IoTerminal – Ausleihsystem angebunden an Asset Management Software

Projektdokumentation IoTerminal

Konzeption und prototypische Umsetzung eines Ausleihterminals mit Anbindung an Asset Management Software

Abdurrahman Karakan

Fakultät für Informatik und Ingenieurwissenschaften, TH Köln, abdurrahman.karakan@smail.th-koeln.de

Lisa Fuhrmann

Fakultät für Informatik und Ingenieurwissenschaften, TH Köln, lisa_marie.fuhrmann@smail.th-koeln.de

Niklas Canisius

Fakultät für Informatik und Ingenieurwissenschaften, TH Köln, niklas.canisius@smail.th-koeln.de

Das IoTerminal ermöglicht es Studierenden der TH-Köln das Abholen von angefragter Hardware aus dem things.moxd.io, ohne dass ein Mitarbeiter anwesend sein muss. Ähnlich wie eine Packstation sorgt es für eine einfache Ausleihe und Rückgabe. Hierbei wird über ein Pick-by-light System Schnelligkeit und Ordnung garantiert. Jegliche Transaktion wird protokolliert und ist einer Person zugeordnet.

CCS CONCEPTS • Ubiquitous computing • Object identification • Client-server architectures

Additional Keywords and Phrases: Smart Environments, Microcontroller, Sensors, Actors, Web, Asset Management, ESP8266, Raspberry Pi, HTTP, RESTful, API

1 EINLEITUNG

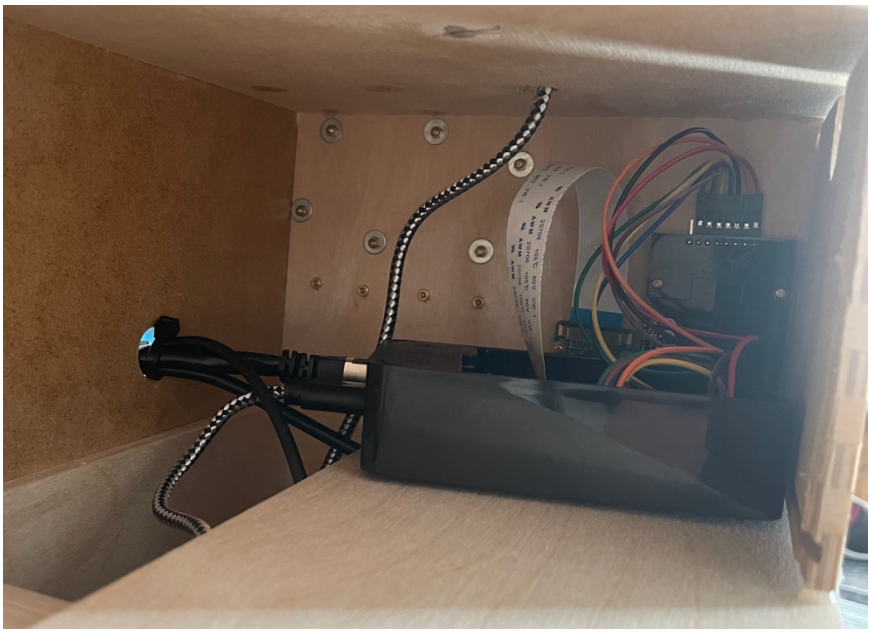
Die ursprüngliche Projektidee ist im Bereich Asset Management in Unternehmen oder im Privaten zu verordnen. Erste Konzeptionen und kleinere Prototypen wurden auf dieser Basis erstellt.

Nach Rücksprache mit den Modulverantwortlichen wurde der Wunsch nach einem optimierten Ausleihservice an der TH-Köln, konkret im moxd lab, klar. Da Studierende und Personal sich zeitlich kaum überschneiden, kann es zu Problemen bei Abholmöglichkeiten kommen. Diese resultieren, zum Beispiel bei Studierenden, die aus entfernteren Gebieten anreisen, in Frust, Ärger und in einigen Fällen auch in Konflikten.

Die Projektidee wurde im Anschluss an dieses Feedback zu einer Art Packstation umgearbeitet. Die Anbindung an eine Asset Management Software bleibt ein zentraler Projektbestandteil. Grundsätzliches Projektziel bleibt die Unterstützung von Menschen durch sogenannte Smart Environments, hier konkret durch die asynchrone Vernetzung von Menschen, die Hardware übergeben bzw. abholen müssen. Gerade die Corona Pandemie hat gezeigt, dass diese kontaktfreien Prozesse sehr wertvoll sein können.

2 PROJEKTBESCHREIBUNG

Das IoTerminal besteht aus den nachfolgenden Komponenten. Diese sind teilweise versteckt angebracht (siehe Abbildung).



Ablagestation

Damit das Equipment, welches ausgeliehen wurde, permanent zur Verfügung steht, muss es unabhängig vom Personal gelagert werden. Dies wird durch einen Container, im Prototyp durch eine “Moppe” Kommode von Ikea, mit mehreren Fächern sichergestellt.

Raspberry Pi 3 Model B+

Der Raspberry Pi ist das Herz der Anwendung. Die Datenhaltung über die Fachbelegung findet sich hier ebenso wie die Authentifizierung der Nutzer über NFC/RFID. Durch die Ethernet-Schnittstelle ist der Pi der optimale Ort für eine robuste Anbindung an Internet oder Intranet, je nachdem, wo die Asset Management Software gehostet wird. Außerdem verfügt er über hinreichende Computing Ressourcen, um die aufwendige Erkennung der Objekte mittels Kamera umzusetzen.

ESP8266

Der ESP steuert und kommuniziert mit Buttons, Motor, Display sowie dem LED-Strip. Die im Projekt bereits verfügbaren Komponenten ermöglichten eine effiziente Arbeit am Prototypen mit dem ESP. Durch den USB-to-serial-Converter ist es möglich, eine robuste kabelgebundene Schnittstelle zum Pi umzusetzen. Für Steuersignale mit kleinen Datenpaketen ist diese Schnittstelle gut geeignet.

LED-Strip

Als Indikator für das relevante Fach in der Station wird ein NeoPixels Strip über die Adafruit_NeoPixel Library angebunden. Die Steuerung funktioniert über eine Bibliothek, die das interne Schieberegister verwaltet, und ist sehr zuverlässig.

Display

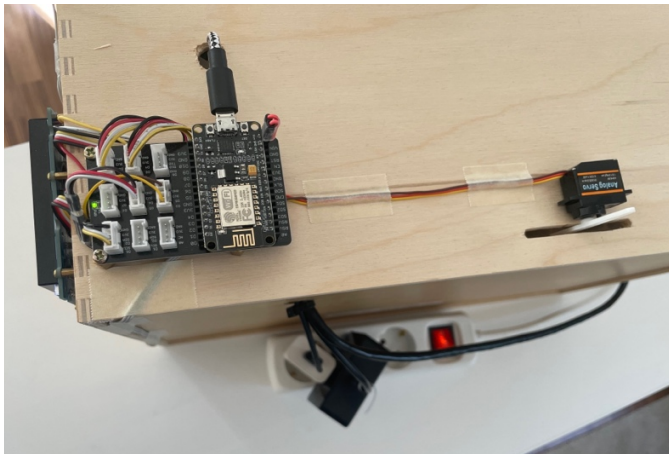
Als Display kommt ein 2x16 LED-Display zum Einsatz, dieses zeigt dem Nutzer Nachrichten an und dient kontextbasiert zur Beschreibung der Buttons.

Buttons

Die Buttons lösen mittels Interrupts einen Steuerbefehl auf dem ESP aus, der an den Pi übermittelt wird. Dieser kann je nach Kontext auf Befehle warten und diese dann umsetzen.

Motor

Der Motor dient als simulierter Schließmechanismus, indem er mit seinem Rotor ein Fach freigibt oder verriegelt.



2.1 Verwendete Technologien

RFID & NFC

Zur Authentifizierung der Nutzer des IoTerminals wird der RFID-Standard (NFC) genutzt. Die NFC-Chipkarten können einfach und schnell neu mit Nutzerdaten beschrieben und ausgelesen werden. Außerdem braucht man hier keine eigene Stromquelle oder Datenbank. Durch die NFC-Technologie werden mögliche Störsignale außerdem auf ein Minimum reduziert. Des weiteren gibt es für den Raspberry Pi sehr gute Dokumentationen [1] zur Einrichtung und Verwendung von NFC-Shields, wodurch die Integration ins System sehr einfach erfolgen kann.

QR-Code

Jedes ausleihbare Asset wurde für das Projekt mit einem QR-Code versehen. Hierbei war vor allem die Speicherung von Daten auf kleinstem Raum ausschlaggebend, da die Assets teilweise sehr klein sind und auch der Umweltaspekt eine Rolle spielt. Deswegen werden auch keine Barcodes verwendet, auch wenn diese bspw. auf Kabeln besser scannbar sind. Außerdem können QR-Codes schnell und einfach kostenlos erstellt werden. Ebenso wie bei der NFC-Technologie gibt es für den Raspberry Pi gute Dokumentationen [2] für die Einrichtung des "Camera Module V2". Diese liest die QR-Codes schnell und sicher aus. Datamatrix-Codes werden nicht verwendet, da es kaum Informationen für ihre Verwendung und Verlässlichkeit in Verbindung mit dem Raspberry Pi gibt.

Serielle Schnittstelle (im Vergleich zu Bluetooth / MQTT)

Die Datenübertragung zwischen Raspberry Pi und ESP erfolgt über eine USB-Verbindung. Ein Faktor bei dieser Entscheidung ist der geringere Verkabelungsaufwand, da der ESP über den USB-Port bereits Strom vom Raspberry Pi erhält. Außerdem ist die Datenübertragungsgeschwindigkeit für den Prototypen dieses Projekts zu vernachlässigen, da nur kleine Datenpakete übertragen werden. Des weiteren ist die USB-Verbindung weniger störungsanfällig als bspw. Bluetooth LE. MQTT wäre ebenfalls eine Möglichkeit für das Projekt gewesen, allerdings kann durch die serielle Schnittstelle komplett auf WIFI sowie das Hosting eines MQTT-Brokers verzichtet werden, beides sorgt für einen geringeren Stromverbrauch und eine höhere Zuverlässigkeit.

Die Dokumentation der seriell zu übermittelnden Datenpakete mit Beispielen befindet sich im Wiki des Projektrepositories an folgendem Ort: https://github.com/nikcani/io-terminal/wiki/Technical-Docs#serial-protocol

Python

Auf dem Raspberry Pi wurde für das NFC-Modul, den QR-Code Scanner und die Verbindung zu Snipe-IT mit Python gearbeitet. Der große Vorteil von Python war zum einen die hardwarenahe Programmierung, aber auch die breit gefächerten Bibliotheken. Des weiteren gibt es, wie bereits erwähnt, viele Dokumentationen für Python auf dem Raspberry Pi.

C++

Für die Firmware des ESP wird C++ verwendet, da dies die gängige Programmiersprache im Bereich rund um Arduino ist. PlatformIO war eine sinnvolle Entscheidung zu Beginn des Projekts. Durch den technischen Ausfall eines ESP32 im Projekt musste ein Ersatzgerät organisiert werden, dies wurde allerdings ein ESP8266, welcher ausreichend für das Projekt ist und durch PlatformIO auch direkt mit dem bisherigen Projektstand geflasht werden konnte.

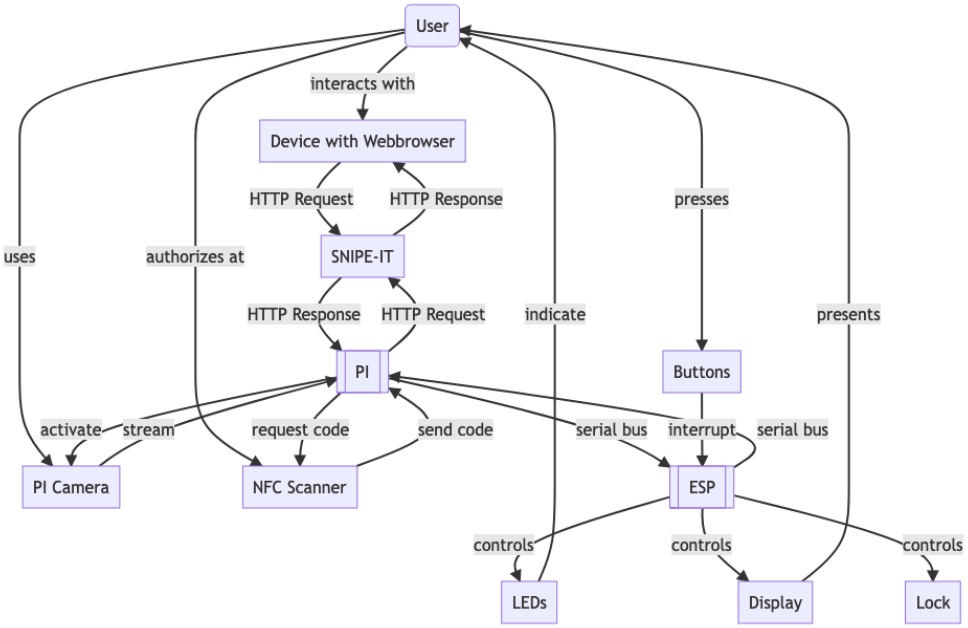
Snipe-IT

Bei der ursprünglichen Recherche nach Asset Management Software, die quelloffen und leistungsstark ist, ergab sich Snipe-IT als Favorit. Die Entscheidung für Snipe-IT liegt seit der Projektanpassung auf die Ausleihe des moxd lab vor allem darin begründet, dass im moxd lab genau diese Software sowieso bereits im Einsatz ist. Die gute Dokumentation [3] der RESTful HTTP Schnittstelle, auch für die Einbindung in Python, war ein weiterer wichtiger Faktor.

Das Hosting von Snipe-IT wurde auf einem virtuellen Server mit Ubuntu und dem Administrationspanel Plesk umgesetzt und die Software ist im Internet aktuell unter https://snipe-it.nikcani.de/ erreichbar.

2.2 Architektur

Die Interaktionen zwischen den einzelnen technischen Komponenten sowie die Interaktionen mit Usern werden im nachfolgenden Diagramm ersichtlich.

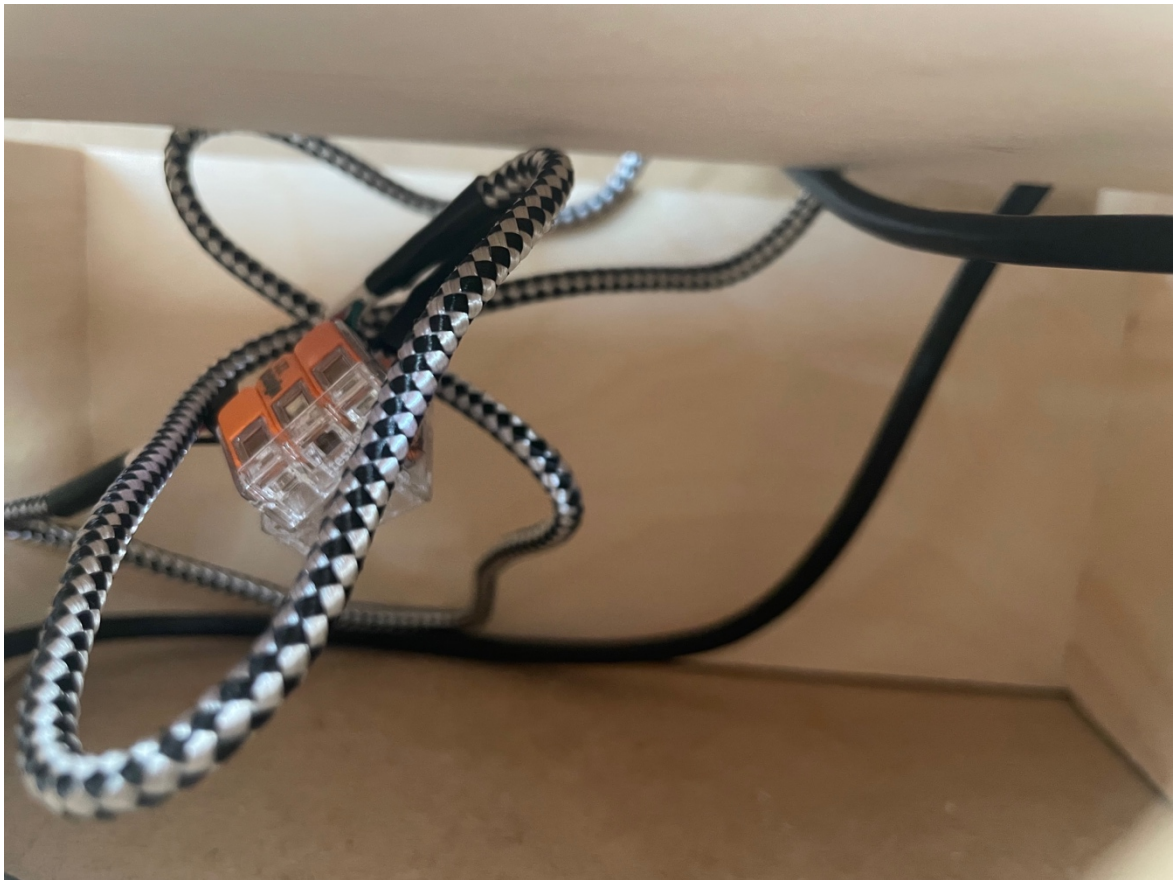


2.2.1 Technische Hürden

Eine der technischen Hürden war der hohe Zeitaufwand beim Flashen des Raspberry Pi. Um den QR-Code Scanner an dem Camera Module V2 verwenden zu können, benötigt man OpenCV contrib. Diese Version von OpenCV nimmt auf dem Raspberry Pi 3 Model B+, welcher im Projekt verwendet wird, zwei Stunden zur kompletten Installation in Anspruch. Da der Raspberry Pi im Rahmen des Projektes zweimal neu aufgesetzt wurde, kam es hier zu einem enormen Zeitaufwand.

Ein weiteres Problem waren die teilweise veralteten Bibliotheken in den Dokumentationen. Hier wird nicht immer automatisch die aktuelle Version durch den Raspberry Pi vorgeschlagen, sondern lediglich eine Fehlermeldung ausgegeben. Auch hier kommt es zu einem weiteren unerwarteten Zeitaufwand durch das Suchen und Testen anderer Bibliotheken.

Kurz vor Projektende ergibt sich beim Zusammenführen aller Komponenten ein weiteres Problem. Die Stromversorgung des ESP durch den Raspberry Pi über USB 2.0 funktioniert nur in kleineren isolierten Tests, bei der gleichzeitigen Verwendung von ESP, Display und Motor zeigt sich aber, dass die Stromstärke von 0,5 Ampere [4] nicht ausreichend ist. Daher wurde das USB Kabel zwischen Pi und ESP geöffnet, die beiden Datenadern intakt gelassen und über ein zweites Netzteil und aufgeschnittenes USB Kabel eine Stromstärke von 2 Ampere direkt mit dem ESP verbunden. Seitdem laufen die Komponenten auch bei mehreren Versuchen hintereinander stabil.



2.3 Reflexion der Projekt- und Teamarbeit

Die Entwicklung der Projektidee sowie der Architekturentwurf und die Aufteilung der einzelnen Aufgaben erfolgten zu gleichen Teilen im Team.

Danach war es Lisas Hauptaufgabe, den Raspberry Pi betriebsbereit zu machen [9], sowie den NFC- und QR-Code Scanner einzurichten. Nach der Installation des Betriebssystems zeigte sich, dass die Bibliotheken des ersten NFC-Shields nicht mehr unterstützt werden. Deshalb übernahm Lisa die Abholung, Verkabelung [10] und Einrichtung des neuen Shields und richtete die Read.py und Write.py ein. Anschließend richtete sie das Camera Module V2 ein [11], übernahm die Installation von OpenCV und schrieb einen ersten funktionstüchtigen Code für den QR-Code Scanner [2].

Parallel setzten Abdurrahman und Niklas die ersten Prototypen am ESP um. Abdurrahman übernahm die erste Anbindung der Buttons, des Displays, sowie des Motors. Niklas übernahm primär sowohl die Organisation der Kommode als auch die Anbringung der Komponenten und deren Verkabelung untereinander. Finale Schritte davon wurden teilweise im Team umgesetzt. Nach Abdurrahmans Vorarbeit hat Niklas den ESP Code zusammengeführt, strukturiert und Fehler behoben. Außerdem war es Niklas Aufgabe den LED-Strip korrekt anzusteuern.

Im Team wurde das Protokoll für die serielle Schnittstelle entworfen, umgesetzt wurde dieses im Anschluss von Niklas. Er hat ebenfalls die Snipe-IT Einrichtung und das dazugehörige Hosting übernommen. Im Anschluss daran konnte Abdurrahman sich in die Funktionsweise von Snipe-IT einarbeiten, notwendige Einträge anlegen und die RESTful HTTP API in Python anbinden und erkunden. Dabei hat er auch Endpunkte vorbereitet, die im Anschluss doch nicht mehr notwendig waren.

Im nächsten Arbeitsschritt haben Lisa und Abdurrahman die Verbindung zwischen dem Raspberry Pi und Snipe-IT [3] umgesetzt. Hierbei wurde im Pair Programming Verfahren gearbeitet. Während Abdurrahman den Code schrieb, hat Lisa über die Problemstellungen nachgedacht, die Programmabfolge strukturiert, den geschriebenen Code kontrolliert und auffallende Probleme angesprochen, sowie Lösungsvorschläge unterbreitet. Hierbei wurde auch der Code vom QR-Code Scanner optimiert.

Bei der Stromversorgung wurde von Abdurrahman und Niklas ein Fehler in der Stromversorgung entdeckt und eine erste Alternativlösung, in Form eines powered USB-Hub, organisiert, die aber nicht funktionierte. Daraufhin hat Niklas die Verkabelung manuell angepasst und damit die Hardware finalisiert.

Anschließend hat Niklas die Verbindung von ESP und Pi umgesetzt und dabei den Python Code für den Pi refactored, optimiert und den Programmfluss angepasst. Die kleinen Fehler, die sich dabei eingeschlichen haben, konnten Abdurrahman und Lisa beim nächsten Treffen lösen, im Team wurden noch letzte Optimierungen an der QR-Code Funktion umgesetzt.

Beim Videodreh hat Niklas die Kamera geführt und den Schnitt übernommen, Lisa und Abdurrahman spielten die Abläufe vor der Kamera durch. Lisa hat das Voiceover aufgenommen, den Schnitt finalisiert und im Team wurde die Dokumentation geschrieben.

3 FAZIT

3.1 Projektergebnis

3.1.1 Technisch

Zusammenfassend kann gesagt werden, dass durch den asynchronen Ansatz von IoTerminal der Austausch von Assets zwischen dem Personal von moxd lab und Studierenden der TH Köln vereinfacht werden kann. Die Zwischenlagerung in den Schließfächern löst die zeitliche Komponente auf und macht beide Partien unabhängiger voneinander. Die technische Lösung funktioniert stabil und zuverlässig und ist erweiterbar konzipiert. Gleichzeitig ist sie frei genug gestaltet für etwaige Anpassungen oder Weiterentwicklungen. Die gesamte Codebasis ist unter https://github.com/nikcani/io-terminal öffentlich auffindbar und daher archiviert und für zukünftige Projekte nutzbar.

3.1.2 Teamarbeit

Die Teamarbeit verlief sehr positiv, jedes Teammitglied konnte eigene Stärken einbringen und viele Probleme ließen sich schnell in gemeinsamen Diskussionen lösen.

Das Projekt wurde sowohl in person als auch remote umgesetzt. Dadurch, dass das Team sich mindestens einmal die Woche für mehrere Stunden traf, war ein stetiger Informationsfluss zu jeder Zeit gewährleistet. Wichtige Absprachen gab es sowohl in Textform als auch gesprochen, Struktur bekamen die Aufgaben durch ein dafür eingerichtetes Board bei GitHub. Jederzeit war der Stand klar festgehalten und die nächsten Schritte geplant. Jedes Teammitglied war in seinen Projektanteilen engagiert und hat sich bei Problemen oder für Verbesserungsvorschläge regelmäßig mit dem restlichen Team ausgetauscht. Konsistenz und Weiterentwicklung bereits bestehender Ideen auch außerhalb des Vorlesungskontexts waren zu jeder Zeit Ziel und Wunsch des gesamten Teams. Insgesamt empfindet jedes Teammitglied durch die unterschiedliche Komplexität der einzelnen Aufgabenteile, einen gleichmäßig aufgeteilten und gerechten Arbeitsanteil am Gesamtprojekt geleistet zu haben.

3.2 Mögliche Weiterentwicklung

3.2.1 Technisch

Eine sinnvolle Weiterentwicklung ist das Versehen der Assets mit einer Frist zur Abholung aus dem IoTerminal. Bislang kann der Admin nur Assets entnehmen, die bereits vom User zurückgegeben wurden. Falls diese jedoch nie abgeholt werden, hat er aktuell keinen Zugriff darauf.

Da es außerdem sinnvoll ist, insgesamt im Betrieb mehr Schließfächer zur Verfügung zu stellen, wäre der Verkabelungsaufwand über USB höher. Hier könnte man bspw. dann mit MQTT arbeiten, allerdings sollte man dabei auch abwägen, wie zuverlässig und wartungsintensiv der Einsatz von MQTT ist.

Ein weiterer Gedanke ist der Austausch von Assets nicht nur zwischen User und Admin, sondern auch zwischen Usern selbst, um Komponenten einfach und sicher übergeben zu können, ohne zur selben Zeit an einem Ort sein zu müssen.

Eine weitere mögliche Erweiterung wäre der Einsatz des Rhasspy Voice Assistant, um die Steuerung der Station für Menschen mit körperlichen Einschränkungen zu vereinfachen.

3.2.2 Teamarbeit

Die Aufgaben auf GitHub wurden im Projektverlauf nicht immer komplett aktuell nachgehalten, dies könnte etwas besser laufen. Die Aufgaben waren aber jederzeit klar verteilt, daher war dies keine große Einschränkung. Insgesamt war die Organisation der Treffen in Person durch die unterschiedlichen Fachsemester, in denen sich die Teammitglieder befinden, schwierig zu gestalten. Hier wäre eine strikte Einhaltung festgelegter Termine wünschenswert. Dies hat das Projektergebnis jedoch schlussendlich nicht negativ beeinflusst.

4 DANKSAGUNGEN

Wir danken Professor Doktor Böhmer für den inhaltlichen Input in der Vorlesung, sowie für die Betreuung im Projekt, ebenso danken wir Jannik Bläser für die Betreuung. Weiterer Dank geht an Johannes Frielingsdorf für die Unterstützung bei elektronischen Problemstellungen.

5 REFERENZEN

[1] https://circuitdigest.com/microcontroller-projects/qr-code-scanner-using-raspberry-pi-and-opencv
[2] https://snipe-it.readme.io/docs
[3] https://www.elektronik-kompendium.de/sites/com/2212141.htm
[4] https://csatlas.com/python-import-file-module/
[5] https://dronebotworkshop.com/esp32-servo/
[6] https://github.com/esp8266/Arduino/issues/584#issuecomment-123715951
[7] https://joy-it.net/files/files/Produkte/SBC-NodeMCU/SBC-NodeMCU_pinout.png
[8] https://www.bitblokes.de/raspberry-pi-imager-1-7-mit-mehr-erweiterten-einstellungen/
[9] https://joy-it.net/files/files/Produkte/SBC-RFID-RC522/SBC-RFID-RC522-Anleitung-09-06-2020.pdf
[10] https://electreeks.de/raspberry-pi-kamera-installieren-anschliessen/