Svxlink, SHARI/Aliexpress-Hotspot & MQTT + Node-RED

letzte Aktualisierung: 10.5.2023

1. Vorwort

Neben den bekannten Hotspot-Lösungen wie DJSpot und SPSpot (auch „Daniel-Spot“ genannt) hat sich ja nun auch der auf dem SHARI-Projekt basierende Hotspot etabliert, den man bei Aliexpress als fertiges Modul erwerben kann.
Da sich ja nun inzwischen auch Zubehör wie der SVXCube immer größerer Beliebtheit erfreuen, will ich hier mal kurz darstellen, wie diese Erweiterung auch auf diesem Hotspot installiert und eingesetzt werden kann. Dazu erweitern wir unsere Linux-Umgebung um zwei Dinge, einerseits einem MQTT-Broker und einem Tool namens Node-RED. MQTT steht für MQ Telemetry Transport und ist ein offenes Netzwerkprotokoll für Machine-to-Machine-Kommunikation (M2M), das die Übertragung von Telemetriedaten in Form von Nachrichten zwischen Geräten ermöglicht.
Node-RED ist ein von IBM entwickeltes grafisches Entwicklungswerkzeug. Die Software ermöglicht es, Anwendungsfälle im Bereich des Internets der Dinge (IoT) mit einem einfachen Baukastenprinzip umzusetzen.
Wie wir diese beiden Erweiterungen auf unserem Linux zum Laufen bekommen, werde ich hier mal in komprimierter Form darstellen.

2. Voraussetzungen

Meine Anleitung wird sich auf ein normales Raspian/Debian 11 („Bullseye“) beziehen, wo diese beiden Komponenten noch nicht installiert sind.
Auch hier muss ich nochmals deutlich darauf verweisen, es ist keine Plug'n'Play-Lösung, grundlegende Kenntnisse im Umgang mit Linux sind dafür notwendig.
Da wir das ja für die Steuerung unseres SVXLINK einsetzen wollen, hier noch mal die Voraussetzungen:

- ein Raspian/Debian-Linux, empfohlen in der Version 11 „Bullseye“
- ein funktionierendes, lauffähiges SVXLINK in möglichst aktueller Version
- ein funktionierendes lokales Netzwerk

Ebenfalls sollte man sich mit den Möglichkeiten des Node-RED eingehend beschäftigen, besonders was die Erstellung von Workflows angeht. Einfach mal was „zusammenklicken“ wird nicht funktionieren, man muss verstehen, wie das System arbeitet und wie es funktioniert:

MQTT-Client (sendet Nachrichten an den MQTT-Broker) ↔ Netzwerkübertragung ↔ MQTT-Broker (empfängt die Nachrichten der MQTT-Clients) ↔ Node-RED (erzeugt Anweisungen oder Befehle für das Betriebssystem oder Applikationen aus dem Inhalt der MQTT-Nachrichten) ↔ OS (führt diese Befehle aus)

Ergänzenderweise will ich noch anfügen, die Kommunikation ist eigentlich bi-direktional, also auch der MQTT-Broker kann Nachrichten erzeugen, die die MQTT-Clients empfangen können, das wäre z.B. bei Daten wie Temperatur sinnvoll.

3. Installationen

3.1 MQTT

Unter Linux ist der MQTT-Broker das Paket Mosquitto. Das installieren wir jetzt:

```
$ sudo apt-get install mosquitto mosquitto-clients
```

Anschließend stellen wir es so ein, das es beim Systemstart automatisch gestartet wird:

```
$ sudo systemctl enable mosquitto
$ sudo systemctl start mosquitto
```

Für einen ersten Test müssen wir noch etwas nachkonfigurieren, wir beginnen erstmal ohne Authentifizierung.
Wir erstellen dazu eine neue Konfig-Datei:

```
$ sudo nano /etc/mosquitto/conf.d/local.conf
```

mit folgendem Inhalt:

```
# Port auf dem der Dienst arbeitet
listener 1883
# ohne Authentifizierung
allow_anonymous true
```

und speichern die Datei.
Anschließend starten wir mosquitto neu:

```
$ sudo sudo systemctl restart mosquitto
```

Das wär's dann auch schon - fast.
Natürlich sollten wir den Zugriff absichern und mit Passwortauthentifizierung arbeiten.
Lest dazu einfach Michaels/DG6IMF Artikel "Mosquitto installieren" ab dem Abschnitt „Konfiguration: Kommunikation mit Authentifizierung mit Benutzername und Passwort“, da steht alles, wie es geht und was man ändern muss und ebenfalls, wie man testen kann, ob mosquitto korrekt läuft.

3.2 Node-RED

Jetzt kommt Teil 2, wir installieren jetzt Node-RED. Node-RED sorgt dafür, das die per MQTT eingelieferten Daten in konkrete Befehle umgesetzt werden, also eine Aktion auslösen. Ein Beispiel wäre das Umschalten von Sprechgruppen am SVXLINK.

 In einigen Anleitungen wird eine Installation wie üblich mit *apt-get install nodered* gemacht.
DAS IST ABER lt. nodered.org NICHT DIE EMPFOHLENE METHODE !
Wir machen das also so, wie nodered.org das empfiehlt.
Zuerst loggen wir uns als User svxlink ein, NICHT(!!!) als root. Sollte der User svxlink nicht zum einloggen verfügbar sein, sondern zwar angelegt, das aber ohne Login-Shell, muss das geändert werden. Es muss ein Login als User svxlink möglich sein.
Wenn wir uns also als User svxlink eingeloggt haben, installieren wir Node-RED wie folgt (Internetzugriff erforderlich!):
Zuerst müssen wir prüfen, ob alle Abhängigkeiten erfüllt sind:

```
$ sudo apt-get install build-essential git curl
```

Dann kommt die eigentliche Node-RED-Installation an die Reihe (NICHT mit sudo ausführen !!!):

```
$ bash <(curl -sL
https://raw.githubusercontent.com/node-red/linux-installers/master/deb/updat
e-nodejs-and-nodered) --nodered-user=svxlink
```

Das installiert Node-RED, welches mit den Rechten des Users svxlink läuft. Das ist wichtig für spätere Steuerung des SVXLINK über dessen PTY. Dazu später noch einige Anmerkungen.

 Es kommt zum Ende der Installation ggf. eine Abfrage des Installationsprogramms, ob gewisse Sicherheitseinstellungen des Linux geändert werden sollen. Bitte NICHT mit Y(es) bestätigen, sondern mit N(o) beantworten, was auch bereits so vorgegeben ist !
Jetzt machen wir wieder alles auto-startend beim Boot-Vorgang:

```
$ sudo systemctl enable nodered.service
$ sudo systemctl start nodered.service
```

Nach der Installation müssen wir noch eine kleine Änderung bzw. Anpassung vornehmen:
Wir editieren die /lib/systemd/system/nodered.service wie folgt:

```
$ sudo nano /lib/systemd/system/nodered.service
```

und ändern folgende Zeile:

```
ORIGINAL:
Environment="NODE_OPTIONS=--max_old_space_size=512"
wird geändert in
Environment="NODE_OPTIONS=--max_old_space_size=256"
```

Das reduziert den möglichen RAM-Verbrauch des Node-RED von 512MB auf 256MB, falls Ihr das auf schwachen Systemen wie dem Pi ZERO mit nur 512MB RAM ingesamt laufen lassen wollt.
Anschließend starten wir Node-RED wie folgt neu:

```
$ sudo systemctl daemon-reload && sudo systemctl restart nodered.service
```

Jetzt sollten wir per Webbrowser auf das Webinterface vom Node-RED zugreifen können, am besten von einem PC aus:

```
http://<IP-Adresse Pi>:1880
```

 Per default ist der Webzugriff auf das Node-RED NICHT abgesichert, d.h. es gibt keine User/Passwortabfrage. Hinweise, welche Möglichkeiten der Absicherung bestehen, findet man in der Dokumentation vom Node-RED.
Ich empfehle, zumindest eine Passwortauthentifikation zu aktivieren.

3.2.1 Node-RED und die PTY des SVXLINK

Da wir ja MQTT und Node-RED zur Steuerung unseres SVXLINK einsetzen wollen, z.B. mit dem SVXCube, ein paar grundlegende Hinweise und Hintergrundinformationen, warum wir Node-RED als User svxlink laufen lassen sollten.
PTY sind sog. Pseudo-Terminals, die auf einen Speicherbereich laufender Prozesse schreiben/lesen dürfen, dem sog. shared memory. Diese sind häufig besonders abgesichert, d.h. deren Zugriffsrechte sind sehr restriktiv eingestellt. So auch bei unserem SVXLINK. Die u.a. zur DTMF-Steuerung erzeugten PTY

```
svxlink.conf:
[SimplexLogic]
DTMF_CTRL_PTY=/control/dtmf
bzw.
[RepeaterLogic]
DTMF_CTRL_PTY=/control/dtmf
```

dürfen nur durch Prozesse beschrieben werden, die unter dem User svxlink laufen. In unserem Fall wäre das u.a. der Webserver, der das Dashboard bereitstellt oder eben Node-RED, welches Befehle, die per MQTT eingeliefert werden, weiter zum SVXLINK schickt und ausführt. In meinem Beispiel zeigt das PTY auf /control/dtmf, was nur einen symbolic-link darstellt, der eigentlich auf /dev/pts/1 verweist:

```
lrwxrwxrwx 1 svxlink svxlink 10  9. Mai 12:56 dtmf -> /dev/pts/1
```

Sehen wir uns die Rechte auf /dev/pts/1 einmal an

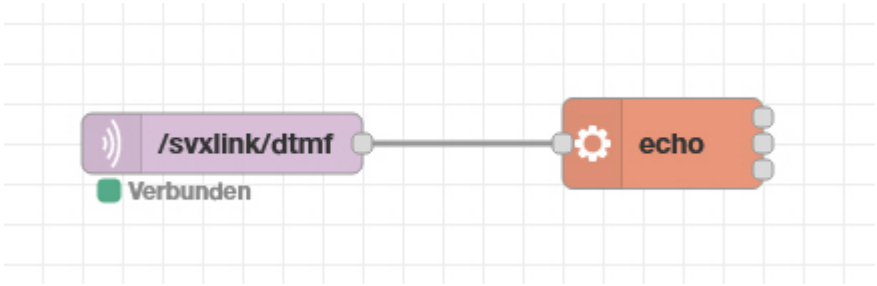
```
crw--w---- 1 svxlink tty  136, 1  9. Mai 19:06 1
```

erkennen wir, nur der User svxlink und die Gruppe tty darf schreiben. Alle anderen bekämen dann „Permission denied“, also „Zugriff verweigert“. Um also dieses Problem möglichst zu umgehen, lassen wir der Vereinfachung halber also Node-RED als User svxlink laufen, damit werden alle Befehle, die es ausführt, auch als User svxlink bzw. mit seinen Rechten ausgeführt.

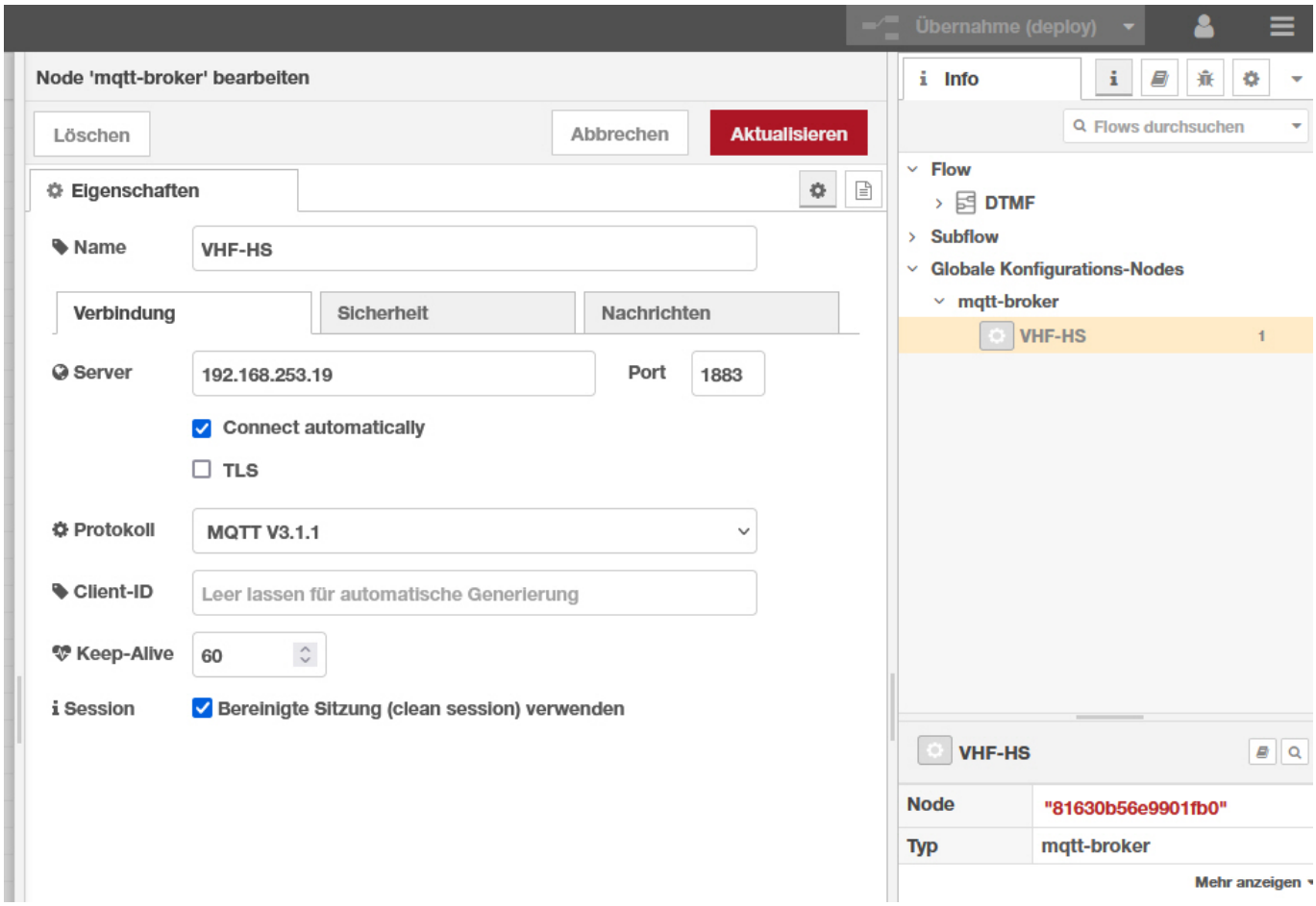
3.2.2 Konfiguration Node-RED

Ich möchte erneut darauf verweisen, Node-RED ist ein grafisches Entwicklungswerkzeug, dessen Anwendung man verstehen und trainieren muss. Andernfalls erzielt man nicht die Ergebnisse, die man sich wünscht. Man muss es also in gewisser Weise „erlernen“, um es richtig anwenden zu können. Dieses Grundwissen kann hier an dieser Stelle nicht vermittelt werden.

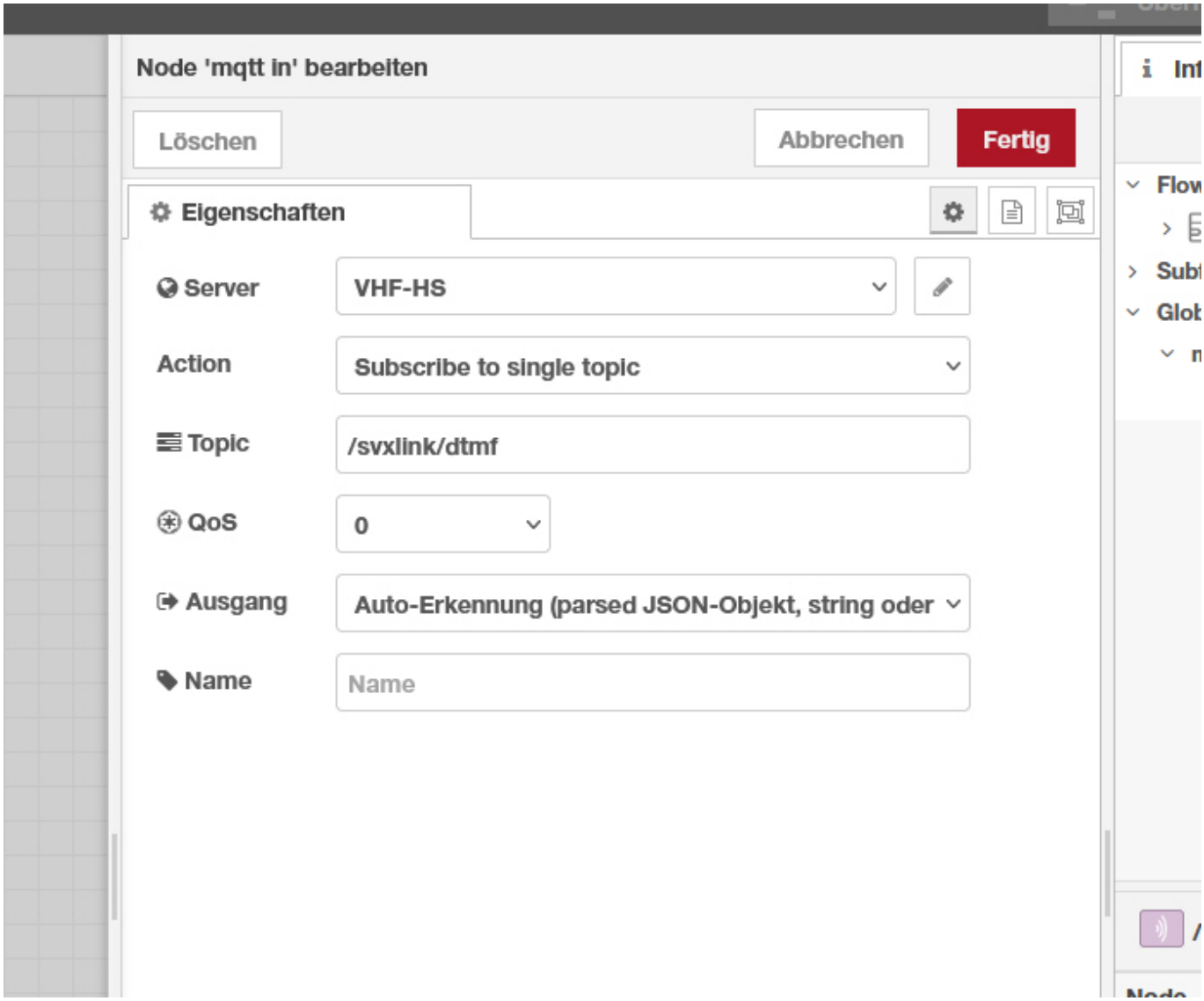
Wir loggen uns ins Node-RED ein und definieren einen Workflow:



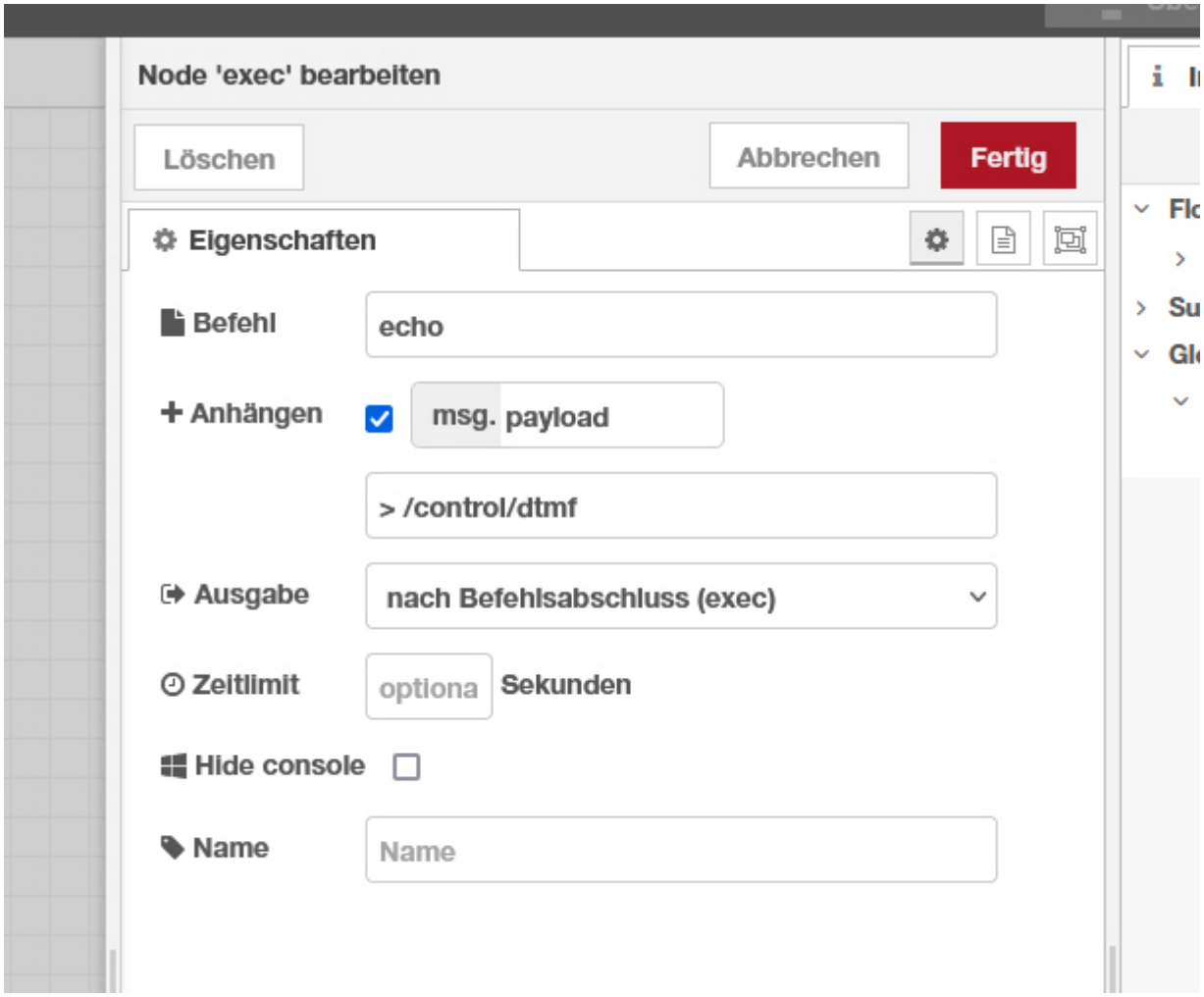
Zuerst müssen wir unser Ziel, also den MQTT-Broker mal definieren:



Dann müssen wir die sog. Topic angeben, hier wäre das /svxlink/dtmf, auf die Node-RED reagieren soll:



Jetzt müssen wir noch den Befehl definieren, der nach Eintreffen der Nachricht ausgeführt werden soll:



Jetzt können wir z.B. mit EasyMQTT oder einen anderen MQTT-Client eine Message mit dem Inhalt „*9120#“ absetzten, was dann durch das Node-RED folgenden Befehl erzeugt:

```
echo '*9120#' > /control/dtmf
```

und in diesem Fall ein Umschalten des SVXLINK via dessen DTMF-PTY auf die TG20 auslöst. Die Variable *msg.payload* stellt hier den Inhalt der vom MQTT-Client gesendeten Nachricht dar und wird in dem Befehl durch das Node-RED entsprechend ergänzt/eingefügt und der Befehl wird ausgeführt. Als Ergebnis schaltet SVXLINK dann auf die TG20.

4. Anwendungsmöglichkeiten

Wie zu Beginn dieses Artikels bereits beschrieben, sind die Einsatzmöglichkeiten vielfältig. Ich denke, so liesse sich z.B. auch der SVXCube mit dem Aliexpress/SHARI-HS steuern. Ich selbst habe das mit EasyMQTT vom iPhone aus getestet und es funktionierte ohne Probleme (entsprechend der Anleitung Siri & Co. steuert DJ-Spot). Weitere Anwendungen sind natürlich auch denkbar, was man mittels MQTT noch so alles machen könnte. Weitere Ideen sind also gefragt

Heiko / DL1BZ