

Entwicklung eines Gamification-basierten Unterstützungs- und Motivationsgeräts zur Rehabilitation von Schlaganfall-Patienten

Seminarfacharbeit der KLASSENSTUFE 11/12 (Schuljahre 2017-2019)
am
Albert-Schweitzer-Gymnasium Erfurt, Spezialschulteil

Lukas Rost

Fachbetreuer: Johannes Süpke
Seminarfachbetreuer: Dr. Marion Moor
Außenbetreuer: Hannes Weichel

20. Dezember 2018
Erfurt

Inhaltsverzeichnis

1 Einleitung	2
2 Die Erkrankung Schlaganfall und geeignete Therapiemethoden	3
2.1 Schlaganfall als Krankheitsbild	3
2.2 Rehabilitation und Effektivität von Bewegungsübungen	4
3 Umgesetzte Konzepte	5
3.1 Gamification	5
3.1.1 Grundlegende Mechanismen der Gamification	5
3.1.2 Beispiele für die motivationssteigernde Wirkung	6
3.2 Biofeedback	7
4 Schaltung und Implementierung des Mikrocontroller-Systems	8
4.1 Aufbau der Schaltung	8
4.2 Entwicklung des Programms auf dem Mikrocontroller	8
5 Entwicklung der Begleitapp für Android	10
5.1 Grundlegender Aufbau und Konzept der App	10
5.2 Kommunikation mit dem Mikrocontroller	11
5.3 Konzept und programmiertechnische Umsetzung des Minispiele	11
5.4 Umsetzung der Gamification in der App	13
5.5 Funktionsweise des Benachrichtigungssystems	15
6 Zusammenfassung	16
7 Literatur- und Quellenverzeichnis	17
8 Anhang	20

1 Einleitung

'Die Inzidenz des Schlaganfalls beträgt in Deutschland ca. 180/100.000. Nach Herzerkrankungen und Krebsleiden ist der Schlaganfall die dritthäufigste Todesursache in Deutschland und die häufigste Ursache für Langzeitbehinderung.' (ApoFlex) -> noch ändern

39499 Todesfälle in Deutschland durch Schlaganfall und Folgen 2015 (Destatis)

Große Studien zeigen, dass lediglich 5% der Patienten ihre Arme und Hände wieder uneingeschränkt einsetzen konnten und dass in 20% keinerlei Arm-Handfunktion zurückkehrte. Hingegen werden etwa 75% der hemiparetischen Patienten - selbstständig oder mit Hilfe - gehfähig. 25% bleiben auf den Rollstuhl angewiesen oder sind bettlägrig. (RehabNelles)

2 Die Erkrankung Schlaganfall und geeignete Therapiemethoden

2.1 Schlaganfall als Krankheitsbild

Beim Schlaganfall, auch *Apoplexia cerebri* genannt, handelt es sich allgemein um eine „plötzliche Durchblutungsstörung im Gehirn.“²¹ Durch diese kommt es zu einem regionalen Mangel an Sauerstoff und Nährstoffen, welcher zu einem Absterben von Gehirngewebe führt.

Es existieren zwei mögliche Ursachen: Der ischämische oder Hirninfarkt tritt bei 80 bis 85 % der Fälle auf. In diesem Fall ergibt sich eine mangelnde Durchblutung aufgrund von Gefäßverschlüssen, auch unter dem Begriff Arteriosklerose oder Thrombose zusammengefasst. Folgend kann es dabei zusätzlich auch zu einem Schlaganfall der zweiten Art kommen, dem sogenannten hämorrhagischen Infarkt bzw. der Hirnblutung, die 10 bis 15 % der Fälle zugrundeliegt. Dieser wird durch geplatzte und eingerissene Gefäße verursacht, aus denen Blut ins Hirngewebe austritt. Dieses schädigt durch eine verminderte Sauerstoffversorgung, seinen Druck sowie seine neurotoxische Wirkung das Gehirn. Ein solcher Infarkt kann seinerseits wiederum eine Ischämie verursachen.¹³

Im Vorfeld eines Schlaganfalls treten oft transitorisch-ischämische Attacken, also vorübergehende neurologische Ausfälle, auf. Symptome eines Schlaganfalls reichen von halbseitigen Körperlähmungen über Sprachstörungen und eingeschränktes Sprachverständnis (motorische und sensorische Aphasie) bis zu Sehstörungen und Gleichgewichtsproblemen. Auch Verwirrtheit, Übelkeit und Kopfschmerzen können auftreten.¹³

Zur Erkennung von Schlaganfällen wird meist der FAST-Test benutzt, der fachsprachlich unter dem Begriff „Cincinnati Prehospital Stroke Scale“ bekannt ist.¹⁹ Dieser besteht aus folgenden Punkten:

1. Face/Gesicht: Person kann nur mit einer Gesichtshälfte lächeln
2. Arms/Arme: Unfähigkeit, beide Arme mit nach oben geöffneten Handflächen nach vorne zu strecken
3. Speech/Sprache: undeutliche Aussprache, kann nicht sprechen, versteht nichts mehr

Sollten diese Punkte zutreffen, zählt der vierte Punkt: **T**ime (Zeit). Der Rettungsdienst sollte umgehend verständigt werden, um den Schaden zu begrenzen, denn „Zeit ist Hirn“.^{13†} Die Diagnostik erfolgt in einer geeigneten Klinik mit „Stroke Unit“. Dabei werden meist Verfahren wie Computertomografie und Magnetresonanztomografie genutzt. Die Schwere des Schlaganfalls wird mit Scoresystemen wie der „National Institutes of Health Stroke Scale“ beurteilt.

Risikofaktoren umfassen unter anderem Bluthochdruck, Rauchen, Diabetes sowie Übergewicht und Bewegungsmangel. Dementsprechend wirkt eine gesunde Lebensweise präventiv. Dazu gehören gesunde Ernährung und regelmäßige Bewegung sowie ausreichende Flüssigkeitsaufnahme, aber auch Stressvermeidung.¹³ Es existieren jedoch auch andere, nicht beeinflussbare Risikofaktoren wie Alter, Blutgruppe und genetische Veranlagung.

Zu den Basismaßnahmen bei der Therapie gehört als erstes die Stabilisierung der Vitalfunktionen wie Blutdruck, Puls und Körpertemperatur. Auch sollte der Patient mit erhöhtem Oberkörper gelagert werden. Diese Maßnahmen stabilisieren den Patienten und dienen der Verhinderung eines weiteren Schlaganfalls. Weiterhin kann bei einem ischämischen Infarkt bis zu drei Stunden nach Auftreten des Schlaganfalls abhängig von der Größe des Infarkts eine intravenöse Thrombolyse-Therapie durchgeführt werden, um eventuell verschlossene Blutgefäße wieder zu öffnen. Bei Hirnblutungen dagegen sind operative Behandlungen sinnvoll, beispielsweise zur Hirndruck-Entlastung.¹³

[†]siehe Abb. 2

2.2 Rehabilitation und Effektivität von Bewegungsübungen

Armlähmungen zählen zu den häufigsten Folgen einer Hirnschädigung, wie sie durch einen Schlaganfall hervorgerufen wird.⁷ Aber auch Lähmungen anderer Körperteile oder Aphasie können auftreten.¹⁹ Meist betrifft eine solche Hirnschädigung nur eine Gehirnhälfte, sodass es nur auf einer Körperseite zu Lähmungen kommt.

Die meisten Rehabilitationsmaßnahmen dienen dazu, die Körperwahrnehmung des Patienten zu fördern und verlorene Fähigkeiten zu kompensieren. Dabei sind Ansätze wie die „Constraint-Induced Movement Therapy“ vielversprechend. Bei dieser auch als „Taubische Bewegungsinduktion“ bekannten Methode wird der gesunde Arm täglich über längere Zeit immobilisiert und der Betroffene somit gezwungen, die erkrankte Hand zu benutzen.¹⁹ So kann ein „erlernter Nichtgebrauch“ verhindert werden.²⁷

Ein weiteres bekanntes Rehabilitationskonzept ist das Bobath-Konzept, welches annimmt, dass gesunde Hirnregionen die Aufgaben geschädigter Hirnregionen übernehmen können. Durch das Konzept soll eine entsprechende Vernetzung innerhalb des Gehirns gefördert und die vom Schlaganfall betroffene Körperseite wieder in Bewegungen einbezogen werden.²³ Solche Ansätze erfordern interdisziplinäre Zusammenarbeit. So können beispielsweise mit Physiotherapeuten Gangmuster eingeübt werden, während Ergotherapeuten an der Wiederherstellung der sensomotorischen Fähigkeiten arbeiten und Logopäden mit Sprachtherapie die Aphasie behandeln.¹⁹

Armlähmungen als solches zeigen sich unter anderem in einer stark beeinträchtigten willentlichen Bewegungsfähigkeit, aber auch durch „erhöhte Muskelanspannung (‘Spastik’) mit einer Fehlstellung des Armes in Ruhe“ sowie der Schwierigkeit, den Arm passiv zu bewegen.⁷ Zunächst sollten Armlähmungen medizinisch beurteilt werden, wozu spezielle Verfahren wie der Fugl-Meyer-Test existieren. Auf diese soll hier jedoch nicht genauer eingegangen werden.

Es existieren verschiedene Therapiemethoden mit und ohne Technikeinsatz. Das bilaterale Training beispielsweise besteht darin, dass mit beiden Armen gleichzeitig symmetrische Bewegungen ausgeführt werden, sodass in beiden Armen eine gleichmäßige Bewegungsfähigkeit hergestellt wird. Es wird in verschiedenen Studien neutral bis positiv beurteilt.⁷

Das schädigungsorientierte Training zielt darauf ab, spezifische Behinderungen bei alltäglichen Tätigkeiten zu beheben. Es existieren zwei Formen. Das Arm-Basis-Training beübt alle Bewegungsmöglichkeiten des Arms, also Schulter, Ellenbogen, Handgelenk und Finger. Es ist dabei auf Patienten mit schweren Lähmungen ausgelegt.⁷ Das Arm-Fähigkeits-Training dagegen schult verschiedene Formen von Geschicklichkeit und wird bei leichten Lähmungen angewendet.⁷ Beide Formen dieses Trainings zeigen einen positiven Effekt.²⁷ Das aufgabenorientierte Training stellt eine weitere Trainingsform dar, bei der über Bewegungsaufgaben aus dem Alltag die funktionellen Fähigkeiten des Arms wiederhergestellt werden sollen. Es wird neutral beurteilt.⁷

Doch auch ein Technikeinsatz kann bei einer solchen Therapie erfolgen. So existiert beispielsweise die Armrobot-Methode, bei der ein Roboter nicht selbstständig ausführbare Bewegungen mechanisch unterstützt. In Bezug auf die Effektivität wird diese positiv beurteilt.⁷ Auch die neuromuskuläre Elektrostimulation ist eine mögliche Therapie, bei der ein Gerät per Elektromyographie Bewegungsversuche des Muskels erkennt und diesen daraufhin elektrisch stimuliert, was zu einer großen Bewegung führt.⁷ Die Therapie wurde neutral beurteilt.²⁷

Ausgehend von diesen Informationen entschied ich mich dazu, das Gerät als Unterstützung für Arm-Basis- und Arm-Fähigkeits-Training zu konzipieren. Im Zuge einer Erweiterung könnten auch Ansätze wie Armrobot und Elektrostimulation einbezogen werden.

3 Umgesetzte Konzepte

3.1 Gamification

3.1.1 Grundlegende Mechanismen der Gamification

Es existieren verschiedene Definitionen für Gamification. Beispielhaft sei hier die Definition nach Breuer zitiert, die besagt, dass es sich bei Gamification um die „Verwendung von spieltypischen Mechaniken außerhalb reiner Spiele, mit dem Ziel, das Verhalten von Menschen zu beeinflussen“²⁵ handelt. Entsprechend werden Spielkonzepte verwendet, um die Nutzungsmotivation zu steigern und die Nutzer dazu zu bewegen, mehr oder länger mit einem Produkt zu arbeiten als ohne Gamification. Diese Technik wird innerhalb eines Produktes dazu verwendet, dessen Nutzung zu proklamieren.²⁵

Indem spieltypische Merkmale außerhalb spielerischer Zusammenhänge verwendet werden, nutzt man den menschlichen Spieltrieb aus. Es werden positive Anreize gesetzt, um Menschen zu einem bestimmten Verhalten anzuregen, während der Nutzer ebenfalls vorhandene negative Anreize wie eine Bestrafung vermeiden will.²⁰ Es kommt zu einer „Actio-et-Reactio“-Erfahrung.²⁵ Diese Gestaltung und die entsprechende motivationsssteigernde Wirkung lassen sich mithilfe von psychologischen Theorien erklären, auf welche hier jedoch nicht genauer eingegangen werden soll. Gamification an sich ist an sich ein relativ neues Phänomen und wurde auch erst in letzter Zeit umgesetzt. So wurde der Begriff erst 1978 durch Richard Bartle, einen britischen Informatiker und Computerspiel-Pionier, geschaffen.²⁰

Dieser postulierte auch, dass die Wirkung von Anreizen sich bei verschiedenen Menschen unterscheide. Er entwarf deshalb ein Spielertypen-Koordinatensystem, in dem er Menschen in vier Kategorien verortete, wobei ein Mensch mehreren Kategorien zugleich angehören kann.¹⁷ Im Einzelnen beschrieb er folgende Typen, von denen in einem gelungenen Spiel oder bei einer gelungenen Gamification möglichst viele eine passende Motivation finden sollten:

- die Erfolgstypen/Achiever, welche im Spiel nach konkreten Maßstäben möglichst viel erreichen wollen.¹⁷ 10 Prozent aller Menschen werden hauptsächlich diesem Typ zugeordnet.¹
- die Geselligen/Socializer, denen Kontakte und Interaktionen mit Mitspielern, wie sie auch in klassischen Gesellschaftsspielen vorkommen, besonders wichtig sind.¹⁷ 75 Prozent aller Menschen sind vor allem Socializer.¹
- die Forscher/Explorer, deren Ziel es ist, möglichst viel zu entdecken und erkunden.¹⁷ Dieser Typ ist bei ca. 10 Prozent aller Menschen am meisten ausgeprägt.¹
- die Killer, welche Wettbewerb, Wettkampf und Konflikt lieben. Damit sie motiviert sind, müssen andere verlieren und ihnen Respekt erweisen.¹⁷ Dieser Typ macht 5 Prozent aller Menschen aus.¹

Um Gamification mit Erfolg anwenden zu können, ist es wichtig, die Kriterien und Mechanismen zu kennen, die ein erfolgreiches Spiel hervorbringen. Wie also kann Gamification angewendet werden, um langweilige, frustrierende, monotone oder unbeliebte Tätigkeiten, wie die dieser Arbeit zugrundeliegende Armlähmungs-Therapie, einfacher und motivierender zu gestalten? Zunächst sollten klare Ziele und Regeln aufgestellt werden, anhand derer der Nutzer weiß, welche Rückmeldung er auf eine bestimmte Aktion bekommt. Diese sogenannte Resultatstransparenz führt zu einer gesteigerten Handlungsmotivation.²⁵ Außerdem sollten immer neue Herausforderungen gewährleistet sein.³¹

Eine motivierende Wirkung erzielt man auch, indem man den Spieler immer oder möglichst leicht gewinnen lässt. Der Spieler sollte beim Spielen in einen „Flow“ kommen, der durch eine starke Fokussierung auf das Spiel gekennzeichnet ist.¹ Dafür darf ein Spiel weder zu einfach

noch zu schwierig sein. Eine Belohnung des Spielers in einem festgelegten Intervall führt zu einer schnellen Abnahme der Motivation. Stattdessen sollte man auf die sogenannte operante Konditionierung setzen, indem man in einem variablen Intervall und in variabler Menge belohnt.¹

Ein Spiel kann nach dem MDA-Modell durch drei Bestandteile charaktisiert werden: Mechanics (Spielkomponenten und -funktionen), Dynamics (Interaktion zwischen Spieler und Spiel) und Aesthetics (Emotionen, die beim Spieler erzeugt werden).¹ Von diesen kann der Entwickler nur die Mechanics mithilfe spieltypischer Mechanismen beeinflussen. Zu diesen zählen unter anderem:

- ein Punktesystem gemeinsam mit einer einsehbaren Rangliste (Highscore) bei Mehrspieler-Spielens. Mit diesen können Aktionen des Spielers bewertet werden. Ein Highscore erlaubt einfache Vergleiche durch z.B. eine metrische Punkteskala.²⁵ Highscores können auch suggestiv eingesetzt werden, indem sie den Spieler möglichst in der Mitte der Liste zeigen und somit gleichzeitig belohnen und motivieren.¹
- ein sichtbarer Status durch Titel oder Badges. Diese repräsentieren nach außen, dass der Spieler ein bestimmtes Ziel erreicht hat und bieten somit Vergleichs- und Wettbewerbsmöglichkeiten.²⁵ Motivierend sind sie außerdem, da Menschen gerne sammeln. Sie sollten jedoch nicht zu viel eingesetzt werden.¹
- entdeckbare Aufgaben (Quests), die dem Spiel ein Ziel und Struktur geben und dafür sorgen, dass das Spiel das Interesse des Spielers behält. Wenn Spieler zur Lösung einer Aufgabe mit Mitspielern zusammenarbeiten müssen, kann dies sozial sehr stark motivieren.¹
- eine Fortschrittsanzeige, die eine dynamische Visualisierung des bisherigen Erfolgs und noch zu erledigender Aufgaben erlaubt.²⁵
- gegebenenfalls ein Epic Meaning, also die Arbeit an etwas Erstrebenswerten und an sinnvollen Zielen.²⁵

3.1.2 Beispiele für die motivationssteigernde Wirkung

Gamification kann erstaunliche Effekte erzeugen. Exemplarisch deutlich wurde das bei einer Reihe von Experimenten der schwedischen Werbeagentur DDB unter dem Namen „The Fun Theory“. Bei einem dieser Experimente wurde eine U-Bahn-Treppe so umgebaut, dass sie wie eine Klaviertastatur bei Bedienung mit den Füßen Töne erzeugte. [§] Im Ergebnis wurde diese Treppe sogar öfter als die danebenliegende Rolltreppe benutzt.²⁰

Auch ein Flaschencontainer, der bunt blinkte, wenn Flaschen in die richtige Öffnung geworfen wurden, und der „tiefste Mülleimer der Welt“, der beim Hineinwerfen von Müll einen Pfiff und einen Aufprall ertönen ließ, stellen erfolgreiche Beispiele dar. Im genannten Mülleimer beispielsweise erhöhte sich die Müllmenge drastisch.²⁰ Der Spieldesigner Kevin Richardson schlug für dieses Experiment eine Radarfallenlotterie vor, bei der alle, die die Geschwindigkeitsbegrenzung einhielten, an einem Gewinnspiel um die Strafen der Raser teilnahmen. Als dieser Vorschlag umgesetzt wurde, sank die Geschwindigkeit an der kontrollierten Stelle um 20 %¹

Im wissenschaftlichen Umfeld kann Gamification ebenfalls die Motivation steigern und so bessere Ergebnisse erzielen. So gelang es beispielsweise einigen Gamern, mithilfe eines Tetris-Nachbaus innerhalb von zehn Tagen die Proteinstruktur des AIDS-Virus zu entschlüsseln, was die Wissenschaftler hinter dem Projekt 15 Jahre gekostet hätte.²⁸ Unternehmen versuchen ebenfalls, durch Gamification geeignete Mitarbeiter zu finden. So entwarf der Bayer-Konzern eine Karriere-App

[§]siehe Abb. 3

im Stil von Quizshows und die Management-Simulation „BIMS Online“, die dort nun zur Rekrutierung geeigneter Fachkräfte dienen.

Selbst eine Universität hat Gamification schon ausprobiert: An der Indiana University wurde zeitweise nach einem Experience-Point-System statt nach Noten bewertet.²⁶ Ein weiteres Beispiel für gelungene Gamification findet sich bei Frage-Antwort-Websites wie Quora, Stack Exchange oder Stack Overflow, bei welchen andere Nutzer entsprechende Punkte für gute Antworten auf bestimmte Fragen verleihen können. Außerdem kann ein Fragesteller die beste Antwort auf seine Frage markieren. All dies führt dazu, dass die Qualität der Antworten meist konstant hoch ist.¹

Die bei solchen Experimenten gewonnenen Beobachtungen legen zumindest empirisch nahe, dass Gamification eine motivationssteigernde Wirkung besitzt. Unter Wissenschaftlern ist dieses Thema jedoch strittig. Einige Studien belegen diese These, andere widersprechen ihr. So wurde in einem Experiment der TU München versucht, das Berufsfeld der Kommissionierung, welches noch relativ wenig automatisiert ist, zu gamifizieren. Im Ergebnis arbeiteten die Kommissionierer schneller und zufriedener als ohne Gamification.³¹

Bei einer amerikanischen Studie, die sich auf den Erfolg von Fitnessmaßnahmen mit und ohne Wearable fokussierte, kam man jedoch zu gegenteiligen Ergebnissen. Obwohl Wearables ein klassisches Beispiel für Gamification darstellen, war keine signifikant höhere Gewichtsabnahme zu messen. Die mit Wearable ausgestatteten Probanden verloren im Vergleich zur Kontrollgruppe über den Zeitraum von 24 Monaten meist sogar weniger Gewicht.⁵

3.2 Biofeedback

In unserem Körper kommt es durch Regulationsvorgänge ständig zu Veränderungen von messbaren Zustandsgrößen bei biologischen Vorgängen und Körperfunktionen. Diese sind, der unmittelbaren Sinneswahrnehmung, also dem Bewusstsein, nicht zugänglich. Sie können jedoch mit technischen und elektronischen Hilfsmitteln beobachtbar gemacht werden, was fachsprachlich unter dem Begriff Biofeedback bekannt ist.¹⁸ Oft wird dieses Verfahren zur Rehabilitation von erlahmten Muskeln, wie z.B. bei einer Armlähmung, eingesetzt.

Da solche Körperfunktionen dem Bewusstsein normalerweise nicht zugänglich sind, können sie auch nicht beeinflusst werden. Durch Biofeedback wird genau das jedoch möglich. Durch die Visualisierung der Messwerte für den Patienten kommt es zu einer Rückkoppelung und dieser kann Kontrolle über die Körperfunktion ausüben. Weiterhin können über die operante Konditionierung ganze Reiz-Reaktions-Muster erlernt werden. Insgesamt wird eine Bewusstseinsschärfung für die eigenen inneren Zustände erreicht und die Einflussnahme auf das Nervensystem somit vereinfacht.¹⁸

Die Messwerte können nicht nur visualisiert werden, sondern werden bei manchen Anwendungen auch als Töne dargestellt. Die oft kleinen und tragbaren Messgeräte, die beim Biofeedback benutzt werden, messen meist nichtinvasiv.¹⁸ Ihre Messergebnisse werden dann in einem Analog-Digital-Wandler konvertiert, gemittelt und verstärkt. Anschließend werden sie per kabelloser Bluetooth-Übertragung auf ein zur Darstellung geeignetes Gerät übertragen.

Grundsätzlich können per Biofeedback viele Größen gemessen werden, wie beispielsweise Atem, Blutdruck, Blutwerte, der Hautwiderstand oder Gehirnströme. Im hier beschriebenen Anwendungsfall entschied ich mich jedoch für die Messung von Muskelpotentialen per Elektromyografie, da mit dieser die Beweglichkeit des Arms am besten erfasst werden kann. In vielen Fällen erzielt Biofeedback positive Ergebnisse, sodass es in manchen Fällen sogar eine Alternative zu Medikamenten sein kann.¹⁸

4 Schaltung und Implementierung des Mikrocontroller-Systems

4.1 Aufbau der Schaltung

Das Mikrocontroller-System, welches für das Aufnehmen und Übertragen der Messdaten zuständig ist, besteht im Wesentlichen aus vier Bestandteilen, die miteinander verbunden sind:

1. Dem **Grove EMG-Sensor**²⁹, der per Elektromyografie Messungen über die Stärke der Armmuskel-Kontraktionen erhebt. Dabei wird die elektrische Muskelaktivität anhand von Potentialänderungen auf der Haut mithilfe von drei Oberflächenelektroden gemessen. Diese Signale werden durch den Sensor verstärkt und gefiltert. Zuletzt gibt dieser eine Spannung im Bereich von 1,5 bis 3,3 Volt aus, wobei eine höhere Spannung höhere Muskelaktivität bedeutet. Dabei wird eine Versorgungsspannung von 3,3 bis 5 Volt benötigt.
2. Dem Mikrocontroller **Atmel ATmega 88PA**¹⁴, welcher die vom EMG-Sensor gelieferten Daten an den Bluetooth-Chip weitergibt. Es handelt sich hierbei um einen 8-Bit-Mikrocontroller¹⁵, d.h. es können pro Takt maximal 8 Bit verarbeitet werden. Dieser folgt der sogenannten RISC-Architektur, welche einen reduzierten Befehlssatz im Vergleich zu Standardcomputern besitzt, dafür aber schneller arbeitet. Damit eignet er sich gut für den beabsichtigten Einsatzzweck, bei dem Daten möglichst schnell auf das Mobilgerät übertragen werden sollen.
Weiterhin setzt dieser Controllertyp die Harvard-Struktur um.⁸ Es existieren also getrennte Speicher- und Adressbereiche für Befehle und Daten. Die peripheren Schnittstellen können über Portadressen angesprochen werden. Zu diesen Schnittstellen zählen zwei für die beabsichtigte Anwendung nötige, denn es werden sowohl eine UART-Schnittstelle zur Kommunikation mit dem Bluetooth-Chip als auch ein Analog-Digital-Wandler zum Einlesen der Ausgangsspannung des EMG-Sensors angeboten. Das Programm kann bei diesem Controller über ein Entwicklungsgerät in den Programmspeicher (Flash) geladen werden.
3. Dem Bluetooth-Chip **HC-05**²⁴, welcher eine Möglichkeit zur drahtlosen Kommunikation mit dem Android-Mobilgerät über Funk nach dem Bluetooth-Standard bereitstellt. Dabei sind alle zur Kommunikation nötigen Bestandteile auf einem Chip integriert. Der HC-05 wird über die UART-Schnittstelle (Universal Asynchronous Receiver Transmitter) angesprochen, wobei diese eine digitale serielle Schnittstelle zur Datenübertragung realisiert. Der Bluetooth-Chip kommuniziert standardmäßig mit 9600 Baud.
4. Einem **Quarzoszillatoren** mit eingebautem Schwingquarz, in diesem Fall mit der Frequenz 8 MHz, welcher einen genauen Systemtakt für den Mikrocontroller liefert. Dieser wird benötigt, um eine möglichst störungsfreie UART-Kommunikation realisieren zu können. Der Oszillator muss in den Einstellungen (Fuses) des Mikrocontrollers als Taktquelle ausgewählt (CKSEL-Bit) und die interne Teilung des Taktes durch 8 abgeschaltet (CKDIV8-Bit) werden.

Diese Bestandteile wurden nach dem auf der folgenden Seite sichtbaren Schaltplan (Abb. 1) verbunden.

4.2 Entwicklung des Programms auf dem Mikrocontroller

Das Programm (siehe Quellcode 1, S. 22) wurde in der Programmiersprache C geschrieben und besteht aus sechs Funktionen, die jedoch auf einige in den C-Standardbibliotheken vorhandene Funktionen und Definitionen für den C-Präprozessor zurückgreifen. Zusätzlich tätigt das Programm auch einige eigenen Definitionen für den Präprozessor. Alle Funktionen außer der Main-Funktion, mit der das Programm startet, werden dem Compiler zunächst als Funktionsprototypen mithilfe ihrer Signaturen bekanntgemacht und erst nach der Main-Funktion definiert. Im einzelnen sind folgende Funktionen definiert:

- `init()`. Diese Funktion initialisiert die einzelnen Funktionen des Controllers, indem sie die entsprechenden Einstellungsregister¹⁴ setzt. Der Analog-Digital-Wandler wird aktiviert und dessen interner Taktteiler gesetzt. Die Sende- und die Empfangsfunktion der UART-Schnittstelle werden ebenfalls aktiviert und deren Baudratenfaktor, der angibt, wie schnell kommuniziert werden soll, gesetzt. Der Baudratenfaktor¹⁴ berechnet sich aus (gerundet):

$$UBBR = \frac{f_{CPU}}{16 \cdot r_{BAUD}} - 1$$

Anschließend werden beide Schnittstellen zum ersten Mal ausgelesen, um ihre Funktionsfähigkeit sicherzustellen.

- `u_putchar()`. Hier werden einzelne Zeichen (*Char*) über die UART-Verbindung übertragen. Dabei muss gewartet werden, bis diese Verbindung freigegeben wird.
- `u_puts()`. Diese Funktion überträgt Zeichenketten (*String*) über UART. Hierzu wird über die einzelnen Zeichen mithilfe der in C integrierten Zeigerarithmetik iteriert und jeweils `u_putchar()` aufgerufen.
- `delay()` lässt den Mikrocontroller für die angegebene Zahl an Sekunden warten. Dazu wird die in der Bibliothek `util/delay.h` definierte Funktion `_delay_ms()` benutzt.
- `sendCurrentVoltage()`. In dieser Funktion wird eine Analog-Digital-Wandlung gestartet und nach deren Abschluss das Ergebnis ausgelesen. Dabei handelt es sich um einen relativen Wert im Vergleich zur Referenzspannung,¹⁶ weshalb die eigentliche Spannung aus der Gleichung $U = \frac{x}{1024} \cdot U_{ref}$ gewonnen wird.* Die dabei entstandene Gleitkommazahl wird mithilfe der Bibliotheksfunction `sprintf()` in einen String geschrieben, der per `u_puts()` übertragen werden kann.
- `main()`. In dieser Funktion wird zuerst `init()` aufgerufen. Anschließend wird in einer Endlosschleife im Abstand einer halben Sekunde `sendCurrentVoltage()` gestartet und die Spannung an das Mobilgerät übertragen. Es muss sich hier um eine Endlosschleife handeln, da der Mikrocontroller nach der Ausführung von `main()` in ein undefiniertes Verhalten übergeht, aus dem er nur durch Neustart befreit werden kann.

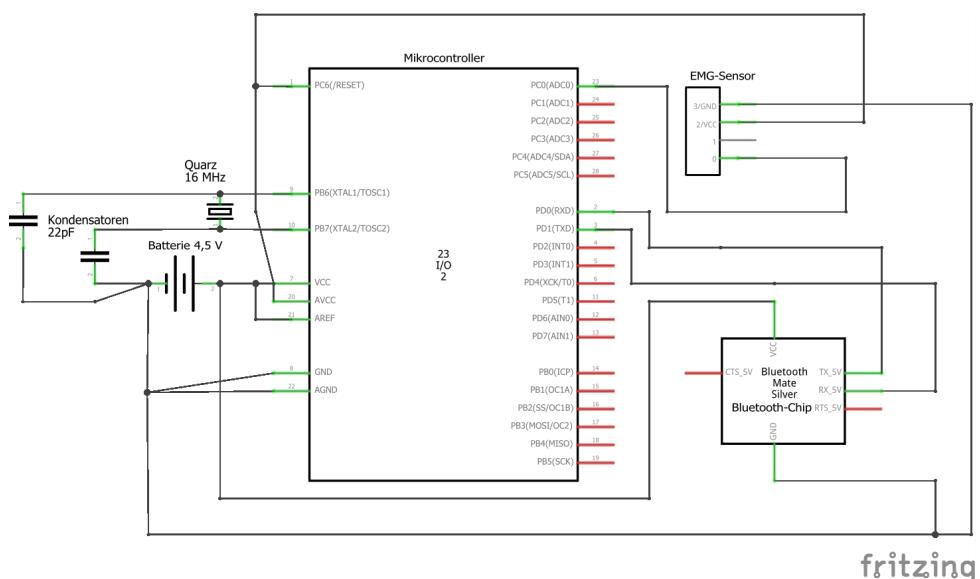


Abbildung 1: Der Schaltplan des Mikrocontrollers

*Hierbei ergibt sich jedoch insoweit ein Problem, als dass die Versorgungsspannung bei Batteriebetrieb schwanken kann.

5 Entwicklung der Begleitapp für Android

5.1 Grundlegender Aufbau und Konzept der App

Um die vom EMG-Sensor gelieferten Daten zu verarbeiten und dem Benutzer bzw. Patienten anschaulich darzustellen, erschien es sinnvoll, eine App für Mobilgeräte zu programmieren. Hierbei entschied ich mich, das weitverbreitete Betriebssystem Android zu verwenden und die App zu diesem (ab Android 6.0) kompatibel zu machen, wobei zur Umsetzung die Programmiersprache Kotlin⁴ verwendet wurde. Diese App sollte folgende grundlegende Funktionen enthalten:

- **Durchführung von Bewegungsübungen:** Es sollten einfache Bewegungsübungen möglich sein, bei denen sowohl der aktuelle Messwert als auch die Messwerte im zeitlichen Verlauf angezeigt werden sollten. Dabei sollte auch ein Vergleich mit früheren Übungen möglich sein.
- **Ein durch Bewegungsübungen steuerbares Minispiel:** Die App sollte ein durch den Benutzer mithilfe von Bewegungen steuerbares Minispiel enthalten. dieses sollte möglichst einfach verständlich sein.
- **Ein verbindendes Gamification-System:** Sowohl bei normalen Übungen als auch beim Minispiel sollten Gamification-Elemente eingebracht werden. So sollte das Sammeln von Erfahrungspunkten (Experience Points, XP) möglich sein und es sollten für besondere Leistungen sogenannte Badges (Abzeichen) vergeben werden können. Dabei sollten die zu erbringenden Leistungen für die Badges in Form von Quests bzw. Aufgaben vorher für den Benutzer sichtbar sein.
- **Erinnerungen an die Übungen:** Die App sollte den Benutzer zu von ihm festgelegten Zeiten durch z.B. eine Benachrichtigung an die Durchführung seiner Übungen erinnern.

Ausgehend davon bot sich eine Gliederung in insgesamt vier für den Anwender sichtbare und miteinander verknüpfte Bildschirmseiten (bei Android Activities⁶ genannt) an. Dies sind:

- Eine **Startseite**, die die bisher erreichten Gamification-Erfolge zusammenfasst und kurze Informationstexte sowie Verknüpfungen zu den Übungsmöglichkeiten anbietet. Damit soll der Einstieg möglichst einfach gestaltet werden. Um die einzelnen Bereiche klar voneinander zu unterscheiden, kommt hier das einer Karteikarte ähnelnde Oberflächenelement CardView zum Einsatz.
- Eine **Übungsseite**, auf der man Übungen durchführen kann. Dabei zeigt ein tachoähnliches Oberflächenelement den aktuell festgestellten Messwert an, während mithilfe eines Diagramms die Messwerte während der gesamten Übung dargestellt werden. Diese Übung kann durch den Benutzer beliebig gestartet und beendet werden, während diese Seite auch eine Funktion zur Ansicht der aktuell verfügbaren Quests bereitstellt.
- Eine Seite für das **Minispiel**. Da dieses in einem späteren Abschnitt noch genauer beschrieben wird, soll hier nicht darauf eingegangen werden.
- Eine **Einstellungsseite**. Hier können Einstellungen getroffen werden, die für die restlichen Teile der App von Bedeutung sind. So kann man hier den eigenen Namen einstellen, die Gamification-Datenbank importieren und exportieren und die Übungserinnerungen konfigurieren. Für solche Seiten stellt das Android-SDK die Klasse PreferenceFragment⁶ bereit.

Activities werden von Android intern auf dem sogenannten *Back Stack* verwaltet, der basierend auf den abstrakten Datentyp Stack ein Zurückkehren zur vorherigen Activity erlaubt. Allgemein ist beim Arbeiten mit Activities das Lebenszyklus-Modell zu beachten, das beschreibt, auf welche Weise Activities gestartet, pausiert und gestoppt werden können.⁶ Die einzelnen Bestandteile der App sollen im folgenden näher erläutert werden. Screenshots der jeweiligen Activities befinden sich im Anhang.

5.2 Kommunikation mit dem Mikrocontroller

Wie bereits erläutert, sendet der Mikrocontroller die Messdaten über eine Bluetooth-Verbindung. Um nun mit diesem kommunizieren zu können, muss die Android-App eine solche Verbindung implementieren. Glücklicherweise enthält das Android-SDK (Software Development Kit) bereits eine Softwarebibliothek, die genau dies vereinfacht.⁹⁶

Um eine Verbindung herzustellen, müssen zunächst einige Schritte durchlaufen werden, die nur in den Activities durchgeführt werden können. In dieser App benötigen zwei Activities Bluetooth-Zugriff: die Übungsseite und die Minispiel-Seite. Beide müssen unabhängig voneinander den Code zum Auffinden des zu verbindenden Bluetooth-Geräts implementieren. Dabei müssen beispielsweise die nötigen Berechtigungen überprüft, der Bluetooth-Adapter eingeschaltet und, sofern noch nicht geschehen, das Bluetooth-Gerät gekoppelt werden.⁹⁶

Ist dies geschehen, kann mit dem Einlesen der per Bluetooth über das Protokoll RFCOMM (Radio Frequency Communication)⁶ eintreffenden Werte begonnen werden. Diese Funktionalität ist in der App in der Klasse `BluetoothNoService` gekapselt. Um kurzzeitige Schwankungen der Messwerte auszugleichen, bietet es sich an, diese in einer *Queue* zwischenzulagern. Eine Queue oder Warteschlange ist eine in der Informatik häufig eingesetzte Datenstruktur, die nach dem First In - First Out - Prinzip (FIFO) arbeitet, d.h. das Objekt, welches als erstes der Warteschlange hinzugefügt wurde, verlässt sie auch als erstes wieder. Für diesen Zweck eignet sie sich sehr gut, da im Laufe der Zeit immer wieder neue Messwerte hinzukommen, während ältere entfernt werden müssen. Damit keine zu alten Werte verwendet werden, ist es sinnvoll, die Länge der Queue auf 10 Werte zu beschränken.

Zum Hinzufügen der Werte ist es sinnvoll, einen Thread zu implementieren, d.h. eine Funktion, die parallel zum übrigen Programm abläuft. Dieser Thread kann der Queue dann beständig neue Werte hinzufügen, sobald diese eintreffen.

Um nun zu jedem Zeitpunkt einen Durchschnittswert aus der Queue berechnen zu können, bietet sich der Median der Werte an. Da die Werte jedoch als Spannung abgelegt sind und der EMG-Sensor Werte zwischen $U_{min} = 1.5V$ und $U_{max} = 3.3V$ ausgibt, ist es sinnvoll, dem Nutzer die Werte in Prozent des Maximalwertes zu präsentieren. Ein solcher Wert lässt sich mit folgender Gleichung berechnen (wobei x der Median der Messwerte ist):

$$p = 100 * \frac{x - U_{min}}{U_{max} - U_{min}}$$

Dieser Wert kann nun an die Benutzeroberfläche zur weiteren Darstellung übergeben werden.

5.3 Konzept und programmiertechnische Umsetzung des Minispiele

Das Minispiel soll es erlauben, ein virtuelles Flugzeug über eine Gebirgslandschaft mit Bergen unterschiedlicher Höhe zu steuern. Dabei bestimmt die gemessene Muskelaktivität die Höhe des Flugzeugs. Je höher die Aktivität, desto höher fliegt das Flugzeug. Das Ziel ist es dabei, das Flugzeug möglichst lange fliegen zu lassen, ohne gegen einen Berg zu stoßen. Dieses Spielprinzip ähnelt teilweise dem erfolgreichen Smartphone-Spiel „Flappy Bird“. Es wäre wünschenswert, wenn es gelänge, an den Erfolg des genannten Spielprinzips anzuknüpfen.

Zur Umsetzung des Minispiele wurde die Möglichkeit des Android-SDKs genutzt, eigene Oberflächenelemente (sogenannte *Views*¹¹) zu erstellen. Eine solche View beinhaltet in diesem Fall das Spiel und übernimmt dessen Darstellung. Das Android-System fordert dabei, dass man bestimmte, für die eigene View spezifische Methoden überschreibt. Dies sind im einzelnen:

- `onDraw()`: Hier wird der sichtbare Teil der View gezeichnet.¹¹⁶ Was gezeichnet wird, hängt von der boolesche Objektvariable `inGame` der View ab, welche den Spielzustand

(innerhalb oder außerhalb des Spiels) speichert. Befindet man sich außerhalb des Spiels, wird ein Text angezeigt, der darauf hinweist, dass durch einen Klick das Spiel gestartet werden kann. Innerhalb des Spiels müssen mehr Elemente gezeichnet werden. Zunächst jedoch wird der aktuelle Messwert des `BluetoothNoService` übernommen.

Nun werden anhand dieses Messwerts die Koordinaten der oberen linken Ecke des Flugzeugs bestimmt. Damit kann festgestellt werden, ob das Flugzeug gegen den Berg direkt vor ihm gestoßen ist. In diesem Fall ist das Spiel verloren. Ein entsprechender Infotext wird angezeigt und das Spiel durch die `handleTap()`-Methode beendet.

Ansonsten können der blaue Hintergrund, die grünen Berge sowie das als Icon vorliegende Flugzeug gezeichnet werden.

- `onSizeChanged()` und `onMeasure()`: Diese Methoden werden aufgerufen, wenn das System beispielsweise bei einer Drehung des Bildschirms eine Größenänderung der View anfordert.¹¹ Entsprechend wird dann eine neue Höhe und Breite festgelegt. Daraufhin können alle von diesen abhängigen Teile der View neu generiert werden. Hier sind dies der Hintergrund und die prozedural zufällig (bezüglich ihrer Höhe) generierten Berge.
- `onTouchEvent()`: Diese Methode verarbeitet Touch-Eingaben bzw. Klicks.¹² Android bietet dazu eine sogenannten `GestureDetector` an, der einfache und komplizierte Gesten erkennen kann. In diesem Fall muss jedoch nur das Tippen auf den Bildschirm erkannt werden und entsprechend das Spiel gestartet oder gestoppt werden. Dies übernimmt die Methode `handleTap()`.

`handleTap()` führt nun einige Schritte aus, die für das Starten bzw. Stoppen des Spiels nötig sind. Die bereits erwähnte `inGame`-Variable wird nun negiert, so dass der Spielzustand wechselt. Befindet man sich danach innerhalb des Spiels, so müssen nur die Höhen der Berge neu generiert werden. Dazu benutzt man einen einfachen Zufallsgenerator.

Befindet man sich jedoch anschließend außerhalb des Spiels, so sind mehr Schritte durchzuführen. Die Liste der Berge sowie die Liste der Bluetooth-Messwerte müssen zurückgesetzt werden. Davor jedoch sollten entsprechende Gamification-Bewertungsfunktionen aufgerufen werden, die feststellen, wie viele XP beziehungsweise welche Badges der Nutzer für diese Leistung erhält. Die dazu nötigen Funktionen werden im nächsten Abschnitt genauer beschrieben. Nach den jeweiligen Änderungen wird die View natürlich neu gezeichnet.

Das Android-Grafik-Framework unterscheidet beim Zeichnen im Übrigen dazwischen, was gezeichnet wird (bestimmt von der Klasse `Canvas`) und wie es gezeichnet wird (geregelt durch die Klasse `Paint`).¹¹ Ein solches `Paint`-Objekt bestimmt dabei unter anderem Farbe, Stil und Schrift, die auf ein Objekt angewendet werden. Durch diese Teilung ist es möglich, `Paint`-Objekte schon vor der Benutzung zu erstellen und anschließend wiederzuverwenden. Da Views oft neu gezeichnet werden, kann dadurch die Performance verbessert und die Benutzeroberfläche flüssiger werden.¹¹

Das fertige Spiel lässt sich nun folgendermaßen steuern: Zu Beginn erscheint der Text „Um das Spiel zu beginnen, berühre den Bildschirm.“ Tippt man auf diesen erscheint die Oberfläche des Spiels. Das Flugzeug kann auf dieser nun ausschließlich durch Muskelaktivität und nicht durch Touch-Gesten oder ähnliches gesteuert werden. Stößt man nun gegen einen Berg oder tippt man erneut auf den Bildschirm, wird das Spiel beendet, wobei im ersten Fall ein zusätzlicher Hinweis („Du bist gegen einen Berg gestoßen. Du verlierst!“) erscheint. Anschließend wird die für das Spiel zu vergebende XP-Punktzahl berechnet und angezeigt sowie auf eventuell abgeschlossene Quests überprüft.

5.4 Umsetzung der Gamification in der App

Im Bereich der Gamification entschied ich mich, ein (Erfahrungs-)Punktesystem und Quests bzw. Badges einzuführen. Für jede durchgeführte Übung bzw. für jedes Spielen des Minispiels werden dabei abhängig von den aufgenommenen Messwerten Punkte verteilt. Dabei gelten für die Verteilung folgende Kriterien:

- Bei einer **Übung** wird eine Liste L der prozentualen Messwerte erstellt. Aus dieser kann dann die Punktzahl P berechnet werden:

$$P = \frac{\min(L) + \text{avg}(L) + \max(L)}{3}$$

Dabei berechnet $\min(L)$ den Minimalwert der Liste, $\text{avg}(L)$ den Durchschnittswert der Liste und $\max(L)$ den Maximalwert der Liste.

- Beim **Minispiel** gelten die Regeln für eine Übung grundsätzlich weiter. Jedoch werden als Bonus zusätzlich $2 \cdot d$ XP vergeben, wobei d die Anzahl der aufgenommenen Messwerte bis zum Absturz des Flugzeugs darstellt.

Natürlich wird der Nutzer auch jeweils über die vergebene Punktzahl mithilfe einer im unteren Bildschirmteil eingeblendeten **Toast**-Benachrichtigung informiert.

Bei den bereits erwähnten Quests entschied ich mich, verschiedene Kriterien bzw. Anforderungen für die Fertigstellung der Quest vorauszusetzen. Um diese bei den einzelnen Quests flexibel regulieren zu können, erschien es sinnvoll, solche Daten in einer lokalen Datenbank zu speichern, auf welche die App dann zugreifen kann. Android bietet dafür die sehr kompakte Datenbankbibliothek *SQLite* an, welche ein relationales Datenbanksystem implementiert und über die standardisierte Datenbanksprache *SQL* angesprochen werden kann.⁶ Besonders wichtig ist hierbei, dass keine Server-Software benötigt wird, sondern die Datenbank einfach in einer einzelnen Textdatei gespeichert werden kann, was im Kontext einer Android-App vieles vereinfacht.⁶

Die Datenbanktabelle besteht nun aus den folgenden Spalten, welche eine einzelne Quest kennzeichnen:

Spaltenname	Spaltentyp	Beschreibung
<code>_id</code>	INT	Die eindeutige Identifikationsnummer der Quest, über die sie aufgerufen und verändert werden kann.
<code>title</code>	VARCHAR(50)	Der Titel der Quest. Dieser kann bis zu 50 Zeichen lang sein
<code>description</code>	VARCHAR(150)	Eine Beschreibung der Quest, die bis zu 150 Zeichen lang sein kann. Sie sollte möglichst die in den restlichen Spalten definierten Ziele zum Erfüllen der Quest zusammenfassen.
<code>icon</code>	INT	Ein Zahl zwischen 1 und 5, die für die Anzeige der Quest eines von 5 vordefinierten Symbolen auswählt.
<code>requiredXP</code>	INT	Gibt den XP-Stand an, der zur Anzeige der Quest nötig ist.
<code>finishedXP</code>	INT	Gibt den XP-Stand an, der zum Beenden der Quest notwendig ist.

Spaltenname	Spaltentyp	Beschreibung
earnedXP	INT	Gibt die Anzahl an XP an, die der Nutzer für die Beendigung der Quest erhält.
overPercentage	INT	Gibt einen Messwert an, über dem man für eine bestimmte Zeit sein muss, um die Quest zu beenden.
timeOverPercentage	INT	Gibt die Zeitspanne an, während der man über dem genannten Messwert sein muss.
minimumPercentage	INT	Gibt eine Prozentzahl bzw. einen Messwert an, die mindestens einmal während einer Übung erreicht werden muss, um die Quest zu beenden.
isCompleted	BOOL	Gibt an, ob die Quest bereits beendet und damit in ein Badge umgewandelt wurde.

In allen INT-Spalten außer `_id` und `icon` kann auch eine 0 stehen. Dies bedeutet dann, dass die entsprechende Anforderung bei dieser Quest nicht benötigt wird.

Nun ist es relativ einfach, die Quests und Badges in der App anzuwenden. Nach jeder Übung und jedem Spiel kann nun anhand der in der Tabelle festgelegten Kriterien überprüft werden, ob eine der Quests beendet wurde. Dazu müssen drei Bedingungen erfüllt sein: die `minimumPercentage`-Bedingung (der Wert wurde mindestens einmal erreicht), die `overPercentage`-Bedingung (der Wert wurde für die angegebene Zeitspanne überschritten) und ein entsprechender XP-Stand.

Ist dies der Fall, kann die Quest durch eine Datenbankabfrage auf „erledigt“ gesetzt werden. Die durch die Quest verdienten XP-Punkte können dem Nutzer nun gutgeschrieben werden. Gleichzeitig wird mithilfe eines `DialogFragments` dem Benutzer ein Dialog angezeigt, der ihn über die abgeschlossenen Quests und die dafür erhaltenen Erfahrungspunkte informiert.

Natürlich ist es dem Nutzer auch möglich, sich die erhaltenen Badges bzw. die zur Verfügung stehenden Quests anzuzeigen. Im ersten Fall wird diese Funktion durch einen Button auf der Startseite aufgerufen, im letzteren Fall durch einen Button auf der Übungsseite. In beiden Fällen wird ein `Dialog` mit einer `ListView` angezeigt, die Elemente auflisten kann. Hier sind dies die Quests bzw. Badges, welche über einen `CursorAdapter` einfach aus der Datenbank abgerufen werden können.

In Hinsicht auf die Umsetzung der Gamification ist jedoch auch zu betrachten, inwieweit die App der Gamification-Theorie folgt, um daraus möglicherweise Rückschlüsse auf die Wirksamkeit zu ziehen. Bezüglich der Spielertypen ist die App beispielsweise eher für Forscher und Erfolgstyphen ausgelegt, da sich die Gamification-Elemente aktuell darauf fokussieren, den Nutzer Erfolge erreichen zu lassen. Gesellige und Killer dagegen haben wenig Möglichkeiten, mit anderen zu interagieren und mit ihnen in Wettbewerb zu treten, da eine Mehrspielerfunktion fehlt, welche den zeitlichen Rahmen der Arbeit gesprengt hätte. Die Einbindung einer solchen Funktion wäre jedoch eine Erweiterungsmöglichkeit.

Jedoch ist die für Gamification wichtige Handlungstransparenz gegeben, da der Nutzer sich seinen XP-Stand und die verfügbaren Quests einfach ansehen kann. Neue Herausforderungen sind trotzdem nur begrenzt gegeben, da das Hinzufügen von Quests in der aktuellen Version noch schwierig ist. Dies ließe sich durch eine Updatefunktion über das Internet beheben. Das Minispiel ist weder zu einfach noch zu schwer, da es sich am bekannten Prinzip des Spiels *Flappy Bird* orientiert. Das ermöglicht es dem Nutzer, in einen Flow zu kommen.

Auch das Prinzip der operanten Konditionierung wurde umgesetzt, denn die Vergabe von Erfahrungspunkten und Badges erfolgt abhängig von den Messwerten immer unterschiedlich. Die Belohnung in einem variablen Intervall ist dabei jedoch nicht möglich, da die Vergabe nur jeweils nach einer Übung oder einem Spiel erfolgen kann.

Im Bereich der spieltypischen Mechanismen wurden einige umgesetzt. So enthält die App, wie bereits beschrieben, ein Erfahrungspunkte-System. Ein Highscore jedoch ist aufgrund der fehlenden Mehrspieler-Möglichkeiten nicht vorhanden. Hingegen ist ein System aus Quests und Badges vorhanden, die so verknüpft sind, dass aus einer abgeschlossenen Quest ein Badge erzeugt wird. Damit existiert ein sichtbarer Status, der der App ein Ziel und eine Struktur gibt und den Fortschritt des Nutzers symbolisiert. Ein sinnvolles Ziel bzw. Epic Meaning ist hier in jedem Fall vorhanden, denn der Benutzer möchte seine Armlähmung bekämpfen.

5.5 Funktionsweise des Benachrichtigungssystems

Das Benachrichtigungssystem besteht aus einer Reihe von Funktionen, die vorrangig durch die Einstellungsseite gesteuert werden. Die erste dieser Funktionen erledigt eine seit Android 8 für das Senden von Benachrichtigungen nötige Maßnahme, indem sie einen Benachrichtigungskanal einrichtet. Mithilfe dessen ist es dem Benutzer möglich, die Benachrichtigungen über die Systemeinstellungen zu unterdrücken.¹⁰ Das Erstellen des Kanals erfolgt bei jedem Aufruf der Startseite, aber nach dem ersten Mal bleibt dies wirkungslos, sodass nur genau ein Kanal erstellt wird.

Eine weitere Funktion ist für das Setzen einer Erinnerung zuständig. Sie liest aus den Einstellungen (sogenannte SharedPreferences) aus, ob und wann erinnert werden soll. Nachdem alle bisherigen Erinnerungen gelöscht wurden, kann eine neue gesetzt werden. Dazu verwendet die App den System-Service AlarmManager.²² Sobald die gewünschte Zeit erreicht wurde, ruft dieser einen BroadcastReceiver auf. Solche BroadcastReceiver sind ein fester Bestandteil des Android-SDK und können auf vielfältige Meldungen durch das Android-System, welche auf Systemereignisse hinweisen, reagieren.⁶ In diesem Fall wird hier eine Methode zum Senden der Benachrichtigung aufgerufen.

Wird eine solche Benachrichtigung gesendet, ist sie systemweit für den Nutzer sichtbar und erinnert ihn daran, seine Übungen durchzuführen. Bei Klick auf die Benachrichtigung öffnet sich die Startseite der App.

Schließlich lassen sich, wie bereits erwähnt, sämtliche im AlarmManager gespeicherten Erinnerungen auch wieder löschen, wovon entsprechend der Einstellungen Gebrauch gemacht wird.[‡]

[‡]Die Implementierung des Benachrichtigungssystems basiert teilweise auf einer in²² vorgestellten Lösung.

6 Zusammenfassung

Ausblick: "Die Rehabilitation [ist] ein großes Thema, also das Bewegungstraining bei Schlaganfallpatienten [...]." (Sami Haddadin in Interview zu Robotern) (Src:CTHaddadin)

7 Literatur- und Quellenverzeichnis

Literaturquellen

- [1] Cunningham, Christopher; Zichermann, Gabe: *Gamification by Design - Implementing Game Mechanics in Web and Mobile Apps*, 1. Auflage, Sebastopol, O'Reilly Verlag, 2011, [https://doc.lagout.org/programmation/GameDesign/GamificationbyDesign-Zichermann,Cunningham-O'Reilly\(2011\)/GamificationbyDesign-Zichermann,Cunningham-O'Reilly\(2011\).pdf](https://doc.lagout.org/programmation/GameDesign/GamificationbyDesign-Zichermann,Cunningham-O'Reilly(2011)/GamificationbyDesign-Zichermann,Cunningham-O'Reilly(2011).pdf)
[Zugriff am 30.1.2018, 18:20 Uhr]
- [2] Gargenta, Marko: *Einführung in die Android-Entwicklung*, 1. Auflage, Köln, O'Reilly Verlag, 2011
- [3] Grävemeyer, Arne: *Ära starker Bots mit zarten Fingern*, Roboterforscher Haddadin erwartet Wandel in Industrie und Haushalt, in: c't 11/2018, S. 68 - 69
- [4] Isakova, Svetlana; Jemerov, Dmitry: *Kotlin in Action*, 1. Auflage, Shelter Island, Manning Publications, 2017
- [5] Jakicic, John M. et al.: *Effect of Wearable Technology Combined With a Lifestyle Intervention on Long-term Weight Loss*, The IDEA Randomized Clinical Trial, In: Journal of the American Medical Association, 316(11)/2016, S. 1161 - 1171, <https://jamanetwork.com/journals/jama/articlepdf/2553448/joi160104.pdf>
[Zugriff am 27.12.2017, 15:37 Uhr]
- [6] Künneth, Thomas: *Android 8 - Das Praxisbuch für Java-Entwickler*, 5. aktualisierte Auflage, Bonn, Rheinwerk Verlag, 2018
- [7] Platz, Thomas; Roschka, Sybille: *Rehabilitative Therapie bei Armlähmungen nach einem Schlaganfall*, Patientenversion der Leitlinie der Deutschen Gesellschaft für Neurorehabilitation, Bad Honnef, Hippocampus Verlag, 2011, http://www.kompetenznetz-schlaganfall.de/fileadmin/download/Arm-Reha/Leitlinie_Therapie_Armlaehmung_220911-verlinkt.pdf
[Zugriff am 27.12.2017, 16:04 Uhr]
- [8] Schmitt, Günter: *Mikrocomputertechnik mit Controllern der Atmel AVR-RISC-Familie*, Programmierung in Assembler und C - Schaltungen und Anwendungen, 4. Auflage, München, Oldenbourg Verlag, 2008

Internetquellen

- [9] Android Developer Team: *Bluetooth overview*, <https://developer.android.com/guide/topics/connectivity/bluetooth>
[Zugriff am 11.11.2018, 15:36 Uhr]
- [10] Android Developer Team: *Create a Notification*, <https://developer.android.com/training/notify-user/build-notification>
[Zugriff am 11.11.2018, 15:38 Uhr]
- [11] Android Developer Team: *Custom Drawing*, <https://developer.android.com/training/custom-views/custom-drawing>
[Zugriff am 11.11.2018, 15:40 Uhr]

- [12] Android Developer Team: *Making the View Interactive* , <https://developer.android.com/training/custom-views/making-interactive>
 [Zugriff am 11.11.2018, 15:43 Uhr]
- [13] Antwerpes, Frank et al.: *Schlaganfall*, <http://flexikon.doccheck.com/de/Schlaganfall>
 [Zugriff am 31.12.2017, 12:35 Uhr]
- [14] Atmel Corporation: *ATmega48A/PA/88A/PA/168A/PA/328/P*, Atmel 8-Bit Microcontroller with 4/8/16/32 KBytes In-System Programmable Flash, Datasheet, http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf
 [Zugriff am 27.12.2017, 11:46 Uhr]
- [15] Atmel Corporation: *ATmega48PA/88PA/168PA*, 8-bit AVR Microcontrollers, Datasheet Complete, http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42734-8-bit-AVR-Microcontroller-ATmega48PA-88PA-168PA_Datasheet.pdf
 [Zugriff am 27.12.2017, 11:48 Uhr]
- [16] Autorengemeinschaft: *AVR-GCC-Tutorial*, <https://www.mikrocontroller.net/articles/AVR-GCC-Tutorial>
 [Zugriff am 27.12.2017, 12:10 Uhr]
- [17] Autorengemeinschaft: *Bartle-Test*, <https://de.wikipedia.org/wiki/Bartle-Test>
 [Zugriff am 4.4.2018, 14:11 Uhr]
- [18] Autorengemeinschaft: *Biofeedback*, <https://de.wikipedia.org/wiki/Biofeedback>
 [Zugriff am 5.4.2018, 16:25 Uhr]
- [19] Autorengemeinschaft: *Schlaganfall*, <https://de.wikipedia.org/wiki/Schlaganfall>
 [Zugriff am 31.12.2017, 13:01 Uhr]
- [20] Drescher, Frank: *Spiele und Spielzeug - Gamification*, https://www.planet-wissen.de/gesellschaft/spiele_und_spielzeug/gamification/index.html
 [Zugriff am 30.3.2018, 10:12 Uhr]
- [21] Feichter, Martina: *Schlaganfall*, <https://www.netdoktor.de/krankheiten/schlaganfall/>
 [Zugriff am 31.12.2017, 13:25 Uhr]
- [22] Fernando, Jaison: *How to schedule notifications using AlarmManager?*, <https://droidmentor.com/schedule-notifications-using-alarmmanager/>
 [Zugriff am 11.11.2018, 15:49 Uhr]
- [23] Freyer, Timo; Mörkl, Sabrina; Ostendorf, Norbert: *Bobath-Konzept*, <http://flexikon.doccheck.com/de/Bobath-Konzept>
 [Zugriff am 3.1.2018, 17:42 Uhr]

- [24] ITead Studio: *HC-05, Bluetooth to Serial Port Module*, <http://www.electronicaestudio.com/docs/istd016A.pdf>
 [Zugriff am 27.12.2017, 11:13 Uhr]
- [25] Koch, Michael; Ott, Florian: *Gamification – Steigerung der Nutzungsmotivation durch Spielkonzepte*, Projekt mit der Forschungsgruppe Kooperationssysteme an der Universität der Bundeswehr München, <http://www.sozio.tech.org/gamification-steigerung-der-nutzungsmotivation-durch-spielkonzepte/>
 [Zugriff am 30.1.2018, 18:03 Uhr]
- [26] Parrish, Kevin: Professor Uses RPG-like Exp Rather Than Grades, <https://www.tomsguide.com/us/Experience-Points-XP-Indiana-University-news-6183.html>
 [Zugriff am 5.4.2018, 15:32 Uhr]
- [27] Nelles, Gereon et al.: *Motorische Rehabilitation nach Schlaganfall*, <http://www.friedehorst.de/nrz/rehabilitation.pdf?m=1140520827>
 [Zugriff am 27.12.2017, 16:16 Uhr]
- [28] Reinhardt, Anja: *Motivation und Manipulation im Alltag*, Spieltheorie „Gamification“, <http://www.deutschlandfunk.de/spieltheorie-gamification-motivation-und-manipulation-im.724.de.html>
 [Zugriff am 5.4.2018, 15:59]
- [29] Seeed Technology Co. Ltd.: *Grove - EMG Detector*, http://wiki.seeed.cc/Grove-EMG_Detector/
 [Zugriff am 30.12.2017, 14:03 Uhr]
- [30] Statistisches Bundesamt: *Ergebnisse der Todesursachenstatistik für Deutschland 2015*, ausführliche 4-stellige ICD10-Klassifikation, https://www.destatis.de/DE/Publikationen/Thematisch/Gesundheit/Todesursachen/Todesursachenstatistik5232101157015.xlsx?__blob=publicationFile
 [Zugriff am 2.1.2018, 11:16 Uhr]
- [31] W wie Wissen: *Gamification - Wie Spielen den Alltag interessanter macht*, Ausschnitt aus der gleichnamigen Sendung in Das Erste am 19.12.2015 (16.00 Uhr), <http://www.ardmediathek.de/tv/W-wie-Wissen/Gamification-Wie-Spielen-den-Alltag-in/Das-Erste/Video?bcastId=427262&documentId=32368232>
 [Zugriff am 4.4.2018, 14:32 Uhr]

Bildquellen

- [Abb. 2] [http://www.volkskrankheiten.at/images/1237/widgets/Schlaganfall-\(2\).svg](http://www.volkskrankheiten.at/images/1237/widgets/Schlaganfall-(2).svg)
 [Zugriff am 31.12.2017, 13:37 Uhr]
- [Abb. 3] https://scontent-frx5-1.xx.fbcdn.net/v/t1.0-9/225807_10150179024982659_311531_n.jpg?_nc_cat=0&oh=986b13c0619d22e40dc9de883a6ed930&oe=5B5CEE4C
 [Zugriff am 17.5.2018, 15:56 Uhr]

8 Anhang

2.1 Schlaganfall als Krankheitsbild

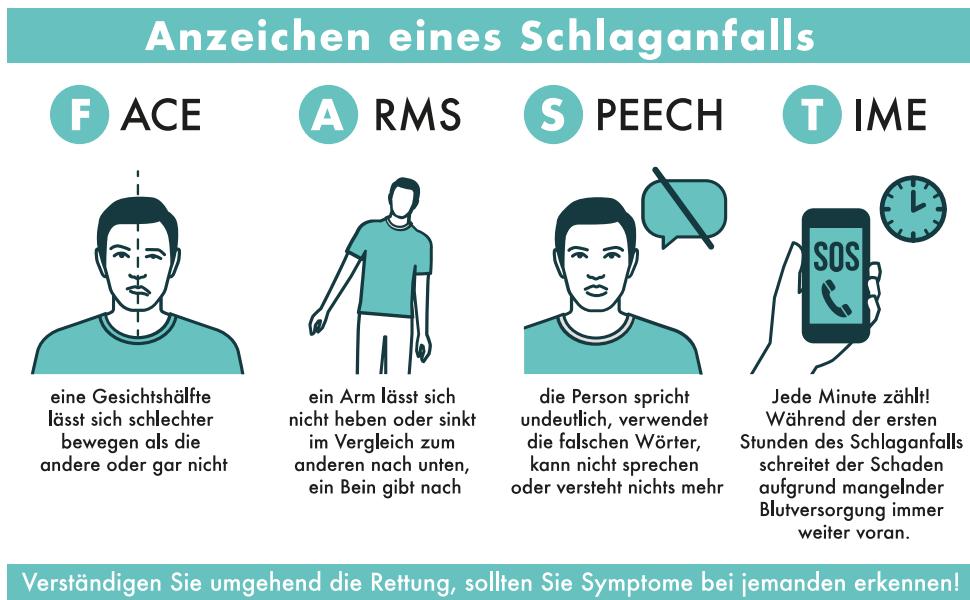


Abbildung 2: Schaubild zum FAST-Test

3.1 Gamification



Abbildung 3: Die Klavier-Treppe aus dem Projekt *The Fun Theory*

4.1 Aufbau der Schaltung

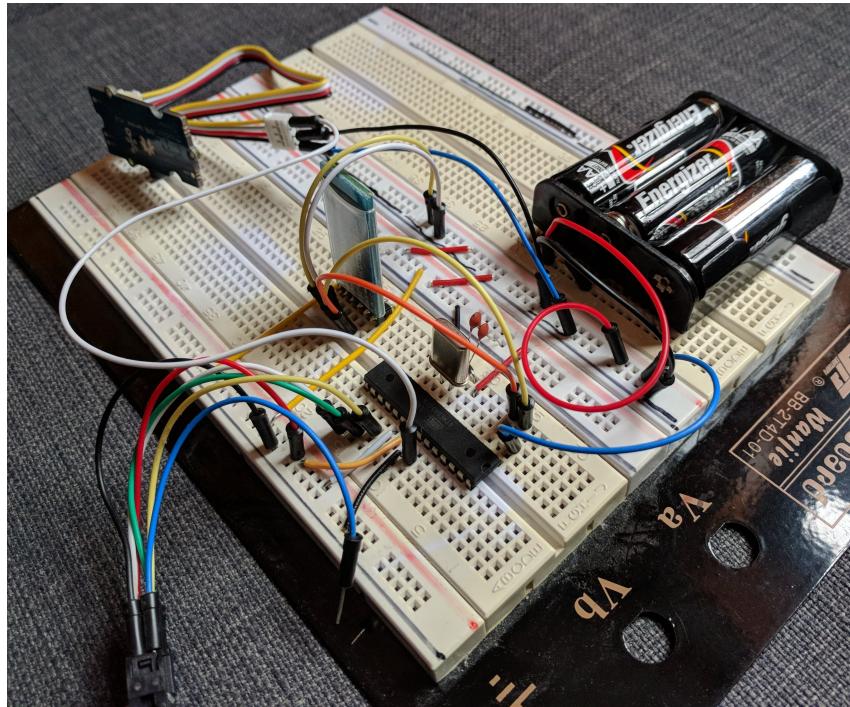


Abbildung 4: Foto der aufgebauten Schaltung auf einem Breadboard

4.2 Entwicklung des Programms auf dem Mikrocontroller

```
1 #include <avr/io.h>
2 #include <stdio.h>
3 #include <string.h>
4 #include <stdint.h>
5
6 #define F_CPU 8000000ul
7 #include <util/delay.h>
8 #define BAUD 9600ul
9 #define UBBRVAL 51
10
11 void init(void);
12 void u_putchar(uint8_t x);
13 void u_puts(char *s);
14 void sendCurrentVoltage(void);
15 void delay(uint16_t millisec);
16
17 int main(void) {
18     init();
19     while(1) {
20         sendCurrentVoltage();
21         delay(500);
22     }
23     return 0;
24 }
25
26 void init(void) {
27     ADMUX = 0x00;
28     ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1);
29     ADCSRA |= (1<<ADSC);
30     while (ADCSRA & (1<<ADSC));
```

```

31     (void) ADC;
32
33     UBRR0H = UBBRVAL >> 8;
34     UBRR0L = UBBRVAL & 0xFF;
35     UCSR0B |= (1 << TXEN0) | (1 << RXEN0);
36     UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00);
37     (void) UDR0;
38 }
39
40 void u_putchar (uint8_t x) {
41     while (!(UCSR0A & (1 << UDRE0)));
42     UDR0 = x;
43 }
44
45 void u_puts (char *s) {
46     while (*s) {
47         u_putchar (*s++);
48     }
49 }
50
51 void sendCurrentVoltage (void) {
52     ADCSRA |= (1 << ADSC);
53     while (ADCSRA & (1 << ADSC));
54     uint16_t out = ADC;
55     double voltage = (out/1024.0) * 4.5;
56
57     char str[10];
58     sprintf(str, "% .3f \r\n", voltage);
59     u_puts(str);
60 }
61
62 void delay (uint16_t millisec) {
63     while (millisec--) {
64         _delay_ms(1);
65     }
66 }
```

Quellcode 1: Das Programm für den Mikrocontroller

5.1 Grundlegender Aufbau und Konzept der App



Abbildung 5: Die Startseite der App

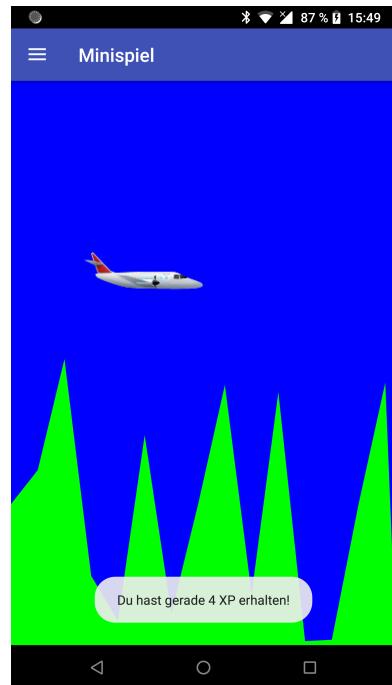


Abbildung 7: Das Minispiel

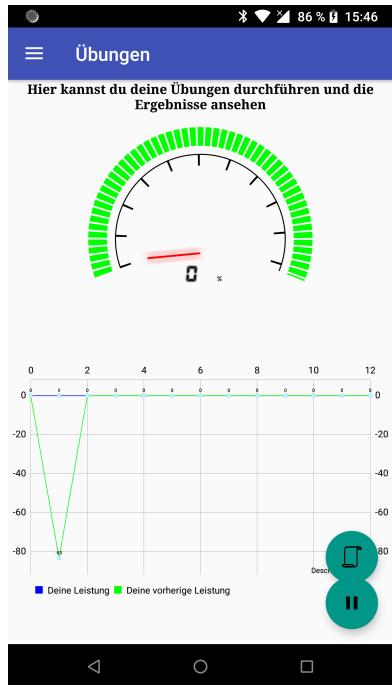


Abbildung 6: Die Übungsseite der App

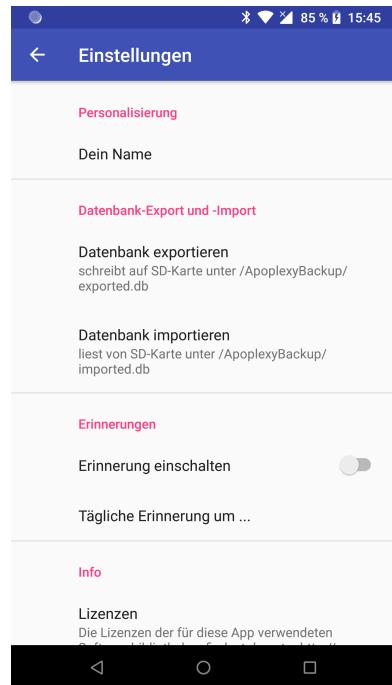


Abbildung 8: Die Einstellungsseite der App

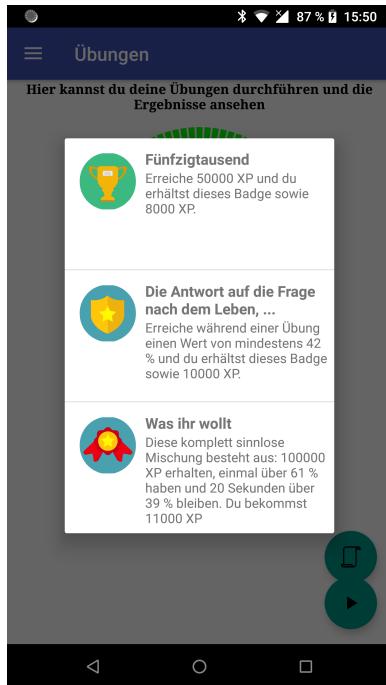


Abbildung 9: Die Liste der verfügbaren Quests

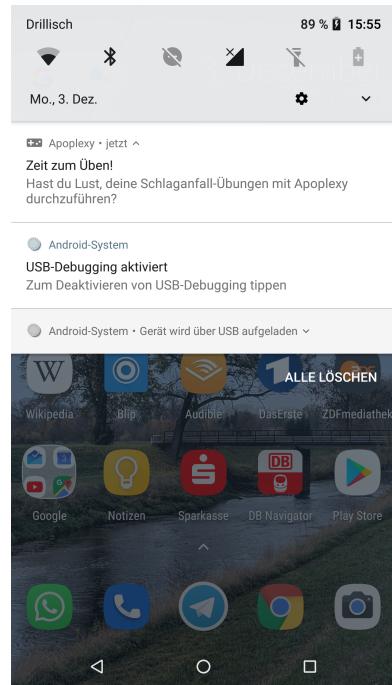


Abbildung 10: Eine Übungsbenachrichtigung

5.2 Kommunikation mit dem Mikrocontroller

```

1 package de.lukasrost.apoplexy
2
3 import android.bluetooth.BluetoothDevice
4 import android.bluetooth.BluetoothSocket
5 import java.nio.charset.StandardCharsets
6 import java.util.*
7
8 // Steuerung der Bluetooth-Verbindung
9 class BluetoothNoService {
10     // Bluetooth-Gerät und Socket
11     private lateinit var device: BluetoothDevice
12     private var bluetoothSocket: BluetoothSocket? = null
13
14     // Queue der Messwerte
15     private val btQueue = mutableListOf<Double>()
16
17     // Thread zum Updaten der Queue
18     private var keepRunning = false
19     private val updateQueueRunnable = Runnable {
20         var read = 0
21         val data = ByteArray(1024)
22
23         // wenn verbunden
24         if (bluetoothSocket != null) {
25
26             // solange nicht gestoppt und Datenempfang vorhanden
27             while (this@BluetoothNoService.keepRunning &&
28                 ((bluetoothSocket!!.inputStream.read(data).let { read =
29                     it; it != -1 }))) {
30
31                 // Einlesen der Bluetooth-Daten
32                 val readdata = Arrays.copyOf(data, read)
33                 val value = String(readdata, StandardCharsets.UTF_8)
34             }
35         }
36     }
37 }
```

```

32         // Daten der Queue hinzufügen
33         for (number in value.split("\r\n")) {
34             if (btQueue.size == 10) {
35                 btQueue.removeAt(0)
36             }
37             val num = number.toDoubleOrNull()
38             num?.let { if( num < 1.5) btQueue.add(1.5) else
39                         → btQueue.add(it) } // nur Werte > 1.5
40         }
41     }
42 }
43
44
45 // Verbindung beginnen
46 fun establishConnection(device: BluetoothDevice) {
47     this.device = device
48 }
49
50 // mit Bluetooth-Gerät verbinden und Thread starten
51 fun startReading() {
52     bluetoothSocket = device.createRfcommSocketToServiceRecord(UUID.↓
53         ← fromString("00001101-0000-1000-8000-00805F9B34FB"))
54     bluetoothSocket?.connect()
55     keepRunning = true
56     Thread(updateQueueRunnable).start()
57 }
58
59 // Verbindung beenden, Thread stoppen
60 fun stopReading() {
61     keepRunning = false
62     Thread.sleep(1000)
63     bluetoothSocket?.close()
64 }
65
66 // aktuellen Durchschnittswert der Queue in Prozent des Maximalwerts
67 // berechnen
68 fun getCurrentValuePercent() : Float = 100 * ((btQueue.median() -
69     ← MIN_VOLTAGE_EMG) / (MAX_VOLTAGE_EMG - MIN_VOLTAGE_EMG)).toFloat()
70 }
```

Quellcode 2: Die BluetoothNoService-Klasse mit der Bluetooth-Funktionalität

5.3 Konzept und programmiertechnische Umsetzung des Minispiels

```

1 package de.lukasrost.apoplexy.game
2
3 import android.annotation.SuppressLint
4 import android.content.Context
5 import android.graphics.∗
6 import android.support.v4.app.FragmentActivity
7 import android.util.AttributeSet
8 import android.view.GestureDetector
9 import android.view.MotionEvent
10 import android.view.View
11 import de.lukasrost.apoplexy.BluetoothNoService
12 import de.lukasrost.apoplexy.R
13 import de.lukasrost.apoplexy.helpers.GamificationGraderHelper
```

```

14 import java.util.*
15
16 // eigene View, beinhaltet das Minispiel
17 class PlaneGameView : View {
18     // Konstruktoren: zeichnet sich nicht selbst
19     constructor(context : Context) : super(context) {
20         setWillNotDraw(false)
21     }
22     constructor(context: Context, attributeSet: AttributeSet) :
23         super(context,attributeSet){
24         setWillNotDraw(false)
25     }
26
27     // Bluetooth- und Gamification-Variablen
28     private lateinit var bluetoothNoService : BluetoothNoService
29     private var bluetoothPercList = mutableListOf<Float>()
30     private var inGame = false
31     private val graderHelper = GamificationGraderHelper(context)
32
33     // Höhe und Breite der View
34     private var effWidth = width - (paddingLeft + paddingRight)
35     private var effHeight = height - (paddingTop + paddingBottom)
36
37     // Zufällige Berge als Queue
38     private val random = Random()
39     private var randomHills = mutableListOf<Int>()
40
41     // Gesten-Detektor für Klicks
42     private val gestureListener = object
43         : GestureDetector.SimpleOnGestureListener() {
44             override fun onDown(e: MotionEvent?): Boolean {
45                 return true
46             }
47         }
48     private val gestureDetector = GestureDetector(context,gestureListener)
49
50     // Texte
51     private val pausedText = "Um das Spiel zu beginnen, berühre den"
52         " Bildschirm."
53     private val lostText = "Du bist gegen einen Berg gestoßen. Du"
54         " verlierst!"
55
56     // Hintergrund-Rechteck
57     private var backgroundRect =
58         Rect(paddingLeft,paddingTop,effWidth,effHeight)
59     // Farben, Stile, Textgrößen
60     private val backgroundPaint = Paint(0).apply {
61         color = Color.BLUE
62         style = Paint.Style.FILL
63     }
64     private val hillPaint = Paint(Paint.ANTI_ALIAS_FLAG).apply {
65         color = Color.GREEN
66         style = Paint.Style.FILL
67     }
68     private val textPaint = Paint(Paint.ANTI_ALIAS_FLAG).apply {
69         style = Paint.Style.FILL
70         textSize = 60f
71         color = Color.BLACK
72     }

```

```

68     private val planePaint = Paint(Paint.ANTI_ALIAS_FLAG)
69
70     // Flugzeug-Bild (nur halb so groß wie Originalbild)
71     private var opts = BitmapFactory.Options().apply {
72         inSampleSize = 2
73     }
74     private val planeBitmap =
75         → BitmapFactory.decodeResource(context.resources,
76             → R.drawable.ic_airplane,opts)
77
78     // Zeichnen der View
79     override fun onDraw(canvas: Canvas?) {
80         super.onDraw(canvas)
81         if(inGame) {
82             // im Spiel
83             canvas?.apply {
84                 bluetoothPercList.add(bluetoothNoService.|)
85                 → getCurrentValuePercent())
86                 // Verschiebung des Flugzeugs nach oben
87                 val verschieb = 70
88                 // obere linke Ecke des Flugzeugs durch *Magie* bestimmen
89                 val top = paddingTop + effHeight - (effHeight *
90                     → (bluetoothPercList[bluetoothPercList.size-1] +
91                     verschieb) / 100 )
92                 val left = paddingLeft + effWidth/2 - planeBitmap.width
93
94                 // Kollision mit Hügel vor Flugzeug -> verloren
95                 if(top + planeBitmap.height >= paddingTop + effHeight -
96                     → randomHills[randomHills.size/2]){
97                     drawText(lostText,50f,(effHeight/2).|)
98                     → toFloat(),textPaint)
99                     Thread.sleep(2500)
100                    // Spiel beenden
101                    handleTap()
102                } else {
103                    // Spiellandschaft (Flugzeug, Himmel, Berge) zeichnen
104                    drawRect(backgroundRect, backgroundPaint)
105                    drawHills(this)
106                    drawBitmap(planeBitmap, left.toFloat(), top,
107                         → planePaint)
108                }
109            }
110        // Berge zeichen
111        private fun drawHills(canvas: Canvas){
112            // Höhe des nächsten Bergs zufällig bestimmen
113            val k = random.nextInt(effHeight) - effHeight / 5
114            randomHills.add(if (k < 0) effHeight / 4 else k)
115            randomHills.removeAt(0)
116
117            // Berge im Abstand von 100 Pixel zeichnen
118            val p = Path()

```

```

119     canvas.apply {
120         var offset = 0
121         for (hill in randomHills) {
122             p.lineTo((paddingLeft+offset).toFloat(), (paddingTop+
123                         ↵ effHeight - hill).toFloat())
124             offset += 100
125         }
126         p.lineTo(paddingLeft +
127                         ↵ effWidth).toFloat(), (paddingTop+effHeight).toFloat())
128         p.lineTo(paddingLeft.toFloat(), (paddingTop+effHeight).]
129                         ↵ toFloat())
130         p.close()
131         drawPath(p, hillPaint)
132     }
133 }
134
135 // Bildschirmgröße verändert -> abhängige Werte anpassen
136 override fun onSizeChanged(w: Int, h: Int, oldw: Int, oldh: Int) {
137     effWidth = width - (paddingLeft + paddingRight)
138     effHeight = height - (paddingTop + paddingBottom)
139     backgroundRect = Rect(paddingLeft, paddingTop, effWidth, effHeight)
140     genRandomHills()
141     super.onSizeChanged(w, h, oldw, oldh)
142 }
143
144 // neue initiale Berge-Liste zufällig generieren
145 private fun genRandomHills(){
146     randomHills = mutableListOf<Int>().apply {
147         for (i in 0..effWidth/100){
148             val k = random.nextInt( effHeight) - effHeight / 5
149             add(if (k < 0) effHeight / 4 else k)
150         }
151     }
152 }
153
154 // diese und nächste Funktion setzen Höhe und Breite der View bei
155 // Anforderung durch System
156 override fun onMeasure(widthMeasureSpec: Int, heightMeasureSpec: Int)
157     ↵ {
158
159     val desiredWidth = suggestedMinimumWidth + paddingLeft +
160                         ↵ paddingRight
161     val desiredHeight = suggestedMinimumHeight + paddingTop +
162                         ↵ paddingBottom
163
164     setMeasuredDimension(measureDimension(desiredWidth,
165                                         ↵ widthMeasureSpec),
166                           measureDimension(desiredHeight, heightMeasureSpec))
167 }
168
169 private fun measureDimension(desiredSize: Int, measureSpec: Int): Int
170     ↵ {
171     var result: Int
172     val specMode = View.MeasureSpec.getMode(measureSpec)
173     val specSize = View.MeasureSpec.getSize(measureSpec)
174
175     if (specMode == View.MeasureSpec.EXACTLY) {
176         result = specSize
177     } else {
178

```

```

169         result = desiredSize
170         if (specMode == View.MeasureSpec.AT_MOST) {
171             result = Math.min(result, specSize)
172         }
173     }
174     return result
175 }
176
177 // bei Klick Spiel starten bzw. beenden
178 @SuppressLint("ClickableViewAccessibility")
179 override fun onTouchEvent(event: MotionEvent?): Boolean {
180     return gestureDetector.onTouchEvent(event).let {
181         if(it){
182             handleTap()
183             true
184         } else false
185     }
186 }
187
188 // Spiel starten bzw. beenden
189 private fun handleTap() {
190     inGame = !inGame
191     if(!inGame) {
192         // Beenden
193         // Gamification durchführen
194         graderHelper.gradeForGame(bluetoothPercList.size,
195             bluetoothPercList)
196         val fragment =
197             graderHelper.checkBadgesForCompletion(bluetoothPercList)
198         val activity = context as FragmentActivity
199         activity.runOnUiThread { fragment?.show(activity.|
200             supportFragmentManager, "badgecompleted")
201             }
202
203         // Bluetooth- und Berge-Listen leeren
204         bluetoothPercList = mutableListOf()
205         randomHills = mutableListOf()
206     } else {
207         // Starten -> Berge generieren
208         genRandomHills()
209     }
210     // View neu zeichnen
211     invalidate()
212 }
213
214 fun setBluetoothNoService(bb : BluetoothNoService){
215     bluetoothNoService = bb
216     bluetoothNoService.startReading()
217 }
218 }
```

Quellcode 3: Die PlaneGameView-Klasse mit der Implementierung des Minispiele

5.4 Umsetzung der Gamification in der App

```
1 package de.lukasrost.apoplexy.helpers
```

```
2
```

```

3 import android.app.Activity
4 import android.content.Context
5 import android.preference.PreferenceManager
6 import android.widget.Toast
7 import de.lukasrost.apoplexy.*
8 import de.lukasrost.apoplexy.badges.*
9 import kotlin.math.roundToInt

10
11 // Bewertungshelfer für Gamification-Funktionen
12 class GamificationGraderHelper(val context: Context) {
13     private val prefs =
14         PreferenceManager.getDefaultSharedPreferences(context)
15     private val activity = context as Activity
16     private lateinit var dbHelper : GamificationDBHelper

17     // XP für Übungen vergeben
18     fun gradeForExercise(data : MutableList<Float>){
19         // Minimal-, Maximal- und Durchschnittswert der Liste bestimmen
20         val min = data.min()?:0f
21         val max = data.max()?:0f
22         val avg = if (data.average().isNaN()) 0.0 else data.average()

23
24         // Bewertungsfunktion
25         val points = ((avg + min + max) / 3 )
26         val pointsInt = if(!points.isNaN() && points.roundToInt() > 0)
27             points.roundToInt() else 0

28         // neue XP zu bisherigen XP hinzufügen
29         val pointsBefore = prefs.getInt(PREFS_POINTS,0)
30         prefs.edit().putInt(PREFS_POINTS,pointsBefore + pointsInt).apply()
31         activity.runOnUiThread { Toast.makeText(context,"Du hast gerade
32             → $pointsInt XP erhalten!",Toast.LENGTH_LONG).show() }

33
34     // XP für Minispiel vergeben
35     fun gradeForGame(distanceUntilCrash: Int, data: MutableList<Float>){
36         // Minimal-, Maximal- und Durchschnittswert der Liste bestimmen
37         val min = data.min()?:0f
38         val max = data.max()?:0f
39         val avg = if (data.average().isNaN()) 0.0 else data.average()

40
41         // Bewertungsfunktion
42         val points = ((avg + min + max) / 3 ) + distanceUntilCrash * 2
43         val pointsInt = if(!points.isNaN() && points.roundToInt() > 0)
44             points.roundToInt() else 0

45         // neue XP zu bisherigen XP hinzufügen
46         val pointsBefore = prefs.getInt(PREFS_POINTS,0)
47         prefs.edit().putInt(PREFS_POINTS,pointsBefore + pointsInt).apply()
48         activity.runOnUiThread { Toast.makeText(context,"Du hast gerade
49             → $pointsInt XP erhalten!",Toast.LENGTH_LONG).show() }

50
51     // Freischaltung von Badges überprüfen
52     fun checkBadgesForCompletion(data: MutableList<Float>) :
53         BadgeDialogFragment?{
54         dbHelper = GamificationDBHelper(context)
55         val cursor = dbHelper.getAvailableQuests()
56         var shouldShowDialog = false

```

```

56
57     var icon = 0
58     var title = ""
59     var earnedXP = 0
60
61     // durch alle verfügbaren Quests iterieren
62     while (cursor.moveToFirst()) {
63         val neededXP =
64             cursor.getInt(cursor.getColumnIndex(QUEST_FIN_XP))
65         val minPerc =
66             cursor.getInt(cursor.getColumnIndex(QUEST_MIN_PERC))
67         val isMinPercCompleted = data.any { it >= minPerc.toFloat() }
68         val isOverPercCompleted = checkOverPercCondition(data, cursor.)
69             getInt(cursor.getColumnIndex(QUEST_OVER_PERC)), cursor.
70             getInt(cursor.getColumnIndex(QUEST_TIME_OVER_PERC)))
71
72         // spezifische Bedingungen überprüfen
73         if (neededXP <= prefs.getInt(PREFS_POINTS, 0) &&
74             isMinPercCompleted && isOverPercCompleted) {
75
76             // Quest fertig -> Badge freigeschaltet
77             dbHelper.setQuestCompleted(cursor.getInt(cursor.
78                 getColumnIndex(QUEST_ID)))
79
80             // entsprechend XP vergeben
81             val pointsBefore = prefs.getInt(PREFS_POINTS, 0)
82             prefs.edit().putInt(PREFS_POINTS, pointsBefore + cursor.
83                 getInt(cursor.getColumnIndex(QUEST_EARN_XP))).apply()
84
85             // Dialog soll angezeigt werden
86             icon = cursor.getInt(cursor.getColumnIndex(QUEST_ICON))
87             title += "\\" +
88                 cursor.getString(cursor.getColumnIndex(QUEST_TITLE)) +
89                 "\\", "
90             earnedXP +=
91                 cursor.getInt(cursor.getColumnIndex(QUEST_EARN_XP))
92             shouldShowDialog = true
93         }
94     }
95     cursor.close()
96     dbHelper.close()
97
98     // Dialog an Aufrufer zurückgeben
99     if (shouldShowDialog) {
100         title = title.substring(0, title.length - 2)
101         return BadgeDialogFragment().setDialogInformation(icon, title,
102             earnedXP)
103     }
104     return null
105 }
106
107 // Überprüfen der Bedingung, dass man bestimmte Zeit über bestimmtem
108 // Prozentwert war
109 private fun checkOverPercCondition(data: MutableList<Float>, overPerc:
110     : Int, timeOverPerc: Int): Boolean{
111     var count = 0;
112     for (el in data) {
113         if (el >= overPerc) {
114             count++
115         } else {
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
823
824
825
825
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1660
1661
1661
1662
1662
1663
1663
1664
1664
1665
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1670
1671
1671
1672
1672
1673
1673
1674
1674
1675
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1680
1681
1681
1682
1682
1683
1683
1684
1684
1685
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1690
1691
1691
1692
1692
1693
1693
1694
1694
1695
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1700
1701
1701
1702
1702
1703
1703
1704
1704
1705
1705
1706
1706
1707
1707
1708
1708
1709
1709
1710
1710
1711
1711
1712
1712
1713
1713
1714
1714
1715
1715
1716
1716
1717
1717
1718
1718
1719
1719
1720
1720
1721
1721
1722
1722
1723
1723
1724
1724
1725
1725
1726
1726
1727
1727
1728
1728
1729
1729
1730
1730
1731
1731
1732
1732
1733
1733
1734
1734
1735
1735
1736
1736
1737
1737
1738
1738
1739
1739
1740
1740
1741
1741
1742
1742
1743
1743
1744

```

```

102         count = 0
103     }
104     if (count >= timeOverPerc) {
105         return true
106     }
107     }
108     return false
109 }
110 }
```

Quellcode 4: Die GamificationGraderHelper-Klasse mit den Gamification-Bewertungsfunktionen

```

1 package de.lukasrost.apoplexy.helpers
2
3 import android.content.ContentValues
4 import android.content.Context
5 import android.database.Cursor
6 import android.database.sqlite.SQLiteDatabase
7 import android.database.sqlite.SQLiteOpenHelper
8 import android.preference.PreferenceManager
9 import de.lukasrost.apoplexy.*
10
11 // Verbindung zur Gamification-Datenbank
12 class GamificationDBHelper(val context: Context) :
13     SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {
14     // Tabelle erstellen, Standard-Quests einfügen
15     override fun onCreate(db: SQLiteDatabase?) {
16         db?.execSQL(CREATE_TABLE_GAMIFICATION)
17         questOne(db!!)
18         questTwo(db)
19         questThree(db)
20         questFour(db)
21         questFive(db)
22     }
23
24     // Upgrade der Tabelle bei neuer Datenbankversion
25     override fun onUpgrade(db: SQLiteDatabase?, old: Int, new: Int) {
26         db?.execSQL(DROP_TABLE_GAMIFICATION)
27         onCreate(db)
28     }
29
30     // alle erledigten Quests / erreichten Badges zurückgeben
31     fun getCompletedQuests(): Cursor{
32         return readableDatabase.|
33             query(TABLE_NAME_GAMIFICATION, null, "$QUEST_COMPLETED =
34             1", null, null, null, "$QUEST_ID ASC")
35     }
36
37     // alle gerade durchführbaren Quests zurückgeben
38     fun getAvailableQuests(): Cursor{
39         val xp = PreferenceManager.getDefaultSharedPreferences(context).|
40             getInt(PREFS_POINTS, 0)
41         return readableDatabase.|
42             query(TABLE_NAME_GAMIFICATION, null, "$QUEST_REQ_XP <= ? AND
43             $QUEST_COMPLETED = 0", arrayOf(xp.toString()),
44             null, null, "$QUEST_REQ_XP ASC")
45     }
46 }
```

```

40
41 // Quest auf erledigt setzen
42 fun setQuestCompleted(id: Int) {
43     val values = ContentValues()
44     values.put(QUEST_COMPLETED, 1)
45     writableDatabase.update(TABLE_NAME_GAMIFICATION, values, "$QUEST_ID
46     ↪ = ?", arrayOf(id.toString()))
47     writableDatabase.close()
48 }
49
50 // die folgenden Funktionen fügen die Standard-Quests in die Datenbank
51 ↪ ein
52 fun questOne(db: SQLiteDatabase) {
53     val values = ContentValues()
54     values.put(QUEST_TITLE, "Fünfzigtausend")
55     values.put(QUEST_DESCRIPTION, "Erreiche 50000 XP und du erhältst
56     ↪ dieses Badge sowie 8000 XP.")
57     values.put(QUEST_ICON, ICON_ONE)
58     values.put(QUEST_COMPLETED, 0)
59     values.put(QUEST_REQ_XP, 0)
60     values.put(QUEST_FIN_XP, 50000)
61     values.put(QUEST_EARN_XP, 8000)
62     values.put(QUEST_MIN_PERC, 0)
63     values.put(QUEST_OVER_PERC, 0)
64     values.put(QUEST_TIME_OVER_PERC, 0)
65     db.insert(TABLE_NAME_GAMIFICATION, null, values)
66 }
67
68 fun questTwo(db: SQLiteDatabase) {
69     val values = ContentValues()
70     values.put(QUEST_TITLE, "Die Antwort auf die Frage nach dem Leben,
71     ↪ ...")
72     values.put(QUEST_DESCRIPTION, "Erreiche während einer Übung einen
73     ↪ Wert von mindestens 42 % und du erhältst dieses Badge sowie
74     ↪ 10000 XP.")
75     values.put(QUEST_ICON, ICON_TWO)
76     values.put(QUEST_COMPLETED, 0)
77     values.put(QUEST_REQ_XP, 0)
78     values.put(QUEST_FIN_XP, 0)
79     values.put(QUEST_EARN_XP, 10000)
80     values.put(QUEST_MIN_PERC, 42)
81     values.put(QUEST_OVER_PERC, 0)
82     values.put(QUEST_TIME_OVER_PERC, 0)
83     db.insert(TABLE_NAME_GAMIFICATION, null, values)
84 }
85
86 fun questThree(db: SQLiteDatabase) {
87     val values = ContentValues()
88     values.put(QUEST_TITLE, "Das Problem mit der Ausdauer")
89     values.put(QUEST_DESCRIPTION, "Halte 20 Sekunden lang einen Wert
90     ↪ von mindestens 50 %. Als Belohnung gibt's zusätzlich 6000
91     ↪ XP.")
92     values.put(QUEST_ICON, ICON_THREE)
93     values.put(QUEST_COMPLETED, 0)
94     values.put(QUEST_REQ_XP, 15000)
95     values.put(QUEST_FIN_XP, 0)
96     values.put(QUEST_EARN_XP, 6000)
97     values.put(QUEST_MIN_PERC, 0)
98     values.put(QUEST_OVER_PERC, 50)

```

```

91     values.put(QUEST_TIME_OVER_PERC, 20)
92     db.insert(TABLE_NAME_GAMIFICATION, null, values)
93 }
94
95 fun questFour(db: SQLiteDatabase) {
96     val values = ContentValues()
97     values.put(QUEST_TITLE, "Was ihr wollt")
98     values.put(QUEST_DESCRIPTION, "Diese komplett sinnlose Mischung
99     ↳ besteht aus: 100000 XP erhalten, einmal über 61 % haben und 20
100    ↳ Sekunden über 39 % bleiben. Du bekommst 11000 XP")
101    values.put(QUEST_ICON, ICON_FOUR)
102    values.put(QUEST_COMPLETED, 0)
103    values.put(QUEST_REQ_XP, 0)
104    values.put(QUEST_FIN_XP, 100000)
105    values.put(QUEST_EARN_XP, 11000)
106    values.put(QUEST_MIN_PERC, 61)
107    values.put(QUEST_OVER_PERC, 39)
108    values.put(QUEST_TIME_OVER_PERC, 20)
109    db.insert(TABLE_NAME_GAMIFICATION, null, values)
110 }
111
112 fun questFive(db: SQLiteDatabase) {
113     val values = ContentValues()
114     values.put(QUEST_TITLE, "Viel zu viel")
115     values.put(QUEST_DESCRIPTION, "Eine Million XP und einmal über 80 %
116     ↳ erreichen. Schaffst du das?")
117     values.put(QUEST_ICON, ICON_FIVE)
118     values.put(QUEST_COMPLETED, 0)
119     values.put(QUEST_REQ_XP, 20000)
120     values.put(QUEST_FIN_XP, 1000000)
121     values.put(QUEST_EARN_XP, 20000)
122     values.put(QUEST_MIN_PERC, 80)
123     values.put(QUEST_OVER_PERC, 0)
124     values.put(QUEST_TIME_OVER_PERC, 0)
125     db.insert(TABLE_NAME_GAMIFICATION, null, values)
126 }

```

Quellcode 5: Die GamificationDBHelper-Klasse, welche für den Zugriff auf die Datenbank verantwortlich ist.

5.5 Funktionsweise des Benachrichtigungssystems

```

1 package de.lukasrost.apoplexy.notifications
2
3 import android.content.Context
4 import android.preference.PreferenceManager
5 import java.util.*
6 import android.content.pm.PackageManager
7 import android.content.ComponentName
8 import android.app.AlarmManager
9 import android.content.Context.ALARM_SERVICE
10 import android.app.PendingIntent
11 import android.content.Intent
12 import android.support.v4.app.NotificationCompat
13 import android.app.NotificationManager
14 import android.support.v4.content.ContextCompat.getSystemService
15 import android.app.NotificationChannel

```

```

16 import android.os.Build
17 import android.support.v4.app.NotificationManagerCompat
18 import de.lukasrost.apoplexy.*
19
20 // Funktionen zur Benachrichtigungsverwaltung
21
22 // Benachrichtigungszeitpunkt neu setzen
23 fun setReminder(ctx: Context) {
24     prefs = PreferenceManager.getDefaultSharedPreferences(ctx)
25
26     // wenn Benachrichtigungen aktiviert
27     if (prefs.getBoolean(PREFS_ALARM_ENABLED, false)) {
28
29         // Zeit aus Einstellungen lesen
30         val time = prefs.getString(PREFS_ALARM_TIME,
31             "00:00")?.split(":")!!
32         val hour = time[0].toInt()
33         val minute = time[1].toInt()
34         val calendar = Calendar.getInstance()
35         val setcalendar = Calendar.getInstance()
36
37         // Zeit setzen
38         setcalendar.set(Calendar.HOUR_OF_DAY, hour)
39         setcalendar.set(Calendar.MINUTE, minute)
40         setcalendar.set(Calendar.SECOND, 0)
41
42         // bisherige Reminder löschen
43         cancelReminder(ctx)
44
45         // wenn Zeit heute schon vergangen -> für morgen setzen
46         if (setcalendar.before(calendar)) {
47             setcalendar.add(Calendar.DATE, 1)
48         }
49
50         // Receiver des Alarms setzen
51         val receiver = ComponentName(ctx,
52             NotificationReceiver::class.java)
53         ctx.packageManager.setComponentEnabledSetting(receiver,
54             PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
55             PackageManager.DONT_KILL_APP)
56         val intent1 = Intent(ctx, NotificationReceiver::class.java)
57         val pendingIntent = PendingIntent.getBroadcast(ctx,
58             DAILY_RemINDER_CODE, intent1,
59             PendingIntent.FLAG_UPDATE_CURRENT)
60
61         // Benachrichtigung dem AlarmManager übergeben
62         val am = ctx.getSystemService(ALARM_SERVICE) as AlarmManager
63         am.setInexactRepeating(AlarmManager.RTC_WAKEUP,
64             setcalendar.timeInMillis,
65             AlarmManager.INTERVAL_DAY, pendingIntent)
66     } else {
67         // wenn Benachrichtigung deaktiviert -> Erinnerung löschen
68         cancelReminder(ctx)
69     }
70 }
71
72 // Alarm des AlarmManagers mithilfe von *Magie* löschen
73 fun cancelReminder(context: Context) {

```

```

71     val receiver = ComponentName(context,
72         ↳ NotificationReceiver::class.java)
73     val pm = context.packageManager
74     pm.setComponentEnabledSetting(receiver,
75         PackageManager.COMPONENT_ENABLED_STATE_DISABLED,
76         PackageManager.DONT_KILL_APP)
77
77     val intent1 = Intent(context, NotificationReceiver::class.java)
78     val pendingIntent = PendingIntent.getBroadcast(context,
79         DAILY_RemINDER_CODE, intent1,
80         ↳ PendingIntent.FLAG_UPDATE_CURRENT)
80     val am = context.getSystemService(ALARM_SERVICE) as AlarmManager
81     am.cancel(pendingIntent)
82     pendingIntent.cancel()
83 }
84
85 // Benachrichtigung anzeigen
86 fun showNotification(context: Context) {
87
88     // Activity, die geöffnet werden soll
89     val intent = Intent(context, HomeNavActivity::class.java)
90     intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or
91         Intent.FLAG_ACTIVITY_CLEAR_TASK
92     val pendingIntent = PendingIntent.getActivity(context, 0, intent, 0)
93
94     // Aussehen der Benachrichtigung
95     val builder = NotificationCompat.Builder(context,
96         ↳ DAILY_RemINDER_CHANNEL_ID)
97         .setSmallIcon(R.drawable.ic_minigame_icon)
98         .setContentTitle("Zeit zum Üben!")
99         .setContentText("Hast du Lust, deine Schlaganfall-Übungen mit
100             ↳ Apoplexy durchzuführen?")
101         ..setStyle(NotificationCompat.BigTextStyle()
102             .bigText("Hast du Lust, deine Schlaganfall-Übungen mit
103                 ↳ Apoplexy durchzuführen?"))
104         .setPriority(NotificationCompat.PRIORITY_DEFAULT)
105         // Set the intent that will fire when the user taps the
106         ↳ notification
107         .setContentIntent(pendingIntent)
108         .setAutoCancel(true)
109
110     // Benachrichtigung senden
111     val notificationManager = NotificationManagerCompat.from(context)
112     notificationManager.notify(DAILY_RemINDER_CODE, builder.build())
113 }
114
115 // Benachrichtigungskanal auf unterstützten Geräten erstellen
116 fun createNotificationChannel(context: Context) {
117     if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
118         val name = "Erinnerungen"
119         val description = "Apoplexes Übungserinnerungen"
120         val importance = NotificationManager.IMPORTANCE_DEFAULT
121         val channel = NotificationChannel(DAILY_RemINDER_CHANNEL_ID, name,
122             ↳ importance)
123         channel.description = description
124
125         val notificationManager =
126             ↳ getSystemService(context, NotificationManager::class.java)
127         notificationManager.createNotificationChannel(channel)

```

```
121      }
122 }
```

Quellcode 6: Der NotificationScheduler zur Verwaltung der Benachrichtigungen

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides Statt, dass ich meine Seminarfacharbeit „Entwicklung eines Gamification-basierten Unterstützungs- und Motivationsgeräts zur Rehabilitation von Schlaganfall-Patienten“ selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel verfasst habe. Außerdem erkläre ich, dass ich alle wörtlich oder sinngemäß aus fremden Werken entnommene Stellen als solche kenntlich gemacht habe.

Die vorliegende Arbeit entspricht in ihrer Vollständigkeit der auf der gegebenenfalls beigelegten DVD gespeicherten digitalen Version.

Erfurt, 20. Dezember 2018

Lukas Rost

Digitale Version

