Nibe Gateway

Hallo zusammen,

da es oft hilft, wenn man gemeinsam an etwas arbeitet, möchte ich hier einfach mal meine Lösung zur Integration einer Nibe Heizung vorstellen.

Das Ganze ist noch sehr stark WIP (work in progress). Also sicherlich an vielen Stellen noch unvollständig und wenig dokumentiert (weil das ja auch immer so riesig Spaß macht...). Wer aber selbst ein wenig mitdenkt, oder die nötigen Kenntnisse mitbringt oder sich nicht scheut einfach konkret nachzufragen , bekommt das auf jeden Fall schon irgendwie ans Laufen!

Meine Bitte vorweg ist daher ganz dringend: **Helft mit, korrigiert, hakt nach und habt Spaß damit!**

Ziel / Warum?

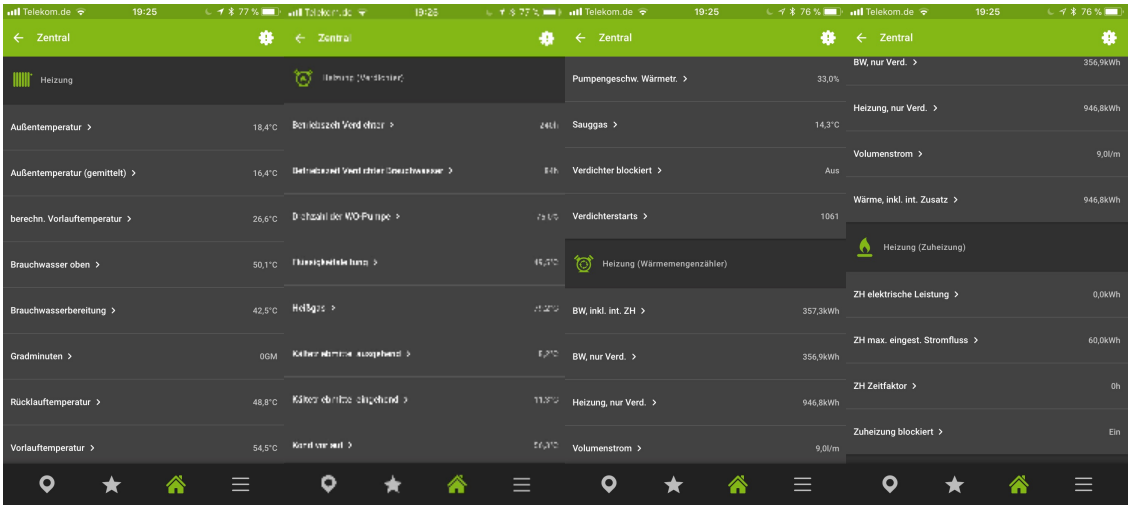
Ich fand es gelinde gesagt sehr ***unsmart***, dass meine Heizung und mein "Haus" nicht miteinander reden!
Nachdem ich es dann über die IFTTT-Integration des Nibe Uplinks bereits hinbekommen habe, dass Loxone seine Soll- und Ist-Temperaturen (Touch Tree) an die Heizung schickt, weiß diese ja schonmal, welche Vorlauftemperatur im Heizkreis nun sinnvoll wäre (Heizkurve war gestern).
Und über die Automatik und die eingestellte Außentemperatur (hier 18°C im 24h-Mittel) fängt sie beim Unterschreiten nun ja auch prima an, diese Temperatur auf den Heizkreis zu schicken.

Nur was hab ich davon, wenn im Haus/Loxone dann trotzdem noch die Sommerzeit herrscht und damit trotz Unterschreiten der Wunsch-Tenperatur von 23°C im Zimmer die FBH-Thermostate komplett zu lässt ;-) Dann hab ich eine Heizung die mich wegen Überdruck auf dem Heizkreis anschreit. :-)

Also war der Gedanke geboren, dass Loxone endlich smart damit umgeht und weg von den festen Zeiten die gleiche Logik fährt und bei Unterschreiten der besagten Außentemperatur auf den "Wintermodus" schaltet! Das hätte ich nun natürlich über den Wetter-Service lösen können, aber ob das so exakt auf die Berechnung der Heizung passt?
-> Also warum fragt mein Haus nicht einfach die Heizung nach dieser Außentemperatur?

Da ich aktuell leider keine Lust & Zeit hatte mit einen Modbus-Adapter zusammenzustricken (Hardware liegt da, aber ich bin Software-Entwickler und das würde wohl über den Winter dauern *g*), wollte ich die Nibe Uplink API dafür verwenden. Dumm war dort aber dann, dass diese oAuth2 verwendet -> also wurde ein passender Gateway nötig ... here we go!

Screenshots



Was wird benötigt

- RaspberryPI (oder ähnlich) mit Webserver + php
- Nibe Uplink (ich bin mir gerade nicht sicher, ob zwingend Premium nötig ist)
- Eine Nibe Heizung (hier eine Nibe F-1245)
- Loxone (nee ach...)

GitHub Projekt

https://github.com/DerFlash/Loxone-Nibe-Gateway

Anleitung

1. **RaspberryPI vorbereiten**
   Je nach Notwendigkeit installieren:
   # sudo apt-get install apache2 php5-curl

2. Gateway installieren
   Kopiere die Dateien aus dem Ordner raspberryPi aus dem GitHub Projekt ins Web-Server-Verzeichnis deines RaspberryPi.
   In meinem Fall liegen die Dateien index.php & token dann also in /var/www/html/nibe/
   Die Datei token muss vom Webserver geschrieben werden können:
   # chown www-data:www-data token
   # chmod 600 token

   Das Ganze ist dann beispielsweise im LAN wie folgt erreichbar: http://raspberrypi.fritz.box/nibe/
   Je nachdem, welche Adresse euer RaspberryPI in eurem Netzwerk hat, passt dies entsprechend an und versucht den Gateway einfach mal aufzurufen. Die URL benötigt ihr dann in Punkt 3.
   Es ist absolut ausreichend, dass diese Adresse nur von eurem internen Netz aufgerufen werden kann. Vielmehr rate ich sogar dazu.

3. **Nibe Uplink Vorbereiten**
   - Wenn du noch einen Nibe Uplink Account hast, erstelle einen auf https://www.nibeuplink.com
   - Log dich damit dann auf https://api.nibeuplink.com/Account/LogIn ein
   - Erstelle eine neue Application auf https://api.nibeuplink.com/Applications/Create
     Als Callback URL gibst du dabei die (lokale) Adresse zum Gateway ein (Beispiel aus Punkt 2: *http://raspberrypi.fritz.box/nibe/*)
     PS: Die Uplink-API muss NICHT von außen darauf zugreifen können!

4. **Gateway anpassen**
   Nun die index.php auf dem RaspberryPI wie folgt editieren:
   - **<nibe_api_client_id>** ersetzen durch den **Identifier** aus der erstellten Nibe Uplink Application
   - **<nibe_api_client_secret>** ersetzen durch das **Secret** aus der erstellten Nibe Uplink Application
   - ggf. auch hier wieder die Callback URL (**$REDIRECT_URL**) an eure Gegebenheiten anpassen

5. **Gateway verbinden**
   Sofern ihr alles korrekt erledigt habt, ruft ihr nun erneut den Gateway im LAN auf. Ihr werdet dabei aufgefordert die Bridge mit der API zu verbinden (auf "here" klicken).
   Dabei werdet ihr auf die Nibe Uplink API weiter geleitet, die euch dann um Erlaubnis bittet, dass euer Gateway auf die API zugreifen darf. Der Gateway bekommt dann schließlich einen Token zurück, mit dem er nun auf die Uplink API zugreifen darf.
   FYI: Das Ganze nennt sich eine "oAuth2 Autorisierung" und muss grundsätzlich nur einmalig erfolgen. Mit dem Token bzw. einem ebenfalls bekommenen Refresh-Token hält der Gateway ab jetzt quasi die Verbindung zur API aufrecht (fordert also nach Ablauf des Tokens mit dem Refresh-Token einen Neuen an)...

   Sofern alles funktioniert hat, landet ihr auf der Gateway-Statusseite. Dort findet ihr auch eine erste Test-Antwort von der Uplink API mit hoffentlich euren Status-Informationen zu eurer Heizung ;-)
   Etwas weiter unten findet ihr auch den Link zur Uplink Doku und könnt über das Eingabefeld Anfragen an die API senden.

6. **Fast geschafft - Abfragen mit Loxone**
   Jetzt kommt der klassische Teil: Die einzelnen Anfragen, die ihr über den Gateway nun jederzeit in Richtung Uplink API stellen könnt, liefern euch Antworten im JSON-Format. Daraus könnt ihr mit Hilfe von "Virtueller HTTP Eingang" und jeweiligen "Virtueller HTTP Eingang Befehl" in Loxone entsprechende Eingänge erstellen.

   Damit das Ganze flott geht, hab ich entsprechende Templates erstellt und ebenfalls im GitHub Projekt im Ordner "**Loxone Vorlagen**" hochgeladen.
   Diese XML Dateien kopiert ihr in den Ordner **C:\ProgramData\Loxone\Loxone Config 9.0\Templates\VirtualIn** (bei Windows 10 - je nach Windows-Version kann dies auch anders lauten).
   Danach solltet ihr die Eingänge bequem über den Menüpunkt "Vordefinierte HTTP-Geräte" importieren können.

   Achtung:
   Auch hier muss ggf. wieder die Adresse zum Gateway angepasst werden -> Parameter "**URL**" im jeweiligen Eingang!
   Was aber unbedingt angepasst werden muss, ist die **SystemID** (37034) innerhalb der **URL.** Diese findest du z.B. in der Test-Response der Gateway Statusseite.

Und nun viel Spaß damit!