15.07.2021 - Meta-Pad Zugangskontrollsysteme f. Offene Werkstätten

Meta-Pad Zugangskontrollsysteme f. Offene Werkstätten

Hier entsteht ein Zugangssystem, das, basierend auf den verschiedenen Ansätzen und gemachten Erfahrungen verschiedener Offener Werkstätten, eine möglichst breite Nutzbarkeit ermöglichen soll und Open Source ist. Ziel ist die Entwicklung möglichst objektbasierter und generischer Schnittstellen, die die diversen Bedürfnisse der Werkstätten abdecken und bestehende Systeme integrieren.

Parallel werden bereits bestehende Systeme gesammelt und ein Überblick über diese Systeme erstellt, um die daraus gewonnenen Erkenntnisse sinnvoll in das entstehende System einfließen zu lassen.

- In Offenen Werkstätten
  - FabLab L’Aquila
  - Google makerspace-auth
  - Dresden (Roseguarden)
  - München (FabLab)
  - MakerSpace Graz (“LEASIE”)
  - Capstone (ergänzt am 29.11.2019)
- In Hackerspaces
  - LgHS (Liege Hackerspace, Belgien) (ergänzt am 29.11.2019)
  - Flipdot e.V. Kassel (ergänzt am 29.11.2019)
  - TH Köln
  - Frankreich
  - Italien
  - Coredump (Hackerspace Rapperswil-Jona, Schweiz) (ergänzt am 26.06.2020)
- Alternativen
  - open-taffeta
  - Commons Booking
- Brainstorming - Pad(s) / Arbeitspads

Timeline - Entwurf

- Leute einsammeln bis 23.11. (wer danach kommt, kommt halt danach - muss damit leben, dass die anderen schon los gelaufen sind)
- Core-Entwickler entscheiden sich bis zur ersten Mumble-Session ob sie einer sein wollen
  - erste Mumble-Session: https://dudle.inf.tu-dresden.de/-nkSQJu6-Q/
- Beim ersten Mumble-Treffen:
  - Vorstellen
  - 36C3-Umfrage
  - Grundlagenfragen zu Infrastruktur klaeren

    - Wie viel von derzeitigen Systemen uebernehmen / wie viel backwards-Kompatibilität können / wollen wir
    - Was sind MUSS / SOLL / KANN Randbedingungen bzw. explizit verwendete Standards?
    - Versionierung / Sprache / Aufbau / …
    - Was soll bis März angepeilt werden, was wird evtl. später integriert, wo grenzen wir ab?
  - Termine für stetige, regelmäßige Treffen (Mumble)
  - Termin & Ort für Hackathon(s)
  - Was passiert bis zum Congress
- Treffen auf dem 36C3 - wer ist da und Assembly / Space festlegen
- Hack-Wochenende Ende Januar (nach 22.01.) mit VoW (primär Core-Entwickler)
- Zum 01.02.2020 erste Version vom Modul-API/-System einsatzbereit
  - Ab da können sinnvoll Module fuer FabLabs/Makerspaces gebaut werden
  - Erster Fokus auf die meistverbreiteten Anwendungsfaelle (Mensakarte + evtl. Pin, ESP8266, etc => Rausfinden!)
  - Erste Module als Test
- Hack-Wochenende (nach 22.02.) mit VoW (incl. Modul-Entwickler)
- Deadline für V1.0 am 01.03. (falls finale Deadline am 31.03., sonst früher).
  - Ab hier gelten dann bestimmte (noch festzulegende) Garantien fuer Stabilitaet

Anforderungsliste / Ziele / Funktionen

Das (ideale) System soll…

- Nutzer (nur) mit Einweisung & Zugangsberechtigung unkompliziert die Arbeit an Maschinen ermöglichen. Maschinen stromlos schalten, so lange sie nicht von jemand mit entsprechenden Voraussetzungen bedient werden.
- Manche Geräte qualifizierter Überwachen und aktivieren / deaktivieren / verwalten.
- Qualifikation der Nutzer in der Relation “Nutzer - Maschine” abbilden und bei der Zugangskontrolle berücksichtigen.
  - z.B. in 5 Stufen: [-1] explizit gesperrt [0] darf zusehen [1] darf unter Aufsicht und Anleitung arbeiten [2] darf selbständig arbeiten [3] kann einrichten und einstellen [4] kann warten / instandsetzen [5] kann schulen & einweisen
- Auf abgelaufene Einweisungen hinweisen & einfordern.
- Dabei helfen, Öffnungszeiten durchzusetzen.
  - z.B. 30 min. vor Feierabend im Display herunterzählen und bestimmte Maschinen zum Feierabend hart abstellen.

    - Nicht alle Maschinen!! - 3D-Drucker z.B. laufen auch nachts.
- Türen öffnen? - (Diskussionspunkt)
- Einfach installierbar sein und keine besonderen Anforderungen an Hardware / Netzwerkumgebung stellen. (RaspberryPi in der Ecke sollte im Notfall als zentrale Hardware reichen)
    - Aus Grynden der Stabilität und Datensicherheit ist natürlich ein “ordentlicher” und gut eingerichteter Server besser & ab einer gewissen Größe / Professionalität des Labs ratsam. Die Doku muss das abdecken.

Das System muss modular aufgebaut und flexibel (einfach!) einricht- anpass- und erweiterbar sein, um auf lokal verschiedene Anforderungen eingehen zu können.

Bereits laufende Aktivitäten

an Hochschulen

Fab Lab Siegen (David Amend)

- fablab-siegen.de
- Github: https://github.com/FabLabSiegen/FabLabAccessControl
- “Key-Features”
    - Switch plugwise plugs on and off via RFID chips
    - Access control via ESP8266 and RFID-reader
    - Device management of plugwise devices, RFID chips and ESP8266 via webinterface
    - Usermanagement via webinterface
    - Communication between ESPs and backend via MQTT

BHT Berlin

qrhello (niedrigschwelliges, einfach zu “überlistendes” QR-Code-System)

- https://github.com/dequbed/qrhello

- Nutzerschnittstelle: QR-Etiketten mit (lokaler) URL
- Komponenten:

    - Flask / Python / UWSGI / Apache-Server
    - (noch) SQLite-DB für Einträge
    - Schnittstelle zu PostgreSQL-Datenbank von Leihs (ZHDK) Leihs
    - Verlinkt zu einem Labor-Wiki (Media-Wiki) mit Infos und Bedienungsanleitungen.

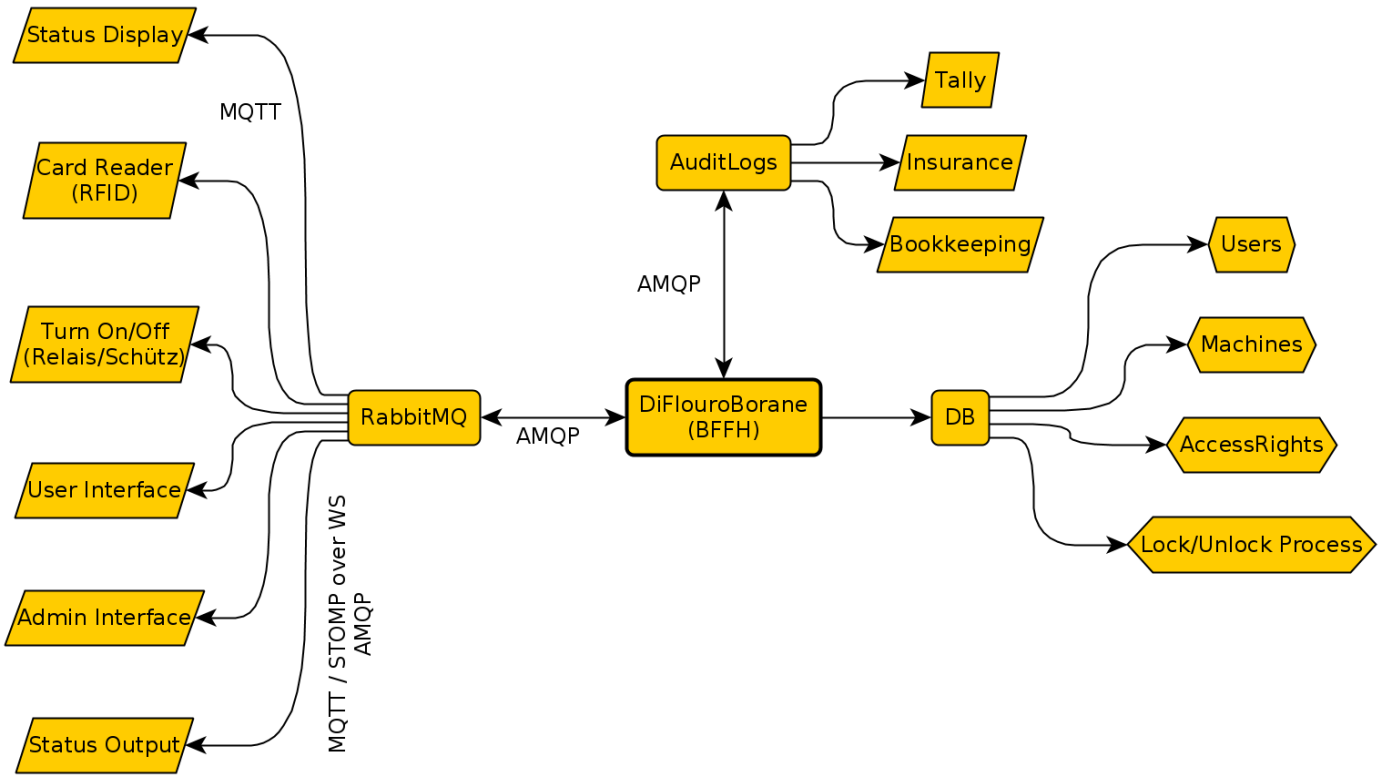
Keine Einrichtung von einzelnen Maschinen notwendig - Etikett ausdrucken, draufkleben - that’s it.
Die Nummerierung und Bezeichnung der Maschinen wird (falls dort hinterlegt) aus Leihs übernommen.
Aktuell keine Logik für Zugangskontrolle und / oder Maschinenschaltung integriert.

card2go (RFID-gesichertes System mit Datenbank-Backend)

- noch private (https://github.com/HazeCore/Card2Go/)
- in GO geschrieben.
- Middleware-Server ohne grafischem Front-/Backend … bildet “Geschäftslogik” und Zugangskontrolle mit den o.a. 5 Berechtigungsstufen im FabLab ab.
- Etliche Edge-Cases im FabLab-Betrieb abgebildet.
    - Maschinenabnahme durch Lab-Betreuer nach Nutzung
    - Unkomplizierte Übergabe einer Maschine machbar
    - Signalisieren / Ausschalten zum Feierabend
        - laufenlassen bestimmter Maschinen (z.B. 3D-Drucker)
    - Etliche Fehler bei Verbindungsverlusten zu Server / DB abgefangen.
    - Zwangs-Einschalten / -Ausschalten mit Master-RFID-Karten
    - E-Check v. Maschinen hinterlegbar
    - Manche Maschinen (z.B. Tische / Werkbänke) brauchen keine Einweisung / keinen E-Check
    - Display-Steuerung (0,96" OLED-Display hard-coded)
    - …

Diflouroborane (Konzeptprototyp fuer ein von Anfang an fuer weiten Einsatz gebautes System)

Stukturkonzept:



Kurzbeschreibung in ~~Fliesstext~~ Stichpunkten:

- Zentrales RabbitMQ spricht MQTT, STOMP und AMQP, erstere zwei auch ueber WebSockets (z.b. fuer Webapplikation hilfreich)
- N-Step Authentication um flexibel genug zu sein um jeden Typ Authentifizierung zu implementieren.
- Diflouroborane (kurz BF_2H) ist eine relativ monolitische Applikation die die eigentliche Logik bzw. die im FabLab-Betrieb auftretenden “Geschäftsprozesse” implementiert.
  - Auch wenn es ein Monolith ist wird BF_2H darauf ausgelegt modular und anpassbar genug zu sein um mit allen denkbaren Steuersystemen interagieren zu koennen und agnostisch gegenueber Authentifizierungs- bzw. Authorisierungssytemen zu sein.
- `Auditlogs` ist ein getrenntes System was ueber AMQP ueber Verwendung informiert wird
  - Anwendungsfall sind speziell Labore die Verwendung/Material/o.ae. abrechnen muessen/wollen oder die wegen Vorschriften oder Versicherung genauere Buchfuehrung ueber Anwesenheit/Verwendung haben muessen.
- Ein Web-/Graphisches-UI ist im Moment nicht Teil des Konzepts weil dequbed wenig Erfahrung damit hat. Vorschlaege sind gerne gesehen!

Das System ist ein frueher Prototyp und noch stark in Flux, deswegen sei fuer genauere Architektonische Details so wie den Code derzeit auf GitHub verwiesen.

RFID-Terminal

Kleiner Prototyp als Kombination verschiedener Gieß-Materialien und -Verfahren (Epoxid, PU, Acryl).
Display eingebettet, RFID-Spule eingebettet, Formteil für RFID-Karten & Fobs als negativ-Gießteil.

Verbesserungsfähiger Demonstrator / Mock-Up bei Tasso --> wer’s weiterentwickeln will, kann vorbei kommen und übernehmen.

ESP-Schütz

Für Drehstrom-Maschinen (Drehbank / Fräsen / …)
Kleine Ergänzungs-Platine, auf oder neben Schütz montierbar.
Bekommt Kommandos über WLAN und MQTT (z.B. aus OpenHab)
Schaltet die Schütz-Spule über ein SolidStateRelay (z.B. Panasonic AQH2213).
Das Schütz wiederum schaltet die Stromzufuhr zur Maschine.
Wichtig: Wiederanlaufschutz direkt an der Maschine vorsehen! - nicht “einfach so mal schnell” ein Schütz reinklemmen. Andernfalls kann schnell mal eine Drehbank “remote” eingeschaltet werden, an die sich jemand gerade “nur mal angelehnt” hatte --> wär blöd!
Auch nichts für “Niedervolt”-Anwender! - aktuell als Eingriff in die Zuleitung zur Maschine realisiert und damit hart am Regelungsbereich der Maschinenrichtilinie. Als Lösung evtl. als “Zwischen-Schaltgerät”, in jedem Fall aber **Netzspannung** und nur was für “VDE-Kundige” bzw. mit einiger Aufmerksamkeit anzugehen.

Mock-up existiert bei Tasso an der Drehbank --> wer’s weiterentwickeln will, kann vorbei kommen und übernehmen.

Düsseldorf (Fabian Meyer)

- https://fabaccess.bian-meyer.de/
  Zugangssystem zum Schalten von Steckdosen entstanden an der Hochschule Ruhr-West. Momentan in Weiterentwicklung mit dem Verbund offener Werkstätten

- **Eigenschaften**

    - Authentifizieren des Benutzer und Buchen/Schalten einer Maschine über Android/(iOS) App mit RFID Karte an NFC fähigem Tablet
    - odoo (ehemals openERP OpenSoruce ERP-System) als Nutzerdatenbank und zur Buchführung für die Benutzung von Geräten
        - Nach Beendigung der Buchung wird eine Rechnung in odoo erstellt
        - Buchung einer Maschine nur möglich, wenn Sicherheitseinweisung stattgefunden hat
    - Abstraktion der Schaltbaren Hardware (Steckdosen) durch openHAB
        - durch openHAB einheitliche Schnittstelle zum schalten von Geräten unterschiedlicher Hersteller/Protokolle
            - Getestet mit zWave Steckdosen
- **Bestandteile**
    - odoo
    - openHAB
    - **App:** Stellt schaltbare Geräte an einem konfigurierten Ort visuell dar und erlaubt dem Nutzer sich per RFID Tag am Gerät zu authentifizieren und Geräte zu buchen/schalten (Und die Buchung nach erneuter Authentifizierung wieder zu Beenden)
        - GitHub: https://github.com/faaaaabi/fablab-access-app
        - Sprache/Frameworks: ReactNative, JavaScript, Redux, …
        - Status: Prototyp, Authentifizierungsfunktion und Buchen von Geräten funktional.
    - **API-Gateway:** Bietet REST-API (Websocket für Realtime-Kommunikation ist vorbereitet) für die App an und enthält Businesslogik welche die Funktionalität über odoo und openHAB abbildet. Zugrifssgeräte authentifizieren sich über API Key und erhalten JWT-Token, welches bei allen witeren Request mitgeschickt wird. Bei der der Authentifizierung einse Benutzers am Zugriffsgerät wird ein Intermediate-Token erzeugt, welches nur 20 Sekunden gültig ist und nur für das Buchen eines Gerätes. Dies soll Replay-Attacken verhindern. Für die Datenhaltung wird momentan MongoDB verwendet. Das fühlt sich für die Entwicklung ganz gut an, da kein striktes Schema für die Entitäten vorhanden (Diese quasi noch entsteht). Architekturell lässt sich für Datenhaltung aber auch relativ schmerzfrei ein ander DBMS einsetzen
        - Architektur: https://github.com/faaaaabi/fablab-api-gateway#architecture
        - GitHub: https://github.com/faaaaabi/fablab-api-gateway
        - Sprache/Frameworks: Node.js, TypeScript, Express, MongoDB Passport (Authentifizierung), …

    - Status: Prototyp, kein Installationsscript oder docker-compose file vorhanden, kein Webinterface vorhanden. Grundfunktionalität wie - Authentifizierung der Zugriffsgeräte (in diesem Konzept die App) und User, das Buchen von Geräten Und damit verbunden das Schalten von an openHAB angebunden z-Wave Steckdosen, das Erzeugen von Rechnungen - funktional. Für das Anlegen von anderen Resourcen (Maschinen, Zugriffsgeräte, Orte) fehlen noch REST-Endpoints, das läuft bisher manuell in der Datenbank. Die Berechtigungsverwaltung ist bisher sehr rudimentär und dekt nur einen binären Wert ab, ob eine globale Sicherheitseinweisung stattgefunden hat. Vorhandene Teile der REST-API sind dokumentiert
  - **Admin-Interface:**
    - GitHub: https://github.com/faaaaabi/fabaccess-admin-interface
    - Sprache/Frameworks: React, TypeScript, Redux, ...
    - Im Internetz: https://fabaccess.bian-meyer.de/
    - Status: Clickdummy. Bildet ein flexibleres Berechtigungssystem ab.
- **Key Features**:
  - Kein Hardware-Selbstbau notwendig
    - Benutzersteuerung und Authentifizierung wird über Android/iOS abgebildet. App flexibel anpassbar, kein Hardwareumbau notwendig.
  - Durch openHAB flexibel in der Wahl eingesetzter (Schalt)-Hardware
  - Netzspannung schalten mit zertifizierten Geräten von der Stange
  - Kombination mit odoo
- **Derzeit in Entwicklung**
  - Webinterface zur Administration. Clickdummy: https://fabaccess.bian-meyer.de/
- **Demnächst in Entwicklung**
  - Abbildung eines einfachen Berechtigungssystems
    - Benutzer a hat Berechtigungen b,c,d – Maschine e benötigt Berechtigung b und c
  - Anbindung von Selbstbau-Hardware (Türschlösser etc.)
  - Entwicklung einer App, welche der Nutzer auf seinem eigenen Smartphone installiert. Diese soll das Buchen von Geräten ermöglichen, indem ein angebrachter QR-Code gescannt wird

---

Kunsthochschule Halle

- Fa. Rutte (https://rutte.de/)

ETH Zürich

acos: https://sph.ethz.ch/acos_project/

- Poster: https://sph.ethz.ch/wp-content/uploads/2017/09/Handout_final.pdf
- Authentifizierung an (zentralem) Terminal mit RFID Token
- Buchung von Maschinen und Verbrauchsmaterialien am Terminal
- Schaltung Strom/Öffnen von Schlössern über ESP 8266 und Relay/Türschlosse
- Letzte Aktualisierung der Infopage im Oktober 2018
- Software auf dem Terminal
  - C++, Qt, QML
- **Kein Source Code verfügbar**

In Offenen Werkstätten

https://cowiki.offene-werkstaetten.org/uploads/2017-06-23_11-31-15_Zugangssysteme Überblick (WS).pdf

FabLab L'Aquila

https://github.com/FabLabAQ/LabAccessControl

Google makerspace-auth

- https://github.com/google/makerspace-auth

Dresden (Roseguarden)

- Nutzerverwaltung über Web
- Zugangskontrolle mit unterschiedlichen Räumen
- Verifikation mit RFID
- basierend auf Raspberry Pi
- Github: https://github.com/mdrobisch/roseguarden

Neue Version in Entwicklung seit 01/2019

- Details siehe https://pad.gwdg.de/v-xVnpWQREmBGuLG_hYTfQ?both
- Repositories (v2): https://gitlab.com/roseguarden
- Demo: https://roseguarden.fabba.space/dashboard

München (FabLab)

- Nutzeverwaltung über Web
- Zugangskontrolle Räume und Geräte
- RFID-basiert
- Github
  - Backend: https://github.com/sschaeffner/fabXaccess
  - Frontend: https://github.com/sschaeffner/fabXdashboard
  - Client für ESP32/M5Stack: https://github.com/sschaeffner/fabXclient

---

MakerSpace Graz (“LEASIE”)

Zitat: "Grundidee/Basistechnologie:

Jedes zu vermietende Gerät wird mit einem IOT Gerät, basierend auf Open-Hardware, ausgestattet, dass über WLAN direkt oder indirekt über einen Cloudserver (aktuell) mit der Blockchain kommuniziert. Diese IOT Geräte regeln den Zugang und messen mit unterschiedlichen Sensoren (aktuell Strom) den Zustand des Gerätes. Zwei Varianten: Eine simple „Zwischenstecker“ Lösung und eine erweiterte (aktuell) Variante für größere Maschinen. Beispiel Lasercutter: Es wird das Gerät nicht abrupt abgeschalten, sondern zuerst der Laser. Der Zugang wird über RFID Tags (aktuell) oder NFC (Handy) überprüft. Der Server ist ein Full Node und macht die Kommunikation mit der Blockchain sowie die Abrechnung. Abgerechnet wird 2 stufig: 1 Rechnung geht vom Provider (der der den Platz für die Vermietung stellt) an den Kunden. Eine 2te Rechnung wird für den Maschinenbesitzer an den Provider ausgestellt. Provider erhält eine Fee, Auch wird in eine gemeinsame Maschinenbruchversicherung eingezahlt. Vision: Wenn zB. viele FabLabs oder MakerSpaces an dem System hängen würden, wäre das ein brauchbarer Versicherungstopf. Dapp läuft Clientseitig und erledigt den Zahlungsverkehr, die Vermietung, die Stornierung und die Fehlermeldungen. Später soll noch eine Admin DApp geben, die die Kundenanlage, Maschinenanlage und Fehlermeldungen behandeln kann 2 Smart Contracts: 1 Renting und ein Invoice Contract. Möglichkeit einen eigenen gemeinsamen Tokens, ähnlich eines Gutscheinssystem mit Insentivemöglichkeiten zu verwenden (aktuell) oder an einen Token mit z.B. Eurobindung anzuhängen. Wenn der

nächste Mieter keinen Fehler meldet, rechnet das System die tatsächlich verwendete Zeit und Strom ab und überweist den Diffbetrag inkl. Kaution retour. Bei Fehlermeldung wird die Kaution eingefroren, bis der Provider den Fehler mit dem Maschinenbesitzer klärt und die Fehlerkosten erhebt. Gleichzeitig wird die Maschine auch für weitere Buchungen gesperrt. Weitere geplante größere ToDo's: Admin DApp, DATEV Schnittstelle zur direkten Verbuchung der Rechnungen (Buchhaltung) Ergänzend: Grundsätzlich geht es nicht nur um Maschinen, sondern generell um Ressourcen. z.b. wird darüber auch der Zugang zu den Räumlichkeiten freigeschalten. Weiterer grösser angedachter Entwicklungschritt: diese Ressourcen handelbar machen und eine allgemeine Cloud Plattform dafür zur Verfügung zu stellen (Bsp. Nachbar vermietet Rasenmäher). Ergänzend zum Ressourcenhandel: Wenn du die Ressource gemietet hast und jemand anderer sie dringender braucht, sollte sie dir automatisch zu einem höheren Preis abkaufbar sein. Generell: Für FabLabs, MakerSpaces sollte dieses ganze Konzept völlig OpenSource sein. Jeder soll sich z.B. frei entscheiden können, ob er einen eigenen Token auflegt oder die Vorteile des gemeinsamen Tokens nutzen will. Auch ob er die Hardware selbst bastelt (OpenHardware) etc. Umsatz soll über einen Shop und einem Cloudservice gemacht werden. Jedoch vorrangig über Eröffnung neuer Kundenkanäle wie z.b. Gewerbekunden oder auch Privatkunden über ein Portal, sowie professional Features wie Buchhaltungsschnittstellen, etc."

- http://makerspace.at/news/blockchain-mietsystem-leasie-gelauncht
- https://leasie.makerspace.at

Capstone (ergänzt am 29.11.2019)

- https://fabacademy.org/2019/labs/echofab/students/philippe-libioulle/capstone.html

In Hackerspaces

LgHS (Liege Hackerspace, Belgien) (ergänzt am 29.11.2019)

- https://www.lghs.be
- https://github.com/LgHS/sign-in

Flipdot e.V. Kassel (ergänzt am 29.11.2019)

- https://flipdot.org/blog
- https://github.com/flipdot/gsm-door

TH Köln

- Kombination mit Ilias

Frankreich

- https://www.fab-manager.com/?lang=en
  - https://github.com/sleede/fab-manager
- https://gitlab.com/fabaccess

Italien

- https://github.com/FabLabAQ/LabAccessControl

Coredump (Hackerspace Rapperswil-Jona, Schweiz) (ergänzt am 26.06.2020)

- Inventar: https://interna.coredump.ch/inventory/ (Source)

[Projektbeschreibung - MakeThings - Zugangs- & Verwaltungstool für Spaces]

Alternativen

FabMan (https://fabman.io/)

- zu teuer
- nicht OpenSource

---

open-taffeta

Offener Türöffner mit mobile app. Ausgelegt um an Gegensprechanlagen den Türöffner auszulösen und ein Klingeln zu verhindern wenn man sich selbst reinlässt.

- https://github.com/apiraino/open-taffeta
- https://github.com/apiraino/open-taffeta-mobile-app

Commons Booking

Verleihsystem für Lastenräder.

- https://prototypefund.de/project/commons-booking/

Brainstorming - Pad(s) / Arbeitspads

https://hazeco.re/codimd/7HkCBkSBRG6sYvT65wMPFg#

[OpenAccess Sytem Dokument zum Verbundstreffen]

[Mumble 26.11.19]

[Roseguarden]

[DiFlourBorane]

[Dudle für Sprachkonferenz 2]

[Agenda 09.12.2019]

[Protokoll 09.12.2019]

[Materialsammlung 29.12.2019 @36C3]

[Hackathon am 06.02.2020]

[Jitsi-Call am 18.02.2020]

[Diskussion in der Telegram-Gruppe des VoW am 25.02.]

[Treffen in der BHT am 04.03.2020]

[Call zu Werkstatt-Zugangssystemen (Schweiz) am 24.06.2020]

[Juli-Call Nr. 1]

[August Nr. 1]

Version #1
Erstellt: 2024-10-13 00:14:17 CEST von Mario Voigt (Stadtfabrikanten e.V.)
Zuletzt aktualisiert: 2025-02-25 21:22:13 CET von Mario Voigt (Stadtfabrikanten e.V.)