Building an Open Source Access Control System for Fablabs based on odoo and openHAB

Fabian Meyer[a] and Michael Schäfer
Institute of Computer Science, Hochschule Ruhr West, University of Applied Sciences, Lützowstraße 5 Bottrop, Germany

Keywords: Makerspaces, Fablab, Access Control.

Abstract: Controlling machine access in Fablabs and makerspaces is a crucial task. Different types of machines require different types of briefings. This is especially important to avoid damage and injury. Controlling access automatically is thereby desirable, as it is otherwise labor-intensive. Currently available software to organize Fablabs and makerspaces have either a rather high price tag or lacking the functionality for automated access control. Self-developed hardware is also quite common but often, due to regulatory constraints, not allowed to operate on mains. Also, there is a wide range of home automation devices that are certified for switching mains voltage. We have developed a prototypical system that makes these devices available for use in the access control of Fablabs and Makerspaces. We have identified openHAB as a useful solution for the abstraction of devices from various manufacturers.

1 INTRODUCTION

Fablabs and Makerspaces are usually small workshops that offer their visitors tools for building and repairing all kinds of things. Such Spaces have had stable growth in recent years. As of October 2019, there are 1788 worldwide Fablabs registered at fablabs.io[1]. We are already talking about the next digital revolution: after *personal computing* there comes *personal fabrication* (Mikhak et al., 2002). Fablabs are at the forefront of this movement because they enable anyone to engage in this field. However Fablab operators have to ensure that visitors are well instructed before they start using machines. This especially is important for those machines which could be damaged if handled incorrectly or those which could harm people. The easiest way to reduce these risks is to restrict machine access to people who have been instructed. Doing this manually is very labor-intensive and impracticable. For this reason, an automatic access control system would be the preferred way to limit access. Besides limiting access, as a Fablab operator one also has to keep records about machine usage, material, and power consumption. One simple but effective way to achieve the access control part is to control the power supply of a certain machine, by simply switching mains voltage.

[a] https://orcid.org/0000-0001-5882-510X
[1] https://api.fablabs.io/0/labs.json

There are some open-source projects available in this context which are mainly based on self-build hardware for user authentication and for switching mains voltage. In addition to insurance and regulatory problems, self-built hardware often cannot be controlled via a uniform and abstract interface. This makes the integration of a wide variety of different devices time-consuming because different protocols or communication standards are used.

Other open-source projects offer tools that simplify the lab organization (e.g. room and machine reservation) but lack the access control functionality. Besides proprietary and relatively costly solutions are also available.

In recent years, home automation has also made its way into the consumer market. This trend led to the high availability of relatively inexpensive home automation devices. In our work, we therefore focused on developing a cost-effective and flexible access control system which aims at the integration of home automation devices.

2 STATE OF ART

During research on available (open source) tools regarding the topic we encountered three different categories of tools. The categories identified are: *Solely access control*, *Solely lab organization*, and *Access*

Meyer, F. and Schäfer, M.
Building an Open Source Access Control System for Fablabs based on odoo and openHAB.
DOI: 10.5220/0008911900850092
In *Proceedings of the 9th International Conference on Sensor Networks (SENSORNETS 2020)*, pages 85-92
ISBN: 978-989-758-403-9; ISSN: 2184-4380

control and lab organization. This section describes some projects.

2.1 Solely Access Control

- **acos.** (Zürich, 2019) (Breucha and Först, 2017) Is a system currently under development by the *ETH Zürich*. Besides booking and switching machines, it is also able to operate electric door locks. It logs the usage of machines and locks. For fabrication tasks it also enables users to select and account for the used material. It incorporates a so-called *base station* and *receiver unit*. The base station consists of a Raspberry Pi, a touch display, an RFID Reader and an enclosure. It serves as an access device, where users can authenticate themselves via an RFID Card to book a machine. The receiver unit serves to switch mains and is installed between the wall plug and the machine. It consists of a microcontroller of type ESP 8266 and a Relay. For door un-/locking it incorporates a solenoid door lock instead of a relay. The projects claim to use off the shelf components to build this system, which is true but it's dependent on self-build hardware which switches mains. This could be a problem, as this part of the system seems to fall under *EU Low Voltage Directive 2014/35/EU* (Recast, 2014). This directive requires verification through a conformity assessment procedure. This could lead to legal and actuarial problems when operating these devices. This project is not public, yet.
- **CarontePass (v2).** (Figueras, 2016) is an Open Source project which is similar to the *acos* system described above. The focus is on operating electric door locks, not on switching machines. It also incorporates an ESP 8266 microcontroller with relays to operate a door lock. Users also authenticate themselves using an RFID tag/card. It provides an admin interface to create users and permissions and to view access logs. The project seems to be inactive for a while, ass the GitHub page indicates[2]

2.2 Solely Lab Organization

- **Fab Manager.** An Open Source platform that provides several tools for organizing Fablabs. Among other things, this includes *user management, equipment management, room booking, analytics, payment*. It seems, besides other systems, well maintained, as the GitHub statistics are indicating (GitHub, 2019). (Manager, 2019). Besides the abilities to support organizing, it lacks an access control integration for machines.

[2]https://github.com/torehc/carontepass-v2

2.3 Access Control and Lab Organization

- **Fabman.** A proprietary platform which delivers following functionality: *user management, subscriptions/plans, machine access control, dead man's switch, booking/reservation of machines/rooms, billing & payment, activity log, API & Webhooks* (Fabman, 2019d). Fabman meets all criteria as described in 1 and more. It's present in form of a SaaS[3], so one does not have to provide own infrastructure for hosting the service. A permanent internet connection seems to be inevitable. For access control, one can add a machine (e.g. a 3D printer) in the web interface and connect it to a so-called bridge (Fabman, 2019b). A bridge is plugged between the mains and the machine and is thereby able to switch the power supply of a connected device (Fabman, 2019a). A user now can authenticate him/herself by using an NFC/RFID Tag or by scanning a QR-Code with a mobile device and then perform a login in a web form (Fabman, 2019g; Fabman, 2019f). Despite the well defined and implemented Fablab specific functionality, the service alone has a minimum cost of €39 per month for up to 100 Members and an additional premium of €12 per month per device if one incorporates Fabman's access control device (Fabman Bridge) or their API. To use Fabman's access control functionality, one could use their *Bridges* which are at €199 per piece. As a rough estimate, for a Fablab with 20 machines and up to 100 members, this would result in a one-time payment of €3582 (20 x €199 - 10% discount) and continuous payment of €279 per month. Resulting in annual costs of €6930 for the first year and €3348 per year for the following years[4].

3 ARCHITECTURE

Our goal was to implement a solution, build on Open Source Software and of the shelf, certified hardware

[3]**S**oftware **as a S**ervice

[4]All calculations based on price information published on the following websites: (Fabman, 2019e) and (Fabman, 2019c)

for mains switching. Besides that, the costs for needed hardware should remain at a reasonable limit, compared to the other Open Source and proprietary systems. Another requirement was to be able to remotely sense power consumption. The solutions presented in section 2 are mainly using Wireless LAN for communication. As a Wireless LAN connection, especially in the 2,4 GHz band, tend to get quite unstable in crowed places (many emitting nodes in the same area), we also looked for a more stable and specialized wireless communication protocol.

The following section will first describe the main parts of our proposed system and then presents the architecture in detail

3.1 openHAB

As described in the introduction, the integration of hardware from different manufacturers is relatively complex, since ways of communicating with this hardware must be implemented using different standards and protocols.

openHAB acronym for *open **H**ome **A**utomation **B**us* is an Open Source home automation platform. openHAB enables a user to automate the control for a multitude of devices. It is technology agnostic regarding the controlled devices, as different protocols and third party systems can be accessed through so-called *bindings*. Through these bindings, a connected device can offer its functionality. For a switchable power plug, this could, for example, be a switching and power sensing capability. These capabilities are then organized through so-called *things* and *items*. A thing could be an outlet with multiple switchable plug sockets, an item would then be a specific plug socket. One feature, which is key for our solution, is the REST-API provided by openHAB. Through it, one can control the aforementioned things by other software. (openHAB Community, 2019)

For our project, we used version 2.3.0. We also used the *Z-Wave-Binding* to enable openHAB to control Z-Wave hardware and the *REST Documentation Add-on* to explorer available endpoints.

3.2 Z-Wave

Z-Wave is a wireless RF-based communications protocol. It operates in the sub-1GHz band and is therefore not prone to interference with other wireless protocols in the 2.4 GHz range (Bluetooth, ZigBee, Wireless LAN). Besides the ability to better penetrate walls, compared to 2.4 GHz, Z-Wave supports mesh networks without a master node. For automation tasks, this makes it superior to Wireless LAN. (Z-Wave-Alliance, 2019)

For our project we used the following parts:

- **Wallplug:** *Fibaro FGWPF-102 ZW5 v3.2*. This wall plug is rated for 230V AC, 50/60 Hz and 2500 Watt for connected machinery. Besides the switching ability, it also provides a remote reading of the current power consumption. This is especially important to sense and log power consumption during usage. The price for one plug is at about €47(Alternate, 2019)
- **Z-Wave Transceiver:** *Aeon Labs Z-Stick Gen 5 Model ZW090-C (868,42MHz)*. The price is at about €55(Rakuten, 2019a)

3.3 Access Device

As described in section 3.6, our system in this prototypical implementation used an Android Tablet as an access device. We investigated for an inexpensive Tablet with NFC capabilities. We found the Lenovo TB3-X70F with a price of about €195 (Rakuten, 2019b) to be a good fit.

3.4 odoo

odoo formerly known as *openERP*, is an Open Source System for *Enterprise Resource Planing*. It provides several modules for different tasks: Accounting, Invoicing, Inventory, and CRM[5], to name just a few. To integrate with other software, odoo provides an XML-RPC-API. (SA, 2019)
For our project, we used the community edition in version 12.0. The following modules were installed:

- **Contacts:** Base module to manage customers
- **Sales:** Base module to manage offers
- **Billing:** Base module to manage invoices and payments
- **Warehouse:** Base module to manage inventory

3.5 FabAccess-API

To integrate with *odoo* and *openHAB*, we developed an application that serves as a business logic layer and API for devices that users authenticate against to book and switch on machines. Like shown in Figure 1, our application integrates with *odoo* and *openHAB* through their respective APIs. *odoo* thereby serves as user database and bookkeeper, *openHAB* as an abstraction layer between specific actuator hardware and our application. It provides a REST-API with following endpoints:

[5]Customer Relationship Management

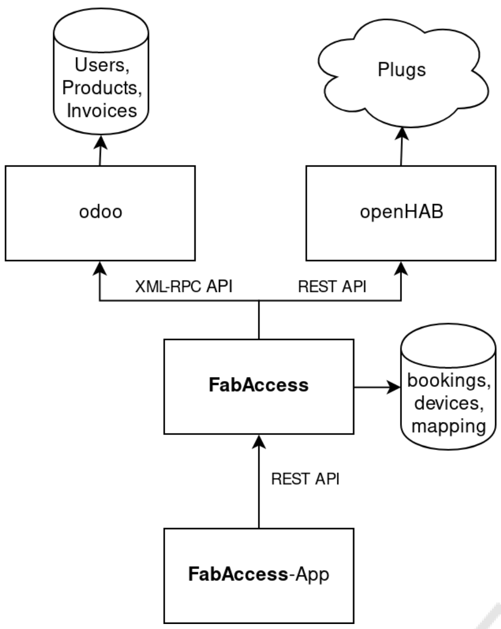

Figure 1: Schematic architecture of our proposed system.

- **Devices:** Provides available devices (switchable machines) and meta data
- **Booking:** Provides current bookings and possibility to book a machine or end a booking
- **Place:** Provides information, which machines are located at which physical location
- **Auth:** Provides endpoint for the access devices to authenticate against

Our application uses its own database to store relevant information.

3.6 FabAccess-App

The solutions presented in section 2 made use of RFID tokens or cards for user authentication. We also choose RFID cards for user authentications as passive cards are relatively inexpensive and widely accepted. Despite the presented solutions, we did not make use of custom hardware for the access devices but rather used an Android tablet with NFC capabilities. This for us seemed like the choice with the most flexibility, regarding future features. Our concept provides for one access device to be used for several switchable machines. We solve this through a concept of so-called places. A place can, for example, be a shelf or table where several machines are located.

3.7 RFID Keycards

For the user to authenticate we used *ISO14443A RFID MIFARE Classic® 1K* cards, which have a unique 4 byte UID that can not be changed.

4 IMPLEMENTATION

4.1 odoo

odoo is used in version and with plugins described in section 3.4. Despite that odoo's contact model (our user model) is extended with a field of type string for the UID of the RFID card and a field of type boolean, which represents if a security briefing took place. Furthermore, the GUI Form of the contact view was also extended with these two fields. odoo provides a graphical user interface for these steps.

4.2 FabAccess-API

Our applications was implemented in Node.js[6] with TypeScript[7] and the Express[8] framework. As database MongoDB was used. The following subsection describes essential parts of our FabAccess-API and the interaction with odoo and openHAB. As described, the FabAccess-API provides a REST-API. An access device can interact with this API via basic CRUD[9] requests on different routes.

- **Authentication:** Our application requires authentication in two ways. First of all, an access device has to authenticate itself against the provided REST-API. Our application, therefore, keeps every access device in its database with a specific API key. An access device has to use its API key to authenticate itself against the API. When successfully authenticated, a JWT token is offered. The access device then uses this token to authenticate further requests. For a user to book and switch a machine, a second authentication step is imposed. The user has to use his or her RFID card for authentication. If a user with the corresponding UID was found in odoo and a security briefing took place, a second intermediate token is provided. This token is valid for only 20 seconds and has to be used with the request, that books and switches the machines.

[6]https://nodejs.org
[7]http://www.typescriptlang.org
[8]https://expressjs.com/de
[9]**C**reate **R**ead **U**pdate **D**elete

- **Actuators:** In our FabAccess-API the openHAB items described in section 3 have an abstract representation that is saved to the local database. They are represented as actuators in our application. We choose this approach to enable a loose coupling between switchable machines and the corresponding switchable wall sockets. This, in our opinion, makes sense, as machines might move ore change quite frequently. Alike the service class which interfaces openHAB is implemented against an Interface so that it is not dependent on a specific implementation. This enables our software to be relatively easily extended with the ability to interface automation software other than openHAB.
- **Devices:** Devices are representations of switchable/bookable machines. The mapping to the corresponding actuator in openHAB is present in the database. For bookkeeping, the product reference to the corresponding product in odoo is connected to a device. In our implementation, this product is a machine hour for a specific machine type. Through the device endpoint of the FabAccess-API, an access device can request metadata like the name of the device.
- **Bookings:** For an access device to initiate a booking and switch on the mains for a connected machine, the FabAccess-API provides an endpoint to initiate a booking. The booking process consists of two steps. The first step involves creating a representation of a booking in the local database. The saved representation of a booking consists of the booked device, the user who booked the device and the start time of the booking. After that, the second step involves switching the plug (actuator) through openHAB. To end a booking and to switch off a device, an access device calls the same endpoint with a DELETE request. This initiates two processes, first the generation of an invoice for the corresponding user in odoo. Second, switching off the corresponding plug (actuator). Besides that, for the access device to display the current state of a machine (booked or free), the FabAccess-API provides the possibility to retrieve the current booking state of each device.
- **Places:** As described in section 3.6, our concept provides for one access device to be in charge of switching multiple machines (devices) located at the same physical location. The FabAccess-API, therefore, maps several devices to one so-called place. These places are also persisted to the local database. The places data model provides for machines to be placed in a two-dimensional grid. An access device can retrieve all devices located at one place via the places endpoint.

4.3 FabAccess-App

The FablabAccess-App was implemented in React Native[10] with JavaScript. For the user interface, we used the React Native Elements component library[11]. We chose React Native for app development as it provides the possibility to generate Android and iOS apps from the same program code. Besides that, it can generate user interfaces with platform-specific look and feel.

To interface the NFC hardware of the device, we used the *react-native-nfc-manager* module[12]. This abstracts the native Android and iOS APIs and makes NFC functionality available in JavaScript.

The app retrieves and sends its data through the FabAccess-API, described in section 4.2

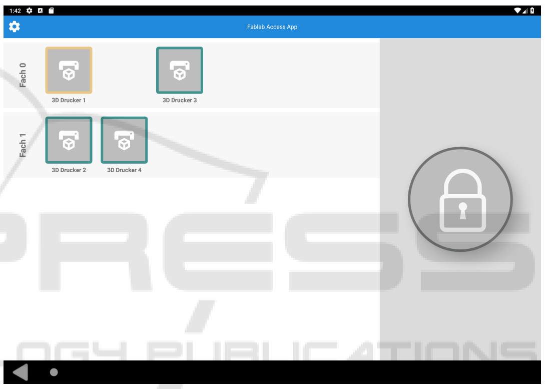

Figure 2: Screenshot of our FablabAccess-App.

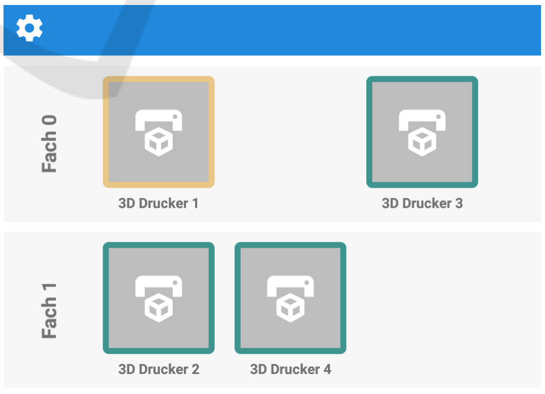

Figure 3: Detailed Screenshot of header section and booking panel.

[10]https://facebook.github.io/react-native/
[11]https://react-native-training.github.io/react-native-elements/
[12]https://github.com/whitedogg13/react-native-nfc-manager

Figure 2 shows a Screenshot of our app running on an Android Tablet. Figure 3 shows a more detailed section of the header and the booking panel. The user interface consists of three areas:

- **Header and Settings:** This area displays a button on the left side, which leads to the settings menu of the app. This menu enables an admin to configure the hostname or IP address of the FabAccess-API, the API key and the name of the device.
- **Booking Panel:** This area shows all machines available at the corresponding place. Machines are visualized through abstract icons. The view is shown in figure 3 shows a section of the booking panel, which in this setup represents a shelf with 3D-Printers in it. The border color of a machine icon reflects the current state of a machine. Green indicates that the machine is available, yellow indicates that the machine is currently booked, red indicates that the machine is currently unavailable. A machine is booked by tapping on a machine symbol.
- **Access Area:** This area indicates the current authentication state. The authentication state is indicated by a lock icon. A closed lock indicates that the user is not authenticated, an open lock indicates that the user is authenticated. A user authenticates oneself by putting his or her RFID card near the integrated NFC reader of the Tablet. Once authenticated the user can book a machine for a period 20 seconds, after that the user has to re-authenticate.

A user is only allowed to end bookings that were initiated by him or herself.

5 EVALUATION

Our goal was to prototypically implement an access control system that is more flexible and cost-effective than the solutions in section 2. Another goal was to only incorporate certified of the shelf hardware, to ensure a safe operation concerning regulatory and actuarial specifications. We met this goal, as we did not use any self-build piece of hardware.

As our proposed system is through *openHAB* hardware-agnostic regarding the protocols used by the actuators (switchable plugs), it can be relatively easily extended. This could, for example, be hardware based on the KNX or ZigBee platform.

We did not meet our self-set requirement to remotely sense and log the power consumption, as this would have taken some more effort on the side of the data handling.

5.1 Cost Effectiveness

Regarding the cost-effectiveness, we achieved a much lower cost footprint than *Fabman*. In comparison to the calculation presented in 2, the one time costs of our system are about 1970€[13] for a Fablab with 20 machines, if one estimates 5 access devices for these amount of machines. This is much less compared to Fabman. This does not include the hardware to run odoo, openHAB, and our FabAccess-API. But as the components do not need any specific hardware, we assume it will be running on a server that is already present. However, this comparison is not quite fair yet, as Fabman is a functioning well-established product and not a prototype. Compared to self-build hardware our system is still much more expensive.

5.2 Security

The Z-Wave protocol is proprietary and parts of it are not yet made public. Routing and topology management seems vulnerable, which makes different attack scenarios for Z-Wave networks conceivable, but does not allow the takeover of devices (Badenhop et al., 2017).

However, our socket approach poses a relatively small obstacle to manipulation. If the mains plug of a machine is easily accessible, a user can simply remove the switchable socket and connect the machine directly to the socket. However, there are also zWave switches that ensure the connection between the mains voltage and the machine by firmly connecting the mains cable to the switch. Regardless of this, cable-based protocols and standards such as KNX can also be used for switching sockets. However, such an installation would be much more complex. Switches that use other standards and protocols can be integrated relatively easily via openHAB.

Furthermore, a second factor for user authentication is missing in our prototypical implementation. Currently, a valid RFID token is sufficient for user authentication. To prevent the card from being passed on to other people and the forgery of RFID tokens, measures such as the query of a PIN would be conceivable.

A restart protection was also not considered in our implementation. Depending on the machine under consideration, this entails different levels of safety and injury risks. Although it would be possible to request and confirm a check of the machine's operating switch in the user interface, physical protection would be more sensible here.

[13]20 x 47€ (Wallplugs) + 1 x 55 (Z-Wave Transceiver) + 5 x 195 (Android Tablet) 3

5.3 Special Electrical Requirements

Machines that are operated with three-phase current were not considered in our implementation. The switching of such machines cannot be done by the switchable sockets we use.

6 CONCLUSION AND OUTLOOK

With our prototypical implementation we showed that by incorporating Open Source tools, it is possible to implement a highly flexible system. Compared to a well-established system like *Fabman*, we could achieve a high cost advantage. On the contrary, our system is still more expensive then any of the presented self build solutions in section 2. This, however, could be tackled by a different authentication method. A way to achieve this would be the incorporation of a user smartphone and a separate app. A user then could authenticate oneself with this app and credentials. Selecting and booking a machine could be achieved by scanning a QR-Code attached to a machine. The FabAccess-API could be relatively easily extended to implement this functionality. Besides that, other kinds of access devices could be used, as the API is agnostic regarding the interfacing hardware.

For machines that require ongoing supervision, one could implement a deadman's switch to the mobile app.

During the project, we tested Z-Wave plugs in relatively low amounts and frequency of use. A broader and longer-lasting test is being performed. Despite that, we did not test the use of other automation platforms than Z-Wave. As most of the switchable plugs can measure the amount of power that is currently used, we would like to keep book about that too. For ease of use, we are currently implementing an administration interface. Our system currently only supports *odoo* ass ERP system. One goal would be to build an abstraction that enables other ERP systems to be used with our system.

Besides the points discussed in section 5, one next step could be to implement the remote sensing and logging of the power consumption during an active booking of a machine. On the data retrieval side, this could be abstract by openHAB. Besides bookkeeping, continuous sensing of the power consumption could enable automatic detection of a finished process, for example, a finished print job of a 3D printer, as the power consumption presumably would be lower during idle.

The source code of the FabAccess server component and mobile application is available on GitHub[14] [15]. With the emergence of this paper, a group distributed over Germany has come together intending to develop an operational access system. The system developed by us will merge into this newly emerging system. All future developments will also be published on GitHub[16].

REFERENCES

Alternate (2019). ibaro wall plug gen5 sm, schalter. visited on 2019-10-30.

Badenhop, C. W., Graham, S. R., Ramsey, B. W., Mullins, B. E., and Mailloux, L. O. (2017). The z-wave routing protocol and its security implications. *Computers & Security*, 68:112 – 129.

Breucha, M. and Först, M. (2017). Student id scanner – how to log andsecure machines using off the shelf parts.

Fabman (2019a). Bridge hardware settings - fabman help guides. visited on 2019-05-01.

Fabman (2019b). Connect your equipment to fabman - fabman help guides. visited on 2019-05-01.

Fabman (2019c). Fabman bridge fb-v2 – fabman gmbh. visited on 2019-05-01.

Fabman (2019d). Fabman: Fabulous shared space management. visited on 2019-05-01.

Fabman (2019e). Fabman: Fabulous shared space management. visited on 2019-05-01.

Fabman (2019f). Supported keycards - fabman help guides. visited on 2019-05-01.

Fabman (2019g). Switch on equipment with your phone - fabman help guides. visited on 2019-05-01.

Figueras, H. (2016). Carontepass: Open access control | hackaday.io. visited on 2019-09-24.

GitHub (2019). Contributors to sleede/fab-manager · github. visited on 2019-09-23.

Manager, F. (2019). Home - fab manager. visited on 2019-09-23.

Mikhak, B., Lyon, C., Gorton, T., Gershenfeld, N., McEnnis, C., and Taylor, J. (2002). Fab lab: an alternate model of ict for development. In *2nd international conference on open collaborative design for sustainable innovation*, pages 1–7.

openHAB Community (2019). Introduction | openhab. visited on 2019-09-24.

Rakuten (2019a). Aeotec z-stick - usb adapter mit batterie | rakuten. visited on 2019-10-30.

Rakuten (2019b). Lenovo tab 3 10 business tb3-x70f | rakuten. visited on 2019-10-30.

Recast, E. (2014). on the harmonisation of the laws of the member states relating to the making available on the

[14]https://github.com/faaaaabi/fablab-api-gateway

[15]https://github.com/faaaaabi/fablab-access-app

[16]https://github.com/FabAccess

market of electrical equipment designed for use within certain voltage limits (recast). *Official Journal of the European Union*, 57(96):374.

SA, O. (2019). Open source erp and crm | odoo. visited on 2019-09-29.

Z-Wave-Alliance (2019). About z-wave technology - z-wave alliance. visited on 2019-09-29.

Zürich, E. (2019). Student project house. visited on 2019-09-23.