eqiva Bluetooth Smart Türschlossantrieb

Intro

Die Firma eQ-3 AG produziert kostengünstige Bluetooth Türschlösser (siehe eq-3.de), welche mit FabAccess kompatibel gemacht werden können. Auf dieser Seite sind ein paar Keynotes beschrieben, wie das realisiert werden kann. Ein Ansprechpartner für die konkrete Umsetzung ist Joris Bijkerk vom Makerspace Bocholt.

Das Hinzufügen eines einfachen Türöffners, der in den Türrahmen eingebaut wird, kann mit dem Standardsetup leicht durchgeführt werden, da ein normaler Türöffner durch einen Shelly aktiviert werden kann und das lässt sich direkt in FabAccess einbinden.

In vielen Fällen ist das Modifizieren des Türrahmen keine Option, wenn man normaler Mieter ist und die Tür nicht zum Eigentum gehört. Eine Möglichkeit ist, ein intelligentes Schloss zu verwenden, das den Türschlüssel automatisch per Motor dreht - so zum Beispiel durch das Eqiva eQ3 Türschloss.

Native Inkompabilität mit MQTT - Alternative "keyble"

Leider ist das Schloss von Haus aus nicht MQTT-kompatibel und kann nicht direkt angebunden werden. Wir benötigen ein paar Umwege und nutzen dazu das Projekt keyble

, welches auf einem ESP32 oder Raspberry Pi installiert werden kann.

Die ESP32-Variante wird aus folgenden 3 Gründen nicht empfohlen:

- Für das Initialisieren des eQ3 Smartlocks benötigen wir ein Linux-System. Wir sparen uns Aufwand, wenn wir dies direkt mit einem Raspberry Pi durchführen, auf dem auch keyble installiert ist.
- Benutzer berichten, dass der ESP32 relativ instabil läuft
- eine Raspberry Pi Zero W Variante ist günstiger und bietet mehr Möglichkeiten

Setup auf einem Raspberry Pi Zero W

Das Einrichten eines Betriebssystems ist nicht Bestandteil dieser Dokumentation.

Nach erfolgreichem Einrichten installieren wir das Projekt keyble auf dem Pi. Dafür benötigen wir zunächst NodeJS. Das lässt sich sehr einfach mit n installieren (kein root-Zugriff nötig). Es wird automatisch eine NodeJS Version installiert.

```
curl -L https://bit.ly/n-install | bash
```

Danach installieren wir keyble:

```
npm install --update --global --unsafe-perm keyble
```

Dann richten wir noch ein paar Tweaks ein:

```
sudo setcap cap_net_raw+eip $(eval readlink -f `which node`)
```

Außerdem installieren wir Mosquitto, um MQTT Befehle später versenden zu können:

```
sudo apt -y install mosquitto-clients
```

Im Anschluss wird ein neuer Nutzer angelegt. Um ein eQ-3 eqiva Bluetooth Smart Lock tatsächlich steuern zu können, werden eine Benutzer-ID und der entsprechende 128-Bit-Benutzerschlüssel benötigt. Da die Original-App keine Möglichkeit bietet, diese Informationen zu erhalten, ist es notwendig, zuerst einen neuen Benutzer zu registrieren, indem man die Informationen verwendet, die im QR-Code der „Key Card“, die mit dem Schloss geliefert wird, verschlüsselt sind.

```
keyble-registeruser -n John -q M0123456789ABK0123456789ABCDEF0123456789ABCDEFNEQ1234567
```

Dann halten wir „Unlock“-Taste gedrückt, bis das gelbe Licht blinkt, um in den Kopplungsmodus zu gelangen.

Output-Beispiel:

```
Press and hold "Unlock" button until the yellow light flashes in order to enter pairing mode
Registering user on Smart Lock with address "01:23:56:67:89:ab", card key
"0123456789abcdef0123456789abcdef" and serial "NEQ1234567"...
User registered. Use arguments: --address 01:23:56:67:89:ab --user_id 1 --user_key
ca78ad9b96131414359e5e7cecfd7f9e
Setting user name to "John"...
User name changed, finished registering user.
```

Im Anschluss können wir das Schloss per Befehlszeile nutzen:

```
keyble-sendcommand --address 01:23:56:67:89:ab --user_id 1 --user_key
0123456789abcdef0123456789abcdef --command open
```

Beim Arbeiten mit MQTT-Befehlen im Teil der Beschreibung („Piping data into keyble-sendcommand“) ist einige Aufmerksamkeit erforderlich, was das Arbeiten mit Hochkommas angeht: `` `"door_lock/action"` `` macht teilweise auf der Kommandozeile Problem, wogegen `` `door_lock/action` `` gut funktioniert.

Nachdem das System wie beschrieben eingerichtet wurde, können wir es auch mit Mosquitto testen:

```
mosquitto_pub -h 192.168.177.3 -m door_lock/status "open"
```

Anbindung an FabAccess via Actor (Python)

Das Script ist nur so konzipiert, dass es die Tür durch den Befehl öffnet, aber **nicht** nach ein paar Sekunden automatisch wieder verschließt, sodass Nutzer eintreten können und aber nicht permanent die Tür offen steht. Dieses Verhalten könnte entweder zusätzlich im Script eingebaut oder direkt im Schloss konfiguriert werden.

Dafür klonen wir https://gitlab.com/fabinfra/fabaccess/actors/eq3-eqiva-smartlock:

```
mkdir -p /opt/fabinfra/adapters/
cd /opt/fabinfra/adapters/
git clone https://gitlab.com/fabinfra/fabaccess/actors/eq3-eqiva-smartlock.git
```

```
cd /opt/fabinfra/adapters/eq3-eqiva-smartlock/
python3 -m venv env
. env/bin/activate #activate venv
pip3 install -r requirements.txt
```

> Im Quellcode muss manuell noch der richtige `hostname` angegeben werden.

Konfiguration in bffh.dhall

Im Anschluss fügen wir das Schloss in die Hauptkonfiguration ein. Folgende Snippets dienen dazu und müssen jedoch individuell angepasst werden. Wir empfehlen dafür auch die Nutzung des Konfigurationsgenerators.

> **Konzeptionelle Hinweise:**
>
> Es ist wichtig, dass alle Benutzer über Admin- bzw. Verwaltungsrechte verfügen, da die Aufforderung zum Öffnen der Tür durch einen Benutzer dazu führt, dass die Tür von diesem Benutzer "in Gebrauch" (`In Use`) ist. Die Tür kann nur wieder aktiviert werden, wenn der vorherige Benutzer die Tür "ent-benutzt" (`GIVEBACK`) oder wenn ein anderer Benutzer die Tür "zwangsbefreien" (`FREE MACHINE`) kann, bevor er sie selbst benutzt.
>
> In diesem speziellen Fall, in dem alle Benutzer Admin-Fähigkeiten benötigen, könnte die Rolle auch nur die Berechtigung lab.door.use enthalten und alle dem Rechner zugewiesenen Berechtigungen (`disclose`, `manage`, `read`, `write`) würden einfach mit `doorrolle.use` übereinstimmen (z.B. `disclose = "doorrool.use"`).

roles

```
roles = {
...
    doorrole = {
     permissions = [
         "doorrole.read",
         "doorrole.write",
         "doorrole.disclose",
         "doorrole.manage"
         ]
```

```
    },
...
```

machines

```
machines = {
...
    Willkommen = {
        category = "Management",
        name = "Aufschließen",
        disclose = "admin.disclose",
        manage = "doorrole.manage",
        read = "doorrole.read",
        write = "doorrole.write"
    },
...
```

actors

```
actors = {
...
    OpenTheDoor = {
    module = "Process",
        params = {
             cmd = "/opt/fabinfra/adapters/eq3-eqiva-smartlock/env/bin/python3",
             args = "/opt/fabinfra/adapters/eq3-eqiva-smartlock/eq3-eqiva-smartlock.py -vvv",
        }
    }
...
```

actor_connections

```
actor_connections = [
...
    { machine = "Willkommen", actor = "OpenTheDoor" },
...
```

Version #20
Erstellt: 21 Oktober 2024 16:30:16 von Mario Voigt (Stadtfabrikanten e.V.)
Zuletzt aktualisiert: 21 Februar 2025 00:48:22 von Mario Voigt (Stadtfabrikanten e.V.)