Homebridge-Setup

Autor	Stefan Schustereit
Erstellt	Freitag, 1. Juni 2018
Letzte Änderung	Sonntag, 25. November 2018
Durch	sschuste
Revision	45

Bitte beachten:

Diese Anleitung zum Homebridge-Setup ist veraltet. Du kannst sie zwar verwenden, aber Erweiterungen und Korrekturen gibt es nur noch hier:

https://smartapfel.de/homebridge/

Fragen, Korrekturen, Wünsche und Hilferufe kannst du hier loswerden:
https://forum.smartapfel.de/forum/thread/1636-homebridge-installationsanleitungen/

Inhaltsverzeichnis

Was ist Homebridge?

Homebridge ist eine Software, die so programmiert ist, dass sie als „Service“ auf einem Computer läuft. Als Service (deutsch: Dienst) bezeichnet man Programme, die im Hintergrund laufen und eine Schnittstelle in ein Netzwerk anbieten, die andere Computer zum Austausch von Daten verwenden können.

Bekannte Beispiele sind der Webserver oder der Mailserver. Zu beiden verbindet sich dein Computer, um eine Web-Seite oder eine Mail herunterzuladen und dann anzuzeigen.

So ähnlich funktioniert auch die Homebridge. Dein iPhone lädt von dort Daten herunter und zeigt sie in einer HomeKit-App an. Die heruntergeladenen Daten beinhalten den Status von Geräten, die über die Homebridge gesteuert werden, beispielsweise ob ein Gerät ein- oder ausgeschaltet ist oder welche Umdrehungszahl dein Ventilator hat. Das iPhone lädt auch Daten auf die Homebridge, nämlich Statusänderungen wie Gerät ausschalten, heller machen, schneller drehen, etc.

Die Aufgabe von Homebridge ist es, nicht HomeKit-fähige Geräte HomeKit-fähig zu machen. Das Programm ist also eine Art Übersetzungsmaschine. Programmiert wurde Homebridge von Nick Farina, der einfach nicht darauf warten wollte, dass Hersteller das HomeKit-Protokoll vielleicht irgendwann in ihre Geräte implementieren. Ich glaube nicht, dass er selbst jemals mit großem Erfolg gerechnet hat, aber da man die Homebridge mit so genannten Plug-Ins erweitern kann, passiert das nun nahezu ununterbrochen durch andere Programmierer. Und da sind schon so einige fähige Leute dabei. In nahezu allen Fällen erweist sich die Homebridge als sehr stabil und frustfrei.

Was ist diese Anleitung?

Diese Anleitung ist vor allem lang. Lass dich dadurch nicht entmutigen. Sie führt dich mit Abbildungen Schritt für Schritt von einem neuen Raspberry Pi, der ab sofort nur noch Raspi genannt wird, bis zu einer funktionstüchtigen Homebridge. Zusätzlich zeigt sie dir, wie du das Homebridge-Plugin *homebridge-config-ui-x* installierst. Es bietet dir eine Benutzeroberfläche im Browser, mit der du deine Homebridge verwalten kannst.

Du wirst mit dem *Terminal*-Programm deines Macs arbeiten müssen. Das findest du in deinem *Programme*-Ordner und dort im Unterordner *Dienstprogramme*. Im *Terminal* wirst du Befehle auf einer *Shell* eingeben. So wird die Oberfläche auf Linux-basierten Computern genannt, in dem du Befehle absetzen kannst.

In dieser Anleitung werden diese Befehle immer so dargestellt (Beispiel):

```
echo "Homebridge"
```

Du kannst den Befehl aus der Anleitung kopieren und im Terminal einsetzen. Danach drückst du die Entertaste. Versuch's mal.

Raspi vorbereiten

Installation des Betriebssystems Raspbian

Als erstes musst du das Betriebssystem herunterladen. Ich empfehle *Raspbian Stretch Lite*. Diese Version bietet keine grafische Benutzeroberfläche an. Der Raspi ist klein und es ist in meinen Augen unnötig, eine grafische Benutzeroberfläche in den Arbeitsspeicher zu laden, die dann am Ende niemand nutzt.

Du findest das Betriebssystem hier: https://www.raspberrypi.org/downloads/raspbian/

Lade es auf deinen Mac herunter. Du findest nach dem Download die Datei *2018-10-09-raspbian-stretch-lite.img* in deinem Download-Ordner. Der Dateiname kann ein anderer sein: je nachdem wann die letzte Version veröffentlicht wurde, ändert sich das Datum im Dateinamen.

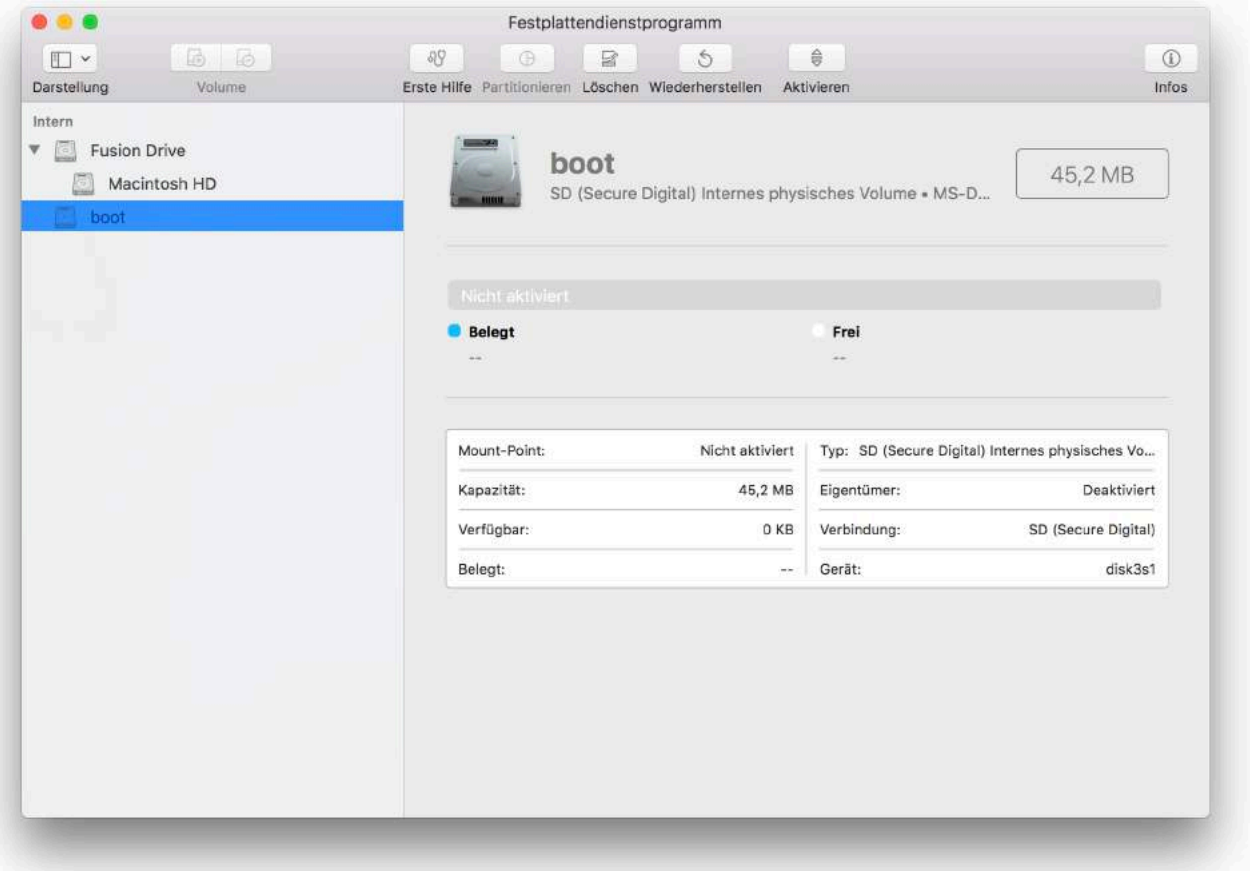
Um das Betriebssystem auf die SD-Card des Raspi zu brennen, benötigst du einen SD-Card-Slot in deinem Mac oder ein externes Gerät. Außerdem benötigst du eine Software, um die SD-Card zu flashen. Ich benutze dazu *Etcher*, das du hier bekommst: https://etcher.io.

Das Flashen ist sehr einfach und dauert keine fünf Minuten. Einfach die SD-Card in den Slot schieben, dann das Image *2018-10-09-raspbian-stretch-lite.img* aus dem Download-Ordner wählen, den richtigen SD-Card-Slot wählen (falls du mehrere haben solltest) und auf *Flash!* klicken. Der Rest geht dann völlig von allein.

Netzwerk-Konfiguration

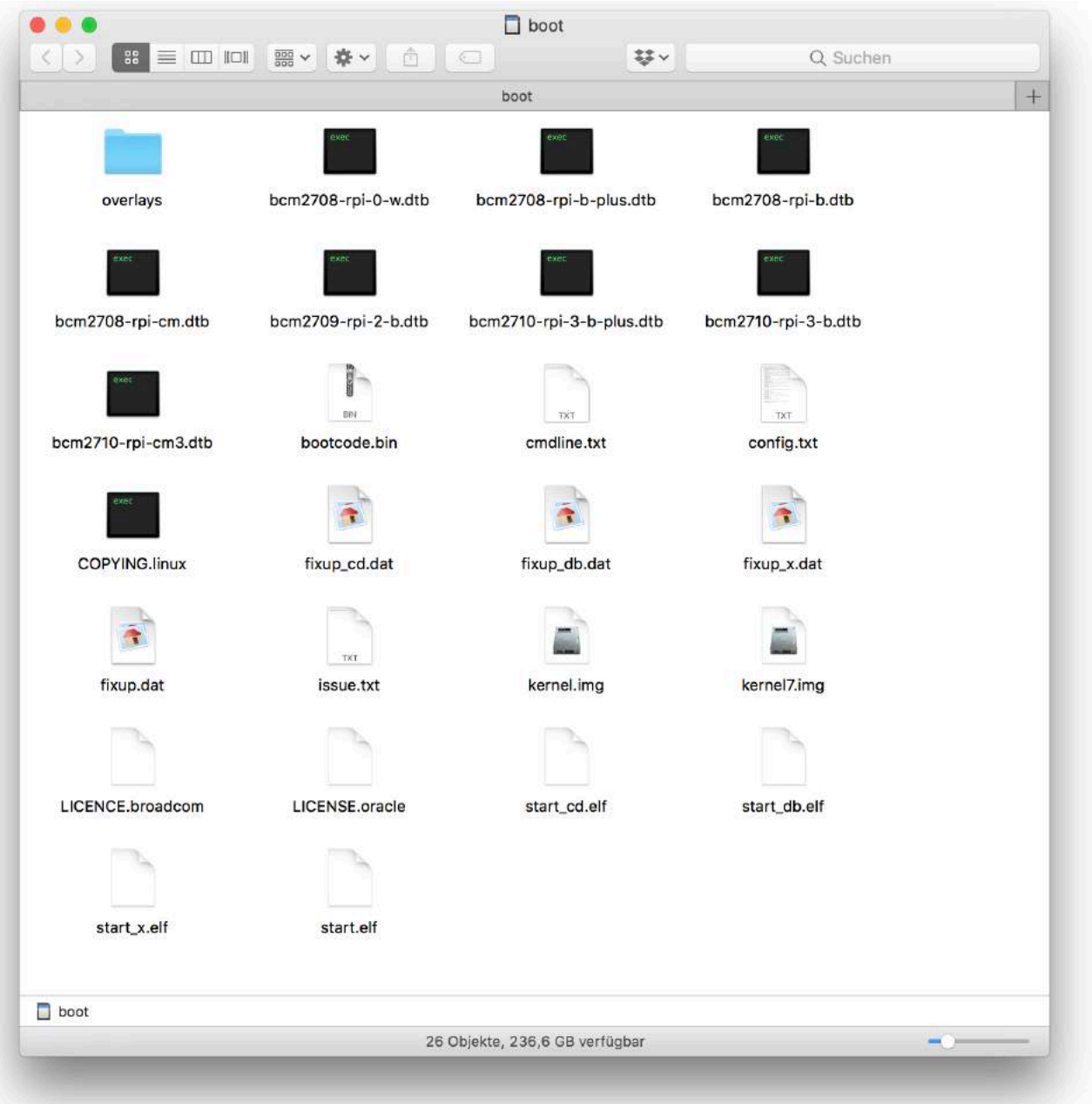
Um den Raspi zu konfigurieren, muss man normalerweise eine Tastatur und einen Monitor anschließen. Das sparen wir uns und machen eine so genannte *headless configuration*.

Möglicherweise ist die geflashte SD-Card mit dem Namen *boot* auf deinem Desktop aufgetaucht. Wenn nicht, musst du sie mit dem *Festplattendienstprogramm* aktivieren.



Wähle auf der linken Seite *boot* aus und klicke dann oben auf *Aktivieren*. Nun sollte auf deinem Desktop ein Icon mit dem Namen *boot* erscheinen.

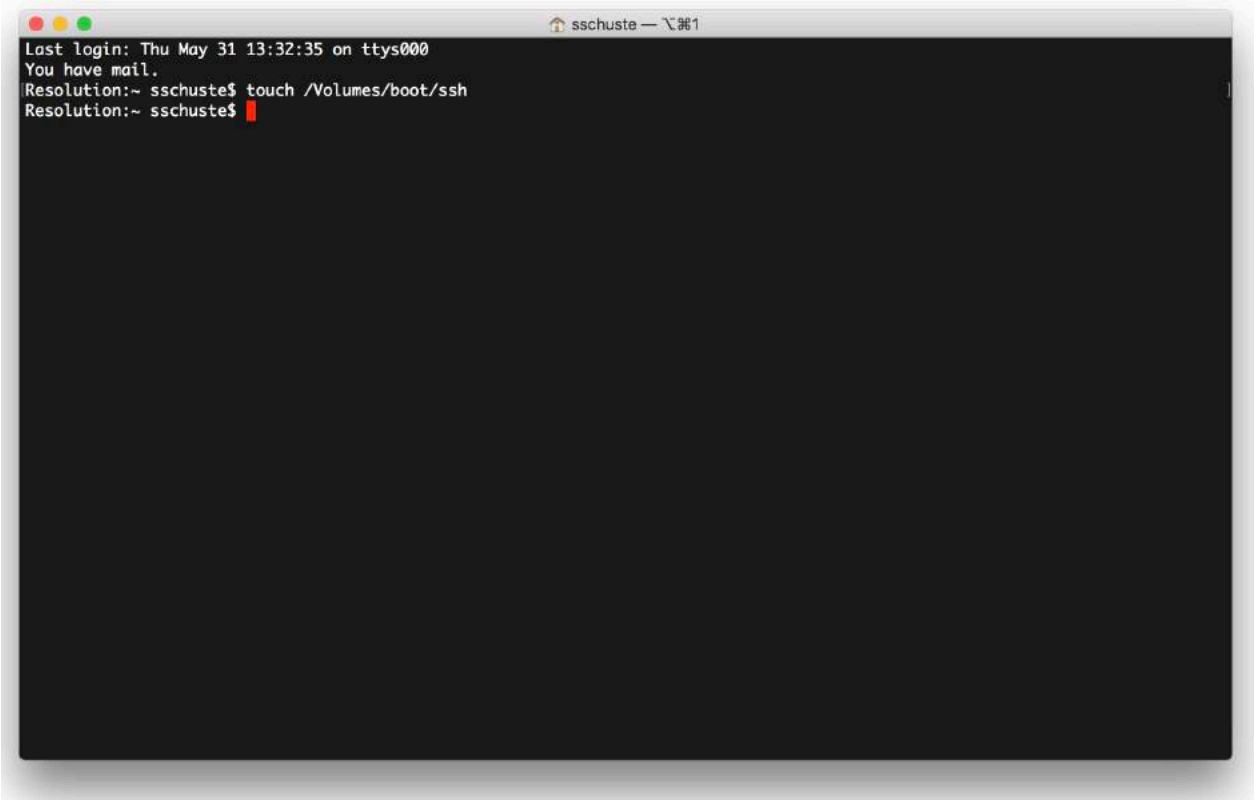
Wenn du da doppelt drauf klickst, öffnet sich ein Fenster mit diesem Inhalt:



Hier musst du nun zwei Dateien hinzufügen. Das geht am besten mit dem Terminalprogramm. Starte *Terminal*. Ein Fenster öffnet sich, in dem du Eingaben machen kannst. Dein Terminalprogramm präsentiert sich wahrscheinlich weiß mit schwarzer Schrift. In dieser Anleitung ist es genau anders herum. Spielt aber keine Rolle.

Als erstes legst du eine leere Datei mit dem Namen *ssh* an.

```
touch /Volumes/boot/ssh
```



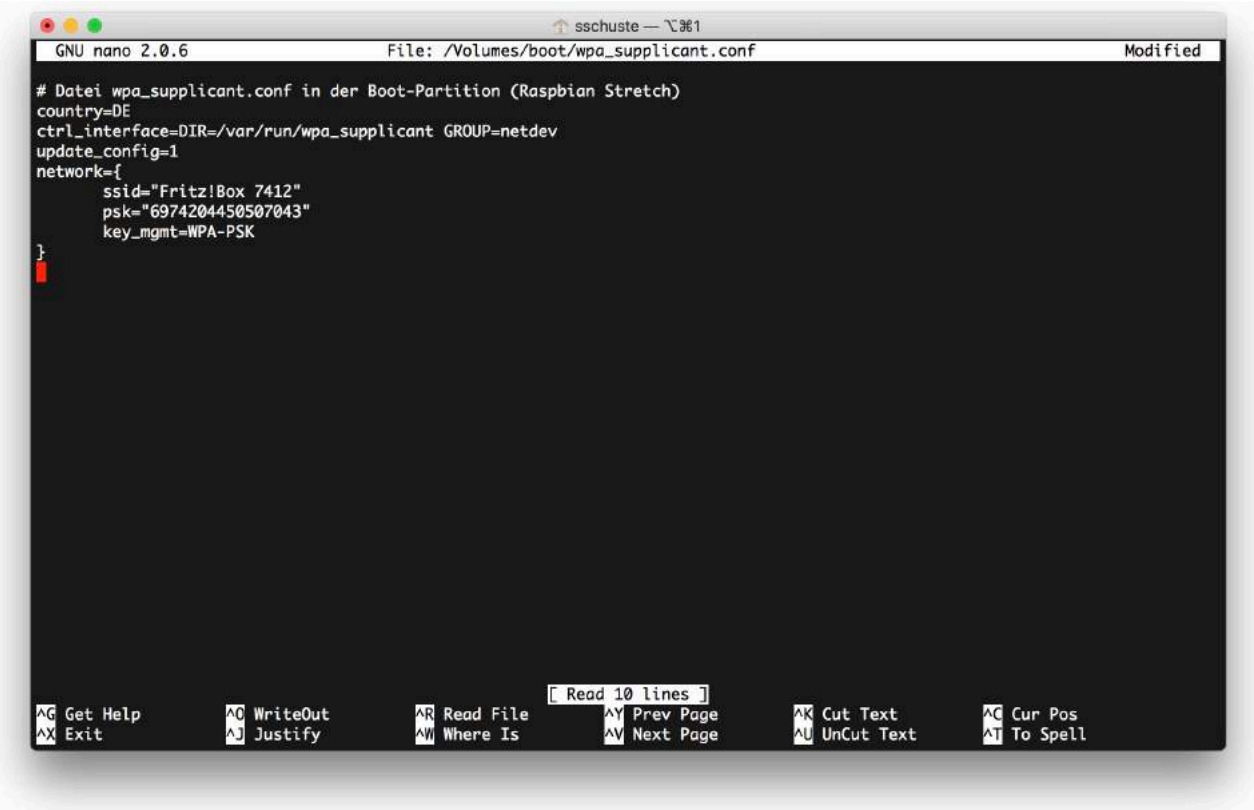
Die erste Datei ist damit angelegt. Nun zur zweiten, die ist schon etwas trickreicher. Diese Datei enthält Text, daher musst du nun im Terminal den Editor *nano* benutzen.

```
nano /Volumes/boot/wpa_supplicant.conf
```

Der Editor startet und zeigt außer seiner Menüleiste am unteren Ende nichts weiter an. Füge nun in den leeren Editor diese Zeilen ein:

```
# Datei wpa_supplicant.conf in der Boot-Partition (Raspbian Stretch)
country=DE
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
       ssid="MEINE_WLAN_SSID"
       psk="MEIN_WLAN_NETZWERKSCHLÜSSEL"
       key_mgmt=WPA-PSK
}
```

Achtung: du musst hier die Daten deines Netzwerkes eingeben. Hinter *ssid*= kommt der Name deines WLANs hin und hinter *psk*= das Passwort für dein WLAN. Etwa so:



Alles andere bleibt so, wie es ist.

Du speicherst die Datei ab, in dem du *ctrl-x* drückst, und danach *y* und dann die Entertaste.

Fertig. Wirf nun die SD-Card aus, indem du sie auf den Mülleimer ziehst (also genauso wie bei einem USB-Stick). Dann nimm sie aus dem SD-Card-Slot.

Raspi starten

Stecke die SD-Card in den Slot deines ausgeschalteten Raspi. Achte darauf, dass sie sich nicht verkantet. Übe keine Gewalt aus. Wenn sie eingelegt ist, schalte den Raspi ein.

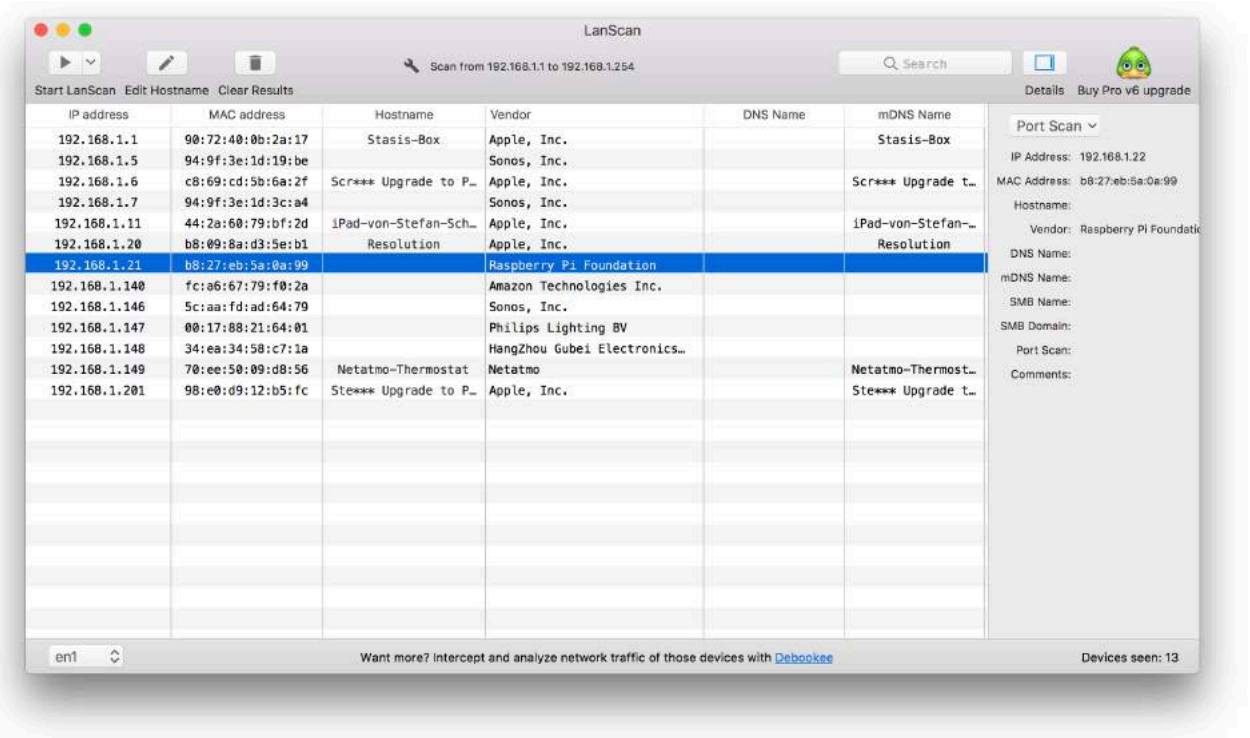
Beim Start versucht der Raspi nun, sich mit deinem WLAN zu verbinden. Falls du einen Monitor angeschlossen hast, kannst du ihm dabei zusehen. Es dauert etwa 60 Sekunden, bis der Raspi bereit ist.

Raspi im Netz suchen

Nachdem der Raspi gestartet ist, kann man ihn mit seiner IP-Adresse im Netzwerk erreichen. Aber welche IP-Adresse hat er erhalten? Besitzer einer Fritz!Box können die danach fragen, aber nicht jeder Router rückt die Information über angeschlossene Geräte heraus. Vor allem das Airport-Dienstprogramm ist in dieser Hinsicht zickig.

Man kann sich behelfen mit dem Programm *LanScan*, das du im MacAppStore findest. Lade es herunter, wenn du die IP-Adresse deines Raspi nicht finden kannst.

Nach dem Start von *LanScan* drückst du oben links auf den Pfeil und das Programm listet alle Geräte in deinem WLAN auf. Das sieht etwa so aus.



LanScan hat den Raspi gefunden - in der obigen Abbildung kann man das gut erkennen. Ganz links steht seine IP-Adresse. Die wird bei dir sicherlich eine andere sein.

Falls du den Raspi nicht finden kannst, dann ist etwas schief gelaufen beim Anlegen der zwei Dateien auf dem *boot*-Device. Dann musst du den Vorgang wiederholen: also Raspi ausschalten, SD-Card entnehmen, in den Mac einlegen und **die beiden Dateien *ssh* und *wpa_supplicant.conf* neu erzeugen**. Die wurden nämlich bei Booten des Raspi entfernt.

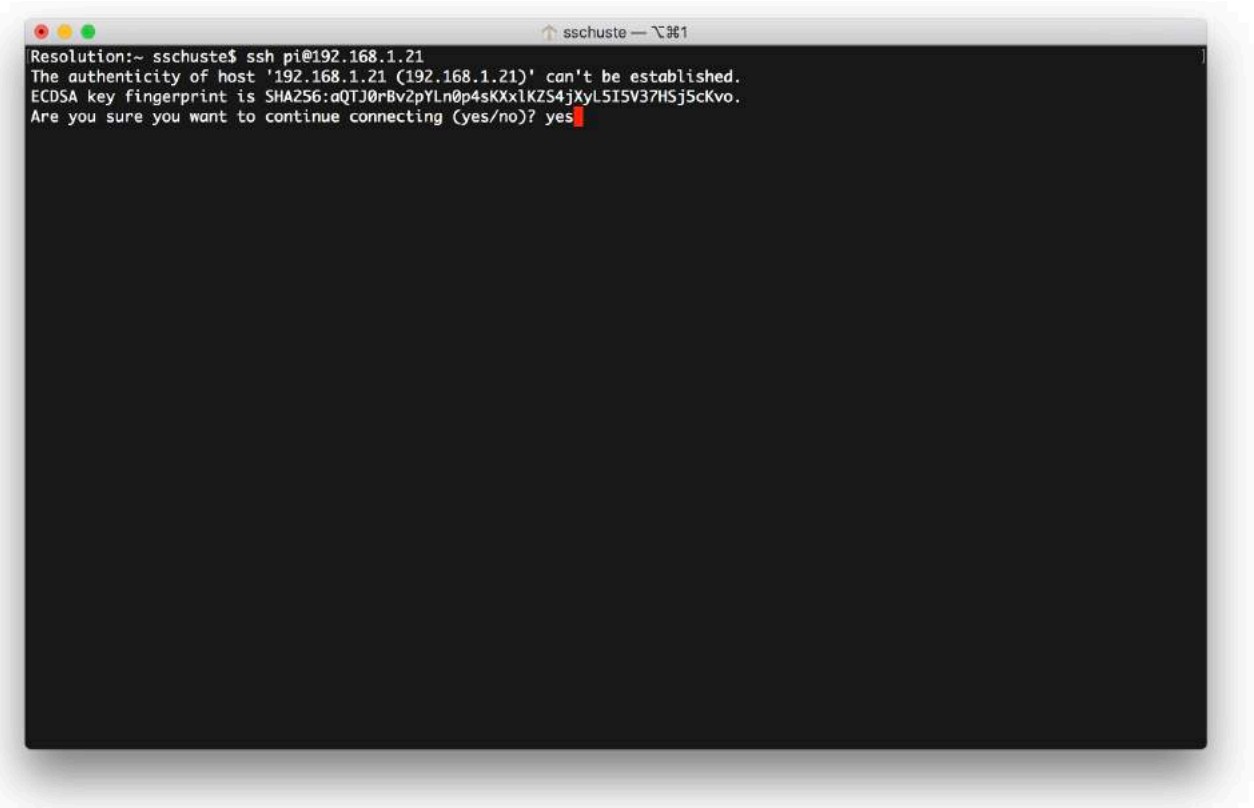
Die SD-Card mit *Etcher* neu flashen musst du nicht.

Erster Login

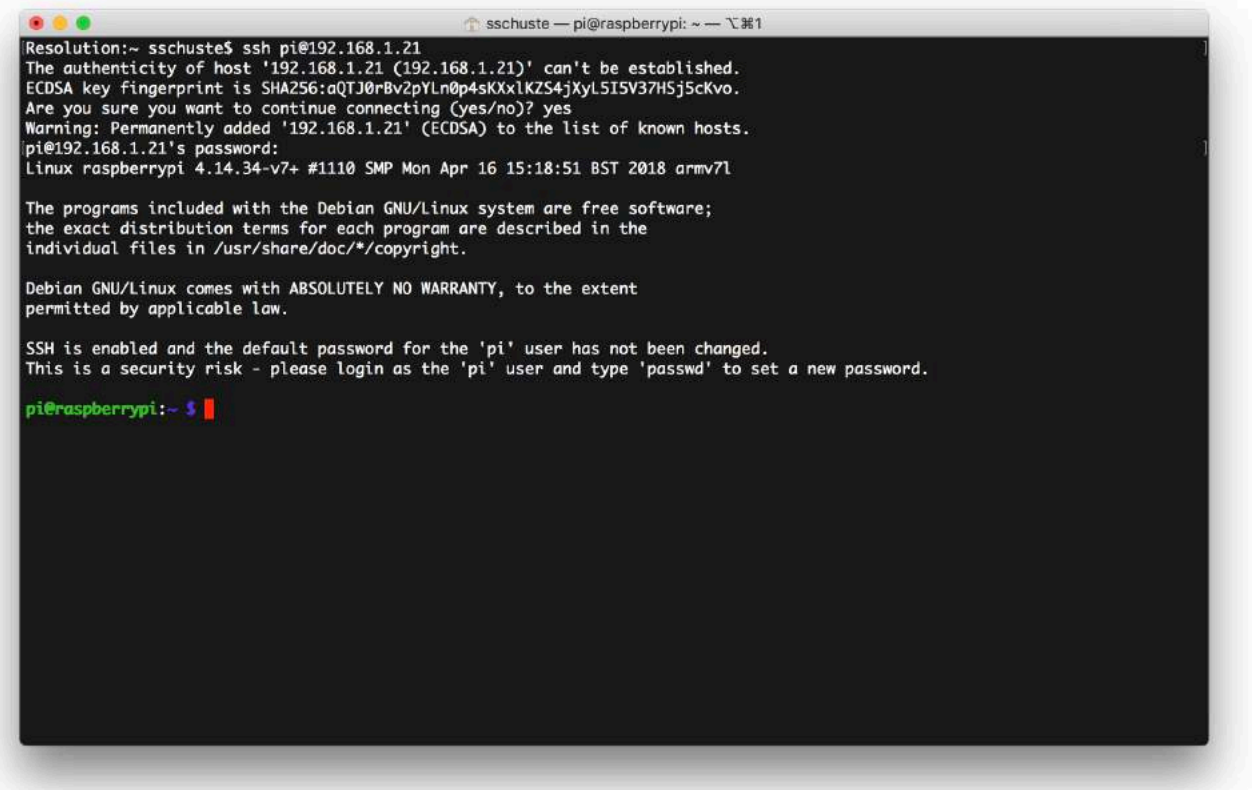
Jetzt, wo du die IP-Adresse des Raspis kennst, wird es Zeit für den ersten Login auf dem Raspi. Verbinden kannst du dich mit dem Programm *ssh*. Gib ein:

```
ssh pi@192.168.1.21
```

Natürlich musst du hier die IP-Adresse einsetzen, die du für deinen Raspi ermittelt hast. Nachdem du Return gedrückt hast, meldet sich der Raspi und will wissen, ob du dich wirklich verbinden willst. Tippe hier das Wort *yes* ein und drücke die Entertaste.



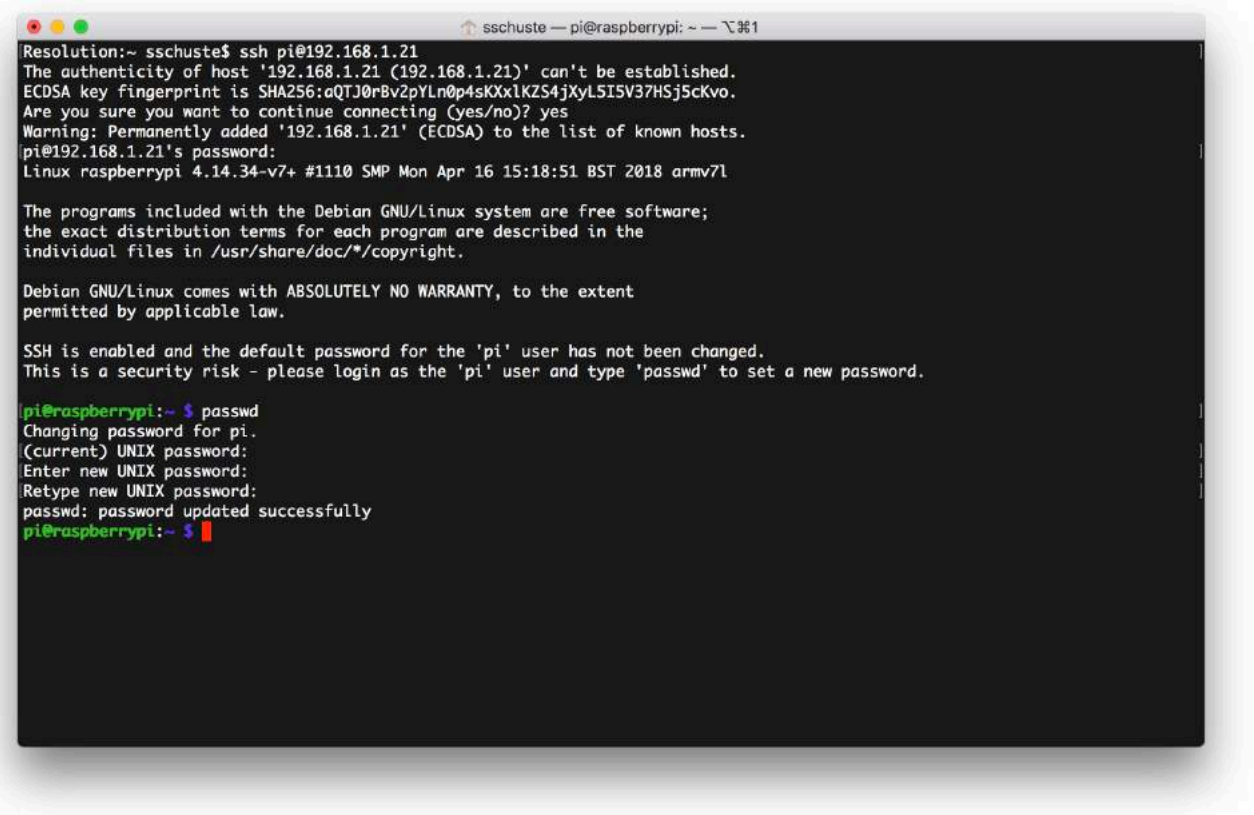
Danach will der Raspi das Passwort für den Benutzer *pi* wissen. Das Passwort lautet *raspberry*. Gib es ein und drücke die Entertaste. Dann bist du auf dem Raspi eingeloggt.



Viele Raspis haben das gleiche Passwort beim ersten Login. Das solltest du sofort ändern. Dazu rufst du das Programm *passwd* auf.

```
passwd
```

Passwd fragt dich erst nach dem aktuellen Passwort (*raspberry*). Gib es ein und drücke Return. Danach fordert dich *passwd* auf, ein neues Passwort zu vergeben. Gib eins ein und drücke Return. Du wirst aufgefordert, das neue Passwort noch einmal einzugeben. Tu das und drücke Return. Dabei wird keins der eingegebenen Passwörter auf dem Bildschirm angezeigt.

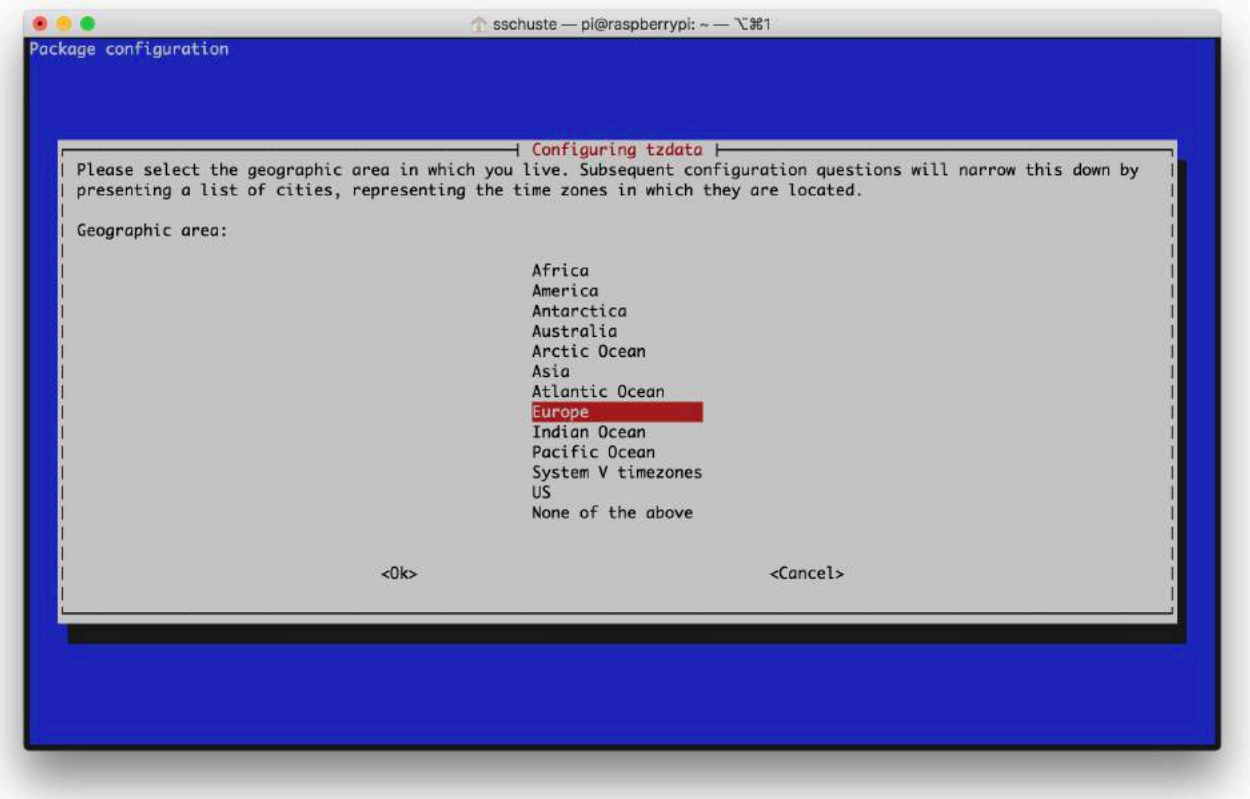


Der erste Schritt ist damit getan. Merk dir das Passwort, du wirst es gleich wieder brauchen.

Zeitzone einstellen

Damit die Uhr richtig geht, muss die Zeitzone eingestellt werden. Gib ein:

```
sudo dpkg-reconfigure tzdata
```



Benutze die Cursortasten, um in der Liste hoch und runter zu navigieren. Wenn du die richtige geographische Region gewählt hast, drücke die Entertaste. Danach wirst du aufgefordert, eine Stadt in deiner Nähe auszuwählen. Navigiere genauso wie eben. Nachdem du die Entertaste gedrückt hast, kehrst du zum System-Prompt zurück und bist fertig.

Raspbian updaten

Es wird Zeit, dem Raspi die neueste Software zu verpassen. Dazu gibst du ein:

```
sudo apt-get update
```

Nachdem das Update beendet ist, startest du das Upgrade:

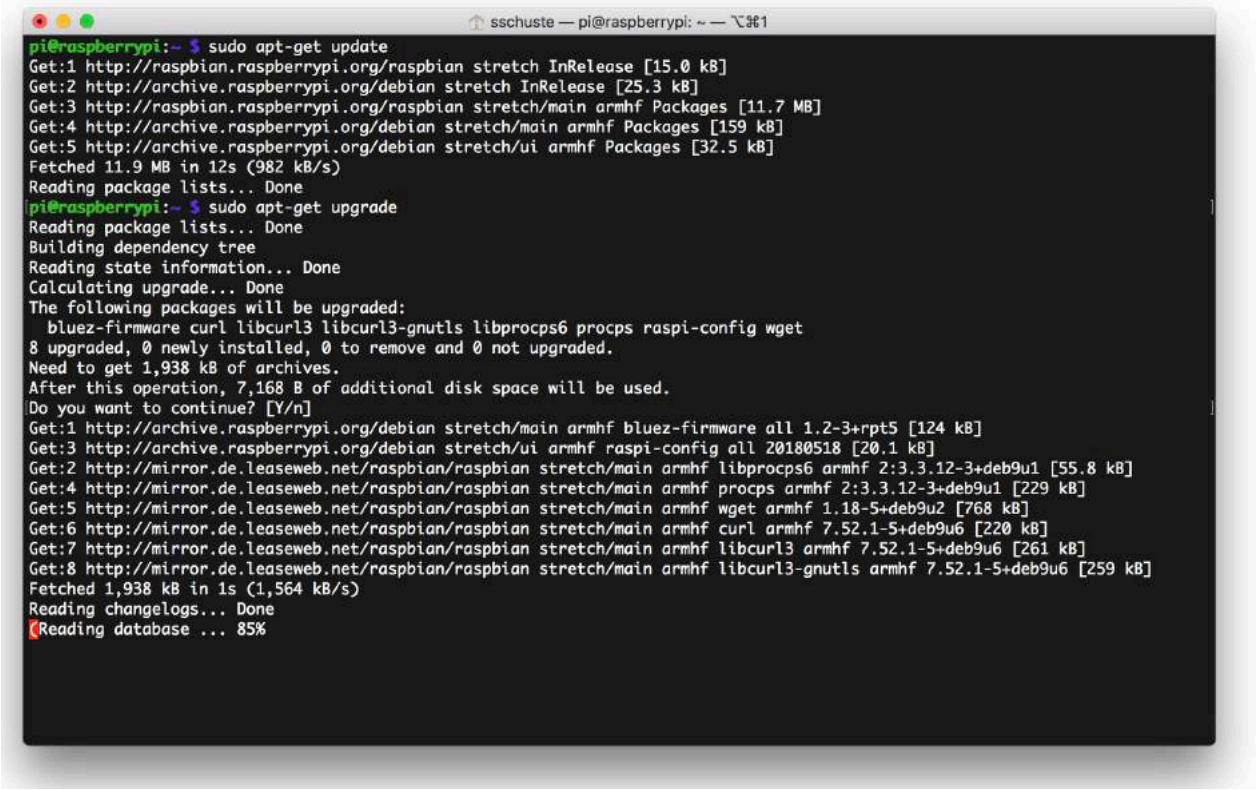
```
sudo apt-get upgrade
```

Die Frage

```
After this operation, 7,168 B of additional disk space will be used.
Do you want to continue? [Y/n]
```

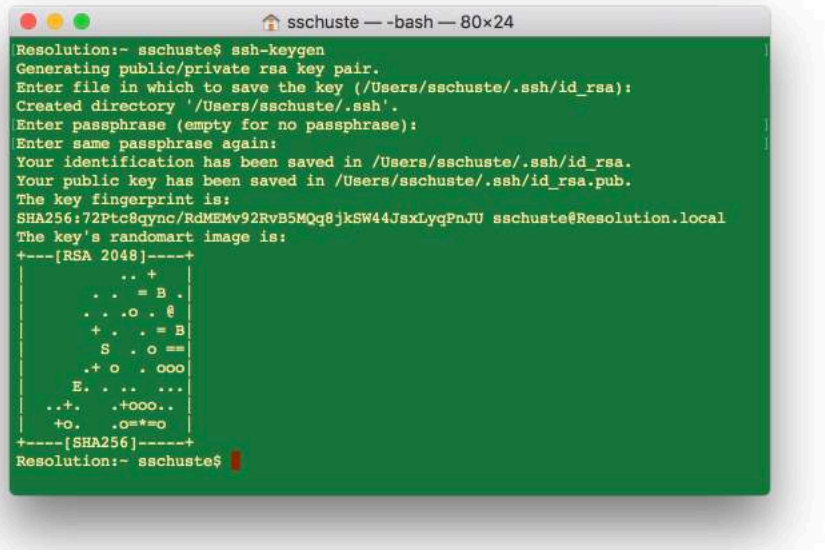
beantwortest du einfach, indem du die Entertaste drückst.

Damit ist der Raspi jetzt einige Minuten beschäftigt. Warte ab, bis er fertig ist.


sschuste — pi@raspberrypi: ~ — ⌥⌘1
pi@raspberrypi:~ $ sudo apt-get update
Get:1 http://raspbian.raspberrypi.org/raspbian stretch InRelease [15.0 kB]
Get:2 http://archive.raspberrypi.org/debian stretch InRelease [25.3 kB]
Get:3 http://raspbian.raspberrypi.org/raspbian stretch/main armhf Packages [11.7 MB]
Get:4 http://archive.raspberrypi.org/debian stretch/main armhf Packages [159 kB]
Get:5 http://archive.raspberrypi.org/debian stretch/ui armhf Packages [32.5 kB]
Fetched 11.9 MB in 12s (982 kB/s)
Reading package lists... Done
pi@raspberrypi:~ $ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
bluez-firmware curl libcurl3 libcurl3-gnutls libprocps6 procps raspi-config wget
8 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,938 kB of archives.
After this operation, 7,168 B of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 http://archive.raspberrypi.org/debian stretch/main armhf bluez-firmware all 1.2-3+rpt5 [124 kB]
Get:3 http://archive.raspberrypi.org/debian stretch/ui armhf raspi-config all 20180518 [20.1 kB]
Get:2 http://mirror.de.leaseweb.net/raspbian/raspbian stretch/main armhf libprocps6 armhf 2:3.3.12-3+deb9u1 [55.8 kB]
Get:4 http://mirror.de.leaseweb.net/raspbian/raspbian stretch/main armhf procps armhf 2:3.3.12-3+deb9u1 [229 kB]
Get:5 http://mirror.de.leaseweb.net/raspbian/raspbian stretch/main armhf wget armhf 1.18-5+deb9u2 [768 kB]
Get:6 http://mirror.de.leaseweb.net/raspbian/raspbian stretch/main armhf curl armhf 7.52.1-5+deb9u6 [220 kB]
Get:7 http://mirror.de.leaseweb.net/raspbian/raspbian stretch/main armhf libcurl3 armhf 7.52.1-5+deb9u6 [261 kB]
Get:8 http://mirror.de.leaseweb.net/raspbian/raspbian stretch/main armhf libcurl3-gnutls armhf 7.52.1-5+deb9u6 [259 kB]
Fetched 1,938 kB in 1s (1,564 kB/s)
Reading changelogs... Done
(Reading database ... 85%

SSH-Schlüsselerstellung und -installation

Dieser Schritt ist nicht unbedingt notwendig, um eine Homebridge zum Laufen zu bekommen. Er dient nur der zukünftigen Bequemlichkeit, weil du nie wieder dein Passwort eingeben musst, um dich auf dem Raspi einzuloggen.



Öffne ein zweites Terminalfenster (Menü *Shell*, Menüpunkt *Neues Fenster*, Unterpunkt *Grass*) Ein grünes Fenster öffnet sich. Dort gibst du ein:

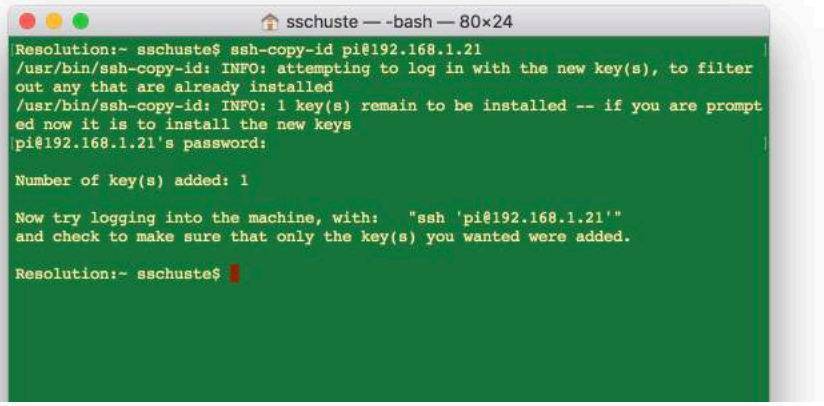
```
ssh-keygen
```

Ssh-keygen wird einige Fragen stellen, die du einfach mit der Entertaste beantwortest. Auch den Anfrage *Enter Passphrase* drückst du mit der Entertaste einfach weg.

Zwei Schlüssel wurden dabei erstellt. Den Inhalt des einen musst du auf den Raspi übertragen. Das geschieht mit dem Befehl *ssh-copy-id.*

```
ssh-copy-id pi@192.168.1.21
```

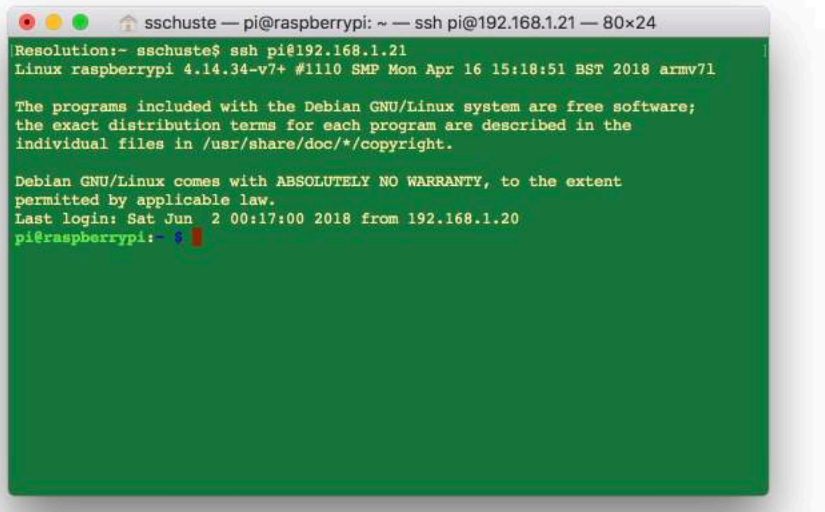
(natürlich musst du hier die IP-Adresse deines Raspis verwenden)



Du wirst bei dem Vorgang nach einem Passwort gefragt. Du gibst hier das neue Passwort für den Benutzer *pi* auf dem Raspi ein, jenes, das du vorhin mit *passwd* erstellt hast.

Das war's. Wenn du dich das nächste Mal von deinem Mac auf den Raspi einloggen willst, wirst du nicht mehr nach einem Passwort gefragt.

```
ssh pi@192.168.1.21
```



Fertig. Schließe das grüne Fenster.

Homebridge installieren

Nachdem der Raspi nun aufgesetzt ist, kannst du mit der Homebridge-Installation beginnen. Homebridge braucht einige Software, die bislang noch nicht auf dem System installiert ist. Bevor also Homebridge installiert werden kann, benötigst du noch die DNS-Software *Avahi* und das *node*-Framework.

Avahi installieren

So installierst du *Avahi*:

```
sudo apt-get install libavahi-compat-libdnssd-dev
```

Git installieren

Git wird für die reine Homebridge-Installation eigentlich nicht benötigt, aber irgendwann kommt mal der Zeitpunkt, wo du es brauchen wirst. Also installierst du es jetzt gleich mit:

```
sudo apt-get install git
```

Node installieren

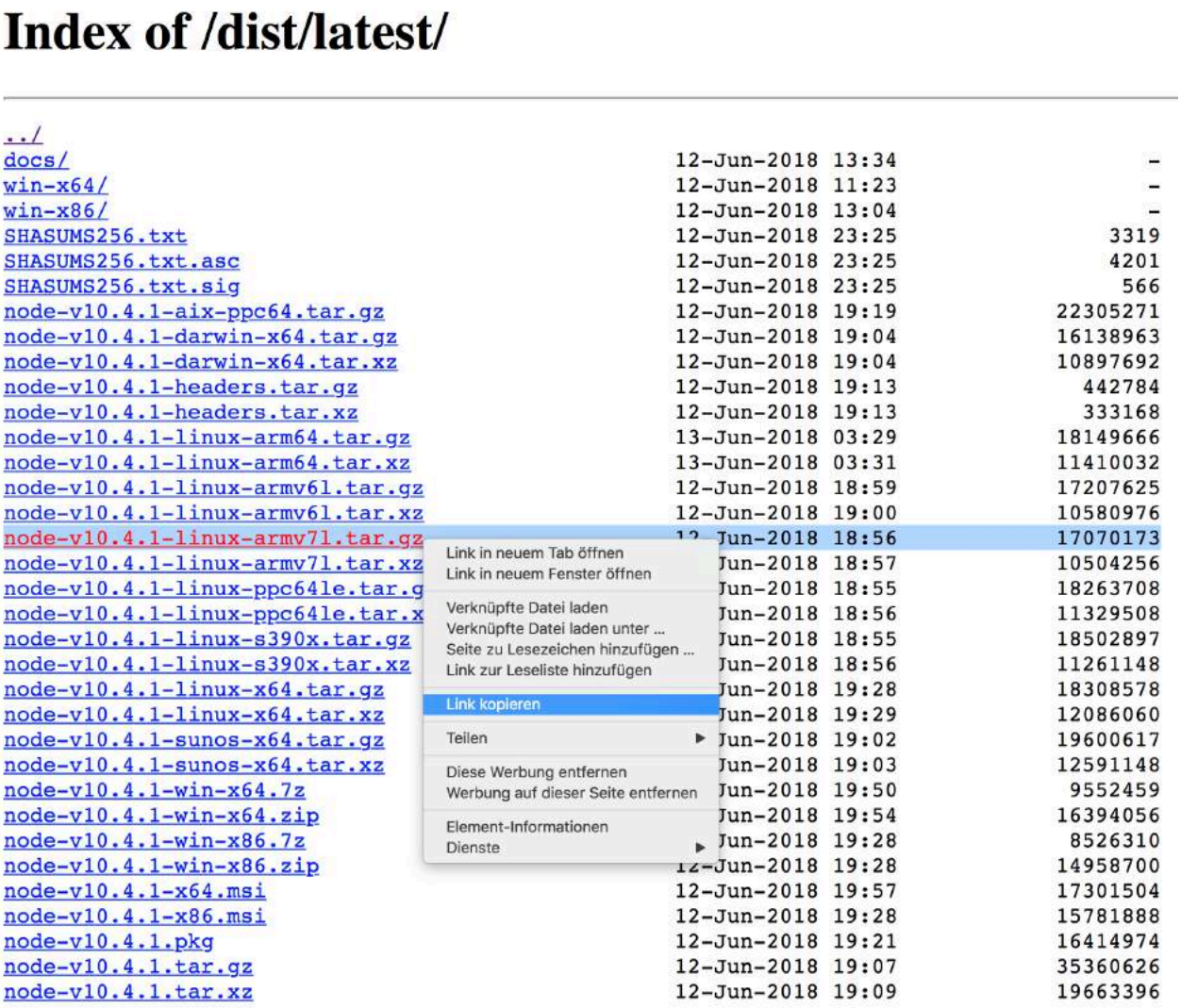
Node zu installieren ist ein wenig aufwändiger. Von *node* gibt es ständig neue Versionen. Daher enthält diese Anleitung keinen direkten Link zur neuestem Version, sondern du musst ein wenig nach ihr suchen. Ist aber nicht so schwer.

Rufe in deinem Webbrowser die Adresse https://nodejs.org/dist/latest/ auf. Eine Liste von Dateien erscheint. Da es *node* für viele Betriebssysteme und Prozessorarchitekturen gibt und die *node*-Macher auch noch für jedes Betriebssystem zwei verschiedene Paketformate gewählt haben, sieht die Liste etwas unübersichtlich aus.

Wenn du einen Raspi 3 hast, dann wähle den Dateinamen, der mit *-linux-armv7l.tar.gz* endet, mache einen Rechtsklick darauf und wähle dann *Link kopieren*. Die Vorgängermodelle haben eine andere Prozessorarchitektur und benötigen daher auch eine andere *node*-Version. Diese endet mit *-linux-armv6l.tar.gz*. Welche Prozessorarchitektur dein Raspi hat, kannst du mit dem Befehl

```
uname -m
```

anzeigen lassen und daraus deine Schlüsse ziehen, was du herunterladen musst. Falls du eine falsche Version erwischt, dann ist das zwar kein Beinbruch, aber *node* wird dann nicht funktionieren.



Achtung: die Anleitung zeigt jetzt die Installation der *node*-Version *node-v10.4.1* für die Prozessorarchitektur *armv7l*. Je nachdem, welche Version du verwendest, muss du die *10.4.1* im Dateinamen anpassen an die Versionsnummer, die du heruntergeladen hast, beispielsweise in 10.5.0 oder 11.2.1 oder was auch immer. Die Stellen, die du ändern musst, sind gelb markiert.

Herunterladen von node:

```
wget https://nodejs.org/dist/latest/node-v10.4.1-linux-armv7l.tar.gz
```

Hier kannst du mit Rechtsklick die oben kopierte Adresse einfügen. Das Kommando *wget* lädt vom Download-Server eine komprimierte Datei, was man an der Endung *.gz* erkennt.

Auspacken von node:

```
tar xf node-v10.4.1-linux-armv7l.tar.gz
```

Nachdem du die Entertaste gedrückt hast, erscheint der System-Prompt wieder. Auch wenn es so aussieht, als wäre nichts weiter passiert, ist tatsächlich die komprimierte Datei entpackt worden. Keine Systemmeldung ist eigentlich immer gleichbedeutend mit „gute Nachricht“.

Node an die richtige Stelle kopieren

```
sudo cp -R node-v10.4.1-linux-armv7l/* /usr/local/
```

Auch hier wird dir keine weitere Meldung ausgegeben. Gut so. Damit ist *node* installiert.

Homebridge-User einrichten

Linux-Administratoren richten gern für jeden Service einen eigenen Benutzer mit eingeschränkten Systemrechten ein. Da du jetzt auch ein Linux-Admin bist, wirst du das auch für Homebridge tun. Diese Anleitung verwendet den Benutzernamen *homebridge*, aber du kannst auch einen anderen wählen. Allerdings bietet es sich an, den gleichen Namen zu wählen, den auch der Service hat, dann weiß man später immer, wer wohin gehört.

```
sudo useradd -m -c "Homebridge Service" -s /bin/bash homebridge
```

Das System gibt keine Meldung aus, wenn der Vorgang korrekt abgeschlossen ist.

Der Benutzer *homebridge* braucht noch besondere Rechte, damit später das Plugin *homebridge-config-ui-*x seine Arbeit verrichten kann. Dazu musst du mit dem Editor *nano* eine Konfigurationsdatei erstellen, die bislang nicht existiert.

```
sudo nano /etc/sudoers.d/homebridge
```

Der Editor startet und zeigt außer seiner Menüleiste am unteren Ende nichts weiter an. Füge nun in den leeren Editor diese Zeile ein:

```
homebridge ALL=(root) SETENV:NOPASSWD: /usr/local/bin/npm, /bin/systemctl
restart homebridge, /bin/journalctl, /usr/local/bin/node
```

Achtung: das ist eine einzige Zeile!

Du speicherst die Datei ab, in dem du *ctrl-x* drückst, und danach *y* und dann die Entertaste.

Abschließend zurrst du noch die Dateirechte für die neu erstellte Datei fest, so dass nur *root* einen Blick hinein werfen kann:

```
sudo chmod 640 /etc/sudoers.d/homebridge
```

Ein kleiner Blick in die Zukunft: es kann passieren, dass du irgendwann Plugins installierst, die mehr Systemrechte benötigen, als es diese Anleitung zulässt. Das wird dann zu unschönen Effekten und Fehlermeldungen führen. In diesem Fall musst du die Liste der privilegierten Befehle (also hier: */usr/local/bin/npm, /bin/systemctl restart homebridge, /bin/journalctl, /usr/local/bin/node*) erweitern. Das kann schwierig sein, weil dir vielleicht nicht klar ist, welchen Systembefehlen man erweiterte Rechte geben muss, damit das Plugin läuft.

Um Frustrationen zu vermeiden, kann man die Rechte auch so setzen, dass der Benutzer *homebridge* alle Systembefehle mit erweiterten Rechten aufrufen kann. Das ist zwar im Sinne eines sicheren Systems kontraproduktiv, dafür aber der Bequemlichkeit sehr zuträglich. Ersetze die abgebildete Konfiguration in solchen Fällen durch:

```
homebridge ALL=(ALL) SETENV:NOPASSWD: ALL
```

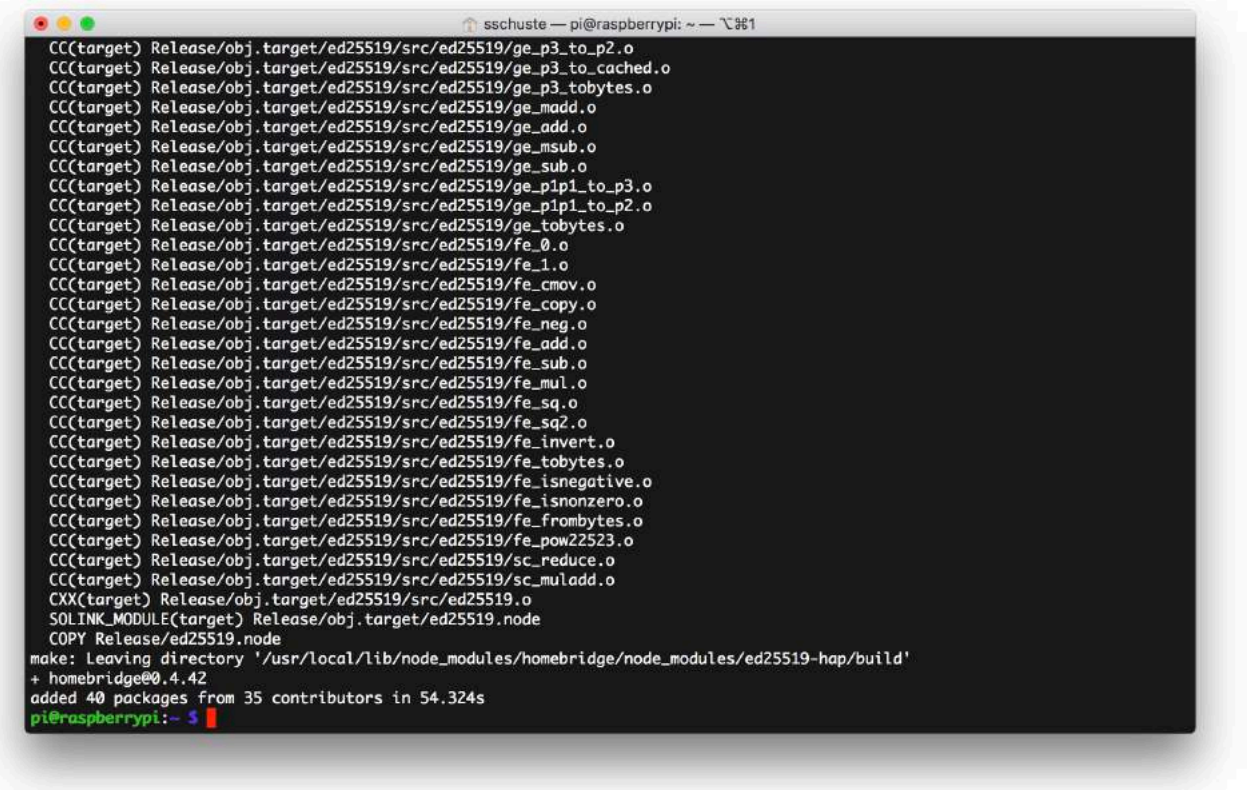
Am besten tust du das nur, wenn es sich nicht vermeiden lässt.

Homebridge installieren

Nachdem alle Vorbereitungen abgeschlossen sind, kannst du jetzt endlich mit der Homebridge-Installation beginnen. Das machst du mit:

```
sudo npm install -g --unsafe-perm homebridge
```

Die Installation erzeugt eine Menge Ausgaben auf dem Bildschirm, die ziemlich unleserlich und unverständlich sind. Keine Angst, du musst das auch nicht verstehen. Warte einfach solange, bis der System-Prompt wieder erscheint. Das wird nach ein paar Minuten der Fall sein.



Zum Abschluss der Installation richtest du noch das Homebridge-Verzeichnis ein, in dem Homebridge später seine Einstellungen speichert. Außerdem legst du die Homebridge-Konfigurationsdatei *config.json* an. Die wird dich zukünftig noch oft beschäftigen. Sie ist das Herz von Homebridge.

```
sudo mkdir -p /var/homebridge
```

Mit dem Editor *nano* legst du *config.json* an:

```
sudo nano /var/homebridge/config.json
```

Der Editor startet und zeigt außer seiner Menüleiste am unteren Ende nichts weiter an. Füge nun in den leeren Editor diese Zeilen ein:

```
{
      "bridge": {
            "name": "Homebridge",
            "username": "CC:22:3D:E3:CE:30",
            "port": 51826,
            "pin": "031-45-154"
      },
      "description": "Home Smart Home",
      "platforms": [],
      "accessories": []
}
```

Du speicherst die Datei ab, in dem du *ctrl-x* drückst, und danach *y* und dann die Entertaste.

All das soll dem Benutzer *homebridge* gehören:

```
sudo chown -R homebridge:homebridge /var/homebridge
```

Fertig. Deine Homebridge ist jetzt schon bereit zum Starten. Aber bei aller Ungeduld, das Ding endlich zum Laufen zu bekommen, ein letzter Schritt fehlt noch. Nicht du wirst in Zukunft die Homebridge starten, sondern das soll das Betriebssystem tun. Du hast garantiert noch nicht darüber nachgedacht, aber du willst bestimmt, dass die Homebridge bei jedem Boot des Raspis automatisch gestartet wird.

Systemd-Startskript erstellen

Um Homebridge zu kontrollieren, brauchst du einen Kontrollmechanismus. Kontrollieren heißt bei einem Service, ihn zu starten oder zu stoppen. Dabei hilft dir der Systemdienst *systemd*, der dir die Programme *systemctl* und *journalctl* zur Verfügung stellt.

Du musst nun zwei neue Dateien erstellen. Erstens:

```
sudo nano /etc/systemd/system/homebridge.service
```

Der Editor startet und zeigt außer seiner Menüleiste am unteren Ende nichts weiter an. Füge nun in den leeren Editor diese Zeilen ein:

```
[Unit]
Description=Node.js HomeKit Server
After=syslog.target network-online.target

[Service]
Type=simple
User=homebridge
EnvironmentFile=/etc/default/homebridge
ExecStart=/usr/local/bin/homebridge $HOMEBRIDGE_OPTS
Restart=on-failure
RestartSec=10
KillMode=process

[Install]
WantedBy=multi-user.target
```

Achtung: Beim Übertragen mit Kopieren & Einfügen kann es passieren, dass Zeilen verrutschen! Bitte überprüfe genau, ob die Konfiguration nach dem Einfügen genauso aussieht wie oben abgebildet und korrigiere sie gegebenenfalls.

Du speicherst die Datei ab, in dem du *ctrl-x* drückst, und danach *y* und dann die Entertaste. Danach erstellst du die zweite Datei:

```
sudo nano /etc/default/homebridge
```

Der Editor startet und zeigt außer seiner Menüleiste am unteren Ende nichts weiter an. Füge nun in den leeren Editor diese Zeilen ein:

```
# Defaults / Configuration options for homebridge
# The following settings tells homebridge where to find the config.json file
and where to persist the data (i.e. pairing and others)
HOMEBRIDGE_OPTS=-I -U /var/homebridge

# If you uncomment the following line, homebridge will log more
# You can display this via systemd's journalctl: journalctl -f -u homebridge
# DEBUG=*
```

Du speicherst die Datei ab, in dem du *ctrl-x* drückst, und danach *y* und dann die Entertaste. Das war's.

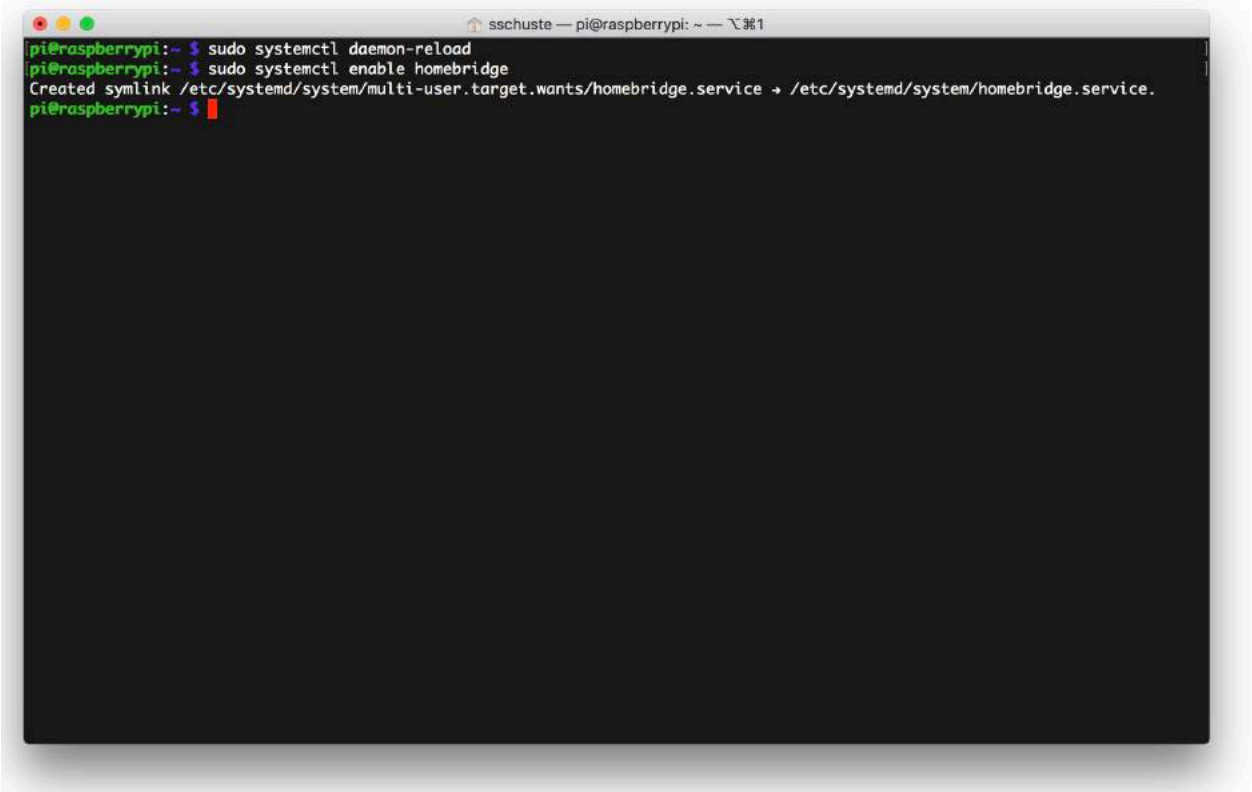
Systemd updaten

Um dem *systemd* deine neue Konfiguration bekannt zu machen, gibst du ein:

```
sudo systemctl daemon-reload
```

Da du willst, dass beim Reboot des Raspi die Homebridge automatisch gestartet wird, gibst du ein:

```
sudo systemctl enable homebridge
```



Homebridge starten

Jetzt kann es losgehen. So kontrollierst du die Homebridge:

`sudo systemctl start homebridge`	Startet die Homebridge ohne weitere Bildschirm-Ausgabe.
`sudo systemctl stop homebridge`	Stoppt die Homebridge ohne weitere Bildschirm-Ausgabe.
`sudo systemctl restart homebridge`	Stoppt erst die Homebridge und startet sie dann wieder ohne weitere Bildschirm-Ausgabe.
`sudo journalctl -fau homebridge`	Zeigt das Logfile an, in dem die Homebridge alle Tätigkeiten vermerkt. Das *f* in *-fau* sorgt für eine Live-Ausgabe. Lässt man es weg, so kann man das Log von Anfang an durchstöbern (seitenweise blättern: Leertaste; zeilenweise blättern: Entertaste; rückwärts blättern: b). Abbruch mit ctrl-c.

Man kann unter Linux zwei Kommandos durch ein Semikolon verbinden. Das ist ganz praktisch wenn man die Homebridge neu starten will und sofort das Log dazu sehen will.

```
sudo systemctl restart homebridge; sudo journalctl -fau homebridge
```

Wenn die Homebridge ordnungsgemäß gestartet ist, zeigt das Log folgendes an:

Bravo! Auf deinem Raspi läuft nun eine Homebridge!

Plugins installieren

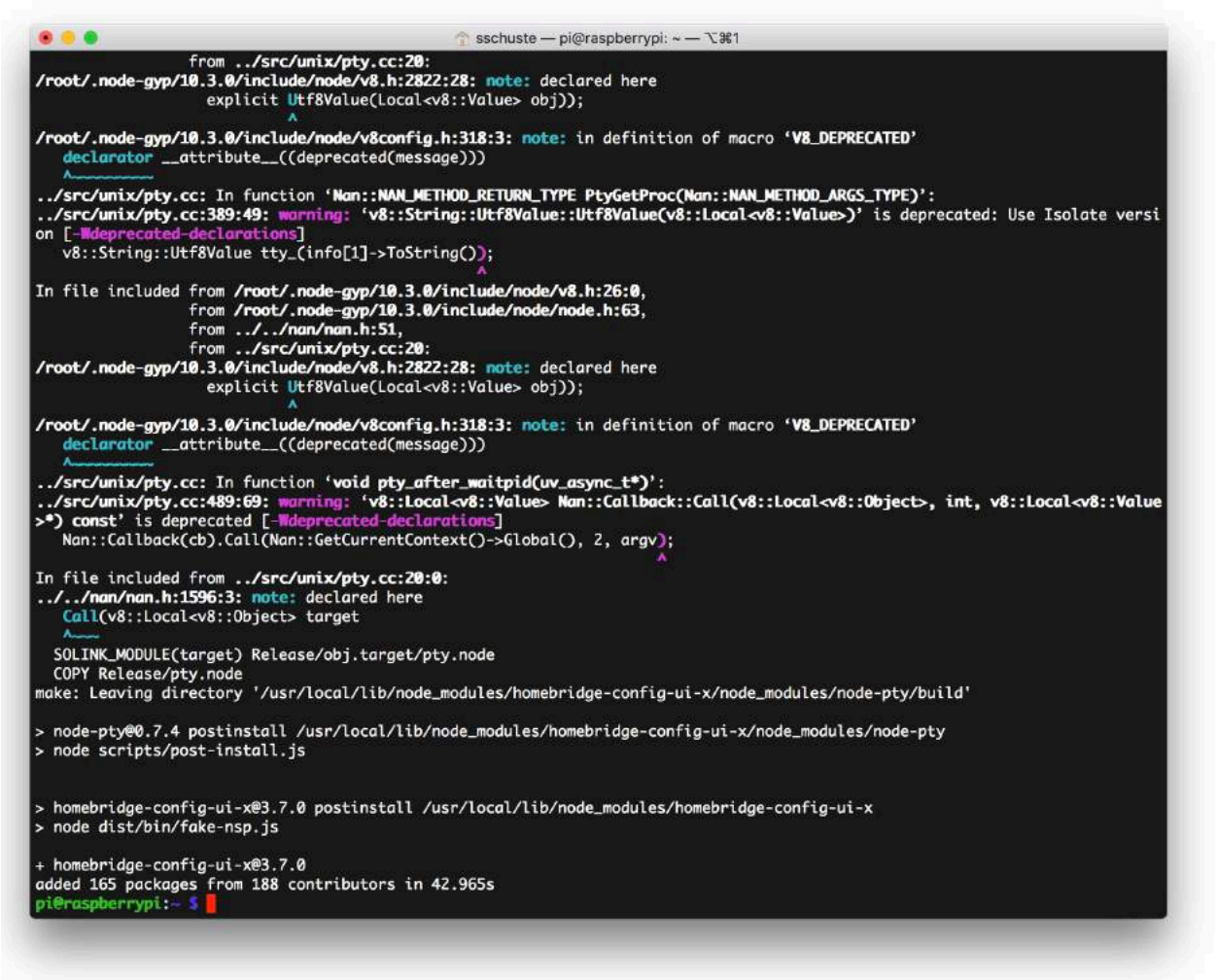
Das Plugin *homebridge-config-ui-x* installieren und konfigurieren

So. Jetzt wird es ernst. Mit dem nächsten Schritt kannst du deine laufende Homebridge zum Absturz bringen, und das mit den üblichen Folgen: langes Gesicht, Ärger, Frust, Zerstörung des Raspi an der nächsten Wand. Also aufpassen. Aber keine Angst: alles, was man kaputt macht, kann man auch reparieren.

Zunächst installierst du das Plugin. Dabei kann erstmal nichts passieren.

```
sudo npm install -g --unsafe-perm homebridge-config-ui-x
```

Der Befehl erzeugt eine Menge hässlicher Ausgaben auf dem Bildschirm. Ignoriere sie, wichtig sind die letzten zwei Zeilen:



Danach musst du Homebridge so konfigurieren, dass das Plugin geladen wird. Öffne mit dem Editor *nano* deine Homebridge-Konfiguration:

```
sudo nano /var/homebridge/config.json
```

Der Editor zeigt dir die Basis-Konfiguration an. Hier wird nun die Konfiguration von *homebridge-config-ui-x* eingefügt. Suche dazu die Zeile

```
      "platforms": [],
```

und füge zwischen die eckigen Klammern die folgenden Zeilen ein:

```
       {
            "platform": "config",
            "name": "Config",
            "port": 8080,
            "auth": "form",
            "theme": "red",
            "restart": "sudo -n systemctl restart homebridge",
            "temp": "/sys/class/thermal/thermal_zone0/temp",
            "sudo": true,
            "log": {
                "method": "systemd",
                "service": "homebridge"
            }
        }
```

Das Ergebnis sieht dann so aus:

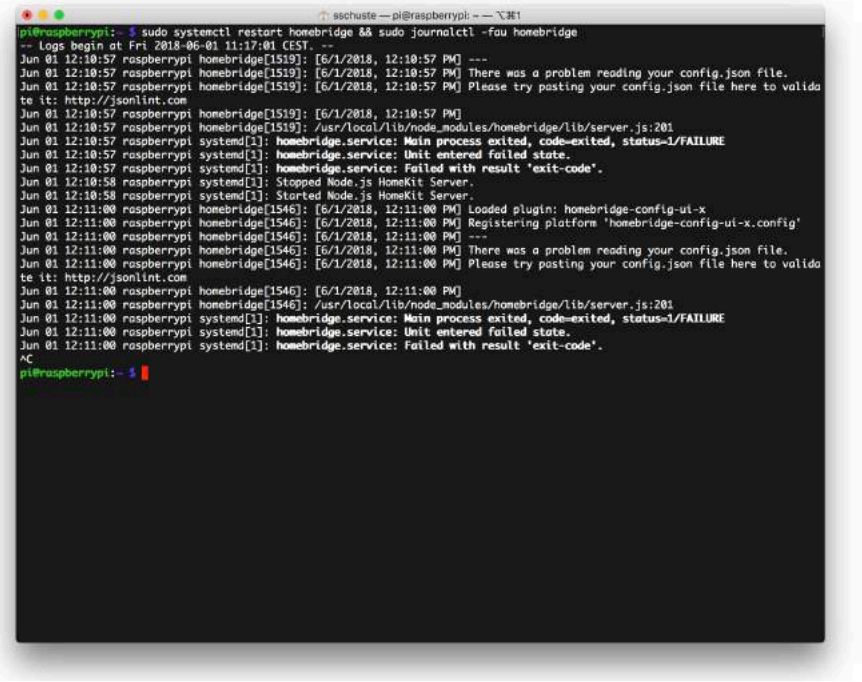
```
{
    "bridge": {
        "name": "Homebridge",
        "username": "CC:22:3D:E3:CE:31",
        "port": 51826,
        "pin": "031-45-154"
    },
    "description": "Home Smart Home",
    "platforms": [
        {
            "platform": "config",
            "name": "Config",
            "port": 8080,
            "auth": "form",
            "theme": "red",
            "restart": "sudo -n systemctl restart homebridge",
            "temp": "/sys/class/thermal/thermal_zone0/temp",
            "sudo": true,
            "log": {
                "method": "systemd",
                "service": "homebridge"
            }
        }
    ],
    "accessories": []
}
```

Du speicherst *config.json* ab, in dem du ctrl-x drückst, und danach y und dann die Entertaste.

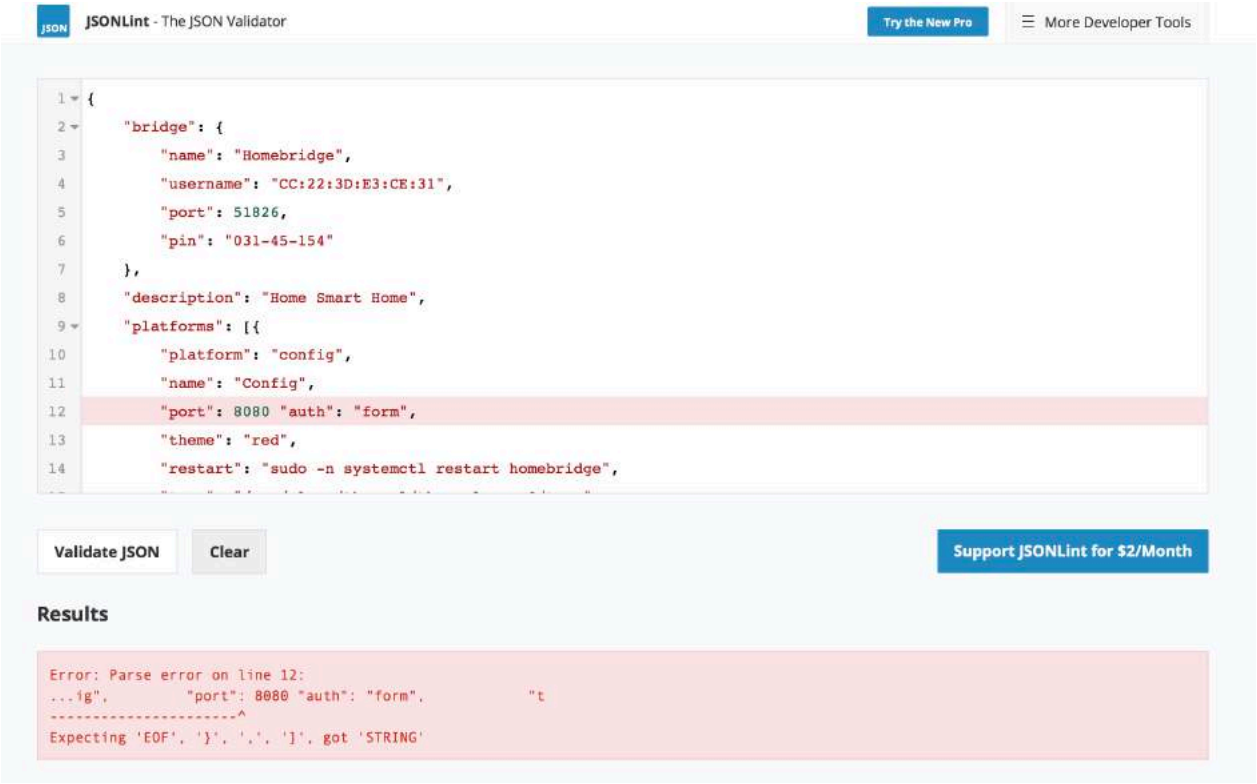
So. Frust oder Freude erfährst du, wenn du die **Homebridge neu startest** und einen Blick ins Log wirfst. So soll es aussehen:

Aber es kann halt auch so aussehen:

Traurig, aber das ewige Schicksal des Homebridgers: irgendwas in *config.json* ist nicht richtig formuliert. Anscheinend ging das Gefecht zwischen dir und den eckigen und geschweiften Klammern sowie mit den Kommas verloren. Der Systemdienst *systemd* versucht die Homebridge alle 10 Sekunden neu zu starten, aber dabei wird er solange scheitern, bis der Fehler behoben ist.

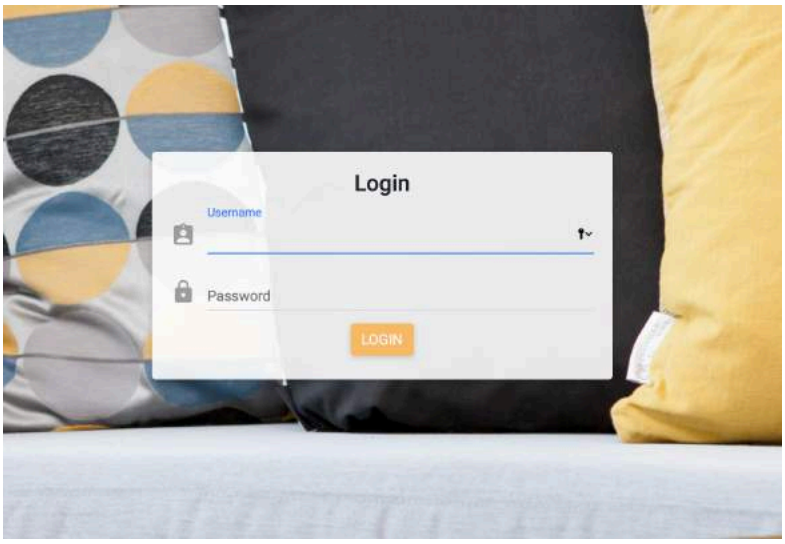


Du kannst deine *config.json* checken, indem du den Inhalt hier überprüfst: https://jsonlint.com. Füge deine Konfiguration in das Textfeld ein und klicke auf `Validate JSON`.



Oops! Hier fehlt wohl ein Komma zwischen *8080* und *„auth“*.

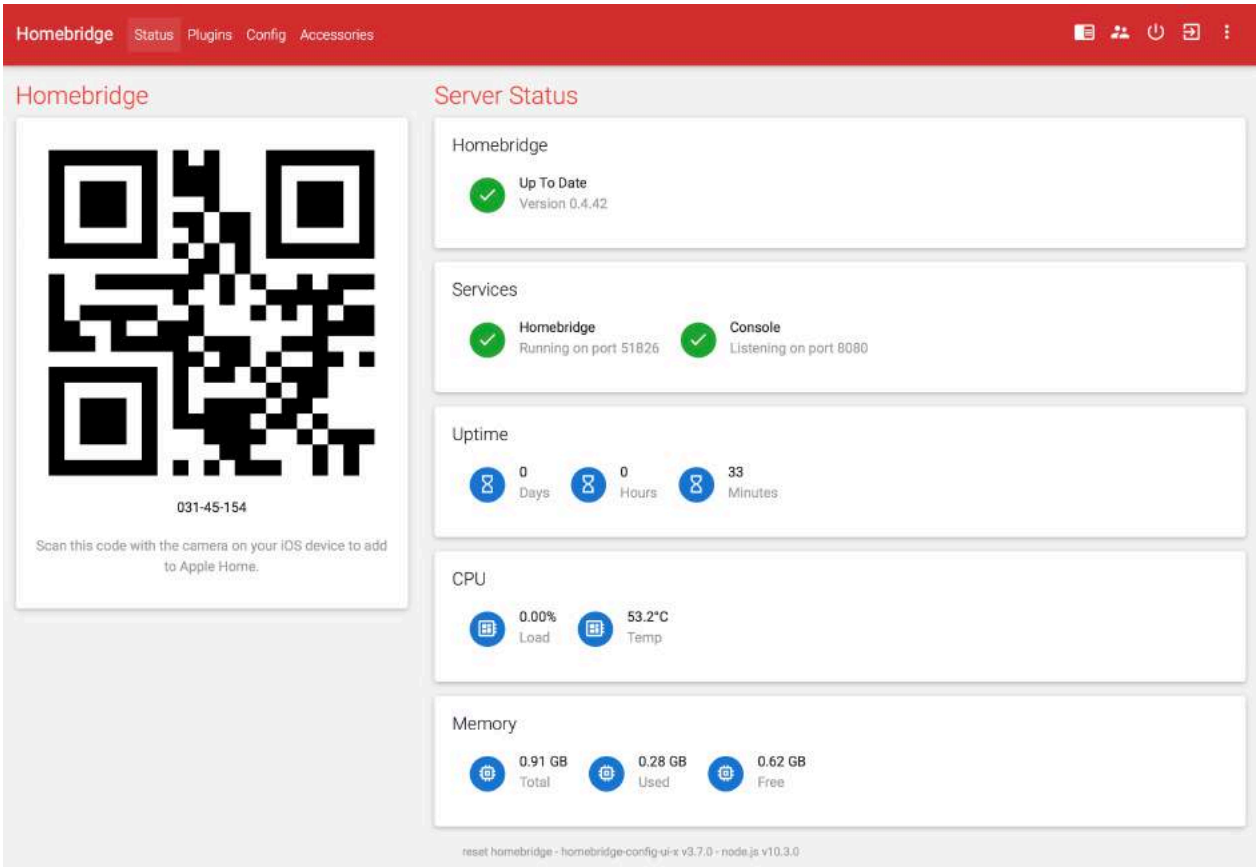
Wenn das Log gut aussieht, dann gib in der Adresszeile deines Webbrowsers ein: http://192.168.1.21:8080, wobei du statt 192.168.1.21 die IP-Adresse deines Raspi verwendest. Und voilà:



Logge dich ein mit dem Benutzernamen *admin* und dem Passwort *admin*.

Homebridge zu HomeKit hinzufügen

Nachdem du dich eingeloggt hast, siehst du diesen Bildschirm:



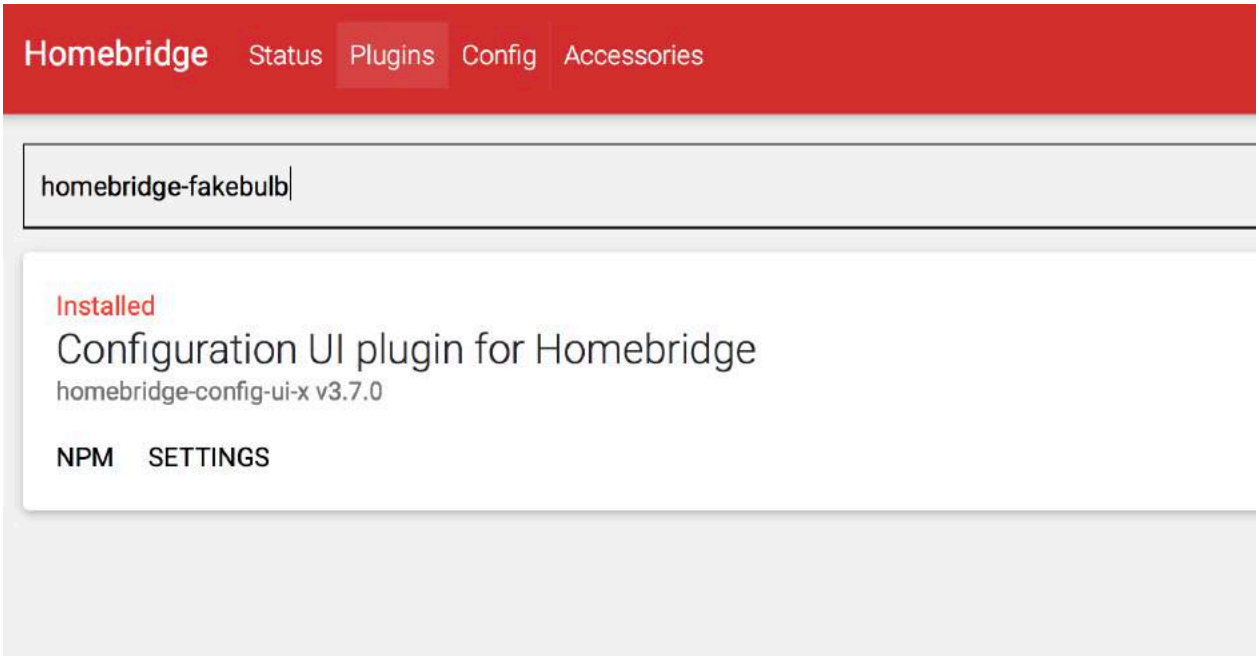
Nimm nun dein iPhone oder iPad und rufe die Home-App auf. Tippe oben rechts auf das *Pluszeichen* und danach auf *Gerät hinzufügen*. Scanne danach den QR-Code auf dem Bildschirm und folge den Anweisungen deines iPhones. Danach kennt HomeKit deine Homebridge und zeigt sie an. Aber noch gibt es keine Schalter, die man antippen könnte oder Auslöser, die irgendwas Bestimmtes tun.

Dazu braucht man weitere Homebridge-Plugins.

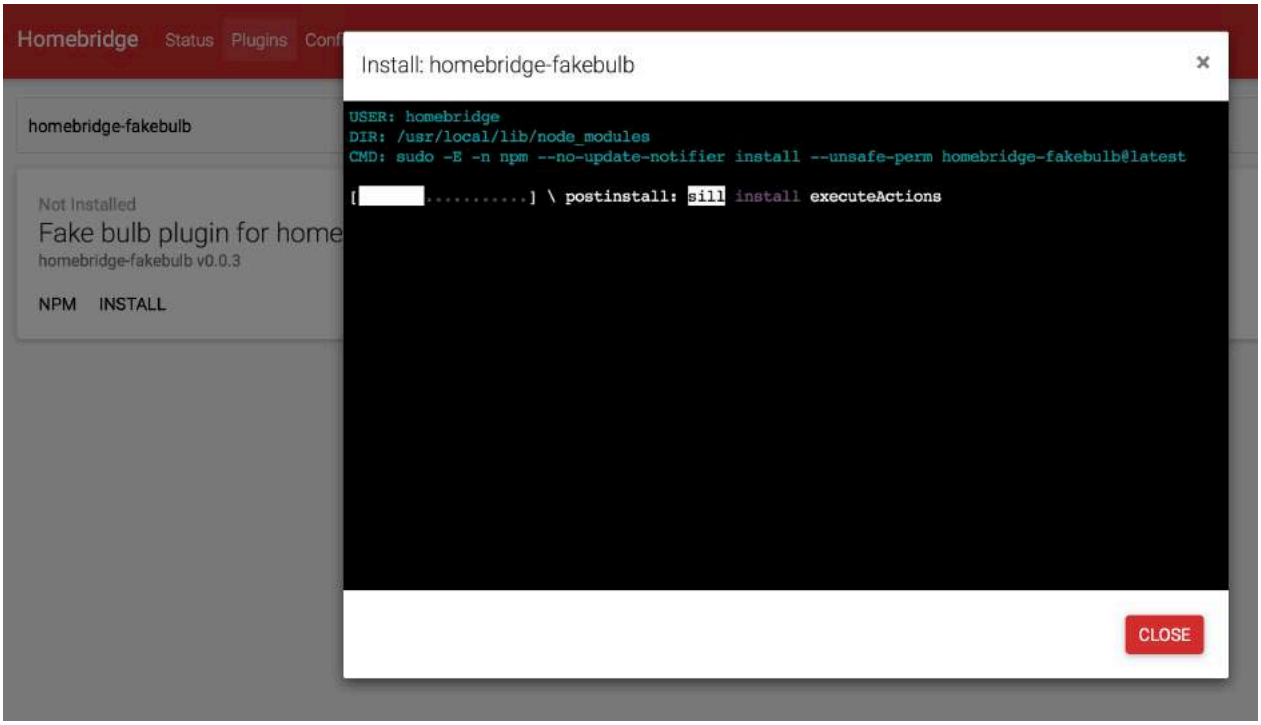
Ein Homebridge-Plugin mit *homebridge-config-ui-x* installieren und konfigurieren

Um ein Homebridge-Plugin zu installieren, klickst du in der Menüzeile von *homebridge-config-ui-x* auf den Menüpunkt *Plugins*. Die Plugin-Liste erscheint mit genau einem installierten Homebridge-Plugin, nämlich *homebridge-config-ui*-x selbst.

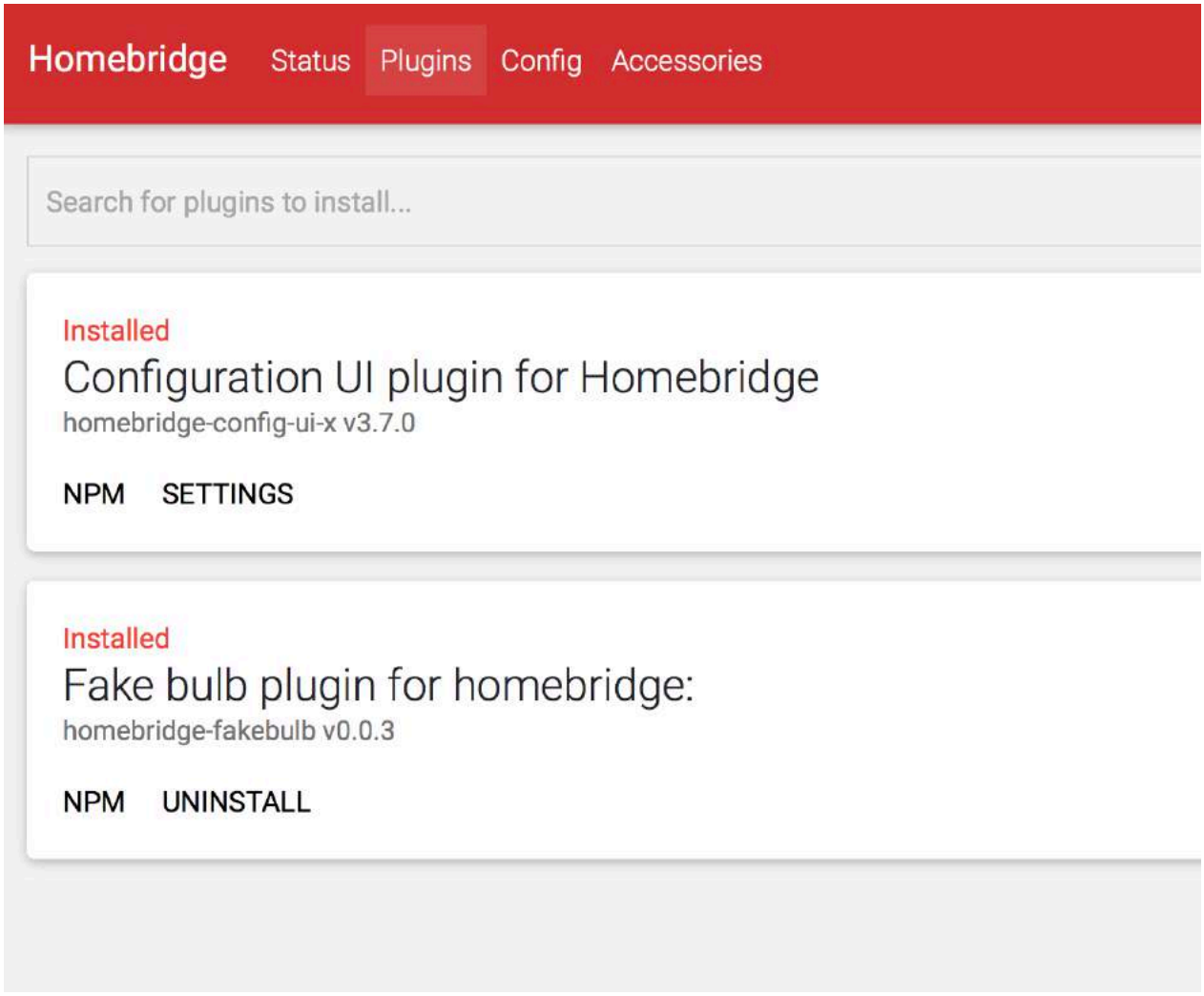
Tippe oben in der Suchzeile den Namen des Plugins, das du installieren willst. In dieser Anleitung ist das *homebridge-fakebulb* als Beispiel.



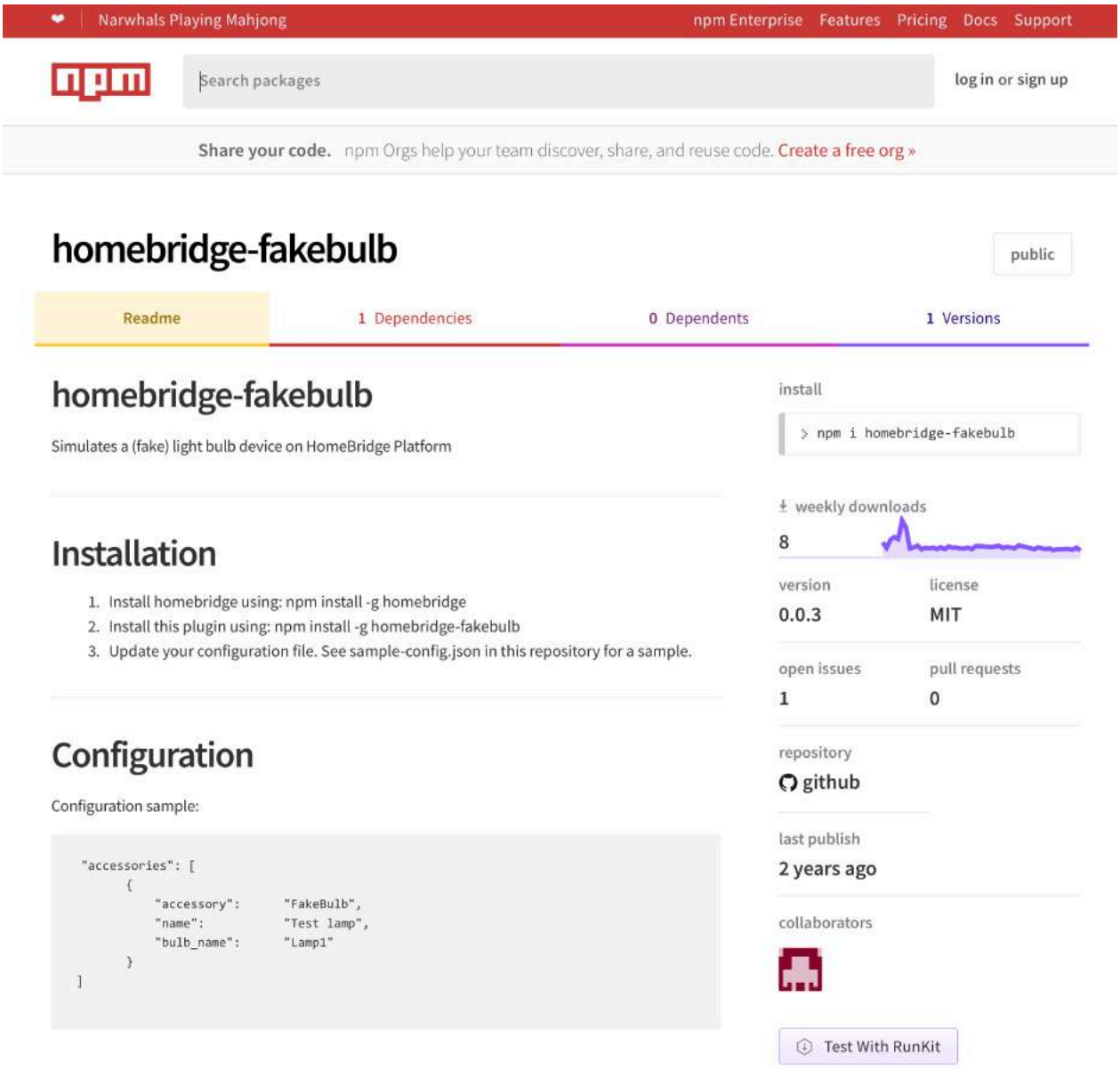
Nachdem dir *homebridge-fakebulb* angezeigt wird, klickst du auf *INSTALL*:



Warte ab, bis die Installation beendet ist. Die Pluginliste wird erneut aufgerufen und zeigt nun zwei Plugins.



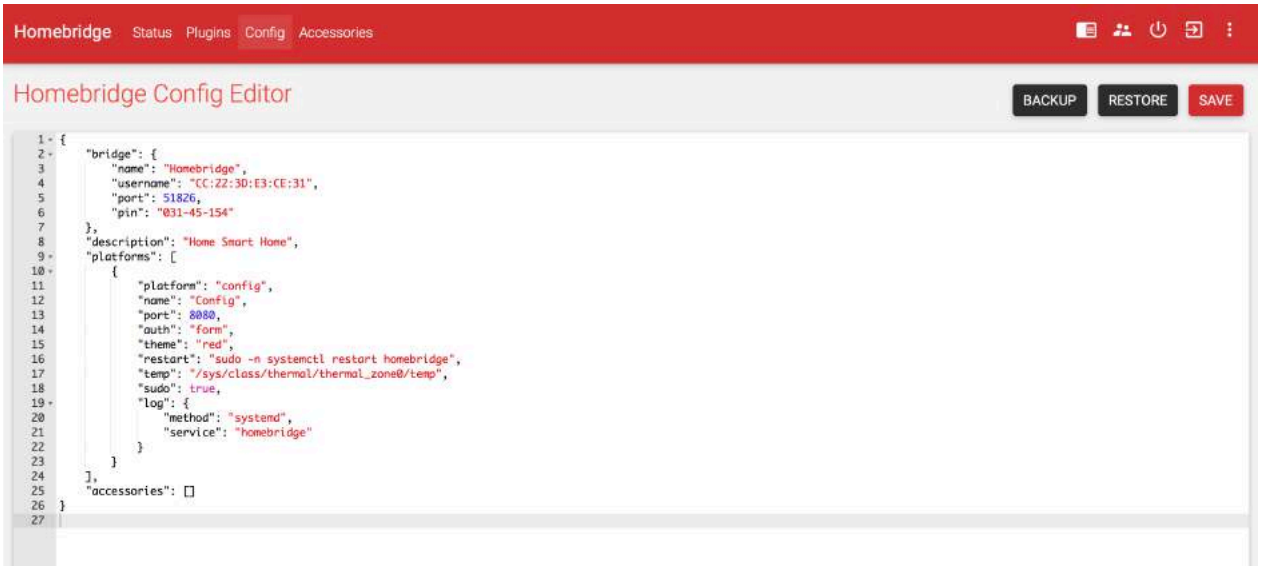
Um das Plugin *homebridge-fakebulb* einzubinden, muss *config.json* bearbeitet werden. Zunächst wirfst einen Blick auf die Anleitung zum Plugin, indem du auf *NPM* klickst. Die Seite zum Plugin öffnet sich in einem neuen Tab.



Die Installationsanleitung weist dich an, die Homebridge zu installieren (hast du schon), dann das Plugin *homebridge-fakebulb* (hast du auch schon) und schließlich *config.jso*n zu updaten. Das machst du, indem du im Browser den folgenden Teil der Konfiguration kopierst:

```
{
    "accessory": "FakeBulb",
    "name":      "Test lamp",
    "bulb_name": "Lamp1"
}
```

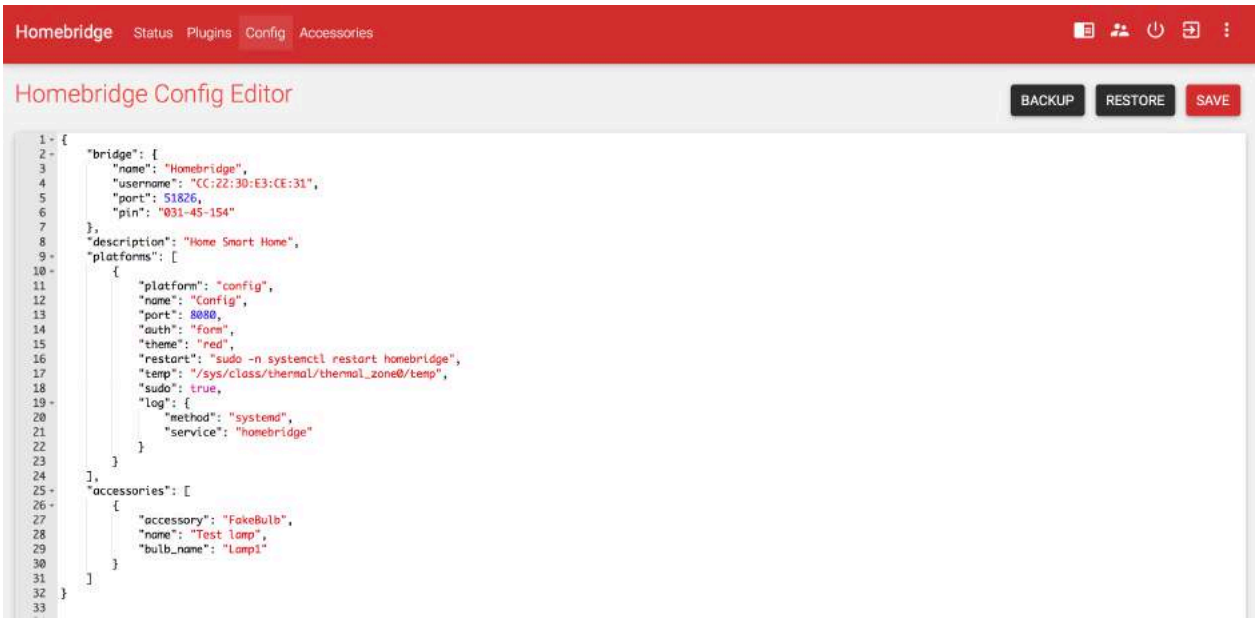
Dann wechselst im Browser du zurück zum Tab von *homebridge-config-ui-x* und klickst oben in der Menüleiste auf *Config*, um den Config-Editor aufzurufen.



Suche folgende Zeile:

```
"accessories": []
```

und platziere den Cursor zwischen die beiden eckigen Klammern. Dort fügst du jetzt die Konfiguration von *homebridge-fakebulb* ein.

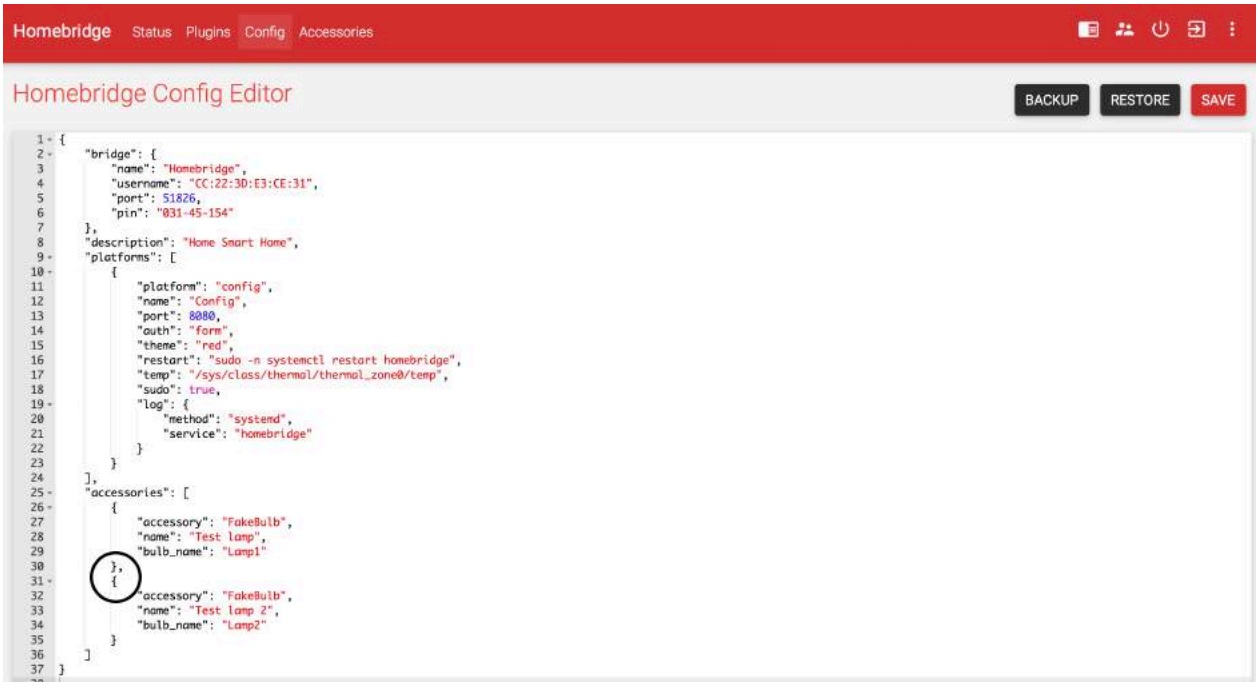


Dann klickst du oben auf *SAVE*. Deine Konfiguration wird erst abgespeichert, wenn alles korrekt ist. Befindet sich ein Fehler in *config.json*, kannst du deine Konfiguration nicht sichern.

Hat alles geklappt, dann starte die Homebridge neu, indem du rechts oben in der Menüleiste auf das Symbol klickst. Nachdem der Neustart beendet ist, taucht die *Lamp1* in der *Home*-App auf deinem iPhone auf.

Eine zweite Fakebulb konfigurieren

Öffne den Config-Editor: Markiere die Konfiguration der vorhandenen Fakebulb kopiere sie und füge sie darunter ein.



Achte auf das Komma zwischen der ersten und zweiten Fakebulb! Accessory-Definitionen werden immer durch ein Komma getrennt. Bitte beachte auch, dass die letzte Definition kein abschließendes Komma hat (hinter „Lamp2“ }). Das gilt genauso für Platform-Definitionen. Meist scheitert ein Speichern der Konfiguration genau daran.

Restarte die Homebridge, um die Änderung wirksam zu machen.

Plugins: Accessories und Plattforms

Ein Plugin kann entweder ein *accessory* sein oder eine *platform*. Je nachdem, was es ist, muss es in *config.json* an der richtigen Stelle eingerichtet werden.

Ein *accessory* ist ein Plugin, das genau eine eine Aktion eines Gerätes steuern kann. Die Fakebulb ist dafür ein gutes Beispiel. Man kann sie nur schalten. Sonst kann sie nichts. Wie ein normaler, nicht-smarter Lichtschalter an der Wand.

Es gibt aber auch Geräte, die mehr können als nur eine Schaltung. Beispielsweise kann ein Ventilator ein- und ausgeschaltet werden und zusätzlich kann man auch verschiedene Umdrehungsgeschwindigkeiten des Propellers einstellen. Oft kann sich der Ventilator auch noch um seine eigene Achse drehen. Für solche Geräte braucht man mehr als einen Schalter, und solche Steuerungen werden dann als *platform* programmiert.

Es kann dir aber letztendlich egal sein. Wichtig ist nur, dass du weißt, ob es sich um ein *accessory* oder eine *platform* handelt, denn davon hängt ab, wie du deine *config.json* bearbeitest.

Ob ein Plugin ein *accessory* ist oder eine *platform*, entnimmst du der Installationsanleitung des Plugins und dem JSON-Code, den du in config.json einsetzt. Bei der Fakebulb kann man deutlich sehen, dass das ein accessory ist. Es steht da nämlich gleich in der ersten Zeile:

```
{
    "accessory": "FakeBulb",
    "name":      "Test lamp",
    "bulb_name": "Lamp1"
}
```

Eine *platform* sieht nahezu genauso aus, aber diesmal findet sich der Begriff *platform* in der Konfiguration:

```
{
    "platform": "WeatherPlus",
    "name": "WeatherPlus",
    "service": "darksky",
    "key": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "displayName": "Wetterdaten",
    "location": [
        50.101384,
        8.650123
    ],
    "language": "de"
}
```

So, nachdem du jetzt weißt, welche Art von Plugin du da vor dir hast, musst du jetzt die Konfiguration an die richtige Stelle setzen. Ein *accessory* wird **immer** innerhalb der eckigen Klammern der *"accessories": [ ]* in *json.config* eingesetzt. Eine *platform* wird **immer** innerhalb der eckigen Klammern der *"platforms": [ ]* eingesetzt. Wenn du das mixt, wird deine Homebridge nicht funktionieren.

Um es mal ganz flach darzustellen: deine *config.json* besteht aus den Abschnitten *"bridge":{ }, "accessories": [ ]* und *"platforms":[ ]*. Etwa so:

```
"bridge":{"name": "Homebridge", "username": "CC:22:3D:E3:CE:30", .... },
"platforms": [{"platform": "Alexa"}, {"platform": "ZP"}, .... ],
"accessories": [{"accessory": "FakeBulb"}, {"accessory": "CMD"}, .... ]
```

Ich habe jetzt vieles weggelassen, aber es soll dir diese Abschnitte verdeutlichen. An der *"bridge":{ }*-Definition wirst du wahrscheinlich nie etwas ändern müssen. Aber aufgepasst mit den anderen beiden. Die Begriffe *"platforms":* und *"accessories":* tauchen nur je ein einziges in deiner config.json auf!

Klammern und Kommas

Meine *config.json* ist schon ziemlich lang und ich füge mal Teile davon in diese Anleitung ein. Da kann man ganz gut sehen, wie die Sache aufgebaut ist. Ich habe das Ganze farbig hinterlegt, um die Struktur deutlicher zu machen. In gelb sind jeweils Anfang und Ende der Abschnitte markiert.

```
{
    "bridge": {
        "name": "Homebridge",
        "username": "CC:22:3D:E3:CE:30",
        "port": 51826,
        "pin": "031-45-154"
    },
    "description": "Home Smart Home",
    "platforms": [
        {
            "platform": "Alexa",
            "name": "Alexa",
            "username": "el presidente",
            "password": "xxxxxxxxxxxxxxx"
        },

        {
            "platform": "ZP",
            "service": "switch",
            "brightness": true,
            "speakers": false
        },

        {
            "platform": "config",
            "name": "Config",
            "port": 8088,
            "auth": "form",
            "theme": "red",
            "restart": "sudo -n systemctl restart homebridge",
            "temp": "/sys/class/thermal/thermal_zone0/temp",
            "loginWallpaper": "/var/homebridge/wallpaper/ploum.jpg",
            "sudo": true,
            "log": {
                "method": "systemd",
                "service": "homebridge"
            }
        },

        {
            "platform": "WeatherPlus",
            "name": "WeatherPlus",
            "service": "darksky",
            "key": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
            "displayName": "Wetterdaten",
            "location": [
                50.101384,
                8.650123
            ],
            "language": "de"
        }
    ], <------ Ende von "platforms":

    "accessories": [
        {
            "accessory": "CMD",
            "name": "iMac shutdown",
            "on_cmd": "ssh 192.168.1.20 sudo -n ./shutdownmac.sh",
            "off_cmd": "/bin/true"
        },
```

```
        {
            "accessory": "CMD",
            "name": "LitePi shutdown",
            "on_cmd": "ssh 192.168.1.21 sudo /sbin/poweroff",
            "off_cmd": "/bin/true"
        },

        {
            "accessory": "CMD",
            "name": "Loop",
            "on_cmd": "/home/homebridge/bin/colorloop.sh on",
            "off_cmd": "/home/homebridge/bin/colorloop.sh off"
        },

        {
            "accessory": "DelaySwitch",
            "name": "Kino ein",
            "delay": 100,
            "disableSensor": true
        },

        {
            "accessory": "DelaySwitch",
            "name": "Kino aus",
            "delay": 100,
            "disableSensor": true
        }
    ] <------ Ende von "accessories":

}
```

Sieh dir das genau an. Die Begriffe "*platforms*": und "*accessories*": tauchen nur je ein einziges in auf und jeweils dahinter kommen dann die einzelnen Definitionen. Diese Definitionen werden mit einer geschweiften Klammer eröffnet und am Ende mit einer weiteren geschweiften Klammer geschlossen.

Jede dieser Definitionen wird mit einem Komma von der vorhergehenden getrennt.

```
        {
            "platform": "Alexa",
            "name": "Alexa",
            "username": "el presidente",
            "password": "xxxxxxxxxxxxxxx"
        }, <----- Achtung: ein Komma trennt diese beiden Plattformen!
        {
            "platform": "ZP",
            "service": "switch",
            "brightness": true,
            "speakers": false
        }, <----- Achtung: ein Komma trennt diese Plattform von der nächsten!
        {
```

Nur die allerletzte Definition hat kein Komma. Wenn du dort doch eins setzt, wird deine Homebridge nicht funktionieren.

```
      {
           "platform": "WeatherPlus",
           "name": "WeatherPlus",
           "service": "darksky",
           "key": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
           "displayName": "Wetterdaten",
           "location": [
               50.101384,
               8.650123
           ],
           "language": "de"
       } <----- Achtung: kein Komma mehr, das war die letzte Plattform!
   ],
   "accessories": [
      {
           "accessory": "CMD",
```

Exakt genauso wird das mit den *accessory*-Definitionen gemacht.

Achtung: Die Plugin-Installationsanleitungen geben die Abschnittsbezeichner "*accessories":* oder "*platforms":* immer mit an. Die kopierst du dann nicht mit! Schau dir dazu noch mal Seite 31 an, da kannst du sehen, dass die Installationsanleitung mehr enthält als du in deine *config.json* übernommen hast.

Ich weiß jetzt schon, dass du an den eckigen und geschweiften Klammern und den Kommas noch verzweifeln wirst. Das bleibt nicht aus, und jeder hat damit zu kämpfen. Wenn du strukturiert arbeitest, wird der Frust geringer sein, und wenn es gar nicht gehen will, hilft man dir im Forum. Nach einer Weile wirst du auf den ersten Blick erkennen, wo es klemmt.

Jetzt ist es an dir, deine Homebridge mit dem zu füllen, was du benötigst. Viel Spaß!