Google-Assistent (Google Home) Sprachsteuerung und Tasmota ohne IO-Broker

Hier wird beschrieben wie mit Hilfe eines Raspberry's, eine Google-Assistent Sprachsteuerung von Tasmota Geräten realisiert werden kann.

EDIT 23.01.23:

Seit der Änderung der Google Assistant API zum 31.08.22 funktioniert eine Steuerung via IFTTT mit "Passphrase" leider nicht mehr.

Eine Sprachsteuerung ohne einen Drittanbieter bzw. Clouddienst ist seitdem nicht mehr möglich.

Nachdem immer wieder mal die Frage nach Google-Assistent Sprachsteuerung in Verbindung mit Tasmota aufkommt, habe ich hier mal ein kleines Tutorial verfasst in dem beschrieben wird was dazu benötigt wird und wie es umgesetzt werden kann.

Dies wird benötigt:

1. Einen Google Konto und einen eingerichteten Google–Sprachassistenten

2. Ein kostenloses IFTTT Konto das mit dem Google Konto verknüpft ist

3. Einen funktionalen Linux Server, in diesem Beispiel einen Raspberry

4. Erreichbarkeit aus dem Internet DNS/Portfreigabe

5. Grundwissen zur Raspberry Installation, HTML und Bash Scripting

Auf das Einrichten des Google-Sprachassistent wird hier nicht weiter eingegangen. Ebenso wie ein IFTTT Account angelegt wird und mit dem Google Konto verknüpft wird. Dazu gibt es genug Anleitungen und Beschreibungen die via Google gefunden und abgerufen werden können.

Funktionsprinzip:

Da Tasmota bisher nicht direkt in Google Home eingebunden werden kann, wird hier der Umweg über den IFTTT (if this, than that) Service gewählt, um eine Sprachsteuerung zu ermöglichen. Ist der Google Assistent mit dem IFTTT Dienst verknüpft werden Sprachbefehle an diesen weitergegeben. IFTTT ermöglicht es sogenannte Applets zu erstellen die nach folgenden Prinzip funktionieren: Wenn der Sprachbefehl z.B. „*Ok Google, Sonoff A ein*“ vom Google-Assistent bei IFTTT eintrifft, mache einen Webhook. Webhook bedeutet im Prinzip nichts anderes als „Rufe einen Internet Link auf und übergib Daten an das Link Ziel“.

Erreichbarkeit aus dem Internet:

Um für diesen Webaufruf immer erreichbar zu sein benötigen wir einen Domainnamen, da durch den Internet Provider in der Regal alle 24 Stunden eine neue IP vergeben wird.

Wer eine Fritzbox hat, hat es relativ einfach. Der kann sich innerhalb der FB einfach ein myfritz Konto erstellen und den myfritz Internetzugang aktivieren. Ist das geschehen bekommt man eine „Ihre myFRITZ-Adresse“ also einen Link angezeigt: *<xyz-irgendwas>.myfritz.net* über den die FB aus dem Internet künftig erreichbar ist. Wichtig ist bei der FB das wir unter Fritz-Box-Dienste den *‚TCP-Port für HTTPS‘* von 443 auf einen anderen unserer Wahl ändern, da wir diesen später noch zwingend für IFTTT benötigen. Merkt euch den neuen *‚TCP-Port für HTTPS‘*, den über diesen ist die FB künftig für die Fernwartung aus dem Internet

erreichbar.

Eine weiter Möglichkeit, die auch für andere Router gilt, ist sich bei einem DynDNS Provider wie. Z.B ddnss.de ein kostenloses Konto einzurichten, sich dort einen Domainnamen zu erstellen und diese Kontodaten in der FB/Router unter DynDNS einzutragen.

Jetzt fehlt nur noch eine TCP Portfreigabe des Port 443 auf die IP unseres Raspberry's

Einrichten des HTTPS/MQTT Servers auf dem Raspberry

Es gibt viele HTTPS Server für Linux. Meine Wahl ist in diesem Fall auf den mini_httpd von acme gefallen. Er ist sehr klein, Resourcen schonend und unterstützt SSL/Benutzer-Verwaltung sowie CGI. Anbei findet ihr in der angehängten Datei mini_httpd.zip den aktuellen mini_httpd fertig vorkonfiguriert.

Das Zip entpackt ihr bitte im Ordner /opt. Ist das geschehen, sollte es einen neuen Ordner mini_httpd geben. In diesem befinden sich nun mehrere Bash Scripte die beim Einrichten helfen.

Zuerst müssen wir uns ein gültiges SSL Zertifikat erstellen.

Dazu einfach das Bash-Script **3_Make_Cert.sh** ausführen und den Anweisungen folgen. Heraus kommt ein Zertifikat: *cert.pem*

Als nächste macht es Sinn einen Individuellen Benutzer mit Passwort anzulegen. Dazu wird das Script **4_Make_Passwordfile.sh** ausgeführt. Standartmäßig ist bereits ein Passwort File vorhanden. Der Benutzer ist **admin**, das Passwort **0000**. Mit dem Anlegen eines neuen Passwortfiles wird das alte und deren Benutzer überschrieben.

Ist dies geschehen kann der Server mit dem Bash-Script **1_Start_mini_httpd.sh** gestartet werden und mit dem Script **2_Stop_mini_httpd.sh** wieder beendet werden.

Das Bash-Script **5_Make_Autostart.sh** erzeugt ein Autostart Datei in /etc/init.d und macht diese ausführbar. Ein erneutes ausführen des Script's löscht diese Datei bei Bedarf wieder. Je nachdem welches Raspian läuft kann es sein das der Autostart noch anderweitig aktiviert werden muss. (*sudo systemctl enable mini_httpd.service*)

Ist der HTTPS Server erfolgreich gestartet, der Internetzugang angelegt, die Portfreigabe erfolgt, können wir jetzt unseren ersten Internetzugriff auf unseren HTTPS Server versuchen:

https://<meine-IP-oder-mein-Domainname>

Hat alles geklappt wird uns jetzt der Browser erst mal warnen das wir uns auf einer „unsicheren Seite" bewegen und fragen ob er das wirklich machen soll. Das kommt daher, weil unser SSL Zertifikat selbst erstellt und signiert wurde und nicht von einer offiziellen Zertifikatsstelle. Nachdem wir zugestimmt haben das wir die Seite trotzdem aufrufen möchten werden wir nach dem vergebenen Benutzernamen und Passwort gefragt und befinden uns nach erfolgter Eingabe auf unserem Raspberry HTTPS Server.

Links oben in der Menu Leiste der Webseite wird angezeigt ob der benötigte MQTT Dienst Mosquitto installiert und aktiv ist. Sollte die Anzeige rot sein (N) dann müssen wir diesen noch installieren.

Dazu geben wir in der Konsole: **sudo apt-get install mosquitto** ein und folgen den Anweisungen.

Anschließend nach Bedarf im Ordner/etc/mosquitto die mosquitto.conf anpassen. Näheres hierzu findet sich in diversen Raspberry Tutorials.

Nach der Installation kann der MQTT Server gestartet werden: **sudo /etc/init.d/mosquitto start**

Oder über die WebOberfäche des HTTPS Servers mit *MQTT on/off*

Auch hier gilt: Autostart Aktivierungen können je nach Raspian Variante abweichen.

Läuft der MQTT Server auf dem Raspberry, können als nächstes die Sonoff's eingebunden werden.

Dazu MQTT im Sonoff aktivieren und anschließend in den MQTT Einstellungen die Server Daten eingeben und einen TOPIC Namen vergeben, in unserm Beispiel, 1.Sonoff: **A,** 2.Sonoff: **B**

Wenn soweit alles funktioniert hat, können wir auch schon über die Weboberflache unseres HTTPS Servers den Sonoff A und B über die Buttons umschalten.

IFTTT Applet einrichten:

Jetzt heißt es ein IFTTT Applet einzurichten über den der Google-Assistent den HTTPS Server erreichen kann um dort ein CGI Script (dazu kommen wir noch) auszuführen das dann den gewünschten Sonoff schaltet. Das Ganze läuft exakt parallel zu der Weboberfäche des HTTPS Servers.

- auf "*myApplets*" und dann auf „*new Applet*"

- auf "+*this*" klicken und nach Google Assistent suchen und diesen anklicken

- „*Say a phrase with a text ingredient*" auswählen

- "*What do you want to say?*" folgendes eintragen: Sonoff $

- "*Language*" auf German stellen

- "*Create Trigger*"

- "+*that*" klicken und nach "*Webhook*" suchen und diesen anklicken

- „*Make a web request*" anklicken und diese URL eingeben:
https://<Benutzer>:<password>@<meine-Domainname>/cgi-bin/mqtt.sh?{{TextField}}

- "*Method*" auf "*GET*" stellen

- "*Content Type (optional)*" auf "*text/plain*" stellen

- "*Save*"

Fertig !

Jetzt kann mit "*Ok Google, Sonoff A ein*"; „*Ok Google, Sonoff A aus*", „*Ok Google, Sonoff B ein*", „*Ok Google, Sonoff B aus*" die Geräte geschaltet werden.

Funktion des CGI/Bash Scripting:

Der IFTTT Webhook, macht im Prinzip nichts anderes als das CGI Script *mqtt.sh* im Ordner */opt/mini_httpd/www/cgi-bin/* auszuführen, wenn als Trigger Wort „Sonoff" gesagt wird und die Worte danach ebenfalls mit zu übermitteln. Der Schaltbefehl von IFTTT schaut im Fall: *"Sonoff A ein"* folgendermaßen aus:

https://<Benutzer>:<password>@<meine-Domainname>/cgi-bin/mqtt.sh?A%20ein

Alles was nach dem „?" in der URL steht sind Daten die an das Bash-Script übergeben werden und stehen bei CGI immer in der Variablen QUERY_STRING. In diesem Fall *„A%20ein"*.

Hier wird dann der QUERY_STRING erst mal ausgewertet und die übergebenen Werte an Variablen (REQ1 für den ersten Parameter, REQ2 für den zweiten) übergeben. Trennzeichen ist dabei im Prinzip *„%20“*

Im Script werden dann die Inhalte der Variablen ausgewertet. Ist der übermittelte Befehl *„ein“* oder *„an“* oder *„1“* (*„1“* wird von google gerne statt *„ein“* erkannt) wird als Schaltbefehl *„on“* verwendet. Bei *„aus“* gilt der Schaltbefehl *„off“*. Dies ist nötig weil Sonoff als Befehle nur *„on“* oder *„off“* annimmt.

Im nächsten Schritt führt das Script einen MQTT Nachricht aus: Schalte das Gerät mit dem Namen der in Variable REQ1 (im Beispiel *„A“*) steht, mit dem Schaltbefehl der ausgewertet wurde also *„on“* oder *„off“* je nachdem was gesagt wurde´.

Bash

```
#!/bin/bash
#created by HoerMirAuf
#MQTT Server:
MQHOST=localhost
# Webseite erstellen:
echo "Content-type: text/html"
echo ""
echo "<html>"
echo "<head>"
echo "<title>system</title>"
echo "</head>"
echo "<body><center><br><br>"
#QUERY_STRING Auswertung (parsen) und Umwandlung in Grossschrift:
REQ1=$(echo $QUERY_STRING | sed "s/%20/\&/g" | cut -d"&" -f1 | tr '[:lower:]' '[:upper:]')
REQ2=$(echo $QUERY_STRING | sed "s/%20/\&/g" | cut -d"&" -f2 | tr '[:lower:]' '[:upper:]')
case $REQ2 in
EIN) CMD="on";;
AN) CMD="on";;
1) CMD="on";;
AUS) CMD="off";;
*) CMD=$REQ2;;
esac
sudo mosquitto_pub -h $MQHOST -t cmnd/$REQ1/POWER -m $CMD
echo "Sonoff <font color='green'>$REQ1</font> wurde geschaltet <font color='green'>$REQ2</font>"
echo "</body>"
echo "</html>"
```

Alles anzeigen

Das kann natürlich je nach Programmierfähigkeiten auf weiter Geräte und Funktionen erweitert werden.