RUHR-UNIVERSITÄT BOCHUM

RUB

P R O J E K T A R B E I T

zum Thema:

Entwicklung eines Server-Client Systems zur Darstellung PDF-basierter Präsentationen

von

René Beckmann
Sascha Brexeler
Diana Castano
Tim Hebbeler
Jens Helge Micke

Betreuender Dozent	Dr. Wolfgang Theimer
Beginn:	13.04.2016
Abgabe:	20.07.2016

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung[1]

Im Rahmen der Veranstaltung „Embedded-Multimedia“der Ruhr-Universität Bochum galt es ein Projekt mit dem Schwerpunkt „eingebettete Multimediasysteme“zu realisieren. Hierzu wurde jeder Projektgruppe ein „Raspberry Pi 3“als Entwicklungsplattform ausgehändigt und nötige Kenntnisse zur Entwicklung eingebetteter Systeme und der QT-Entwicklungsumgebung vermittelt. Ziel dieses Berichtes ist die Dokumentation der Projektarbeit der Gruppe 5, bestehend aus René Beckmann, Sascha Brexeler, Diana Castano, Tim Hebbeler und Jens Helge Micke. Als Projekt wurde die Entwicklung eines Server-Client Systems zur Darstellung PDF-basierter Präsentationen gewählt. Der ausgehändigte „Raspberry Pi 3 “dient hierbei als Server, privat Vorhandene Endgeräte mit den Betriebssystemen Android, Gnu/Linux und Windows als Klienten. In den Folgenden Kapiteln werden die Organisation der Gruppe, die Definition der Anwendungsfälle und deren Umsetzung sowie dabei entstehende Probleme und deren Lösungen besprochen.

[1] Jens Helge Micke

Kapitel 2

Projekt und Organisation[1]

Inhalt dieses Kapitels ist die Vorstellung des Projektes und die Organisation der Gruppe 5.

2.1 Das Projekt

Die Entwicklung eines Server-Client Systems zur Darstellung PDF-basierter Präsentationen kristallisierte sich nach der Abwägung anderer Projektmöglichkeiten[2] heraus. Auch, dass der ausgehändigte „Raspberry Pi 3 “als Server und, in Verbindung mit einer HDMI-fähigen Anzeige, als Primäranzeige dienen soll ergab sich als natürliche Rahmenbedingung.

2.1.1 Anwendungsfälle

Bevor es an die Verteilung von Aufgaben innerhalb des Projektes ging wurden die entsprechenden Anwendungsfälle erarbeitet. Während der Realisierung wurden diese an die derzeitige Situation angepasst und nötigenfalls Erweitert.

[1] Jens Helge Micke

[2] Besprochene Alternativen: Ein kooperatives Jump ’n Run Spiel, Medienserver

1.0	Präsenter Mehrere Personen wollen eine Präsentation halten
1.1	PräsenterEinwahl Präsenter wählt sich in das System ein
1.2 VORRAUSSETZUNG	PräsentationHochladen Präsenter lädt Präsentation hoch 1.1 PräsenterEinwahl
1.3 VORRAUSSETZUNG	PräsentationAnzeigen Die Präsentation wird angezeigt 1.2 PräsentationHochladen
1.4 BEFEHLE VORRAUSSETZUNG BESCHRÄNKUNG BESCHRÄNKT	Navigation Präsenter navigiert Präsentation vor, zurück 1.3 PräsentationAnzeigen Anfang/Ende der Präsentation 2.2 PräsentationAnzeigen
1.4.1 BEFEHLE	NavigationSchaltflächen Benutzeroberfläche stellt NavigationSchaltflächen bereit wie 1.3
1.4.2 BEFEHLE BESCHRÄNKUNG	NavigationBeschleunigungssensor Navigation durch Kippbewegungen zurück, vor Orientierung des Geräts
1.4.3 BEFEHLE	NavigationMikrofon Navigation durch Klopfzeichen 1x Klopfen = vor 2x Klopfen = zurück
1.4.5 BEFEHLE	NavigationKamera Kamera erkennt Gesten zur Navigation Bewegung nach links = zurück Bewegung nach rechts = vor
1.5	Menue Navigationsmöglichkeiten 1.4.x x>1 an/abschalten Die Präsentation kann beendet werden. Eine neue Präsentation kann gestartet werden.

2.0	Publikum Mehrere Personen wollen eine Präsentation verfolgen
2.1	PublikumEinwahl Publikumsperson wählt sich in das System ein
2.2 VORRAUSSETZUNG	PräsentationAnzeigen Die Präsentation wird angezeigt 1.3 PräsentationHochladen 2.1 PublikumEinwahl
2.3 VORRAUSSETZUNG	PräsentationSpeichern Die Präsentation wird gespeichert 1.3 PräsentationHochladen 2.1 PublikumEinwahl
2.4 VORRAUSSETZUNG	Verlassen Die Publikumsperson verlässt die Präsentation 2.1 PublikumEinwahl
2.5	Menue Ein Menue erlaubt die Anwendungsfälle 2.1 - 2.4
3.0	DienstAnforderungen
3.1	Benutzer/Rechteverwaltung Mehrere Präsenter Mehrere passwörgeschützte Publikumspersonen
3.2	Datenverwaltung Sortierung und Verwaltung von Präsentationen
3.3	Klientenschnittstelle Muss derzeitige Seite der Präsentation verteilen.
3.4	Klientenverteilung Muss aktuellste Version des Klienten verteilen

2.2 Die Organisation

Aus den Anwendungsfällen in Kapitel 2.1.1 Seite 2 ff. bzw. Bild 2.1 Seite 5 ließen sich erste Hauptverantwortlichkeiten der Gruppenmitglieder ableiten und verteilen.

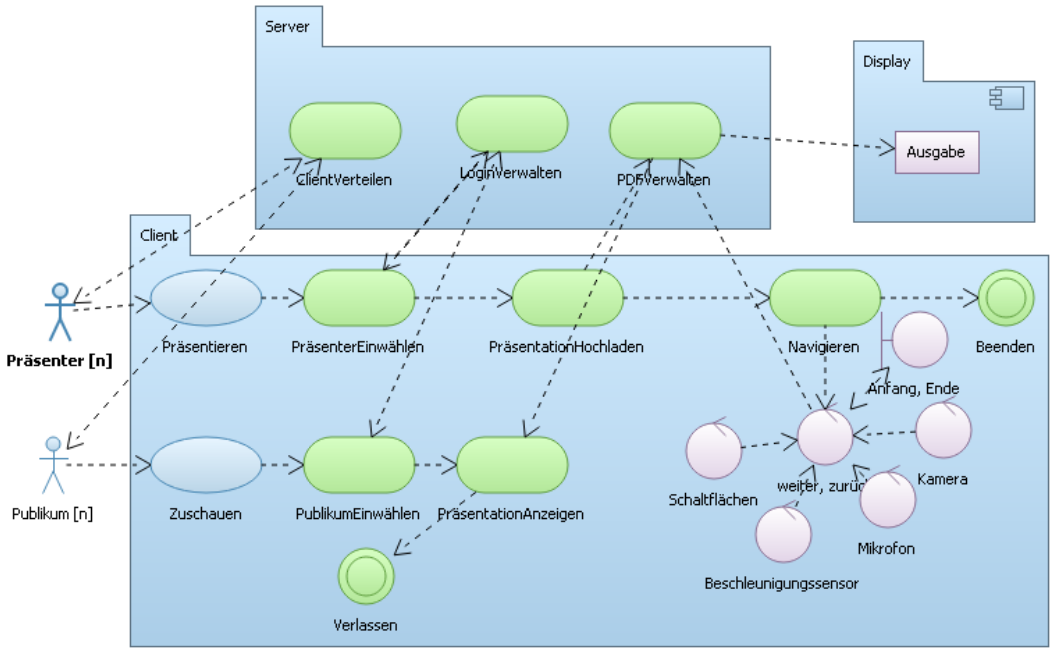

Abbildung 2.1: Anwendungsfälle

Server	Entwicklung des Servers
Klient	Benutzeroberfläche und Serverandbindung
Navigation	Navigationsmöglichkeiten Berührungsschaltflächen, Kamera, Beschleunigungssensor, Mikrofon
Raspberry Pi	Integration auf der Hardware Klientenverteilung
Dokumentation	Projektdokumentation
PDF-Renderer	Bereitstellen des PDF-Renderers

René Beckmann	Server Klient Serveranbindung
Sascha Brexeler	Klient Benutzeroberfläche Navigation Berührungsschaltflächen
Diana Castano	Navigation Mikrofon
Tim Hebbeler	PDF-Renderer Navigation Kamera
Jens Helge Micke	Raspberry Pi Navigation Beschleunigungssensor Dokumentation

GitHub.com[3] wurde als Versionsverwaltungsplatform genutzt.

Absprachen geschahen über regelmäßige Treffen und Gruppenchat.

Die zu bearbeitenden Projektteile wurden in kleinen Modulen entwickelt, getestet und nachein-

[3]https://github.com/BeckmaR/EmbeddedMultimediaSS2016

ander zusammengefügt.
Zu diesem Zweck traf sich die Gruppe zusätzlich zur Heimarbeit zu mehreren Test- und Programmierwochenenden bei denen die Gruppenmitglieder sich gegenseitig halfen und die bearbeiteten Problemlösungen testeten.

Kapitel 3

PDF Renderer[1]

Der PDF-Renderer wurde als separate C++-Klasse entwickelt und ist in die Android App und den Server eingebunden.
Im Laufe der Entwicklung wurden die Bibliotheken Poppler und MuPDF untersucht. Das Projekt Poppler ist eine unter der GPL stehende Bibliothek und basiert auf Xpdf. Poppler bietet schon ein Qt-Binding, doch leider liegt der Fokus auf unixartige Betriebssysteme und nicht auf der Plattformunabhängigkeit [2]. Die PDF Darstellung wird aber auf verschiedenen Plattform, wie Android, dem Raspberry Pi und Windows benötigt, deshalb konnte Poppler nicht verwendet werden. Die Wahl fiel deshalb auf MuPdf von der Firma Artifex Software, Inc. ,welches unter der AGPLv3 steht. Diese Bibliothek bieten kein Qt-Bindung, doch existiert ein von einer Privatperson initiiertes Projekt mupdf-qt, welches Grundfunktionalitäten mit Qt bietet. Hierbei war ein vergleichsweise einfache Cross-/Kompilierung für die verschiedenen Plattformen möglich.
Die GUI-Oberfläche wurde mit Hilfe von QML erstellt, deshalb ist ein Transfer der PDF-Daten von C++ nach QML nötig. Der Slot OpenPDF() öffnet ein betreffendes PDF Dokument. Die Funktion QQuickImageProvider bietet die Möglichkeit Bilder in C++ zu berechnen und in einem QML Image anzuzeigen. Dabei liegt die Steuerung der Berechnung in QML. Der QML Thread ruft in C++ eine Funktion 'requestImage()' auf und fragt ein QImage an, welches dann dargestellt wird. Da die Kontrolle bei dem QML-Element liegt, müssen alle Parameter wie die Seitenanzahl über QString Parameter übergeben werden. Wenn die Funktion über QML aufgerufen wird, wird als erstes die Seitenanzahl berechnet. Mit Hilfe dieser berechnet die MuPdf Lib ein Bild von der gewünschten PDF-Seite. Dabei wird das Bild von seiner Auflösung so berechnet, dass es optimal der angefragten Dimension entspricht. So wird sichergestellt, dass einerseits keine unscharfen Effekte zu sehen sind und andererseits die Berechnungszeit minimiert wird. Die PDF Renderer Klasse ist unter folgendem Link zu finden. Das mupdf-qt Projekt liegt als Submodule in dem Git Verzeichnis[3]. In diesem ist das wirkliche MuPDF Projekt als Submodule eingebunden. Von beiden Projekten wurden über GitHub Forks angelegt um Veränderungen in den Makefiles des

[1] Abschnitt von Tim Hebbeler erstellt

[2] `https://de.wikipedia.org/wiki/Poppler`

[3] `https://github.com/BeckmaR/EmbeddedMultimediaSS2016/tree/master/thirdparty`

MuPDF Projektes durchführen zu können und um in diesen crosskompilierte Bibliotheken für die verscheidenen Plattformen unterzubringen. Als Verbesserungsmöglichkeiten für den PDF Renderer könnte man die MuPDF Lib gegen einen neueren Stand austauschen. Bis jetzt sind keine Fehler in der Verwendeten aufgefallen, doch kann es vorkommen das Dateien mit neueren PDF-Versionen ggf. nicht fehlerfrei dargestellt werden können.

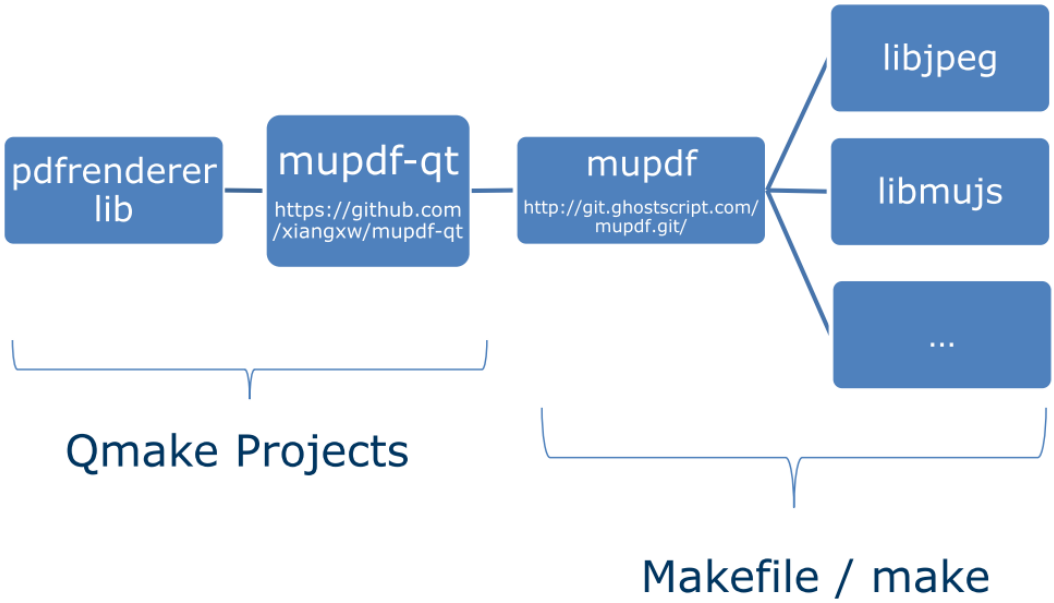

Abbildung 3.1: Struktur und Aufbau des PDF-Renderer

Kapitel 4

Der Server[1]

4.1 Zweck und Aufgaben des Servers

Der Server stellt die Komponente dar, welche die diversen Endgeräte per Netzwerk miteinander verbindet und verwaltet. Hierbei ist im Folgenden von „Clients“ die Rede, wenn es um die Menge aller verbundenen Geräte geht. Der „Master“ stellt den Sprecher dar, welcher eine Präsentation hochladen darf sowie die aktuell angezeigte Seite ändern. Alle anderen Geräte haben lediglich eine Zuhörer-Funktion.
Er authorisiert den Sprecher, der sich mit dem gültigen Master-Passwort anmelden möchte, und erlaubt nach erfolgter Anmeldung das Hochladen einer Präsentation und Steuern der anzuzeigenden Seitenzahl. Die Präsentation wird gleichzeitig auch auf dem Desktop angezeigt, so dass der Server per Monitorkabel an ein Anzeigegerät - z.B. einen Beamer - angeschlossen werden kann um die Präsentation auszugeben. In diesem Projekt lief die Server-Software auf einem Raspberry Pi.
Per Netzwerk wird die Präsentation sowie die anzuzeigende Seitenzahl auf Anforderung an die Clients verteilt, so dass jeder Client die Präsentation synchron mitverfolgen kann.
Nachdem sich ein Master von dem Server getrennt hat, ist das Verbinden eines neuen Mastergerätes ohne Probleme möglich. Dies erlaubt es, nacheinander mehrere Sprecher eine Präsentation halten zu lassen, ohne dass die Server-Software neugestartet wird. Das gleichzeitige Verbinden mehrerer Sprecher ist hingegen nicht möglich, um Verwirrungen und komplizierte Rechteverwaltungen zu vermeiden.

4.2 Das Konzept

Die Implementierung der Netzwerkverbindung erfolgte auf Basis von Websockets. Dies ist ein auf TCP basierendes Protokoll, bei dem eine bidirektionale Datenverbindung möglich ist. Die Verbindung wird außerdem aufrecht erhalten - dies ermöglicht es, serverseitig die Verbindung zum Master separat abzuspeichern. So wird nicht bei jeder Aktion eine Authentifizierung benötigt.

[1] René Beckmann

Außerdem könnte ein Server, der nicht-persistente Verbindungen verwendet, nicht die aktuelle Seitenzahl an alle Clients broadcasten - er kennt seine Clients gar nicht, und jede Aktion wird von den Clients eingeleitet. Diese müssten beispielsweise alle *x* Sekunden den Server pollen, um die aktuelle Seitenzahl zu erfragen.
Diese Technik stellt also kein *Stateless Design* dar, sondern einen sehr simplen Zustandsautomaten.

4.2.1 Kommunikationsprotokoll

Mit Hilfe der Qt-Implementierung von Websockets, den `QWebSockets`, lassen sich Text- und Binärnachrichten versenden. Auf dieser Basis wurde ein sehr einfaches textbasiertes Protokoll für die Kommunikation zwischen Server und Client entwickelt. Die folgenden Befehle können vom Client an den Server gesendet werden, dieser antwortet dann an diesen oder alle Clients.
Alle Kommandos folgen einem einheitlichen Format: Zwei Zeichen, gefolgt von einem Doppelpunkt und weiteren Zeichen beliebiger Länge (einschließlich Länge 0, also keine weiteren Zeichen). Die beiden Zeichen vor dem Doppelpunkt definieren, was getan werden soll, und die weiteren Zeichen stellen die „Payload“ dar, wie zum Beispiel weitere Argumente oder das Passwort bei dem Versuch sich als Master anzumelden.

- `RM` - **R**egister as **M**aster.

  Payload Das korrekte Master-Passwort.

  Fehlermeldungen `BADPW` bei falschem Passwort, `MASTERISSET` wenn es bereits einen Master gibt - per WebSocket an den Client gesendet

  Erfolgsbestätigung `ACK`

  Beispiele Korrekt: `RM:mpw12345`, nicht korrekt: `RM:123`

- `SP` - **S**et **P**age.

  Payload Eine Seitenzahl, die sich in einen integer umwandeln lässt.

  Fehlermeldungen `BADPAGENUM` bei ungültigen Seitenzahlen, die nicht zu einem integer gewandelt werden konnten, `NOTALLOWED` wenn der Sender nicht als Master registriert ist.

  Erfolgsbestätigung Broadcast der Seitenzahl an alle (Weiterleitung)

  Beispiele Korrekt: `SP:3`, nicht korrekt: `SP:x`

- `GP` - **G**et **P**age.

  Payload Wird ignoriert, nichts notwendig.

  Fehlermeldungen Wenn noch kein pdf verfügbar ist, wird die Default-Seitenzahl ’-1’ gesendet.

Erfolgsbestätigung Antwort mit `PN:X`, wobei *X* die Seitenzahl bezeichnet.

Beispiele Korrekt: `GP:`, oder auch korrekt: `GP:x` - Payload wird ignoriert

- `DL` - **D**own**l**oad.

  Payload Wird ignoriert.

  Fehlermeldungen `NOFILE`, wenn keine Präsentation hochgeladen wurde.

  Erfolgsbestätigung Der Inhalt der Datei wird per BinaryMessage an den Client geschickt.

  Beispiele Korrekt: `DL:`,

- `UL` - **U**p **L**oad. Kann verwendet werden, um zu erfragen, ob ein Upload erlaubt ist.

  Payload Wird ignoriert.

  Fehlermeldungen `NOTALLOWED` wenn der Sender nicht als Master registriert ist.

  Erfolgsbestätigung `ACK`

  Beispiele Korrekt: `DL:`

- Die Präsentation hochladen. Dies ist kein Textkommando, sondern erfolgt direkt per BinaryMessage. Diese wird nur akzeptiert, wenn der Sender der aktuelle Master ist. In diesem Fall wird das enthaltene `QByteArray` direkt an den pdfrenderer weitergeleitet, welcher dieses dann in eine Datei schreibt und anzeigt.

Andere Kommandos werden generell mit `BADCMD` beantwortet.

4.3 Mögliche Verbesserungen

Der Server erledigt seine Aufgabe recht gut. Die Schnittstelle ist möglichst simpel gehalten und weist eine geringe Fehleranfälligkeit auf. Diverse Funktionen für die Sicherheit oder Annehmlichkeiten wären aber noch vorstellbar.

- Verschlüsselte Verbindung. Das Master-Passwort wird, wie alles andere auch, per Klartext übertragen. Während der Präsentation ist ein weiterer Master nicht erlaubt, danach könnte ein potentieller Angreifer sich aber anmelden.
- Broadcast des Servers. Der Server könnte in regelmäßigen Abständen per UDP-Broadcast auf sich aufmerksam machen. Dadurch könnten Clients sich automatisch verbinden, und das unelegante Eintippen der ip-Adresse im Client würde entfallen.
- Zuteilung einer ID zu einem Master. Mit dieser ID könnte ein Master sich nach einem Verbindungsabbruch neu verbinden und an der vorherigen Stelle weitermachen.

Kapitel 5

Client

Die Beschreibung des Client bezieht sich im Folgenden nur auf die Applikation, welche Sprecher oder Zuhörer für die Präsentation zur Steuerung und Ansicht auf ihrem Endgerät benutzen. Client meint in diesem Zusammenhang nicht den vom Server gesteuerten Projektor.

5.1 Grafische Benutzeroberfläche (GUI)[1]

Die Gestaltung dieser GUI bezweckt eine leicht verständliche intuitiv bedienbare Oberfläche mit zielführender Benutzerführung und Hilfestellungen. Die GUI ist für Geräte mit Android als Betriebssystem entwickelt, aber theoretisch nach dem kompilieren mit eventuell erforderlichen kleinen Anpassung auch auf andere Plattformen/Geräten ähnlich abgebildet und verwendbar. Erfolgreich getestet ist die App allerdings derzeit nur für Android (4.3 + 4.4) und Windows.

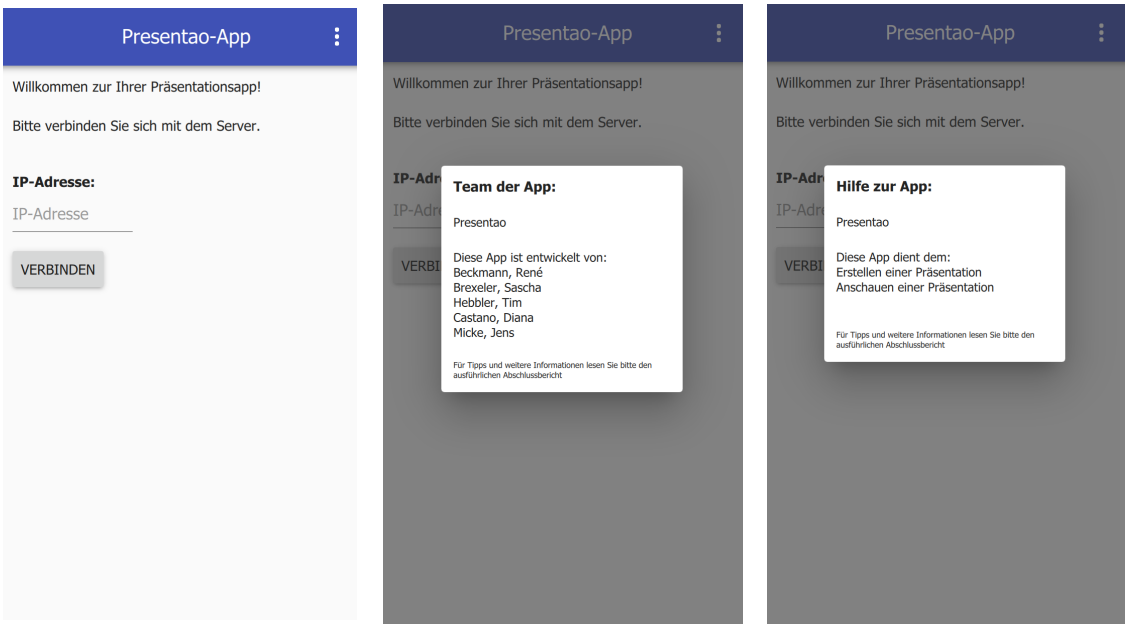

Abbildung 5.1: v.l.n.r.: Startbildschirm, Über die App, Hilfemenü

[1] Sascha Brexeler

Start der App

Von dem Startbildschirm aus besteht die Möglichkeit ein Menü (siehe Abbildung 5.2) durch klicken auf den aus drei Punkten bestehenden Menü-Button (siehe Abbildung 5.1) aufzurufen. In diesem lassen sich zwei Pop-Ups zur Hilfe und zum Team aufrufen (siehe Abbildung 5.1).

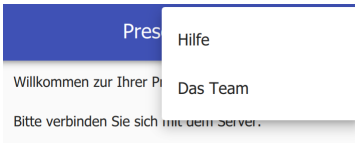

Abbildung 5.2: Menü

Verbindungsaufbau

Des Weiteren kann der Benutzer nach Eingabe einer der gültigen IP-Adresse eine Verbindung zum Server als Zuhörer aufbauen. Die Eingabe der IP-Adresse erfolgt, wie in der üblichen Notation, mit Punkttrennung und eine Portangabe ist nicht nötig, da diese in Client und Server fest einprogrammiert ist.

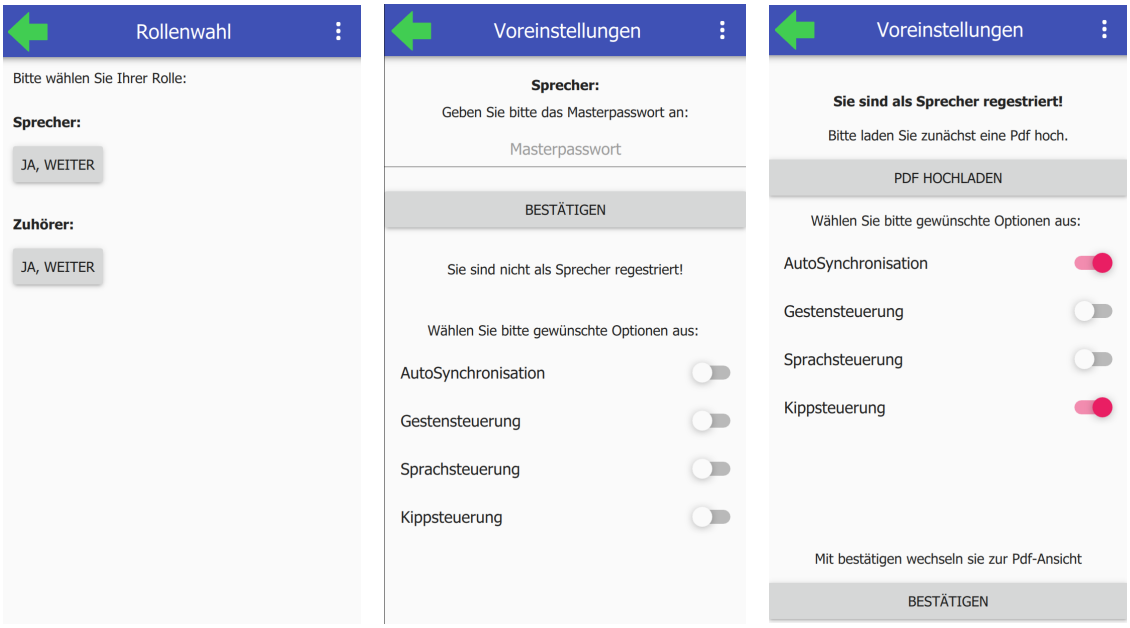

Abbildung 5.3: v.l.n.r.: Rollenwahl, Voreinstellungen des Sprecher vor und nach Registrierung

Auswählen der eigenen Rolle

Nach erfolgreichen Verbindungsaufbau wechselt die Ansicht in zur Rollenauswahl. Ein Label zeigt diese Position mittig der oberen Leiste (Toolbar) an (siehe Abbildung 5.3). In dieser Toolbar befindet sich nun zusätzlich ein grüner Pfeil nach links, um zur vorherigen Ansicht zu wechseln. Die Rollenwahl ist wiederholbar und somit korrigierbar, aber unumgänglich implementiert, da weitere Einstellung und Möglichkeiten auf dieser Entscheidung aufbauen.

Voreinstellungen als Sprecher

Der Sprecher muss sich zunächst als solcher bei dem Server registrieren. Dazu ist eine Passwortabfrage eingerichtet. Das sog. Masterpasswort ist „mpw12345“. Wenn die Eingabe fehlerhaft erfolgte, erscheint zusätzlich unter dem Textfeld zur Passworteingabe (siehe Abbildung 5.3) in Dickschrift der Hinweis: „Bitte überprüfen Sie Ihre Passworteingabe“. Sobald die Eingabe richtig und bestätigt ist, kann der Benutzer eine Pdf-hochgeladen. Dazu muss der Nutzer eine Pdf-Datei über den File-Dialog (siehe Abbildung 5.4) suchen und auswählen. Eine Erfolgreiche Auswahl sendet die Datei an den Server, der diese direkt an über den Projektor auf Seite 0 ausgibt. Bevor der Sprecher mit Bestätigen zur Pdf-Ansicht wechselt, sieht dieser noch eine Liste (ListView) mit einige Optionen aus denen er gewünschte über Schiebeschalter (SwitchDelegates) auswählen kann. Mit dem grünen Pfeil wechselt die Ansicht diesmal zurück zur Rollenwahl.

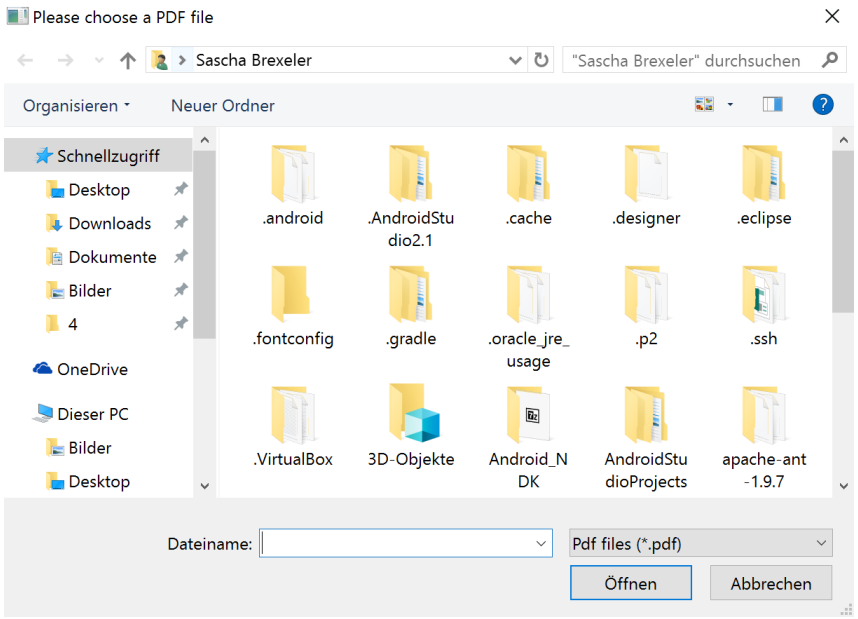

Abbildung 5.4: File-Dialog

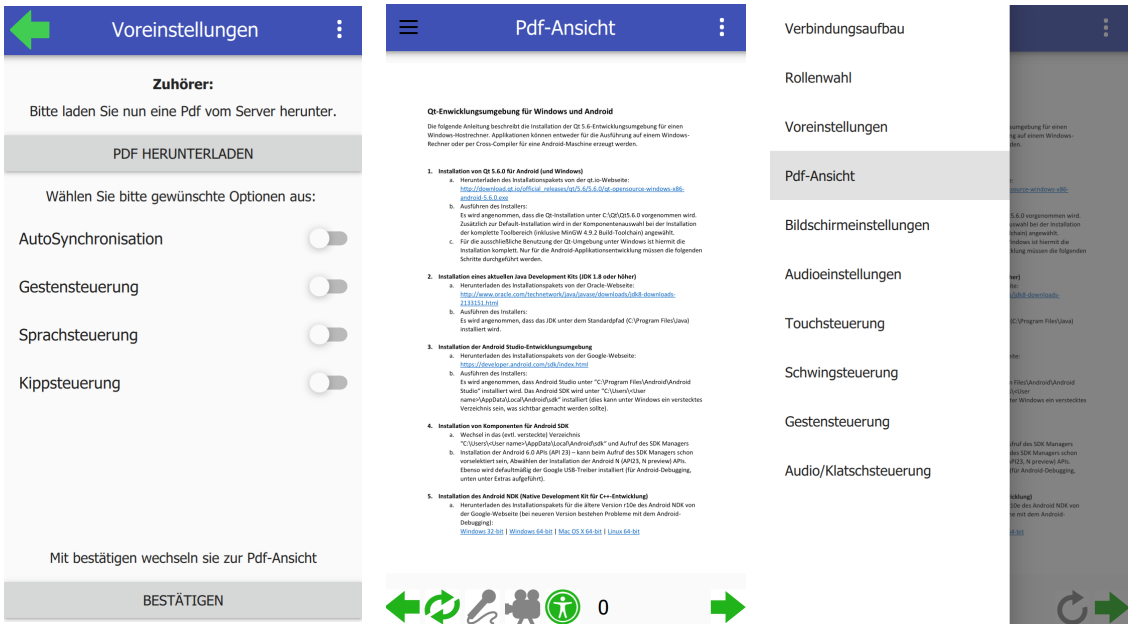

Abbildung 5.5: v.l.n.r.: Voreinstellungen des Zuhörers, PdfAnsicht und Drawer

Voreinstellungen als Zuhörer

Als Zuhörer sind die Voreinstellungen ähnlich, jedoch entfällt die Passworteingabe und Herunterladen einer Pdf-Datei ersetzt den Vorgang des Hochladens (siehe Abbildung 5.5). Hierbei erfolgt (ohne eine Auswahlmöglichkeit) das Herunterladen der zuletzt auf dem Server geladenen Datei.

Pdf-Ansicht

Bestätigen der Voreinstellungen des Sprechers oder Zuhörers bedingt einen Kontextwechsel zur Pdf-Ansicht (siehe Abbildung 5.5).

Drawer

Ab der Pdf-Ansicht ist der grüne Pfeil oben links durch einen weiteren aus drei waagerechten Strichen bestehenden Menü-Button ersetzt. Dieser Button öffnet eine vertikale Leiste (Drawer) am linken Bildschirmrand, welche ein ListView beinhaltet, dass schnelle Navigation zu allen bisherigen Ansichten und Weiteren ermöglicht (siehe Abbildung 5.5). Durch wischen am vom linken Bildschirmrand nach rechts lässt sich dieser ebenfalls einblenden. Die Navigation über das ListView im Drawer ist nur möglich solange man nicht zu vorherigen Ansichten wechselt.

Symbolleiste

Die Pdf-Ansicht beinhaltet ein Symbolleiste (siehe Abbildung 5.6) als Reihe von Icons am unteren Bildschirmrand. Mit ihr ist durch die Pfeile außen ein Blättern in der Pdf möglich. Intuitiv blättert der linke Pfeil zurück, der Rechte vor. Die anderen Buttons dienen dem aktivieren/deaktivieren von Bedienoptionen. Grün signalisiert hierbei aktiviert und grau deaktiviert. Das Mikrofon steht für die Audiosteuerung, das Kamerasymbol für Gestensteuerung und das Männchen mit den ausgestreckten Armen und Beinen im Kreis für die Kippsteurung. Neben

diesem Symbol steht die Seitennummer der angezeigten Seite.
Die ineinander greifenden Pfeile signalisieren bei grün, dass die Autosynchronisation aktiv ist. Sobald diese nicht aktiv ist, erscheint ein weiterer Pfeil geschlungen im Uhrzeigersinn in grau auf der Symbolleiste links neben dem Pfeil. Die Funktion ist das manuelle Aktualisieren der Seitenzahl. Ein Sprecher sendet mit dem Refresh-Button die Seitenzahl, die auf seinem Device aktuell ist zum Server, und dieser leitet sie an die Zuhörer-Clients weiter. Ist als Sprecher Autosynchronisation aktiviert, geschieht die bei jedem Blättern automatisch. Zuhörer können über Refresh die aktuelle Seite manuell anfragen oder mit Autosynchronisation alle 2 Sekunden zu der aktuellen Seite zurück springen. Wenn der Sprecher Blättert, erhalten die Zuhörer jedoch in jedem Fall die aktuelle Seite.

Abbildung 5.6: Symbolleiste

Informationen zur Benutzung

Da das bisherige Hilfe-Pop-Up nur recht allgemeine Informationen enthält sowie für weitere Hilfestellung auf diese eventuell nicht zur Hand liegenden Dokumentation verweist, kann sich der Benutzer über den Drawer zu einer Erklärung der Bedienmöglichkeiten navigieren. Diese Möglichkeit soll einen möglichst einfachen Einstieg ohne viel ausprobieren sicherstellen und Anwendungsfehler verhindern. Um einen mit dieser Anwendung vertrauten Anwender nicht nach jedem Start durch diese Informationen zu führen, ist das Aufrufen optional (siehe Abbildung 5.6).

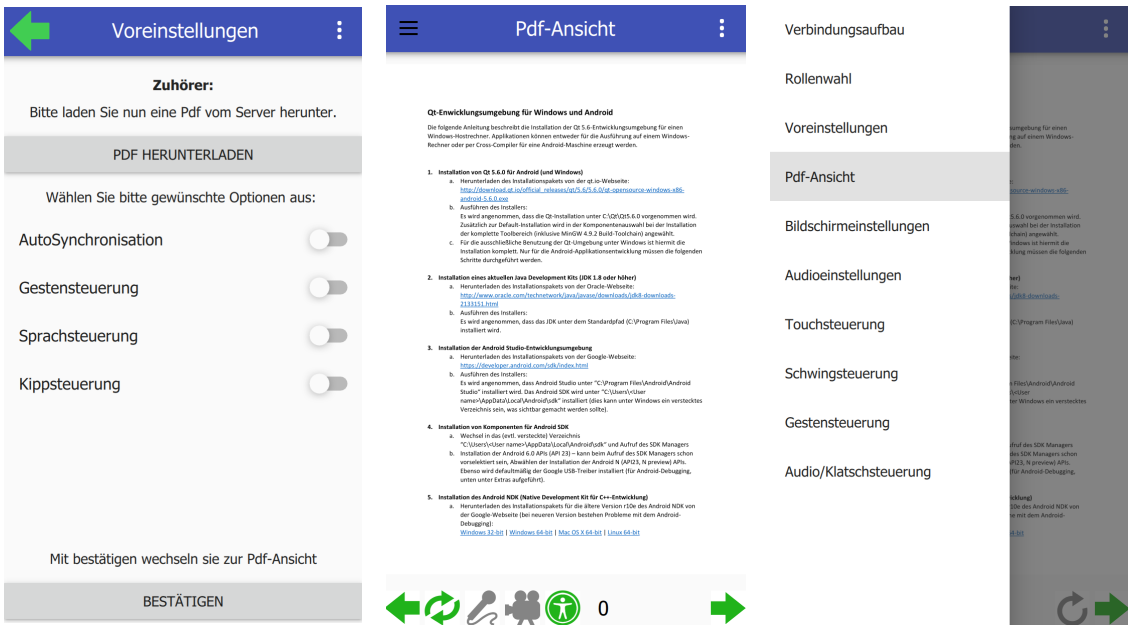

Abbildung 5.7: v.l.n.r.: Gestensteuerung, Kippsteuerung und Klatschsteuerung

5.2 Serveranbindung[2]

Die Einbindung der Funktionsblöcke zur Kommunikation des Clients mit dem Server ist in Zusammenarbeit[3] im Team entstanden. Hier erfolgt nur eine grobe Beschreibung. Details zur Kommunikation finden Sie in Kapitel ????

5.3 Software[4]

Die Software der GUI finden Sie im Git Repository [5]. Die Die diesem Ordner sind nur z.T. Ressourcen für das Steuern mittels AudiDateien für die Navigation mittels K

Konzept
Das Softwarekonzept dient der Umsetzung des Bedienkonzepts mit "Qt-Creator“ programmiert in C++ und Qml. Der Qt-Creator ist dafür wegen der Plattformunabhängigkeit ausgewählt.

Struktur

Zustandssteuerung
Die Realisierung der Grafische Oberflächen entsprechend bisheriger Einstellungen ist über Zustände einer Variablen „appState“ realisiert, die sich die aktuellen Einstellungen bzw. den Ort merkt und entsprechend Informationen mittels der Objekteigenschaft „visible“ ein oder ausblendet.

Elemente

Einbindung der erweiterten Bedienmöglichkeiten

Details

Lektionen

[2] Sascha Brexeler
[3] zwischen Sascha Brexeler und René Beckmann
[4] Sascha Brexeler
[5] https://github.com/BeckmaR/EmbeddedMultimediaSS2016/tree/master/src/app_gui

5.4 Navigation

5.4.1 Navigation mit dem Mikrofon

5.4.1.1 Anforderungen

Die Applikation setzte eine Navigation durch die PDF-Datei mithilfe des Mikrofons voraus, wodurch ein Klatschen oder Klopfen erkannt und als Blättern interpretiert werden konnte. Bei einmaligem oder zweimaligem Klatschen wurden entsprechende Signale zum vor- bzw. zurückblättern gesendet.

Eine der Herausforderung dabei war, die Erkennung unabhängig von den äußeren Gegebenheiten erfolgen zu lassen (Raum mit oder ohne Nachhall). Außerdem sollte kein Signal zum Weiterschalten der Folien erzeugt werden, wenn ein starker Klang oder Ton aufgenommen wurde (z.B. wenn der Präsentator ggf. lauter sprechen musste). Der Algorithmus und die dabei berechneten Parameter sollten dabei nicht für jedes Gerät angepasst werden, selbst wenn die Mikrofone über unterschiedliche Eigenschaften verfügten.

5.4.1.2 Umsetzung

Als erste Implementierung wurde eine Erkennung im Zeitbereich gewählt. Es wurde schnell festgestellt, dass dies eine sehr leise Umgebung voraussetzte. Selbst der Sprecher konnte das Signal aktivieren, wenn er sehr nah am Mikrofon war. Aus diesem Grund erfolgte die Erkennung im Frequenzbereich. Die Signalverarbeitung war hierbei aufwändiger, allerdings wurden damit die Herausforderungen überwunden.

Das Modul Qt Multimedia 5.7 stellt verschiedene C++ Klassen zur Verfügung, die für die Steuerung des Mikrofons hilfreich sind. Eine Abfrage über die verfügbaren Aufnahmegeräte konnte mit *QAudioDeviceInfo* durchgeführt werden. Das Audio-Format und die Darstellung der Daten wurden mithilfe der Klasse *QAudioFormat* festgelegt. Folgende Einstellungen wurden verwendet:

- Abtastrate: 8 kHz
- Anzahl der Kanäle: 1 (mono)
- Bytes pro Abtastwert: 2
- Format der Abtastwerte: Signed Integer
- Byte-Reihenfolge: Little Endian
- Codec: Linear PCM

Die Klasse *QAudioInput* bietet eine Schnittstelle um akustische Signale aus einem Mikrofon aufzunehmen. Dafür muss zuerst ein Aufnahmegerät mithilfe von *QIODevice* im Lesemodus zum Empfangen der Daten geöffnet werden. Jede Sekunde werden neue Daten aus dem Mikrofon gelesen. Die Daten, die sich im Buffer befindet werden zuerst in einem *QByteArray* gespeichert. Anschließend werden die Abtastwerte vom *Signed Integer* Format zum *Float* umgewandelt und in einem *QVector* gespeichert, um eine Fourier-Analyse des Signals zu ermöglichen.

Die Analyse im Frequenzbereich erfolgt durch das Spektrogramm, das eine Zusammensetzung des akustischen Signals in seinen einzelnen Frequenzen im zeitlichen Verlauf darstellt. Dafür müssen zuerst die im Buffer gespeicherten Abtastwerte in sich überlappende Rahmen aufgeteilt werden, wie in Abbildung 5.8 dargestellt wird:

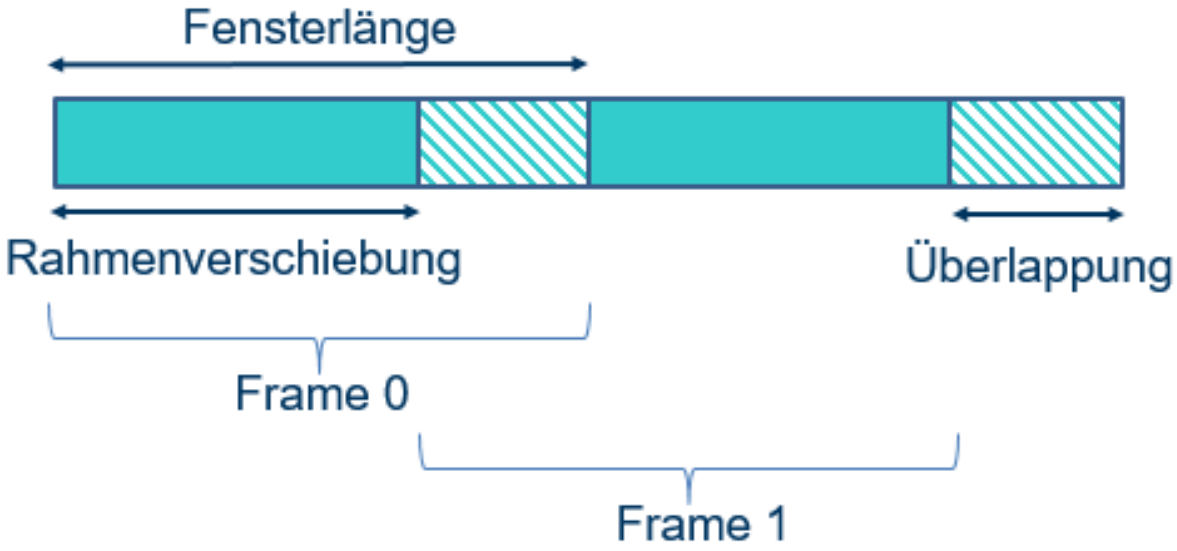

Abbildung 5.8: Aufteilung des Signals in Rahmen

Dafür wurden folgende Größen gewählt:

- Länge des Buffers: 16000 Bytes
- Länge der einzelnen Rahmen: 256 Werte (circa 25 ms)
- Rahmenverschiebung: 80 Werte (10 ms)
- Überlappung: 176 Werte
- Anzahl von Rahmen: 98
- Fenstertyp: Hanning

Falls das Signal sich nicht in genau dieser Anzahl von Rahmen aufteilen lässt, werden Nullen am Ende des Signals hinzugefügt. Da die Länge der Rahmen endlich ist und kein Vielfaches der

Periode des Signals darstellt, müssen die Rahmen mit einer Fensterfunktion gewichtet werden, um den Leck-Effekt zu vermeiden.

Anschließend muss die Diskrete Fourier-Transformation (DFT) für jeden Rahmen berechnet werden. Die Schnelle-Fourier Transformation (FFT) ist ein effektiver Algorithmus zur Berechnung der DFT. Für die Implementierung der FFT in Qt 5.7 wurde die Open-Source Bibliothek FFTReal verwendet, die für schnelle Berechnungen optimiert ist (besonders wenn die Anzahl der FFT-Punkte schon bekannt ist).

Es wurde zunächst der Betrag der Transformation berechnet, daraufhin erfolgte die Berechnung des Logarithmus. Folge dessen wurden die Ergebnisse in Form einer Matrix in einer CSV-Datei gespeichert. Die Spalten stellen die einzelnen Rahmen dar (Zeitindex), wobei die Reihen der Frequenzanteile (Frequenzindex) entsprechen. Anschließend wurde die Datei mit MATLAB gelesen und geplottet. Abbildung 5.9 zeigt eine farbcodierte Darstellung dieser Matrix:

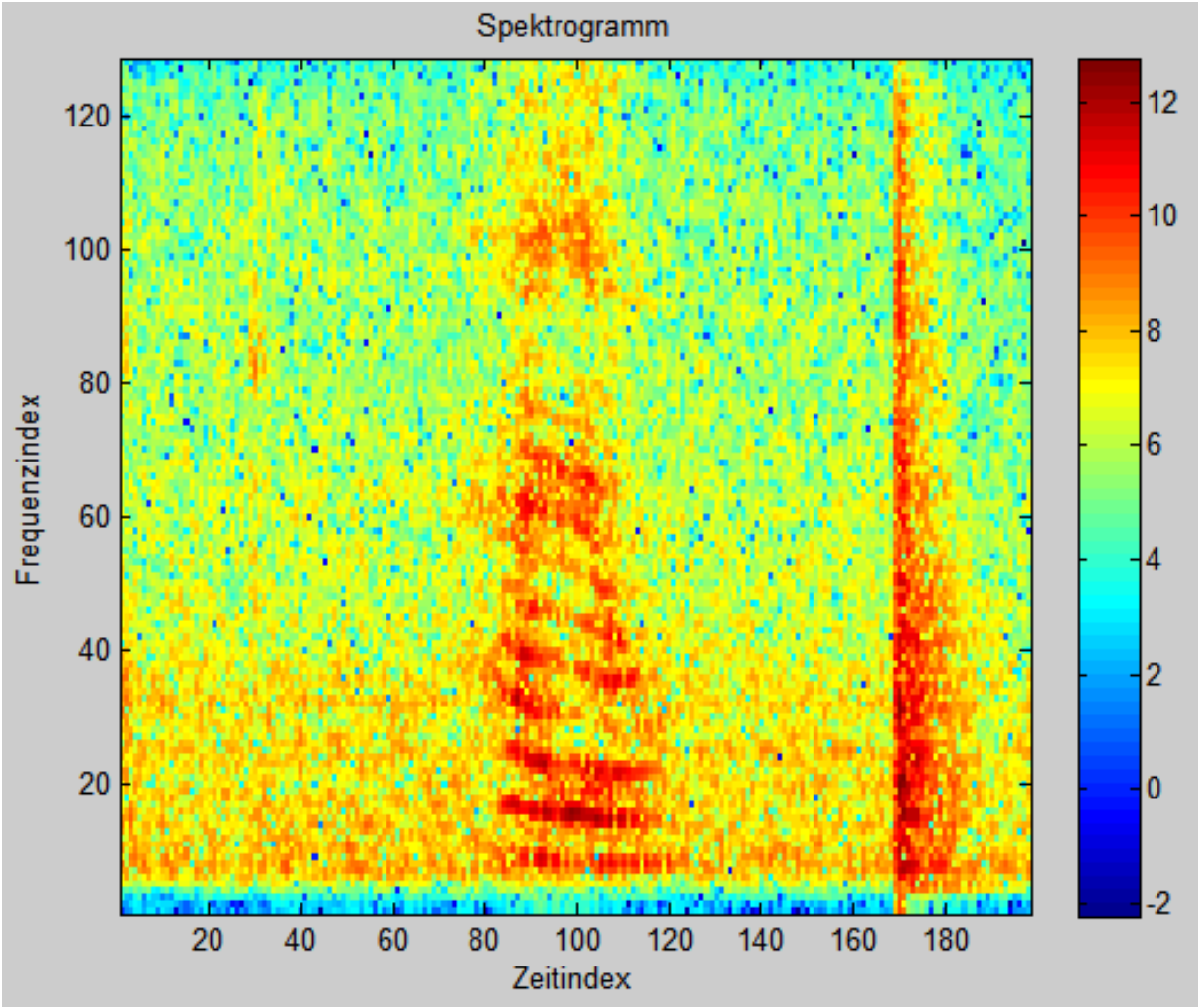

Abbildung 5.9: Spektrogramm („Hallo“ + Klatschen)

Hier ist die zeitliche Entwicklung aller Frequenzkomponenten des Signals zu sehen. In diesem Fall wurde ein „Hallo“ und folglich ein Klatschen aufgenommen. Dabei wird deutlich, dass ein Klatschen ein sehr breitbandiges Signal erzeugt, in welchem alle Frequenzen erhalten sind. Ein „Hallo“ hingegen enthält nur bestimmte und wenige Frequenzanteile. Es wurde festgestellt, dass die Erkennung auf dieser Analyse erfolgen konnte.

Der Algorithmus berechnet die Summe der Frequenzanteile jeder Rahmen, wie in Abbildung 5.10 zu sehen ist (in diesem Fall wurden zwei Klatschen aufgenommen). Wenn die Summe eine bestimmte Schwelle überschreitet, und der darauffolgende Wert wiederum kleiner ist, wird dies als ein Klatschen erkannt. Anschließend wird für eine Sekunde ein Timer gestartet. Wenn innerhalb dieser Zeit die gleiche Erkennung erfolgt, wird das Signal zum Zurückblättern gesendet. Ansonsten wird das Signal zum Vorblättern gesendet.

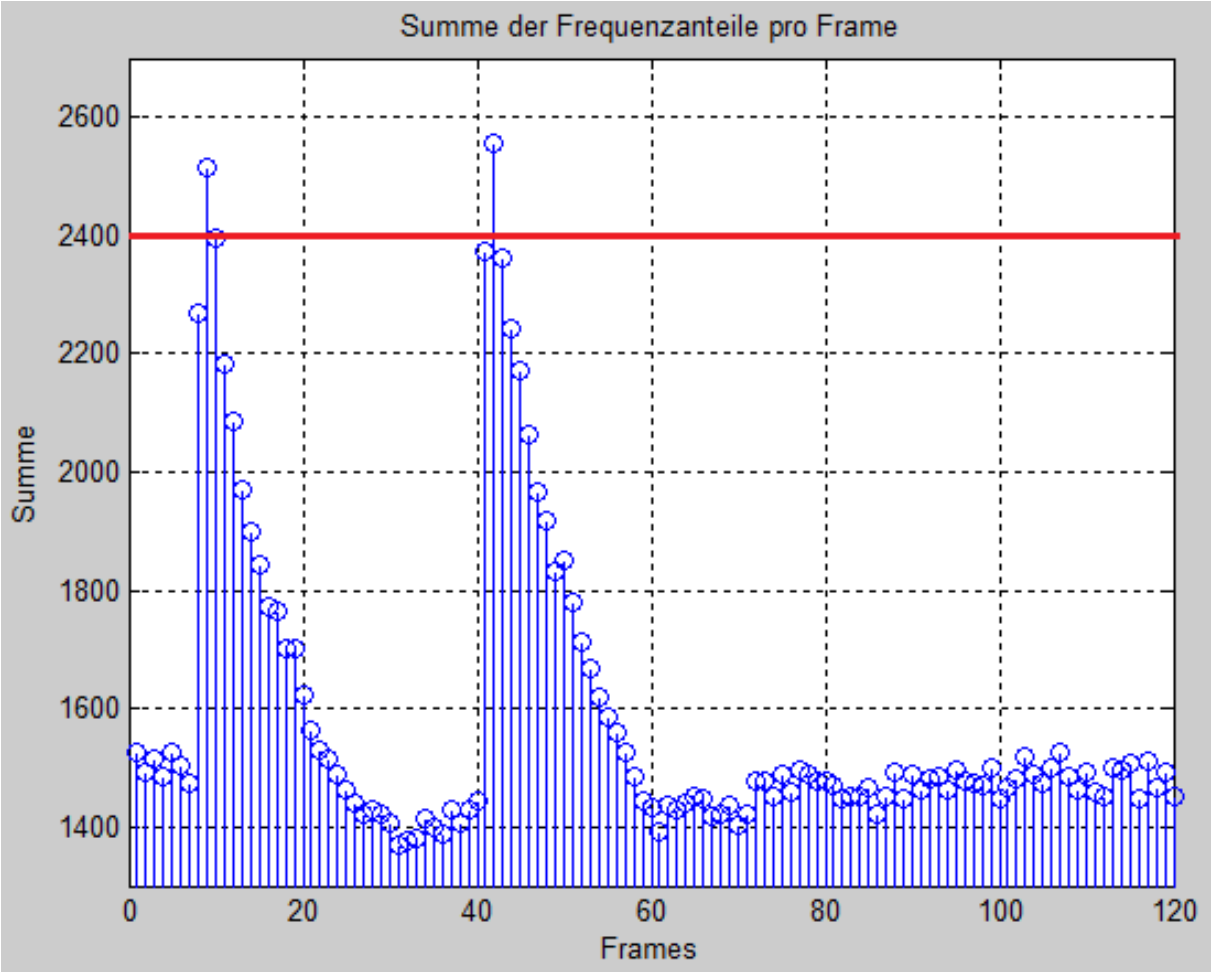

Abbildung 5.10: Summe der Frequenzkomponenten

5.4.1.3 Erweiterungen

Die Steuerung einer Präsentation mittels fließender Sprache ist nur mit umfangreichen Kenntnissen der Wahrscheinlichkeitsrechnung und der Signalverarbeitung möglich. Wenn man die

begrenzte Zeit des Projektes in Betracht nimmt, wird schnell festgestellt, dass eine umfangreiche Spracherkennung durch das Programm in der Kürze der Zeit nur schwer umgesetzt werden kann. Man könnte allerdings Open-Source Bibliotheken verwenden, die beispielsweise bereits ein Erkennungssystem für ein kleines Vokabular implementiert haben.

Allerdings stellt die Applikation einen ersten wichtigen Schritt in diese Richtung dar. Für eine Erkennung im Frequenzbereich bildet das Spektrogramm den Ausgangspunkt für weitere Analysen des Signals.

5.4.2 Gestensteuerung [6]

Die Gestensteuerung hat die Aufgabe Handbewegungen über eine Smartphone-Kamera zu erkennen. Dabei werden zwei Richtungen erkannt und Signale zum vor- und zurückblättert von PDF-Seiten gesendet. Auf Grund der Vorlesung und Recherchen im Internet war schnell klar, dass OpenCV eine geeignete Bibliothek für diese Aufgabe darstellt. Als Qt Basis wurde die Version 5.7 und OpenCV in der Version 3.1 verwendet. Der Code für die Gestenerkennung befindet sich unter folgendem Link. Hierbei wurde ein C++ Klasse „handcontrol“ erstellt, welche in die App eingebunden ist.

5.4.2.1 Auslesen von Videoframes aus einer Kamera

Im Laufe des Projektes stellte sich heraus, dass die Einbindung der Kamera, auf den verschiedenen Plattformen Windows und Android, die größte Herausforderung darstellt. Die Windows Unterstützung wurde hauptsächlich ausgewählt, um den Algorithmus nicht umständlich auf einem Android Gerät jedes mal testen zu müssen. Die Implementierung auf der Windows-Plattform war relativ einfach, da OpenCV schon eine Funktion VideoCapture bietet, welche einzelne Videoframes aus eine Kamera auslesen kann und diese in einer Matrix abspeichert. Diese Methode funktionierte leider nicht auf einen Android-Gerät. Hinweise: nachfolgende Funktionsweisen beziehen sich auf den Entwicklungsstand vom 20.7.2016, ggf. sind schon Bugs behoben oder neue Möglichkeiten zur Kameraauswertung hinzugekommen. Vom Autor wurden unterschiedlichste Varianten zum Auslesen der Kamera über mehrere Stunden untersucht. Von Qt werden hauptsächlich zwei Möglichkeiten für das Auslesen eines VideoFrames angeboten. QAbstractVideoSurface definiert eine abstrakte C++ Klasse welche die Funktion present() beinhaltet, welcher die einzelnen Videoframes nacheinander übergeben werden. Ähnlich verhält es sich mit QVideoProbe wobei man hier die Verbindung über ein Connect() mit Signal und Slot hergestellt werden muss. Eine weitere Möglichkeit besteht seit Qt 5.5 darin, ein Video Filter[7] in QML zu verwenden und mit Hilfe der Klasse QAbstractVideoFilter die einzelnen Videoframes in C++ zu analysieren und ggf. wieder nach QML zu transformieren. Die C++ QCamera funktioniert nicht auf Android-Geräten (1,2), sodass auf das in QML integrierte Camera Objekt zurückgegriffen

[6] Abschnitt von Tim Hebbeler erstellt

[7] https://blog.qt.io/blog/2015/03/20/introducing-video-filters-in-qt-multimedia/

wird. Bei der QAbstractVideoFiler Variante konnten die Videodaten in Android nicht in den CPU-Adressraum gemappt werden. Mit QVideoProbe war dies möglich. Hierbei wird aus QML das QCamera Objekt in C++ adressierbar gemacht und mit dem QVideoProbe verbunden. Seit Qt 5.6 existiert ein VideoOutput Objekt in QML welches das Auslesen und Anzeigen von Video-Frames steuert, sodass das hier angegebene Beispiel noch um ein VideoOutput Objekt ergänzt werden muss. Um keine größeren Unterschiede zwischen dem Algorithmus für die Windows- und der Androidversion zu haben, ist es wünschenswert beide Cameras über Qt auslesen zu lassen. Leider funktioniert der oben für Android vorgestellte Ansatz für Windows nicht. Hier ist ein Workaround mit einer C++ QCamera und dem QAbstractVideoSurface nötig, um ein QVideoFrame zu erhalten. Die neue Variante mit dem „Video Filter“ wurde auch untersucht, hat aber nur auf ein paar Androidgeräten funktioniert QTBUG-47934. Anscheinend gibt es dort noch Fehler in dem Qt-Framework. Unter folgendem Link gibt es eine Auflistung welche Funktionen in dem Qt-Multimedia-Framework-5.7 auf den verschiedenen Plattformen aktuell funktionieren.

Verbesserungen

Die mit Qt 5.5 eingeführten Video Filter scheinen ein gute Weg zu sein, um VideoFrames in Qt analysieren zu können. Leider ist aktuell die Implementierung nicht auf allen Androidgeräten funktional, sodass auf ein Workaround mit QVideoProbe zurückgegriffen werden musste. Diese Variante ist aber leider nicht optimal, da sie Verzögerungen zwischen dem Aufnehmen und dem Aufruf der Funktion present() enthält[8]. Eine weitere Möglichkeit besteht, QML ShaderEffect mit OpenGL zu verwenden. Da die Android Kamera seine Videoframes auf der Grafikkarte in OpenGL Texture vorhält [9], wäre es sinnvoll diese auch dort weiter zu verarbeiten. Das kann seit neuem mit Video Shader Objekten direkt in QML programmiert werden. Außerdem war es dem Autor über QML nicht möglich exakte Auflösungen und Frameraten einzustellen. Anscheinend ignoriert Qt gewisse Parameter auf verschiedenen Platformen oder es stehen nicht alle Einstellung zur Verfügung. Jedenfalls sind diese nicht richtig dokumentiert. Ein Problem ist außerdem, dass es vorkommen kann das Frameraten von 30 fps auf 16 fps einbrechen oder kein reproduzierbares Verhalten zeigen. Dieser Punkt konnte innerhalb der Arbeit leider nicht geklärt werden. Eine andere Möglichkeit, welche nicht weiter verfolgt wurde, wäre über Qt mit den Qt Android Extras die Androidkamera über Java Code in die Qt Anwendung einzubetten, ggf. auch über das vorhandene OpenCV Java Binding.

5.4.2.2 Handerkennungsalgorithmus

Aus Qt liegen die QVideoFrames als RGB (Windows) und als YUV420 (Android) vor. Diese werden als erstes in Grauwertbilder umgewandelt. Bei dem Android QVideoFrame muss keine pixelweise Konvertierung durchgeführt werden, sondern es wird nur der Luminanz Y Teil des Bildes genommen. Nachfolgend wir das Bild auf eine Größe von 640x480 reduziert um die

[8] https://blog.qt.io/blog/2015/03/20/introducing-video-filters-in-qt-multimedia/#comment-1195419
[9] https://blog.qt.io/blog/2015/03/20/introducing-video-filters-in-qt-multimedia/#comment-1195414

weitere Verarbeitung zu beschleunigen. Anschließend wird das Differenzbild zwischen dem aktuellen und vorherigen Grauwertbild berechnet. Hierbei achtet OpenCV selbständig darauf, dass eine Sättigung im Zahlenbereich durchgeführt wird.

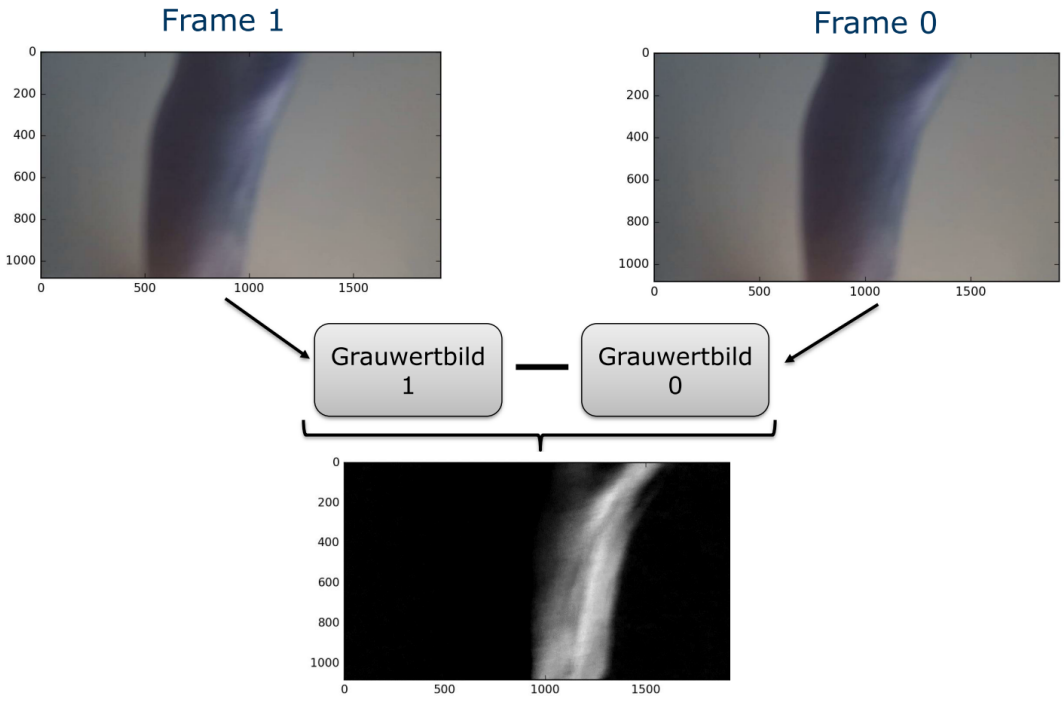

Abbildung 5.11: Differenzberechnung zwischen aktuellem und vorherigen Grauwertbild

Im nachfolgenden Schritt wir mit Hilfe der reduce() Funktion von OpenCV ein Mittelwert über alle Spalten gebildet. Man erhält einen Zeilenvektor wobei jeder Eintrag den Mittelwert eine Spalte repräsentiert. Hierbei kann man schnell erkennen wo sich in der Horizontalen die größten Änderungen ergeben. Das nennt der Autor Histogramm, da es die Häufigkeitsverteilung darstellt.

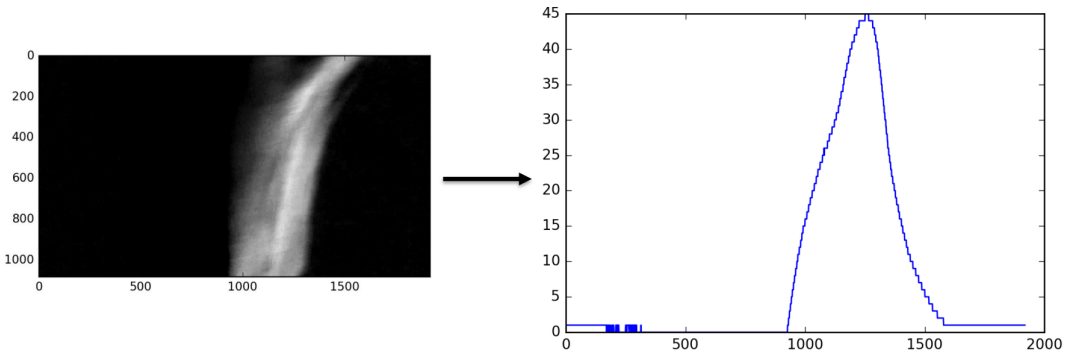

Abbildung 5.12: Berechnung des Histogramms über die Spalten eines Differenzbildes

Untersucht man die Verschiebung des Histogramms über die Horizontalen, kann man Handbewegungen erkennen. Hierbei wird als erstes der Schwerpunkt des Histogramms gebildet. Das geschieht mit Hilfe einer kumulierten Summe über alle Histogrammwerte. Würde man nur den maximalen Wert des Histogramms betrachten, kann bei verrauschten Bildern eine zu große Abweichung auftreten. Vor dem Berechnen des Schwerpunktes wird noch ein konstanter Faktor

subtrahiert, um den ggf. existierenden Rauschteppich zu entfernen. Zur weiteren Fehlerreduktion wurden im folgenden nur diese Schwerpunkte ausgewählt, welche auch mindestens einen gewissen maximalen Histogrammwert aufweisen. Der Wert kann im Sourcecode nachgelesen werden. Um auszuwerten, ob eine PDF vor- oder zurückgeblättert werden soll, untersucht der Algorithmus zeitlich aufeinanderfolgende Schwerpunkte von Histogrammen der Differenzbilder. Er sendet ein Signal, wenn ohne Unterbrechung der Schwerpunkt eine gewisse Zeit in eine der beiden Richtungen gewandert ist. Um den GUI-Thread nicht zu überlasten und um eine flüssige Bedienung der GUI sicher zu stellen, wurde der Algorithmus in ein QThread verschoben. Der so beschriebene Algorithmus benötigt mit 30 Frames pro Sekunde auf Windows ~7ms und auf Android ~10 ms. Somit ist die Echtzeitfähigkeit gegeben. Als Alternative zum dem Differenzbild wurde auch der in OpenCV implementierte Background Substraction Algorithmus (BackgroundSubtractorMOG2) untersucht. Dieser schätzt den Hintergrund über gaußsche Mischungsverhältnisse mit Mittelwert und Kovarianzmatrix. Leider benötigt der Algorithmus viel Berechnungszeit (~14 Windows und ~30 ms Android) und kratzt somit bei Android Geräten an der Echtzeitfähigkeit. Außerdem würde eine solche Implementierung zu viel Energie des Akkus verbrauchen und wurde deshalb verworfen.

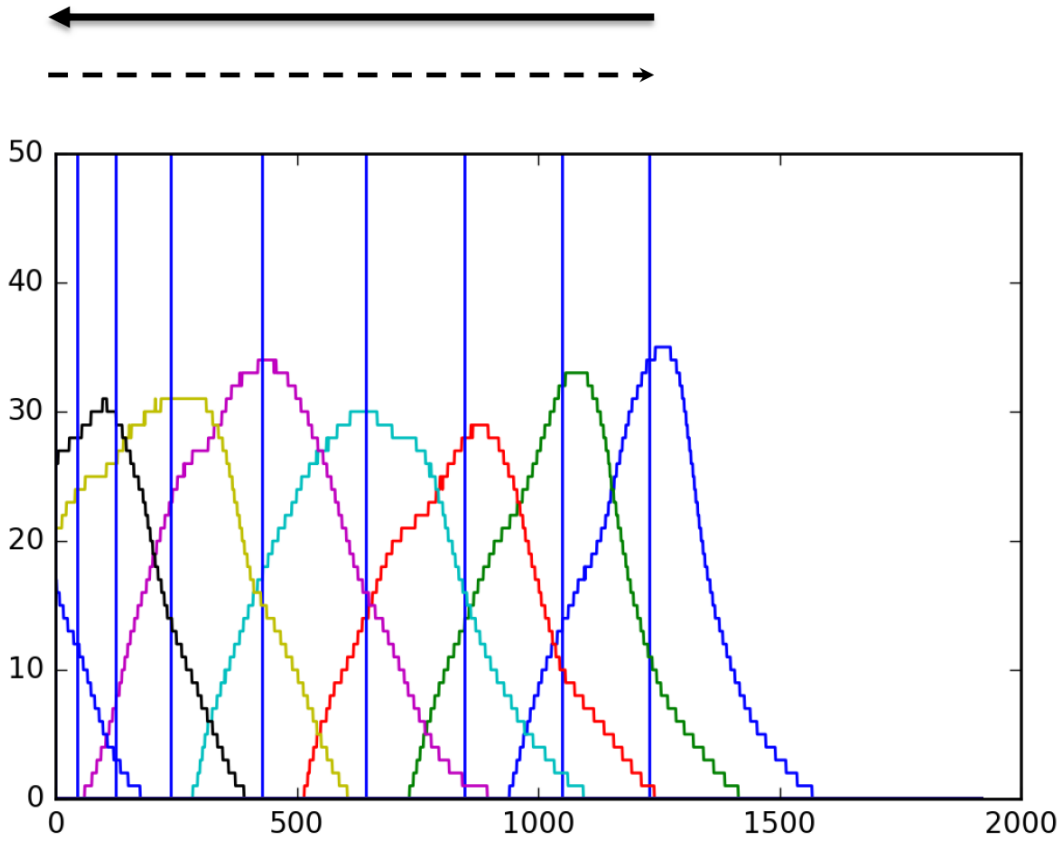

Abbildung 5.13: Histogramme der Differenzbilder mit Schwerpunkt

5.4.2.3 Entwicklung des Algorithmus

Der Algorithmus wurde als erstes in MATLAB mit aufgenommenen Videos entwickelt. Nachher wurde auf Python mit OpenCV Binding umgestellt, um einen besseren Vergleich mit der C++ Version zu gewährleisten. Als IDE für Python wurde Spyder mit dem Anaconda Framework verwendet. Um den Algorithmus separat von der App zu entwickeln wurde ein „test_handcontrol.pro“ Projekt erstellt, welches als eine Art Modultest fungiert. Zu finden in dem schon am Anfang des Kapitels genannten Verzeichnis in Github.

Verbesserungen
Die Verbesserungen zum Auslesen der Kamera wurden schon in einem separaten Abschnitt behandelt. Für den Algorithmus könnte man statt Grauwertbilder auch Bilder aus dem HSV Raum verwenden. Dort könnte man ggf. Helligkeitsveränderung besser abfangen. Außerdem kann es manchmal vorkommen, dass mehrfach Erkennung erfolgen, diese könnten durch einen Timer abgefangen werden. Eine Verschiebung der Berechnung zur Grafikkarte mittels OpenGL oder OpenCL, wie vorher schon erwähnt, könnte die CPU entlasten und ganz andere Möglichkeiten der Analyse des Frames bieten. Bei der Differenzbildung der Frames entstehen positive und negative Abweichungen, diese könnten in einem verbesserten Algorithmus separat berücksichtigt werden.

5.4.3 Beschleunigungssensor[10]

Nach Anwendungsfall 1.4.2 (siehe Seite 2 ff) galt es Blättern durch das Auswerten in den Endgeräten verbauter Beschleunigungssensoren zu ermöglichen.

5.4.3.1 Anforderungen

Zur Auswertung des Beschleunigungssensors müssen dessen Daten ausgelesen werden können.
Relativ zur momentanen Endgerätsorientierung soll das Kippen nach Rechts oder Links erkannt und als Vor- oder Zurück-Blättern interpretiert werden.
Kleinstbewegungen gilt es zu ignorieren.
Mehrfachauslösungen sind zu minimieren.

5.4.3.2 Umsetzung

An dieser Stelle werden kurz die Ansätze der Beschleunigungssensorsteuerung skizziert.
Die Implementierung ist im Quellcode app_gui/pages/PdfSteuerung.qml nachzulesen.

Auslesen des Beschleunigungssensors
Nach Integration von QtSensors kann der Sensortyp Accelerator genutzt und eingestellt werden.

[10] Jens Helge Micke

Unter den Einstellungen können der Sensor aktiviert, die Datenrate[11] eingestellt und das Verhalten bei einem Sensorveränderungsereignis definiert werden.
Die Sensorwerte Der X-, Y- und Z-Achse sind über .reading.x .reading.y .reading.z auszulesen.

Endgerätsorientierung
Zur Erkennung der derzeitigen Endgerätsorientierung stellt Qt QtQuick.Window bereit. Durch die Definition einer Updatemaske können die Wechsel zwischen den Orientierungen 1: Portrait, 2: Landscape, 4: Inverted Portrait und 8: Inverted Landscape erkannt und über Screen.Orientation ausgelesen werden.

Kleinstbewegungen
Kleinstbewegungen werden am einfachsten über eine definierte Mindestauslenkung des Beschleunigungssensors ignoriert.

Mehrfachauslösung
Um Mehrfachauslösungen vorzubeugen und die Beschränkung der nicht immer einstellbaren Datenrate zu umgehen kann mit QtQuick ein Zeitgeber definiert werden der das Blättern nur alle 500ms zulässt.

5.4.3.3 Alternativen

Technisch aufwendigere Alternativen sind denkbar, für den Umfang der Projektarbeit und mit Blick auf die Rechenleistung der Endgeräte jedoch nicht unbedingt Zielführend.
An dieser Stelle seien einige Möglichkeiten motiviert.

Kleinstbewegungen
Kleinstbewegungen und die Eigenschwingung des Endgerätes könnten auch über eine Spektralanalyse des Sensorsignals ignoriert werden.

Mehrfachauslösung
Der Besprochene Mehrfachauslöseschutz könnte auch Global eingesetzt werden.

5.5 Features

- Blättern in einer Pdf ohne dass ein Update auf dem Server erfolgt

INOF ans TEAM:Diese Section ist noch nicht fertig

[11] Nicht von allen Beschleunigungssensoren unterstützt.

5.6 Erweiterungspotential

Einige zusätzliche Funktionalitäten könnten, den Wert der Applikation für Zuhörer und Sprecher steigern.

INOF ans TEAM:Diese Section lade ich später hoch
INOF ans TEAM:Wenn Ihr die Zuverlässigkeit der Gestensteuerung, Kippsteuerung, Audiosteuerung oder die Sicherheit der Kommunikation noch verbessern wollt, könnt ihr ja wie Tim es in euren Abschnitt einfügen. Hier geht es nur um Erweiterungen der allgemeinen mögliche Funktionalität der Clientapp wie Puplikumsfragen oder sowas. Erweiterungen für den Sprecher:

Erweiterungen für den Zuhörer:
- Einstellen der Zeit die vergehen soll bis sich die Applikation synchronisiert

So kann Code eingefügt werden.

Listing 5.1: Kommentierter Start der PWM

```
/*! \brief Starts the PWM
*
* To make sure that the PWM behaves correctly after a Compare Bit Change the PWM is started and reset with a software trigger.
*/
static void vStartPwm( void )
{
tc_start( &AVR32_TC0, PWM_CHANNEL );
tc_software_trigger( &AVR32_TC0, PWM_CHANNEL );
}
```

Kapitel 6

Hardware[1]

Dieses Kapitel behandelt die Eingesetzte Hardware und deren Konfiguration.

6.1 Raspberry Pi 3

Zur Bearbeitung des Projektes wurde an jede Gruppe ein„Raspberry Pi 3“inklusive einer 16GB MicroSD Karte verteilt.
In diesem Projekt wurde der „Raspberry Pi 3“als Server, Verteiler, Zugangspunkt und Primäres Display genutzt.
Dazu mussten Debian, QT5.7, QTCreator 4.0.2, hostapd, dnsmasq, lighttpd, die in Kapitel installiert und, wie in den weiteren Abschnitten erläutert, eingestellt werden.

6.1.1 Debian

Als Betriebssystem kam die neueste für den Raspberry Pi angepasste Debian Version namens „Raspbian“zum Einsatz.
Diese wird als Abbilddatei zur Verfügung gestellt und lässt sich Problemlos auf die Ausgeteilte SD Karte brennen.

6.1.1.1 Hindernisse

Zum Reibungslosen Betrieb des Debian Systems sind einige Hindernisse zu überwinden.

Stromsparmodus
Ein bekanntes Problem der Raspbian Distribution ist, dass ein verdunkeln und Abschalten des Bildschirms nicht immer mit den gängigen Methoden zu unterbinden ist.
Abhilfe schafft das Ausnutzen eines anderen bekannten Fehlers. Dazu muss das Programm

[1] Jens Helge Micke

xscreenserver installiert und in seinen Einstellungen deaktiviert werden. Dadurch beendet sich xscreenserver bei seinem Aufruf selbst und der Bildschirm kann nicht verdunkeln.

HDMI Kompatibilität

Bei Tests mit unterschiedlichen Bildschirmen ist aufgefallen, dass der experimentelle openGL Treiber nicht mit älteren Geräten kompatibel ist.
Um diesen Abzuschalten ist darauf zu achten, dass in der /boot/config.txt die Zeile dtoverlay=vc4-kms-v3d ausgeblendet ist.
Um weiter die Verträglichkeit mit älteren Bildschirmen zu erhöhen lässt sich die Auflösung des HDMI-Ausganges auf VGA begrenzen. Dies ermöglicht auch, dass das Gerät ohne Bildschirm betrieben werden kann.
Dazu muss in der bereits Erwähnten /boot/config.txt der Eintrag hdmi_force_hotplug=1 aktiviert sein.
Für weitere Einstellungsmöglichkeiten sei an dieser Stelle auf die Kommentare in der /boot/config.txt und die Raspbian Dokumentation verwiesen.

6.1.2 QT5.7

Für dieses Projekt wurde die zu dieser Zeit neu herausgekommene QT Version 5.7 benutzt. Da jedoch nur QT 5.3 in den Debian Jessie Bezugsquellen integriert wurde musste diese händisch kompiliert werden.

Vorbereitungen

Da die Dokumentation des Herstellers zur händischen Kompilierung nicht vollständig ist wird diese hier ohne die Integration des WebKit skizziert.
Dieser Vorgang benötigt zwischen 10 und 20 Stunden.

Listing 6.1: Skizze zur Installation von Qt5.7

```
Das Dateisystem auf die gesamte 16GB SD Karte erweitern.
sudo raspi-config
---
Auslagerungsdatei auf 2GB erweitern
sudo nano /etc/dphys-swapfile
CONF\textunderscore SWAPSIZE=2048

sudo dphys-swapfile setup
---
Sicherstellen, dass die Folgenden Dateien gefunden werden:
sudo ln -s /opt/vc/include/interface/vcos/pthreads/vcos_futex_mutex.h /opt/vc/include/interface/vcos/vcos_futex_mutex.h
sudo ln -s /opt/vc/include/interface/vcos/pthreads/vcos_platform.h /opt/vc/include/interface/vcos/vcos_platform.h
sudo ln -s /opt/vc/include/interface/vcos/pthreads/vcos_platform_types.h /opt/vc/include/interface/vcos/vcos_platform_types.h
sudo ln -s /opt/vc/include/interface/vmcs_host/linux/vchost_config.h /opt/vc/include/interface/vmcs_host/vchost_config.h
---
Sicherstellen, dass die richtigen openGLES Treiber geladen werden:
sudo mv /usr/lib/arm-linux-gnueabihf/libEGL.so.1.0.0  /usr/lib/arm-linux-gnueabihf/libEGL.so.1.0.0.backup
sudo mv /usr/lib/arm-linux-gnueabihf/libGLESv2.so.2.0.0 /usr/lib/arm-linux-gnueabihf/libGLESv2.so.2.0.0.backup
sudo ln -s /opt/vc/lib/libEGL.so /usr/lib/arm-linux-gnueabihf/libEGL.so.1.0.0
sudo ln -s /opt/vc/lib/libGLESv2.so /usr/lib/arm-linux-gnueabihf/libGLESv2.so.2.0.0
---
Laden und entpacken von QT5.7.0
mkdir /home/pi/opt
cd /home/pi/opt
wget http://download.qt.io/official_releases/qt/5.7/5.7.0/single/qt-everywhere-opensource-src-5.7.0.tar.gz
mkdir -p /home/pi/opt/qt-everywhere-opensource-src-5.7.0
```

```
ln -s /home/pi/opt/qt-everywhere-opensource-src-5.7.0 /home/pi/opt/qt5
tar zxvf /home/pi/opt/qt-everywhere-opensource-src-5.7.0.tar.gz
---
Pfade anpassen
nano /home/pi/setup_qt.sh
export LD_LIBRARY_PATH=/usr/local/qt5/lib
export PATH=/usr/local/qt5/bin:$PATH

nano /home/pi/setup_general.sh
export PKG_CONFIG_PATH=/usr/lib/arm-linux-gnueabihf/pkgconfig
---
und automatisch laden
echo source /home/pi/setup_qt.sh >> /home/pi/.bashrc
echo source /home/pi/setup_qt.sh >> /home/pi/.profile
echo source /home/pi/setup_general.sh >> /home/pi/.bashrc
echo source /home/pi/setup_general.sh >> /home/pi/.profile
---
Abhaengigkeiten installieren:
sudo apt-get install libfontconfig1-dev libdbus-1-dev libfreetype6-dev libudev-dev libicu-dev libsqlite3-dev \
 libxslt1-dev libssl-dev libasound2-dev libavcodec-dev libavformat-dev \
libswscale-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev gstreamer-tools gstreamer0.10-plugins-good \
 gstreamer0.10-plugins-bad libraspberrypi-dev \
libpulse-dev libx11-dev libglib2.0-dev libcups2-dev freetds-dev libsqlite0-dev libpq-dev \
libiodbc2-dev libmysqlclient-dev firebird-dev libpng12-dev libjpeg62-turbo-dev libgst-dev libxext-dev libxcb1 \
 libxcb1-dev libx11-xcb1 \
libx11-xcb-dev libxcb-keysyms1 libxcb-keysyms1-dev libxcb-image0 libxcb-image0-dev libxcb-shm0 libxcb-shm0-dev \
 libxcb-icccm4 libxcb-icccm4-dev libxcb-sync1 \
libxcb-sync-dev libxcb-render-util0 libxcb-render-util0-dev libxcb-xfixes0-dev libxrender-dev libxcb-shape0-dev \
 libxcb-randr0-dev libxcb-glx0-dev libxi-dev \
libdrm-dev libinput-dev mtdev-tools libmtdev-dev libproxy-dev libts-dev pkg-config pulseaudio libxcb-xkb-dev \
 libxkbcommon-dev libxkbcommon-x11-dev \
libharfbuzz-dev gperf bison flex cmake cmake-data libatspi-dev libxcb-xinlibxcb-xinerama0 libxcb-xinerama0-dev \
libcap-dev libtiff5-dev libwebp-dev libmng-dev \
libjasper-dev libjpeg-dev ruby libxcomposite-dev libxdamage-dev libxrandr-dev libxtst-dev libpci-dev libnss3-dev \
libxss-dev libegl1-mesa-dev libgles2-mesa-dev \
libglu1-mesa-dev "^libxcb.*" build-essential
---
Bauen
cd /home/pi/opt/qt-everywhere-opensource-src-5.7.0
sudo mount --bind /opt/vc/include /usr/local/include
./configure -v -opengl auto -tslib -force-pkg-config -device linux-rpi3-g++ -device-option CROSS_COMPILE=/usr/bin/ -opensource
       -confirm-license -optimized-qmake -reduce-exports -release -qt-pcre -make libs -make tools -skip qtwebengine -nomake
      examples -no-use-gold-linker -prefix /usr/local/qt5
make -j3
---
Falls dies Fehlschlagen sollte muss gcc6 ueber die Stretch Quellen nachinstalliert werden.
```

Einschränkungen

Unter QT 5.7 funktioniert openGL und openGLES auf dem „Raspberry Pi 3“nur mittelmäßig und ausschließlich im Vollbild.

Alternativen

The Qt Company bietet für zahlende Kunden ein funktionierendes QT5.7 Abbild mit dem der „Raspberry Pi 3“über das Netzwerk als direktes Ziel in QTCreator eingebunden werden kann.
Alternativ lassen sich auch über andere Umwege die hier nicht beschritten wurden Programme für die ARMv8 Architektur des „Raspberry Pi 3“Crosskompilieren.
Ausweichen auf die Debian Stretch Quellen liefert vollfunktionstüchtige Versionen von QT 5.6 und QTCreator 4.0.2

6.1.3 WLAN-Accespoint

Da der„Raspberry Pi 3“mit einem AP-Mode fähigem WLAN-Modul ausgestattet ist sei an dieser Stelle die Inbetriebnahme als WLAN-Accesspoint mit hostapd und dnsmasq kurz dargestellt.

Listing 6.2: WLAN-Accespoint

```
Installation von hostapd
sudo apt-get install hostapd
sudo nano /etc/hostapd/hostapd.conf

interface=wlan0
driver=nl80211
ssid=Presentao
channel=3
hw_mode=g
wmm_enabled=1
country_code=DE
ieee80211d=1
ignore_broadcast_ssid=0
auth_algs=1
wpa=2
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
wpa_passphrase=raspberry
---
Konfiguration Zuordnen
sudo nano /etc/default/hostapd

DAEMON_CONF="/etc/hostapd/hostapd.conf"
---
Routerfunktion einrichten
sudo apt-get install dnsmasq
sudo nano /etc/dnsmasq.conf

interface=wlan0
no-dhcp-interface=eth0
dhcp-range=192.168.1.2,192.168.1.254,1h
dhcp-option=option:dns-server,192.168.1.1
---
Portforwarding
sudo nano /etc/network/interfaces

auto lo
iface lo inet loopback
auto eth0
allow-hotplug eth0
iface eth0 inet manual
auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
address 192.168.1.1
netmask 255.255.255.0
up /sbin/iptables -A FORWARD -o eth0 -i wlan0 -m conntrack --ctstate NEW -j ACCEPT
up /sbin/iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
up /sbin/iptables -t nat -F POSTROUTING
up /sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
up sysctl -w net.ipv4.ip_forward=1
up sysctl -w net.ipv6.conf.all.forwarding=1
up service hostapd restart
up service dnsmasq restart
---
Neustarten
```

6.1.4 Klientenverteilung

Für die Verteilung des Klienten wurde lighttpd installiert und eine minimale index.html Datei die auf entsprechende Klientenbinärdateien verweist geschrieben.

6.1.5 Presentao-Server

Der Projektserver wurde direkt auf dem „Raspberry Pi 3“kompiliert und über eine einfache Schleife in den Autostart integriert.

Kapitel 7

Zusammenfassung und Ausblick[1]

Das Projekt verlief mit seinen Höhen und Tiefen erfolgreich.
Am Ende Stand ein funktionstüchtiges Produkt.

7.1 Gelerntes

Bei der Bearbeitung des Projektes wurden viele neue Erfahrungen gesammelt.

- Entwicklung eingebetteter System
- Projektierung von Anfang bis Ende
- Erste Begegnung mit QT
- Entwickeln für Android, GNU/Linux, Windows
- Arbeiten mit Versionskontrollsoftware
- Arbeit im Team
- Kennenlernen von Server/Client Strukturen
- Angewandte Signalverarbeitung
- Kreative Problemlösungen
- Umgang mit Frustration und Erleichterung

[1] Jens Helge Micke

7.2 Weitere Möglichkeiten

Während der letzten Phasen des Projektes wurden weitere Einsatzmöglichkeiten des Produktes offensichtlich.

Digital Signage/Marquee
Mit der kleinen Erweiterung um einen automatisch Blätternden Klienten kann das Projekt als Digitale Werbefläche umfunktioniert werden.

Medienverbreiter und HotSpot
In seiner jetzigen Form könnte das Produkt auch als HotSpot und Menükarte oder Verbreitungsplatform für Bücher im Gastronomiebereich dienen.
Mit Erweiterung des Servers auch direkt als Bestellplatform.

DeadDrop, GeoCache
Solarbetrieben in der Wildnis oder auf privaten Grund kann das Produkt auch als digitale Schatztruhe für kleine und große Abenteurer dienen.

7.3 Schlusswort

Gruppe 5 dankt Herrn Dr. Wolfgang Theimer für diese Möglichkeit Praxiserfahrungen zu sammeln.

Literaturverzeichnis

[1] *Harding Battery Handbook For Quest® Rechargeable Cells and Battery Packs*. January 2004

[2] DARCY, Eric C.: *INVESTIGATION OF THE RESPONSE OF NIMH CELLS TO BURP CHARGING*, University of Houston, Diss., 1998