Mein KellerPi - Projekt

Zielsetzungen - der Raspi (Modell 3b) soll:
- im Keller eingebaut werden √
- als Cloudserver (Nextcloud) zur Verfügung stehen √
- über das Internet angesprochen werden können (joes.goip.de; ddclient) √
- Meine Webseite hosten √
- sich selber ausschalten können (ggf. Hard Reset) (Plan geändert – jetzt externe HD) √
- den Stromkreis auf den Dachboden für 15s unterbrechen können √
- einen Wlan Accesspoint zur Verfügung stellen können √

Hardware:
- Raspi 3b
- Hutschienengehäuse mit Platz für ein Relais
- Zwei Relais auf einer Platine
- Jumperkabel (weiblich-weiblich)
- Hutschinennetzteil (5V)
- Mikro-USB Kabel für das Hutschienennetzteil
- Dlan-Adapter

(Idee: ggf. noch Touchscreendisplay)
- eine externe Festplatte wurde später noch ergänzt

Software:
- raspbian stretch lite
- apache2 webserver
- ddclient (mit Internetdienst; goip oder ddnss)
- Nextcloud
- hostapd

Benutzte Programme:
- Linux Mint auf Laptop
- Bluefish html-Editor

Vorbemerkung:
Das folgende Dokument ist quasi ein Protokoll meiner Aktivitäten zum Aufsetzen des Raspi wie oben beschrieben erstellt damit ich später auch noch weiß was ich getaen habe.
Die Anleitungen sind aus diversen Quellen zusammengesucht und ich hoffe ich verletze hiermit keine Copyrights.
Im wesentlichen gehe ich davon aus, auf dem Raspi als Administrator und auf dem Laptop als normaler Benutzer angemeldet zu sein. Normaler Text wird so geschrieben.
```
Befehlszeilen bzw. Code in Skripts sieht so aus.
```

Und zu guter Letzt, bei mir funktioniert‘s, wer dies jedoch als Anleitung nutzt ist selber schuld.
Nachtrag: Nach ca. 4 Jahren habe ich mal diverse Änderungen nachgepflegt. Daher mag es sein, dass einige Stellen etwas „inkonsitent“ und auch nicht mehr so ausführlich sind.
(schuchardt.j@gmail.com)

Inhaltsverzeichnis

1 Grundinstallation

1.1 Vorbereitungen auf dem Laptop:

Raspian-lite aus dem Internet herunterladen und das .img auf die SD Karte schreiben.

Image entpacken und:

```
dd bs=4M if=2017-08-16-raspbian-stretch-lite.img of=/dev/sdX conv=fsync
```

(sdX, die SD Karte, kann mit `lsblk` ausgelesen werden, kann auch mmcblkX heißen)

Ein Verzeichnis zum Einbinden des Raspi anlegen (`/mnt/pi32`) und Beutzer und Gruppe setzten.
Den entsprechenden Eintrag in der `/etc/fstab` machen:

```
192.168.1.92:/home/pi32/export   /mnt/pi32    nfs     rw,users        0 0
```

(Der Raspi soll die Adresse ...92 lokal bekommen und das entsprechende Verzeichnis exportieren)

1.2 Einstellungen die direkt am Raspi vorgenommen werden müssen

Die SD Karte in den Raspi stecken und booten (Strom und Netzwerkkabel anschließen)

Erstes login: pi

Passwort: raspberry (Achtung amerikanisches Tastaturlayout, y<=>z)

Einstellen der grundlegenden Konfiguration mit dem Tool raspi-config:

```
sudo raspi-config
```

- Passwort festlegen
- Hostname festlegen (raspi32)
- lokalization options

Schema: de_DE ISO8859-1 gewählt

Zeitzone (Europa/Berlin)

Tastatur Layout (105 Tasten/de_DE)

-interface: ssh Zugriff aktivieren

(ggf. später noch GPIO Pin Einstellungen)

- Advanced: Expand filesystem – Damit auch die ganze Karte genutzt wird.

Und zum Abschluß den Raspi neu starten.

1.3 Weitere Konfiguration über das Netzwerk

Login mit ssh vom Laptop. Die IP-Adresse kann man z.Bsp. Mit „Fing“ unter Android bequem ausgelesen werden.

```
ssh -X pi@192.168.1.xxx
```

Und da viele weitere Aktionen Administratorrechte erfordern gleich auf „root“ wechseln.

```
sudo su
```

1.3.1 Einstellen einer statischen IP-Adresse:

Mit netstat die aktuelle Netzwerkonfiguration auslesen:

```
root@raspi32:/home/pi# netstat -r -n
Kernel-IP-Routentabelle
Ziel        Router        Genmask       Flags MSS Fenster irtt Iface
0.0.0.0     192.168.1.11  0.0.0.0       UG    0 0        0    enxb827eb12dff6
192.168.1.0 0.0.0.0       255.255.255.0 U     0 0         0   enxb827eb12dff6
```

und die Datei dhcpcd.conf mit `nano /etc/dhcpcd.conf` bearbeiten.

Der entsprechende Abschnitt sieht dann so aus:

```
# Example static IP configuration:
interface enxb827eb12dff6
static ip_address=192.168.1.92/24
static routers=192.168.1.11
#static domain_name_servers=192.168.0.1 8.8.8.8
```

Das Interface hieß früher eth0, wie schon gesagt soll der Raspi die Adresse ...92 erhalten und mein Router hat die ...11 (#… sind Kommentarzeilen)

Den dhcp Dienst deaktivieren:

```
service dhcpcd stop
```

Den dhcp dienst wieder starten:

```
service dhcpcd start
```

und zur Sicherheit ein Neustart mit

```
init 6
```

Der Raspi sollte jetzt unter 192.168.1.92 erreichbar sein.

1.3.2 Neuen Benutzer anlegen (optional)

```
adduser pi32
```

Mit `id pi` die Gruppenzugehörigkeit von pi auslesen und leider für jede Gruppe einzeln pi32 die Gruppen hinzufügen.

root@raspi32:/home/pi# id pi

```
uid=1000(pi) gid=1000(pi)
Gruppen=1000(pi),4(adm),20(dialout),24(cdrom),27(sudo),29(audio),44(video
),46(plugdev),60(games),100(users),101(input),108(netdev),999(spi),998(i2
c),997(gpio)
```

`usermod -aG „Gruppenname“ pi32` also z. Bsp:

```
usermod -aG input pi32
```

Dann komplett ausloggen und mit:

```
ssh -X pi32@192.168.1.92
```

unter dem neuen Namen anmelden.

Zu guter letzt den Standardbenutzer pi deaktivieren:

```
usermod -L pi
```

oder oder und das Verfallsdatum des Users in die Vergangenheit setzten:

```
usermod -e 2017-09-30 pi
```

1.3.3 Verzeichnis zu Datenaustausch auf dem Raspi erstellen und einrichten

```
mkdir /home/pi32/export
```

(als User pi32 oder den Besitzer mit chown und chgrp anpassen)
und für alles freigeben:

```
chmod 777 /home/pi32/export
```

(ein bisschen radikal, noch optimierungsbedürftig)

Den NFS Server installieren:

```
apt-get install nfs-kernel-server
apt-get install nfs-common
```

Die /etc/exports anpassen:

```
/home/pi32/export       192.168.1.0/255.255.255.0(rw,async,no_subtree_check)
```

Derzeit werden Dateien im export Verteichnis noch als Benutzer+Gruppe „pi“ erstellt. Ggf mit

```
/home/pi32/export
      192.168.1.0/255.255.255.0(rw,async,no_subtree_check,anonuid,anongid)
```

in der /etc/exports ändern.

Die geänderte Datei neu einlesen und den NFS Server neu starten:

```
exportfs -ra
/etc/init.d/nfs-kernel-server restart
```

Auf dem Laptop kann man sich dann die exportierten Verzeichnisse vom Raspi anzeigen lassen:

```
showmount -e <nfs-server>
```

Und wenn alles läuft das Verzeichnis /home/pi32/expport vom Raspi auf dem Laptop unter /mnt/pi32 mit:

```
mount /mnt/pi32
```

einbinden.

1.3.4 Den Ordner public Ordner auf dem NAS einbinden

Als Benutzer pi32 unter /home/pi32 das Verzeichnis nas2:

```
mkdir nas2
```

erstellen und in der /etc/fstab die Zeile:

```
//192.168.1.78/public /home/pi32/nas2 cifs noauto,users 0 0
```

ergänzen.
Dann mit `mount /home/pi32/nas2` den NAS Server einbinden.

1.3.5 Den USB-Stick und die externe Festplatte einbinden (alte Variante)

Da mein Stick mit exFAT formatiert ist welches nicht von Hause aus von Linux unterstütz wird muss ich erst noch mit:

```
apt-get install exfat-fuse exfat-utils
```

die Unterstützung für das Dateisystem installieren. Ist bei ntfs, vfat und ext nicht notwendig.

Als Benutzer pi32 unter /home/pi32 das Verzeichnis Stick:
`mkdir Stick`
erstellen und in der /etc/fstab die Zeile:
`/dev/sda1 /home/pi32/Stick exfat auto,user,rw 0 0`
ergänzen. Der Stick muss natürlich als sda1 erkannt worden sein (lsblk zeigt die Geräte).
Dann könnte man mit mount `/home/pi32/Stick` den Stick einbinden.

Alternativ könnte auch `autofs` genutzt werden.

… was ich dann mit der externen Festplatte auch getaen habe: (und es hat nie zuverlässig funktioniert)
Die Festplatte hängt an einer Steckdose, die mittels Relais über die GPIO‘s und die Webseite ein- und ausgeschaltet werden kann. Die Einbindung per autofs erfolgt dann bei jedem Zugriff auf die Festplatte.

Installieren:
```
apt-get install autofs
mkdir /automount
```
Die Konfigurationsdateien anpassen / erstellen:
```
nano /etc/auto.master //Inhalt:
/automnt /etc/auto.automnt --timeout=5 –ghost

nano /etc/auto.automnt //Inhalt:
usb-hdd -fstype=ntfs,sync,uid=1001,gid=46,umask=000
:/dev/disk/by-uuid/xxxxxxxxxxxxxxx
```

(Die UUID kann z. Bsp. mit blkid -o list -w /dev/null ausgelesen werden).
Und neu starten des Autofs:

```
systemctl reload autofs
```

Die Webseite fragt dann vor dem Ausschalten der Steckdose ab, ob die Festplatte noch eingehängt ist. Ist sie das, wird die Steckdose nicht ausgeschaltet um Datenverlust zu vermeiden.

Die Platte dann auch noch per NFS oder SAMBA zu exportieren habe ich noch nicht versucht.

1.3.6 Den USB-Stick und die externe Festplatte einbinden (neue Variante)

Nach diversen Schwierigkeiten habe ich mich entschieden sowohl Stick als auch Festplatte (beide jetzt mit NTFS formatiert) ohne Rechteüberwachung einzubinden.

Hierzu müssen die NTFS-Tools installiert werden,

„Google NTFS3g“

die blockid der Geräte ausgelesen werden,

blkid
(die ausgabe spare ich mir hier, die relevante ID ist unten fett markiert)

die Verzeichnisse angelegt werden in die /*etc/*fstab folgende Zeilen aufgenommen werden:

```
PARTUUID=27d6e4bd-01 /home/pi32/Stick ntfs defaults    0 2
PARTUUID=78bfa3d7-01 /mnt/usb-hdd ntfs defaults        0 2
```

Da die externe Festplatte beim hochlaufen ausgeschaltet ist/wird scheitert dies natürlich für sie, das `noauto` habe ich mir aber gespart.

Damit steht der Stick zwar als Eigentum des „root“ zur Verfügung, es dürfen aber alle lesen und schreiben.

An dem Einhängen der Festplatte, wenn diese eingeschaltet wird, arbeite ich noch.

2. Installation des Webservers

2.1 Der Webserver selber

Als Webserver wird „Apache2“ verwendet.

Als erstes mal die Packetquellen auf den neuesten Stand bringen:
`apt-get update`
und das System auf den neuesten Stand bringen:
`apt-get upgrade`
Hiebei die bereits geänderte dhcpcd.conf nicht ünerschreiben lassen.

Dann kann der Webserver heruntergeladen werden:
`apt-get install apache2`

Nach der Installation läuft der Webserver sofort, durch Eingabe der IP im Browser erscheint die Standardseite der apache Distribution. Die Webseite liegt auf dem Raspi unter `/var/www/html/index.html` und wird in `indexbak.html` mit
`mv index.html indexbak.html`
umbenannt. Die Datei `index.html` enthält später meine Webseite bzw. den Verweis darauf.

2.2 Die Weiterleitung auf den Webserver

Um den Webserver von außen erreichbar zu machen muß nun die dynamische Weiterleitung eingerichtet werden.
Ich nutze die Dienste goip.de oder ddnss.de. Beide kostenfrei. Die Einrichtung wird auf den jeweiligen Webseiten beschrieben. Daher gehe ich hier hierauf nicht ein.
Wenn man ein Account angelegt hat muss der Raspi dem dienst eigentlich nur noch die IP die der Router von außen hat mitgeteilt werden und dem Router gesagt werden, dass er Anfragen auf den Raspi umleiten soll.

2.2.1 Auslesen und versenden der IP Adresse

Hierzu wird das Paket ddclient genutzt. Installation mit:
`apt-get install ddclient`

Während der Installation werden diverse Informationen abgefragt. Da aber für goip keine Standardeinträge vorhanden sind wird die Datei `/etc/ddclient.conf` manuell angepasst und sieht dann so aus:

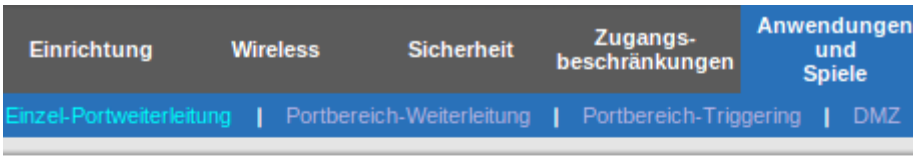
```
# Configuration file for ddclient

### manuell erstellt
#
# /etc/ddclient.conf

protocol=dyndns2
daemon=300
syslog=yes
mail=root
mail-failure=root
pid=/var/run/ddclient.pid
```

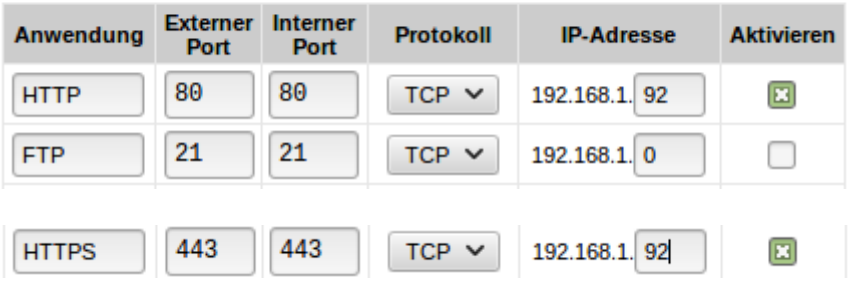
```
use=if, if=xxxName der Netzwerkkartexxx

#### Einträge f goip.de:
use=web, web=checkip.dyndns.com, web-skip='IP Address'
server=www.goip.de
login=xxx goip Loginname xxx
password=xxx goip Passwort xxx
xxx Internetadresse z. Bsp.:joes.goip.de xxx

#### Einträge f ddnss:
#server=www.ddns.de
#use=web, web=checkip.dyndns.com, web-skip='IP Address'
#login=xxx ddnss Loginname xxx
#password=xxx ddnss Passwort xxx
#xxx Internetadresse z. Bsp.:joes.ddns.de xxx
```

Zum starten einfach als root `ddclient` aufrufen.

2.2.2 Konfiguration des Routers

Der Raspi soll für die Anfragen zu http und https über das Internet erreichbar sein. Diese laufen auf Port 80 (http) und Port 443 (https). Ggf. kann man sich noch ssh (Port 22)auf den Raspi leiten, wenn man ihn auch über das Internet konfigurieren möchte.
Hierzu wird in der Routingtabelle des Routers die IP des Raspi (also 192.168.1.92) zu den jeweiligen Ports eingetragen.
Bei mit läuft das in der Routerkonfiguration im Reiter „Anwendungen und Spiele“ und sieht dann so aus:





(ftp ist zum Beispiel nicht aktiv).

Jetzt sollte die (noch nicht vorhandene) Internetseite unter joes.goip.de über das Internet erreichbar sein. Testweise kann man „irgendetwas“ in die index.html schreiben und über das Mobiltelefon die Seite aufrufen. Dann sollte auch „irgendetwas“ angezeigt werden.

2.3 Einstellen der verschlüsselten Datenübertragung

Als erstes werden hierfür Schlüssel und Zertifikat erzeugt:
`openssl genrsa -out /etc/ssl/private/myssl.key 4096` erzeugt den Schlüssel.

```
openssl req -new -key /etc/ssl/private/myssl.key -out
/etc/ssl/certs/myssl.csr
```
erzeugt ein sog. !“Signing request“, damit kann man entweder bei einem Dienstleister einen Schlüssel kaufen oder mit

```
openssl x509 -req -days 365 -in /etc/ssl/certs/myssl.csr -signkey
/etc/ssl/private/myssl.key -out /etc/ssl/certs/myssl.crt
```
das Zertifikat selber erstellen. Nachteil dabei ist, das die meißten Browser so ein Zertifikat als nicht vertrauenswürdig einstufen..
Dabei werden diverse Informationen abgefragt, der „Common Name“ „192.168.1.92“ (bzw. „joes.goip.de“ von außen) sollte angegeben werden, da sonst später die Browser meckern, dass das Zertifikat nicht gültig ist.

Dem Apache Server muß dann noch mitgeteilt werden auch ssl Anfragen (Port 443) zu Verarbeiten.
Mit `nano /etc/apache2/ports.conf` kann man sich die Konfigurationsdatei anzeigen lassen.
Meine sieht dann so aus:

```
Listen 80
<IfModule ssl_module>
        Listen 443
</IfModule>
<IfModule mod_gnutls.c>
         Listen 443
</IfModule>
```

Gegebenenfalls muß dann die Apache Konfiguration neu eingelesen werden:
`service apache2 reload`
und das ssl Modul aktiviert werden:

```
a2enmod ssl
```
War bei mir nicht erforderlich, da schon aktiv.

Abschließend muss nur noch ein Virtual Host eingerichtet werden. Für die Konfiguration mit

nano /etc/apache2/sites-available/ssl.conf mit folgendem Inhalt erstellen:

```
<VirtualHost 192.168.1.92:443>
    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/myssl.crt
    SSLCertificateKeyFile /etc/ssl/private/myssl.key
    # Pfad zu den Webinhalten
    DocumentRoot /var/www/html/
    Header always set Strict-Transport-Security "max-age=31536000;
includeSubDomains;"
</VirtualHost>
```
VirtualHost dann mit

```
a2ensite ssl.conf
```

```
service apache2 force-reload
```

aktivieren.

Ob‘s geplappt hat kann man im Browser mit dem Aufruf von https://192.168.1.92/ überprüfen. Firefox meldet dann eine unsichere Seite mit der Begündung:

The certificate is not trusted because it is self-signed.

Da es selbst erstellt wurde ist erstes klar, und erstellt wurde es für joes.goip.de nicht 192…, wird sich später erledigen.

Da ich den Server später auch als Cloud nutzen möchte soll er auch gleich Verschlüsselte Verbindungen erzwingen.

Zur Umleitung das Modul rewrite aktivieren
```
chown -R www-data /run/apache2
a2enmod actions
/etc/init.d/apache2 force-reload
```

und in jedem Verzeichnis, dass nur über ssl erreichbar sein soll die Datei .htaccess anlegen.

Diese muß mindestens die drei Zeilen am Ende enthalten:

```
RewriteEngine On
RewriteCond %{HTTPS} !=on
RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

(Das L in [L,R=301] sagt dem Server, das hier die Datei zu Ende ist.)
Zusätzlich ergänze ich noch:

```
options -Indexes +FollowSymLinks
```

-Indexes verhindert das Auflisten von Verzeichnissen, +FollowSymLinks wird für die Umleitung auf ssl benötigt.

(Es sollte eigentlich reichen diese drei Zeilen im Dockumentroot (/var/www/html) des Webservers anzugeben da angeblich die Unterverzeichnisse mit umgeleitet werden. Bin mir aber nicht sicher. Bei Nextcloud nichts geändert da größere Konfigurationsdatei und funktioniert trotzdem.)

2.4 Vorbereiten eines passwortgeschützten Bereiches

Hierzu muss zum einen ein Passwort erzeugt werden:

```
htpasswd -cs /etc/apache2/passwd testuser
```

Das zu schützende Verzeichnis angelegt werden:

```
mkdir /var/www/html/pwd
```

sowie in dem zu schützenden Verzeichnis die Datei .htaccess angelegt werden, also:

```
nano /var/www/html/pwd/.htaccess
```

Inhalt:

```
Options -Indexes +FollowSymLinks
AuthType Basic
AuthUserFile /etc/apache2/ passwd
AuthName "KellerPi_aktionen"
require valid-user
IndexIgnore *
DirectoryIndex relais.html
```

In der Apache Konfiguration /etc/apache2/sites-avaliable/000-default.conf muss noch die Option „AllowOverride“ auf „All“ gesetzt werden.

3 Den Cloud-Server aufsetzten.

Hierzu wird das Paket „Nextcloud“ genutzt.

Quelle:

https://nextcloud.com/install/#instructions-server

Das nextcloud.zip herunterladen und nach /mnt/pi32 entpacken (also auf den pi32 in das Verzeichnis /home/pi32/export).

Da mir gerade aufgefallen ist, dass das Tool Midnight Commander noch nicht installiert ist, es eben mit:
`apt-get install mc`
installieren.

3.1 Nextcloud installieren

Damit das Verzeichnis /home/pi32/nextcloud nach /var/www/nextcloud verschieben.
Den Benutzer und die Gruppe ändere ich auf www-data mit:
```
chgrp -R www-data ./nextcloud
chgrp -R www-data ./nextcloud
```
(hier muss man im Verzeichnis /var/www sein sonst anstelle von ./nextcloud /var/www/nextcloud)

Dann dem Nextcloud Admin Manuals folgen.

Packete installieren:
(Das dauert)
```
apt-get install apache2 mariadb-server libapache2-mod-php7.0
```
(ca. 5 min)
```
apt-get install php7.0-gd php7.0-json php7.0-mysql php7.0-curl php7.0-mbstring
```
(ca. 1 min)
```
apt-get install php7.0-intl php7.0-mcrypt php-imagick php7.0-xml php7.0-zip
```
(ca. 3 min)

(apache2 sollte eigentlich nicht notwendig sein, da er schon läuft, schadet aber nicht, da apt das dann überspringt)

3.2 Konfigurieren des Apache2 für Nextcloud

Mit
`nano /etc/apache2/sites-available/nextcloud.conf`

folgendes einfügen:

„
Alias /nextcloud "/var/www/nextcloud/"

```
<Directory /var/www/nextcloud/>
  Options +FollowSymlinks
  AllowOverride All
```

```
 <IfModule mod_dav.c>
  Dav off
 </IfModule>

 SetEnv HOME /var/www/nextcloud
 SetEnv HTTP_HOME /var/www/nextcloud

</Directory>
```

”

Mit
`ln -s /etc/apache2/sites-available/nextcloud.conf /etc/apache2/sites-enabled/nextcloud.conf`
dewm Apache mitteilen, dass da noch eine Webseite liegt.

Weiteres Modul aktivieren:
`a2enmod rewrite`
Die folgenden laufen bei mir schon:
```
a2enmod headers
a2enmod env
a2enmod dir
a2enmod mime
a2enmod setenvif
```

Nach `service apache2 restart` kann man den Server unter

https://192.168.1.92/nextcloud/

starten. Noch nicht einloggen, erst noch die SQL Datenbank MariaDB konfigurieren:

```
mysql -u root -p

grant all on *.* to root@localhost identified by '***Passwort***' with grant option;


flush privileges;

quit;
```

und das Verzeichnis `/home/pi32/Cloud` als Basis für die Cloud anlegen.

Mit

```
chown www-data /home/pi32/Cloud

chgrp www-data /home/pi32/Cloud
```

noch den Benutzer anpassen und dann kann man sich zum erstem mal einloggen.

Administratorname und Passwort vergeben, im Datenverzeichnis muß stehen:

`/home/pi32/Cloud/`

Der Datenbankbenutzer ist root mit dem gerade erstellten ***Passwort***. Nextcloud und localhost bleiben unverändert.

3.3 Einbinden des USB Sticks in die Cloud

Der Stick ist in das Verzeichnis /home/pi32/Stick eingebunden. Um Ihn auch in der Cloud nutzbar zu machen wird er mittels `bind` noch im Datenverzeichniss der Cloud eingebunden.
Hierzu in der `/etc/fstab` folgendes ergänzen:

```
/home/pi32/Stick/Nextcloud-Files/ /var/www/html/nextcloud/data/pi32/files/Stick/  none bind
```

Um die bereits auf dem Stick exitierenden Dateien in Nextcloud nutzbar zu machen muß aus dem Verzeichnis /var/www/html/nextcloud das Tool occ als Benutzer www-data ausgeführt werden:

```
sudo -u www-data php occ files:scan pi32
```

Hierzu ein kurzes Shellskript was ich unter /root ablege:
`nano nextcloud_aktualisieren` (anlegen)

Inhalt:

```
cd /var/www/html/nextcloud
sudo -u www-data php occ files:scan pi32
cd /root
```

`chmod 744 nextcloud_aktualisieren` (Besitzer darf ausführen, andere lesen)

(Aktualisierung nur für den Nutezer „pi32“, die Option -all hat nicht funktioniert.)

Der Stick (sowie die externe Festplatte) werden derzeit ohne Rechteüberwachung eingebunden. Möglich ist noch, den Stick in die /etc/exports einzutragen und als nfs bereitzustellen.

3.4 Die vertrauenswürdigen Adressen einstellen:

In der

`/var/www/nextcloud/config/config.php`

den Punkt „trusted domains“ anpassen:

```
'trusted_domains' =>
 array (
    0 => 'xxxxxxxxxxxxxxx',
    1 => 'xxxxxxxxxxxxxxx',
    2 => 'xxxxxxxxxxxxxxx',
    2 => 'joes.goip.de',
    3 => 'xxxxxxxxxxxxxxx',
  ),
```

Die bis auf die 2 sind hier nicht relevat darum ausgext.

3.5 Empfohlene Optimierungen

Noch lokalen cache einrichten:
`nano /var/www/nextcloud/config/config.php`
Vorletzte Zeile einfügen:
`'memcache.local' => '\OC\Memcache\APCu',`
Funktioniert bei mir nicht, also weggelassen.

und OPcache einrichten:

```
nano /etc/php/7.0/apache2/php.ini
```

Dort einfügen:

```
opcache.enable=1
opcache.enable_cli=1
opcache.interned_strings_buffer=8
opcache.max_accelerated_files=10000
opcache.memory_consumption=128
opcache.save_comments=1
opcache.revalidate_freq=1
```

dann noch ein Neustart:

```
service apache2 restart
```

3.6 Die Dateigrößenbeschränkung für den Upload entfernen

Randbemerkung: Zu diesem Zeitpunkt habe ich diverse Schönheitseinstellungen wie Hintergrundbild, Benutzerbild … gemacht.

Dann mit:

```
nano /etc/php/7.0/apache2/php.ini
```

die Einträge

```
upload_max_filesize = 16000M

post_max_size = 16400M

output_buffering = 0

max_input_time 7200

max_execution_time 7200
```

```
memory_limit = 256M
```
(die Anleitung sagt zwar 1024, da dass aber alles ist was der Raspi hat nur 256)

ändern.

Und:

```
nano /var/www/nextcloud/.htaccess
```

In <IfModule mod_php5.c> und <IfModule mod_php7.c>

die Einträge auskommentieren:

```
#  php_value upload_max_filesize 511M

#  php_value post_max_size 511M

#  php_value memory_limit 512M
```

(stammt aus einer owncloud Anleitung)

und den Server neu starten:

```
service apache2 restart
```

4 Wlan accesspoint

Nach den Kommentaren zur Leistung am Ende von
https://www.elektronik-kompendium.de/sites/raspberry-pi/2002161.htm
weiß ich noch nicht ob ich das wirklich nutzen werde. Einrichten tu ich‘s trotzdem mal.

4.1 Installieren der Pakete

Mit

```
iw list | grep AP
```

überprüft ob dir Wlantools installiert sind.

Den dhcpcd hatten wir zwar oben schon mal

```
systemctl status dhcpcd
```

liefert aber in „active“.

Die Netzwerkgeräte werden durch

```
ip l
```

wie folgt angezeigt:

```
1: lo: <LOOPBACK,UP,LOWER_UP> …..
2: enx*****MAC****: <BROADCAST,MULTICAST,UP,LOWER_UP> ……
3: wlan0: <NO-CARRIER,BROADCAST,MULTICAST,UP>……
```

sieht erst mal gut aus.

`apt-get update` (die Liste der Pakete aktualisieren) und `apt-get upgrade` (die installierten Pakete aktualisieren) kann man auch mal wieder machen.

Als Dienst wird der Host Access Point Daemon (HostAPD) verwendet und Helfer für die Netzwerkbrücke gebraucht. Installieren mit

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

4.2 Konfigurieren des Hostadapters

Dann die Konfigurationsdatei für den Host Adapter schreiben, mit

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

sollte eine neue Datei erstellt werden, das Verzeichnis wurde während der Installation schon angelegt. Diese sieht dann so aus:

```
# Bridge-Betrieb
bridge=br0
# Schnittstelle und Treiber
interface=wlan0
#driver=nl80211
# WLAN-Konfiguration
ssid=KellerPi
channel=1
hw_mode=g
ieee80211n=1
ieee80211d=1
country_code=DE
wmm_enabled=1
# WLAN-Verschlüsselung
auth_algs=1
wpa=2
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
wpa_passphrase=***WPA-Passwort***
```

Wobei ssid=, channel= und wpa_passphrase angepasst werden sollten. Da die Datei das Wlan Passwort enthält werden die Rechte so angepasst, dass nur root sie lesen kann:

```
chmod 600 /etc/hostapd/hostapd.conf
```

4.3 Einrichten der Netzwerkbrücke

Die /etc/network/interfaces anpassen:

Dem Inhalt

```
# Netzwerkbrücke
auto br0
iface br0 inet static
 address 192.168.1.92
 netmask 255.255.255.0
 gateway 192.168.1.11
 bridge_ports wlan0 enx*****MAC****
 bridge_fd 5         #ist ein delay, ggf. auf 0 setzen
 bridge_stp yes      #unter Umständen auf 'no' setzen - ist ein Protokoll
zur optimierung des Datenverkehrs
```

hinzufügen.

Nach dem Neustart mit `init 6` zeigt `brctl show` folgendes an:

```
bridge name  bridge id          STP enabled     interfaces
br0          8000.000000000000  no              enbx……..
```

(eigentlich müsste bei Interfaces auch wlan0 erscheinen, kann sein das das Feld einfach zu klein ist)

4.4 Start und Stop des Accesspoints

Der AP kann mit
`hostapd /etc/hostapd/hostapd.conf`
gestarted werden.

Danach kann ich mich mit dem Telefon mit dem AP verbinden und habe sogar eine Internetverbindung bei ausgeschalteten mobilen Daten. Beenden kann man ihn dann mit <Strg+C>.

Damit hostapd als Dämon automatisch startet wird noch der die Konfigurationsdatei in /etc/default eingetragen. Dazu mit:
`nano /etc/default/hostapd`
die Zeile:
`DAEMON_CONF="/etc/hostapd/hostapd.conf"`
einfügen (bzw. die Auskommentierung löschen).

Hat dann allerdings zur folge, dass der Accesspoint nach einem Neustart des Raspi auch automatisch startet. Damit er nicht dauernd läuft muß er zum einen durch:

`nano /etc/rc.local`
Vorletzte Zeile:
`systemctl stop hostapd`

wieder angehalten werden.

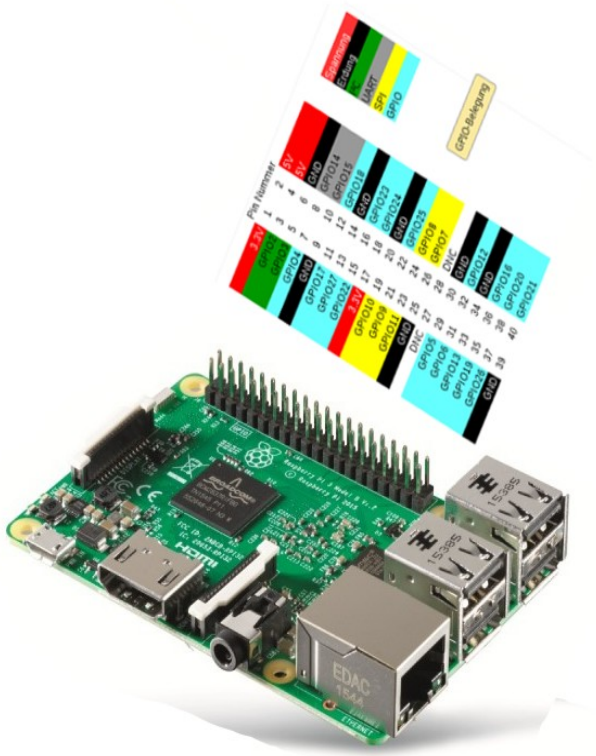
Das umbenennen Links in /etc/rc2.d bis /etc/rc5.d
```
mv /etc/rc2.d/S01hostapd /etc/rc2.d/K01hostapd
mv /etc/rc3.d/S01hostapd /etc/rc2.d/K01hostapd
mv /etc/rc4.d/S01hostapd /etc/rc2.d/K01hostapd
mv /etc/rc5.d/S01hostapd /etc/rc2.d/K01hostapd
```
hat nicht funktioniert.

Nebenbei:
Beim Starten durchläuft Linux die Runlevel 0,S,1 – 5, per Definition:
0 – aus
S – Einzelbenutzer mit rudimentären Funktionen
1 – Einzelbenutzer ohne Netzwerk
2 – Einzelbenutzer mit Netzwerk
3 – Mehrbenutzer mit Netzwerk
4 – nicht einheitlich
5 – Mehrbenutzer mit Netzwerk und x-Server
6 – Neustart
Beim Durchlaufen der Runlevel werden die Skripts, die in die Verzeichnisse /etc/rc0.d bis /etc/rc6.d verlinkt sind, abgearbeitet. Bei den Links K01“Skriptname“ wird dabei der Parameter „stop“ an das Sktipt übergeben, bei den Links S01“Skriptname“ der Parameter start.

Der Dämon wird dann mit:
```
starten:     systemctl start hostapd
             systemctl enable hostapd
neustart:    systemctl restart hostapd
stoppen:     systemctl stop hostapd
status:      systemctl status hostapd
```
gesteuert (Das wäre später was fürs Touchscreen oder die Webseite).

5 GPIO Relaissteuerung

5.1 Die GPIO Schnittstelle



GPIO: Heißt ausgeschrieben General-Purpose Input/Output. Das heißt soviel, dass diese Schnittstelle für Ein- und Ausgaben genutzt werden kann, jedoch nicht näher spezifiziert ist. Somit kann man diese Schnittstelle selber nach eigenem Ermessen verwenden.
I2C Interface Pins: Sind besondere Pins, mit denen entsprechende Hardware über nur 2 Pins verwendet werden kann. Es sind so gesehen spezialisierte Pins.
SPI Interface Pins: Funktionieren genau so wie die I2C Pins, jedoch mit einem anderen Standard.
UART - Rx und Tx Pins: Mit diesen Pins kann man serielle Geräte ansteuern. Wobei hier Rx für Receiver, also Empfänger steht und Tx für Transmitter, also Sender.

Die Pins sind wie folgt belegt:

WiringPi	Pin-Namen	Pin	Pin	Pin-Namen	WiringPi
–	+ 3,3 V	**1**	**2**	+ 5 V	–
8	(SDA1) GPIO 2	**3**	**4**	+ 5 V	–
9	(SCL1) GPIO 3	**5**	**6**	GND	–
7	(GPIO_GCLK) GPIO 4	**7**	**8**	GPIO 14 (TXD0)	15
–	GND	**9**	**10**	GPIO 15 (RXD0)	16
0	(GPIO_GEN0) GPIO 17	**11**	**12**	GPIO 18 (GPIO_GEN1)	1

2	(GPIO_GEN2) GPIO 27	**13**	**14**	GND	–
3	(GPIO_GEN3) GPIO 22	**15**	**16**	GPIO 23 (GPIO_GEN4)	4
–	+ 3,3 V	**17**	**18**	GPIO 24 (GPIO_GEN5)	5
12	(SPI_MOSI) GPIO 10	**19**	**20**	GND	–
13	(SPI_MISO) GPIO 9	**21**	**22**	GPIO 25 (GPIO_GEN6)	6
14	(SPI_SLCK) GPIO 11	**23**	**24**	GPIO 8 (SPI_CE0_N)	10
–	GND	**25**	**26**	GPIO 7 (SPI_CE1_N)	11
30	(nur für I2C) ID_SD	**27**	**28**	ID_SC (nur für I2C)	31
21	GPIO 5	**29**	**30**	GND	
22	GPIO 6	**31**	**32**	GPIO 12	26
23	GPIO 13	**33**	**34**	GND	
24	GPIO 19	**35**	**36**	GPIO 16	27
25	GPIO 26	**37**	**38**	GPIO 20	28
	GND	**39**	**40**	GPIO 21	29

Verwenden werde ich 38 und 40 da 1 bis xxx ggf. später für das Display frei bleiben sollen. Also sollen GPIO 20 und 21 geschaltet werden.

Mit:

```
root@raspi32:/home/pi32# python
Python 2.7.13 (default, Jan 19 2017, 14:48:08)
[GCC 6.3.0 20170124] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import RPi.GPIO as GPIO
>>> GPIO.setmode(GPIO.BCM)
>>> RELAIS_1_GPIO = 21
>>> GPIO.setup(RELAIS_1_GPIO, GPIO.OUT)
>>> GPIO.output(RELAIS_1_GPIO, GPIO.HIGH)
>>> GPIO.output(RELAIS_1_GPIO, GPIO.LOW)
```

kann schon mal 3,3V Spannung an Pin 33 ein und ausgeschaltet werden.

5.2 Anlegen der Skrips zur Steuerung der GPIO‘s

Die Skripts lege ich in /home/ich/gpio ab, also:

```
mkdir gpio
```

Da das ganze über Python laufen soll werden zum einen das Python Skript sowie auch das Shell Skript zum Ausführen des Python Skriptes benötigt. Die Skripte bekommen den selben Namen wobei das Python Skript den Zusatz .py bekommt.

Das Shell Skript zum öffnen des Relais 1:

Anlegen:	`nano r1-open`
Inhalt:	`python r1-open.py`
Berechtigungen setzen:	`chmod 775 r1-open`

(Besitzer und Gruppe darf alles, jeder darf ausführen, ggf. später anpassen, weiß noch nicht genau wer die Steuern soll (www-data von der Homepahe aus?)

Das Python Skript:
Anlegen: `nano r1-open.py`
Inhalt:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
RELAIS_1_GPIO = 21
GPIO.setup(RELAIS_1_GPIO, GPIO.OUT)
# GPIO.output(RELAIS_1_GPIO, GPIO.HIGH)
GPIO.output(RELAIS_1_GPIO, GPIO.LOW)
```

Berechtigungen setzen: `chmod 774 r1-open.py`
(Besitzer und Gruppe darf alles, jeder darf lesen)

Die Shell-Skripte sind analog und werden nicht jedes mal wieder angeführt. Einfach kopieren und das entsprechende Python Skript eintragen.
Das Python Skript zum schließen ebenfalls mit
`cp r1-open.py r1-close.py`
kopieren und dann einfach die Auskommentierung ändern:

```
GPIO.output(RELAIS_1_GPIO, GPIO.HIGH)
# GPIO.output(RELAIS_1_GPIO, GPIO.LOW)
```

`./r1-open` und `./r1-close` funktionieren sogar.

Das gleiche dann noch für das zweite Relais mit

```
./r2-open
r2-open.py
./r2-close
r2-close.py
```

und der geänderten Zuordnung in den .py Skripten:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
RELAIS_2_GPIO = 20
GPIO.setup(RELAIS_2_GPIO, GPIO.OUT)
# GPIO.output(RELAIS_2_GPIO, GPIO.HIGH)
# GPIO.output(RELAIS_2_GPIO, GPIO.LOW)
```

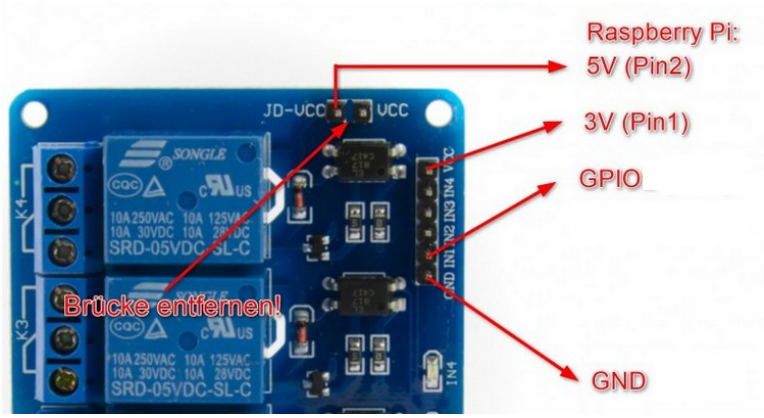
Zum unterbrechen des Stromkreises auf den Dachboden mit Relais zwei wird noch das Skript `r2-reset.py` erstellt:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
RELAIS_2_GPIO = 20
GPIO.setup(RELAIS_2_GPIO, GPIO.OUT)
GPIO.output(RELAIS_2_GPIO, GPIO.LOW)
```

```
time.sleep(15)
GPIO.output(RELAIS_2_GPIO, GPIO.HIGH)
```

5.3 Das Relais

Zum Schalten benötigt das Relais eine konstante Spannungsversorgung von 3 und 5 Volt, siehe:



Das erschwert zum einen die Verwendung meines Touchscreens, das Pin 1 und 2 benötigt werden. Des weiteren wird das selber neustarten erschwert da das wegschalten des Spannung Spannung benötigt.

Das Relais hat Stromlos Durchgang auf COM-NC1 (NC – no current?), sobald sie Schnittstelle aktiviert wird Schaltet das Relais um und damit würde der Raspi ausgehen.

Für die Relais bedeutet rx-close dass COM-NCx auf Durchgang geschaltet werden.

Das ganze sieht dann so aus:

5.4 GPIO Steuerung über die Webseite

Per php können Python Skripte ausgeführt werden.
Da der Apache aber als Benutzer www-data auftritt müssen ihm erst noch die Berechtigungen für den GPIO zugriff erteilt werden:

```
adduser www-data gpio
```

Kann mit

```
deluser www-data gpio
```

wieder rückgängig gemacht werden.

Da gewisse Python Skripte root-Rechte erfordern müssen sie der sudo‘ers Gruppe hinzugefügt werden.

```
/etc/sudoers
#includedir /etc/sudoers.d
```
!!!Wichtig: Das # muß sein, sonst funktioniert sudo nicht mehr, seeehr unpraktisch!!!

in dem Verzeichnis `/etc/sudoers.d` eine Datei mit dem Inhalt:
```
www-data ALL=(ALL) NOPASSWD: /var/www/html/pwd/
```
erstellen.

Die entsprechenden Skripte sind dann auf der Webseite realisiert.
Siehe Anhang.

6 Touchscreen

Auf Grund der gpio-Belegung für das Relais (const3V und const 5V) auf Eis gelegt.

7 Cron Dämon für Backup / Watchdog

Der cron-Dämon läuft bereits auf menem Raspi.

Es müssen nur noch die auszuführenden Skripts in die /etc/crontab eingetragen werden:

Also als root:

```
nano /etc/crontab
```

und am Ende Einfügen:

```
#* * * * * root /home/pi32/apache2/apache2_status.sh
0  1 * * * root /root/nextcloud_aktualisieren.sh
0  3 * * 1 root /root/SD_bereinigen.sh
0  4 * * * root /root/Backup_SD_to_Stick.sh
```

[kurz zur corntab:

Das Skript /… (letzter Eitrag mit vollst. Pfad) wird als root (vorletzter Eintrag) zu den Zeiten:

[Minute]	[Stunde]	[Tag im Monat]	[Monat]	[Tag in der Woche]	
*	*	*	*	*	(1)
0	2	*	*	*	(2)
0	2	*	*	0	(3)
0/10	*	*	*	*	(4)

(1) – jede Minute

(2) – jeden Tag um 2 Uhr

(3) – jeden Sonntag um 2 Uhr

(4) – alle 10 Minuten

ausgeführt.]

Die einzelnen Skripte (ohne das Testskript zum Apache) schreiben jeweils Start und Stop in die Datei

`/home/pi32/Cloud/raspicloud/files/Stick/cron.log.`

Zum einen sieht man dann ob sie ausgeführt wurden und wie lange das dauert. Gerade während des Backups sollte nichts anderes laufen.

7.1 Ausgeben wie viele Instanzen von Apache gerade laufen

(Eigentlich nur ein Test, derzeit in crontab auskommentiert)
`nano /home/pi32/apache2/apache2_status.sh`
Inhalt
```
#! /bin/bash
case "$(pidof apache2 | wc -w)" in
0)  echo "Restarting:     $(date)" >> /var/log/cgminer.txt
    echo "Apache läuft nicht"
    ;;
*)  # all ok
        echo "Es laufen $(pidof apache2 | wc -w) Instanzen von Apache"
        echo "Es laufen $(pidof apache2 | wc -w) Instanzen von Apache: $
(date)" >> /home/pi32/apache2/Meine$
    ;;
esac
```
Und ausführbar machen:
chmod 744 `/home/pi32/apache2/apache2_status.sh`

7.2 SD-Karte auf Stick spiegeln (experimentell)

Da ein 1:1 Image eine 16GB Datei erzeugt, die 1:1 die SD-Karten Daten enthält (auch den ungenutzten Bereich), soll die Datei auch noch komprimiert werden. Zur besseren Komprimierbarkeit wird der ungenutzte Bereich erst mal mit „0“en mit dem Skript /root/SD_bereinigen.sh überschrieben:

Also:

`nano /root/SD_bereinigen.sh`

Inhalt:

```
echo "$(date) - SD-Karte bereinigen gestartet" >>
/home/pi32/Cloud/raspicloud/files/Stick/cron.log

dd bs=1M if=/dev/zero of=/root/dummy.leer

rm /root/dummy.leer

echo "$(date) - SD-Karte bereinigen fertig" >>
/home/pi32/Cloud/raspicloud/files/Stick/cron.log
```

einfügen und

`chmod 744 /root/SD_bereinigen.sh`

(Das erzeugt einfach eine riesen Datei voller „0“en die dann auch gleich wieder gelöscht wird. Nebeneffekt: Das ist auch ein sicheres Löschen des ungenutzten Karteninhaltes. Das komprimierte Image meiner 16 GB Karte mit eff. ca. 2 GB benutzt verringert sich dadurch von ca. 12 GB auf knapp 800 MB. Sollte ca. wöchentlich ausgeführt werden.)

Das Skript
`./root/Backup_SD_to_Stick`
erstellt dann ein vollständiges Image der SD-Karte unter:
`/home/pi32/Cloud/raspicloud/files/Stick/pi32_autobackup-Backup.img`

(Da die SD-Karte noch eingebunden ist, ist dies jedoch ggf. unzuverlässig. Hat jetzt jedoch zwei mal funktioniert.)

Also:
`nano /root/Backup_SD_to_Stick.sh`
Inhalt:

```
echo "$(date) - SD-Karten Backup gestartet" >>
/home/pi32/Cloud/raspicloud/files/Stick/cron.log
dd bs=1M if=/dev/mmcblk0 of=/home/pi32/Cloud/raspicloud/files/Stick/pi32_autobackup.img

echo "$(date) - SD-Karten Backup komprimieren gestartet" >> /home/pi32/Stick/cron.log
zip /home/pi32/Cloud/raspicloud/files/Stick/pi32_autobackup.zip
/home/pi32/Cloud/raspicloud/files/Stick/pi32_autobackup.img
ls -l /home/pi32/Cloud/raspicloud/files/Stick/cron.log >> /home/pi32/Stick/cron.log

echo "$(date) - SD-Karten Backup l?schen des Roh-Images" >>
/home/pi32/Cloud/raspicloud/files/Stick/cron.log
rm /home/pi32/Cloud/raspicloud/files/Stick/pi32_autobackup.img

echo "$(date) - SD-Karten Backup fertig" >>
/home/pi32/Cloud/raspicloud/files/Stick/cron.log
```

Und ausführbar machen:
`chmod 744 /root/Backup_SD_to_Stick.sh`

welches dann mit zip komprimiert und die Rohdatei löscht.

7.3 Neu einlesen der Nextcloud Dateien

Falls jemand mal wieder manuell was ins Cloud-Verzeichnis kopiert hat und vergessen hat die Datenbank für das Webinterface zu aktualisieren:
`nano /root/nextcloud_aktualisieren.sh`
Inhalt:

```
echo "$(date) - Nextcloud aktualisierung gestartet" >>
/home/pi32/Cloud/raspicloud/files/Stick/cron.log
cd /var/www/nextcloud
sudo -u www-data php occ files:scan --all
cd /root
echo "$(date) - Nextcloud aktualisierung fertig" >>
/home/pi32/Cloud/raspicloud/files/Stick/cron.log
```

(Der Verzeichniswechsel ist unschön, aber ich weiß nicht ob das nextcloud-skipt occ aus jedem Vereichnis aufgerufen sauber arbeitet.
Und ausführbar machen:
`chmod 744 /root/nextcloud_aktualisieren.sh`

8 Fertig

Eingebaut sieht das ganze dann so aus:

und stellt tatsächlich u.a. diese Website zur Verfügung.