

# KPMI Projektdokumentation

Florian Röder, Yorick Behme, Dieter Nachname :)

9. Februar 2024

## Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>2</b>
1.1 Projekthintergrund . . . . .	2
1.2 Projektüberblick . . . . .	2
1.3 Arbeitsteilung / Projektmanagement . . . . .	2
<b>2 Hardwareprototyping</b>	<b>3</b>
2.1 Konzeptfindung . . . . .	3
2.2 Konzeptentwicklung . . . . .	3
2.3 Konzeptrealisierung . . . . .	5
2.4 Konzeptfinalisierung . . . . .	6
2.5 a . . . . .	7
<b>3 Smartwatch-App Prototyp</b>	<b>8</b>
3.1 Ideenfindung . . . . .	8
3.2 Lo-Fi-Prototyp . . . . .	8
3.3 Tizen . . . . .	8
3.4 Erstfassung der Smartwatch-App . . . . .	8
3.5 Websockets . . . . .	9
3.6 Zweitfassung der Smartwatch-App . . . . .	10
3.7 Systemintegration . . . . .	10
<b>4 Mixed Reality Umsetzung</b>	<b>11</b>
4.1 Motivation und Idee . . . . .	11
4.2 Grundkonzept . . . . .	11
4.3 Gestaltung . . . . .	12
4.4 Message Handling . . . . .	12
4.5 Entwicklungsprozess . . . . .	12
4.6 Endergebnis und Bedienung . . . . .	13

# 1 Einleitung

## 1.1 Projekthintergrund

Das Projekt fand im Rahmen des Moduls INF-B-490 Komplexpraktikum Medieninformatik-Projekt im Zeitraum des Sommersemesters 2023 und Wintersemesters 2023/24 statt. Dabei diente das Sommersemester zum Einarbeiten in die jeweiligen Technologien via 'hands-on-Seminaren' in welchen jeweils ein Schwerpunkt beleuchtet wurde. Das Wintersemester 2023/24 diente der prototypischen Realisierung der Projektziele.

Zielstellungen der Lehrveranstaltung beinhalten das praktische Kennenlernen von Mixed-Reality-Technologien und Physical Computing Ansätzen via Konzeption und Realisierung von Komponenten sowie Funktionen einer kombinierten Anwendung innerhalb eines 2–3-köpfigen Teams. Genauer beschrieben ist das Ziel ein physisches Mensch-Computer-Interface, welches am Körper getragen werden kann, engl.: "Wearable" unter der Verwendung von e-Textile Sensoren prototypisch umzusetzen und dieses in eine Mixed-Reality beziehungsweise Augmented-Reality Applikation einzubinden.

## 1.2 Projektüberblick

Unser Projekt drehte sich um die Realisierung eines 'smartwatch-bracelet' Prototyps. Zentrale Ziele waren dabei die Erweiterung sowie Kombination der Eingabemodalitäten herkömmlicher 'Smartwatches'. Die Idee: Uhrenarmbänder, welche für gewöhnlich nur den Zweck verfolgen, die Uhr am Armgelenk zu befestigen, in die Interaktion mit der Smartwatch einzubinden und somit neue Interaktionsschnittstellen zu erforschen. Der Armband-Prototyp hat die Zielstellung auch ohne die Mixed-Reality Komponente eine Grundlage für eine erweiterte Smartwatch-Eingabeschnittstelle zu bilden.

Das Projekt lässt sich in 3 Teile unterteilen: Armband, Uhr, Augmented-Reality.

## 1.3 Arbeitsteilung / Projektmanagement

@TODO hier reinschreiben was ihr gemacht habt

Die 3 Bereiche wurden daraufhin auf die Gruppenmitglieder aufgeteilt. Yorick beschäftigte sich mit der Smartwatch App , sowie Teilen der Augmented Reality Visualisierung. Dieter ist für die Augmented Reality Visualisierung zuständig und Florian beschäftigte sich mit dem Entwurf und Design des Uhrenarmbands, die Verarbeitung der Sensordaten auf Arduino Seite sowie dessen Gegenpart auf der Unity Seite, sowie der Präsentation des Prototypen.

## 2 Hardwareprototyping

### 2.1 Konzeptfindung

Nach dem Beschluss sich als Team für einen Prototypen eines Uhrenarmbandes zu beschäftigen wurden verschiedene Konzepte entwickelt, um piezoresistive Drucksensoren und kapazitive Berührungssensoren in den Formfaktor Uhrenarmband zu integrieren. Die Rahmenbedingungen der Anforderungsanalyse waren möglichst ergonomische Interaktionsmodalitäten zu integrieren, mit dem Kerngedanken der von Smartwatches bekannten Interaktionen via Tastenbetätigungen und Wischbewegungen. Daraus resultierte neben anderen Entwürfen unter anderem der folgende Entwurf (siehe Abbildung 1).



Abbildung 1: Konzept Armband

Dieses Konzept sieht vor, ähnlich wie in einem metallenen Armband die Drucksensoren auf die einzelnen Glieder des Armbandes zu verteilen und als Buttons zu verwenden. Zudem soll die Seite des Armbandes als Slider via Berührungssensoren Wischbewegungen unterstützen.

Man entschied sich mit den Kerngedanken des Konzepts fortzufahren. Die nächste Frage, welche Beantwortung finden musste war, ob ein bereits existierendes Uhrenarmband für den Prototypen modifiziert werden soll, oder ob man selbst von Grund auf einen Prototyp entwickeln möchte? Es wurde sich festgelegt, nicht bereits existierende Armbänder bzw. Uhrenarmbänder zu modifizieren, sondern ein Prototyp von Grund auf zu entwerfen. Insbesondere für inkrementelle Änderungen oder schnelles Testen von Ideen bietet dieses Vorgehen mehr Flexibilität innerhalb des Prozesses. Unter anderen war diese Entscheidung auch dadurch begründet, den finanziellen Faktor möglichst gering zu halten, sowie dass keiner der Gruppenmitglieder adäquates Werkzeug zur Bearbeitung von Leder besitzt oder Metall besitzt. Das Armband der Galaxy Watch zu verwenden wurde auch schnell ausgeschlossen.

Im nächsten Schritt wurden die Zuständigkeiten der Bearbeitung der einzelnen Projektbereiche festgelegt und das restliche Hardwareprototyping unter Sektion 2 wurde von Florian übernommen.

### 2.2 Konzeptentwicklung

Abbildung 2 wurde als Konzeptzeichnung für die Anordnung der jeweiligen Sensoren entworfen. Kerngedanke dabei ist die jeweiligen Stoffsichten als enkapsulierendes Element zu sehen. Das Konzept sieht vor auf die Innenseite der Stoffsicht die jeweiligen Sensoren und deren Stromversorgung unterzubringen. Dabei wären auf Abbildung 2 die untere Stoffsicht abgebildet, welche alle Sensoren und Kabel für das Layer enkapsuliert, in diesen Fall alle Sensoren und deren Kabel zu den Analogpins bzw. GND.

Die entgegengesetzte Stoffschicht würde somit die Stromversorgung von Arduino zu den Sensoren darstellen. Es musste nun geklärt werden welche Materialien zur Entwicklung infrage kämen. Die Anforderungen an das Material waren: Der Stoff des Armbandes darf nichtleitend sein und muss sich nähen lassen, zudem sollte er verformbar sein, um Bewegungen der Uhr am Armband standzuhalten, jedoch nur minimal dehnbar und natürlich den Formfaktor umsetzen können. Nach Initialrecherche für infrage kommende Materialien wurde schnell klar, Mantelstoffe wie (gekochte) Wolle mit variablem Verhältnis von Naturfaser zu Kunstoff (oft 80% Schurwolle, 20% Polyamid) gibt es in unterschiedlichen Stärken und wären ideal geeignet für die Umsetzung. Preislich jedoch sprenge das den Rahmen als Substitution bat sich Filz an, Kompromisse müssen damit im Erscheinungsbild in Kauf genommen werden, jedoch erfüllt Filz alle Anforderungen und bieten den großen Vorteil die Schnittkante des Stoffes muss wie bei der Wolle nicht nachbearbeitet werden.

Zudem bietet Filz weitere Vorteile, die leitenden Kabel können halb in den Stoff eingenäht werden, ohne den Stoff vollständig zu durchdringen. Somit sind ungewollte Kurzschlüsse noch einfacher vermeidbar. Zudem sind die Kabel der Stromversorgung auf der entgegengesetzten Seite zu den Kabeln welche von Sensor zu Arduino verlaufen untergebracht, die piezoresistiven Sensoren haben lediglich Kontakt zu beiden Kabeln (Stromversorgung auf der Oberseite und Sensor zu Arduino auf der Unterseite).

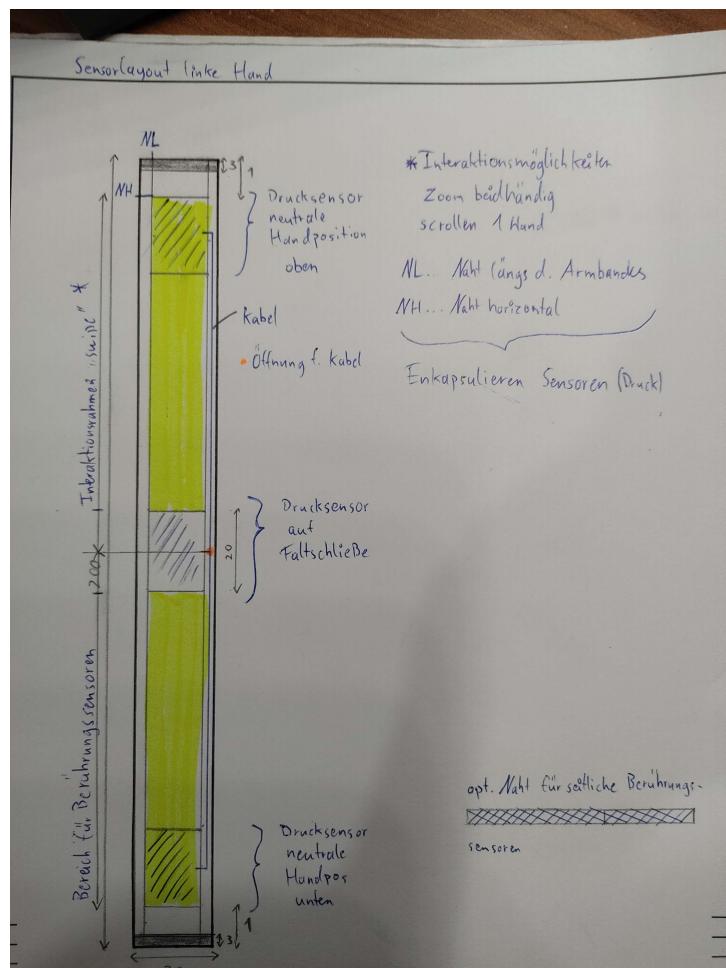


Abbildung 2: Konzept Armband Layout

## 2.3 Konzeptrealisierung

Im Anschluss an die konzeptuelle Ausarbeitung erfolgte die Umsetzung des ersten Prototypen siehe Abbildung 3. Dabei wurden zunächst die verschiedenen Schichten des Stoffes mit Sensoren und der dazugehörigen Stromversorgung umgesetzt.

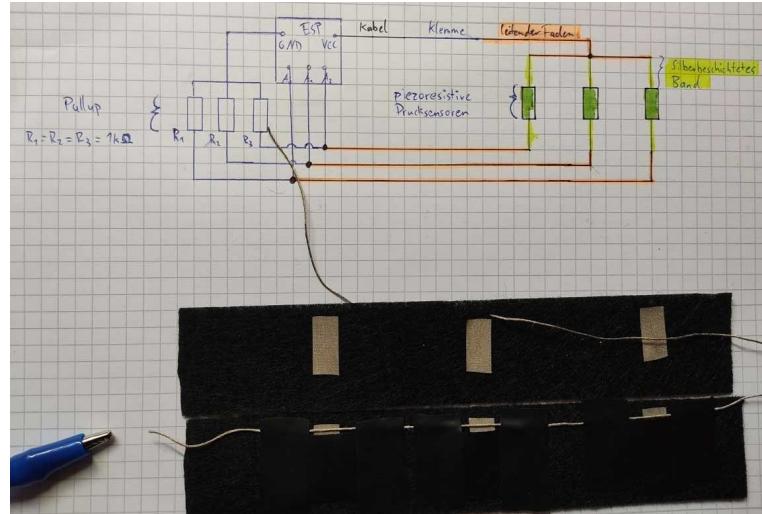


Abbildung 3: "proof of concept" Prototyp

Nachdem das erste "Proof of Concept" Armband mit Sensoren bestückt wurde und der Adafruit ADS1015 zur Verfügung stand wurde die Schaltung umgesetzt und der Arduino Code entwickelt. Hierbei sollte möglichst wenig Arbeit vom ESP selbst übernommen werden und der Großteil der Verarbeitung der Werte auf RCV Seite umgesetzt werden. Der Code ist dafür branchless und möglichst simpel gehalten. Dabei fielen beständige Verhaltensweisen auf: der ADS liefert Analogwerte Bereich von  $x_{ADS} \in \{0, \dots, 1100\}$ , währenddessen die ESP Analogwerte im Bereich  $x_{ESP} \in \{0, \dots, 1023\}$  liegen. Zudem haben die Analogwerte des ADS in zufälligen Verhalten "Noise"  $2^{16}$  angenommen. Abbildung 4 beschreibt die Vorgehensweise, diese Einzelheiten anzugeleichen.

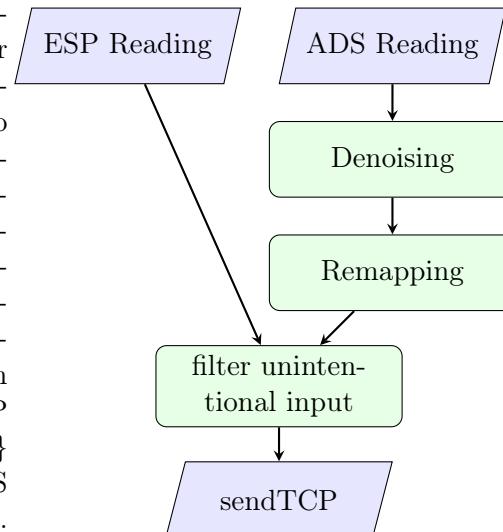


Abbildung 4: Arduino Datenpipeline

In erster Iteration waren auch die Timer für das detektieren multipler Eingaben auf Arduino Seite implementiert, die Behandlung der einzelnen Eingaben auf Unity-Seite wurde auch umgesetzt und getestet. Die Behandlung der Eingaben erfolgt Statebasiert, für jeden Input gibt es einen Handler, der für jeden von der Uhr gesendeten State also welche App so eben offen ist den korrespondierenden Buttonhandler aufruft. Beim Testen entschied man sich als Gruppe auch für eine statische Präsentation des Armbandes an einem Schaufensterpuppenarm. Dazu wurde ein 3D Modell eines Armes abgeändert und 3D gedruckt.

Beim Testen stellte sich zudem heraus, dass das silberbeschichtete Band, sowie das leitende Garn ohne Probleme in Kombination funktionierten. Der piezoresistive Stoff wies bei Positionierung des Silberbandes übereinander das erwartete Verhalten auf, falls jedoch das Band leicht seitlich auf den Sensor drückte, waren die Werte nicht innerhalb des Erwartungsrahmens. Somit war klar in den nächsten Schritten der Entwicklung musste Wert auf die korrekte Positionierung der Sensoren und des Silberbandes geachtet werden. Der Code für Unity funktionierte auch, womit ich die Sensorik als funktionstüchtig betrachten konnte.

As nächste Iterationen wurde mit der Anzahl der Sensoren variiert, um Sliderbewegungen über weitere Drucksensoren zu realisieren, die dann innerhalb einer Schicht des Armbandes untergebracht werden können. Idee war wie im anfänglichen Konzept 2 weitere Drucksensoren zwischen die bereits existierenden Sensoren zu nähen. Damit würden Swipebewegung von den Rändern des Armbandes hin zur Mitte beabsichtigt.

Es stellte sich nach testweiser Befestigung des Armbandes am Arm heraus, dass die Erkennung der Sequenz der Berührung der Sensoren relativ ungenau war, auch nach verschiedenen Softwaretechnischen Änderungen war dies nicht gut zu beheben, die Auflösung der Sensoren bzw. die Genauigkeit war nicht in der Lage die Swipebewegung in allen Fällen zu erfassen. Als Konsequenz daraus wurde innerhalb der Gruppe die Idee unterbreitet diese Swipebewegung mit der "Bezelrotation" der Smartwatch abzubilden, das wurde dann auch umgesetzt und erwies sich schnell als bessere Interaktion.

## 2.4 Konzeptfinalisierung



Abbildung 5: Fitting bracelet to hand3

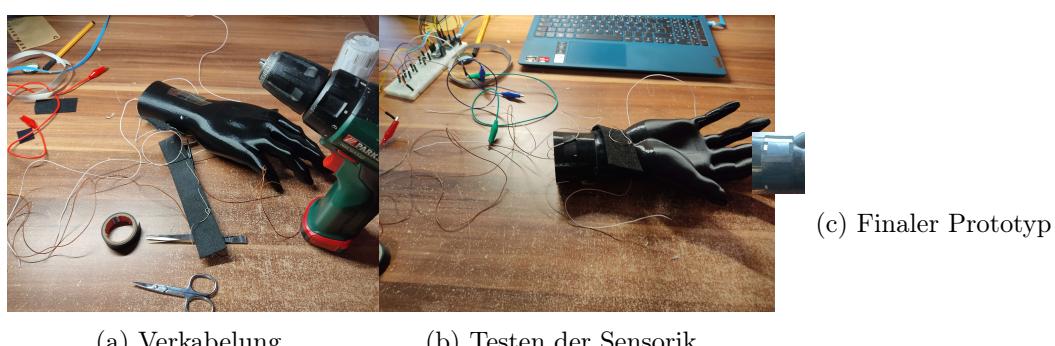


Abbildung 6: Fitting bracelet to hand

Abbildungen 5a bis 6c zeigen den Prozess der Finalisierung des Prototyp, dabei wurde zu Beginn die Sensorik platziert und die korrespondierenden Bohrungen vorgenommen, anschließend wurde die unterste Schicht mit der darüberliegenden befestigt und am Arm festgeklebt.

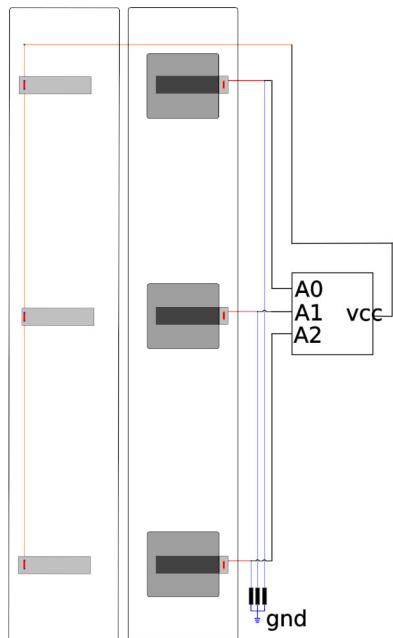


Abbildung 7: ayay

## 2.5 a

## 3 Smartwatch-App Prototyp

### 3.1 Ideenfindung

Zunächst war für unseren Prototyp keine Smartwatch-App vorgesehen. Auch, da uns dafür die nötige Hardware gefehlt hat und eine Beschaffung nicht in unserem Budget liegen würde. Erst im Gespräch mit unseren Betreuern haben wir eine Samsung Galaxy Gear s3 für die Einbindung in unseren Prototypen in Aussicht gestellt bekommen. Daraufhin mussten wir uns zunächst die Frage stellen, welchen Zweck die Uhr in unserem Produktprototyp erfüllen sollte. Nach einem Überlegen haben wir uns dazu entschieden, die Smartwatch als zentrales Element unseres Prototyps zu sehen und die auf der Uhr gewählten Anwendungen durch eine Visualisierung und eine Steuerung durch das Armband zu ergänzen. Somit sollte auf der Uhr eine App laufen, die eine Auswahl aus verschiedenen Anwendungen erlaubt.

### 3.2 Lo-Fi-Prototyp

Als Start in die Umsetzung haben wir zunächst einen Lo-Fi-Prototyp skizziert, um die App, die Sichten und die Navigation zu entwickeln. Dabei ist diese Skizze entstanden (siehe Abbildung 8).

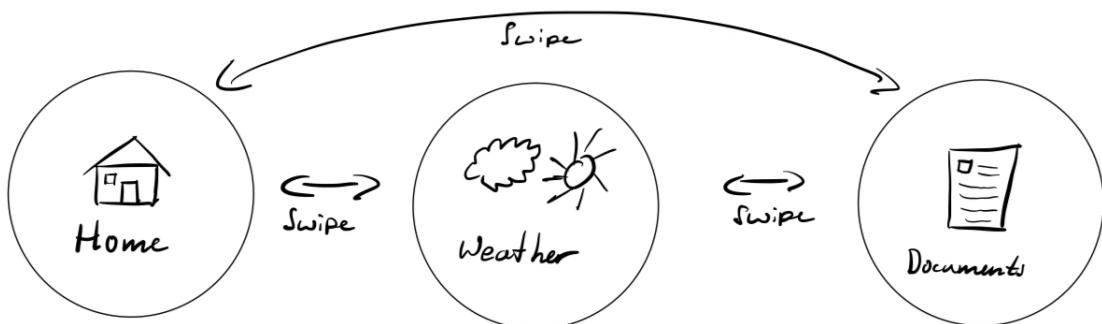


Abbildung 8: Lo-Fi-Prototyp

In diesem Lo-Fi-Prototyp waren zunächst drei Sichten definiert. Die Erste sollte ein Home Bildschirm sein. Dies soll ein Default Case sein, in welchem die Smartwatch-App gestartet wird, aber noch keine Anwendung ausgewählt ist. Durch eine Swipe Geste nach rechts oder links, hier durch die Pfeile gekennzeichnet, sollte zwischen den verschiedenen Sichten gewechselt werden können. Die anderen zwei Sichten sollen symbolisieren, dass hier eine Anwendung hinterlegt ist.

### 3.3 Tizen

Nachdem wir nun also einen ersten theoretischen Prototyp entwickelt hatten, ging es nun an die praktische Implementierung. Dazu mussten wir uns zunächst in die Grundlagen der Tizen Entwicklung einarbeiten und ein neues Projekt aufsetzen. Dabei haben wir uns an dem Watch+Strap Tizen App Projekt der TU Dresden [Quelle: <https://github.com/imldresden/WatchStrap-tizen-app>] orientiert. Nach der Einarbeitung und dem Aufsetzen des neuen Projekts galt es zunächst, die Grundfunktionalitäten der Smartwatch sicherzustellen. Dazu müssten Klassen wie der low-BatteryCheck oder auch die Button-Events für "back" und "main" implementiert werden.

### 3.4 Erstfassung der Smartwatch-App

Als die Grundfunktionalitäten schließlich implementiert waren, konnten wir uns an die Umsetzung unserer App machen. Dazu haben wir zunächst eine einzelne index.html Seite implementiert,

die sich in mehrere sections gegliedert hat. Durch einen Swipe oder durch die Basal Rotation wurde die sectionChange Methode aufgerufen, welche wiederum die nächste section sichtbar gemacht hat.



Abbildung 9: Weather Anwendung in früher Version

Die Abbildung zeigt beispielhaft eine ausgewählte section, die mit dem Wetter-Icon versehen ist. Das Produkt dieses Abschnittes war eine Tizen-App, die durch Swipegesten oder die Basal Rotation gesteuert durch die sections Home, Weather und Documents bewegt werden konnte.

### 3.5 Websockets

Als nächster Schritt stand nun die Verbindung zu unserem Unity-Projekt an. Dabei mussten wir zunächst eine geeignete Technologie für unsere Netzwerkübertragung finden. Der erste Versuch war, eine HTTP-Massage zu verschicken. Dies ließ sich zwar aufseiten der Smartwatch-App als Client gut implementieren, dennoch stellte es sich als umständlich heraus, auf Unityseite einen HTTP-Server zu implementieren. Deswegen entschieden wir uns, die Technologie Websockets umzustellen. Nun können folgende Nachrichten von der Smartwatch-App an die Unity-Anwendung verschickt werden:

1. “0” - Home section / zurück auf Home section
2. “1” - weather section
3. “2” - graph section
4. “3” - documents section
5. “a1” - weather Anwendung gestartet
6. “a2” - graph Anwendung gestartet
7. “a3” - documents Anwendung gestartet
8. “cw” - Clockwise Rotation
9. “ccw” - counterClockwise Rotation

Auf Seiten von Unity gibt es einen mit WebSocketSharp erstellten Websocket-Server. Dieser kann die Nachrichten empfangen und in der Methode StateChange.SetAppState() zur Weiterverarbeitung in Unity zur Verfügung stellen.

### 3.6 Zweitfassung der Smartwatch-App

Nachdem die WebSocket-Kommunikation implementiert war, kam noch der Wunsch auf, auch innerhalb der einzelnen Anwendungen die Basal-Rotation zur Steuerung der Visualisierung nutzen zu können. Deswegen braucht es eine Trennung zwischen der Anwendungsauswahl, die ebenfalls über die Basal-Rotation möglich ist, und der Anwendung selbst. Aus diesem Grund haben wir einen Button auf das jeweilige App-Auswahlfeld gemacht, der eine HTML-Unterseite aufruft, die eigene Scripte lädt und so eine Trennung von der Auswahl einer App und dem in einer App sein ermöglicht.



Abbildung 10: Finale Documents Anwendung

Wie in der Abbildung zu erkennen, haben wir die Anwendungsseite auch farblich an die Visualisierung angelehnt gestaltet, um so auch visuell noch eine Verbindung zwischen der Smartwatch-App und der Unity Visualisierung zu schaffen.

### 3.7 Systemintegration

Die Smartwatch-App gliedert sich in das Gesamtprojekt als Element zur Anwendungsauswahl und zur Steuerung ein. So kann über die App eine Unter-Anwendung Weather, Graph oder Documents ausgewählt und durch einen Button-Klick gestartet werden. Durch die Basal-Rotation können bestimmte Elemente der Visualisierung darüber hinaus auch gesteuert werden. Außerdem kann über den Home Button eine Anwendung wieder geschlossen werden.

## 4 Mixed Reality Umsetzung

### 4.1 Motivation und Idee

Der Umsetzung in einem Mixed-Reality-Umfeld lag der Gedanke zu Grunde, dass es derzeit oft schwierig ist, größere Informationsmengen oder Darstellungen auf modernen Smartwatches zu betrachten. Dies liegt daran, dass diese Geräte aufgrund ihrer Bildschirmgröße stark eingeschränkt sind. Zum Beispiel können längere Texte nur sehr klein und kaum lesbar angezeigt werden oder nur in kleinen Ausschnitten, da ihre Schriftgröße zu hoch skaliert wurde. Zudem lassen sich Bilder und Darstellungen nicht gut betrachten, und viele wichtige Details können unter Umständen unbeachtet bleiben.

Aus diesen Problemen ergab sich die Idee, Smartwatch-Apps durch die HoloLens größer oder detaillreicher darzustellen, da die Mixed-Reality-Umgebung eine fast uneingeschränkte Skalierung von Visualisierungen ermöglicht. Der "virtuelle Bildschirm" würde dabei auf Wunsch die Rolle des physischen Bildschirms übernehmen und die ausgewählte App abbilden. Auch das Armband könnte man so für den Nutzer virtuell darstellen und darauf Informationen abbilden oder seine Textur verändern.

### 4.2 Grundkonzept

Nach der Ideenfindung haben wir uns darauf konzentriert, eine realistische Umsetzung zu planen. Uns war bewusst, dass wir für dieses Projekt keine umfangreiche Kommunikation mit dem Betriebssystem der Smartwatch realisieren könnten, um in Echtzeit Informationen aus aktiven Apps zu erhalten, da uns die Techniken mit denen wir arbeiten würden zu diesem Zeitpunkt noch komplett neu waren. Unsere Mixed-Reality-Anwendung sollte daher lediglich eine theoretische Demonstration sein, basierend auf der Annahme, dass die Uhr und ihr Betriebssystem in Verbindung mit der HoloLens-Kommunikation und unserer Armband-Erweiterung entwickelt worden wären.

Für dieses Vorhaben haben wir eine App entwickelt, die ein App-Menü beinhaltet. Über dieses Menü kann man durch Aufruf eines Icons, das eine theoretische App der Uhr symbolisiert, mit der Unity-Anwendung kommunizieren. Diese soll dann das entsprechende Verhalten ausführen. Die resultierenden Konzeptbilder visualisierten die geplante Bedienung und die Grundideen der Anwendungen (siehe Abbildung 11 und 12).



(a) Kontaktauswahl



(b) Nachrichten-Ansicht

Abbildung 11: Konzept einer Nachrichten-App (in MR)



(a) Wetter-App (MR deaktiviert)



(b) Wetter-Ansicht

Abbildung 12: Konzept einer Wetter-App (in MR)

### 4.3 Gestaltung

Als es darum ging, eine endgültige Gestaltung für das Interface der Mixed-Reality-Darstellung zu entwickeln und mit der Implementierung in Unity zu beginnen, richteten wir unser Augenmerk zunächst auf das Erscheinungsbild der Uhr. Aufgrund des schwarzen Gehäuses und des dunklen Armbands lag es nahe, für den Hintergrund des Interfaces Graustufen und Schwarz zu verwenden. Um weiterhin eine Identifikation mit der Uhr zu gewährleisten, sollte der Kreis als wiederkehrende Form dienen und die Grundlage für die Darstellung bilden (siehe Abbildung 13). Generell sollte die Darstellung jedoch minimal gehalten werden, um den Fokus des Benutzers auf den dargestellten Informationen zu lenken.

Da das Interface in seiner farblosen Form jedoch wenig lebendig erschien, entschieden wir uns, an einigen Stellen farbliche Hervorhebungen zu verwenden. Später haben wir auch das Armband virtuell in Farbe dargestellt, wobei seine Farbe der Identifizierung der aktuell ausgewählten App diente.

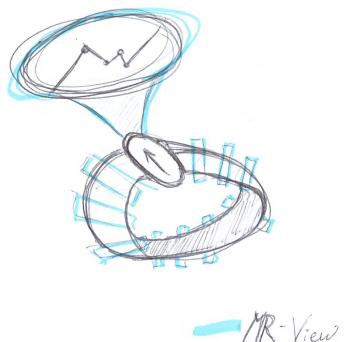


Abbildung 13: Konzept-Skizze

### 4.4 Message Handling

Vor dem Start der Implementierung der Mixed-Reality-Visualisierung mussten zunächst die Nachrichten der Bedienelemente empfangen und verarbeitet werden. Bei den Nachrichten des Arduino orientierten wir uns an der Implementierung in den uns zur Verfügung gestellten Beispieldatenprojekten. Nach dem Empfang einer Nachricht des Arduino wird diese dem entsprechend gedrückten Button zugeordnet, der als `Sensor0/FWD`, `Sensor2/Confirm` und `Sensor4/BWD` bezeichnet wird (siehe Abbildung 15). Unmittelbar nach dem Empfang wird eine Methode aufgerufen, die den `MessageContainer` entpackt, den Button als gedrückt klassifiziert und seinen gesendeten Wert an die allgemeine `MessageHandler`-Methode (`HandleButtonPress`) übergibt. Die Kommunikation mit der Tizen-App erfolgt über eine separate Klasse (`WebSocket_watch_communication`). Diese übergibt die empfangenen Werte einer statischen Klasse (`State Changes`), in der die Werte kategorisiert und in Variablen abgelegt werden. Die Skripte, die diese Werte benötigen, überprüfen Veränderungen der `StateChanges`-Klasse mithilfe der `Update`-Methode und führen bei Bedarf ebenfalls die allgemeine `MessageHandler`-Methode (`Handle ButtonPress`) aus. Die Methode `HandleButtonPress` kann vererbt und überschrieben werden. Je nach aktuell gewählter App, betätigtem Button oder anderen Parametern ruft sie das auszuführende Verhalten auf. In der Basis-Klasse (`HandleSensorData`) dient sie dazu, bei Betätigung eines Buttons die zugehörigen Methoden in den Kindklassen aufzurufen. Von dort aus wird dann das Verhalten der MR-Visualisierung ausgelöst, und die Darstellung entsprechend angepasst.

### 4.5 Entwicklungsprozess

Nachdem wir uns in Unity und C eingearbeitet hatten und die Befehle der Bedienelemente empfangen werden konnten, begannen wir mit der Arbeit an der MR-Visualisierung und ihrem Verhalten bei Eingaben. Um ein Fundament für die Implementierung zu schaffen, fügten wir der Unity-Szene eine Plane hinzu, die als "Bildschirm" dienen sollte, sowie das in Blender erstellte "virtuelle Armband". Die Klassen in den Skripten der `GameObjects` sollten alle von der Basis-Klasse (`HandleSensorData`) erben, die überschreibbare Methoden für alle Eventualitäten enthielt.

Zunächst implementierten wir das Verhalten für das App-Menü, in dem verschiedene Anwendungen ausgewählt werden konnten: Wetter, Dokumente und Graph (Graph sollte dazu dienen

verschiedene Graphen darzustellen). Anfänglich waren Platzhalter-Icons im Menü zu sehen, deren Farbe durch die Armband-Buttons geändert werden konnte, um die Bedienelemente und das Message-Handling zu testen. Durch die äußeren Buttons konnte zwischen den Icons vor und zurück geblättert werden, wodurch sich auch die Armband-Farbe änderte. Das alles bewerkstelligten wir durch simple Textur-Änderungen der `GameObject`-Materialien.

Zur Vollendung des Prototyps implementierten wir die Wetter-App, die sich durch Auswahl des Icons auf der Smartwatch öffnen ließ (siehe Abbildung 14a). In dieser konnte man mithilfe der äußeren Armband-Buttons durch Temperatur- und Niederschlagsinformationen blättern und die dargestellten Wettergraphen durch den mittleren Button vergrößern oder verkleinern.

Bei der Auseinandersetzung mit dem Prototyp fiel uns auf, dass sich die Bedienung der MR-Anwendung noch sehr umständlich und unbequem anfühlte, insbesondere das Blättern mit den seitlichen Buttons. Daher entschieden wir uns, die Bedienung neu zu strukturieren und eine andere Funktion für die seitlichen Armband-Buttons zu finden. Wir wählten die Bezel der Smartwatch für das Blättern, da diese eine schnelle und bequeme Rotation ermöglichten. Die seitlichen Armband-Buttons erhielten eine Funktion für eine sekundäre Darstellung des MR-Interface sowie die Möglichkeit, das Interface nach Bedarf zu deaktivieren oder zu aktivieren. Damit hatten wir also die Kontrolle der Apps, bis auf den mittleren Armband-Button, komplett auf die Smartwatch-Bedienelemente ausgelagert und die Kontrolle der MR-Darstellung auf die seitlichen Armband-Buttons gelegt.

Schließlich implementierten wir die Dokumenten-App, die dem initialen Ziel großer Dokumentdarstellungen (siehe 4.1 Motivation und Idee) am nächsten kam. Nach dem Start der App wurde der Benutzer in ein Auswahlmenü gebracht, in dem er die verfügbaren Dokumente sehen konnte. Nach der Selektion wurde die Datei angezeigt, und der Benutzer konnte ggf. durch verfügbare Seiten blättern. Um die Seitenanzeige klarer zu gestalten, fügten wir dem Armband in der Unity-Szene eine `PageIndicator`-Anzeige hinzu, die bei Bedarf erscheinen konnte (siehe Abbildung 14b). Damit endete die Entwicklungsphase unseres Projektes.

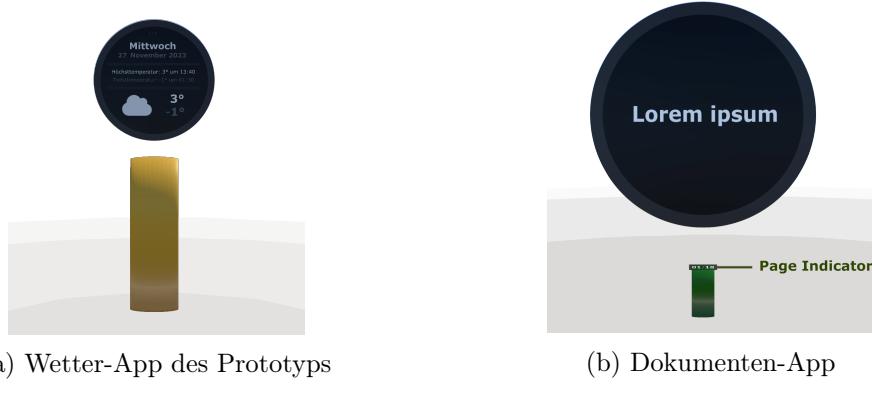


Abbildung 14: Apps zu verschiedenen Phasen der Entwicklung

## 4.6 Endergebnis und Bedienung

Zum Schluss konnte unsere Unity-Anwendung also Nachrichten vom Arduino und der Tizen-Applikation empfangen. Sie beinhaltete das App-Menü, die Wetter-App und die Dokumenten-App, zwei unterschiedliche Darstellungsmöglichkeiten sowie die Funktion, die MR-Darstellungen aus- und anzuschalten. Die Graph-App lässt sich immer noch unimplementiert als Platzhalter im App-Menü finden.

Zur besseren Übersicht bezeichneten wir den mittleren Armband-Button als Confirm-Button und die beiden seitlichen als BWD (backward) und FWD (forward) -Button (siehe Abbildung 15).

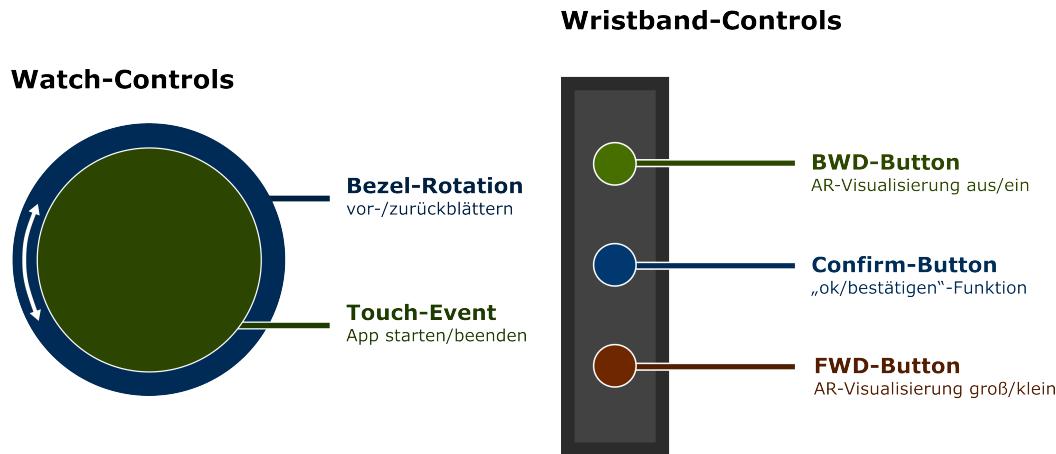


Abbildung 15: Bedienelemente

Im App-Menü kann durch Rotation der Bezel an der Smartwatch eine App gewählt werden, und durch das Drücken des Icons auf dem Display der Uhr kann die App gestartet werden. Bei erneutem Drücken auf den Bildschirm wird die App wieder verlassen, und man gelangt zurück ins App-Menü.

Die Wetter-App zeigt bei Rotation der Bezel verschiedene Wetter-Informationen an. Mit dem Confirm-Button kann man die angezeigte Abbildung vergrößern und wieder verkleinern.

Beim Starten der Dokumenten-App gelangt man zunächst in ein Untermenü, in dem man eines der angebotenen Dokumente mit der Bezel-Rotation auswählen kann. Mit dem Betätigen des Confirm-Buttons kann das gewählte Dokument geöffnet und wieder geschlossen werden. Durch die gewählte Datei kann man dann ggf. ebenfalls mit der Bezel-Rotation durchblättern.

Mit dem FWD-Button kann man jederzeit zwischen der größeren Zweitdarstellung und der normalen Ansicht wechseln.

Der BWD-Button dient dazu, die gesamte Mixed-Reality-Ansicht zu deaktivieren und erneut zu starten (siehe Abbildung 16).

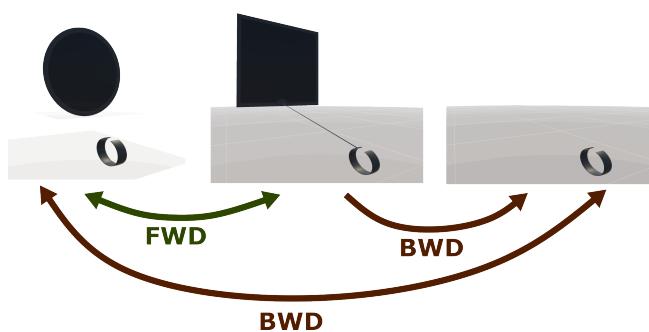


Abbildung 16: Funktion seitlicher Armband-Buttons

## Abbildungsverzeichnis

1	Konzept Armband . . . . .	3
2	Konzept Armband Layout . . . . .	4
3	”proof of concept” Prototyp . . . . .	5
4	Arduino Datenpipeline . . . . .	5
5	Fitting bracelet to hand3 . . . . .	6
6	Fitting bracelet to hand . . . . .	6
7	ayay . . . . .	7
8	Lo-Fi-Prototyp . . . . .	8
9	Weather Anwendung in früher Version . . . . .	9
10	Finale Documents Anwendung . . . . .	10
11	Konzept einer Nachrichten-App (in MR) . . . . .	11
12	Konzept einer Wetter-App (in MR) . . . . .	11
13	Konzept-Skizze . . . . .	12
14	Apps zu verschiedenen Phasen der Entwicklung . . . . .	13
15	Bedienelemente . . . . .	14
16	Funktion seitlicher Armband-Buttons . . . . .	14