



Dokumentation zu Quiddi-github

Quiddi



Inhaltsverzeichnis

1	Git	5
1.0.1	neues Projekt	5
1.0.2	GitHub Upload	5
1.1	eigener Git Hub Server	6
2	Verschlüsseln von Passwörtern	7
2.1	GPG Schlüssel importieren exportieren erstellen	7
2.1.1	Export Keys	7
2.1.2	Import Keys	7
2.1.3	neues Key-Paar erstellen	8
2.2	pass	8
3	Microsoft Office 2013	9
3.1	Wine installieren	9
3.2	andere Pakete	9
3.3	Konfiguration Wine	10
4	DarkNet	11
5	VMware	13
5.0.1	Linux als Guest System	13
6	Debian	15
7	Bluetooth Lautsprecher	17

8	Dynamische IP updaten	19
9	Feste IP einstellen mit dhcpcd	21
10	SSH mit RSA-Keyfile	23
10.1	Keyfile am Client einspielen	23
11	ESP8266	25
12	HomeAutomation	29
12.1	SonOff S20 flash	29
12.2	SonOff T1 flash	31
12.3	SonOff RF Flash	33

1. Git

1.0.1 neues Projekt

In den Ordner im Terminal reinnavigieren und mit

```
git init
```

das Git-Projekt initialisieren. Danach:

```
git add *.*  
git add README  
git commit -m 'Projekt neu angelegt'
```

1.0.2 GitHub Upload

Danach kann auf der GitHub Homepage ein neues Projekt angelegt werden. Um dies pushen zu können muss man

```
git remote add origin https://github.com/quiddi-github/linux.bibel.git
```

origin adden. Danach kann man das Projekt auf origin pushen, mit:

```
git push -u origin master
```

Heirmit wird es auf dem Master-Branch gepushed. Zuvor müssen aber noch alle Dateien reingeladen werden mit:

```
git add .
```

und ein Kommentar angelegt werden mit

```
git commit -m 'Projekt neu angelegt'
```

Die remote Reposities können mit

```
git remote show
```

bzw. wenn ein remote Repository namens origin vorliegt, dies mit

```
git remote show origin
```

anzeigen lassen.

1.1 eigener Git Hub Server

Git sollte installiert sein. Der Rest läuft dann wie lokal. Am Server sollte nun folgendes passieren:

1. ordner in /home/git/usb_drive als xxx.git erstellen

2. git –bare init ausführen im ordner

Am Client nun folgendes machen:

- Neuen Ordner anlegen \$ cd myproject
- Initialisieren \$ git init
- Dateien hinzufügen mit \$ git add .
- Kommentar zu den Änderungen schreiben \$ git commit -m 'initial commit'
- Server nur einmal konfigurieren \$ git remote add origin git@gitserver:/opt/git/project.git
- Git push ausführen \$ git push origin master

2. Verschlüsseln von Passwörtern

2.1 GPG Schlüssel importieren exportieren erstellen

GPG verwaltet unter Linux die Schlüssel.

Man kann sich die **öffentlichen** Schlüssel anzeigen lassen mit:

```
gpg --list-keys
```

oder die **privaten** mit:

```
gpg --list-private-keys
```

2.1.1 Export Keys

Zum exportieren bedient man sich folgendem Befehl:

öffentlich

```
gpg -a --output public.gpg --export <Key-ID>
```

für den **privaten**:

```
gpg --export-secret-keys -a <Key-ID> > secret.asc
```

2.1.2 Import Keys

Die Schlüssel müssen nun auf dem anderen PC importiert werden, Dies passiert für den **öffentlichen** Schlüssel mit

```
gpg --import public.gpg
```

und für den **privaten** Schlüssel mit

```
gpg --allow-secret-key-import --import secret.asc
```

Um mit dem öffentlichen Schlüssel abreiten zu können, muss diesem noch vertraut werdem. Andernfalls erhält man z.B. in dem Programm pass eine Fehlermeldung. Zum vertrauen des Schlüssels geht man wie folgt vor:

```
gpg --edit-key <Key-ID>
```

Danach öffnet sich ein separates Programm im Terminal. Hier gibt man „trust“ ein und bestätigt dies mit „5“ für „Ich vertraue ihm absolut“. Danach kann man es wieder schließen. Danach kann mit dem öffentlichen Schlüssel gearbeitet werden. Der private Schlüssel wird nur zum entschlüsseln gebraucht und ihm muss daher nicht vertraut werden.

2.1.3 neues Key-Paar erstellen

Ein neues GPG-Key-Paar, also der öffentliche- mit einem privatem Schlüssel, wird erzeugt mit

```
gpg --full-generate-key
```

Danach muss „1“ gedrückt werden um einen RSA-Key zu erhalten. Danach wird nach der bit-Größe gefragt. Hier kann z.B. „1024“ eingegeben werden. Danach wird nach dem Ablaufdatum gefragt. Ist das Ablaufdatum erreicht so kann mit dem öffentlichen Schlüssel nichts neues mehr verschlüsselt werden. Es können aber noch 100 Jahre nach dem Ablaufdatum vorhandene verschlüsselte Dateien entschlüsselt werden. Er fragt dann weiter ob die Daten korrekt sind, Nach einem Namen, e-Mail, Kommentar. Danach wird nach einer Passphrase gefragt. Die Passphrase ist notwendig um den Key nachher benutzen zu können. Ohne die Passphrase zu wissen kann er später nicht benutzt werden. Danach ist der Key erstellt.

2.2 pass

Das Programm pass verschlüsselt, mit dem zuvor definierten öffentlichen Schlüssel, Text, und entschlüsselt dies mit dem dazugehörigen privaten Key wieder. Hat man ein Schlüsselpaar, bestehend aus öffentlichem- und privatem Schlüssel erstellt so kann man dieses mit

```
pass init <Key-ID>
```

für pass definieren. Achtung, wurde der Key zuvor importiert, so muss dem öffentlichen Key erst vertraut werden, ansonsten kann ein Verschlüsseln mit pass nicht stattfinden. Sollte man nun alte verschlüsselte Dateien haben, so können diese nun in „*~/home/USER/.password-store*“ kopiert werden. Das Entschlüsseln sollte nun mit

```
pass <verschlüsselte Datei>
```

funktionieren. Mit „pass“ werden alle verschlüsselten Inhalte angezeigt. Mit „pass insert <verschlüsselte Datei>“ kann dies dann wieder im Klartext dargestellt werden. Mit „pass rm <verschlüsselte Datei>“ kann das Teil wieder gelöscht werden. Mehrere Zeilen können mit

```
pass insert -m <verschlüsselte Datei>
```

angelegt werden.

3. Microsoft Office 2013

Vorraussetzungen:

- Office 2013 als 32-bit Version
- Die Pakete: wine winetricks mono-devel samba-winbind

3.1 Wine installieren

Wine wird in Debian wie folgt installiert:

32-Bit Architektur hinzufügen:

```
sudo dpkg --add-architecture i386
```

Den entsprechenden Key hinzufügen

```
wget -nc https://dl.winehq.org/wine-builds/Release.key  
sudo apt-key add Release.key
```

Danach kann in die */etc/apt/sources.list* folgendes hinzugefügt werden:

```
deb https://dl.winehq.org/wine-builds/debian/ DISTRO main
```

DISTRO ist hierbei urch das Debian System zu erssetzen. Danach das ganze updaten:

```
sudo apt-get update
```

Nun die StableVersion von Wine installieren:

```
sudo apt-get install --install-recommends winehq-stable
```

3.2 andere Pakete

Es fehlern nun noch

winetricks mono-devel samba-winbind

Diese bitte noch installieren. Am einfachsten mit

```
sudo apt-get install winetricks mono-devel winbind
```

Das war soweit die Vorbereitung. Nun kann Wine konfiguriert werden

3.3 Konfiguration Wine

Für Office 2013 einen neuen Ordner anlegen:

```
WINEPREFIX=~/.wine/Office2013 WINEARCH=win32 winecfg
```

Es muss Windows 7 verwendet werden. Nun noch folgendes nachinstallieren

```
WINEPREFIX=~/.wine/Office2013 WINEARCH=win32 winetricks corefonts
WINEPREFIX=~/.wine/Office2013 WINEARCH=win32 winetricks msxml4
WINEPREFIX=~/.wine/Office2013 WINEARCH=win32 winetricks msxml6
WINEPREFIX=~/.wine/Office2013 WINEARCH=win32 winetricks riched30
WINEPREFIX=~/.wine/Office2013 WINEARCH=win32 winetricks vb6run
WINEPREFIX=~/.wine/Office2013 WINEARCH=win32 winetricks d3dx11_42
WINEPREFIX=~/.wine/Office2013 WINEARCH=win32 winetricks d3dx11_43
```

Es soll nun in der Konfig bei Bibliotheken unter Bestehende ein Eintrag „*riched20 (Native, Builtin)“ sein. Die Konfig kann mit

```
WINEPREFIX=~/.wine/Office2013 WINEARCH=win32 winecfg
```

aufgerufen werden. Nun kann Office 2013 installiert werden

```
WINEPREFIX=~/.wine/Office2013 WINEARCH=win32 wine /PATH_TO_MOUNTPOINT/setup.e
```

Bei mir blieb die Installation irgendwann hängen, starten tut mein Word,... trotzdem. Word kann nun gestartet werden mit

```
WINEPREFIX=~/.wine/Office2013 WINEARCH=win32 wine "C:\\Program Files\\Microso
```

Hierbei wird es wahrscheinlich zu einem schwarzen Bildschirm kommen. Dies kann man folgendermaßen gerichtet werden. Wir machen zunächst die Regedit auf

```
WINEPREFIX=~/.wine/Office2013 WINEARCH=win32 wine regedit
```

Unter „HKEY_CURRENT_USER\\Software\\Wine“ legen wir den neuen Schlüssel „Direct3D“ an. Danach wird ein neuer DWORD Wert mit dem Namen „MaxVersionGL“ angelegt. Als Hexwert schreiben wir hier „30002“ hinein. Sollte dies immer noch nicht funktionieren so wird noch unter „HKEY_CURRENT_USER\\Software\\Wine“ ein Schlüssel mit „Direct2D“ angelegt. Auch hier muss wieder ein DWORD angelegt werden mit dem Namen „max_version_factory“. Hier sollte dann der Wert „0“ drinnenstehen. Das war's

4. DarkNet

Auf die Funktionen, wie das DarkNet funktioniert, soll hier nicht eingegangen werden. Es kann unter Linux Tor installiert werden. Wird dieser gestartet, so wird lokal (127.0.0.1) ein Proxy auf Port 9050 gestartet. Sofern nicht anders konfiguriert, startet er mit dem SOCKS Protokoll. Zum Starten einfach `tor` eingeben im Terminal. Vereinzelt kann es vorkommen, dass die Anwendung nicht richtig startet. Mit

```
sudo netstat -plnt | fgrep 9050
```

kann geschaut werden, was Port 9050 blockiert. Andernfalls kann auch mit

```
sudo killall tor
```

der Tor-Service beendet werden. Chrome kann nun bei Einstellen des richtigen Proxys sofort .onion Domains darstellen. Im Firefox muss der Proxy Socks remote dns einmal aktiviert werden. Hierzu

```
about:config
```

die Konfiguration aufmachen und „`network.proxy.socks_remote_dns`“ auf `true` setzen. Danach muss auch noch aktiviert werden, dass .onion Domains aufgelöst werden, dazu den Wert „`network.dns.blockDotOnion`“ auf `false` setzen. Mit dem Link

```
https://check.torproject.org/
```

kann geschaut werden, ob man richtig mit dem Tor-Netzwerk verbunden ist. Viel Spaß bei Erkunden des DarkNets.

5. VMware

5.0.1 Linux als Guest System

Mount Share Folder

In VMware Workstation muss für die Linux-VM unter „Settings/Options/Shared Folders“ der geteilte Ordner mit dem HostSystem angelegt und aktiviert sein. Zur Linux Guest VM: Es müssen die Programme

```
open-vm-tools  
open-vm-tools-desktop  
open-vm-tools-dkms
```

installiert sein. Auf einem DebianBasiertem System geht dies mit:

```
sudo apt-get install open-vm-tools open-vm-tools-desktop  
open-vm-tools-dkms
```

Danach kann der Ordner in der VM gemountet werden. Dies geschieht mit:

```
vmhgfs-fuse .host:dokumente /home/quiddi/shares/
```

Wobei hier „dokumente“ für den freigegebenen Ordnernamen steht und „/home/quiddi/shares/“ für das Verzeichnis auf dem GuestSystem, in welches gemountet wird.

6. Debian

Debian Version anzeigen:

```
cat /etc/issue
```

```
cat /etc/debian_version
```


7. Bluetooth Lautsprecher

Um einen Bluetooth Lautsprecher zu verbinden, werden die Pakete pulseaudio-alsa, pulseaudio-bluetooth, bluez, bluez-libs, bluez-utils benötigt. Diese bitte installieren.

```
pulseaudio-alsa pulseaudio-bluetooth bluez bluez-libs bluez-utils
```

Nach der Installation mit systemctl den entsprechenden Dienst mit

```
sudo systemctl start bluetooth.service
```

starten. Im Terminal den Lautsprecher verbinden. Hierzu

```
bluetoothctl
```

starten. Danach nach Geräten suchen lassen mit

```
[bluetooth]# power on
[bluetooth]# agent on
[bluetooth]# default-agent
[bluetooth]# scan on
```

Irgendwo sollte das Bluetooth Gerät nun auftauchen. Im besten Falle sieht man nun auch einen Namen des Gerätes. Nun das Gerät mit der Geräte-ID verbinden

```
[bluetooth]# pair 00:1D:43:6D:03:26
```

Nun nach dem Pairen verbinden

```
[bluetooth]# connect 00:1D:43:6D:03:26
```

Die ganze Sache beenden mit

```
[bluetooth]# scan off
[bluetooth]# exit
```

Um die Lautstärke nun vom Terminal aus steuern zu können, muss man noch

```
pulsemixer
```

installieren. Danach dies mit

`pulsemixer`

starten. Das Gerät auswählen und Musik hören.

Bei Problemen kann man noch

`pulseaudio –alsa`

installieren. Dies sollte möglich machen, dass man mit Alsamixer das PulseaudioGerät steuern kann. Dies klappte bei mir jedoch nicht.

Weiter gibt es das grafische Tool `blueman-manager`. Dies kann man mit

`blueman –manager`

starten.

8. Dynamische IP updaten

Kostenlos wird dies momentan nur noch von ddnss.de bzw. ddnss.ch angeboten. DynDNS kostet mittlerweile Geld. Voraussetzung ist, dass ein ddnss-Benutzer eingerichtet ist. Weiter muss ddclient installiert sein. Nun kann es mit

```
sudo dpkg-reconfigure ddclient
```

eingesetzt werden. Hier wird nun eingegeben:

- anderen Anbieter auswählen
- Name des Servers: ddnss.de
- Protokoll dyndns2
- Benutzer: testbenutzer
- PW eingeben
- Netzwerkschnittstelle, normal eth0
- Vollständige Adresse: meineadresse.ddnss.ch
- PPP-Verbindung, Nein
- als Daemon starten, JA
- Intervall, 300

Der status hierzu kann mit

```
ddclient -query
```

abgefragt werden. Nun sollte noch umgestellt werden, dass ddnss die externe IP und nicht die interne IP bekommt. Hierzu mit dem Editor nano die config verändern mit

```
sudo nano /etc/default/ddclient
```

Hier muss dann die „use=if“ auf

```
protocol=dyndns2
use=web, web=my.ip.fi /
server=ddnss.de
```

umgestellt werden. Danach sollte der Dienst ddclient die externe IP an ddnss.de weiterschicken. Um regelmäßig die IP Adresse an ddnss zu schicken muss man in der Datei „/etc/default/ddclient“

```
run_daemon="false" auf true setzen  
run_dhclient="true" auf false setzen  
run_ipup="true" auf false setzen
```

um alle 300 Sekunden die IP-Adresse an ddns zu schicken. Einmalig kann man den Dienst mit

```
sudo ddclient -debug -verbose -noquiet
```

starten.

9. Feste IP einstellen mit dhcpcd

```
sudo nano /etc/dhcpcd.conf
```

Hier sollte sehen als bsp.:

```
interface eth0
static ip_address=192.168.0.4/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1
```


10. SSH mit RSA-Keyfile

10.1 Keyfile am Client einspielen

```
$ mkdir ~/.ssh
$ cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
$ rm ~/id_rsa.pub
$ chmod 600 ~/.ssh/authorized_keys
```

Dann eine Konfig-Datei für SSH anlegen

```
nano /home/quiddi/.ssh/config
```

Hier dann reinschreiben:

```
Host testhost
  HostName testhost.ddnss.ch
  Port 1234
  User ssh-allowed-UserName
  IdentityFile /home/quiddi/.ssh/authorized_keys
  IdentitiesOnly yes
```

nun kann mit

```
ssh testhost
```

eine ssh verbindung aufgebaut werden

11. ESP8266

Software:

- Software für Arduino
- Bibliothek für die ESP8266, bekannt unter dem Namen „ESP8266wifi“. Dieser Ordner muss in den Ordner „libraries“ im Installationsordner der Arduino-Entwicklungsumgebung hineinkopiert werden.
- Nachtrag: vom 31.1.2020: Zusätzlich musste ich noch „Adafruit Unified Sensor“ von „Adafruit“ für den DHT-Temperatursensor aus der Bibliothek installieren.

Hardware:

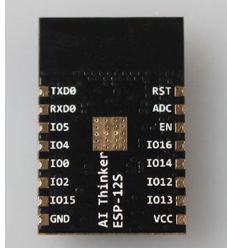


Um das Programm zu übertragen wurde der oben dargestellte Adapter verwendet. Nachteil ist, dass der Reset nicht vom Adapter kontrolliert werden kann. Deshalb muss ein ResetKnopf angebracht werden und dieser von Hand im richtigen Moment gedrückt werden (Weiteres siehe unten).

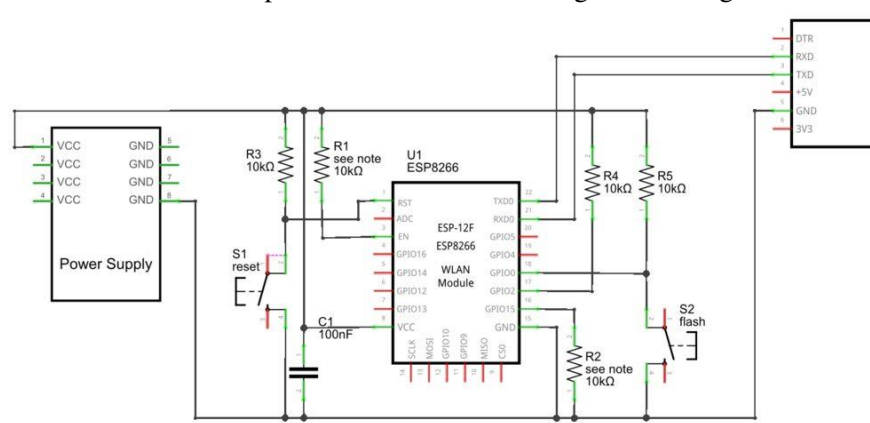
- Herzstück: ESP-12S
- Betriebsspannung: 3,3V manche USB/TTL Adapter liefern nicht genügend Strom. Es empfiehlt sich eine seperable Spannungsquelle mit 3,3V zu benutzen.

Anschluss des ESP-12S:

Anschlüsse des ESP-12S von hinten:



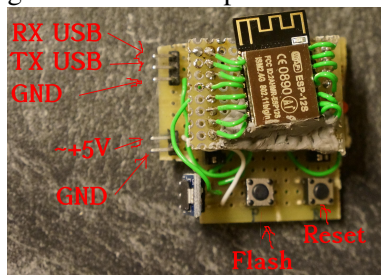
Diese sollten dann folgendermaßen verkabelt werden. Die Entstörkondensatoren und Pull-UP/Down Widerstände wurden weggelassen. Funktioniert hatte es trotzdem. Weiter ist zu beachten, dass der RX des USB/TTL-Adapters auf TX am ESP-12S geht und umgekehrt.



Danach wird die ArduinoEntwicklungsumgebung gestartet, es wird folgendes eingestellt:

- Board: NodeMCU 1.0, um dies auswählen zu können muss man in der ArduinoEntwicklungsumgebung auf Werkzeuge/Board/Boardverwalter gehen, dort bei Suchen „ESP8266“ eingeben und „esp8266 by esp8266 Community“ installieren.
- Baudrate: 115200

Danach kann man den Sketch kompilieren. Um ihn zu übertragen muss man auf den Button „Kompilieren mit Upload“ drücken. Dies ist der Button mit dem Pfeil. Nachdem die Kompilierung abgeschlossen ist und bevor die ArduinoEntwicklungsumgebung mit dem Upload beginnt muss am ESP-12S der Reset- und der FlashKnopf gleichzeitig gedrückt werden. Den Resetknopf gleich wieder loslassen und kurze Zeit später (vll. ca 1 sek) den Flashknopf auch wieder loslassen. Nun startet der Upload. Nach dem Upload startet die Software auf dem ESP erst nach einem Reset des ESP. Mit Werkzeuge/Serieller Monitor kann man nun den Output der Seriellen Ausgabe sehen. Bitte hier die Baudrate richtig anpassen. Im Quellcode für den ESP-12S ist es hier auf 115200 gestellt. Mein Adapter sah so aus:



31.3.2018 Authentifizierung mit geschalteten Ausgängen und DHCP(webserver_aut_mit_DHCP)

Änderung: Hier wurden Pin 4 und 5 auf Ausgang geschaltet und diese mit einer Zugangssperre geschützt. Bei Schalten der Ausgänge wird nun jedes mal ein Benutzername abgefragt.

Um die Webseite darzustellen hatte sich nun etwas verändert. Mit

```
void make_html()
```

wird in die String-Variable "content" der html-Code für die Webseite geschrieben. Die Funktion zum checken, welcher Link der Seite offen ist wird mit

```
server.on("/4/on",.....)
```

gemacht. In diesem Beispiel die Seite **IP/4/on**. Der ausführbare Code wird meist mit einer λ -Funktion übergeben. In dem Falle hier wird mit einer if-Abfrage geschaut, ob die Zugangsdaten passen. Bei Korrekter Eingabe wird in den String "output4State" der aktuelle Zustand geschrieben und Pin 4 auf HIGH geschaltet. Danach wird mit "make_html();" der String für die HTML erzeugt und dieser mit "server.send(200, text/html, content);" ausgegeben. Die Funktion "server.on(„/4/on“,.....)" wird für jeden Seitenaufruf geschrieben. Also insgesamt 5 Stück, Hauptseite, 4 an, 4 aus, 5 an, 5 aus.

12. HomeAutomation

12.1 SonOff S20 flash

Der Artikel baut auf dem GitHub Projekt von Tasmota auf. <https://github.com/arendst/Sonoff-Tasmota> Erklärt wird, wie man einen SonOff S20 mit der Tasmota-Firmware flasht.

Vorraussetzungen:

- SonOff S20 (hier am Bsp mit der Version v1.3, ist auf er Platine im Gerät aufgedruckt)
- Computer mit Linux
- USB-TTL Adapter
- ESPTool

Verbindung zum SonOff

Die Verbindung wird hier mit einem USB-TTL Adapter erklärt. In diesem Beispiel ist er unter „/dev/ttyUSB0“ eingehängt. Zu erwähnen ist nochmals, dass die ESP8266 mit 3,3 Volt betrieben werden. Bitte die Spannung hier sicherstellen, bei 5V gehen sie nämlich kaputt. Der USB-TTL

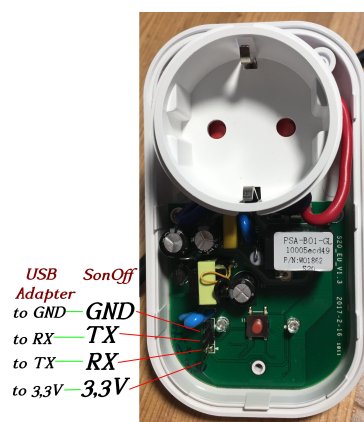


Abbildung 12.1.1: SonOff S20

Adapter ist wie in Abbildung 12.1.1 zu verbinden. Nun ist zu erwähnen, dass beim Flashen von

ESP8266 der GPIO0 auf Masse gezogen werden muss. Glücklicherweise ist beim SonOff S20 dieser mit dem Knopf verbunden. Dies kann dann durch Drücken des Knopfes erreicht werden. Wichtig ist, damit der ESP8266 dies erkennt um in den Flash-Modus zu kommen, auch ein Reset während des Drückens ausgelöst werden muss. Am einfachsten geht es, Betriebsspannung trennen, Knopf drücken und halten, Betriebsspannung anlegen, Knopf weiterhin gedrückt halten, Flashen am PC starten, Knopf weiterhin gedrückt halten, nach erfolgreicher Übertragung loslassen.

ESPTool:

Zum Ausführen des ESPTool wird Python benötigt. Das ESPTool selber kann unter <https://github.com/espressif/esptool> heruntergeladen werden.

Nach dem Herunterladen sollte als Erstes ein Backup mit

```
sudo python esptool.py --port /dev/ttyUSB0 read_flash
0x00000 0x100000 image1M.bin
```

gemacht werden (ESP8266 in den Flash-Modus vorher bringen). Danach kann man, muss man aber nicht, den ESP8266 abnullen, um alles zu löschen. Dies kann man mit

```
sudo python esptool.py --port /dev/ttyUSB0 erase_flash
```

machen. Auch hier gilt, ESP8266 in den Flash-Modus bringen.

Tasmota stellt nun eine universelle Firmware für alle möglichen SonOff Geräte zur Verfügung. Diese nennt er „sonoff.bin“. Auf seiner GitHub Seite findet man sie unter dem Link <http://thehackbox.org/tasmota/release/>. Diese kann nun einfach auf den SonOff S20 übertragen werden mit

```
sudo python esptool.py --port /dev/ttyUSB0 write_flash
--fs 1MB --fm dout 0x0 /home/quiddi/Downloads/sonoff.bin
```

Auf die Übertragung abwarten und nach einem Reboot findet sich ein neues W-Lan Netzwerk mit der SSID sonoff... Zu diesem bitte verbinden. Hier kann man nun zwei W-Lan Netzwerke einrichten. Danach macht das Gerät einen ReConnect. Nun hat man mehrere Möglichkeiten rauszufinden mit welcher IP der SonOff nun erreichbar ist. Ich tat dies mit nmap (Software für Linux um IP-Adressbereiche durchzuscannen). Dies geht mit

```
sudo nmap -sP 192.168.0.*
```

um beispielsweise von 192.168.0.1 bis 192.168.0.254 zu scannen. Das ESPTool zeigt beim Flashen die MAC-Adresse an. Auch nmap zeigt die MAC-Adresse an. Es sollte also einfach möglich sein das Gerät hier zu finden. Hat man die IP Adresse gefunden, kann man sich mit dieser auf dem Browser wie gehabt verbinden. Im Browser bietet die Tasmota-Firmware eine Konsole. Hierüber kann man nun eine feste IP vergeben.

```
IPAddress1 192.168.0.70
```

40 sekündiges Drücken des Knopfes macht übrigens einen Reset des Gerätes auf die Tasmota Standardeinstellungen.

GPIO:

Dargestellt in Tabelle 12.1.1

GPIO0	Knopf auf dem Gerät
GPIO1	RX pin
GPIO3	TX pin
GPIO12	Blaue LED
GPIO13	Grüne LED

Tabelle 12.1.1: GPIO Übersicht

Die Original SonOff S20 firmware befindet sich in data/sonoff/sonoff_s20_v1-3.bin.

12.2 SonOff T1 flash

Der Artikel baut auf dem GitHub Projekt von Tasmota auf. <https://github.com/arendst/Sonoff-Tasmota> Erklärt wird, wie man einen SonOff T1 mit der Tasmota-Firmware flasht.

Vorraussetzungen:

- SonOff T1 (hier am Bsp mit der Version v1.0, ist auf er Platine im Geré4t aufgedruckt)
- Computer mit Linux
- USB-TTL Adapter
- ESPTool

Verbindung zum SonOff

Die Verbindung wird hier mit einem USB-TTL Adapter erklärt. In diesem Beispiel ist er unter „/dev/ttyUSB0“ eingehängt. Zu erwähnen ist nochmals, dass die ESP8266 mit 3,3 Volt betrieben werden. Bitte die Spannung hier sicherstellen, bei 5V gehen sie nämlich kaputt. Der USB-TTL

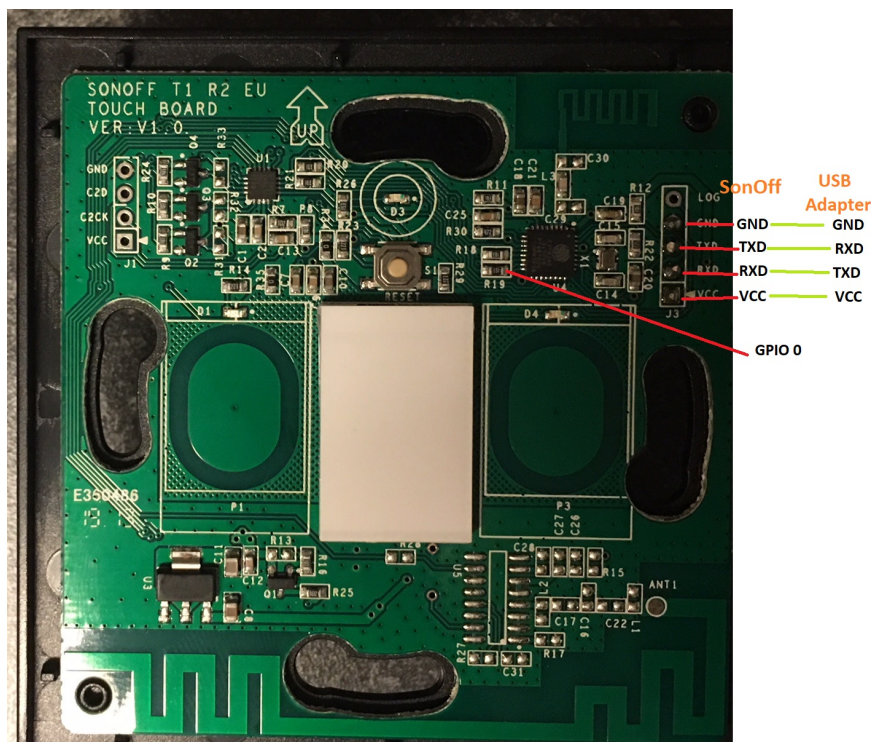


Abbildung 12.2.1: SonOff T1 Wandschalter

Adapter ist wie in Abbildung 12.2.1 zu verbinden. Nun ist zu erwähnen, dass beim Flashen von ESP8266 der GPIO0 auf Masse gezogen werden muss. Leider lässt sich beim SonOff T1, nicht wie beim SonOff S20, durch drücken des Einschalters dies Brücken. Hierzu muss Masse an R19 an der näher gelegenen Seite zum ESP8266 gelegt werden. Wichtig ist, damit der ESP8266 dies erkennt um in den Flash-Modus zu kommen, auch ein Reset nach dem Anlegen ausgelöst werden muss. Am einfachsten geht es, Betriebsspannung trennen, Drahtbrücke verbinden, Betriebsspannung anlegen, Flashen am PC starten und warten bis es fertig ist. Nach erfolgreichem Flashen die Drahtbrücke wieder entfernen.

ESPTool:

Zum Ausführen des ESPTool wird Python benötigt. Das ESPTool selber kann unter

GPIO0	Knopf auf dem Gerät
GPIO0	Taster 1
GPIO9	Taster 2
GPIO10	Taster 3
GPIO12	blaue LED Taster 1
GPIO5	blaue LED Taster 2
GPIO4	blaue LED Taster 3
GPIO13	blaue LED mit W-Lan Symbol
GPIO1	RX pin
GPIO3	TX pin

Tabelle 12.2.1: GPIO übersicht T1

url <https://github.com/espressif/esptool> heruntergeladen werden.

Nach dem Herunterladen sollte als Erstes ein Backup mit

```
sudo python esptool.py --port /dev/ttyUSB0 read_flash
0x00000 0x100000 image1M.bin \
```

gemacht werden (ESP8266 in den Flash-Modus vorher bringen). Danach kann man, muss man aber nicht, den ESP8266 abnullen, um alles zu flashen. Dies kann man mit

```
sudo python esptool.py --port /dev/ttyUSB0 erase_flash \
```

machen. Auch hier gilt, ESP8266 in den Flash-Modus bringen.

Tasmota stellt nun eine universelle Firmware für alle möglichen SonOff Geräte zur Verfügung. Diese nennt er

glqq sonoff.bin

grqq. Auf seiner GitHub Seite findet man sie unter dem Link <http://thehackbox.org/tasmota/release/>. Diese kann nun einfach auf den SonOff T1 übertragen werden mit

```
sudo python esptool.py --port /dev/ttyUSB0 write_flash
-fs 1MB -fm dout 0x0 /home/quiddi/Downloads/sonoff.bin
```

Auf die Übertragung abwarten und nach einem Reboot findet sich ein neues W-Lan Netzwerk mit der SSID sonoff... Zu diesem bitte verbinden. Hier kann man nun zwei W-Lan Netzwerke einrichten. Danach macht das Gerät einen ReConnect. Nun hat man mehrere Möglichkeiten rauszufinden mit welcher IP der SonOff nun erreichbar ist. Ich tat dies mit nmap (Software für Linux um IP-Adressbereiche durchzuscannen). Dies geht mit

```
sudo nmap -sP 192.168.0.*
```

um beispielsweise von 192.168.0.1 bis 192.168.0.254 zu scannen. Das ESPTool zeigt beim Flashen die MAC-Adresse an. Auch nmap zeigt die MAC-Adresse an. Es sollte also einfach möglich sein das Gerät hier zu finden. Hat man die IP Adresse gefunden, kann man sich mit dieser auf dem Browser wie gehabt verbinden. Im Browser bietet die Tasmota-Firmware eine Konsole. Hierüber kann man nun eine feste IP vergeben.

```
IPAddress1 192.168.0.70
```

40 sekündiges Drücken des Knopfes macht übrigens einen Reset des Gerätes auf die Tasmota Standardeinstellungen.

GPIO:

Dargestellt in Tabelle 12.2.1 Die Original SonOff T1 firmware befindet sich in data/sonoff/sonoff_t1_v1-0.bin.

12.3 SonOff RF Flash

Es wird der Flash mit Tasmota beschrieben. Es handelt sich hier um einen „Sonoff RF R2 Power v1.0 1937“. Er ist abgebildet in Abbildung 12.3.1.

SONOFF®



Abbildung 12.3.1: SonOff RF Basic

- Geflasht werden kann die normale Tasmota Firmware.
- Um in den Flash-Modus zu kommen, muss GPIO0 auf Masse gezogen werden. GPIO0 kann auf Masse gezogen werden, durch drücken des Knopfes am Gerät. Wie immer, ESP ausschalten, GPIO0 auf Masse legen, ESP einschalten. Danach kann das Flashen am PC gestartet werden mit:

```
sudo python esptool.py --port /dev/ttyUSB0 write_flash  
-fs 1MB -fm dout 0x0 /home/quiddi/Downloads/sonoff.bin
```

- Danach muss unter Einstellungen der Gerätetyp auf 2 „Sonoff RF“ umgestellt werden.
- Zum Pairen eines RF-Sender den GPIO0-Knopf für 3 Sekunden drücken, bis das rote Licht einmal blinkt. Danach muss auf dem 433MHz-Sender einmal der Knopf zum Schalten des Kontaktes gedrückt werden. Als dritten Schritt drückt man nun den GPIO0-Taster noch einmal für 5 Sekunden, bis die rote Lampe zweimal blinkt. Danach ist der RF-Sender eingelernt und kann schalten.