Projektbreicht Smart Music Player

Anton Bracke
Jan Eberlein
Tom Calvin Haak
Julian Hahn
Nick Loewecke

8. Dezember 2020

Inhaltsverzeichnis

1 Einleitung

1.1 Unternehmen

was macht macio aus

1.2 Projekt

Im Rahmen des Bachelor-Moduls Projekt Informatik möchte Macio deren Portfolio im IoT-Bereich erweitern sowie den Empfangsraum in Kiel verschönern. Smarte Spielzeuge mit NFC-Funktion gibt es im Kommerzbereich viele, daher soll dieses Projekt eine Open-Source-Alternative dafür schaffen. Genauer handelt es sich um einen Smarten Music Player, der NFC sowie Spotify Connect unterstützt. Dieser Music Player hat einen Microcontroller mit NFC-Reader. Auf die Box können durch kleine Figuren in Form von z.B. Figuren mit dem integrierten NFC-Tag Events getriggert werden. Bei einem Event wird vom Microcontroller über jeweilige Musikdienst-API auf einem verbundenen Gerät Musik abgespielt. Die NFC-Tags und zugehörigen Events sollen über ein Web-Frontend konfiguriert werden können. Dieses Web-Frontend wird von der Box ausgeliefert und soll für primär eine Smartphone bedienung gestaltet werden. Da es sich um ein Open Source Projekt handelt mit entsprechender Lizenz, muss auch eine aussagekräftige, öffentliche Dokumentation verfasst werden. Macio stellt die benötigte Hardware und ist für Technische Fragen unterstützend verfügbar.

1.2.1 Minimal Requirements

1. NFC-Tags lesen, schreiben und entschlüsseln
2. Mit Spotify Connect verbinden und arbeiten
3. Responsive UI konzeptionieren und umsetzen
4. Aussagekräftige Dokumentation mit Benutzerhandbuch

1.2.2 Stretch Goals

1. Sound Wiedergabe auf der Box selbst
2. Unterstützung anderer Musikdienste / Plugin-Subsystem
3. 3D-Modellierung und Print einer passenden Box
4. Cloud-Anbindung der Box, Auslieferung des UI aus der Cloud

2 Machbarkeitsstudie

2.1 NFC Tag

2.1.1 lesen

Um mit NFC Tags arbeiten zu können, müssen diese auch auf entschlüsselt bzw. gelesen werden können. Hierfür ist ein Hardware NFC-Reader notwendig. Dieser kann über USB mit dem Chip der Box verbunden werden, um von dieser ausgelesen zu werden. welche library wird genutzt?

2.1.2 schreiben

Einen NFC-Tag zu beschreiben ist davon abhängig, ob der NFC-Tag über einen eigenen Speicher verfügt. Ist dieser vorhanden, können dort z.b. Kontaktdaten hinterlegt werden. Werden diese dann von einem Smartphone gelesen, öffnet sich die Kontakte-App und der auf dem NFC-Tag gespeicherte Kontakt kann abgespeichert werden. Möglich, kann die gleiche library wie zum lesen genutzt werden?

2.2 Raspberri Pi

2.2.1 Docker Integration

möglich [1]

2.2.2 Öffentlich zugängliches Web Interface

Um den Web Server vom Pi von außerhalb des Netzwerkes erreichen zu können, muss innerhalb des Routers der Port 80 port forwarded werden. Dann kann der Pi unter dessen IP xxx.xxx.xxx.xxx:80 erreicht werden. Wenn die IP Adresse sich nicht ändern soll, kann auch eine statische IP festgelegt werden [2]. Dazu könnte dann eine Domain gekauft werden, die dann z.b. mit Cloudflare auf diese statische IP weiterleitet. Alternativ ginge es auch ohne Port-Forwarding mit nginx und ngrok [3].

2.2.3 URL für UI festlegen

todo

[1] https://phoenixnap.com/kb/docker-on-raspberry-pi
[2] https://howchoo.com/pi/configure-static-ip-address-raspberry-pi
[3] https://vatsalyagoel.com/setting-up-a-public-web-server-using-a-raspberry-pi-3/

2.3 User Interface

2.3.1 Zugriff auf NFC Reader im Cloud Szenario

Tendenziell sollte es nicht notwendig sein, dass der Raspberri Pi, der das Web-UI hosted, auf den jeweiligen NFC-Reader der Box zugreifen muss. Sollte das doch der Fall sein, könnte der Chip der Box ein API Call an den Raspberri Pi mit den NFC-Tag Daten machen, um das jeweilige hinterlegte Event zu triggern.

2.3.2 Login via Spotify, Youtube, etc.

todo: ask julian

2.3.3 Gleichen Nutzer bei verschiedenen Loginvarianten wiedererkennen

Um gleiche Nutzer zu erkennen, müssten Merkmale angelegt werden, über diese gleiche Nutzer wiedererkennbar wären. Die E-Mail wäre hierbei das einzigartigste Merkmal. Über Scopes in der OAuth Anfrage kann diese auch mitgeliefert werden. Das setzt aber voraus, dass Nutzer immer die gleiche E-Mail nutzen, was nicht immer der Fall ist. Man könnte dem (bereits eingeloggten) User die Möglichkeit geben, weitere Accounts zu diesem hinzuzufügen, sodass diese Accounts den gleichen Nutzer einloggen. Bei diesem Ansatz sollten dann die jeweiligen verbundenen Accounts zu einer unique User-ID verbunden werden.

2.3.4 Musik Artwork laden

Sollte kein Problem sein, da alle Anfragen ein Array von Bildern enthalten [4]

2.3.5 Eigene Bilder hochladen

Eigene Bilder hochzuladen sollte möglich sein. In unserem Kontext mit Vue.js und Node.js würde das Plugin "vue-picture-input"helfen. Damit könnte mit einem Axios Post an das Backend das Bild gesendet werden. [5]

2.3.6 Spotify connect Lautsprecher auswählen

Das auswählen von einem spezifischen Spotify connected Lautsprecher ist möglich. Über einen API Call an die Spotify API mit dem Endpunkt /v1/me/player/device"liefert eine Liste von allen verbundenen Geräten. Es kann auch der letzte aktive Lautsprecher genutzt werden. Dieser hat bei ïs$_active$"$denWert$"$true$".[6]

2.3.7 Spotify connect Lautsprecher speichern

Die Liste von verbundenen Geräten, die über den API Call an die Spotify API erhalten wird, enthält auch einen unique Eintrag ïd". Dieser kann gespeichert werden.

[4] https://stackoverflow.com/questions/10123804/retrieve-cover-artwork-using-spotify-api
[5] https://www.digitalocean.com/community/tutorials/vuejs-uploading-vue-picture-input
[6] https://developer.spotify.com/documentation/web-api/guides/using-connect-web-api/

2.3.8 In Cloud UI die jeweilige Box auswählen

In dem Szenario, dass wir nur ein Cloud UI bieten für alle Boxen bieten, muss es auch eine Möglichkeit geben, die Boxen auszuwählen, die gerade konfiguriert werden soll. Jede Box könnte ein QR-Code erhalten, der beim Einloggen gescannt werden muss. Je nachdem kann diese Box dann im Account hinterlegt werden oder muss bei jedem Login gescannt werden.

2.3.9 Boxdaten über Cloud UI ändern

Dafür müsste eine API gebaut werden, damit der jeweilige Microcontroller der Box mit dem Cloud UI kommunizieren kann.

2.3.10 Unterstützung von Youtube

2.3.11 Unterstützung von Apple Music

2.3.12 Unterstützung von Deezer

2.3.13 Unterstützung von Tidal

2.3.14 Unterstützung von eigener Musik (USB Stick, MicroSD Karte, Cloud)

2.4 Sonstiges

2.4.1 3D Print version

Da unsere Box nicht übermäßig groß sein wird, sollten die meisten 3D-Drucker ausreichend sein. Das Modellieren einer 3D-Print Version ist am Ende abhängig von der Expertise der Gruppe. Abgesehen davon sollte es kein Problem darstellen.

2.4.2 Sound Wiedergabe auf der Box selbst

2.4.3 Box unter 30€ Kosten

Ausgehend davon, dass wir nur einen Microcontroller wie den ESP32 nutzen, ist dieses Ziel einfach zu erreichen. Dazu wird noch das Web-UI in der Cloud benötigt sowie Speicher benötigt. wie viel?

3 Design Mockups

setze pdfs ein

4 Durchführung

4.0.1 Technologien und Hilfsmittel

Vue, vscode, devops krams, etc Entwickelt wird mit Visual Studio Code, da dort einfach ein Subsystem mit Linux genutzt werden kann [7].

[7] https://code.visualstudio.com/docs/remote/wsl

4.0.2 Projekt Management

ticket pool in github, alles zentral

4.0.3 Deployment Cycle

ziehe ticket > assigne dir selbst > draft PR > wenn fertig, setzte ündraft» assigne 2 reviewer > merge mas

4.0.4 Probleme während der Durchführung

zb

5 Code Walkthrough

vielleicht interessante Code Snippets?

6 Testing

wie haben wir getestet, haben wir getestet?

7 Technische Diagramme

ER Diagramme, UML, solcher krams