

Türschlösser (Doors)

Um mit FabAccess Türen steuern zu können, ist es wichtig, bei der Auswahl der Aktoren darauf zu achten, dass es zu dem Zustand kommen kann, dass die Tür noch geöffnet ist, jedoch das Schloss verschlossen ist. Dieses Szenario kann mit geeigneten Sensoren wie Türkontakten behoben werden.

Zur Steuerung kann entweder der [Trait "Doorable"](#) oder "[Lockers](#)" verwendet werden, je nachdem, wie die Tür Rückmeldung über ihren Zustand geben kann.

- [dormakaba c-lever pro](#)
- [eqiva Bluetooth Smart Türschlossantrieb](#)
- [Nuki Smartlocks](#)
- [dormakaba Selbstverriegelndes Panik-Schaltschloss SVP6000](#)
- [Sammlung von Systemen](#)

dormakaba c-lever pro



- läuft über KeyBLE mit LAN-Implementierung. Siehe <https://t.me/c/1401139456/1001>
- <https://oyoooyo.github.io/keyble>
- <https://github.com/lumokitho/esp32-keyble>
- Das ganze ist (oder war bei mir) ein mehrstufiger Prozess:
 1. Zuerst musste ich unter Linux auf einen RasPi Zero W das Schloss initiieren und ein Nutzer anmelden. Dieser Nutzer bekommt dann einen Schlüssel. Das ganze habe ich einen RasPi Zero W gemacht, weil ich Bluetooth nicht auf einen anderen RasPi zum laufen gebracht habe.
 2. Mit dem Nutzer und Schlüssel kann man dann ein ESP32 mit esp32-keyble programmieren. Diesen meldet man dann in seinem WLAN und bei MQTT Broker an. Der ESP32 wechselt dann kontinuierlich zwischen WLAN und Bluetooth hin und her und Kommuniziert auf diese Weise mit Türschloss und WLAN.
 3. die Anbindung in FabAccess läuft dann über ein Skript-Actor (wie für Tasmota Schalter). Da FabAccess es aktuell nur einem Nutzer erlaubt eine Ressource zu buchen, haben wir für das Öffnen der Tür, einschalten des Hauptschalters und ausschalten der Alarmanlage eine Ressource namens "Tür auf" definiert. Dieser kann von jedem (freigeschalteten) Nutzer übernommen werden und erneut gebucht werden (bei "giveback" oder "free Machine" passiert also nichts). Eine zweite Ressource "Licht aus" schaltet den Hauptschalter aus und die Alarmanlage ein. Dieser sollte wirklich nur vom letzten Nutzer aktiviert werden, der den Makerspace verlässt.
 4. Da unser WLAN manchmal Probleme hatte, haben wir den ESP32 Code für einen Ethernet Anbindung über einen enc28j60 LAN Modul angepasst.

5. Da die Bluetooth Kommunikation zwischen Schloss und ESP32 nicht immer stabil funktioniert hängt sich der ESP32 ganz selten mal auf (ca. 1x im Monat). Das haben wir durch ein "reset Tür" Aktor gelöst, der den ESP32 für 5 Sekunden ausschaltet.
- Joseph: "Wir haben mit Dormakaba C-Lever Pro angefangen und mit Matrix One angesteuert. "

- “MATRIX ONE ist die einfache, sichere und webbasierte Zutrittslösung, die perfekt auf die Bedürfnisse kleiner und mittelständischer Unternehmen angepasst ist. Sie profitieren jederzeit von einem rundum einfachen System, sowohl in Bezug auf die Installation als auch die Bedienung: Innerhalb kürzester Zeit ist die Lösung ausgerollt und die moderne Oberfläche lässt sich intuitiv bedienen.

- https://www.dormakaba.com/de-de/angebot/produkte/systemloesungen-zutritt-und-zeit/zutrittskontrollloesungen-fuer-kleine-und-mittlere-unternehmen/matrix-one--ka_500011

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](https://makerspace-bocholt.de).

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](https://keyble.de)

, 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
ca78ad9b96131414359e5e7cecf7f9e
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 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" },  
    ...
```


Nuki Smartlocks



Intro

Mit Nuki Türschlössern können bestehende Türen einfach umgerüstet werden. Die Ansteuerung kann dann je nach Hardware-Revision des Schlosses direkt oder indirekt über MQTT erfolgen. Außerdem können die Schlösser mit den Nuki Türsensoren kombiniert werden.

Für die Ansteuerung mit MQTT muss entweder ein ...

- Smart Lock Ultra,
- Smart Locks (4. Generation),
- Smart Lock 3.0 Pro

... verwendet werden, das direkt über WLAN gesteuert werden kann, oder ...

- Nuki mit [Bridge](#) (Gateway)

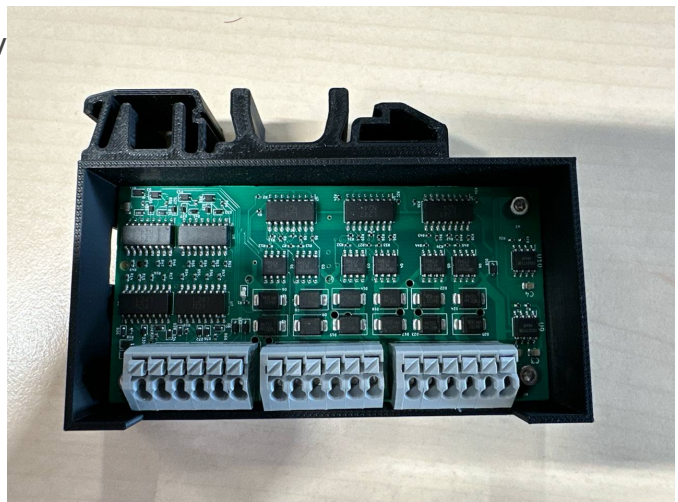
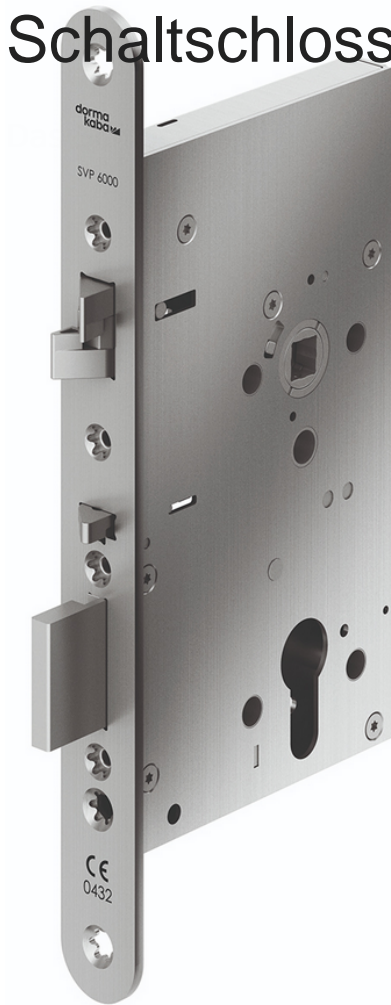
Ältere Nukis mit Bridge

Hierfür kann <https://github.com/bluewalk/NukiBridge2Mqtt> genutzt werden. Dieses Projekt macht auch ältere Schlösser MQTT-kompatibel und damit FabAccess-kompatibel.

Nuki mit TLS-Unterstützung

Nuki Smart Lock Pro können kein MQTT mit TLS ☐☐

dormakaba Selbstverriegelndes Panik-Schloss SVP6000



Unsere Steuerplatine, welche per USB an einem Raspi hängen (ein oder mehrere). Steuern Türen und Schütze mit 12-24V an. Mehr Details gibt's bei [@chca42](#) der sie Entworfen und fertigen lassen hat.

Es gibt die Software dahinter auf github, aber eine ordentliche Doku zum nachbauen noch nicht. Platinen layouts auch noch nicht, aber bei Interesse hilft [@chca42](#) vermutlich weiter.

Ist darauf ausgelegt bei JLCPCB mit Bestückung gefertigt zu werden, im Ordner gerber liegen die Fertigungsdaten dafür. Features sind:

- Galvanische Trennung zwischen USB und Ein-/Ausgängen
- 12-24 V Spannung an Ein-/Ausgängen (externe Versorgung)
- 8 Eingänge, 6 Ausgänge (bis 3A), Schalten induktiver Lasten (Motor/Relais/Schnapper/etc.) ist möglich
- unterstützt wird bei Eingängen die Erkennung von Versorgungsspannung, Masse und hochohmigem Eingang, bei Ausgängen Schalten gegen Masse, Versorgung und hochohmiger Zustand (Halbbrücke)

“ Wir haben SVP 6000 Schlösser von Dormakaba im Einsatz mit eigener Ansteuerungsplatine. Funktioniert seit 2 Jahren sehr zuverlässig. Die Schlösser sind mit 700-800€ nicht günstig und brauchen auch einen speziellen Panikbeschlag. Dafür dann aber auch an Fluchtwegtüren zugelassen.

https://github.com/zam-haus/door_pi

Sammlung von Systemen

- <https://wiki.maglab.space/de/Hackspace/Zugang>
- <https://git.mittelab.org/proj/keycard-access>
- <https://github.com/fablab-luenen/Tueroeffner>