kann.

Um das Zeitintervall der Backups einzustellen wird der 'Cron' Dienst verwendet. Hiermit können Skripte und Programme zu festgelegten Zeiten gestartet werden. Wenn ein hinterlegter Job täglich zu einer bestimmten Uhrzeit ausgeführt wird muss allerdings auch der Rechner zu dem Zeitpunkt aktiv sein. Ist dies nicht der Fall, startet der Prozess nicht. Um dies zu umgehen wird 'Anacron' verwendet. Durch ablegen des Skripts in eines der entsprechenden Verzeichnisse wird der Prozess entsprechend ausgeführt. [11]

- /etc/cron.hourly/ - Stündlich ausführen
- /etc/cron.daily/ - Täglich ausführen
- /etc/cron.weekly/ - Wöchentlich ausführen
- /etc/cron.monthly/ - Monatlich ausführen

```
Woechentliches Backup wird ausgefuehrt. Diesen Vorgang bitte nicht abbrechen!  
sending incremental file list  
deleting security.update.log  
rsync: symlink "/media/sf_OrdnerWindows/Backup/initrd.ing" -> "boot/initrd.ing-3.13.0-119-generic" failed: Read-only file system (30)  
rsync: symlink "/media/sf_OrdnerWindows/Backup/initrd.ing.old" -> "boot/initrd.ing-3.13.0-117-generic" failed: Read-only file system (30)  
rsync: symlink "/media/sf_OrdnerWindows/Backup/vmlinuz" -> "boot/vmlinuz-3.13.0-119-generic" failed: Read-only file system (30)  
rsync: symlink "/media/sf_OrdnerWindows/Backup/vmlinuz.old" -> "boot/vmlinuz-3.13.0-117-generic" failed: Read-only file system (30)  
./
```

Abbildung 7: Ausführung wöchentliches Backup

3.2 Update des Banana Pi

Ziel war es das Betriebssystem und alle Programme immer auf dem aktuellsten Stand zu halten. Hierbei kann es jedoch zu Inkompatibilität bestimmter Funktionen oder Konfigurationen kommen. Daher wurde die Umsetzung auf die relevantesten Updates (Sicherheitsupdates) reduziert. Im Normalfall können mittels des Konsolenaufrufs 'apt-get update' die Updateliste und mit 'apt-get upgrade' die Programm Pakete selbst aktualisiert werden. Durch Nutzung des Aufrufs 'unattended-upgrade' wird auf Sicherheitsupdates des Systems überprüft und diese anschließend installiert.

Durch 'unattended-upgrade --dry-run -d' wird auf Verfügbarkeit von Updates geprüft ohne anschließende Installation. Nach jedem Durchgang wird eine Logdatei in /var/log/unattended-upgrades/ angelegt welche genauere Informationen zu den aktualisierten Dateien liefert. [12]

3.3 Implementierung einer Displaystatusanzeige

Zur besseren Übersicht des Netzwerktraffics als auch der Ressourcen des Banana Pi sollte eine Displaystatusanzeige implementiert werden. Der Aufruf 'htop' bietet eine Übersicht aller laufenden Prozesse und deren Ressourcennutzung. 'iftop' zeigt die Netzwerkinterfaces und die eingehende und ausgehende Kommunikationen. Da das Terminal jedoch nur einen der Befehle zu einem Zeitpunkt ausüben kann wird ein Terminal-Multiplexer verwendet. Zur Auswahl stehen hierbei 'Terminator', 'screen', und 'Tmux'. Aufgrund der geringen Einarbeitungszeit und einfachen Anwendbarkeit wurde letzteres zur Implementation ausgewählt. Alle Multiplexer bieten die Möglichkeit Sitzungen zu erstellen. Leider kann dies nicht zur Implementierung der Displaystatusanzeige verwendet werden, da die Sitzung beim Herunterfahren des Betriebssystems gelöscht wird. Daher wird zum Systemstart ein Bash-Skript eingesetzt, welches automatisch die benötigten Fenster zur Überwachung anlegt. [13]

```

bastian@bastian-VirtualBox:~/Dokumente/Studium_HFU/ProjektSS2017/BackupRestoreUpdate/Fertige_Skripte/Ubuntu$

```

```

TX:          cum:    0B    pearates:    0b    0b    0b
RX:          0B
TOTAL:       0B

```

```

1  [ ] 3.3% Tasks: 134, 194 thr; 1 running
2  [ ] 6.7% Load average: 0.26 0.13 0.10
Mem [ ] 563/993MB Uptime: 01:02:01
Swp [ ] 2/1021MB

```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Comm
2176	bastian	20	0	1389M	63908	15240	S	4.7	6.3	0:30.17	comp
1057	root	20	0	455M	90016	8088	S	3.3	8.8	0:28.12	/usr
3574	root	20	0	31440	1984	1400	R	2.7	0.2	0:00.43	htop
2492	bastian	20	0	638M	16612	5504	S	0.7	1.6	0:07.57	gnom
1096	mysql	20	0	609M	50116	900	S	0.7	4.9	0:05.85	/usr
1320	root	20	0	235M	1072	664	S	0.7	0.1	0:02.25	/usr
2230	bastian	20	0	857M	26360	13044	S	0.0	2.6	0:04.08	naut

```

F1Help F2Setup F3Search F4Filter F5Free F6SortBy F7nice F8nice F9

```

Abbildung 8: Verwendung von Tmux mit 'htop' und 'iftop'

3.4 Mail-Server

Um Statusbenachrichtigungen zu erhalten, wurde ein Mail-Server auf dem Pi eingerichtet, welcher als Relay über Google Mail fungiert. Dazu wurde ein Google Mail Konto eingerichtet, auf welches Post Fix zugreift und Mails verschickt.

Alle System Mails werden über das Google-Konto weitergeleitet, dazu gehören auch Statusmeldungen des Backup-Skripts.

Dieser Weg wurde wegen des geringen Aufwands gewählt. Ohne Relay würde man eine eigene Domain und sehr viel mehr Konfiguration benötigen. Da ein Google Mail Konto nie verfällt, war dies die beste und pflegeleichteste Möglichkeit.

Weiterhin können die Mails auch an jede beliebige Adresse verschickt werden, dies ist im Mail-Server frei konfigurierbar. Die Mails gehen momentan an die Google Mail Adresse.

Einrichtung:

Voraussetzungen:

- Internetzugriff
- Google Mail-Konto
- Texteditor (Nano)
- SSH/Physikalischen Zugriff

1. Post Fix installieren:

```
apt-get update  
apt-get install postfix libsasl2-modules bsd-mailx
```

2. Das Konfigurationsfenster öffnet sich.

3. TLS/SSL aktivieren:

```
nano /etc/postfix/main.cf
```

3.1 Folgendes Einfügen:

```
mtp_sasl_auth_enable = yes  
smtp_sasl_security_options = noanonymous  
smtp_sasl_password_maps = hash:/etc/postfix/sasl_password  
# verschluesselung einschalten  
smtp_tls_security_level = may
```

4. Nutzerdaten des Google Mail-Kontos hinterlegen:

```
nano /etc/postfix/sasl_password  
smtp.gmail.com Bananapihfu:<Password>
```

5. Datei nur für root lesbar machen, da Klartext:

```
chmod 600 /etc/postfix/sasl_password
```

6. Postfix lookup Tabelle erstellen:

```
postmap hash:/etc/postfix/sasl_password
```

7. Postfix neustarten:

```
/etc/init.d/postfix restart
```

8. Für Weiterleitung der Systemnachrichten aliases bearbeiten:

```
nano /etc/aliases  
root: bananapihfu@gmail.com
```

Oder Wunschemail an welche Systemnachrichten gesendet werden

9. Änderungen an Aliases wirksam machen:

```
Newaliases
```

Quelle: [14]

3.5 Samba

Um den Dateizugriff zu erleichtern, wurde ein Samba-Server auf dem Pi implementiert.

Samba ermöglicht es von nahezu jedem Gerät auf ein Freigegebenes Verzeichnis auf dem Server zuzugreifen. Voraussetzung ist, dass das Client Betriebssystem das SMB-Protokoll unterstützt.

Die meisten modernen Betriebssysteme, wie Windows, MacOS und andere Unixoiden besitzen Samba Funktionalität.

Einrichtung

Voraussetzungen:

- Internetzugriff
- Texteditor (Nano)
- SSH/Physikalischen Zugriff

1. Samba installieren:

```
sudo apt-get update  
sudo apt-get install samba
```

2. Benutzer für Samba erstellen (Hat keinen Shell-Zugriff):

```
useradd sambaur --shell /bin/false
```

3. Passwort für den Benutzer in Samba setzen:

```
smbpasswd -a <user_name>
```

4. Verzeichnis im Home erstellen:

```
mkdir /home/sambaur  
mkdir /home/sambaur/samba
```

5. Berechtigungen setzen:

```
chown sambaur:sambaur /home/sambaur/  
chown sambaur:sambaur /home/sambaur/samba/
```

6. Backup der Samba Konfiguration im Homeverzeichnis machen:

```
cp /etc/samba/smb.conf ~
```

7. Config bearbeiten:

```
nano /etc/samba/smb.conf
```

7.1 folgendes am Ende der Konfiguration Einfügen:

```
[samba]  
path = /home/sambausr/samba  
valid users = sambausr  
read only = no
```

8. Service neustarten:

```
service smbd restart
```

9. Config testen:

```
testparm
```

Quelle: [15]

3.6 DDNS

Um immer die aktuelle IP-Adresse des Pi zur Hand zu haben, wurde ein DynDNS-Client von no-ip implementiert, bei jedem Systemstart wird die bei der DNS hinterlegte IP-Adresse aktualisiert.

Nachteil hierbei ist jedoch, dass man alle 30 Tage diese Domain aktivieren muss, da diese sonst verfällt.

Um dies zu umgehen wurde die Kostenfreie Domain bananapihf.tk gebucht. Von Vorteil ist hier, dass diese Domain eine Laufzeit von einem Jahr hat und nach Ablauf auch ohne Mehrkosten verlängert werden kann.

Diese Domain wird bei Cloudflare verwaltet, da hier auch eine API angeboten wird, mit welcher man Theoretisch komplett auf eine Dynamische DNS bei No-IP verzichten kann.

3.6.1 Einrichtung No-IP

Voraussetzungen:

- Internetzugriff
- SSH/Physikalischen Zugriff
- NoIP.com Account

1. NoIP Hostnamen anlegen:

Man muss in No-IP den gewünschten Hostnamen einrichten, welcher später auf die aktuelle IP-Adresse verweist.

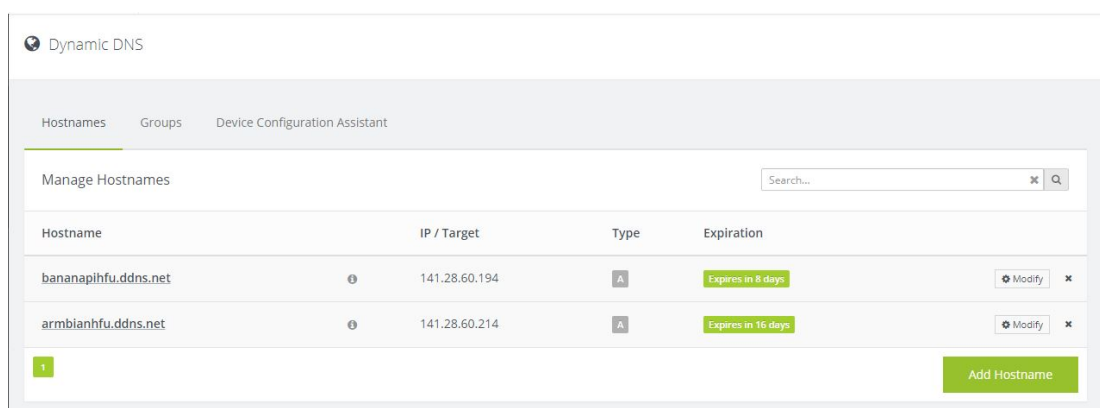


Abbildung 9: noip Domain

2. Verzeichnis für DDNS im Homeverzeichnis erstellen und wechseln:

```
mkdir DDNS
cd DDNS
```

3. NoIP-Client von NoIP.com beziehen:

```
wget https://www.noip.com/client/linux/noip-duc-linux.tar.gz
```

4. Das Archiv entpacken:

```
tar xvf noip-duc-linux.tar.gz
```

5. In den Entpackten Ordner wechseln:

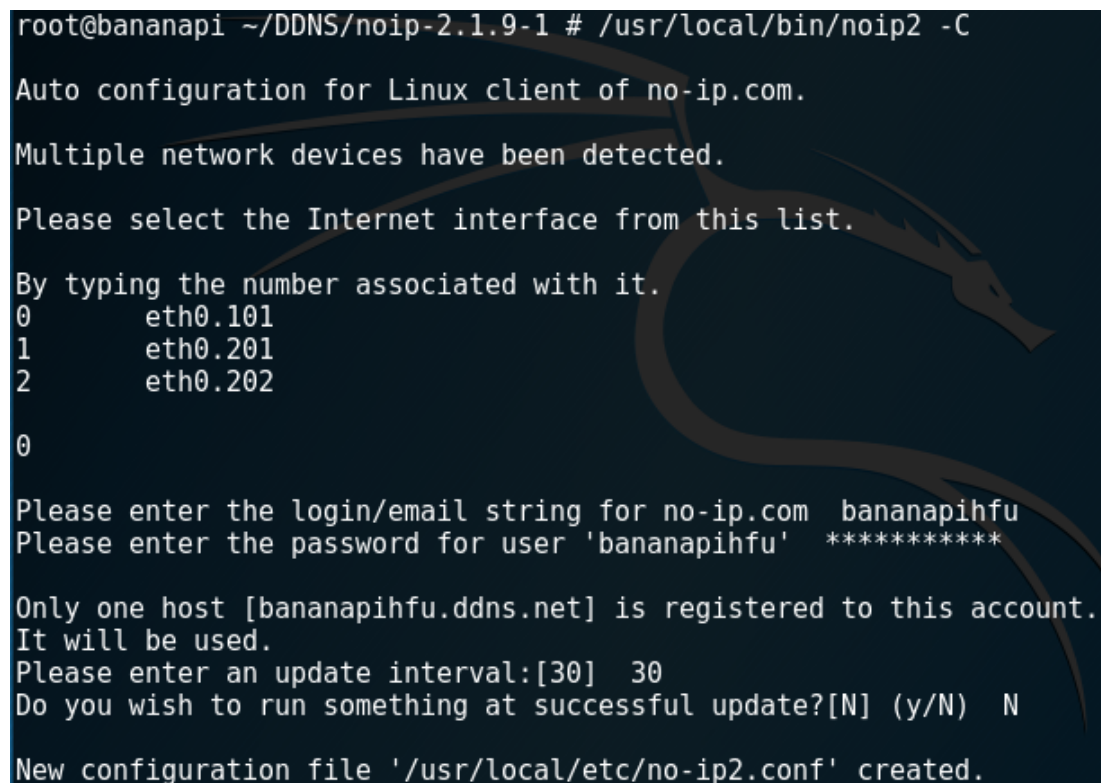
cd noip eingeben und mit TAB vervollständigen

```
cd noip-x.x.x-x
```

6. Client bauen und installieren:

```
make && make install
```

7. Konfiguration wird gestartet:



```
root@bananapi ~/DDNS/noip-2.1.9-1 # /usr/local/bin/noip2 -C
Auto configuration for Linux client of no-ip.com.
Multiple network devices have been detected.
Please select the Internet interface from this list.
By typing the number associated with it.
0      eth0.101
1      eth0.201
2      eth0.202
0

Please enter the login/email string for no-ip.com  bananapihf
Please enter the password for user 'bananapihf'  *****

Only one host [bananapihf.ddns.net] is registered to this account.
It will be used.
Please enter an update interval:[30]  30
Do you wish to run something at successful update?[N] (y/N)  N

New configuration file '/usr/local/etc/no-ip2.conf' created.
```

Abbildung 10: noip Konfiguration

Um den Daemon automatisch bei Systemstart zu starten muss noch folgende Konfiguration vorgenommen werden:

1. Startscript unter `/etc/init.d/noip2` ablegen:

```
vim /etc/init.d/noip2
```

2. Script "noip2" kopieren und einfügen:

Verweis auf Anhang

3. Script ausführbar machen:

```
chmod a+rx /etc/init.d/noip2
```

Quelle: [16]

3.6.2 Einrichtung Custom-Domain

Voraussetzungen:

- Account bei Cloudflare

1. Domain bei Freenom.com aussuchen:

Hier gibt es jede Menge kostenfreie Domains

2. Wunsch-Domain registrieren

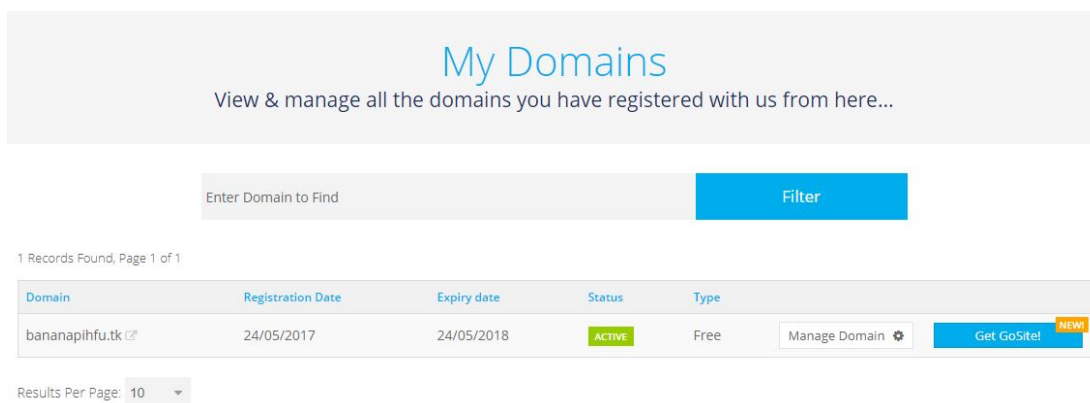


Abbildung 11: freenom Domain

3. Nameserver bei Freenom ändern:

Um die Domain bei Cloudflare zu verwalten, müssen die Nameserver angepasst werden. Diese erhält man, wenn man sich bei Cloudflare anmeldet.

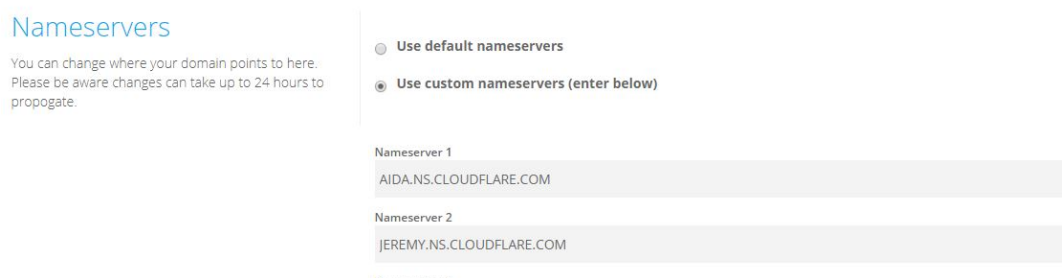


Abbildung 12: freenom Nameserver

3. CNAME Weiterleitung auf No-Ip einrichten

DNS

Manage your Domain Name System (DNS) settings.



DNS Records

A, AAAA, and CNAME records can have their traffic routed through the Cloudflare system. Add more records using this form, and click the cloud next to each record to toggle Cloudflare on or off.

- ⚠ An A, AAAA or CNAME record was not found for the **www** subdomain. The **www.bananapihf.tk** subdomain will not resolve.
- ⚠ An A, AAAA or CNAME record was not found pointing to the root domain. The **bananapihf.tk** domain will not resolve.
- ⚠ An MX record was not found for your root domain. An MX record is required for mail to reach **@bananapihf.tk** addresses.

Q Search DNS records

A Automatic TTL

Type	Name	Value	TTL	Status
CNAME	armbian	is an alias of armbianhf.ddns.net	Automatic	 <input type="button" value="X"/>
CNAME	bananian	is an alias of bananapihf.ddns.net	Automatic	 <input type="button" value="X"/>

[Advanced](#) [API](#) [Help](#)

Abbildung 13: Cloudflare DNS

3.7 Radius

3.7.1 Allgemein

Der Remote Authentication Dial-In User Service (RADIUS, deutsch Authentifizierungsdienst für sich einwählende Benutzer) ist ein Protokoll zwischen Benutzer und Server, das für die 3 A's (dem sogenannten Tripple-A-System), also der Authentifizierung, Autorisierung und das Accounting zuständig ist. Laut Internetquellen ist es der „De-facto Standard bei der zentralen Authentifizierung von Einwahlverbindungen über Modem, ISDN, VPN, WLAN (IEEE 802.1X) und DSL“. [17]

3.7.2 Funktionsweise

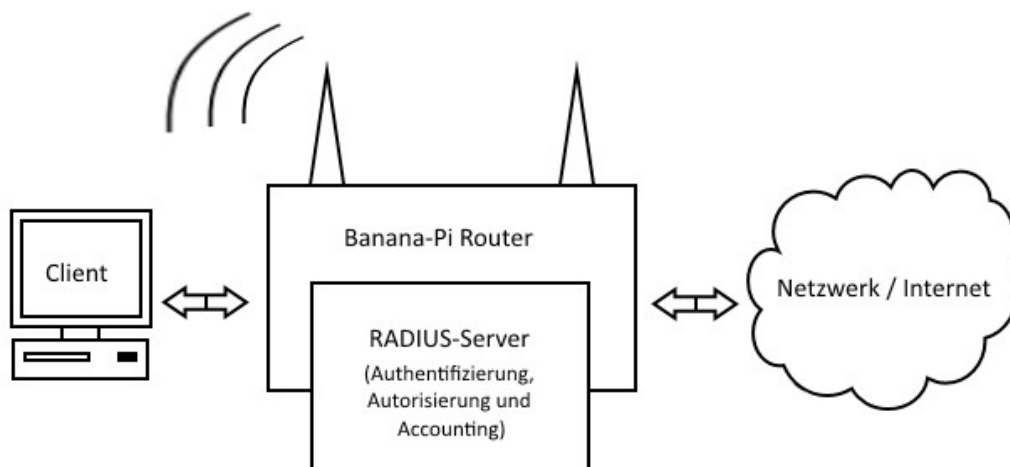


Abbildung 14: Radius Funktionsweise

Anhand der obenstehenden Abbildung lässt sich gut erkennen wie der Aufbau ist. Der Client (Smartphone, PC, etc.) kann sich wahlweise per WLAN oder LAN mit dem Banana-Pi Router verbinden, dabei sendet er eine Authentifizierungsanfrage an diesen, welche der Router an den Radius-Server, welcher auf dem Banana-Pi läuft, weiterleitet. Dieser verarbeitet nun die Anfrage indem er, je nach Konfiguration, in unserem Fall über eine SQL-Datenbank überprüft ob der Client berechtigt ist dem Netzwerk beizutreten. Zusätzlich kann diese Verbindung dann auch limitiert werden (Volumen-Limit, Bandbreiten-Drosselung, beschränkter Zugriff auf Subnetze/VLANs).

3.7.3 FreeRADIUS

FreeRADIUS ist wie der Name schon vermuten lässt eine freie und kostenlose Implementierung des RADIUS-Protokolls und unter der GNU General Public License, version 2 lizenziert. Es ist laut eigenen Angaben der weltweit am meisten eingesetzte RADIUS-Server und wird von den meisten Internetdiensteanbietern (Providern) sowie den 500 umsatzstärksten Unternehmen der Welt benutzt. [18]

Es findet außerdem auch in der Hochschule sowie im akademischen Forschungsnetzwerk Eduroam Einsatz. Es unterstützt den meistverbreiteten Authentifizierungsstandard EAP, auf deutsch etwa „Erweiterbares Authentifizierungsprotokoll“ [19], der ca. 40 verschiedene Verfahren anbietet, welche heutzutage unter anderem in den Sicherheitsimplementationen WPA und WPA2 Verwendung finden.

Installation:

FreeRADIUS ist auch in den offiziellen Paket-Quellen von Debian enthalten und kann somit ganz einfach über den Debian-Paketmanager apt-get installiert werden. Als erstes stellen wir sicher dass das System up-to-date ist und die aktuellen Paketquellen besitzt. [20]

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install freeradius freeradius-utils freeradius-mysql >
mysql-server
```

Bei der Installation des MySQL-Servers wird man gebeten das Passwort für den Root-User zu setzen und zu bestätigen (in unserem Fall ist dies bananapi). Im nächsten Schritt wird eine Datenbank für RADIUS angelegt.

```
ql -uroot -p
```

```
CREATE DATABASE radius;
GRANT ALL PRIVILEGES ON radius.* TO root@localhost >
IDENTIFIED BY "bananapi";
flush privileges;
exit
```

Danach noch die SQL Schemas in die Datenbank laden: [21]

```
mysql -uroot -p radius < /etc/freeradius/sql/main/mysql/ >
schema.sql
mysql -uroot -p radius < /etc/freeradius/sql/main/mysql/ >
setup.sql
```

Zukünftig können nun Benutzer mit folgendem MySQL-Befehl angelegt werden:

```
insert into radcheck (username,attribute,op,value) values
("USERNAME", "Cleartext-Password", ":", "PASSWORD");
```

Nun muss RADIUS dafür konfiguriert werden, das SQL-Modul zu benutzen, dafür editiert man die Datei `/etc/freeradius/sql.conf` und trägt im Bereich `#Connection` info die korrekten Login-Daten ein. Desweiteren muss man in der Datei `radius.conf` die Zeile `$INCLUDE sql.conf`, sowie in der Datei `/etc/freeradius/sites-available/default` zwei mal `sql` in den Sektionen `authorize` und `accounting` ein kommentieren.

Testen:

Um die Konfiguration zu überprüfen kann der RADIUS-Server im Debugging-Modus gestartet werden. Dazu sollte der RADIUS-Dienst erst beendet werden:

```
service freeradius stop
freeradius -X
```

Danach kann in einem zweiten Terminal (oder ggf. mit Terminal-Multiplexer) das Programm `radtest` verwendet werden um Authentifizierungsanfragen an den RADIUS-Server zu senden: [22]

```
radtest USERNAME PASSWORD 127.0.0.1 0 mysecret
```

Eine typische Ausgabe einer erfolgreichen Anfrage sollte wie folgt aussehen:

```
root@bananapi ~ # radtest mysqluser1 "testpass" localhost 1812 testing123
Sending Access-Request of id 140 to 127.0.0.1 port 1812
  User-Name = "mysqluser1"
  User-Password = "testpass"
  NAS-IP-Address = 141.28.60.191
  NAS-Port = 1812
  Message-Authenticator = 0x00000000000000000000000000000000
rad_recv: Access-Accept packet from host 127.0.0.1 port 1812, id=140, length=20
```

Abbildung 15: Radius Test

3.8 Inbetriebnahme von FreeRADIUS2 auf OpenWRT

3.8.1 Instalation

1. Installation des FreeRADIUS2 mit den üblichen Paketen die von verschiedenen anderen Paketen benutzt werden:

```
> opkg install freeradius2 freeradius2-common freeradius2-utils
Unknown package 'freeradius2'.
Unknown package 'freeradius2-common'.
Unknown package 'freeradius2-utils'.
Collected errors:
* opkg_install_cmd: Cannot install package freeradius2.
* opkg_install_cmd: Cannot install package freeradius2-common.
* opkg_install_cmd: Cannot install package freeradius2-utils.
```

-> Hat nicht funktioniert, auch nach halbstündiger Investigation konnte keine Lösung gefunden werden, auch mit

```
> opkg search freeradius2
> opkg info freeradius2
```

konnten keine Informationen über das Paket ermittelt werden. Schlussendlich konnten die Pakete über das Web-Interface installiert werden, dort klappte es auf Anhieb (mit den gleichen Bezeichnungen)

2. Installation der FreeRADIUS2 Pakete für die MYSQL Unterstützung (+ Packet für Logging):

```
> opkg install freeradius2-mod-sql freeradius2-mod-sql-log
mysql freeradius2-mod-sqllog
```

Nun sind die benötigten Pakete installiert.

Die Ausgabe nach der Installation sieht folgendermaßen aus:

```
Installing freeradius2 (2.2.8-2) to root...
Downloading file:///etc/packages/packages/freeradius2_2.2.8-2_sunxi.ipk.
Installing freeradius2-common (2.2.8-2) to root...
Downloading file:///etc/packages/packages/freeradius2-common_2.2.8-2_sunxi.ipk.
Configuring freeradius2-common.
Configuring freeradius2.
ifconfig: br-lan: error fetching interface information: Device not found
ifconfig: br-lan: error fetching interface information: Device not found
radiusd: Invalid IP Address or hostname "-p"
```

Mit br-lan ist hier das sogenannte „bridged lan virtual interface“ gemeint (liegt daran dass es standardmäßig so vorkonfiguriert ist).

Beim Versuch FreeRADIUS2 im Debugging-Modus zu starten kommt:

```
> radiusd -X
radiusd: FreeRADIUS Version 2.2.8, for host arm-openwrt-linux-gnu, built on Oct 3 2016 at 22:22:30
Copyright (C) 1999-2015 The FreeRADIUS server project and contributors.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
You may redistribute copies of FreeRADIUS under the terms of the GNU General Public License.
For more information about these matters, see the file named COPYRIGHT.
Starting - reading configuration files ...
including configuration file /etc/freeradius2/radiusd.conf
including configuration file /etc/freeradius2/clients.conf
including files in directory /etc/freeradius2/modules/
including configuration file /etc/freeradius2/eap.conf
Unable to open file "/etc/freeradius2/eap.conf": No such file or directory
Errors reading or parsing /etc/freeradius2/radiusd.conf
```

EAP ist das Extensible Authentication Protocol, ein allgemeines Authentifizierungsprotokoll. Also wurden die noch benötigten FreeRADIUS2 Abhängigkeiten installiert:

```
> opkg install freeradius2-mod-chap freeradius2-mod-
  detail freeradius2-mod-eap freeradius2-mod-eap-md5
  freeradius2-mod-eap-mschapv2 freeradius2-mod-eap-peap
  freeradius2-mod-eap-tls freeradius2-mod-eap-ttls
freeradius2-mod-exec freeradius2-mod-files freeradius2-
  mod-logintime
freeradius2-mod-mschap freeradius2-mod-pap freeradius2-
  mod-passwd
freeradius2-mod-preprocess freeradius2-mod-radutmp
```

3. Installation des MYSQL-Servers

```
> opkg install mysql-server
Installing mysql-server (5.1.73-1) to root...
Downloading file:///etc/packages/packages/mysql-server_5
  .1.73-1_sunxi.ipk.
Installing libmysqlclient (5.1.73-1) to root...
Downloading file:///etc/packages/packages/
  libmysqlclient_5.1.73-1_sunxi.ipk.
Configuring libmysqlclient.
Configuring mysql-server.
/etc/init.d/mysqld: Error: datadir '/mnt/data/mysql/' in
  /etc/my.cnf
doesn't exist
```

Nach einiger Zeit stieß ich im Internet auf vergleichbare Probleme und es gab einen Work-Around mit dem der MYSQL-Server sich schließlich doch installieren lies.

4. Einrichtung einer Zertifikatskette für das EAP

```
>opkg install openssl-util
```

->Erstellung der Zertifikatskette mit OpenSSL (...)

5. Bevor man allerdings mit der Konfiguration beginnt, sollte man jedoch den Radius-Daemon stoppen: Über das Web-Interface LuCi unter dem Menüpunkt „System“ -> „Startup“ wird der radiusd deaktiviert („disable“) und erstmal vom automatischen Starten beim Booten ausgenommen. Für Test- und Debuggingzwecke empfiehlt es sich außerdem, den Radius Server umzukonfigurieren, sodass dieser auch auf Localhost horcht und nicht nur auf die IP Adresse im Netzwerk. Dafür editiert man die Datei

```
> vim /etc/init.d/radiusd
```

und ersetzt die Zeile 17

```
radiusd -i $IPADDR -p 1812,1813 $OPTIONS
```

durch

```
radiusd $OPTIONS
```

3.9 Captive Portal

3.9.1 Allgemeines

Als Captive Portal (zu deutsch etwa „Umleitung auf ein Web-Portal“) wird eine Webseite bezeichnet, die dafür verwendet wird Authentifizierungen durchzuführen, meistens findet sie in WLANs Einsatz. In unserem Fall dient sie dem Login auf einer HTTP-Seite um sich beim RADIUS-Server einzuloggen. [23]

3.9.2 Funktionsweise

Prinzipiell ist die Funktionsweise relativ simpel. Man benötigt lediglich einen Webserver und eine Firewall, in unserem Fall Apache2 und iptables. Wenn nun ein Benutzer in ein WLAN-Netzwerk möchte, wird erst einmal der komplette Netzwerkverkehr ignoriert, bis er einen Browser öffnet. Dieser Zugriff wird immer auf das Captive Portal weitergeleitet, sodass der Benutzer immer zu der Login-Seite kommt. Je nach Anwendungsfall können dort dann auch Bezahlungssysteme implementiert werden oder Nutzungsbedingungen die akzeptiert werden müssen.

Es gibt verschiedene Möglichkeiten wie die Umleitung von Statten gehen kann, durchgesetzt hat sich aufgrund verschiedener sicherheitstechnischer Aspekte die Umleitung via HTTP (direkte IP- sowie DNS-Umleitungen sind aufgrund des dann nicht mehr gültigen DNSSEC-Zertifikats nicht möglich). [24]

3.9.3 CoovaChilli

Allgemeines:

CoovaChilli ist ein open-source Tool das unter der GNU General Public License 3 veröffentlich wurde und frei zugänglich ist. Es bietet eine Zugriffskontrolle als Captive Portal und liefert die dafür benötigten HTML/CSS/Javascript Dateien. Es basiert auf dem mittlerweile nicht mehr unterstützten Projekt CoovaSpot und funktioniert mit dem RADIUS-Protokoll.

Funktionsweise:

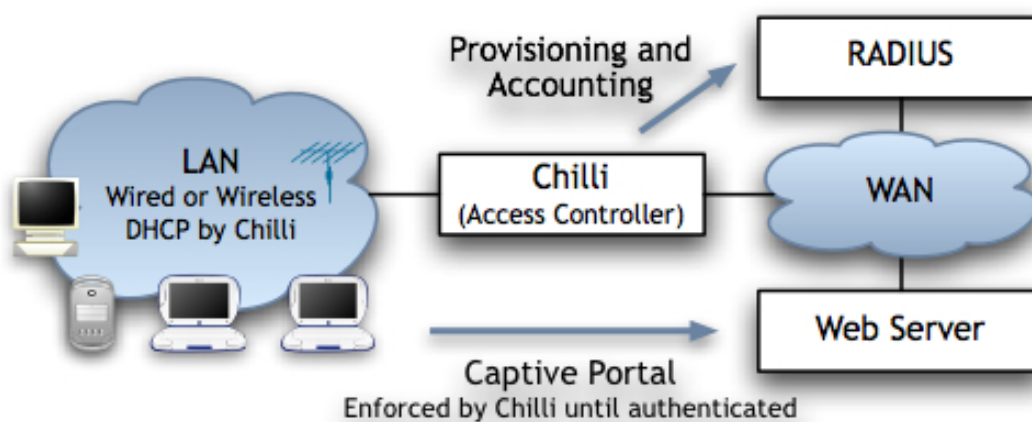


Abbildung 16: CoovaChilli Funktionsweise

Quelle: [25]

Wie in der Abbildung oberhalb zu erkennen bildet CoovaChilli die Instanz zwischen dem Access- Point des Netzwerks und der Netzwerkinfrastruktur dahinter (in unserem Fall laufen Netzwerk- Access-Server, also dem WLAN-AP, CoovaChilli und der RADIUS-Server auf der gleichen Maschine). Will ein Benutzer nun ins Internet verbindet er sich wahlweise per LAN oder WLAN mit dem AP, ruft den Browser auf und wird auf das Captive Portal auf dem Webserver weitergeleitet. Beim Login kommuniziert CoovaChilli mit dem RADIUS-Server, bei erfolgreichem Login wird der Internetzugang für den Benutzer freigegeben.

Installation:

4 Benutzeranleitung

4.1 Pi Hoch- und Herunterfahren

Um den Pi Hochzufahren, muss lediglich das Netzteil eingesteckt werden:

1. Netzteil in Steckdose einstecken
2. Micro-USB Port an den äußeren Port des Pis anschließen. Der andere Port ist nur für OTG!

Um den Pi Herunterzufahren muss benötigt man Zugriff auf die Kommandozeile

1. Pi an Tastatur und Bildschirm anschließen || SSH-Verbindung zum Pi aufbauen
2. Anmelden (User: root / Passwort: bananapi)
3. shutdown -h 0 eingeben und mit ENTER bestätigen

4.2 Verbindung über SSH

Um eine Verbindung über SSH aufzubauen wird folgendes benötigt:

- Putty (Windows)
- ssh-Packet (Linux)
- Netzwerkverbindung

Um eine Verbindung aufzubauen muss der Pi hochgefahren sein und über den Wan-Port an ein Netzwerk angeschlossen sein.

Windows:

1. Putty öffnen
2. Hostnamen armbian.bananapihf.tk angeben

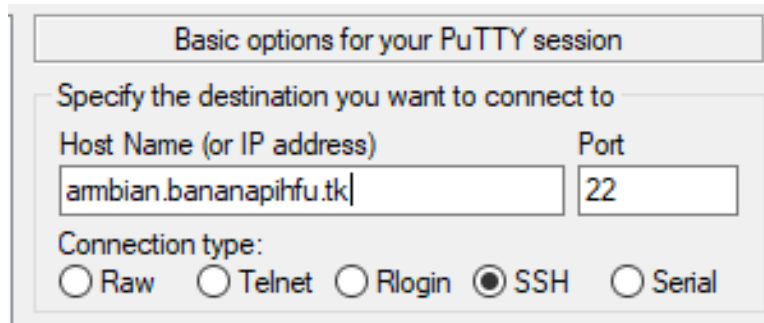


Abbildung 17: Eingabe des Hostnamens

3. Durch Klick auf „Open“ Verbindung aufbauen -> Terminal öffnet sich

```
login as: root
password: bananapi
```

4. Verbindung erfolgreich!

Linux:

1. Terminal öffnen
2. Folgenden Befehl eingeben:

```
ssh root@armbian.bananapihf.tk
```

3. Passwort eingeben:

```
password: bananapi
```

4. Verbindung erfolgreich!

4.3 Lan, Wlan und Vlans

Übersicht Vlans:

Der Pi besitzt 4 Lan Ports und einen Wlan Access-Point.

Vlan 1:

- Internetzugriff
- Lan 1 + Lan 2
- Gateway:192.168.1.1

Vlan 2:

- Kein Internetzugriff
- Lan 3 + Lan 4
- Gateway:192.168.2.1

Vlan 3:

- Internetzugriff
- WLAN AP
- Gateway:192.168.3.1

Wlan:

Der Pi besitzt einen Wlan-AP, über welchen eine Internetverbindung möglich ist. Lokal ist jedoch nur Zugriff auf Geräte im Vlan 3 möglich. Um eine Verbindung zum AP

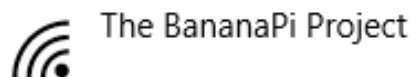


Abbildung 18: Die SSID des AP

herzustellen muss folgendermaßen Vorgegangen werden:

1. Wlan Übersicht am Client Gerät öffnen und nach APs suchen
 - a. The BananaPi Project auswählen
 - b. WPA/WPA2 PSK auswählen
 - c. Passwort: bananapi
2. Verbindung hergestellt

Lan:

Der Pi besitzt insgesamt 5 Lan-Ports. Einer davon ist der WAN-Port, welcher an ein externes Netzwerk angeschlossen wird. Die vier restlichen Ports werden über den Pi geroutet und sind in Vlans unterteilt (siehe Übersicht Vlans).

4.4 Samba

Verbindung aufbauen

Mit Samba können Dateien zwischen dem Pi und einem anderen Gerät ausgetauscht werden.

Um den Samba-Share im am eigenen PC einzubinden muss folgendes gemacht werden:

Windows:

1. Ausführen-Dialog aufrufen mit Win + R
2. \\armbian.bananapihf.tk\samba eingeben und mit ENTER bestätigen
3. Login Fenster öffnet sich
 - a. Login: sambausr
 - b. Passwort: bananapi

Linux:

1. smbclient installieren:
 - a. sudo apt-get update
 - b. sudo apt-get install smbclient
2. Verbindung aufbauen:
 - a. smbclient \\armbian.bananapihf.tk /samba -U sambausr
 - b. Passwort eingeben: bananapi

MacOSX:

1. Im Finder mit Server verbinden (Command + K)
2. Serveradresse angeben:
 - a. smb://armbian.bananapihf.tk/samba
3. Verbinden als Registrierter Nutzer:
 - a. Name: sambausr
 - b. Passwort: bananapi

4.5 Mailserver

Um auf die Mails zuzugreifen, muss man <https://www.google.com/gmail> besuchen und sich mit folgenden Benutzerdaten anmelden:

- Email: bananapihf
- Passwort: 63!gr&8FJZdd

Um die Emails über einen Mail-Client aufzurufen, kann die Anleitung von Google zur Hand genommen werden:

<https://support.google.com/mail/answer/7126229?hl=en>

4.6 Domainverwaltung

Ni-IP

Um die Dynamische DNS bei No-IP zu verwalten muss no-ip.com besucht werden: -

Login: bananaphfu

- Passwort: BananaPhone

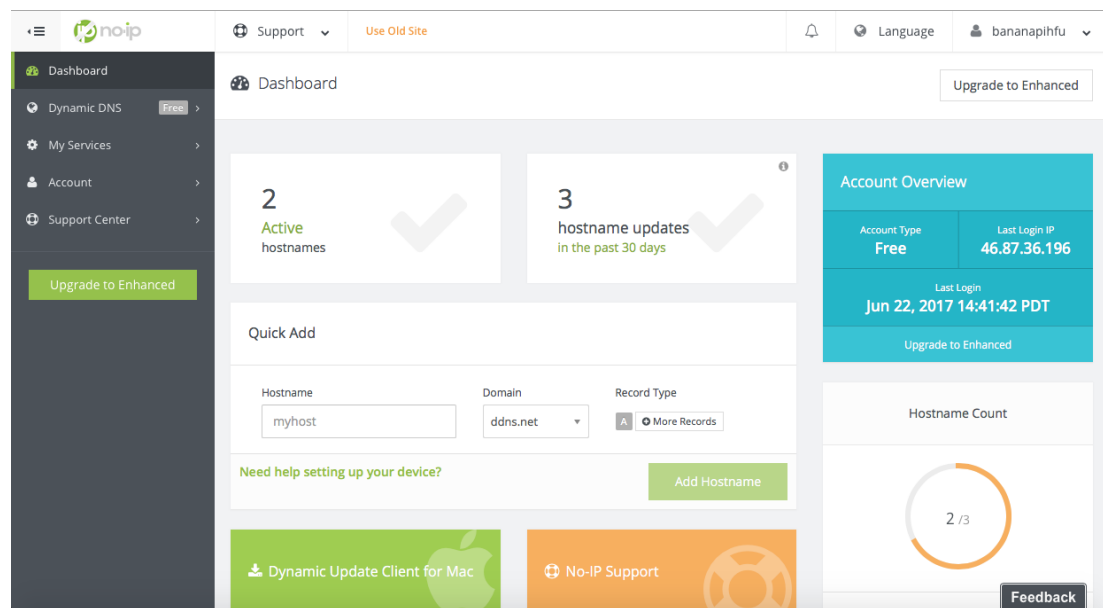


Abbildung 19: No-IP Startseite

Freenom.com

Über Freenom wurde die kostenfreie Domain bananapihf.tk gebucht.

Login: bananapihf@gmail.com

Passwort: bananapi

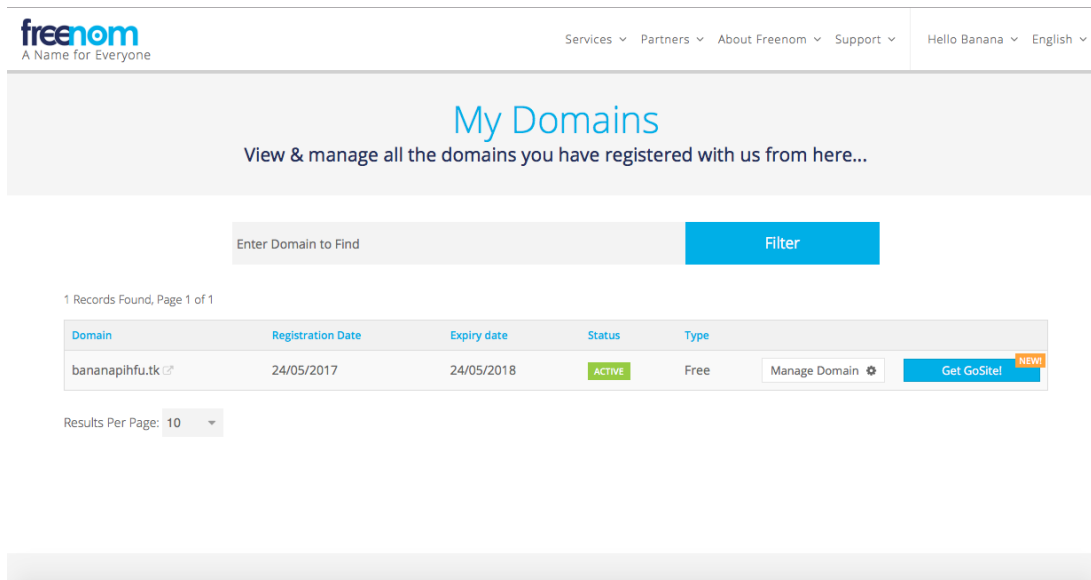


Abbildung 20: Freenom Domainverwaltung

CloudFlare

Über CloudFlare werden die DNS-Einträge der Domain verwaltet:

Login: bananapihf@gmail.com

Passwort: k%KKx!HkNW23

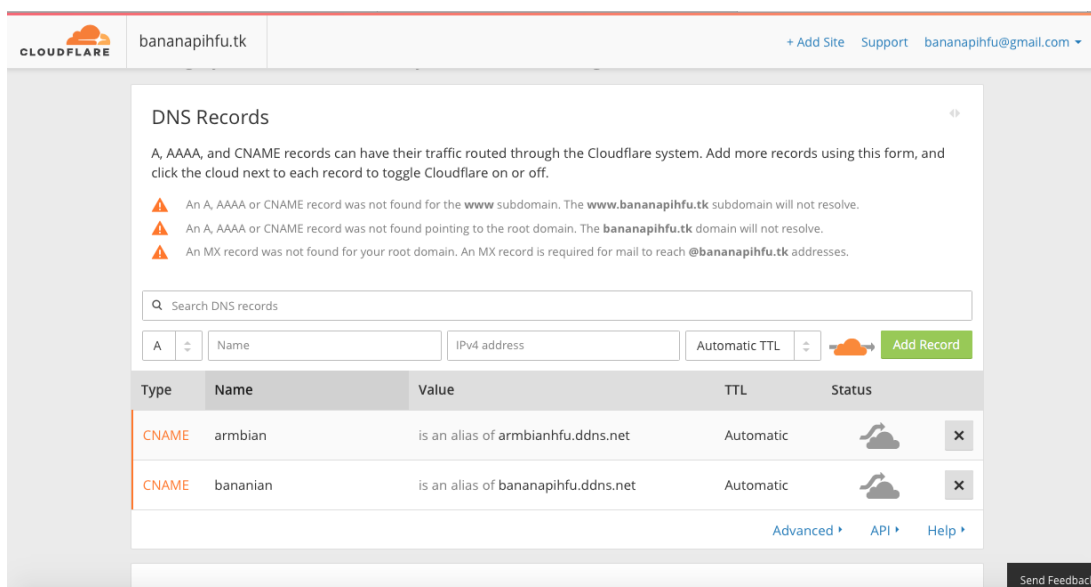


Abbildung 21: Cloudflare DNS

4.7 Backup und Restore

Das Backup wird über zwei Skripte gestartet, welche in den Verzeichnissen `/etc/-cron.weekly/` und `/etc/cron.monthly/` liegen. Das Backup wird somit regelmäßig ausgeführt.

Manuelles Backup

Sollte man ein aktuelles Backup benötigen, kann man das Backup-Script manuell ausführen:

```
bash /etc/cron.weekly/Backup_Weekly
```

Hierbei wird das Backup unter `/mnt/SSD/Backup_Weekly` gespeichert.

Zur Archivierung des Backups, kann auch das `Backup_Monthly` aufgerufen werden:

```
bash /etc/cron.monthly/Backup_Monthly
```

Dieses Skript speichert dann das Wöchentliche Backup unter `/mnt/SSD/Backup_Monthly` als tar-Archiv.

4.8 Statusanzeige

Die Statusanzeige wird bei Systemstart geöffnet, da das Skript für diese unter `/etc/init.d/` abgelegt ist. Das Skript baut eine `tmux`-Session auf, welche mit folgendem Befehl aufgerufen werden kann:

```
tmux attach
```

Beendet wird die Session mit folgender Tastenkombination:

```
CTRL + D
```


5 Projektbewertung

Die technischen Herausforderungen des Projekts waren für alle Teilnehmer eine spannende und lehrreiche Erfahrung. Durch die Umsetzung der verschiedenen Implementationsziele auf der gegebenen eingebetteten Plattform können alle Projektteilnehmer einen großen Zugewinn an fachspezifischen Wissen verzeichnen.

Mit den wöchentlichen Projekttreffen war es möglich, frühzeitig auf aktuelle Probleme zu reagieren und Lösungsansätze gemeinsam zu erörtern.

Aufgrund des Wechsels von Bananian auf Armbian wurde der Zeitplan des Projekts teils weit zurückgeworfen. Weniger vorweisbare Projekt-Ergebnisse und kleinere Entwicklungsschritte waren die Folge, da Kernfunktionen wie die Switcharchitektur des Banana Pi komplett neu entwickelt werden mussten. Zudem wirkten sich Updates des Betriebssystems teils deutlich auf die Verlässlichkeit und Bedienbarkeit des Geräts aus, was mutmaßlich dem noch unausgereiften Betriebssystem geschuldet ist, das sich zum Zeitpunkt des Projekts in einer sehr aktiven Entwicklungsphase befindet.

6 Teilnehmer und Rollenverteilung

Die Rollen der studentischen Projektteilnehmer wurden wie folgt aufgeteilt:

Name	Hauptrolle
Bastian	<ul style="list-style-type: none">- Projektleiter- Entwicklung Backup Skripte- Entwicklung Displaystatusanzeige
Jonas	<ul style="list-style-type: none">- Implementierung Mail-Server- Implementierung Samba-Server- Implementierung DDNS
Tom	<ul style="list-style-type: none">- Implementierung einer WLAN-Benutzerverwaltung- Implementierung eines RADIUS-Server mittels CoovaChilli
Jakob	<ul style="list-style-type: none">- Erstellung der Dokumentation- Untersuchung von IP-Fire als Alternatives Betriebssystem
Elias	<ul style="list-style-type: none">- Überprüfung des WLAN-Chips in Bananian- Integration des WLAN und VLAN in Armbian

Die Verteilung der Rollen und mancher Aufgabengebiete änderte sich im Verlauf des Projekts. Die hier dargestellte Unterteilung bezieht sich auf den Stand zum Semesterende.

7 Zukünftige Ziele

Wenn das Projekt in den kommenden Semestern fortgeführt wird, so können folgende Verbesserungen und Implementationen umgesetzt werden:

- Überarbeitung der Backup Skripte. Manuelles Ausführen benutzerfreundlicher gestalten. (Beispielabfrage: Sicherheitsupdates überspringen? Monatliches Backup ebenfalls ausführen?)
- WLAN Sticks funktionsfähig integrieren
- Tmux Displaystatusanzeige zur Anmeldung automatisch ausführen (Remote und Lokal)
- Implementation eines Monitoring Tools um Netzlast und Zugänge an das Gerät zu überwachen
- Zugangssystem überarbeiten Alternativen finden und ggf. Implementieren SSH Zugriff von externen IP Adressen blockieren
- DynDNS komplett unabhängig von NoIP verwenden z.B. Direkte Anbindung an Cloudflare über API
- Samba Shares einbinden und Backup Repository anlegen
- Neuen WLAN Chip auf BananaPi löten und im Anschluss WLAN Treiber integrieren / konfigurieren damit Bananian OS verwendet werden kann

Literaturverzeichnis

- [1] http://static.ipfire.org/static/images/tux/ipfire_tux_512x512.png?v=5f0dbf32ef0e0715a3783ca904e5addc.
- [2] <http://www.ipfire.org/features>.
- [3] <http://www.bananapi-kaufen.de/wp-content/uploads/2014/08/logo.png>.
- [4] <https://www.bananian.org/>.
- [5] https://www.bananian.org/news#the_end_-_2017-04-02.
- [6] https://forum.armbian.com/uploads/monthly_2017_02/logo_facebook.png.52c4718b8776c2e0ee2cfc82fae297b1.png.
- [7] <https://www.armbian.com/>.
- [8] <https://wiki.ubuntuusers.de/dd/>.
- [9] <https://wiki.ubuntuusers.de/rsync/>.
- [10] <https://wiki.ubuntuusers.de/tar/>.
- [11] <https://wiki.ubuntuusers.de/Cron/>.
- [12] <https://wiki.debian.org/UnattendedUpgrades>.
- [13] <https://wiki.ubuntuusers.de/tmux/>.
- [14] <http://my5cent.spdns.de/allgemein/banana-pi-postfix-installieren-und-einrichten.html>.
- [15] https://help.ubuntu.com/community/How%20to%20Create%20a%20Network%20Share%20Via%20Samba%20Via%20CLI%20%28Command-line%20interface/Linux%20Terminal%29%20-%20Uncomplicated%2C%20Simple%20and%20Brief%20Way%21#About_This_Guide.
- [16] https://www.togaware.com/linux/survivor/No_IP_Manual.html.
- [17] http://i.techrepublic.com/downloads/PDF/SolutionBase_RADIUS_deployment_scenarios.pdf.
- [18] <http://freeradius.org/>.
- [19] <https://www.heise.de/glossar/entry/Extensible-Authentication-Protocol-397255.html>.
- [20] <https://www.vultr.com/docs/install-freeradius-on-debian-7>.

- [21] <https://wiki.freeradius.org/guide/SQL-HOWTO-for-freeradius-3.x-on-Debian-Ubuntu>.
- [22] <https://extremeshok.com/5486/debian-7-freeradius-server-mysql-authentication/>.
- [23] <https://andrewwippler.com/2016/03/11/wifi-captive-portal/>.
- [24] https://en.wikipedia.org/wiki/Captive_portal.
- [25] https://coova.github.io/img/Chilli_2.jpg.

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig verfasst und hierzu keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Stellen der Arbeit die wörtlich oder sinngemäß aus fremden Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt oder an anderer Stelle veröffentlicht.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

Furtwangen, den ??.??2017