Technical Report

hochschule 21 : Nr. 8

# Projektbericht „Parkinson-Brille“

Thorsten Hermes, Christoph Ebler, Nantwin Möller, Niko Krofta

hochschule 21-Bericht, Nr. 8

2017

**hochschule 21-Berichte**

Herausgeber:
hochschule 21 gemeinnützige GmbH
Staatlich anerkannte private Fachhochschule
Harburger Straße 6
21614 Buxtehude
Telefon: +49 4161 648 0
Fax: +49 4161 648 123
E-Mail: info@hs21.de
http://www.hs21.de
ISSN 2196-5153

# Projektbericht "Parkinson-Brille"

Thorsten Hermes, Christoph Ebler, Nantwin Möller, Niko Krofta
hochschule 21 gemeinnützige GmbH
Staatlich anerkannte private Hochschule
21614 Buxtehude
*hermes*@hs21.de,
*christoph.ebler|nantwin.moeller|niko.krofta*@stud.hs21.de

6. Juli 2017

**Zusammenfassung**

Aufgrund der aktuellen demographischen Entwicklung wächst der Anteil der älteren Menschen an der Weltbevölkerung stetig. Damit nimmt auch die Anzahl der von Parkinson Morbus Betroffenen zu. Das so genannte *Freeizing of gait* ist ein mögliches Symptom dieser Krankheit. Untersuchen geben Hinweise, dass die Zustände durch Applizierung von auditiven und/oder visuellen Cues (Schlüsselreizen) vermieden oder doch zumindest abgeschwächt werden können (je nach Schweregrad der Erkrankung).

Diese Arbeit berichtet über den aktuellen Stand des Projektes an der hochschule 21, in dem Studierende der Mechatronik eine so genannte Parkinson-Brille entwickeln, die genau diese auditiven und visuellen Cues applizieren soll.

## 1 Einleitung

Aktuell ist Morbus Parkinson die zweithäufigste neurodegenerative Erkrankung [12]. Nach Schätzungen sind weltweit über vier Millionen Menschen an Parkinson erkrankt. Das entspricht nicht ganz zwei Prozent der Bevölkerung der über 60 Jährigen. Weiterhin wird davon ausgegangen, dass sich aufgrund der Demographie sowie durch die zunehmend bessere Behandlung und die damit verbundene längere Lebenszeit die Zahl der Patienten bis 2030 auf ca. 8,7 Millionen weltweit verdoppelt. In der Regel sind die Patienten bei der Diagnose im Mittel 60 Jahre alt. Für Deutschland ergeben sich folgende

Werte: Es sind zwischen 250.000 bis 280.000 Personen von der Krankheit betroffen.

Parkinson-Patienten können unter dem Ausfall automatischer Bewegungsprozesse leiden. Dieses wird unter anderem besonders beim Gehen deutlich. Die Patienten leiden vielfach unter dem so genannten *Freezing of gait* Phänomen [20], welches ein „Einfrieren" der Bewegung bedeutet und ein flüssiges Gehen erheblich beeinflusst. An dieser Stelle ist das so genannte Cueing (setzen von externen Reizen) im Bereich der konventionellen physiotherapeutischen Intervention anerkannt und üblicherweise wird es manuell/personell durch das Darbieten von visuellen Cues (z.B. Streifen auf dem Boden kleben) und auditive Cues durch die Stimme des Therapeuten, ein Metronom oder ähnliches gegeben. Dieses stellt für den Alltag des Patienten aber keine Lösung dar.

Genau hier setzt dieses Projekt an. Auditive und visuelle Cues können auch mit Hilfe technischer Hilfsmittel appliziert werden, welche Patienten im täglichen Leben unterstützen [8]. Die angesprochenen auditiven und visuellen Cues können über eine Brille appliziert werden.

Dieser Bericht stellt den aktuellen Stand des Projektes „Parkinson-Brille" an der *hochschule 21* dar. Diese Projekt hat als Ziel ein *proof-of-concept* einer Datenbrille für die Applizierung von Schlüsselreizen zu implementieren.

# 2 Stand der Kunst

In [27] wird die Google Glass verwendet, um die Auswirkungen von externen rythmischen Reizen auf Parkinson-Patienten zu erproben.

Im Kontext dieses Projektes wird von Datenbrillen oder Smart Glasses gesprochen. Diese zeichnen sich dadurch aus, dass sie einerseits Daten erheben können, andererseits aber auch Daten über verschiedene Technologien (z.B. WLAN oder Bluetooth) an andere Geräte weitergeben können. Um Daten zu erheben, können sie mit einer Reihe von Sensoren ausgestattet sein, wie Kamera, GPS, Kompass, Gyroscope, Beschleunigungsmesser, usw. Mit diesen Sensoren wird die Umgebung analysiert und entweder direkt auf der Brille interpretiert oder über die erwähnten Technologien weitergegeben und z.B. von einem Smartphone oder einem Server interpretiert (vgl. [14]).

Als prinzipielle Kommunikationsmöglichkeiten mit dem Patienten stehen visuelle, akustische und sensorische Signale zur Verfügung:

- *Visuelle Signale* können durch verschiedene Displaytypen, die im Brillenglas wie bei der Epson Moverio (s. [3]) oder vor dem Brillenglas (s. [1]) verbaut wurden, übermittelt werden.

- *Akustische Signale* können entweder über Kopfhörer (via AUX-Buchse) oder über Bluetooth (s. [2]) an ein mit der Brille verbundenes Gerät weitergegeben werden. Es besteht außerdem die Möglichkeit, die Signale über einen BCT weiterzugeben (s. [1]), wie es auch bei der Glass von Google gemacht wurde.

- *Sensorische Signale* können über einen Vibrationsmotor, ähnlich wie in Mobiltelefonen, appliziert werden.

# 3 Eigener Ansatz

In diesem Kapitel wird zunächst auf den mechanischen Aufbau der Parkinson-Brille eingegangen. Anschließend erfolgt eine Beschreibung der aktuellen Energievorsorgen. Die IT Hardware wird dann im folgenden Abschnitt erläutert. Der Programmablauf bzw. die Software ist Gegenstand der nachfolgenden Betrachtungen. Den Abschluss bilden die beiden Abschnitte zu den Themengebieten Power Management bzw. Skripte und Netzwerk.

## 3.1 Mechanischer Aufbau und Recheneinheit

In diesem Abschnitt wird sowohl Brillengestell als auch die Recheneinheit thematisiert. Es wird dargestellt wie die Komponenten aufgebaut sind und welche Gründe dahinter stehen.

### 3.1.1 Brillengestell

Das Brillengestell ist einer 3D-Kinobrille nachempfunden. Diese sind, relativ zu gewöhnlichen Brillen, größer aufgebaut. Dies ist von Nutzen, da an der Brille diverse Komponenten angebracht werden müssen, um die Funktion zu gewährleisten und um dabei dem Aussehen einer normalen Brille zu ähneln.

Wie in Abbildung 1 zusehen ist, sind die Bügel der Brille extra breit ausgelegt. Dies bietet ausreichend Platz, um die Recheneinheit samt Gyroskop zu montieren. Der gegenüberliegende Bügel ist für die Montage einer Akkueinheit geplant. Der Knochenlautsprecher ist am hinteren Teil des Bügels angebracht. Er soll auf dem Knochen hinter dem Ohr anliegen. Das Display ist anstelle des Brillenglases eingebaut und zeigt die visuellen Cues an. Der Rahmen ist mittels 3D-Druckverfahren (vgl. [13]) aus *PLA* (Polylactid) hergestellt.

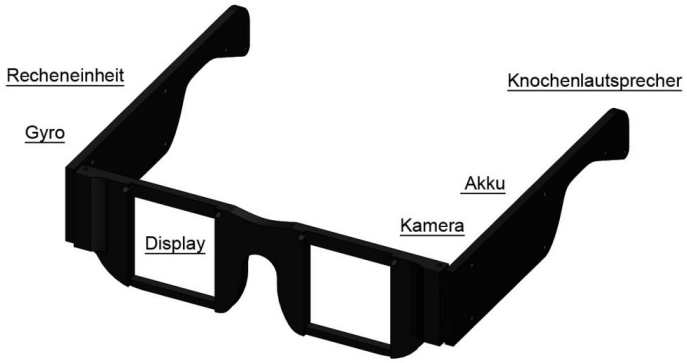


Abbildung 1: Darstellung Brillengestell mit den groben Vorortungen der einzelnen funktionalen Komponenten.

### 3.1.2 Bedienteil/Recheneinheit

Wegen des begrenzten Raumes an der Brille wurde die Hauptrecheneinheit der Brille in ein externes Bedienteil ausgelagert. Dieses Bedienteil besteht aus einem 14,4cm × 8,3cm × 2,9cm großen Gehäuse in welches ein 4 Zoll Display, die Recheneinheit und die Energieversorgung samt Laderegelung verbaut sind. An dem Bedienteil sind ein Schalter zum Ein- und Ausschalten des Geräts, eine Aufnahme für den Touchstift zur komfortablen Bedienung des Displays sowie eine Mikro-USB-Buchse als Ladeanschluss integriert.

## 3.2 Energieversorgung

Da es sich bei der Brille um ein mobiles System handelt, ist die Energieversorgung durch zwei Akkus realisiert. Diese, sowie die Laderegelung mit der zugehörigen Power Management Schaltung, werden im Folgenden beschrieben.

### 3.2.1 Akku

Die Brille sowie das Bedienteil werden von zwei Lithium-Ionen-Batterien (z.B. [7]) (Li-Ion) mit Strom versorgt. Es handelt sich dabei um 2 Zellen des Modells *US18650TVC6* der Marke Sony welche im Bedienteil verbaut sind. Die Batterie hat eine Nennspannung von 3,6 Volt bei einer Nennkapazität von

3120 Milliamperestunden. Der Vorteil dieser Batteriezellen ist der verhältnismäßig kleine Bauraum bei einer großen Kapazität. Außerdem bieten Li-Ion Akkus über einen weiten Kapazitätsbereich ein konstantes Spannungsniveau.

Die Batterien versorgen momentan das gesamte System (Brille und Bedienteil sind via Kabel verbunden). Das System kann für etwa 1,25h betrieben werden. Es wurde eine Stromaufnahme von ca. 2,4 Ampere gemessen.

### 3.2.2 Laderegelung

Li-Ion Batterien gelten im Bereich des Ladens und Endladens als empfindlich. Deshalb wurde eine Laderegelung integriert. Dafür werden je Batteriezelle ein *Powerboost 1000 Charger* von der Marke Adafruit verwendet (vgl. [18]). Dieser wird durch eine Mikro-USB-Buchse mit 5 Volt Ladespannung versorgt und lädt die Zellen mit einem Ladestrom von bis zu einem Ampere. Die interne Schaltung passt die Ladespannung für die Batteriezellen an. Der ebenfalls integrierte Überspannungsschutz schaltet die Ladespannung bei Erreichen der vollen Kapazität ab und schützt die Batterie davor zerstört zu werden. Der Unterspannungsschutz übernimmt die gleiche Aufgabe in die andere Richtung. Wird ein Kapazitätslevel unterschritten schalten die Spannungswächter die Ausgangsspannung ab.

### 3.2.3 Power Management Schaltung

Um den Energieverbrauch bei deaktiviertem Gerät so gering wie möglich zu halten, wurde eine die in Abbildung 2 zu sehende Platine entworfen.

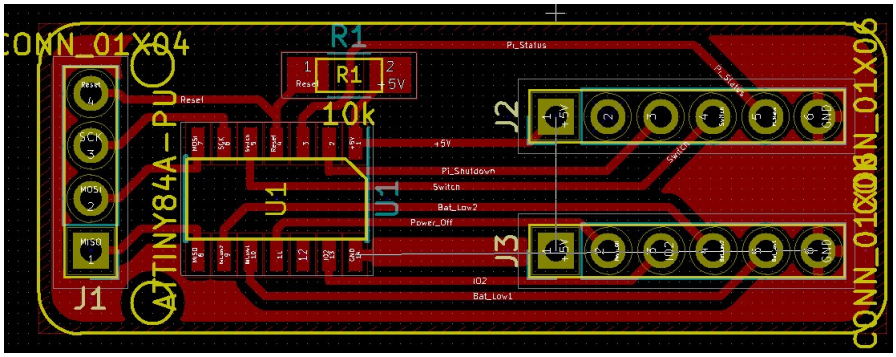


Abbildung 2: Platinen-Layout

Wie in Abbildung 3 dargestellt ist, wird für die Logik der Schaltung ein ATtiny (vgl. [5]) verwendet. Dieser wird permanent mit Spannung versorgt und ist damit der einzige Verbraucher im ausgeschaltetem Zustand.

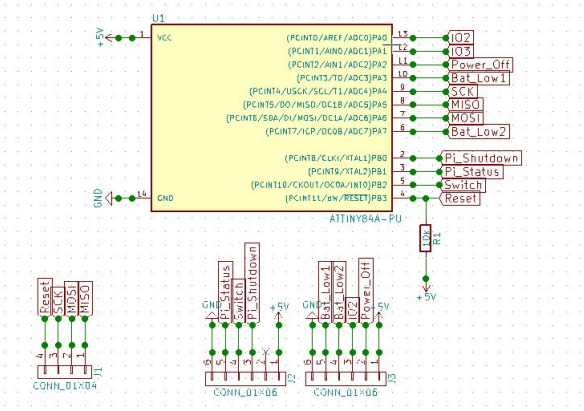


Abbildung 3: Selbstbestimmter Anschlussplan für den ATtiny

Mithilfe der Schaltung werden beim drücken des Tasters an dem Bedienteil die Powerbooster freigeschaltet. Diese Versorgen dann die weiteren Komponenten mit der Systemspannung. Wird das System ausgeschaltet deaktiviert der ATtiny die Spannungsversorgung sobald alle Prozesse heruntergefahren wurden.

## 3.3 IT-Hardware

Für den Aufbau werden folgende Komponenten verwendet:

- Raspberry Pi 3
- Raspberry Pi Zero
- ATtiny
- 1,5 Zoll OLED Display
- 4 Zoll LCD Display
- Knochenlautsprecher
- Sensorik

Der Raspberry Pi 3 ist ein Einplatinenrechner der im Bedienteil verbaut ist. Er dient als Hauptrechner und ist via WLAN mit dem Raspberry Pi Zero verbunden. Dieser ist an der Brille montiert und verarbeitet dort die Informationen der Sensorik. Weiterhin ist er mit dem OLED Display verbunden über welches er die visuellen Cues wieder gibt. Mithilfe des ATtiny wird wie oben erläutert die Power Management Schaltung realisiert. Das 4 Zoll LCD

Display ist im Bedienteil untergebracht und dient, wie oben erläutert, als Ein- und Ausgabeschnittstelle für den Bediener. Der Knochenlautsprecher dient zur Übertragung der auditiven Cues. Mit Hilfe der Sensorik wird die Bewegung der Brille im Raum erfasst, um die visuellen Cues entsprechend anzupassen.

## 3.4 Programmablauf

Dieser Abschnitt befasst sich zunächst mit dem allgemeinen Thema der Threads und im Anschluss mit dem Thema der Übertragung von Bildern. Abschließend wird der MJPG-Stream behandelt.

### 3.4.1 Konzept: Threads

Während der Laufzeit der Programme müssen verschiedene Aufgaben möglichst zeitgleich abgearbeitet werden. Um ein möglichst flüssiges Programm zu erhalten ist es sinnvoll, die verschiedenen Aufgaben in so genannten Threads aufzuteilen. Threads werden vom Betriebssystem verwaltet und zeitlich versetzt an die CPU vergeben. Bei Prozessoren mit nur einem Kern – wie dem Raspberry Pi Zero – werden die Aufgaben nur quasi-parallel ausgeführt (s. [6]). Trotzdem erscheint die Ausführung aller Aufgaben als parallel.

Das Umschalten zwischen den verschiedenen Aufgaben ermöglicht eine verbesserte Performance des Programmes. Der Raspberry Pi 3 hat vier Prozessorkerne. Das bedeutet, dass dieser verschiedene Aufgaben echt-parallel ausführen kann (s. [6]). Das Betriebssystem Rasbian, welches auf den Raspberry Pi 3 ausgeführt wird, sorgt für zeitliches Multiplexing zwischen den einzelnen Threads. Dies führt zur quasi-parallelen Ausführung der Programme. Hinzukommend kann das Betriebssystem die Threads auf verschiedene Prozessoren verteilen (s. [11]). Dies führt dann bei der entsprechenden Hardware zur echt-parallelen Ausführung. In der Abbildung 4 ist schematisch dargestellt, wie die Threads zeitlich ausgeführt werden. Das Betriebssystem sorgt selbst für das Multiplexing und das Aufteilen der Prozesse auf die verschiedenen Kerne. Der Programmierer hat darauf kaum Einfluss. Er kann lediglich die Prioritäten der Threads vergeben.

Die Brille verfügt über eine kleine Recheneinheit, dem Raspberry Pi Zero. Dieser hat lediglich einen Prozessorkern, welcher mit 800MHz Taktung arbeitet (vgl. [19]). Der Rechner kann also nur quasi-parallel arbeiten und sollte daher so wenig wie möglich tun, um eine möglichst schnelle Reaktionszeit zu ermöglichen.

Es werden zyklisch Neigungs- und Beschleunigungswerte aktualisiert. Die aktualisierten Werte werden per UDP-Protokoll (UDP – User Datagram

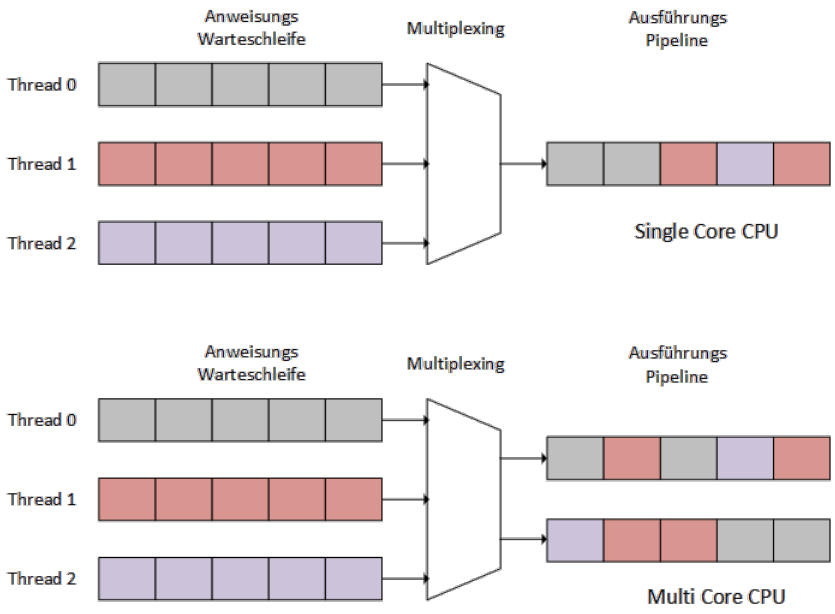


Abbildung 4: Multiplexing der Threads bei Single Core und Multi Core Architektur

Protocol, [26]) an die Hauptrecheneinheit gesendet. Zudem wird ein Video-Stream der Kamera an der Brille mittels HTTP-Server zur Verfügung gestellt, sodass die Hauptrecheneinheit die Daten von diesem Server abholen kann. Die Hauptrecheneinheit ermittelt, über den Video-Stream, die Bewegungen des Patienten. Des Weiteren werden kontinuierlich die neuen Bilder für die visuellen Cues erneuert, um ein fließendes Bild bei ca. 15 FPS (Bilder pro Sekunde) zu erzeugen. Dabei werden über einen UDP-Stream die neuen Bilder von der Hauptrecheneinheit abgeholt, dann konvertiert und auf das Display in der Brille projiziert. Zuletzt werden mit Zuhilfenahme der Sensordaten die Knochenlautsprecher rhythmisch angesprochen, damit der Patient eine hörbare Rückmeldung erhält.

Es werden vier verschiedene Threads ausgeführt:

- Neigungs- und Beschleunigungswerte aktualisieren
- Videostream zur Verfügung stellen
- Generierte Bilder auf dem Display anzeigen
- Knochenlautsprecher steuern

Die Hauptrecheneinheit erhält, wie nun klar ist, einen Videostream und einen Stream aus Neigungs- und Beschleunigungswerten. Diese Recheneinheit

wird durch den Raspberry Pi 3 realisiert. Dieser hat vier Kerne, welche mit 1,2GHz getaktet werden (vgl. [15]). Diese Recheneinheit ist demnach wesentlich leistungsfähiger als die Recheneinheit an der Brille. Hier geschieht nun die hauptsächliche Arbeit des Gesamtsystems. Aus den Daten werden nun Bilder generiert. Dies geschieht anhand des 3D-Modells eines Zebrastreifens. Dieses Modell wird an die Bewegungen und Position des Patienten angepasst. Wurde dieses Bild erstellt, wird dieses über einen UDP-Stream zurück an den Pi Zero in der Brille geschickt, wo dieses Modell dann angezeigt wird. Das Angezeigte Bild überlagert sich mit der Sicht des Patienten.

Hier werden vier verschiedene Threads ausgeführt:

- Das Abholen des Video-Streams der Brille
- der Stream der Neigungs- und Beschleunigungswerten
- die Generierung eines neuen Bildes anhand des 3D-Modells
- der Stream des generierten Bildes zurück zur Brille

### 3.4.2 Übertragung eines Bildes von der Hauptrecheneinheit an die Brille

Die Übertragung der Bilddaten von der Hauptrecheneinheit zur Brille wird in der Abbildung 5 komprimiert dargestellt.

Die Bildgenerierung auf der Hauptrecheneinheit erzeugt ein Datenarray im Framebuffer der Hauptrecheneinheit. Der Framebuffer stellt eine Schnittstelle der CPU mit der GPU dar (s [4]). Dieses ist ein Speicherbereich des Video-RAMs. Die GPU ist für Bildoperationen zuständig. Daher macht es Sinn alle Grafikoperationen von der GPU ausführen zu lassen, um Rechenzeit der CPU zu sparen.

Um die Bilddaten an die Brille zu schicken, werden also zunächst die Daten aus dem Framebuffer mittels OpenGL befehlen gelesen. Hier wird zeitgleich eine Konvertierung vorgenommen. Die Pixelanzahl und die Pixeltiefe werden direkt an das Display der Brille angepasst. Während dieser Operation wird dieser Speicherbereich gesperrt, sodass nicht zeitgleich gelesen und geschrieben werden kann. Nach dieser Operation steht im Arbeitsspeicher eine Kopie der Daten aus dem Framebuffer im richtigen Format zur Verfügung.

Die Kopie des Bildes aus dem Framebuffer wird nun über einen UDP-Stream an die Brille übertragen. UDP/IP (User Datagram Protocol) ist nicht verbindungsorientiert und ist mit weniger Overhead verbunden als TCP/IP (s. [24]). Falls eine Nachricht mal nicht fehlerfrei, oder gar nicht ankommt, kommt es nicht zum Abbruch des Streams. Es wird einfach weiter versucht

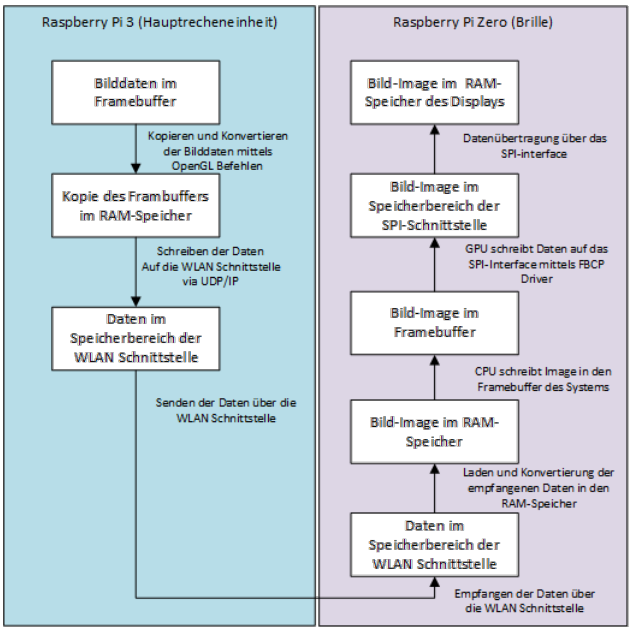


Abbildung 5: Übertragung der Bilddaten

neue Bilder zu versenden. Somit werden Fehler nicht unnötig behandelt. TCP/IP hat, im Gegensatz zu UDP/IP, Mechanismen implementiert, um eine fehlerfreie Datenübertragung zu garantieren (vgl. [21]. Diese Mechanismen würden bei einem kleinen Verbindungsabriss zu großen Verzögerungszeiten führen. Dieses Verhalten ist absolut nicht erwünscht.

Auf dem Raspberry Pi Zero, an der Brille, werden die Daten über UDP empfangen. Nach dem Empfang wird aus den Rohdaten ein Image erzeugt, welches dann in den Framebuffer des Pi Zeros geladen wird.

Ab dieser Stelle übernimmt die GPU ihre Arbeit. Die Übertragung aus dem Framebuffer an das Display wird komplett von der GPU übernommen. Dies ist nur mit einem entsprechenden Treiber möglich. Dieser Treiber nennt sich FBCP (Framebuffer Copy) (s. [17]). Er wird verwendet um das SPI-Interface, statt dem HDMI Interface zu verwenden. Dabei legt der Treiber eine Kopie des Framebuffers an, sodass der GPU suggeriert wird, sie würde auf das HDMI Interface schreiben. Nach der Kalkulation des Bildes wird die Kopie des Framebuffers verwendet um die Daten mit der GPU auf die SPI Schnittstelle zu schreiben, sodass das Bild von dem OLED-Display angezeigt wird. Würde die Bedienung der SPI-Schnittstelle mit der CPU erfolgen, würden lange Verzögerungszeiten entstehen, da hier viel Rechenarbeit nötig ist, die an andere Stelle gebraucht wird. Es ist daher zwingend notwendig, dass die GPU das Display anspricht. Da derzeit für das Rasbian-Betriebssystem

nur Treiber für HDMI und SPI präsent sind, können nur Displays mit diesen beiden Schnittstellen verwendet werden.

### 3.4.3 Kamera-Stream von der Brille zur Hauptrecheneinheit

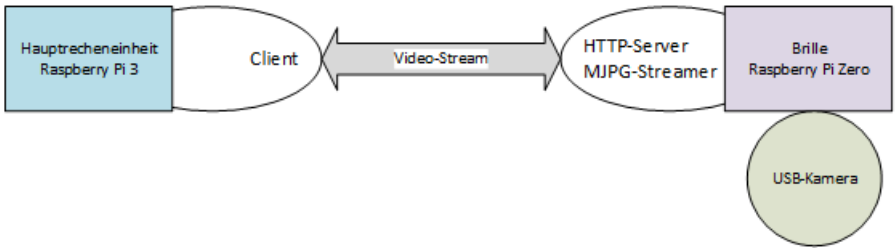


Abbildung 6: Server und Client des Video-Streams

Um der Hauptrecheneinheit einen Video-Stream aus Sicht des Patienten zur Verfügung zu stellen, wird ein Open-Source Programm verwendet. Es nennt sich MJPG-Streamer [22]. Dieses liest autark die Daten der USB-Kamera aus und stellt diese als Stream auf einem Webserver zur Verfügung (s. [16]). Die Hauptrecheneinheit muss daher lediglich auf diesen Server mit einem Passwort verbinden und die Bilder abholen. Auf der Hauptrecheneinheit wird von einem Thread dieser Stream gelesen. Mit diesen Daten können Bewegungsdaten des Patienten extrahiert werden, sodass das 3D-Modell mit der Bewegung des Patienten mitbewegt werden kann. Dieses Prinzip kann in der Abbildung 6 nachvollzogen werden.

Die Generierung der Visuellen Cues beinhaltet die Erzeugung und Darstellung eines Zebrastreifen-artigen Musters. Dieses soll auf der Brille dargestellt werden. Des Weiteren soll sich die Ausrichtung der Streifen an die Kopfposition anpassen.

Farbe sowie Größe (breite) der Streifen soll variabel gestaltet werden, sodass diese an die Bedürfnisse des Patienten angepasst werden können. Zur Generierung des Streifenmusters wurde die Programm Bibliothek OpenGL (Open Graphics Library) verwendet. Mit dieser wird ein 3D Modell des Zebrastreifens erzeugt. Die Erzeugung und Anzeige des Modells übernimmt hierbei die GPU des Raspberry Pi 3. Somit wird die CPU entlastet. Das Modell enthält eine einstellbare Anzahl von Slots für die einzelnen Streifen. Die Sicht auf das 3D Modell kann über OpenGL Funktionen verändert werden. Diese Veränderungen können beispielsweise eine Rotation oder Translation der Streifen sein. Die Daten der IMU werden dafür genutzt, um die Streifen an die entsprechende Kopfposition anzupassen. Bei einer Vorwärts-Bewegung wird überprüft, ob einer der Streifen den sichtbaren Bereich verlassen hat.

Sollte dieses der Fall sein, wird der entsprechende Streifen gelöscht und der Slot neu beschrieben. Hierdurch Entsteht für den Patienten der Eindruck, dass es sich um unendlich viele Streifen handelt. Die tatsächliche Anzahl an Slots ist jedoch sehr begrenzt wodurch das Programm nur relativ wenig Rechenleistung erfordert. In Abbildung 7 ist ein solches Streifenmuster dargestellt.

Abbildung 7: Projizierte Zebrastreifen

## 3.5 Power Management

Das Power Management wird vom ATtiny übernommen. Dieser ist direkt mit der Lithium Zelle verbunden und befindet sich normalerweise im so genannten *Deep Sleep*-Zustand, um den Akku nicht unnötig zu belasten. Dieser überwacht sowohl den Start/Stopp-Taster-Eingang als auch die digitalen Signale der Akkuwächter. Sobald an diesen eine Flankenänderung auftritt, geht der Mikrocontroller vom *Deep Sleep*-Zustand in den normalen Betriebsmodus über.

Wenn einer oder beide Akkus einen zu geringen Ladestand anzeigt und der Raspberry Pi 3 ist im Betriebsmodus so wird dieser sofort heruntergefahren und die Spannungswandler werden im Anschluss ausgeschaltet.

Wurde der Start/Stopp-Taster gedrückt, so wird zunächst für ca. zwei Sekunden gewartet und dann erneut geprüft, ob der Taster noch gedrückt ist. Dieses dient dazu sicher zu gehen, dass der Taster absichtlich gedrückt wurde. Sollte dieses der Fall sein gibt es zwei Möglichkeiten. Wenn der Raspberry Pi 3 nicht läuft wird dieser hochgefahren. Läuft der Raspberry Pi 3 bereits wird dieser heruntergefahren und die Spannungswandler werden abgeschaltet. Im Anschluss an diese Aktionen geht der Controller wieder in den *Deep Sleep*-Zustand.

## 3.6 Starter-Skripte und Netzwerk

Um das Starten der einzelnen Softwareteile zu vereinfachen, wurden entsprechende Starter-Skripte entwickelt. Nach dem Ausführen des Haupt-Starter-

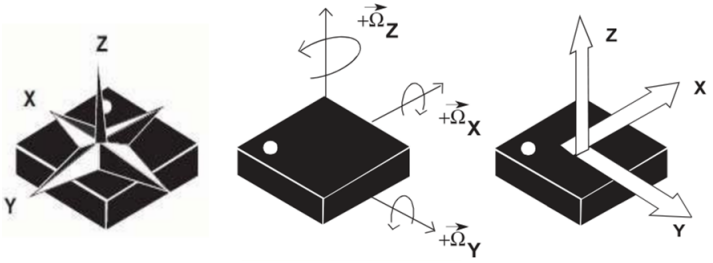


Abbildung 8: Prinzip der 9 DOF einer IMU. Quelle [9]

Skiptes auf dem Raspberry Pi 3 wird zunächst per *ssh* (secure socket layer) der MJPG Streamer und das Threading-Skript auf dem Raspberry Pi Zero ausgeführt. Im Anschluss daran wird auch das entsprechende Threading-Skript auf dem Raspberry Pi 3 gestartet und als letztes wird das Skript zum einlesen des MJPG-Streams gestartet.

Der Raspberry Pi 3 stellt ein eigenes WLAN Netzwerk zur Verfügung. Dieses dient zum einen der Kommunikation zwischen dem Raspberry Pi Zero und dem Raspberry Pi 3. Zum anderen wurde über dieses Netzwerk auch die gesamte Software Entwicklung durchgeführt. Zur Generierung des Netzwerkes wurden zwei Software verwendet. Erstens *hosapd* zur Erstellung des Accesspoints und *dnsmasq* als DNS Server, um den Netzwerkteilnehmern IP Adressen zuzuweisen.

# 4 Sensorik

Die inertiale Messeinheit (*inertial measurement unit, IMU*) dient dazu, die räumliche Orientierung der Brille zu bestimmen. Hierbei kommt eine Kombination von mehreren Sensoren zum Einsatz. Zum einen wird ein Sensor für translatorische Beschleunigungen (*Accelerometer, ACC*) und zum anderen ein Sensor für Winkelgeschwindigkeiten (*Gyroscope, GYR*) eingesetzt. Der ACC und der GYR sind in der Regel in einer Messeinheit zusammen verbaut.

In vielen inertialen Messeinheiten ist zusätzlich ein Kompasssensor zur Detektierung des Erdmagnetfeldes integriert. Diese Sensoren nehmen Werte in allen 3 Achsen (X,Y,Z) auf. Daraus folgt auch die häufig verwendete Bezeichnung einer 9 DOF (*degrees of freedom*) IMU. Diese Freiheitsgrade sind in Abbildung 8 dargestellt. Des Weiteren ist hier auch das Bezugskoordinatensystem des Sensors zu sehen.

## 4.1 IMU

Die in diesem Projekt verwendete Inertiale Messeinheit ist die LSM9DS1 der Firma ST Microelectronics (vgl. [10] bzw. [23]). Die verwendete IMU ist nur in einem SMD Format (24-lead package outline) verfügbar. Um die Entwicklung des Prototypen zu vereinfachen, wurde hier eine Entwicklungsplatine der Firma *Sparkfun* verwendet.

Die IMU besitzt eine $I^2C$-Schnittelle (s. [25]). Über dieses können sowohl die Beschleunigungsdaten ausgelesen als auch die Messeinheit konfiguriert werden. Des Weiteren ist die Messeinheit über die $I^2C$-Schnittelle mit dem Raspberry Pi Zero an der Brille verbunden.

### 4.1.1 Messung der Winkelgeschwindigkeit

Der Sensor zur Bestimmung der Winkelgeschwindigkeit stellt ein Ausgangssignal zur Verfügung, das proportional zur aktuellen Winkelgeschwindigkeit der jeweiligen Achse ist. Der Wert liegt in Form von einer vorzeichenbehafteten 16 Bit Ganzzahl vor. Der Messbereich des Sensors lässt sich auf +-245,+-500 oder +-2000 Grad/Sekunde einstellen.

Das bedeutet, dass bei einer Einstellung von 500 Grad/Sekunde für den Messbereich und einer konstanten Drehung des Sensors mit 500 Grad/Sekunde genau $2^{15} = 32768$ Werte dargestellt werden können. Des Weiteren bietet der Sensor zusätzlich die Möglichkeit, unterschiedliche Filterarten einzustellen.

### 4.1.2 Messung der translatorischen Beschleunigung

Der Sensor zur Bestimmung der translatorischen Beschleunigung liefert ein Ausgangssignal, das proportional zum einwirkenden Beschleunigungsvektor ist. Befindet sich der Beschleunigungsvektor in Ruhelage, so besteht der Beschleunigungsvektor ausschließlich aus dem Gravitationsvektor. Der Messbereich des Sensors lässt sich auf +/- 2,4,8,16 g einstellen. Der Ausgangswert besteht aus einer vorzeichenbehafteten 16 Bit Glanzzahl.

### 4.1.3 Bestimmung der Lagenwinkel

Die Bestimmung der Lagenwinkel basierend auf den Daten des GYR Sensors erfolgt indem die aktuelle Winkelgeschwindigkeit integriert wird. Zu Beginn der Messung muss der Sensor (GYR) kalibriert werden. Dieses dient dazu den systembedingten Drift zu minimieren. Dennoch gilt es zu beachten, dass die Lagenwinkel, die nur aus dem GYR ermittelt werden, einem gewissen Drift unterliegen.

Alternativ können die Lagenwinkel durch die Ausgangsdaten des Beschleunigungsmessers (ACC) über eine trigonometrische Berechnung ermittelt werden. Hierfür wird davon Ausgegangen, dass der Gravitationsvektor der einzige Beschleunigungsvektor ist. Ist diese Voraussetzung gegeben, können die Lagenwinkel bestimmt werden. Sie entsprechen dem jeweiligen Winkel zwischen dem Bezugssystem des Sensors und dem Gravitationsvektor. Die so ermittelten Lagenwinkel sind jedoch sehr empfindlich gegenüber Vibrationen beziehungsweise Beschleunigungen, die nicht von der Erdgravitation hervorgerufen werden.

Die reinen Lagenwinkel aus den GYR Daten unterliegen wie oben erwähnt einem gewissen Drift. Die Lagenwinkel aus dem Beschleunigungsmesser unterliegen starken Störungen durch Vibrationen.

Demnach sind beide Ansätze für sich genommen einzeln nicht nutzbar. Um verwertbare Lagenwinkel aus den IMU Daten zu errechnen, werden die Daten der Einzelsensoren – GYR und ACC – kombiniert. Konkret heißt das, dass der Gesamtlagenwinkel zu etwa 90% aus den Daten des GYR und zu ca. 10% aus den Daten des Beschleunigungsmessers berechnet wird.

## 4.2 Übertragung der IMU Daten

Die so errechneten Lagenwinkel werden wie die Bilddaten über eine UDP Verbindung an die Hauptrecheneinheit gesendet. Die Hauptrecheneinheit fragt die Daten hierbei zyklisch ab.

Der entscheidende Vorteil einer UDP Verbindung ist das die Kommunikation weiter läuft auch wenn zwischendurch einzelne Datenpakete verloren gehen. Der Verlust ist in diesem Fall kein Problem, da die Daten etwa 100 mal pro Sekunde angefordert werden.

# 5 Fazit

In dieser Arbeit wurde der aktuelle Stand des Parkinson-Brillen-Projektes an der *hochschule 21* dargestellt. Wie bereits erwähnt, geht es um die Implementierung eines *proof-of-concepts*, also einer Machbarkeitsstudie.

Im Rahmen des Projektes wurde eine Brille in Anlehnung an eine 3D-Kinobrille selbst konstruiert und mit additiven Druckverfahren erstellt. Schon bei der Konstruktion wurde darauf geachtet, dass es Möglichkeiten zur Anbringung von Kleinstrechner wie dem Raspberry PI Zero und auch der weiteren Sensorik (Kamera, IMU) gibt.

Die existierende Brille wurde bereits im Kontext des Future Days im Juni 2017 vorgestellt mit äußerst positivem Feedback.

Zur Zeit wird an einem zweiten Prototypen gearbeitet, der zum einem die Verlegung der Verkabelung in den Brillenrahmen berücksichtigen soll. Zum anderen soll eine Energieversorgung direkt in die Brille integriert werden. Und der wichtigste Punkt ist die Neugestaltung des Displays als ein – wie auch immer geartetes – transparentes Display.

# Literatur


[1] Google Glass - Tech specs. `https://support.google.com/glass/answer/3064128?hl=en&ref_topic=3063354`. zuletzt aufgerufen am 4.7.2017.

[2] SHIMA Tech Specs. `http://www.laforgeoptical.com/shima/product-info/tech-specs`. zuletzt aufgerufen am 5.7.2017.

[3] heise online permalink `https://heise.de/-3114955`, 2016. zuletzt aufgerufen am 4.7.2017.

[4] A. Buell. Framebuffer howto. `http://www.tldp.org/HOWTO/pdf/Framebuffer-HOWTO.pdf`, 2010. zuletzt aufgerufen am 5.7.2017.

[5] Atmel Corporation. Atmel 8-bit AVR Microcontroller with 2/4/8K Bytes In-System Programmable Flash. `http://www.atmel.com/images/atmel-2586-avr-8-bit-microcontroller-attiny25-attiny45-attiny85_datasheet.pdf`, 2013. zuletzt aufgerufen am 5.7.2017.

[6] D. Duschl. *Softwareentwicklung mit C++*. Springer Vieweg, 2014.

[7] EEMB Co., Ltd. Lithium-ion battery - data sheet. `https://www.ineltro.ch/media/downloads/SAAItem/45/45958/36e3e7f3-2049-4adb-a2a7-79c654d92915.pdf`, 2010. zuletzt aufgerufen am 5.7.2017.

[8] M.S. Ekker, S. Janssen, J. Nonnekes, B.R. Bloem, and N. M. de Vries. Neurorehabilitation for Parkinson's disease: Future perspectives for behavioural adaptation. In *Parkinsonism & Related Disorders – Proceedings of XXI World Congress on Parkinson's Disease and Related Disorders, December 6-9, 2015, Milan, Italy, XXI World Congress on Parkinson's Disease and Related Disorders*, volume 22, pages 73–77. Elsevier, 2016.

[9] SparkFun Electronics. lsm9ds1_axes. https://cdn.sparkfun.com/assets/learn_tutorials/3/7/3/lsm9ds1_axes.png. zuletzt aufgerufen am 5.7.2017.

[10] SparkFun Electronics. Sparkfun 9dof imu breakout - lsm9ds1. https://www.sparkfun.com/products/13284. zuletzt aufgerufen am 5.7.2017.

[11] H. Ernstl. *Grundkurs Informatik*. Springer Vieweg, 2016.

[12] Deutsche Parkinson Gesellschaft. Aktuelles - Von der Forschung in die Klinik: Die Deutsche Parkinson Gesellschaft mit neuer Präsenz im Web. zuletzt aufgerufen am 5.7.2017.

[13] EOS GmbH. Additive Fertigung, Laser-Sintern und industrieller 3D Druck - Vorteile und Funktionsprinzipien. https://www.eos.info/additive_fertigung/fuer_technologie_interessierte. zuletzt aufgerufen am 5.7.2017.

[14] D.W.E. Hein and P.A. Rauschnabel. Augmented Reality Smart Glasses and Knowledge Management: A Conceptual Framework for Enterprise Social Networks. In A. Rossmann, G. Stei, and M. Besch, editors, *Enterprise Social Networks*, pages 83–109. Springer, 2016.

[15] http://www.rs components.com. Raspberry pi 3 model b. http://docs-europe.electrocomponents.com/webdocs/14ba/0900766b814ba5fd.pdf. zuletzt aufgerufen am 5.7.2017.

[16] L. Jackson. github – mjpg-streamer. Available:https://github.com/jacksonliam/mjpg-streamer. zuletzt aufgerufen am 5.7.2017.

[17] M. Kohns. github – framebuffer. https://github.com/notro/fbtft/wiki/Framebuffer-use, 2016. zuletzt aufgerufen am 4.7.2017.

[18] Adafruit Industries lady ada. Adafruit powerboost 1000c. https://cdn-learn.adafruit.com/downloads/pdf/adafruit-powerboost-1000c-load-share-usb-charge-boost.pdf. zuletzt aufgerufen am 5.7.2017.

[19] Adafruit Industries lady ada. Introducing the raspberry pi zero. https://cdn-learn.adafruit.com/downloads/pdf/introducing-the-raspberry-pi-zero.pdf. zuletzt aufgerufen am 5.7.2017.

[20] P. Messner, F. Antwerpes, and N. Ostendorf. Freezing of gait, 2014. zuletzt aufgerufen am 4.7.2017.

[21] G.B.W Schnell. *Bussysteme in der Automatisierungs- und Prozesstechnik*. Vieweg & Sohn, 2006.

[22] H. Schwietering. MJPG-Streamer. `https://wiki.ubuntuusers.de/MJPG-Streamer/`, 2017. zuletzt aufgerufen am 4.7.2017.

[23] STMicroelectronics. Lsm9ds1 - data sheet. `http://www.st.com/content/ccc/resource/technical/document/datasheet/1e/3f/2a/d6/25/eb/48/46/DM00103319.pdf/files/DM00103319.pdf/jcr:content/translations/en.DM00103319.pdf`. zuletzt aufgerufen am 5.7.2017.

[24] G.D.Z. Wellenreuter. *Automatisieren mit SPS - Theorie und Praxis*. Vieweg + Teubner, 2008.

[25] Wikipedia. I2C. `https://de.wikipedia.org/wiki/I%C2%B2C`. zuletzt aufgerufen am 5.7.2017.

[26] Wikipedia. User datagram protocol. `https://de.wikipedia.org/wiki/User_Datagram_Protocol`, 2017. zuletzt aufgerufen am 4.7.2017.

[27] Y. Zhao, J. Nonnekes, E.J.M. Storcken, S. Janssen, E.E.H. van Wegen, B.R. Bloem, L.D.A. Dorresteijn, J.P.P. van Vugt, T. Heida, and R.J.A. van Wezel. Feasibility of external rythmic cueing with the Google Glass for improving gait in people with Parkinson's desease. *J. Neurol.*, 263:1156–1165, 2016.