

# Höhere Technische Bundeslehranstalt Kaindorf

## Abteilung für Mechatronik



Abgabedatum: 18. März 2022

Erstellt im Schuljahr 2021/22 von:			Betreut von:
Georg Kaufmann	Mechatronik	5AHME	Dipl. Ing. Thomas Jerman
Paul Resch	Mechatronik	5AHME	
Christoph Sebernegg	Mechatronik	5AHME	Dipl. Ing. Werner Harnisch
Kilian Wade	Mechatronik	5AHME	Dipl. Ing. Dr. Gerhard Pretterhofer



## Eidesstattliche Erklärung

Hiermit wird bestätigt,

1. dass das in dieser Diplomarbeit beschriebene Projekt von den Gruppenmitgliedern selbst durchgeführt wurde - sowohl in organisatorischer als auch in technischer Hinsicht,
2. dass jedes Teammitglied nur jene Arbeitsbereiche als seine bezeichnet und beschreibt, wofür es verantwortlich war/ist, bzw. welche es selbst umgesetzt hat;
3. dass Beweisführungen jeglicher Art hinsichtlich der Vorgehensweise in der Projektumsetzung durch korrekte Zitierung erkenntlich sind,
4. dass fremdes Bildmaterial mit Quellenangabe versehen ist,
5. dass die schriftliche Diplomarbeit von den Teammitgliedern selbst verfasst wurde und die Dokumentation und Erörterung der spezifischen Arbeitspakete von jedem einzelnen Projektmitglied selbst erfolgte.

Es wird durchgehend die männliche Form für Kollektivbezeichnungen verwendet.

Arnfels, am 10.03.2022

f. Kaufmann  
Georg Kaufmann

Arnfels, am 10.03.2022

Paul Resch  
Paul Resch

Arnfels, am 10.03.2022

Christoph Sebernegg  
Christoph Sebernegg

Arnfels, am 10.03.2022

Kilian Wade  
Kilian Wade



## Danksagung

Das Projektteam möchte sich bei all denjenigen bedanken, die bei der Erstellung und Entwicklung der Diplomarbeit unterstützt und motiviert haben.

Besonderer Dank gilt dabei unseren Betreuern Dipl.-Ing. Thomas Jerman, Dipl.-Ing. Werner Harnisch und Dipl.-Ing. Dr. Gerhard Pretterhofer die maßgeblich daran mitgewirkt haben, dass diese Arbeit in dieser Form vorliegt, sei es in technischer oder formaler Hinsicht.

## Abstract

In this diploma project, a concept for the conversion of a commercially available walking aid into an iWalk was developed. The aim is to record health data as well as detecting potentially hazardous falls and ensuring quick help for elderly people. Steps are also counted and a stress analysis is performed. All data can be accessed via an app.

The project was divided into the following four divisions: Mechanics, programming and sensor and electrical engineering.

## Zusammenfassung

In diesem Diplomprojekt wird theoretisch erarbeitet, wie eine handelsübliche Gehhilfe zum iWalk umfunktioniert werden kann. Es soll dazu dienen Gesundheitsdaten zu erfassen, Stürze zu erkennen und älteren Personen dadurch schnellere Hilfe gewährleisten zu können. Weiters sollen Schritte gezählt und eine Belastungsanalyse durchgeführt werden können. Diese Daten sollen per App ausgegeben werden können.

Das Projekt wurde in folgenden vier Teilbereichen erarbeitet. Mechanik, Informatik und Sensorik bzw. Elektrotechnik.



## Über dieses Dokument

Diese Arbeit wurde in L<sup>A</sup>T<sub>E</sub>X verfasst. Diese Art der Dokumentation bietet gegenüber den normalen Textverarbeitungen gewisse Vorteile hinsichtlich der Formatierung und des Einbindens von Grafiken. Auch Formeln können sehr einfach und effizient angegeben werden. Die Rohfassung des Dokuments befindet sich auf dem Arnfelser Gitweb Server der HTBLA Kaindorf Abteilung Mechatronik.

## Projektteam

### Georg Kaufmann

**Aufgabenbereich:**

Akkusystem und Kraftanalyse

**Betreuer:**

Dipl.-Ing. Thomas Jerman

### Paul Resch

**Aufgabenbereich:**

Sturzerkennung und Schrittzähler

**Betreuer:**

Dipl.-Ing. Thomas Jerman

---

## Projektteam

### Christoph Sebernegg

**Aufgabenbereich:**

App-Programmierung

**Betreuer:**

Dipl.-Ing. Werner Harnisch

### Kilian Wade

**Aufgabenbereich:**

Konstruktion und Berechnung

**Betreuer:**

Dipl.-Ing. Dr. Gerhard Pretterhofer



# Inhaltsverzeichnis

<b>Einleitung</b>	<b>1</b>
Aufgabenstellung . . . . .	2
Übersicht der Gehhilfe . . . . .	3
Übersicht der Aufgabenbereiche . . . . .	4
Übersicht der Zeitpläne . . . . .	5
<b>1 Teil A - Mechanik</b>	<b>7</b>
1.1 Aufgabenstellung . . . . .	8
1.2 Schemaskizze . . . . .	8
1.2.1 Gesamte Krücke . . . . .	9
1.2.2 Krückenkopf . . . . .	10
1.2.3 Rohr Eins . . . . .	11
1.2.4 Rohr Zwei . . . . .	12
1.2.5 Gumminoppe . . . . .	13
1.3 Inventorzeichnungen . . . . .	14
1.3.1 Was ist Inventor? . . . . .	14
1.3.2 Was kann Inventor? . . . . .	14
1.3.3 Inventor Konstruktion 1.0 . . . . .	15
1.3.3.1 Gesamte Krücke . . . . .	15
1.3.3.2 Krückenkopf . . . . .	16
1.3.3.3 Rohr Eins . . . . .	17
1.3.3.4 Rohr Zwei . . . . .	18
1.3.3.5 Gumminoppe . . . . .	19

---

1.3.3.6	Rohrverbinder . . . . .	20
1.3.4	Inventor Konstruktion 2.0 . . . . .	21
1.3.4.1	Gesamte Krücke mit Sensoren . . . . .	21
1.3.4.2	Gesamte Krücke 2.0 . . . . .	22
1.3.4.3	Beschleunigungssensor und Microcontroller . . . . .	22
1.3.4.4	Problem Microcontroller . . . . .	22
1.3.4.5	Lösung Eins . . . . .	22
1.3.4.6	Lösung Zwei . . . . .	22
1.3.4.7	Microcontroller ESP32 . . . . .	23
1.3.4.8	Halterung des Microcontroller . . . . .	23
1.3.4.9	Rohr Eins Mit Akku und Akkuhalterung . . . . .	24
1.3.4.10	Rohr Zwei mit Berührungssensor . . . . .	25
1.3.4.11	Kraftsensor . . . . .	25
1.3.4.12	Rohr Zwei mit Noppenhalterung und Berührungssensor . . . . .	26
1.3.4.13	Stoppel für Rohr Zwei . . . . .	26
1.3.4.14	Gumminoppe 2.0 . . . . .	27
1.3.4.15	Rohr Zwei und Gumminoppe . . . . .	28
1.3.4.16	Ansicht der Bauteile von Innen . . . . .	29
1.4	Versuch mit Waage . . . . .	30
1.5	Kraftanalyse . . . . .	31
1.5.1	Inventor . . . . .	31
1.5.1.1	Krückenkopf Belastung Inventor . . . . .	31
1.5.1.2	Rohr Zwei Belastung . . . . .	32
1.5.1.3	Zehn Bohrungen an Rohr Zwei . . . . .	32
1.5.1.4	Bohrung Verbindungsstück Rohr Zwei . . . . .	33
1.5.1.5	Rohr Zwei Belastung Unterseite . . . . .	34
1.5.1.6	Belastung Gumminoppe an Rohr Zwei . . . . .	35
1.5.1.7	Belastung Gumminoppe zum Boden . . . . .	36
1.5.2	Mechanische Berechnung . . . . .	37
1.5.2.1	Biegung an Krückenkopf . . . . .	37

---

1.5.2.2	Berechnung der Biegesicherheit . . . . .	38
1.5.2.3	Knickung an Rohr Zwei . . . . .	40
1.5.2.4	Berechnung des Schlankheitsgrades Lambda . . . . .	41
1.5.2.5	Berechnung der Knickungssicherheit . . . . .	43
1.5.2.6	Berechnung der Abscherung des Rohrverbinder . . . . .	44
1.5.2.7	Berechnung der Flächenpressung des Rohrverbinder . . . . .	45
1.6	Materialauswahl . . . . .	46
1.6.1	Krückenkopf . . . . .	46
1.6.1.1	Materialvergleich . . . . .	47
1.6.1.2	Fazit . . . . .	47
1.6.2	Rohr Eins und Zwei . . . . .	48
1.6.2.1	Materialvergleich . . . . .	48
1.6.2.2	Fazit . . . . .	48
1.6.3	Gumminoppe . . . . .	49
1.6.3.1	Materialvergleich . . . . .	49
1.6.3.2	Fazit . . . . .	49
1.7	Fertigung des iWalk . . . . .	50
1.7.1	Fertigung der Akkus sowie Microcontrollerhalterung . . . . .	50
1.7.2	Fertigung des Rohrstoppel . . . . .	50
<b>2</b>	<b>Teil B - Informatik</b>	<b>51</b>
2.1	Anforderungen . . . . .	52
2.1.1	Entwicklungsumgebung . . . . .	52
2.1.1.1	Auswahl der verwendeten IDE . . . . .	52
2.1.2	Android . . . . .	53
2.2	Kommunikation . . . . .	54
2.2.1	Theoretischer Hintergrund . . . . .	54
2.2.1.1	Kommunikation über Wlan . . . . .	54
2.2.1.2	Kommunikation über Bluetooth . . . . .	54
2.2.2	AsyncTask . . . . .	54

2.2.3	Bluetoothverbindung . . . . .	57
2.3	Design der Android App . . . . .	58
2.3.1	Erste Designvorschläge . . . . .	58
2.3.1.1	Ladebildschirm . . . . .	58
2.3.1.2	Startbildschirm . . . . .	59
2.3.1.3	Schritte Activity . . . . .	60
2.3.1.4	Stürze Activity . . . . .	61
2.3.1.5	Belastungsanalyse Activity . . . . .	62
2.3.1.6	Notfallkontakt . . . . .	63
2.3.2	Finales Design der App . . . . .	64
2.3.2.1	Ladebildschirm . . . . .	64
2.3.2.2	Startbildschirm . . . . .	65
2.3.2.3	Schritte Activity . . . . .	66
2.3.2.4	Stürze Activity . . . . .	67
2.3.2.5	Belastungsanalyse Activity . . . . .	68
2.3.3	Sprachen . . . . .	69
2.3.4	Farben . . . . .	71
2.3.5	Logo . . . . .	71
2.3.6	Icon . . . . .	71
2.3.7	Options Menu . . . . .	71
2.3.8	Linearlayout mit Radius . . . . .	73
2.3.9	AndroidManifest.xml-Datei . . . . .	73
2.4	Ladebildschirm . . . . .	74
2.5	Diagramme . . . . .	75
2.5.1	Einfügen der Bibliotheken . . . . .	75
2.5.2	Balkendiagramm . . . . .	76
2.5.3	Liniediagramm . . . . .	76
2.6	Aufruf des Dialogfensters . . . . .	77
2.6.1	Positiv Button . . . . .	78
2.6.2	Negativ Button . . . . .	78

---

2.6.3	Löschen der Kontakte . . . . .	79
2.7	Entwicklungsprozess, Selbstkritische Analyse, Resümee . . . . .	79
2.7.1	Entwicklungsprozess . . . . .	79
2.7.2	Selbstkritische Analyse . . . . .	79
2.7.3	Resümee . . . . .	80
<b>3</b>	<b>Teil C - Sensorik</b>	<b>81</b>
3.1	Aufgabenstellung . . . . .	82
3.2	Funktion . . . . .	82
3.2.1	Schrittzähler . . . . .	82
3.2.1.1	Theoretischer Hintergrund . . . . .	82
3.2.2	Sturzerkennung . . . . .	82
3.2.2.1	Theoretischer Hintergrund . . . . .	83
3.2.3	MEMS . . . . .	83
3.2.3.1	Eigenschaften . . . . .	83
3.2.3.2	Einsatzgebiete . . . . .	84
3.3	Beschleunigungssensor . . . . .	85
3.3.1	Funktionsweise . . . . .	85
3.3.2	Arten . . . . .	85
3.3.2.1	Piezoresistiver Beschleunigungssensor . . . . .	86
3.3.2.2	Piezoelektrischer Beschleunigungssensor . . . . .	86
3.3.2.3	Kapazitiver Beschleunigungssensor . . . . .	86
3.3.3	MEMS Beschleunigungssensor . . . . .	87
3.3.3.1	Kondensator . . . . .	87
3.3.3.2	Aufbau des MEMS Beschleunigungssensors . . . . .	88
3.3.3.3	Funktion . . . . .	89
3.3.4	Auswahl des Beschleunigungssensors . . . . .	89
3.3.4.1	Fazit . . . . .	90
3.3.5	GY-521 MPU-6050 . . . . .	91
3.3.5.1	Spezifikationen . . . . .	92

---

3.3.5.2	Achsen . . . . .	92
3.3.5.3	Pins . . . . .	93
3.3.5.4	Ansteuerung Grundlagen . . . . .	93
3.3.5.5	Fazit . . . . .	93
3.4	Auswahl der Entwicklungsumgebung . . . . .	94
3.4.1	Programmiersprachen . . . . .	94
3.4.1.1	Assembler . . . . .	95
3.4.1.2	C . . . . .	95
3.4.2	Arduino IDE . . . . .	96
3.4.3	Fazit . . . . .	96
3.4.4	Einrichten der Entwicklungsumgebung . . . . .	96
3.5	Ansteuern des Sensors . . . . .	97
3.5.1	Pins . . . . .	101
3.5.2	Daten auswerten . . . . .	102
3.6	Hauptprogramm . . . . .	104
3.6.1	Sensorposition . . . . .	104
3.6.2	Überlegungen zum Schrittzähler . . . . .	105
3.6.2.1	Messdaten . . . . .	105
3.6.2.2	Definition Wertebereich . . . . .	105
3.6.3	Programmcode Schrittzähler . . . . .	106
3.6.4	Überlegung zur Sturzerkennung . . . . .	108
3.6.4.1	Messdaten . . . . .	109
3.6.4.2	Analyse . . . . .	109
3.6.5	Programmcode Sturzerkennung . . . . .	109
<b>4</b>	<b>Teil D - Sensorik</b>	<b>113</b>
4.1	Aufgabenstellung . . . . .	114
4.2	Projektplanung . . . . .	114
4.2.1	Projektdefinition . . . . .	114
4.2.2	Projektorganisation . . . . .	115

4.2.3	Projektphasen . . . . .	116
4.3	Kraftmessung . . . . .	117
4.3.1	Anforderungen an den Kraftsensor . . . . .	117
4.3.2	Arten von Kraftsensoren . . . . .	118
4.3.2.1	Piezo-Sensor . . . . .	118
4.3.2.2	Dehnmessstreifen . . . . .	119
4.3.2.3	Fazit . . . . .	120
4.3.3	Gewählter Sensor . . . . .	121
4.3.4	DMS - Bückschaltung . . . . .	121
4.3.4.1	Viertelbrücke . . . . .	122
4.3.4.2	Halbbrücke . . . . .	122
4.3.4.3	Vollbrücke . . . . .	122
4.3.5	Code zur Umsetzung . . . . .	123
4.3.6	Anschluss des Sensors . . . . .	123
4.4	Kommunikation mit der App . . . . .	124
4.4.1	Anforderungen an die Kommunikation . . . . .	124
4.4.2	Arten der Kommunikation . . . . .	124
4.4.2.1	NFC - Near Field Communication . . . . .	124
4.4.2.2	WLAN - Wireless Local Area Network . . . . .	124
4.4.2.3	Bluetooth . . . . .	125
4.4.2.4	Fazit . . . . .	126
4.4.3	Arten von Bluetoothmodulen . . . . .	126
4.4.3.1	nRF52840 . . . . .	126
4.4.3.2	ESP32 . . . . .	126
4.4.3.3	Fazit . . . . .	126
4.4.4	Code zur Kommunikation . . . . .	127
4.4.5	App zum Testen . . . . .	128
4.5	Microcontroller . . . . .	129
4.5.1	Was ist ein Microcontroller? . . . . .	129
4.5.1.1	Arten . . . . .	129

---

4.5.1.2	Architekturen . . . . .	130
4.5.1.3	Funktionsweiße . . . . .	130
4.5.2	Anforderungen an den Microcontroller . . . . .	130
4.5.3	Vergleich ATmega328P mit ESP32 . . . . .	132
4.5.4	Fazit . . . . .	132
4.6	Akkusystem . . . . .	132
4.6.1	Akku Allgemein . . . . .	132
4.6.2	Anforderungen an den Akku . . . . .	134
4.6.3	Auswahl des Akkus . . . . .	134
4.6.4	Ladeverfahren . . . . .	134
4.6.4.1	Energy Harvesting . . . . .	134
4.6.4.2	Fazit . . . . .	135
4.7	Speicherung der Daten . . . . .	135
4.7.1	Anforderungen an die Speicherung . . . . .	135
4.7.2	Ringspeicher . . . . .	136
<b>5</b>	<b>Anhang</b>	<b>137</b>
5.1	Gesamter Programmcode . . . . .	138
5.2	Anschluss der Sensoren . . . . .	144
5.3	Teil A - Mechanik . . . . .	145
<b>6</b>	<b>Zeitaufzeichnung</b>	<b>149</b>
6.1	Kilian Wade - Teil A . . . . .	150
6.2	Christoph Sebernegg - Teil B . . . . .	150
6.3	Paul Resch - Teil C . . . . .	151
6.4	Georg Kaufmann - Teil D . . . . .	151
<b>7</b>	<b>Abbildungsverzeichnis</b>	<b>153</b>
<b>8</b>	<b>Tabellenverzeichnis</b>	<b>157</b>
<b>9</b>	<b>Literaturverzeichnis</b>	<b>159</b>

# **Einleitung**

## Aufgabenstellung

Gehstützen bieten selbstbestimmte Mobilität und verbessern das Leben vieler Menschen, doch nur die richtige Verwendung bietet eine Verbesserung der Beweglichkeit. Dazu werden in einer gewöhnlichen Gehhilfe ein Schrittzähler, eine Kraftanalyse für therapeutische Behandlung sowie eine Sturzerkennung integriert. Der Datenabruf erfolgt über eine Smartphone-App. Weiters ist ein Akku-System zu entwickeln.

Das Ziel besteht darin, Stürze zu erkennen und Betroffenen schnelle Hilfe mittels einer intelligenten Gehhilfe bereitzustellen. Weiters soll der iWalk ein Monitoring der Benutzerbewegung durch eine Schritt- und Kraftanalyse ermöglichen. Die zu erfassenden Daten sollen in einer Smartphone-App abrufbar sein. Die benötigte Elektronik soll in einer herkömmlichen Gehhilfe untergebracht werden können.

Das geplante Ergebnis ist eine funktionsfähige App, eine mit geeigneten Materialien konstruierte Gehhilfe, ein Teilaufbau und die Sicherstellung folgender Funktionen: Schrittzähler, Kraftmesser, Sturzerkennung und Akku-System.

# Übersicht der Gehhilfe

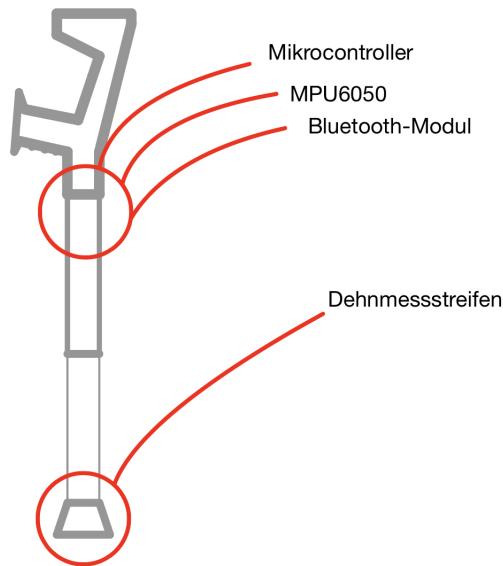


Abbildung 0.1: Übersicht Gehhilfe

Bezeichnung	Beschreibung
<i>Microcontroller</i>	Reagiert auf externe Signale
<i>MPU6050</i>	3-Achsen Beschleunigungssensor und 3-Achsen Gyroskop
<i>Bluetooth-Modul</i>	Kommunikation mit der App
<i>DMS</i>	Kraftsensor

# **Übersicht der Aufgabenbereiche**

## **Mechanik**

Die zu untersuchenden Aufgabenstellungen von Kilian Wade sind die Konstruktion und ein Teilaufbau. Für den Bau der Gehhilfe sollen mehrere Materialen untersucht werden. Die Betreuung erfolgt durch Prof. Preterhofer.

## **Informatik**

Die zu untersuchende Thematik von Christoph Sebernegg ist die Entwicklung einer Android-App für die Gehhilfe. Die Betreuung erfolgt durch Prof. Harnisch.

## **Elektronik - Sensorik**

Die zu untersuchende Thematik von Georg Kaufmann ist die Entwicklung des Akkusystems und der Kraftanalyse. Die Betreuung erfolgt durch Prof. Jerman.

Die zu untersuchenden Aufgabenstellungen von Paul Resch sind die Sturzerkennung und der Schrittzähler. Die Betreuung erfolgt durch Prof. Jerman.

# Übersicht der Zeitpläne

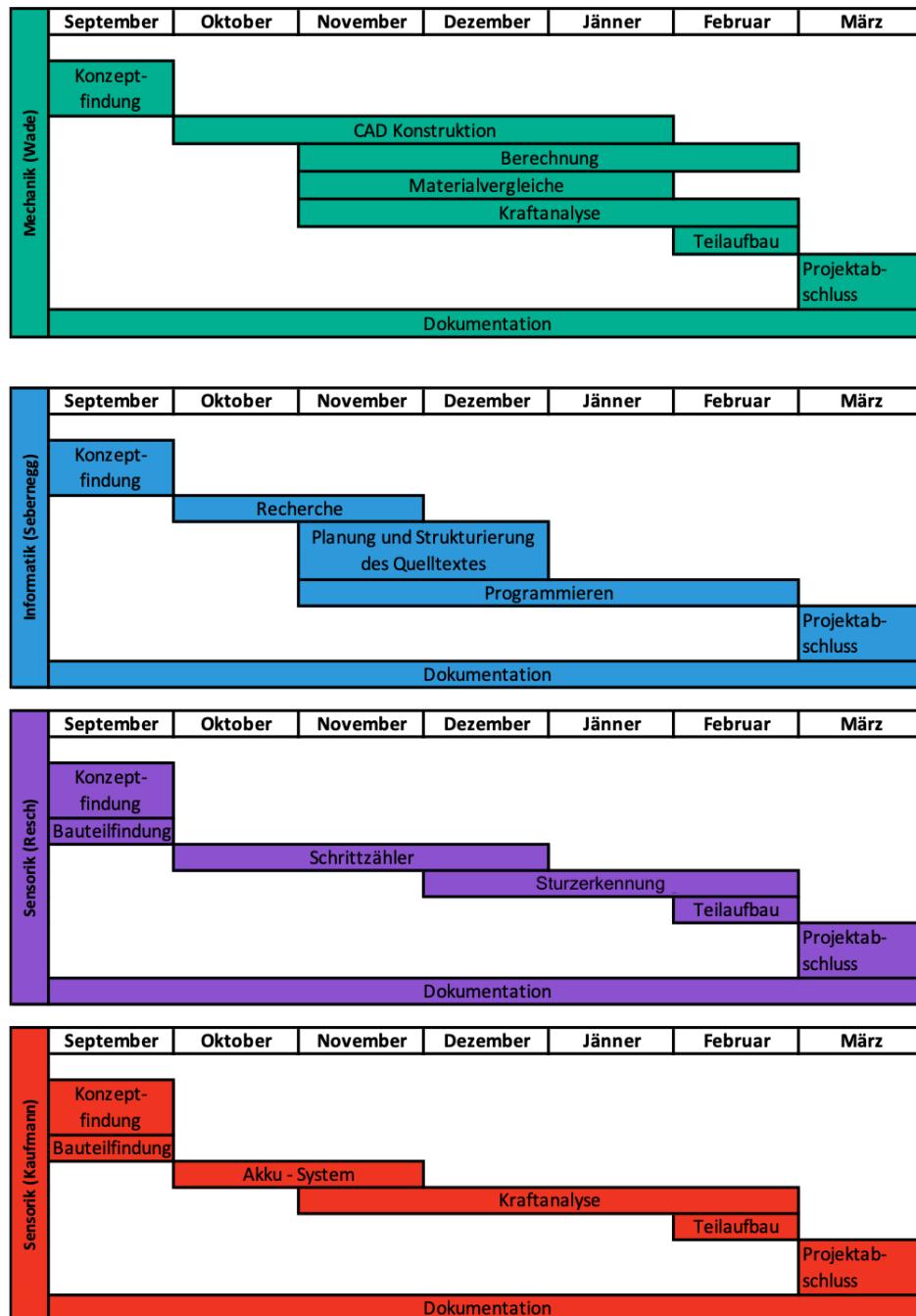


Abbildung 0.2: Übersicht Zeitpläne



# 1 Teil A - Mechanik

## 1.1 Aufgabenstellung

Die zu untersuchende Aufgabenstellung ist die Konstruktion des iWalks. Für den Bau der Gehhilfe sollen mehrere Materialen untersucht werden. Die Betreuung erfolgt durch Prof. Pretterhofer.

## 1.2 Schemaskizze

Um einen groben Überblick zu schaffen aus welchen Teilen eine Krücke besteht wurde als Erstes eine Schemaskizze gezeichnet. Dafür wurde eine Krücke von der Firma *Rebotec*<sup>1</sup> verwendet und vollständig ausseinernder gebaut. Die Krücke besteht im wesentlichen aus dem Krückenkopf, Rohr Eins sowie Rohr Zwei und der Gumminoppe. Diese Teile wurden weiters grob vermessen und anschließend mit Hilfe einzelner Schemaskizzen dargestellt.

---

<sup>1</sup>vgl. Rebotec, (2022)

### 1.2.1 Gesamte Krücke

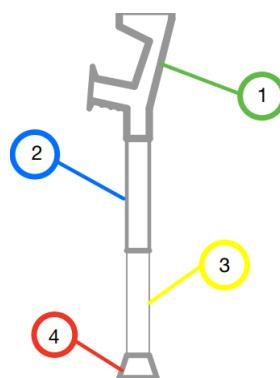


Abbildung 1.1: Die Schemaskizze der gesamten Krücke mit Einteilung in Teile Eins bis Vier

In der Abbildung ist die Schemaskizze der gesamten Krücke zu sehen. Die Einzelteile sind farblich und mit den Nummern Eins-Vier gekennzeichnet.

#### Teilbeschreibung Eins bis Vier:

- Beim Teil Eins, welcher Grün farblich hinterlegt ist, handelt es sich um den Krückenkopf. Dieser ist der komplexeste Teil der Krücke. Er wird aus Kunststoff gefertigt.
- Beim Teil Zwei, welcher Blau farblich hinterlegt ist, handelt es sich um Rohr Eins. Dieses Rohr ist kraftschlüssig mit dem Krückenkopf verbunden. Das Rohr Eins wird aus Metall gefertigt.
- Beim Teil Drei, welcher Gelb farblich hinterlegt ist, handelt es sich um Rohr Zwei. Rohr Zwei ist im Durchmesser dünner als Rohr Eins und kann daher ins erste Rohr gesteckt werden. Rohr Eins ist durch eine Steckverbindung mit Rohr Zwei verbunden. Durch Zehn Löcher in Rohr Zwei kann die Krücke in ihrer Länge auf die benützende Person angepasst werden. Diese Steckverbindung wird erst im Inventorteil konstruiert. Rohr Zwei ist aus demselben Metall.
- Bei Teil Vier, welcher Rot farblich hinterlegt ist, handelt es sich um die Gumminoppe. Diese ist kraftschlüssig auf Rohr Zwei aufgesteckt und soll die Haftung am Boden gewährleisten. Weiters wird über diese Noppe die Kraft, welche auf die Krücke ausgeübt wird, gemessen.

## 1.2.2 Krückenkopf

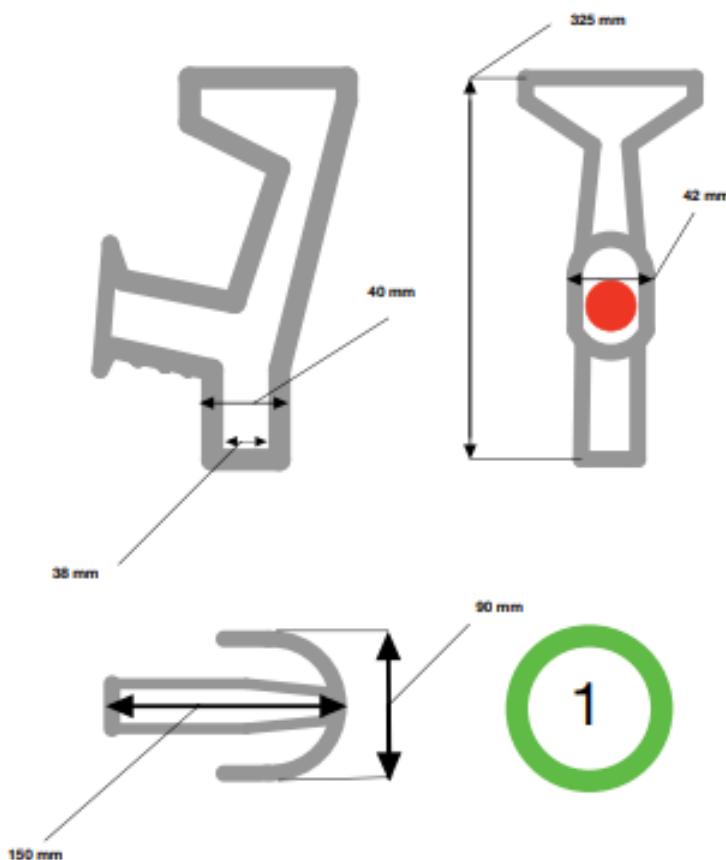


Abbildung 1.2: Schemaskizze Krückenkopf Teil Eins

In der obigen Abbildung wird die Schemaskizze vom Krückenkopf dargestellt. Der Krückenkopf ist der komplexeste Teil der Krücke, da dieser der direkte Kontakt mit der Person ist und ergonomisch geformt sein soll. Die wichtigsten Maße wurden mit den Hauptmessutensilien, einem Messschieber und einem Maßband, abgenommen. Als Messobjekt diente eine handelsüblichen Krücke der Firma "Rebotec". Die Schemaskizzen wurden mit Hilfe einer 2-Dimensionalen Zeichenumgebung gefertigt.

### 1.2.3 Rohr Eins

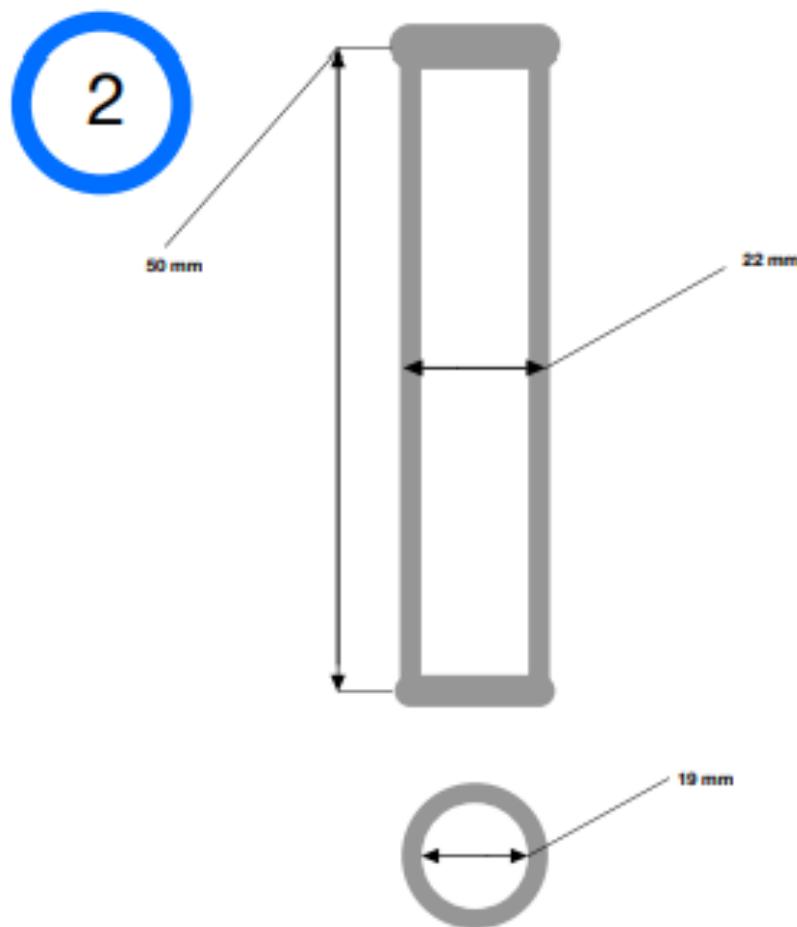


Abbildung 1.3: Schemaskizze Rohr Eins Teil Zwei

In der obigen Abbildung ist die Schemaskizze des oberen Rohrs ersichtlich. Bei diesem Bauteil wurde die Gesamtlänge ermittelt sowie der Innen- und Außendurchmesser. Diese Maße sind in der Abbildung ersichtlich. Hierbei handelt es sich um ein handelsübliches metallenes Rohr. Dieses Rohr ist mit einer Bohrung versehen. Die Bohrung mit einem Durchmesser von 8 mm befindet sich 320 mm vom unteren Rand. Die Funktion dieser Bohrung ist Teil der Höhenverstellbarkeit zusammen mit dem unteren Rohr.

### 1.2.4 Rohr Zwei

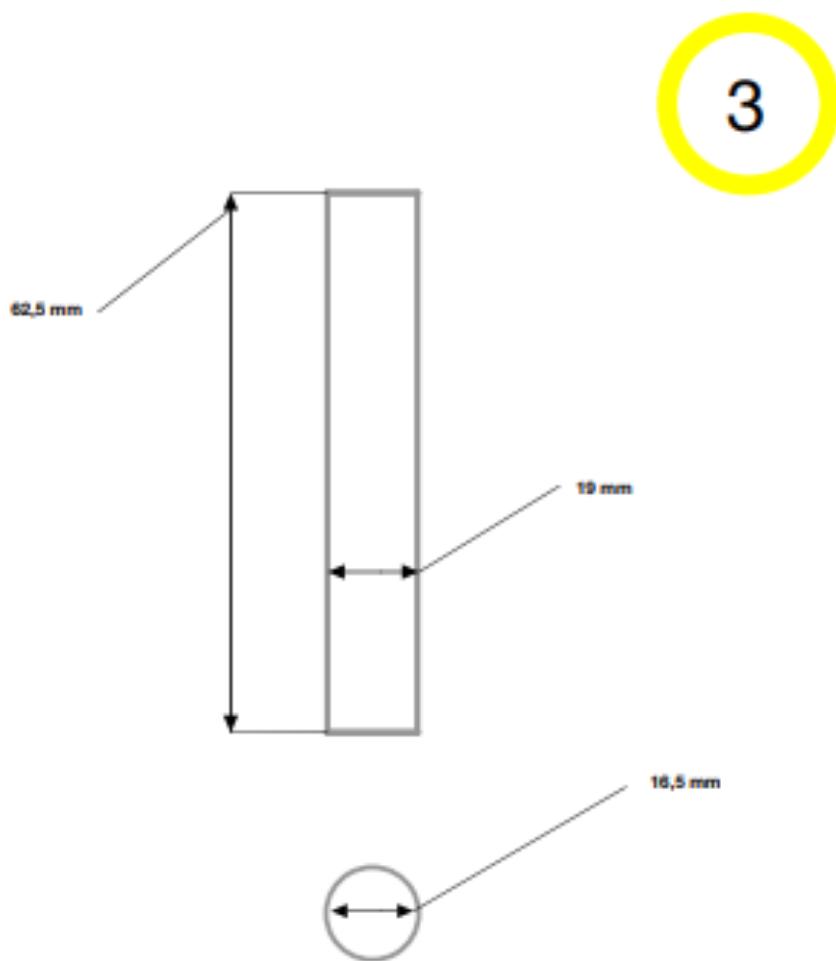


Abbildung 1.4: Schemaskizze Rohr Zwei Teil Drei

Die obige Schemaskizze zeigt das untere Rohr. An Rohr Zwei, Teil Drei wurden ebenfalls nur die Maße der Länge, der Innen sowie dem Außen-Durchmesser ermittelt. Diese sind in der obigen Abbildung ersichtlich. Rohr Zwei ist im Durchmesser dünner als Rohr Eins. Daher wird Rohr Zwei in Rohr Eins gesteckt. Mit Hilfe von Zehn Bohrungen an Rohr Zwei, sowie einem Steckverbinder und der Bohrung an Rohr Eins kann die Höhe der Krücke auf die Größe des jeweiligen Menschen angepasst werden.

### 1.2.5 Gumminoppe

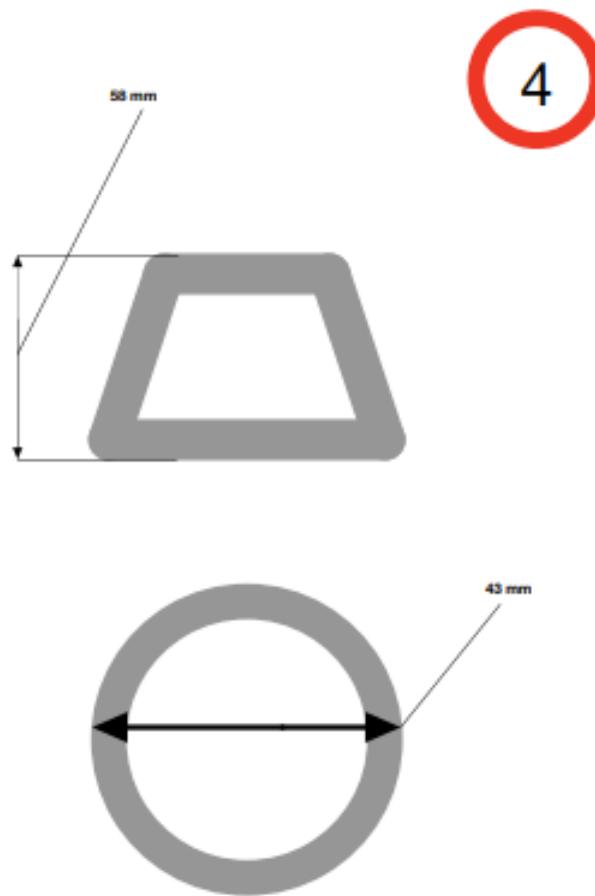


Abbildung 1.5: Schemaskizze Gumminoppe Teil Vier

Dies ist die Schemaskizze der Gumminoppe, wieder mit den wichtigsten Maßen versehen. Die Höhe der Noppe sowie den Durchmesser der Gumminoppe an der Unterseite. Die Gumminoppe ist, wie der Krückenkopf, kraftschlüssig mit dem angrenzenden Bauteil verbunden. Weiters wurde ein Kraftsensor in der Gumminoppe verbaut dieser ermittelt die Kraft welche jeweils auf die jeweilige Krücke gestützt wird.

## 1.3 Inventorzeichnungen<sup>2</sup>

Die Konstruktion der Krücke wurde mit Hilfe von Autodesk Inventor, einer 3D-CAD-Software, umgesetzt. Dieses Programm ist sehr umfangreich und bei einfachen Querschnitten wie Rohr Eins und Zwei gibt es keine Probleme mit der Konstruktion. Inventor funktioniert grundsätzlich einfach. Um ein 3-Dimensionales Objekt zu erzeugen wird im 2-Dimensionalen Raum eine Geometrie erstellt. Diese Skizze wird nun entweder um eine Achse gedreht oder in der Höhe extrudiert, sodass ein 3-Dimensionales Objekt entsteht. Beim Krückenkopf jedoch, war dies nicht so einfach. Hierfür wurden immer wieder verschiedenste Techniken verwendet um die jeweiligen Rundungen und Maße einzuhalten.

### 1.3.1 Was ist Inventor?

Inventor® ist eine CAD-Software die professionelle Werkzeuge für die mechanische 3D-Konstruktion, Dokumentation und Produktsimulation bietet.

### 1.3.2 Was kann Inventor?

Inventor ist ein sehr komplexes Programm welches die verschiedensten mechanischen Prozesse umsetzen kann und nicht nur neue Produkte erschaffen kann. Weiters können diese Produkte auch einfach mechanisch belastet sowie getestet werden. Dadurch ist ihre Belastbarkeit sowie Nutzdauer, Festigkeit und vieles mehr feststellbar.



Abbildung 1.6: Logo Inventor

---

<sup>2</sup>vgl. Inventor, (2022)

### 1.3.3 Inventor Konstruktion 1.0

Die folgende Konstruktion zeigt eine handelsüblichen Krücke ohne Sensoren.

#### 1.3.3.1 Gesamte Krücke



Abbildung 1.7: Gesamte Krücke mit Inventor 1.0

Diese Bild zeigt die Zusammenstellung aller Bauteile via Inventor. Die Bauteile wurden zuerst jeweils einzeln konstruiert und anschließend als eine Baugruppe zusammengefügt. Weiters wurden die einzelnen Teile voneinander abhängig gemacht, sodass sie als ein Bauteil agieren. Man sieht den Krückenkopf ganz oben. Dieser ist kraftschlüssig mit Rohr Eins verbunden. In Rohr Eins ist Rohr Zwei gesteckt. Der Rohrverbinder, wie der Name sagt, verbindet die beiden Rohre und hält sie in der gewünschten Position. Am unteren Ende befindet sich die Gumminoppe, welche kraftschlüssig mit Rohr Zwei verbunden ist.

### 1.3.3.2 Krückenkopf



Abbildung 1.8: Krückenkopf mit Inventor

Dies ist der fertig konstruierte Krückenkopf. Hierfür wurden 64 Einzelskizzen und 43 Rundungselemente mit Hilfe von Inventor gefertigt. Um den Krückenkopf zu fertigen wurde der Krückenkopf zuerst nur einseitig konstruiert und anschließend die andere Seite aufgrund der Symmetrie gespiegelt. Weiters wurde als Material in Inventor ein grauer Kunststoff ausgewählt. Leider war es nicht möglich jedes Detail der Krücke Eins zu Eins zu übernehmen. Da manche Rundungen mit Inventor nicht umsetzbar waren. Das Grund-Modell war eine Krücke von "Rebotec"<sup>3</sup>. Diese habe ich genau vermessen und die jeweiligen Maße übernommen um den Krückenkopf des iWalk zu konstruieren.

---

<sup>3</sup>vgl. Rebotec, (2022)

### 1.3.3.3 Rohr Eins



Abbildung 1.9: Rohr Eins mit Inventor

Rohr Eins ist das obere Rohr der Krücke. Es ist mit dem Krückenkopf fest verbunden und hat eine Bohrung. Diese Bohrung dient dazu das Rohr Zwei (untere Rohr) mithilfe des Verbindungssteckers zu fixieren. Das Rohr Eins ist aus Metall gefertigt und muss den Belastungen einer Person von bis zu 100kg standhalten. Dabei darf sich das Rohr weder Verformen noch seine statischen Eigenschaften verlieren. Außerdem sollte ein leichtes Metall verwendet werden, sodass das Heben der Krücke so leicht wie möglich fällt.

### 1.3.3.4 Rohr Zwei

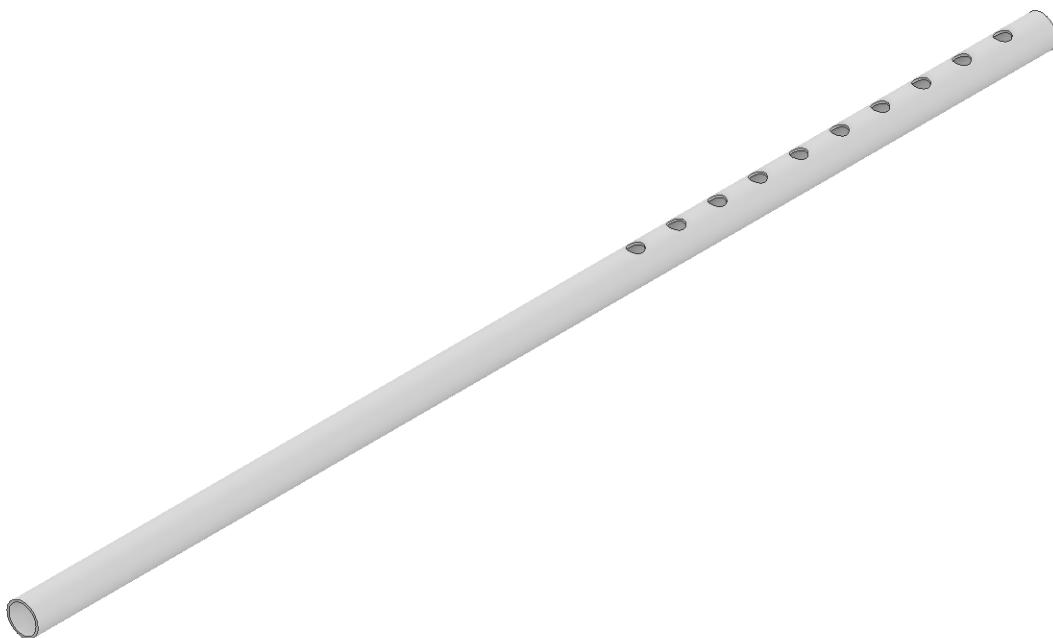


Abbildung 1.10: Rohr Zwei mit Inventor

Bei Rohr Zwei handelt es sich auch um ein Metallrohr. Dieses darf auf keinen Fall chemisch mit dem anderen Rohr reagieren, deshalb wird das gleiche Material verwendet, wie bei Rohr Eins. Es darf ebenfalls nicht zu schwer sein, muss dennoch den Belastungen standhalten. Rohr Zwei hat mehrere kritische Querschnitte am oberen Ende des Rohres. Da dort zehn Bohrungen im Abstand von 25 mm angebracht sind, könnte es sein, dass die Dicke des Rohres größer sein muss als bei Rohr Eins. Diese Löcher sind vorhanden, um die Länge der Krücke auf die entsprechende Höhe der Person einzustellen.

### 1.3.3.5 Gumminoppe

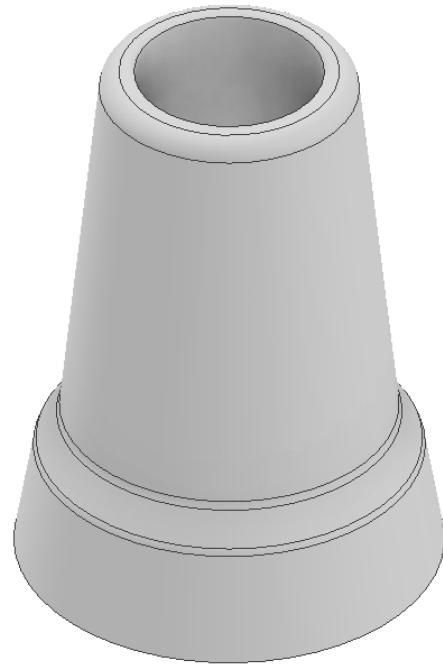


Abbildung 1.11: Gumminoppe mit Inventor

Die obige Abbildung zeigt die Gumminoppe. Diese ist auf Rohr Zwei (unteres Rohr) aufgesteckt. Hier befindet sich auch der Kraftsensor. Dieser misst wie viel Kraft auf die jeweilige Krücke drückt. Die Gumminoppe ist aus einem elastischen Kunststoff gefertigt. Die Noppe sorgt für den richtigen Halt der Krücke. Außerdem sorgt sie dafür, dass jeder einzelne Schritt ein wenig gedämpft wird.

### 1.3.3.6 Rohrverbinder



Abbildung 1.12: Rohrverbinder mit Inventor

Dieser Rohrverbinder besteht außen aus Kunststoff und der runde Stift in der Mitte ist aus Metall. Es handelt sich um das Mittelstück der Krücke. Der Rohrverbinder hält Rohr Eins und Rohr Zwei zusammen an den zuvor erwähnten Bohrungen. Mit den Bohrungen ist eine gesamte Längenänderung von 250 mm möglich. Durch die Steckverbindung von Rohr Eins, Rohr Zwei und dem Rohrverbinder ist die Krücke komplett. Jetzt spricht man von einer Baugruppe aus dem zuvor genannten Teilen.

### 1.3.4 Inventor Konstruktion 2.0

Die Konstruktion 2.0 basiert auf der Konstruktion 1.0 und verändert eine handelsübliche Krücke zum iWalk. Hierfür werden die Sensoren sowie der Microcontroller und die Akkuzelle hinzugefügt. Einige Bauteile mussten dafür bearbeitet werden, um Platz zu schaffen oder die Funktion der Sensoren zu gewährleisten.

#### 1.3.4.1 Gesamte Krücke mit Sensoren



Abbildung 1.13: Gesamte Krücke 2.0

### 1.3.4.2 Gesamte Krücke 2.0

Die obige Abbildung zeigt die Zusammenstellung aller Bauteile. Da die Sensoren sowie der Akku und der Microcontroller in der Krücke verbaut sind, sind diese von außen nicht ersichtlich. Nur die Halterung für den Beschleunigungssensor ist sichtbar. Diese ist in der roten Abdeckung des Krückenkopfes erkennbar. Außerdem wurde die Gumminoppe neu konstruiert. Dies wird auch dargestellt.

### 1.3.4.3 Beschleunigungssensor und Microcontroller

Der Krückenkopf selbst musste nicht bearbeitet werden. Es wurde nur der Beschleunigungssensor sowie der Microcontroller hinzugefügt und im Krückenkopf platziert.

### 1.3.4.4 Problem Microcontroller

Der ausgewählte Microcontroller ESP 32 sollte passend in die Krücke verbaut werden. Hierbei trat folgendes Problem auf. Der Microcontroller ESP 32 ist mit den Maßen von 52 x 31 x 12 Millimeter leider zu groß. Daher gab es zwei Lösungsmöglichkeiten.

### 1.3.4.5 Lösung Eins

Der Krückenkopf muss dafür in seiner Form geändert werden. Es müsste eine Konstruktion angefügt werden, in welcher der Microcontroller Platz finden würde. Dies würde nicht nur die Stabilität sondern auch das Gewicht der Krücke verändern. Daher wurde dieser Punkt nicht umgesetzt.

### 1.3.4.6 Lösung Zwei

Es wird eine Platine hergestellt, welche den Hauptchip und das Bluetooth-Modul um einiges verkleinert, sodass die Platine auf eine gesamte Größe von 70 x 20 x 9 Millimeter kommt. Der Kontroller ändert dabei nicht seine Funktion. Bei diesem Prozess ändert sich ausschließlich die Bauform der Platine. Diese Platine findet dann Platz im Griff des Krückenkopfs.

#### 1.3.4.7 Microcontroller ESP32

Der Microcontroller wird Gelb dargestellt. Er wird mit Hilfe einer Halterung im Krückenkopf befestigt. Die Maße des Kontroller sind 70 x 20 x 9 Millimeter.



Abbildung 1.14: Microcontroller ESP 32

#### 1.3.4.8 Halterung des Microcontroller

Die folgende Halterung ist unbedingt notwendig. Sie sorgt dafür, dass der Microcontroller immer am gleichen Punkt der Krücke ist. Weiters verhindert sie Schäden bei Erschütterungen sowie dem losen herumbewegen in der Krücke. Die Halterung ist in der nachfolgenden Abbildung in rot gehalten. Diese Halterung wird anschließend in die vordere Öffnung des Krückenkopfs gesteckt und sorgt dadurch für die Sicherheit des ESP32.

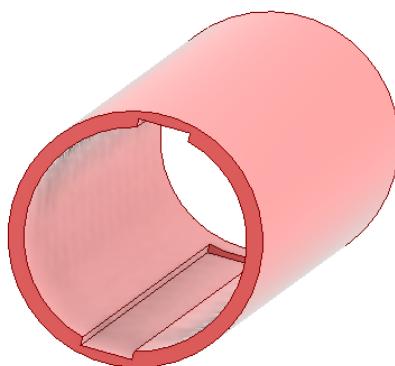


Abbildung 1.15: Microcontroller Halterung

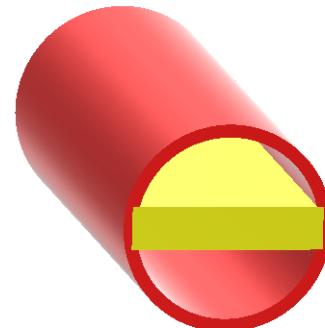


Abbildung 1.16: Microcontroller Halterung mit ESP32

### 1.3.4.9 Rohr Eins Mit Akku und Akkuhalterung

Beim Akku handelt es sich um den Beltrona 18650XH 2.54 Spezial-Akku. Dieser Akku hat mit einem Durchmesser von 18.6 Millimeter perfekt in Rohr Eins Platz. Er ist mit Hilfe eines einfachen Stecksystems in Rohr Eins versenkt. Der Akku selbst wird an diese Halterung angeklebt. In der folgenden Abbildung ist die Konstruktion des Akkus ersichtlich. Der Akku hat eine Gesamtlänge von 68.3 Millimeter.

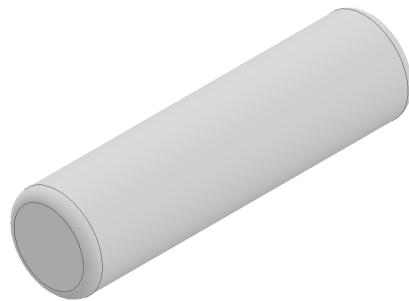


Abbildung 1.17: Beltrona 18650XH 2.54 Spezial-Akku

Die Akkuhalterung ist wieder in Rot gehalten. Die Akkuhalterung weist eine Einbuchtung auf. Diese Einbuchtung sorgt dafür, dass die Kabel des Akkus, sowie die Kabel des Drucksensors zum Microcontroller gelangen.

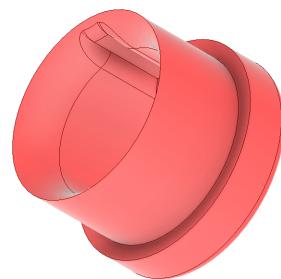


Abbildung 1.18: Akkuhalterung

Die Halterung ist mit einer Rundung versehen. Hier wird der Akku mit der Halterung verklebt. Die Kabel des Akkus und des Drucksensors können anschließend durch die Einkerbung geführt werden und somit den Microcontroller erreichen. In der folgenden Abbildung ist der Akku und die Halterung vereint.



Abbildung 1.19: Akku und Akkuhalterung

Diese beiden Teile werden anschließend passend in Rohr Eins geführt und verschwinden danach im Krückenkopf. Der obere Durchmesser beträgt 22 Millimeter, der untere Durchmesser 19 Millimeter.

#### 1.3.4.10 Rohr Zwei mit Berührungssensor

Rohr Eins und Zwei sind immer noch mit der Steckverbindung verbunden. Diese hat sich in keinerlei Hinsicht verändert. Bei Rohr Zwei dagegen gab es einige große Änderungen. In Rohr Zwei wurden der Berührungssensor sowie eine neue Fixierung der Gumminoppe hinzugefügt.

#### 1.3.4.11 Kraftsensor

Der Berührungssensor Joy-it SEN-Pressure20 ist im Messbereich 10 mm breit und die Platine beträgt 14 Millimeter. Der gesamte Sensor ist 70 Millimeter lang. Der biegsame Messteil erstreckt sich über 40 Millimeter und die Platine ist 30 Millimeter lang. Der Sensor ist gebogen, sodass er auf den Stoppel von Rohr Zwei geklebt werden kann und ganzflächig auf die Gumminoppe drückt. Dies muss gewährleistet sein, um keine falschen Ergebnisse zu messen.



Abbildung 1.20: Kraftsensor

### 1.3.4.12 Rohr Zwei mit Noppenhalterung und Berührungssensor

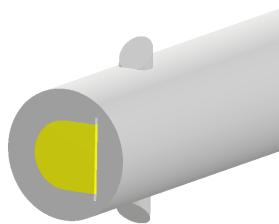


Abbildung 1.21: Rohr Zwei mit Noppenhalterung und Berührungssensor

Der Berührungssensor kann durch die Öffnung in Rohr Zwei gesteckt werden und anschließend auf der Fläche verklebt werden. Weiters wurde ein neuer Bolzen hinzugefügt. Dieser ist nach unten abgeschrägt. Dies sorgt für eine einfache Montage der Gumminoppe. Dieser Bolzen ist nun notwendig, da die Noppe nicht mehr kraftschlüssig mit dem Rohr verbunden sein kann (Verfälschung der Messergebnisse).

### 1.3.4.13 Stoppel für Rohr Zwei

Für Rohr Zwei wird ein handelsübliches Aluminiumrohr verwendet. Dieses wird nicht in seiner Form verändert. Jedoch muss das Rohr an der Unterseite geschlossen und der Berührunssensor soll möglichst einfach einzubauen sein. Dadurch wurde ein Stoppel konstruiert, welcher Rohr Zwei an der Unterseite verschließt. An diesem Stoppel wird der Berührungssensor verklebt. Der Stoppel wird mit Hilfe einer Bohrung und dem Verbindungsbolzen von Rohr Zwei und der Gumminoppe an Rohr Zwei fixiert. Die Konstruktion ist in der folgenden Abbildung ersichtlich.



Abbildung 1.22: Stoppel an Rohr Zwei

#### 1.3.4.14 Gumminoppe 2.0

Die Gumminoppe musste ebenfalls in ihrer Form bearbeitet werden. Es wurde eine Hülse hinzugefügt. Diese ist mit einer horizontalen Einkerbung beidseitig versehen. Durch diese Einkerbung ist Rohr Zwei mit der Gumminoppe verbunden. Außerdem wird der Berührungssensor entlastet, sobald die Krücke den Boden verlässt. Die Hauptfunktion des Bolzen besteht darin, dass die Gumminoppe nicht bei jedem Schritt verloren geht. Außerdem musste ein Plateau konstruiert werden, sodass nur die Gumminoppe auf den Berührungssensor drückt. Das Plateau ist 10mm breit und 5mm hoch.



Abbildung 1.23: Gumminoppe Hülse Einkerbung



Abbildung 1.24: Gumminoppe Plateau

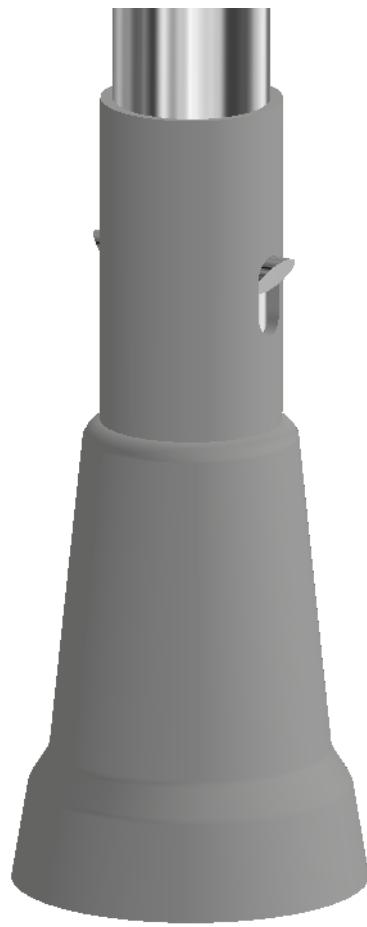
**1.3.4.15 Rohr Zwei und Gumminoppe**

Abbildung 1.25: Rohr Zwei mit Gumminoppe

Die obige Abbildung zeigt die Gumminoppe und Rohr Zwei. Der Bolzen sorgt dafür, dass die Noppe am Rohr befestigt ist. Der Berührungssensor befindet sich im Inneren und ist von außen nicht zu sehen.

#### 1.3.4.16 Ansicht der Bauteile von Innen

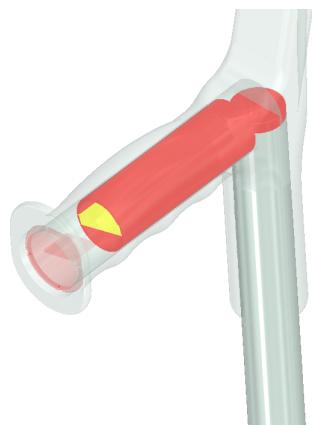


Abbildung 1.26: Krückenkopf sowie Rohr Eins mit Innenleben



Abbildung 1.27: Gumminoppe und Rohr Zwei mit Innenleben

Die obigen Abbildungen zeigen den Innenaufbau der Krücke. In rot sind alle Halterungen und in gelb die Elektronikbauteile dargestellt.

## 1.4 Versuch mit Waage



Abbildung 1.28: dynamische Gewichtsmessung mit einer Waage, Marke Korona

Mit Hilfe einer handelsüblichen Waage wurde gemessen mit wieviel Gewicht die Krücke maximal einseitig belastet wird. Der Versuch wurde von einer Testperson mit 100 Kilogramm durchgeführt und das maximale Ergebnis beträgt circa dem halben Gewicht der Person. Mithilfe dieses Tests wurden weitere Berechnungen durchgeführt.

### Werte der Messversuche:

Versuch 1	Versuch 2	Versuch 3
52.3kg	54.8kg	56.4kg

Tabelle 1.1: Messwerte

Der Mittelwert der drei gemessenen Werte liegt bei 55 Kilogramm. Daher werden für weitere Berechnung dieser ermittelte Wert hergenommen. Die Testreihe wurde vergleichsweise auch mit einer leichteren und einer schwereren Person durchgeführt. Auch hier kam zirka die Hälfte des Eigengewichts heraus.

## 1.5 Kraftanalyse

Die Kraftanalyse soll zeigen, ob die Krücke jegliche Belastungen einer Testperson mit maximal 100 Kilogramm aushält. Dafür wurden Analysen mit Inventor und Berechnungen mit Hilfe des Decker Maschinenelemente 20 Auflage durchgeführt.

### 1.5.1 Inventor

Mit Hilfe von Inventor wurden die Analysen des Krückenkopfs, Rohr Zwei und der Gumminoppe simuliert. Diese Teile wurden am meisten belastet. Daher wurden die Krücke mit der Kraft von 539,55Newton bzw der angenommen Belastung von 55 Kilogramm belastet.

#### 1.5.1.1 Krückenkopf Belastung Inventor

Der Krückenkopf wird mit der Kraft der Hand in der Mitte des Griffes am Krückenkopf belastet. Die Krücke aus Polyamid wird nur minimalst verbogen. Dies ist in der folgenden Abbildung zu sehen. In der Abbildung ist die Biegung überzeichnet dargestellt.

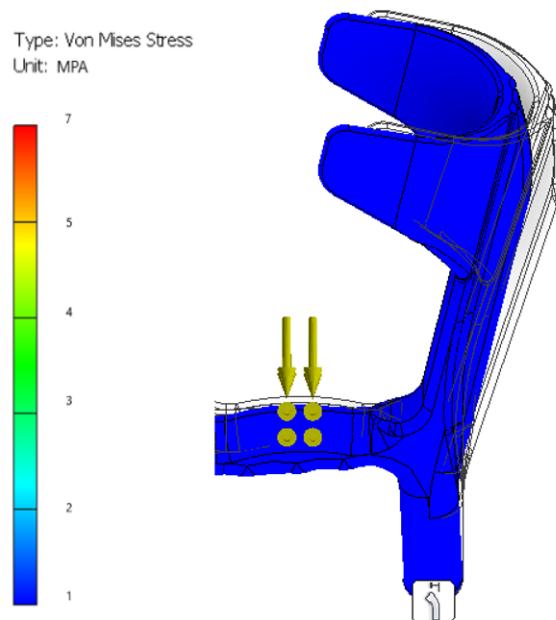


Abbildung 1.29: Belastung Krückenkopf

### 1.5.1.2 Rohr Zwei Belastung

Rohr Zwei wird über den Rohrverbinder von oben mit der Kraft von 539.55 Newton belastet. Diese Belastung ist an allen 10 Bohrungen ersichtlich. Außerdem herrscht eine große Belastung am Verbindungsstück zwischen Rohr Eins und Zwei. Ersichtlich ist das am Loch von Rohr Zwei. Weiters wird Rohr Zwei am unteren geschlossenen Ende belastet, da dort der Berührungssensor auf die Gumminoppe drückt. Diese drei Belastungsanalysen werden in den folgenden Abbildungen sichtbar gemacht.

### 1.5.1.3 Zehn Bohrungen an Rohr Zwei

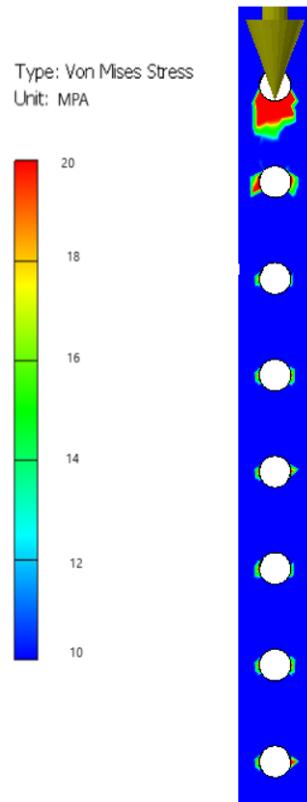


Abbildung 1.30: Belastung der zehn Bohrungen

#### 1.5.1.4 Bohrung Verbindungsstück Rohr Zwei

Es ist ersichtlich, dass die Bohrung an der oberen Stelle einer großen Belastung ausgesetzt ist. Diese Belastung entsteht an dieser Stelle da die gesamte Kraft von Rohr Eins über den Rohrverbinder auf Rohr Zwei übertragen wird.

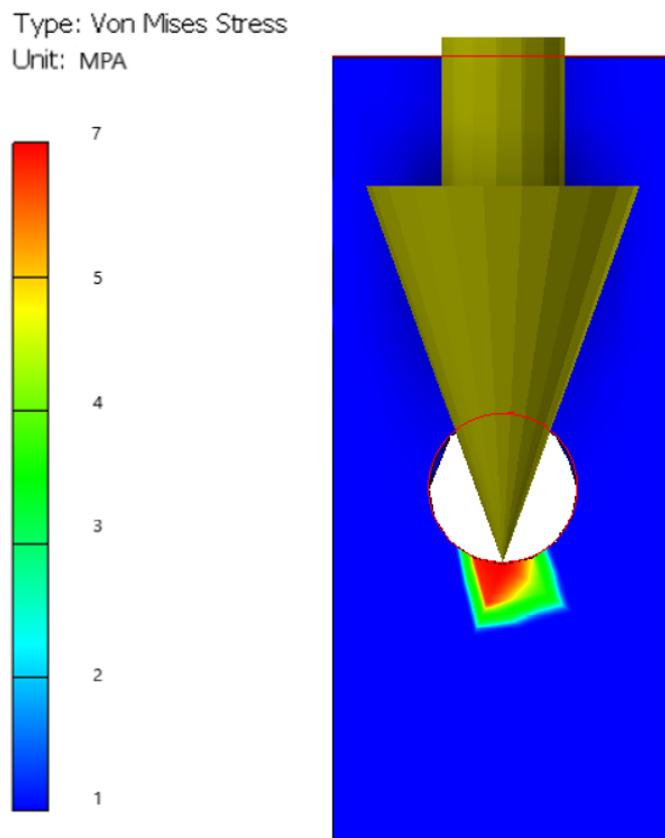


Abbildung 1.31: belastete Bohrung

### 1.5.1.5 Rohr Zwei Belastung Unterseite

An dieser Stelle überträgt Rohr Zwei die Belastung auf die Gumminoppe. Zwischen Gumminoppe und Rohr Zwei befindet sich der Berührungssensor. Dieser misst die genaue Belastung und gibt diese dem Microcontroller weiter. Die folgende Abbildung zeigt die Belastung.

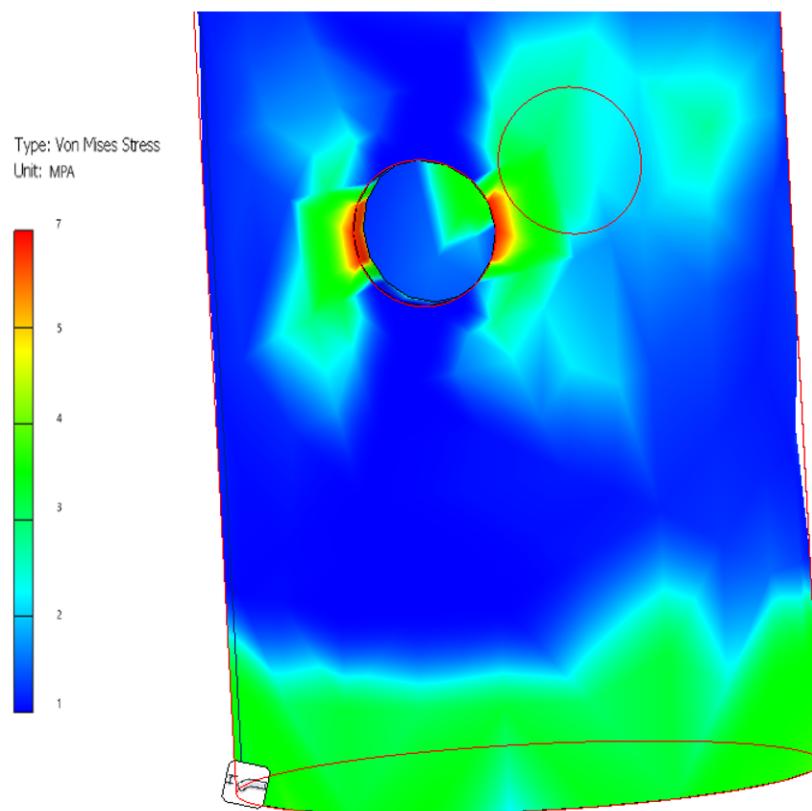


Abbildung 1.32: Belastung Rohr Zwei Unterseite

### 1.5.1.6 Belastung Gumminoppe an Rohr Zwei

Die Gumminoppe wird hauptsächlich an zwei Punkten belastet. Als Erstes direkt an der Verbindung von Rohr Zwei zur Gumminoppe. Hier drückt der Berührungssensor mit der Kraft von 539.55 Newton auf die Gumminoppe. Weiters entsteht eine Belastung zwischen der Gumminoppe und dem Untergrund auf den sie drückt. Diese zwei Belastungen werden in den folgenden Abbildungen dargestellt.

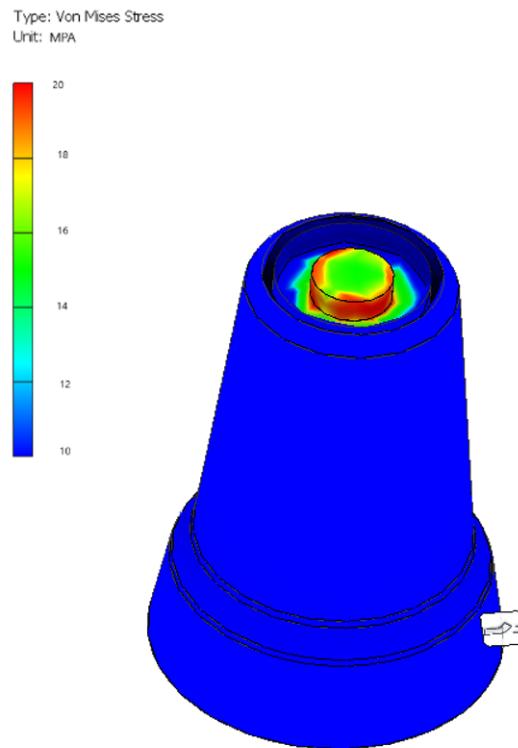


Abbildung 1.33: Belastung Gumminoppe Rohr Zwei

Die obige Abbildung zeigt, dass sich die Krücke an dieser Stelle am meisten verformt. Diese Ansicht wurde ohne die obige Hülse präsentiert. Da die Belastung ansonsten nicht ausreichend sichtbar ist.

### 1.5.1.7 Belastung Gumminoppe zum Boden

Die folgende Abbildung zeigt die Belastung zwischen Gumminoppe und dem jeweiligen Untergrund, auf den sie gestellt wird. Die Belastungsskala wurde angepasst, um die Belastung besser darzustellen. Die Hauptbelastung wird auf das Plateau abgegeben. Für diese Analyse wurde die Hülse wieder eingeblendet. Die Hülse wird nicht von Rohr Zwei belastet. Der Bolzen ist rein zur Sicherung der Gumminoppe gedacht. Er sorgt dafür, dass die Gumminoppe bei der Bewegung mit der Krücke nicht verloren geht.

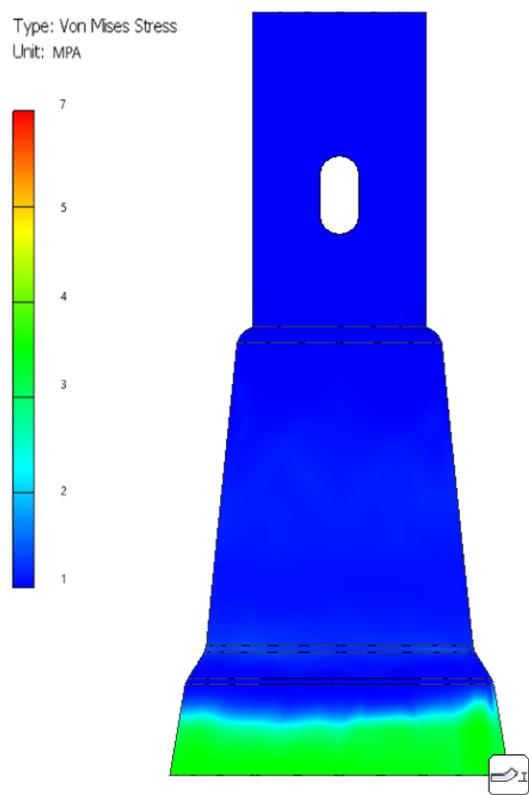


Abbildung 1.34: Belastung Gumminoppe Boden

## 1.5.2 Mechanische Berechnung

Mit Hilfe des Decker Maschinen Elemente 20 Auflage wurden die wichtigsten mechanischen Berechnungen der Bauteile durchgeführt. Zuerst wurde die Biegesicherheit des Krückenkopfes bestimmt. Als nächstes wurde die Knicksicherheit am meistbelasteten Rohr Zwei ermittelt und anschließend wurde die Abscherung sowie Flächenpressung am Rohrverbinden ermittelt.

### 1.5.2.1 Biegung an Krückenkopf

Um die Biegung am Krückenkopf zu berechnen wurde eine Freimachskizze erstellt. Diese ist in der folgenden Abbildung ersichtlich. Der rote Pfeil beschreibt die Kraft, die auf den Krückenkopf wirkt. Diese beträgt 539,55 Newton. Die Länge von 50 Millimeter beschreibt den Abstand der Kraft zum voraussichtlichen Punkt der Abbiegung.

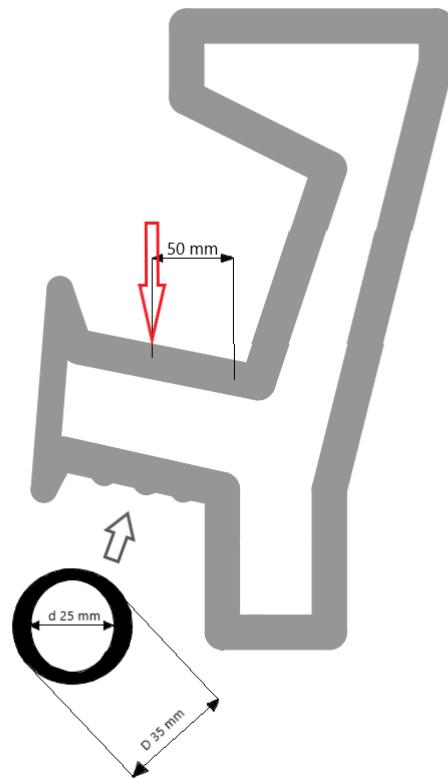


Abbildung 1.35: Freimachskizze Krückenkopf Biegung

### 1.5.2.2 Berechnung der Biegesicherheit<sup>4</sup>

Die Biegespannung Sigma b wird aus dem Biegemoment Mb durch den Biegewiderstandsmoment Wb ermittelt.

Die Formel lautet:

$$\sigma_b = M_b / W_b$$

$$\sigma_b = 26977.5 / 3113.522 = 8.664$$

Das Biegemoment Mb wird aus der wirkenden Kraft F und dem Abstand der Kraft zum voraussichtlichen Punkt der Abbiegung l berechnet.

Die Formel lautet:

$$M_b = F * l$$

$$M_b = 539.55 * 50 = 26977.5 Nmm$$

Das Biegewiderstandsmoment Wb wird aus dem Flächenträgheitsmoment I durch den Randfaserabstand e ermittelt.

$$W_b = I / e$$

$$W_b = 54486.635 / 17.5 = 3113.522$$

---

<sup>4</sup>vgl. Decker, (2022)

Das Flächenträgheitsmoment I wird mit Pi mal Durchmesser Außen (da) hoch 4 durch 64 minus Pi mal Durchmesser Innen (di) hoch 4 durch 64 berechnet:

$$I = \pi d_a^4 / 64 - \pi d_i^4 / 64$$

$$I = \pi 35^4 / 64 - \pi 25^4 / 64 = 54486.635$$

Der Randfaserabstand e ist der halbe Durchmesser Außen (da/2).

$$e = d_a / 2$$

$$e = 35 / 2 = 17.5$$

Zuletzt wurde die Biegesicherheit bestimmt. Hierfür muss Sigma b kleiner sein als die zulässige Streckgrenze für Polyamid durch die Sicherheit. Die zulässige Streckgrenze steht im Decker Seite 294 zur Sicherheit wurde 3 gewählt.

$$\sigma_b < \text{Streckgrenze/Sicherheit}$$

$$8.6645 < 85 / 3 \rightarrow 8.6645 < 28.3333$$

Die Biegesicherheit ist Hoch genug, da 8.6645 kleiner ist als 28.3333.

### 1.5.2.3 Knickung an Rohr Zwei

Um die Knickung richtig zu berechnen wird der Querschnitt des knickenden Objektes (siehe folgende Abbildung) sowie die freie Knicklänge  $l_k$  und der Knickungsfall benötigt. Bei Rohr Zwei traf der Fall 4 ein. Dieser beschreibt das  $l_k$  die halbe Gesamtlänge (625 Millimeter) des Objekts beträgt, weil das Rohr Zwei oben in Rohr Eins geführt wird.

$$l_k = l/2$$

$$l_k = 625/2 = 312.5\text{mm}$$

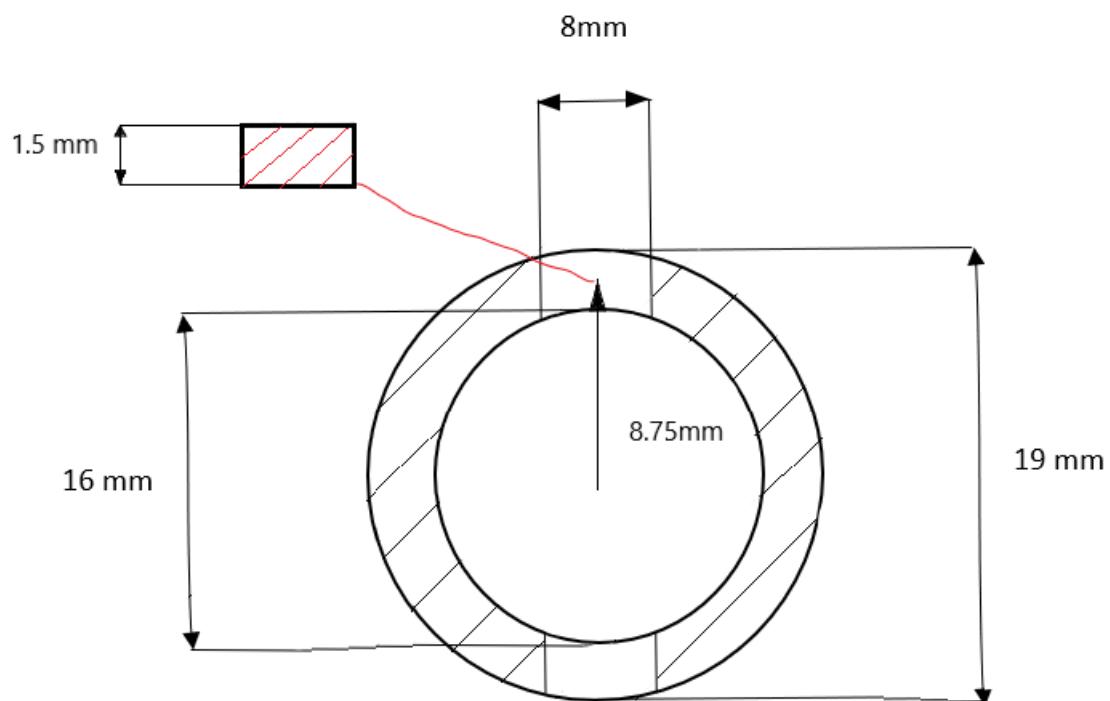


Abbildung 1.36: Querschnitt des Rohres an der Bohrung

#### 1.5.2.4 Berechnung des Schlankheitsgrades Lambda

Der Schlankheitsgrad Lambda wird berechnet, um zu ermitteln, ob die Knicksicherheit nach Euler oder Tetmajer bestimmt wird.

$$\lambda = l_k/i$$

$$\lambda = 312.5/2.09 = 149.4 N/mm^2$$

Um Lambda zu berechnen wird außerdem der Trägheitsradius  $i$  benötigt. Dieser wird aus der Wurzel vom Flächenträgheitsmoment  $I$  durch die Querschnittsfläche  $A$  berechnet. Das folgende Flächenträgheitsmoment  $I$  wird mit Pi multipliziert Durchmesser Außen (da) hoch 4 dividiert durch 64 Minus Pi multipliziert mit dem Durchmesser Innen (di) hoch 4 dividiert durch 64 abzüglich der beiden Löcher mit Hilfe des Satz von Steiner berechnet. Dadurch wird noch extra die Breite der Löcher  $b$  multipliziert die Höhe der Löcher  $h$  Hoch Drei durch Zwölf Plus dem Durchmesser zum Mittelpunkt der Fläche siehe obige Abbildung (8.75mm) zum Quadrat multipliziert der Fläche des Rechtecks multipliziert mit Zwei abgezogen.

$$I = \pi d_a^4 / 64 - \pi d_i^4 / 64 - (bh^3 / 12 + r^2 * A) * 2$$

$$I = \pi 19^4 / 64 - \pi 16^4 / 64 - (8 * 1.5^3 / 12 + 8.75^2 * 8 * 1.5) * 2 = 1338.126$$

Die Querschnittsfläche  $A$  ohne Löcher (siehe schwarz schraffierte Fläche obige Abbildung) wird wie folgt berechnet. Die Fläche beträgt 305.86 Quadratmillimeter.

$$A = r_a^2 * \pi - r_i^2 * \pi - 2 * b * h$$

$$A = 9.5^2 * \pi - 8^2 * \pi - 2 * 8 * 1.5 = 305.86 mm^2$$

Weiters wird der Trägheitsradius  $i$  mit Hilfe der Wurzel vom Flächenträgheitsmoment  $I$  durch die Fläche  $A$  berechnet.

$$i = \sqrt{I/A}$$

$$i = \sqrt{1338.126/305} = 2.09$$

Knickung nach Euler oder Tetmajer. Ist Lambda größer als Lambda Grenz so wird Euler verwendet, anderenfalls Tetmajer. Lambda Grenz wird mit der Formel Pi mal Wurzel aus Elastizitätsmodul durch Streckgrenze berechnet. Der Elastizitätsmodul von Aluminium ist 70.000. Dieser Wert ist ersichtlich im Decker. Sigma p ist 0.8 mal 200 dies entspricht 160. Da Sigma größer ist als Sigma Grenz wird mit der Formel von Euler gerechnet.

$$\lambda_{grenz} = \pi * \sqrt{E/\sigma_p}$$

$$\lambda_{grenz} = \pi * \sqrt{70000/160} = 65.711 N/mm^2$$

$$\lambda > \lambda_{grenz}$$

$$149.4 N/mm^2 > 65.711 N/mm^2 \rightarrow Euler$$

### 1.5.2.5 Berechnung der Knickungssicherheit

Die Knickungssicherheit S wird mit Sigma k durch Sigma v berechnet. Sigma k ist die Knickspannung und Sigma v ist die vorhandene Spannung.

$$\sigma_k = \pi^2 * E / \lambda^2$$

$$\sigma_k = \pi^2 * 70000 / 149.5^2 = 30.95 N/mm^2$$

$$\sigma_v = F/A$$

$$\sigma_v = 539.55 / 305.86 = 1.764 N/mm^2$$

$$S = \sigma_k / \sigma_v$$

$$S = 30.95 / 1.764 = 17.55$$

Die Knicksicherheit ist hoch genug. Das Rohr wird unter dieser Belastung nicht knicken.

### 1.5.2.6 Berechnung der Abscherung des Rohrverbinder

In der folgenden Abbildung sind die wichtigsten Maße sowie ein Schnitt der Verbindung zwischen Rohr Eins und Zwei mit Hilfe des Rohrverbinder sichtbar.

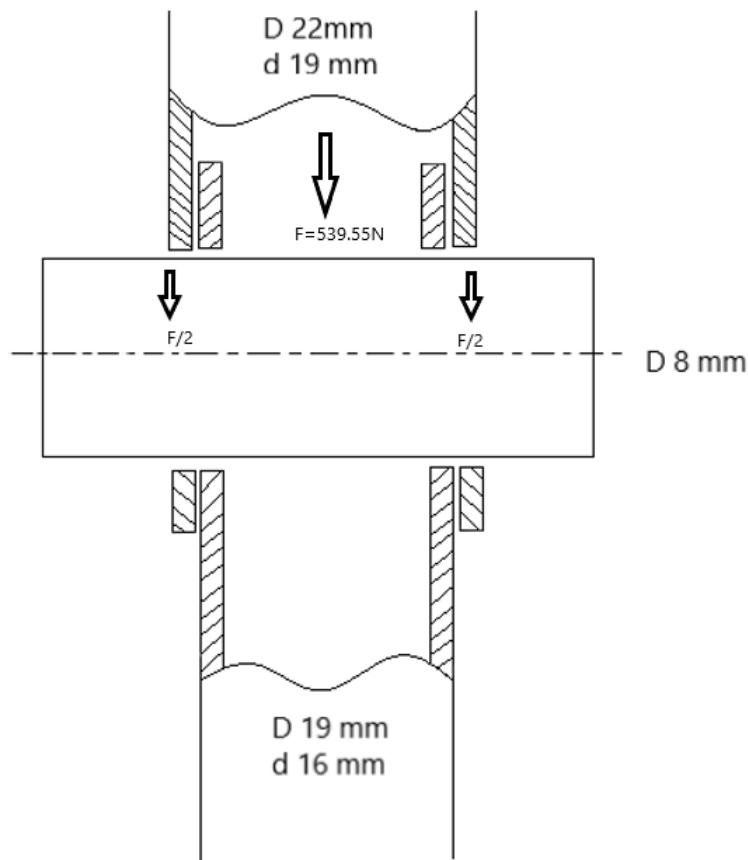


Abbildung 1.37: Verbindung Rohr Eins, Zwei und Verbinder

Die Abscherung Tau Ab wird mit der Kraft  $F_q$ , halbe Gesamtkraft (siehe obige Abbildung) durch die Fläche des Verbinder  $A$  berechnet.

$$T_{ab} = F_q / A$$

$$T_{ab} = 269.775 / 50.265 = 5.36 \text{ N/mm}^2$$

### 1.5.2.7 Berechnung der Flächenpressung des Rohrverbinder

Diese Berechnung bezieht sich auf die folgende Skizze. Die Flächenpressung  $p$  wird ebenfalls mit der Kraft  $F_q$  halbe Gesamtkraft durch die Fläche  $A$  berechnet. Doch bei der Flächenpressung handelt es sich nicht um die Fläche des Verbinder sondern die projizierte Fläche des abscherenden Verbinder (8\*1.5). 8 Millimeter ist in diesem Fall der Durchmesser des Verbinder und durch die Wandstärke des Rohres von 1,5mm ist die Abscherung, daher (8\*1.5=12).

$$p = F_q / A_{pro}$$

$$p = 269,775 / 12 = 22.481 N/mm^2$$

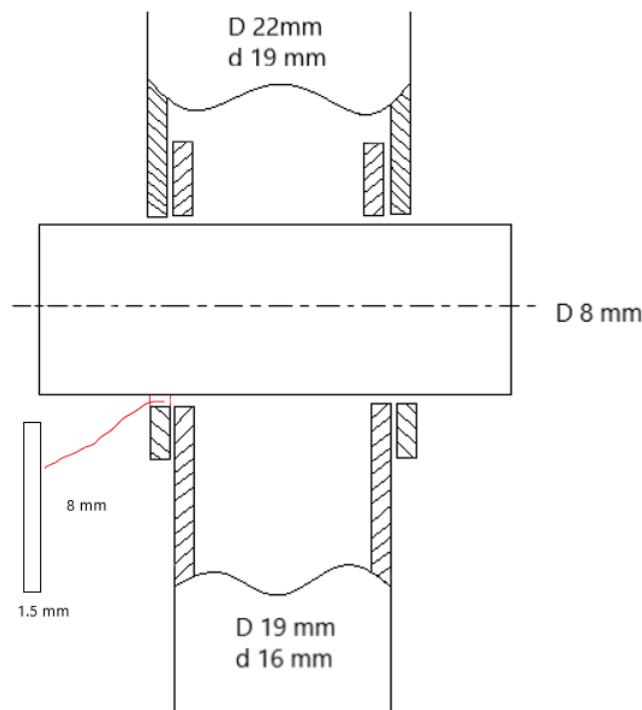


Abbildung 1.38: Verbindung Rohr Eins, Zwei und Verbinder mit Projektion

## 1.6 Materialauswahl

In der Krücke sind grundsätzlich drei Werkstoffe verbaut. Zwei Arten Kunststoffe, der Eine für den Krückenkopf und der weiteren Hartplastikteile, sowie der Andere für die Gumminoppe. Alle Hartplastikteile werden aus Spritzguss gefertigt. Der dritte Werkstoff ist ein Metall. Dieses Metall hat möglichst leicht und trotzdem stabil zu sein für die Rohre. In der Materialauswahl wird bestimmt welches Material die besten Eigenschaften hat und trotzdem preislich zum iWalk passt. Weiters soll die Produktion von rund 10.000 Stück möglich sein.

### Materialdetails:

Material	Dichte in g/cm <sup>3</sup>	Preis pro kg in €	E-Modul in N/mm <sup>2</sup>
PA 6.6	1.13	1	1200
POM-H	1.43	1.95	2600
Aluminium	2,71	2	70.000
Stahl	7,83	1	210.000
TPR	1	1.6	
TPU	1.22	1	

Tabelle 1.2: Materialien und Eigenschaften

### 1.6.1 Krückenkopf

Beim Krückenkopf wird ein Kunststoff gewählt der günstig ist und dazu noch der Belastung eines Menschen mit maximal 100 Kilogramm aushalten kann. Weiters soll der Kunststoff spritzgussfähig sein um die Stückzahlen von 10.000 zu erreichen. Zur Auswahl standen Polyamide (PA) sowie Polyoxytmethylen (POM C ).

### Materialdetails PA und POM C:

Material	Dichte in g/cm <sup>3</sup>	Preis pro kg in €	E-Modul in N/mm <sup>2</sup>
PA 6.6	1.13	1	1200
POM-H	1.43	1.95	2600

Tabelle 1.3: Wichtigste Daten der Kunststoffe

Da Polyamid leichter und günstiger ist als Polyoxytmethylen, wird die Krücke aus Polyamid gefertigt.

Weiters haben diese zwei Kunststoffe kaum Unterschiede der Belastbarkeit sowie Verformung im Fall des Krückenkopfs. Dies wurde mit Hilfe von Inventor dargestellt.

### 1.6.1.1 Materialvergleich

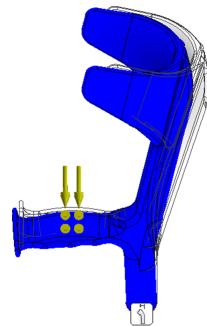


Abbildung 1.39: Krückenkopf Polyamid belastet mit 55 Kilogramm

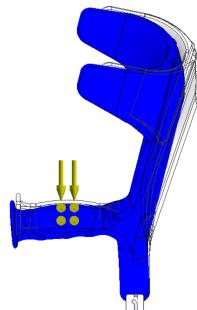


Abbildung 1.40: Krückenkopf Polyoxymethylen belastet mit 55 Kilogramm

Diese zwei Kunststoffe sind in diesem Festigkeitsfall sehr ähnlich. Da der Krückenkopf nur mit 55 Kilogramm belastet wird ist kaum Verformung feststellbar. Jedoch preislich und beim Gewicht wird klar, dass Polyamid die bessere Wahl ist. Polyamid ist um 20 Prozent leichter als Polyoxymethylen und nur halb so teuer.

### 1.6.1.2 Fazit

Der Krückenkopf wird aus Polyamidgranulat hergestellt. Dieser wird durch einen Spritzguss in eine Form gepresst und nach einer Abkühlzeit ist der Krückenkopf fertig.

## 1.6.2 Rohr Eins und Zwei

Bei den beiden Rohren handelt es sich um normale Rohre mit einer Wandstärke von 1.5 mm und einem Außendurchmesser von 22 mm sowie 19 mm. Diese Rohre sollen so leicht und günstig als möglich hergestellt werden. Die folgende Tabelle zeigt die wichtigsten Daten.

### Rohr 1 und 2 Stahl oder Aluminium:

Material	Dichte in g/cm <sup>3</sup>	Preis pro kg in €	E-Modul in N/mm <sup>2</sup>
Aluminium	2,71	2	70.000
Stahl	7,83	1	210.000

Tabelle 1.4: Daten der Metalle

Die Daten der Tabelle zeigen eindeutig, dass Stahl keine mögliche Option ist, da dieser beinahe dreimal so schwer ist wie Aluminium. Aluminium ist zwar teurer als Stahl, dennoch ist das geringe Gewicht dem Preis vorzuziehen.

### 1.6.2.1 Materialvergleich

Die Konstruktion mit Hilfe von Inventor ist hierfür nicht notwendig da die Rohre sowohl aus Aluminium als auch aus Stahl kaum gestaucht werden. Der angenommene Rohrdurchmesser hält einer Person mit maximal 100 Kilogramm locker stand.

### 1.6.2.2 Fazit

Sowohl Rohr Eins als auch Rohr Zwei werden aus Aluminium hergestellt. Anschließend werden die Rohre in ihre Länge gebracht und die Löcher werden gebohrt.

### 1.6.3 Gumminoppe

Die Gumminoppe wird aus einem elastischen Kunststoff hergestellt. Dieser sorgt in erster Linie für eine breitere Auflagefläche der Krücke. Dadurch ist es leichter das Gleichgewicht zu halten. Doch nicht nur die Oberfläche ist entscheidend. Die Beschaffenheit des Materials spielt auch eine Rolle. Die Gumminoppe hat gleichzeitig eine federnde Wirkung. Denn das elastische Material soll Stöße absorbieren und somit die Fortbewegung so einfach als möglich machen. Weiters wird durch die elastische Wirkung die Kraft ermittelt, die auf die Krücke wirkt. Zur Auswahl standen TPR (thermoplastisches Elastomer (TPE) oder TPU (thermoplastisches Polyurethan)).

#### 1.6.3.1 Materialvergleich

**TPU oder TPR:**

Material	Dichte in g/cm <sup>3</sup>	Preis pro kg in €
TPR	1	1.6
TPU	1.22	1

Tabelle 1.5: Wichtigste Daten von TPU und TPR

#### 1.6.3.2 Fazit

Bei der Gumminoppe wurde auf die Dichte und die Elastizität geachtet. Daher wird als Kunststoff TPR verwendet. Dieser ist zwar teurer als TPU, ist aber leichter und weist bessere elastische Eigenschaften auf.

## 1.7 Fertigung des iWalk

Der iWalk besteht aus zwei Kategorien von Bauteilen: Spezialteile sowie handelsübliche Teile. Die handelsüblichen Teile müssen nicht eigenständig produziert werden. Hierzu werden einfach fertige Teile bestellt. Spezialteile jedoch werden eigens dafür geplant und hergestellt.

### 1.7.1 Fertigung der Akkus sowie Microcontrollerhalterung

Diese Spezialteile werden geplant und die Maße in den Werkstättenzeichnungen beschrieben. Für den Prototypenbau sollen diese Bauteile aus PLA Kunststoff mithilfe eines 3D-Druckers gedruckt werden. In der Maßenfertigung von 10.000 Stück wird eine Spritzgussform konstruiert. Diese Spritzgussteile werden auf Maß gefertigt und sollen unter minimalster Nachbearbeitung in den iWalk eingebaut werden.

### 1.7.2 Fertigung des Rohrstoppel

Dieses Spezialbauteil wurde ebenfalls geplant. Für den Prototyp soll ebenfalls ein 3D-Druck erstellt werden. Jedoch wird dieses Bauteil in der Maßenfertigung von 10.000 Stück mithilfe von CNC-Maschinen aus Aluminium hergestellt. Diese werden ebenfalls auf Maß gefertigt. Die folgenden Maße sind in der Werkstättenzeichnung ersichtlich.

## **2 Teil B - Informatik**

## 2.1 Anforderungen

Die Aufgabe der Android-Appliktion ist es, die Daten von den Sensoren mithilfe des Microkontrollers zu empfangen und auszuwerten. Die App soll die empfangenen Daten in den jeweiligen Diagrammen anzeigen. Ein benutzerfreundlicher Aufbau wird vorausgesetzt.

### 2.1.1 Entwicklungsumgebung

Android Studio<sup>1</sup> ist die meist verbreitete und unterstützte IDE für Android-Programmierungen. Android Studio basiert auf der IntelliJ IDEA Community Edition. Die Entwicklungsumgebung ist so gestaltet, dass Android-Applikationen sehr einfach zu erstellen sind. Diese IDE wird auch im Schulunterricht benutzt, was nun wiederum einen weiteren Vorteil bietet. Android Studio ist die offizielle Entwicklungsumgebung. Jährlich werden zwei bis drei neue Versionen veröffentlicht, das heißt, dass immer wieder neue Bibliotheken hinzugefügt werden, welche Neuerungen mit sich bringen.



Abbildung 2.1: Android Studio

#### 2.1.1.1 Auswahl der verwendeten IDE

In der IDE lässt sich mit zwei Programmiersprachen programmieren. Java und Kotlin. Die primäre Programmiersprache dieser IDE ist Java. Bei unserer Dipomarbeit wurde in Java programmiert, weil wir diese Programmiersprache bereits im Unterricht erlernen konnten. Dieser Vorteil war ein ausschlaggebender Punkt für die Entscheidung. Weiters gibt es eine von Google unterstützte Website, in der über fast alle Themen nachgelesen werden kann, die für eine App-Entwicklung notwendig sind. Außerdem sind dort alle in Android Studio verfügbaren Klassen beschrieben. Durch die vorhin beschriebenen Tatsachen fiel mir die Wahl für diese IDE leicht.

---

<sup>1</sup>vlg. Android Studio, (2021)

## 2.1.2 Android

Android<sup>2</sup> ist ein Betriebssystem für mobile Geräte wie Smartphones, Fernseher und Netbooks. Der Marktanteil lag im Mai 2019 bei 75 Prozent. Android ist das meistverbreitete Betriebssystem für Smartphones. 2008 erschien die erste Version von Android mit dem Namen "Base". 2021 erschien die 12. Version von Android.

Die folgende Tabelle soll einen Überblick über die verschiedenen Versionen liefern:

<b>Version</b>	<b>Codename</b>	<b>Erscheinungsdatum</b>	<b>API</b>
1.0	"Base"	23.09.2008	1
1.1	"Base 1.1"	09.02.2009	2
1.5	"Cupcake"	27.04.2009	3
1.6	"Donut"	15.09.2009	4
2.0.x / 2.1	"Éclair"	26.10.2009	5,6,7
2.2.x	"Froyo (Frozen Yogurt)"	20.05.2010	8
2.3.x	"Gingerbread"	06.12.2010	9,10
3.x.x	"Honeycomb"	22.02.2011	11,12,13
4.0.x	"Ice Cream Sandwich"	18.10.2011	14,15
4.1.x / 4.2.x / 4.3.x	"Jelly Bean"	09.07.2012	16,17,18
4.4.x	"Kitkat"	31.10.2013	19,20
5.0.x / 5.1.x	"Lollipop"	12.11.2014	21,22
6.0.x	"Marshmallow"	05.10.2015	23
7.0.x / 7.1.x	"Nougat"	22.08.2016	24,25
8.0 / 8.1	"Öreo"	21.08.2017	26,27
9	"Pie"	06.08.2018	28
10	"— / („Queen Cake“)"	03.09.2019	29
11	"— / („Red Velvet Cake“)"	08.09.2020	30
12	Android 12"	04.10.2021	31

Tabelle 2.1: Android Versionen

---

<sup>2</sup>vgl. Android Studio, (2021)

## 2.2 Kommunikation

### 2.2.1 Theoretischer Hintergrund

Es gibt die Möglichkeit über Wlan oder Bluetooth mit der Android-App zu kommunizieren. Mit der Wlan-Technologie(Wireless Local Area Network) wird ein Wlan-Modul (Wlan shield) für den Arduino Nano benötigt. Die Bluetooth-Technologie bedarf ebenso ein Bluetoothmodul.

#### 2.2.1.1 Kommunikation über Wlan

Die Kommunikation über Wlan basiert auf dem Server-Client Prinzip. Der Server ist eine Software, welche einen Dienst zur Verfügung stellt. Der Client ist ebenso eine Software, welche den vom Server zur Verfügung gestellten Dienst nutzt. Der Arduino ist in diesem Fall der Server und die App der Client. Der Client wird den Arduino (Server) über die IP Adresse und Portnummer des Wifi-Moduls erreichen.

#### 2.2.1.2 Kommunikation über Bluetooth

Für die Übertragung von Daten in hoher Qualität, wird das BR/EDR (Basic Rate/Enhanced Data Rate) Verfahren benötigt. Das BLE (Bluetooth Low Energy) ist dann sinnvoll anzuwenden, wenn Daten in niedriger Qualität übertragen werden.

### 2.2.2 AsyncTask

Der AsyncTask ermöglicht die Ausführung der Anwendung im Hintergrund und synchronisiert diese danach mit dem GUI-Thread.

Die Klasse wird mit der Methode `new GetDataWorker().execute();` ausgeführt. In unserem Fall bekam die Klasse den Namen GetDataWorker. Der GetDataWorker führt alles in der `doInBackground()`-Methode innerhalb eines Hintergrund-Threads aus. Dieser Thread hat keinen Zugriff auf die GUI, in welcher alle Ansichten vorhanden sind.

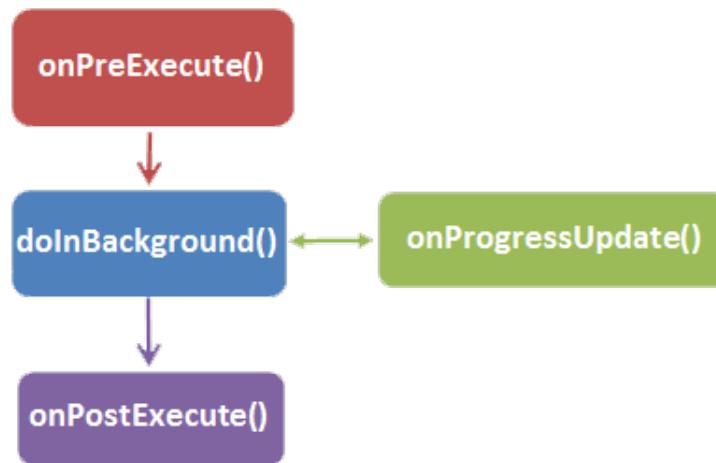


Abbildung 2.2: Asynctask

Das Flussdiagramm in der Abbildung 2.2 beschreibt die Funktion der AsyncTask-Klasse. Die Methoden, die in einer AsyncTask-Klasse zur Anwendung kommen, sind in der folgenden Auflistung beschrieben:

- **onPreExecute():**

Diese Methode wird verwendet, um vor der Ausführung des AsyncTask Werte übergeben zu können.

- **doInBackground():**

Die `doInBackground()`-Methode wird im Hintergrund ausgeführt. Diese Methode kann die Ergebnisse an den GUI-Thread senden. Diese hat einen Rückgabewert, der für die Statusmeldung zuständig ist, ob der Hintergrundprozess fertig ist.

- **onPostExecute():**

Die Methode `onPostExecute()` wird aufgerufen, wenn die `doInBackground()`-Methode durchgeführt wurde. Das Ergebnis der `doInBackground()`-Methode wird diese Methode übergeben.

- **onProgressUpdate():**

Diese Methode wird dazu verwendet die GUI zu aktualisieren. Die `onProgressUpdate()`-Methode erhält die Aktualisierungen der `doInBackground()`-Methode.

Der folgende Code veranschaulicht die Anwendung der AsyncTask-Klasse in der SchritteActivity.java Datei.

```
1 public class GetDataWorker extends AsyncTask<String,Void,RingSpeicher>
2 {
3     @Override
4     protected RingSpeicher doInBackground(String... params)
5     {
6         final Comm fakeComm = new FakeComm();
7         try
8         {
9             fakeComm.connect();
10        }
11        catch (Exception ex)
12        {
13            ex.printStackTrace();
14        }
15        final RingSpeicher speicher = fakeComm.getSchritte();
16        fakeComm.disconnect();
17        return speicher;
18    }
19 }
```

Listing 2.1: Anwendung Asynctask

## 2.2.3 Bluetoothverbindung

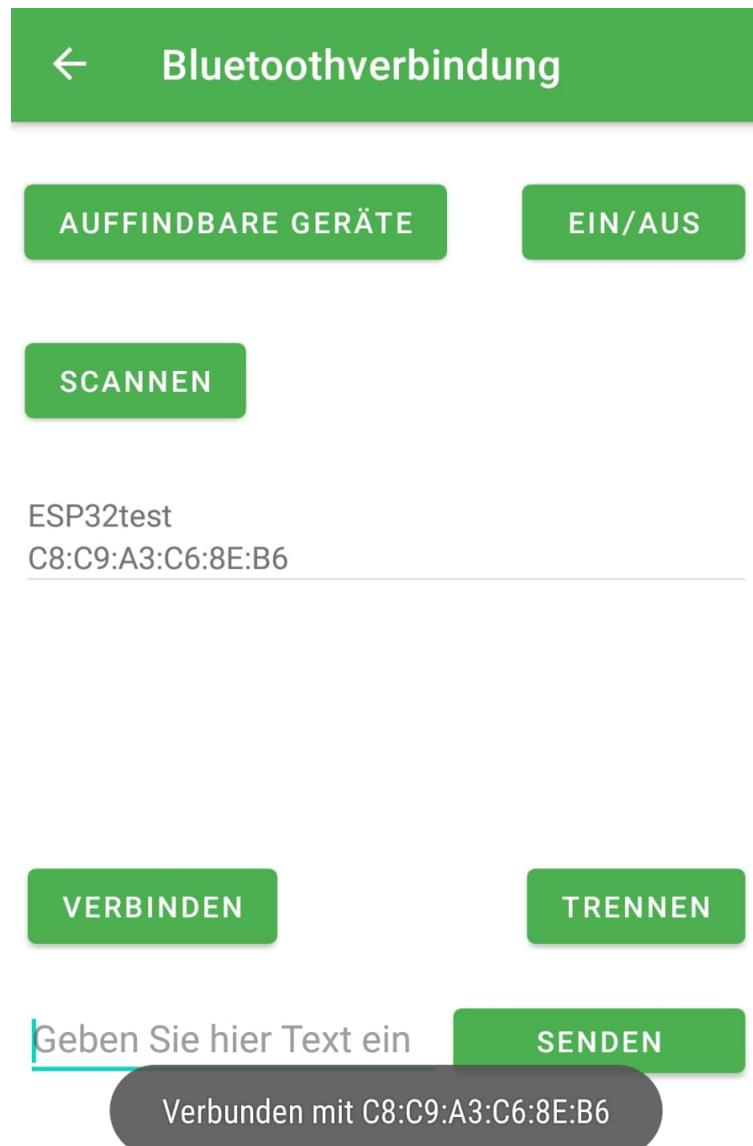


Abbildung 2.3: Activity Bluetoothverbindung

Um eine Bluetoothverbindung herzustellen, muss der Benutzer wie in Abbildung 2.3 dargestellt, den Button „Ein/Aus“ drücken. Damit öffnet sich ein Dialogfenster vom Smartphone. Nach Zulassen der Aktivierung des Bluetooth, muss der Benutzer auf den Button „Scannen“ tippen, damit alle verfügbaren Bluetoothgeräte in der Umgebung gescannt werden können. Die gefundenen, aktiven Bluetoothgeräte werden darunter in einer Liste angezeigt. Nun ist das gewünschte Bluetoothgerät auszuwählen und darauffolgend ist der Tipp auf den „Verbinden“-Button zu tätigen. Zum Schluss kommt die Statusmeldung, ob die Verbindung zustande gekommen ist.

## 2.3 Design der Android App

### 2.3.1 Erste Designvorschläge

#### 2.3.1.1 Ladebildschirm



Abbildung 2.4: Logo

Der Ladebildschirm, wie in Abbildung 2.4 dargestellt, beinhaltet das iWalk-Logo auf einem weißen Hintergrund. Am unteren Bildschirmrand ist zu sehen von wem die Android-App entwickelt wurde. Die Abkürzung „mech17“ steht für Abteilung Mechatronik Jahrgang 2017.

### 2.3.1.2 Startbildschirm



Abbildung 2.5: Startbildschirm

Nach 2000 ms ist der Startbildschirm zu sehen. In Abbildung 2.5 sind die einzelnen Funktionen der Menüpunkte abgebildet. Durch Tippen auf „Schritte“ öffnet sich eine neue Activity, in welcher ein Balkendiagramm angezeigt wird. Durch Tippen auf „Weitere Details“ im Menüpunkt „Stürze“ wird eine separate Activity gestartet, in welcher genauso ein Balkendiagramm erscheint, das die Anzahl der Stürze innerhalb der letzten 12 Monate anzeigt. Wenn man auf „Weitere Details“ im Menüpunkt „Belastungsanalyse“ tippt, gelangt man auf eine neue Activity, in welcher ein Liniendiagramm erscheint. Dort wird der Kraftverlauf innerhalb der letzten 24 Stunden angezeigt. Wenn man auf „Notfallnummer hinzufügen“ tippt, erscheint ein Dialogfenster, in welchem der neue Kontakt hinzugefügt werden kann.

### 2.3.1.3 Schritte Activity



Abbildung 2.6: Schritte

In der Abbildung 2.6 werden die Schritte nochmals in Form eines Balkendiagramms angezeigt. Das Balkendiagramm veranschaulicht, zu welchem Zeitpunkt wie viele Schritte mit der Krücke gegangen worden sind. Auf der X-Achse wird die Zeit in Tagen aufgetragen. Auf der Y-Achse werden die gegangenen Schritte verzeichnet.

### 2.3.1.4 Stürze Activity



Abbildung 2.7: Stürze Activity

Das Balkendiagramm in Abbildung 2.7 zeigt, wie viele Stürze es innerhalb der letzten 12 Monaten gegeben hat. Die einzelnen Balken zeigen die Anzahl der Stürze in einem Monat. Auf der X-Achse wird die Zeit in Monate aufgetragen. Auf der Y-Achse wird die Anzahl der Stürzen verzeichnet.

### 2.3.1.5 Belastungsanalyse Activity

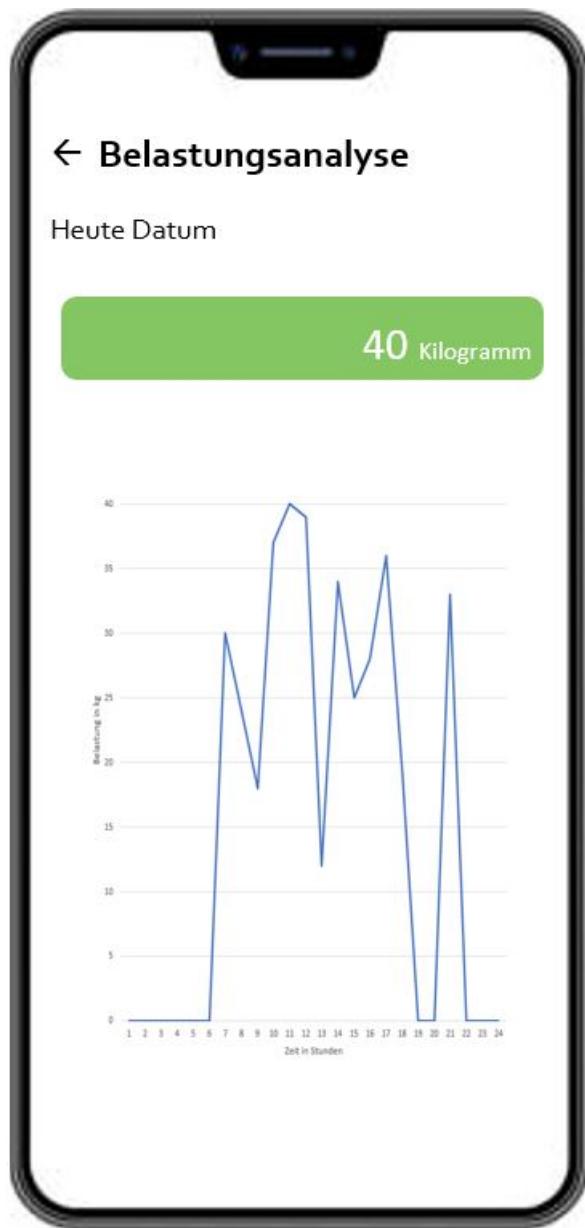


Abbildung 2.8: Belastungsanalyse Activity

Das Liniendiagramm in Abbildung 2.8 zeigt den Kraftverlauf innerhalb der letzten 24 Stunden. Diese Funktion soll veranschaulichen, ob der Anwender die Krücke richtig belastet. Auf der Y-Achse wird die aufgewendete Kraft in kg verzeichnet und auf der X-Achse wird die Zeit in Stunden aufgetragen.

### 2.3.1.6 Notfallkontakt



Abbildung 2.9: Notfallkontakt Dialogfenster

Wie in Abbildung 2.9 dargestellt erscheint beim Tippen auf „Notfallkontakt hinzufügen“ ein Dialogfenster, in welchem die Möglichkeit besteht einen Notfallkontakt hinzuzufügen. Die App soll automatisch den Notfallkonakt anrufen, wenn die Krücke das Signal meldet, dass die Person gestürzt ist.

## 2.3.2 Finales Design der App

### 2.3.2.1 Ladebildschirm



Abbildung 2.10: Screenshot Ladebildschirm

Bei der Erstellung des Ladebildschirms gab es ein Problem. Die App stürzte immer ab. Die Auflösung des iWalk-Logos wurde zu groß gewählt. Das Logo musste komprimiert werden. Nach erfolgreicher Komprimierung des iWalk-Logos, auf die Größe 1920 Pixel x 595 Pixel, wie in Abbildung 2.10 dargestellt, startete die Android-App problemlos.

### 2.3.2.2 Startbildschirm

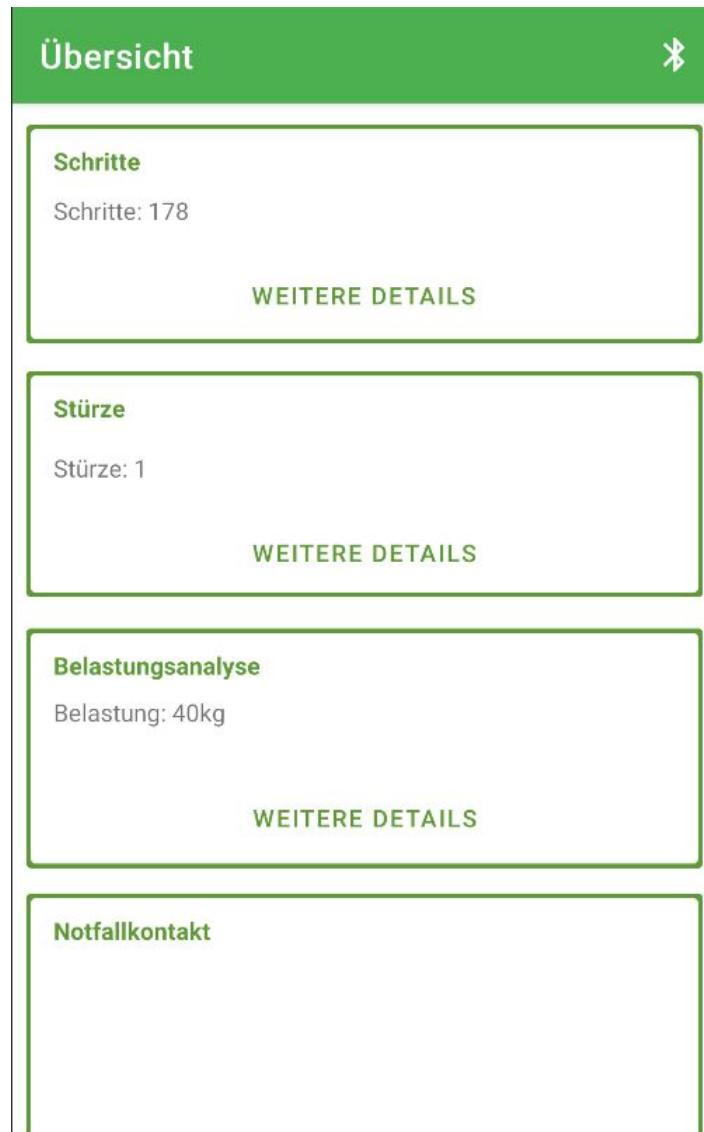


Abbildung 2.11: Screenshot Startbildschirm

Abbildung 2.11 zeigt den Aufbau des Startbildschirms der Smartphone-App:

Vier „Linearlayouts“, worin sich jeweils zwei Textfelder und ein Funktionsbutton befinden, bilden das Grundgerüst des Startbildschirms.

Das Design, der an den Ecken abgerundeten „Linearlayouts“, gefiel mir besonders gut.

Das Label(Bezeichnung) der Activity bekam den Namen „Übersicht“. Auf der rechten Seite der Actionbar befindet sich das Bluetoothsymbol. Durch Tippen auf dieses Symbol öffnet sich die Activity zur Herstellung der Bluetoothverbindung.

### 2.3.2.3 Schritte Activity

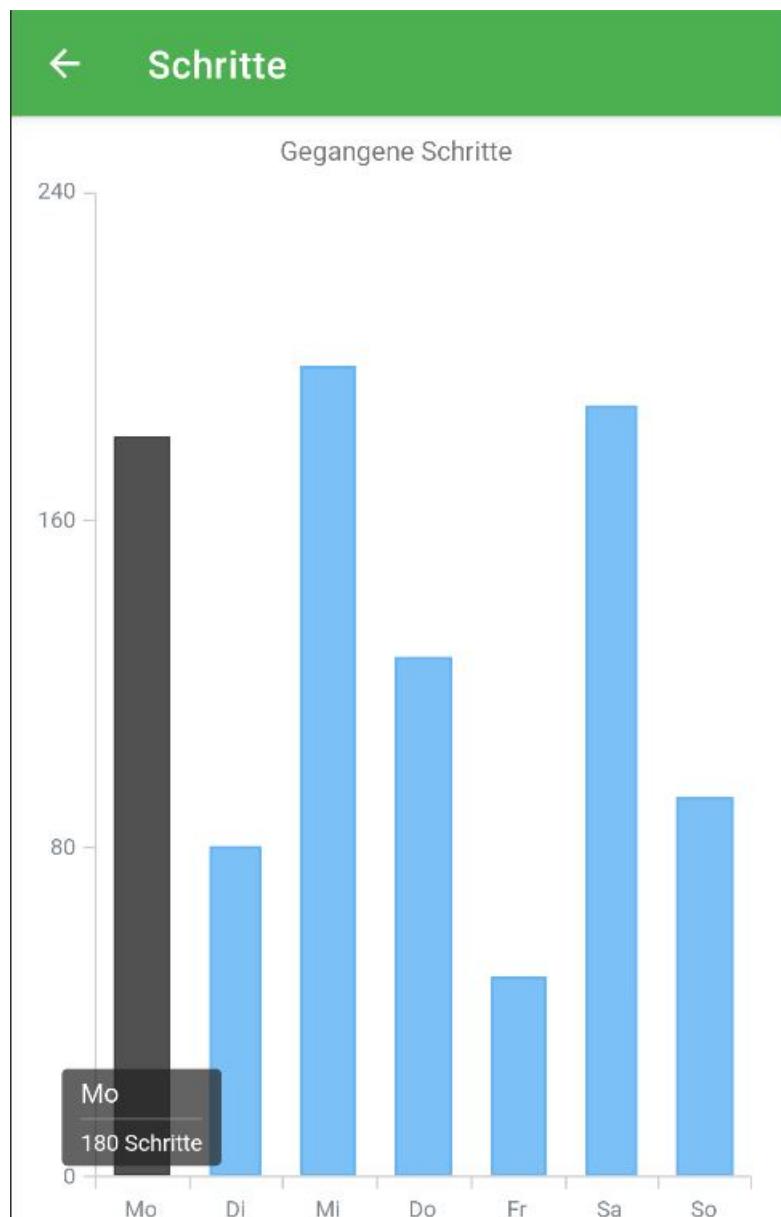


Abbildung 2.12: Screenshot Schritte Activity

Der Benutzer hat die Möglichkeit jeden einzelnen Balken des Diagramms anzutippen. Durch Tippen auf einen der Balken des in Abbildung 2.12 dargestellten Balkendiagramms erscheint über den angetippten Balken ein kleines Informationsfenster, welches den entsprechenden Wochentag und die Anzahl der gegangenen Schritte anzeigt.

### 2.3.2.4 Stürze Activity

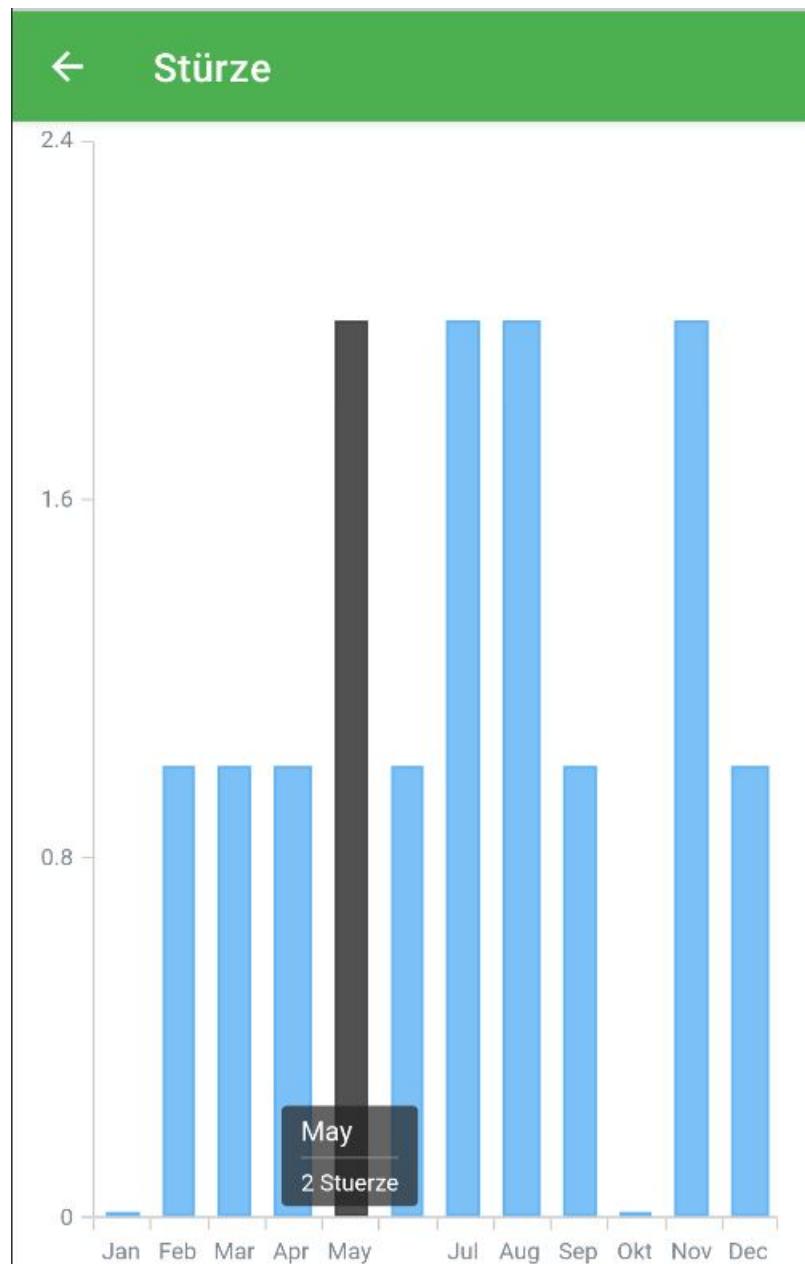


Abbildung 2.13: Screenshot Stürze Activity

Durch Tippen auf einen einzelnen Balken des Sturzdiagramms, wie in Abbildung 2.13 dargestellt, wird dieser grauschwarz und es erscheint über den angetippten Balken ein kleines Informationsfenster. In diesem wird der jeweilige Monat und die Anzahl der Stürze angezeigt.

### 2.3.2.5 Belastungsanalyse Activity

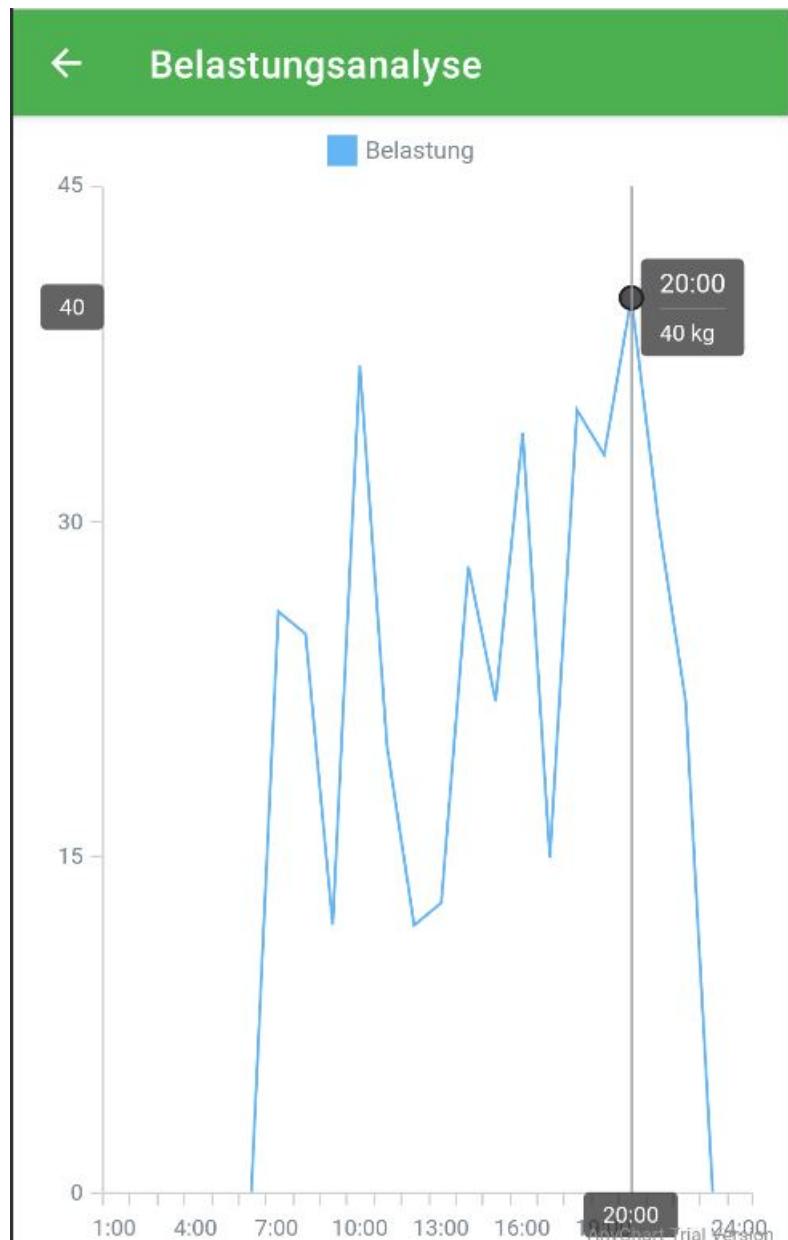


Abbildung 2.14: Screenshot Belastungsanalyse Activity

Wie in Abbildung 2.14 dargestellt besteht die Möglichkeit einen beliebigen Punkt auf der Linie des Liniendiagramms anzutippen. Danach erscheint ein kleines Informationsfenster, in welchem die Belastung der Krücke zur jeweiligen Uhrzeit angezeigt wird. Zusätzlich erscheinen an den Achsen kleine Informationsfenster, die die am Liniendiagramm angetippte Uhrzeit und Krückenbelastung anzeigen.

### 2.3.3 Sprachen

Die Android-App bietet die Möglichkeit der Mehrsprachigkeit. Es werden die Sprachen Deutsch und Englisch unterstützt. Dafür werden xml-Dateien im Entwicklungstool Android Studio angelegt, wo alle benötigten sprachenspezifischen Wörter, gespeichert sind. Im Folgenden sind die zwei xml-Dateien zu sehen. Die deutsche sowie die englische Version.

#### Deutsche Version:

```
1 <resources>
2
3     <string name="app_name">iWalk</string>
4
5     <string name="tfoverview">bersicht</string>
6
7     <string name="tfsteps">Schritte</string>
8
9     <string name="tfstepsfromsteps">Schritte / Schritte</string>
10
11    <string name="btsteps">Schritte</string>
12
13    <string name="btfall">Weitere Details</string>
14
15    <string name="tffall">Strze</string>
16
17    <string name="tfnumberfall">Anzahl der Strze</string>
18
19    <string name="tfload">Belastungsanalyse</string>
20
21    <string name="tfforce">Belastung</string>
22
23    <string name="tfmech17">from mech 17</string>
24
25    <string name="options">Einstellungen</string>
26
27    <string name="tfemergencycontact">Notfallkontakt</string>
28
29    <string name="btemergencycontact">Notfallkontakt hinzufgen</string>
30
31    <string name="tfname">Name</string>
32
33    <string name="tfnumber">Telefonnummer</string>
34
35    <string name="Bluetooth">Bluetoothverbindung</string>
36
37    <string name="btcancel">Abbrechen</string>
38
39    <string name="btok">Ok</string>
40
41    <string name="btONOF">Ein/Aus</string>
42
43    <string name="btconnect">Verbinden</string>
44
45    <string name="btdisconnect">Trennen</string>
46
47    <string name="btsend">Senden</string>
48
49    <string name="btscan">Scannen</string>
```

```

26   <string name="btdiscoverable">Auffindbare Gerte</string>
27   <string name="tfttext">Geben Sie hier Text ein</string>
28   <string name="tfgoingsteps">Gegangene Schritte</string>
29 </resources>
```

Listing 2.2: Deutsche Version

**Englische Version:**

```

1 <resources>
2
3   <string name="app_name">iWalk</string>
4
5   <string name="tfoverview">overview</string>
6
7   <string name="tfsteps">steps</string>
8
9   <string name="tfstepsfromsteps">steps from steps</string>
10
11  <string name="btsteps">steps</string>
12
13  <string name="btfall">click for details</string>
14
15  <string name="tffall">falls</string>
16
17  <string name="tfnumberfall">number of falls</string>
18
19  <string name="tfload">stress analysis</string>
20
21  <string name="tfforce">load</string>
22
23  <string name="tfmech17">from mech 17</string>
24
25  <string name="options">options</string>
26
27  <string name="tfemergencycontact">emergency contact</string>
28
29  <string name="btemergencycontact">add emergency contact</string>
30
31  <string name="tfname">name</string>
32
33  <string name="tfnumber">phonenumerber</string>
34
35  <string name="Bluetooth">bluetooth connection</string>
36
37  <string name="btcancel">cancel</string>
38
39  <string name="btok">ok</string>
40
41  <string name="btONOF">On/Off</string>
42
43  <string name="btconnect">connect</string>
44
45  <string name="btdisconnect">disconnect</string>
46
47  <string name="btsend">send request</string>
48
49  <string name="btscan">scannen</string>
```

```

26     <string name="btdiscoverable">enable discoverable</string>
27     <string name="tftext">enter text here</string>
28     <string name="tfgoingsteps">steps taken</string>
29 </resources>

```

Listing 2.3: Englische Version

### 2.3.4 Farben

Basierend auf dem Logo wurden die Farben in den Grüntönen des Logos gewählt. Der Hintergrund für die Android-App ist weiß mit grünem Rahmen für die jeweiligen Funktionen. Diese Variante hat ein sehr helles Design wie auf den Abbildungen des Punktes 3.2 zu sehen ist.

### 2.3.5 Logo

Das Logo soll beim Start der App gut zu sehen sein. Dementsprechend erscheint beim Starten der App das iWalk Logo.

### 2.3.6 Icon

Das Icon der App ist das iWalk Logo. Zuerst ist eine Bilddatei mit einer korrekt darstellbaren Auflösung in den Ordner `res/drawable` einzufügen. In der `AndroidManifest.xml`-Datei muss folgendener Befehl unter `application` eingefügt werden, um das Icon hinzuzufügen:

```
1     android:icon="@drawable/iwalk_logo_1920"
```

Listing 2.4: App Icon hinzufügen

### 2.3.7 Options Menu

Im „Options Menu“ soll ein Bluetoothsymbol zu sehen sein. Durch Tippen auf dieses Symbol wird eine neue Activity gestartet, um die Bluetoothverbindung herzustellen. Zuerst wird das Icon ausgesucht, welches unter dem Ordner `res/drawable` mit einem neuen Vector Asset erstellt werden kann. Danach muss ein neues `Android resource directory` mit dem Namen

„Menu“ erzeugt werden. In diesem Menu resource directory wird eine Datei angelegt, in welcher die Menüpunkte erzeugt werden können. In diesem Fall wurde das Bluetoothicon ausgewählt. In der MainActivity müssen folgende Klassen aufgerufen werden:

```

1 public boolean onCreateOptionsMenu(Menu menu)
2 {
3     MenuInflater inflater = getMenuInflater();
4     inflater.inflate(R.menu.menubar,menu);
5     return true;
6 }
```

Listing 2.5: Options Menü erstellen

Damit sich beim Tippen auf das „MenuItem“ eine neue Activity öffnet, muss folgende Klasse aufgerufen werden:

```

1 public boolean onOptionsItemSelected(@NonNull MenuItem item)
2 {
3     int id = item.getItemId();
4     if (id == R.id.bluetoothtest2)
5     {
6         Intent intent = new Intent(HomeScreen.this, BluetoothTest2.class);
7         startActivity(intent);
8     }
9     return super.onOptionsItemSelected(item);
10 }
```

Listing 2.6: Aufruf der neuen Activity

Um eine neue Activity aufzurufen, muss zuerst ein Intent erzeugt werden. Nun wird die Activity, in welcher man sich gerade befindet, mit dem Befehl `.this` aufgerufen. Nach dem Beistrich wird die Activity, in welche man möchte, mit dem Befehl `.class` aufgerufen. Zum Schluss wird noch der Befehl `startActivity(intent)` aufgerufen, welcher die gewünschte Activity startet.

*Pro Activity muss eine xml-Datei im Verzeichnis res/layout erstellt werden.<sup>3</sup>*

---

<sup>3</sup>vgl. Android Grundlagen und Programmierung, (2022)

### 2.3.8 LinearLayout mit Radius

Um ein LinearLayout mit abgerundeten Ecken erstellen zu können, wird im Ordner `res/drawable` eine neue xml-Datei angelegt. Der folgende Code wird dort viermal hintereinander geschrieben. Bei jeder Wiederholung kommt die Angabe der Linienstärke für oben, unten, rechts und links hinzu.

```
1 <item>
2     <shape android:shape="rectangle">
3         <solid android:color="@color/schriftgrn"/>
4         <corners android:radius="3dp"/>
5     </shape>
6 </item>
```

Listing 2.7: LinearLayout erstellen

### 2.3.9 AndroidManifest.xml-Datei

Jedes Android-App Projekt besitzt eine `AndroidManifest.xml` Datei mit genau dieser Bezeichnung, welche sich im Projektquellsatz befinden muss. Dort sind viele verschiedene Angaben über die App hinterlegt, welche das Android-System braucht, um die Applikation installieren und ausführen zu können. Diese Datei wird automatisch erstellt, wenn die App in Android Studio gebaut wird.

`<application>` tag ist das Unterelement der Mainfestdatei und sie beinhaltet bestimmte deklarierte Anwendungskomponenten.

- icon
- allowBackup
- label
- theme

`<intent-filter>` tag ist ein Element der Activity. Die Art des Intent, auf die die Android-Komponenten reagieren können, ist dort beschrieben. Sie werden verwendet, um den App-Komponenten Absichten zuzuweisen.

Ein fester Bestandteil der Manifestdatei sind Symbole und Bezeichnungen. Das Symbol charakterisiert das App-Icon, welches auf dem Gerät nach der Installation angezeigt wird. Das `label` beschreibt den Namen der Activity, welches in der „action bar“ angezeigt wird. Das `icon` und das `label` wird im `<application>` tag definiert.

```
1 <application>
2     android:label="@string/app_name"
3     android:icon="@drawable/iwalk_logo_1920"
4 </application>
```

Listing 2.8: AndroidManifest-Datei

## 2.4 Ladebildschirm

Der Ladebildschirm erscheint nur beim Starten der App.

Um einen sogenannten Loadingscreen erstellen zu können, muss der folgende Code in der `HomeScreen.java` Datei implementiert werden.

```
1 public class LoadingScreen extends AppCompatActivity
2 {
3     private static int SPLASH_TIME_OUT = 2000;
4
5     @Override
6
7     protected void onCreate(Bundle savedInstanceState)
8     {
9
10         super.onCreate(savedInstanceState);
11
12         setContentView(R.layout.activity_loading_screen);
13
14         getSupportActionBar().hide();
15
16         new Handler().postDelayed(new Runnable()
17
18             {
19
20                 @Override
21
22                 public void run()
23
24                 {
25
26                     Intent homeIntent = new Intent(LoadingScreen.this,
27
28                         HomeScreen.class);
29
30                 }
31
32             }
33
34         , SPLASH_TIME_OUT);
35     }
36 }
```

```

16         startActivity(homeIntent);
17
18     }
19 }, SPLASH_TIME_OUT);
20 }
21 }
```

Listing 2.9: Programmcode Ladebildschirm

Das SPLASH-TIME-OUT dient dazu, wie lange der Loadingscreen zu sehen ist. Folgende Codezeile wird verwendet, um vom Ladebildschirm den Startbildschirm aufzurufen. Folgende Codezeile wird verwendet, um vom Ladebildschirm den Startbildschirm aufzurufen.

```
1 Intent homeIntent = new Intent(LoadingScreen.this, HomeScreen.class);
```

Listing 2.10: Aufruf einer neuen Activity

Ein Intent dient dazu, eine Aktion auf dem Bildschirm auszuführen. Dieser wird verwendet, um Aktivitäten zu starten. Der Ladebildschirm ist die Main-Activity, welche zuerst aufgerufen wird.

## 2.5 Diagramme

### 2.5.1 Einfügen der Bibliotheken

Für Diagramme<sup>4</sup> gibt es verschiedenste Designvorlagen. Ich entschied mich für die Designvorlage von „AnyChart“. Um die Designvorlage von AnyChart verwenden zu können, muss in die build.gradle(Module; iWalk.app)-Datei folgende Implementaion hinzugefügt werden:

```
1 implementation 'com.github.AnyChart:AnyChart-Android:1.1.2'
```

Listing 2.11: Implementation der Bibliotheken

Folgender Code ist der root build.gradle Datei am Ende des Repositories hinzuzufügen, um die Bibliotheken der gewählten Designvorlage von Any Chart verwenden zu können. (Der Code muss unter allprojects und nicht unter buildscript hinzugefügt werden)

<sup>4</sup>vgl. Diagramme, (2022)

```

1 allprojects
2 {
3     repositories
4     {
5         maven { url 'https://jitpack.io' }
6         google()
7         mavenCentral()
8     }
9 }
```

Listing 2.12: Hinzufügen der jitpack.io Datei

## 2.5.2 Balkendiagramm

Nun ist es erforderlich in der xml-Datei der gewünschten Activity den vorgefertigten Code einzufügen. Zuerst ist eine TextView in der Activity erforderlich, da dort das Balkendiagramm erscheint.

```

1 <com.anychart.AnyChartView
2     android:id="@+id/textView2"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent">
5 </com.anychart.AnyChartView>
```

Listing 2.13: TextView erstellen

## 2.5.3 Liniediagramm

Ein Liniendiagramm einzufügen funktioniert genau gleich, wie das Einfügen des Balkendiagramms. Es ist nur der vorgefertigte Code für Liniendiagramme einzubinden. In der gewünschten Activity ist ebenso eine TextView erforderlich, da dort dann das Liniendiagramm erscheint.

## 2.6 Aufruf des Dialogfensters

Ein Aufruf des Dialogfensters sieht folgendermaßen aus:

```
1 public void openDialog()
2 {
3     Dialog dialog = new Dialog();
4     dialog.show(getSupportFragmentManager(), "dialog");
5 }
```

Listing 2.14: Aufruf des Dialogfensters

Ein Dialogfenster ist ein kleineres Fenster, das den Benutzer auffordert Informationen einzugeben oder eine Entscheidung zu treffen.

In unserem Fall benötigen wir ein Dialogfenster, um die vom Benutzer eingegebenen Informationen an die Home-Activity (Startbildschirm) zu senden.

Es handelt sich um ein benutzerdefiniertes Layout in einem Dialogfeld. Zuerst ist das Layout zu erstellen und danach ist es zu einem Alert-Dialog mit dem Befehl `setView()` hinzuzufügen. Der Befehl `setView()` wird benötigt, um das AlertDialog.Builder-Objekt aufzurufen.

Um einen passenden Namen für den Titel und die einzelnen Schaltflächen vergeben zu können, ist es vorteilhaft AlertDialog.Builder-Methoden zu verwenden.

Die Abbildung 2.15 zeigt das Dialogfenster für das Hinzufügen des Notfallkontakts.

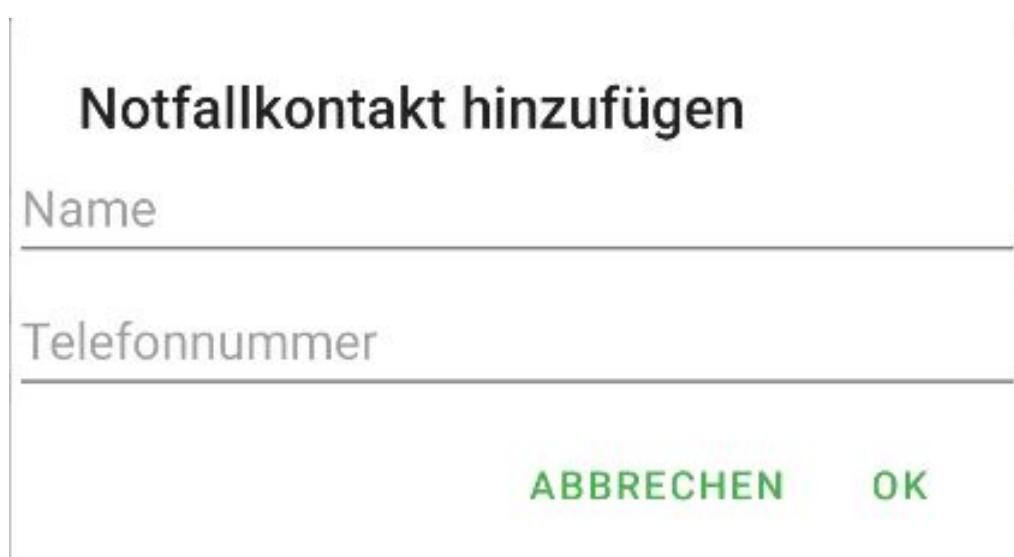


Abbildung 2.15: Screenshot Dialogfenster

## 2.6.1 Positiv Button

Wenn der Benutzer, wie in Abbildung 2.15 dargestellt, auf „OK“ tippt wird der Notfallkontakt hinzugefügt. Im folgenden ist der Code für den PositivButton zu sehen.

```
1 builder.setPositiveButton(R.string.btok, new DialogInterface.OnClickLister() {  
2     @Override  
3     public void onClick(DialogInterface dialogInterface, int i) {  
4         String name = editTextName.getText().toString();  
5         String nummer = editTextTelefonnummer.getText().toString();  
6         listener.applyTexts(name, nummer);  
7     }  
8 } );
```

Listing 2.15: Positive-Button

## 2.6.2 Negativ Button

Wenn der Benutzer den Vorgang abbrechen möchte, genügt es auf „Abbrechen“ zu tippen. Im folgenden ist der Code für den NegativButton dargestellt.

```
1 builder.setNegativeButton(R.string.btcancel, new DialogInterface.OnClickListener() {  
2     @Override  
3     public void onClick(DialogInterface dialogInterface, int i) {  
4     }  
5 } );
```

Listing 2.16: Negative-Button

## 2.6.3 Löschen der Kontakte

Der Kontakt soll in einer ListView gespeichert werden. Das bedeutet, dass der Kontakt nach dem Schließen und neuerlichen Starten der App gespeichert bleibt. Der Kontakt kann durch Tippen auf den Mitskübel gelöscht werden. Wenn der Benutzer den Kontakt löscht, soll sich ein Dialogfenster öffnen, welches den Benutzer fragt „Möchten sie den Kontakt wirklich löschen“. Der Benutzer hat die Auswahl zwischen „Ja“ und „Nein“. Wenn der „Ja“ Button angetippt wird, ist der Kontakt gelöscht. Bei „Nein“ verschwindet das Dialogfenster und der Kontakt wurde nicht gelöscht.

## 2.7 Entwicklungsprozess, Selbtkritische Analyse, Resümee

### 2.7.1 Entwicklungsprozess

Am Anfang wurden alle Designparameter im Programm Powerpoint festgelegt. Unser Teamleiter Georg Kaufmann hatte eine ungefähre Vorstellung, wie das Design der App aussehen sollte. Nach Absprache mit den anderen Teammitgliedern hatte ich mit den ersten Entwürfen der Skizzen im Programm Powerpoint begonnen. Schon der erste Entwurf der Skizzen erhielt große Zustimmung, daher musste ich fast nichts abändern. Zu Beginn der Entwicklung der Android-App gab es keine Probleme mit der Umsetzung. Im Laufe der Zeit gab es ein Problem mit der Erstellung des Balkendiagramms. Es gab einen Versionsfehler von Android Studio. Dieser hat das hinzufügen der vorgefertigten Bibliotheken von AnyChart nicht ermöglicht.

### 2.7.2 Selbtkritische Analyse

Allgemein ist zu sagen, dass die Android App die gestellten Anforderungen erfüllt. Mit dem Endergebnis des Projekts bin ich zufrieden. Es wurde zu viel Zeit für die GUI geopfert. Es wäre sinnvoller gewesen, mehr Zeit in die Funktionen der Android App zu investieren.

### 2.7.3 Resümee

Zusammenfassend ist zu sagen, dass alle Anforderungen der App erfüllt werden konnten. Mein Wissen konnte ich erweitern und neue Fähigkeiten eine Android-App zu programmieren erlernte. Gegen Ende der Diplomarbeit ist eine sehr gute Basis vorhanden, um das Projekt weiterzuentwickeln.

Die Arbeit konnte ich ohne große Hilfeleistungen umsetzen. Alle festgelegten Meilensteine wurden von allen Teammitgliedern eingehalten. Durch die regelmäßige Kommunikation war das Arbeiten mit dem Projektteam sehr angenehm. Es wurde viel kommuniziert, obwohl die einzelnen Teilbereiche unabhängig voneinander realisiert werden konnten. Der Projektleiter motivierte die Teammitglieder häufig und hatte immer einen guten Überblick über den Arbeitsfortschritt jedes einzelnen zu jedem festgelegten Meilenstein.

## **3 Teil C - Sensorik**

## 3.1 Aufgabenstellung

Die zu erarbeitende Aufgabenstellung ist es, einen Schrittzähler und eine Sturzerkennung für eine Krücke zu entwickeln. Weiters sollen die mittels eines Sensors gesammelten Daten per Bluetooth an ein Smartphone gesendet werden. Der Schrittzähler soll ähnlich wie bei einem Smartphone oder einer Smartwatch lediglich zur Überwachung der täglichen Bewegung dienen. Die Sturzerkennung soll raschere Hilfe ermöglichen.

## 3.2 Funktion

### 3.2.1 Schrittzähler

Bewegung ist in jedem Alter wichtig. Vor allem an der frischen Luft. Deshalb ist im Fitnessbereich ein Schrittzähler nicht mehr wegzudenken, denn es erleichtert maßgeblich das Aufzeichnen von Bewegung. Ein in einer Gehhilfe eingebauter Schrittzähler soll somit zur Genesung des Patienten beitragen.

#### 3.2.1.1 Theoretischer Hintergrund

Grundsätzlich funktionieren moderne Schrittzähler immer nach dem gleichen Prinzip. Dabei ist es egal, ob es sich um einen Fitnesstracker, eine Smartwatch oder ein Smartphone handelt. Die verwendeten Beschleunigungssensoren sind mikroelektrisch-mechanische Systeme oder kurz MEMS. Diese Sensoren können Bewegungen in jede Richtung registrieren. Um die aufgezeichneten Bewegungen nun richtig interpretieren zu können, benötigt man eine passende Software. Da jede Bewegung unterschiedliche Muster aufweist, kann ein Schritt mit dieser Technologie von anderen Bewegungen, wie zum Beispiel einem Sprung, unterschieden werden.

Eine mittlerweile veraltete Methode ist es, Schritte mit Hilfe eines Gyrosensors zu erfassen, da Bewegungen sich nur ungenau auswerten und von anderen Bewegungsmustern abgrenzen lassen.

### 3.2.2 Sturzerkennung

Vor allem bei älteren, alleinlebenden Menschen, kann die Angst vor einem Sturz sehr groß sein und im Extremfall zu schweren psychischen Belastungen führen. Für andere Menschen, welche

aus bestimmten Gründen auf eine Gehhilfe angewiesen sind, kann diese Angst auch auftreten. Eine Sturzerkennung gibt Betroffenen somit Sicherheit.

### 3.2.2.1 Theoretischer Hintergrund

Ähnlich wie beim Schrittzähler kommt für die Ermittlung eines Sturzes auch ein Beschleunigungssensor zum Einsatz. Während eines Sturzes kommt es zu deutlich erhöhten Beschleunigungswerten und die anschließende Erschütterung liefert erneut höhere Werte. Auch hier müssen gelieferte Daten per Software ausgewertet werden.

### 3.2.3 MEMS<sup>1</sup>

Wie im obigen Abschnitt erwähnt, sind MEMS mikroelektrisch-mechanische Systeme oder aus dem Englischen micro-electro-mechanical Systems. Sie werden bei der Umsetzung eine wichtige Rolle spielen. Wie der Name schon vermuten lässt, können diese Elemente sowohl elektrische als auch mechanische Informationen verarbeiten. Somit werden MEMS unter anderem als Sensoren, aber auch als Aktoren verwendet. MEMS sind, so wie viele Halbleiter, meist aus Silizium.

*Basis für die Funktionsweise und Eigenschaften der unterschiedlichen Halbleiterbauelemente sind der Aufbau und die spezielle Eigenschaften des Halbleitermaterials.<sup>2</sup>*

Strukturen der MEMS sind zudem extrem klein (können kleiner als ein Mikrometer sein). Dadurch lassen sich diese Systeme grundsätzlich billig und massenhaft produzieren. Viele MEMS kann man sogar mit herkömmlichen Fertigungsverfahren für CMOS (Complementary metal-oxide-semiconductor) herstellen. Für besondere MEMS mit dreidimensionalen oder ungewöhnlich tiefen Strukturen gilt dies nicht. Sie benötigen meist weitere Fertigungsschritte wie zum Beispiel spezielle Ätzverfahren. Weiters werden hierfür besondere Materialien eingesetzt.

#### 3.2.3.1 Eigenschaften

Durch ihre Vielzahl an Eigenschaften, die MEMS mit sich bringen, stellen sie eine technische Grundlage der modernen Elektrotechnik dar, denn viele Anwendungen können ohne MEMS kaum wirtschaftlich umgesetzt werden.

---

<sup>1</sup>vgl. Elektronik Kompendium, (2022)

<sup>2</sup>Grundlagen der Elektrotechnik, (2022)

Ihre Vorteile sind:

- Robustheit
- Langzeitstabilität
- Gleichbleibende Produktqualität
- Geringe Baugröße
- Geringer Preis
- Geringer Energiebedarf
- direkte Montage auf einer Platine

Nanoelektrisch-mechanische-Systeme (NEMS) werden der nächste technische Schritt sein. Dies wird sich voraussichtlich aber erst in den nächsten Jahren zeigen.

### **3.2.3.2 Einsatzgebiete**

Zu den Vorgängern der MEMS zählen teure Spezialbausteine. Diese wurden früher in Mess-, Waffen- und Raumfahrtssystemen eingebaut. Heutzutage werden in so gut wie allen Branchen MEMS benötigt. In technischen Geräten im Bereich der Logistik, Medizin, Sicherheit oder Automobilbranche. Ein Beispiel dafür sind Airbags in Autos, die dank dieser Bauelemente intelligent realisierbar sind. Je nach Einsatzgebiet benötigt man unterschiedliche MEMS-Bausteine.

Einige Beispiele für MEMS Bausteine:

- 3-Achsen-Neigungssensor
- HF-MEMS-Schalter
- Mikrofon
- Beschleunigungssensor

In dieser Arbeit spielt vor allem der Beschleunigungssensor, im Zusammenhang mit sowohl dem Schrittzähler, als auch der Sturzerkennung eine wichtige Rolle.

### 3.3 Beschleunigungssensor

Zunächst sollte man klären was ein Beschleunigungssensor überhaupt ist bzw. nach welchem Grundprinzip dieser funktioniert. Eine weitere übliche Bezeichnung ist Accelerometer. Dieses Bauteil misst Beschleunigungen meist mittels einer wirkenden Trägheitskraft. Dieses Prinzip erlaubt es, negative und positive Beschleunigungen bzw. Geschwindigkeitszunahme oder -abnahme zu ermitteln.

Die SI-Einheit der Beschleunigung  $\left[\frac{m}{s^2}\right]$  wird hier natürlich als Messgröße herangezogen. Die mittlere Erdbeschleunigung  $g$  mit  $9,81 \frac{m}{s^2}$  wird in der Praxis als Referenz verwendet.

#### 3.3.1 Funktionsweise

Bis auf die Messgröße haben frühere Messgeräte mit modernen Sensoren nicht mehr viel gemeinsam. Sie funktionierten noch mit dem Prinzip eines Potentiometers. Das heißt, dass eine federnd gelagerte Masse einen Schleifkontakt an einem Schiebewiderstand betätigt. Nachfolger dieser Messinstrumente sind zum Beispiel MEMS. Sie arbeiten heutzutage meistens dreidimensional, also mit drei unterschiedlichen Achsen. Sie werden als X, Y und Z-Achse bezeichnet. Ihre Messbereiche können von wenigen  $g$  bis über hunderte  $g$  reichen. Dabei kann eine Auflösung von  $0,01 \text{ mg}$  erreicht werden.

#### 3.3.2 Arten<sup>3</sup>

Grundsätzlich gibt es drei verschiedene Arten von Accelerometern.

Diese sind:

1. Kapazitive Beschleunigungssensoren (MEMS)
2. Piezoresistive Beschleunigungssensoren
3. Piezoelektrische Beschleunigungssensoren

---

<sup>3</sup>vgl. TME Website, (2022)

### 3.3.2.1 Piezoresistiver Beschleunigungssensor

Die Funktionsweise dieses Sensors lässt sich mit dem Prinzip eines Dehnmessstreifens (DMS) erklären. Da diese Beschleunigungssensoren mit einem piezoresistiven Stoff behandelt werden, verursachen sie eine Änderung des elektrischen Widerstandes durch eine Krafteinwirkung. Diese Änderung wird in ein Signal umgewandelt und ausgewertet. Sie zeichnen sich durch ihre hohe Messbandbreite aus und sind deshalb vergleichsweise teuer.

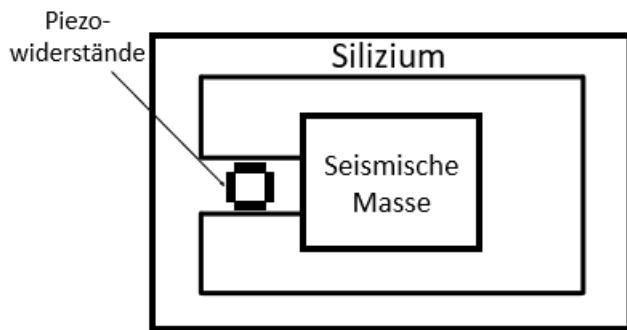


Abbildung 3.1: Darstellung piezoresistiver Beschleunigungssensor

### 3.3.2.2 Piezoelektrischer Beschleunigungssensor

Auch dieser funktioniert ähnlich. Hier ändert sich aber nicht der Widerstand, sondern die elektrische Ladung. Diese Eigenschaft wird meistens durch das Material Blei-Zirkonat-Titanat gewährleistet. Sie sind dadurch äußerst präzise, besitzen eine hohe Empfindlichkeit und gehören deshalb zu den meistverwendeten Sensoren zur Vibrationsmessung.

### 3.3.2.3 Kapazitiver Beschleunigungssensor

Diese Art von Sensoren sind sogenannte MEMS. Es gibt sie mit zwei unterschiedlichen Funktionsprinzipien. Das Feder-Masse-Prinzip oder Stimmagabelprinzip. Bei zweiterem wird die Auslenkung von schwingenden magnetisch angeregten Strukturen ermittelt. Das Feder-Masse-Prinzip ist weiter verbreitet und wird auch im Sensor, welcher in diesem Projekt verwendet wird, angewandt. Dabei wird eine Kapazitätsveränderung durch Bewegung einer Feder erfasst.

Kapazitive Beschleunigungssensoren sind die billigsten und kleinsten Beschleunigungssensoren.

Dabei ist es egal nach welchem Prinzip der Sensor arbeitet. Wie schon erwähnt, werden MEMS heutzutage in fast allen Elektronikgeräten und vor allem in Mobilgeräten verbaut. Für das Messen von hohen Amplituden und Frequenzen sind MEMS nicht ideal.

### 3.3.3 MEMS Beschleunigungssensor<sup>4</sup>

Um im nachfolgenden Kapitel das Feder-Masse-Prinzip zu verstehen, muss zunächst das Prinzip eines Kondensators verstanden werden.

#### 3.3.3.1 Kondensator

Ein Kondensator besitzt die Eigenschaft elektrische Ladung zu speichern. Das Speichervermögen, welches ein Kondensator besitzt, wird als *Kapazität* bezeichnet. Je mehr Kapazität er besitzt, umso mehr Ladung kann er speichern. Extrem vereinfacht ausgedrückt, kann man einen Kondensator deshalb mit einer Batterie vergleichen.

Der prinzipielle Aufbau eines Kondensators erfolgt mit Hilfe folgender zwei Komponenten:

- Zwei leitfähige Platten
- Isolierendes Material

Die Platten (Elektroden) werden durch ein isolierendes Material, dem Dielektrikum, getrennt.

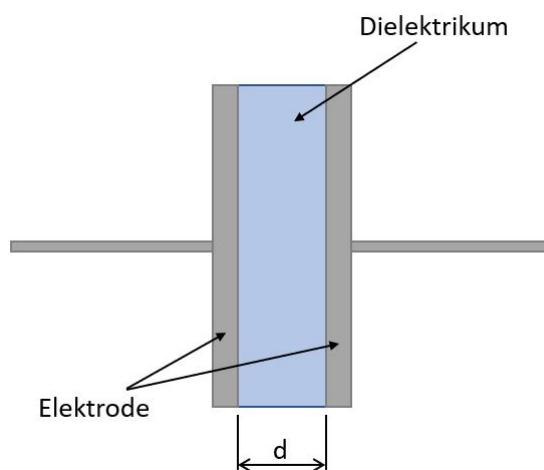


Abbildung 3.2: vereinfachte Darstellung eines Kondensators

<sup>4</sup>vgl. Elektronik Kompendium Website, (2022)

Die Fläche der Platten, das genutzte Material und der Plattenabstand  $d$ , bestimmen die Größe der Kapazität  $C$ . Diese Tatsache lässt sich mit der folgender Formel der Kapazität leicht darstellen.

$$C = \epsilon_0 \cdot \epsilon_r \frac{A}{d}$$

$\epsilon_0$  .....Naturkonstante

$\epsilon_r$  .....Materialkonstante des Dielektrikums

A .....Fläche der Platten

d .....Plattenabstand

### 3.3.3.2 Aufbau des MEMS Beschleunigungssensors

Beim Aufbau eines MEMS Beschleunigungssensors mit dem Feder-Masse-Prinzip werden drei Platten verwendet. Die äußersten zwei Platten sind fixiert und die mittlere über Federn mit diesen verbunden. Es wird eine Reihenschaltung zweier Kondensatoren gebildet. Die Kapazitäten in der Schaltung sind durch die federnde Platte veränderlich.

Verdeutlicht wird dieses Verhalten anhand der oben erwähnten Formel der Kapazität  $C$ . Die Natur- und Materialkonstante, sowie die Flächen der Platten bleiben im System immer die selben. Daraus folgt, dass der Plattenabstand allein das System beeinflusst und somit die Kapazitäten verändert.

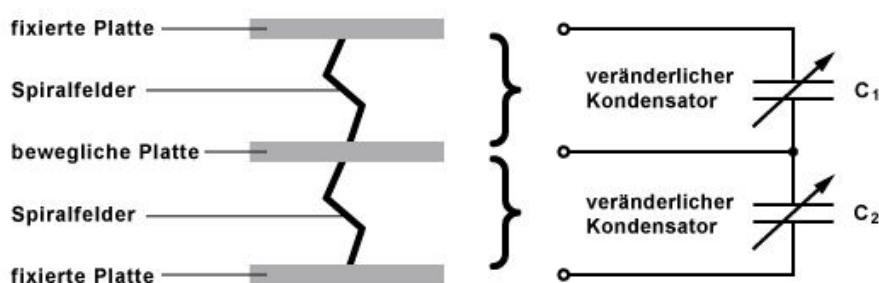


Abbildung 3.3: Aufbau und Prinzipschaltbild<sup>5</sup>

### 3.3.3.3 Funktion

Die federnde mittlere Platte reagiert auf Beschleunigung. Solange das System demnach nicht bewegt wird, ist die Platte in der Mitte und die Kapazitäten  $C_1$  und  $C_2$  sind gleich groß. Bewegt man nun das System, so wird durch die Beschleunigung die Platte in eine Richtung gedrückt. Es kommt zu einer Kapazitätsänderung proportional zur Beschleunigung. Nimmt man eine Beschleunigung nach rechts an, verändert sich der Plattenabstand folgendermaßen. Der Abstand zur linken Platte verringert sich, der Abstand zur rechten vergrößert sich. Aus der Formel ergibt sich, dass  $C_1$  größer als  $C_2$  ist. Diese ständigen Änderungen werden vom Sensor erfasst.

Bei konstanter Geschwindigkeit herrscht keine Beschleunigung und die Platte begibt sich wieder in ihre Ursprungsposition, in die Mitte zurück.

Mathematisch ermittelt man durch das Integrieren der Beschleunigung  $a$  die Geschwindigkeit  $v$ . Durch weiteres integrieren erhält man die Strecke  $s$ .

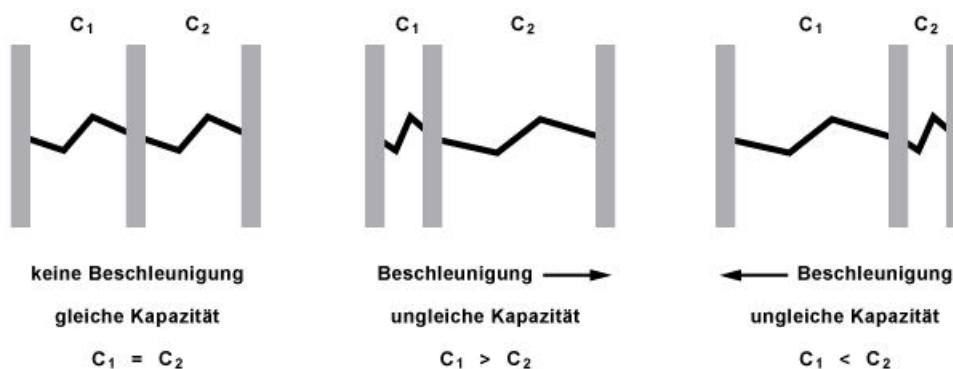


Abbildung 3.4: Funktionsprinzip<sup>6</sup>

### 3.3.4 Auswahl des Beschleunigungssensors

Die drei unterschiedlichen Technologien für Beschleunigungssensoren sind im oberen Teil (3.3.2 Arten) schon genauer beschrieben worden.

Ein piezoresistiver oder piezoelektrischer Accelerometer kommt bei diesem Projekt aufgrund des Preises nicht in Frage. Weiters sind diese beiden Arten hauptsächlich für Industrieanwendungen ausgelegt und können deshalb als Privatperson bei einigen Herstellern nicht gekauft werden.

Folgedessen kommt nur ein MEMS-Beschleunigungssensor in Frage. Die Vor- und Nachteile eines

solchen Sensors, wurden ebenfalls schon in einem oberen Abschnitt behandelt. Die Nachteile, das heißt das Messen hoher Amplituden sowie hoher Frequenzen, spielt bei der Realisierung des Schrittzählers und der Sturzerkennung für die Gehhilfe keine Rolle.

### 3.3.4.1 Fazit

Es gibt unzählige Hersteller solcher Sensoren, die Auswahl ist somit sehr groß. Obwohl die Funktion eines solchen Sensors, also die Beschleunigung zu erfassen, immer die gleiche ist gibt es trotzdem einige Unterschiede.

Diese sind unter anderem

- die Schnittstelle,
- der Messbereich und die Auflösung,
- die Betriebstemperatur,
- die Spannungsversorgung,
- der Stromverbrauch und
- in einigen Fällen der Standby-Strom.

Man kann also nicht einfach irgendeinen Sensor kaufen, sondern muss diese Spezifikationen mit den Anforderungen des Projektes vergleichen.

Für die Gehhilfe wird das GY-521 MPU-6050 Modul verwendet. Entscheidungsgrundlage dafür sind seine Spezifikationen. Weiters wurde dieses Modul schon im Unterricht genutzt und hat die Auswahl eines Sensors somit erheblich erleichtert.

### 3.3.5 GY-521 MPU-6050<sup>7</sup>

Das gewählte Modul verfügt über einen 3-Achsen Beschleunigungssensor, ein 3- Achsen Gyroskop und kann außerdem die Temperatur messen. Letztere Funktion findet jedoch keinen Anwendungsbedarf am iWalk. Der Beschleunigungssensor liefert Daten **in Richtung** der drei Achsen mit der Messgröße  $g$  (Erdbeschleunigung). Das Gyroskop hingegen liefert Daten von Bewegungen **um** die X-, Y- und Z-Achse mit der Einheit  $\text{°}/\text{s}$ . Vorteil der parallelen Nutzung eines Beschleunigungssensors und eines Gyroskops ist, dass jede Bewegung detektiert werden kann. Das folgende theoretische Beispiel verdeutlicht diese Annahme.

Angenommen das Modul liegt völlig ruhig auf einem Tisch. Das Gyroskop liefert ausschließlich den Wert Null. Der Beschleunigungssensor hingegen liefert Werte, da die Erdbeschleunigung natürlich auch im Ruhenden Zustand präsent ist.

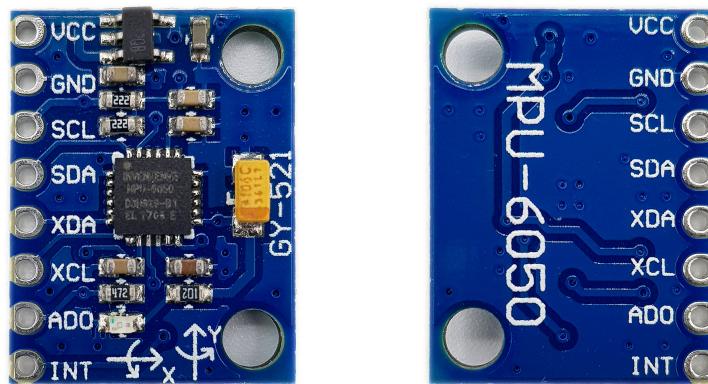


Abbildung 3.5: GY-521 MPU-6050<sup>8</sup>

<sup>7</sup>vgl. Wolles Elektronikkiste, (2022)

### 3.3.5.1 Spezifikationen

Es folgen die wichtigsten technischen Daten zum gewählten Modul.

<b>Spannungsversorgung</b>	<b>Freiheitsgrade</b>
3,3-5V (nur der Chip an sich 2,375 – 3,46V)	6 (2x 3 Achsen)
<b>Eingebauter Chip</b>	<b>Pins</b>
16-Bit AD Wandler	8
<b>Beschleunigungsmessbereich</b>	<b>Gyroskopmessbereich</b>
$\pm 2, \pm 4, \pm 8, \pm 16g$	$\pm 250, \pm 500, \pm 1000, \pm 2000 \text{ } ^\circ/\text{s}$
<b>Auflösung</b>	<b>Schnittstelle</b>
16-Bit	I <sup>2</sup> C
<b>Abmessungen</b>	<b>Gewicht</b>
25 x 20 x 7 mm	1g

### 3.3.5.2 Achsen

Wie auf Abbildung 3.5 zu erkennen ist, sind die X- und Y-Achse anschaulich auf das Modul gedruckt. Die Z-Achse steht senkrecht auf dem Modul. Die geraden Pfeile sind dem Beschleunigungssensor zugeordnet, die geschwungenen dem Gyroskop. Die Folgende theoretische Erklärung erfolgt im liegenden Zustand.

Der Beschleunigungssensor spricht wie folgt an:      Das Gyroskop spricht wie folgt an:

- X-Achse: nach links/rechts bewegen      • X-Achse: nach vorne/zurück kippen
- Y-Achse: nach vorne/zurück bewegen      • Y-Achse: nach links/rechts kippen
- Z-Achse: nach oben/unten bewegen      • Z-Achse: drehen

### 3.3.5.3 Pins<sup>9</sup>

Das GY-521 MPU-6050 Modul besitzt insgesamt 8 Pins (Abbildung 3.5). Folgende Tabelle zeigt, welche Funktionen die Pins übernehmen.

Pin	Funktion
VCC	Stromversorgung (interner Spannungsregler)
GND	Ground
SCL	I <sup>2</sup> C Serial clock
SDA	I <sup>2</sup> C Serial data
XDA	Auxiliary data
XCL	Auxiliary clock
AD0	Änderung der I <sup>2</sup> C Adresse (LOW: 0x68, HIGH: 0x69)
INT	Interrupt digital output (Optionaler Anschluss, um mehrere Module in Reihe zu schalten)

Tabelle 3.1: Pins<sup>10</sup>

### 3.3.5.4 Ansteuerung Grundlagen

Um prinzipiell mit einem Sensor richtig arbeiten zu können, müssen einige Dinge sowohl bei der Software als auch bei der Hardware sichergestellt werden. Die richtige Verbindung vom Sensor mit dem Controller ist dabei das Wichtigste. Weiters muss ein Sensor mit Hilfe passender Software angesteuert werden. Die Software sorgt dafür, dass die vom Sensor gelieferten Daten mit dem Microcontroller ausgewertet werden können.

Informationen zum Controller findet man im Kapitel **4.5 Microcontroller**.

### 3.3.5.5 Fazit

Das gewählte Modul verfügt über gute technische Eigenschaften. 6 Freiheitsgrade und eine hohe Auflösung von 16-Bit ermöglichen vielseitige Anwendung. Da die Gehhilfe wenig Platz für Elektronik bietet, sind die Abmessungen der Bauteile ein wichtiges Kriterium. Außerdem punktet das Modul mit seinem geringen Preis.

Die Kombination aus diesen Punkten spricht für das GY-521 MPU6050 Modul.

---

<sup>9</sup>vgl. Pins, (2022)

## 3.4 Auswahl der Entwicklungsumgebung

Das Ansteuern und Auswerten von Messdaten passiert über den Microcontroller und der darauf laufenden Software. Folge dessen, spielt sie eine wichtige Rolle für die Funktion der Gehilfe. Eine geeignete Entwicklungsumgebung für den Code ist daher für das Endergebnis ausschlaggebend.

Zur Auswahl stehen *Assembler* oder *C* als Programmiersprachen oder die Entwicklungsumgebung *Arduino IDE*.

### 3.4.1 Programmiersprachen

Zunächst muss geklärt werden in welcher Sprache der Microcontrollerchip Codes verarbeiten kann. Diese nennt sich Maschinensprache und ist meistens ein Binärkode. Sie kann aber auch in Hexadezimalen Zahlen dargestellt werden.

Dezimal	Binär	Hexadezimal
1	0001	1
2	0010	2
9	1001	9
10	1010	A
11	1011	B
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12

Tabelle 3.2: Veranschaulichung der Zahlensysteme

Wie man womöglich aus der Tabelle erkennen kann, ist ein Programmieren in Maschinensprache für den Menschen schlicht unmöglich.

Die „nächsthöhere“ Sprache ist die Assemblersprache, kurz Assembler. Diese ist noch eine sogenannte maschinenorientierte Programmiersprache. Danach kommen vereinfacht gesagt schon klassische Programmiersprachen wie zum Beispiel C, C++, Java und viele mehr.

Programmiert man nun in Assembler oder C muss der Code zuerst in Maschinensprache übersetzt werden. Erst dann kann der Microcontrollerchip damit arbeiten. Diese Aufgabe übernimmt der sogenannte Compiler (Übersetzer). Nachstehende Grafik soll dieses Prinzip verdeutlichen.

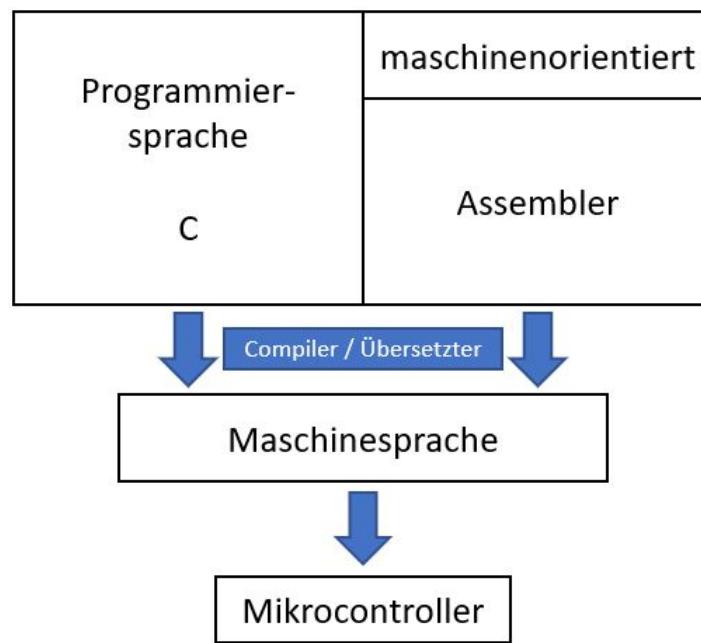


Abbildung 3.6: Veranschaulichung der Übersetzung in Maschinensprache

### 3.4.1.1 Assembler

Assembler ist praktisch eine Weiterentwicklung der Maschinensprache. Man kann im Code leichter den Überblick behalten, da sie besser veranschaulicht ist. Es ist also durchaus möglich, in dieser Sprache zu programmieren und wird in der Praxis auch gemacht. Es bleibt jedoch immer noch kompliziert.

Mit Assembler zu programmieren, wird aus diesem Grund in diesem Projekt ausgeschlossen.

### 3.4.1.2 C

Wenn es um das Programmieren eines Microcontrollers geht, so ist C wahrscheinlich die am weitest verbreitete Programmiersprache. Der Grund dafür ist die hardwarenahe Programmierung. Bibliotheksfunktionen bzw. Standardbibliotheken von C erleichtern das Programmieren. Umgebungen, mit denen man C programmieren kann, wurden im Unterricht schon häufig verwendet. Unter anderem sind das *CodeBlocks* und *NetBeans*.

### 3.4.2 Arduino IDE<sup>11</sup>

Wenn man mit einem Microcontroller zu tun hat, so stößt man sicher auf die Arduino IDE. Das ist eine Open-Source Plattform, die das Programmieren eines Microcontrollers erleichtern soll. Einfache Dinge, wie zum Beispiel das Ansteuern einer LED werden mit Hilfe von Beispielprogrammen direkt verfügbar gemacht. Außerdem verfügt die Arduino IDE über viele Bibliotheken in denen Programme verfügbar sind. Externe Bibliotheken lassen sich direkt in der Entwicklungsumgebung herunterladen. Der Code in der Anwendung wird durch eine eigene Programmiersprache, die *Arduino programming language*, noch übersichtlicher. Elemente aus C++ werden dafür verwendet.

### 3.4.3 Fazit

Alle Vorteile sprechen für die Arduino IDE. Bibliotheken und Beispielprogramme erleichtern maßgeblich die Programmierarbeit. Die Entscheidung für die Entwicklungsumgebung fällt somit auf die Arduino IDE.

### 3.4.4 Einrichten der Entwicklungsumgebung

Vor dem tatsächlichen Programmieren müssen in der Arduino IDE einige Dinge konfiguriert werden. Da jedes Board eine unterschiedliche Beschaltung besitzt, muss festgelegt werden welches man benutzt. Anschließend muss kontrolliert werden, ob der richtige Bootloader verwendet wird. Das ist wichtig, denn er lädt den Hauptspeicher. Ohne passenden Bootloader ist die Funktion des Prozessors deshalb nicht gegeben.

Benötigt man nun weitere Bibliotheken, kann man diese im nächsten Schritt direkt in der Entwicklungsumgebung herunterladen.

---

<sup>11</sup>vgl. Arduino, (2022)

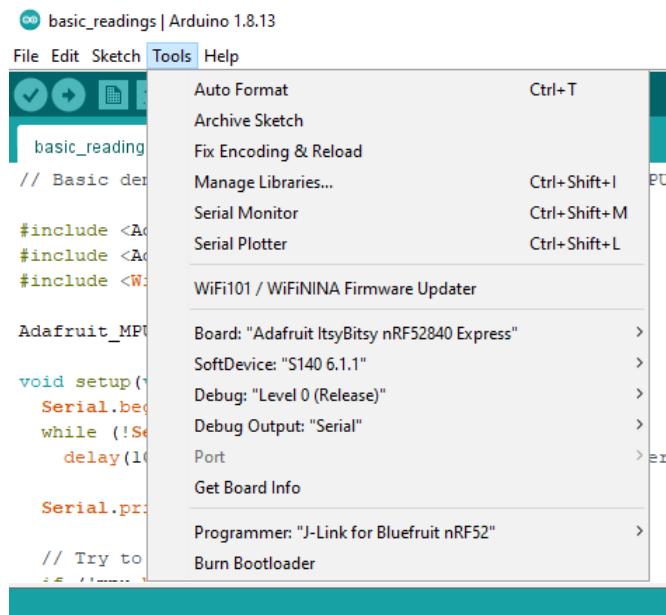


Abbildung 3.7: Einstellungen für das ursprünglich ausgewählte Board

## 3.5 Ansteuern des Sensors

Bevor man sich mit dem Hauptprogrammcode für die Umsetzung des Schrittzählers und der Sturzerkennung beschäftigen kann, muss man Messdaten des Sensors auswerten und vergleichen.

Um Messdaten zu erhalten, wurde zunächst eine sogenannte Custom Library in die Entwicklungs-umgebung eingebunden. Sie nennt sich *Adafruit MPU6050* und beinhaltet 5 Examples (Beispiele).

Mit Hilfe von *basic\_readings* wurden Sensordaten erfasst und verglichen, was es zum relevantesten Beispielprogramm in diesem Projekt macht.

Das Programm erlaubt, dass Messbereiche und ein Low-Pass Filter direkt im Code eingestellt werden können. Es funktioniert fehlerfrei und wird somit im Anschluss unverändert abgedruckt.

```

1 // Basic demo for accelerometer readings from Adafruit MPU6050
2
3 #include <Adafruit_MP6050.h>
4 #include <Adafruit_Sensor.h>
5 #include <Wire.h>
6
7 Adafruit_MP6050 mpu;
8
9 void setup(void) {
10     Serial.begin(115200);

```

```
11  while (!Serial)
12      delay(10); // will pause Zero, Leonardo, etc until serial console
13      opens
14
15  Serial.println("Adafruit MPU6050 test!");
16  // Try to initialize!
17  if (!mpu.begin()) {
18      Serial.println("Failed to find MPU6050 chip");
19      while (1) {
20          delay(10);
21      }
22  Serial.println("MPU6050 Found!");
23
24  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
25  Serial.print("Accelerometer range set to: ");
26  switch (mpu.getAccelerometerRange()) {
27  case MPU6050_RANGE_2_G:
28      Serial.println("+-2G");
29      break;
30  case MPU6050_RANGE_4_G:
31      Serial.println("+-4G");
32      break;
33  case MPU6050_RANGE_8_G:
34      Serial.println("+-8G");
35      break;
36  case MPU6050_RANGE_16_G:
37      Serial.println("+-16G");
38      break;
39  }
40  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
41  Serial.print("Gyro range set to: ");
42  switch (mpu.getGyroRange()) {
43  case MPU6050_RANGE_250_DEG:
44      Serial.println("+- 250 deg/s");
45      break;
```

```
46     case MPU6050_RANGE_500_DEG:  
47         Serial.println("+- 500 deg/s");  
48         break;  
49     case MPU6050_RANGE_1000_DEG:  
50         Serial.println("+- 1000 deg/s");  
51         break;  
52     case MPU6050_RANGE_2000_DEG:  
53         Serial.println("+- 2000 deg/s");  
54         break;  
55     }  
56  
57     mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);  
58     Serial.print("Filter bandwidth set to: ");  
59     switch (mpu.getFilterBandwidth()) {  
60     case MPU6050_BAND_260_HZ:  
61         Serial.println("260 Hz");  
62         break;  
63     case MPU6050_BAND_184_HZ:  
64         Serial.println("184 Hz");  
65         break;  
66     case MPU6050_BAND_94_HZ:  
67         Serial.println("94 Hz");  
68         break;  
69     case MPU6050_BAND_44_HZ:  
70         Serial.println("44 Hz");  
71         break;  
72     case MPU6050_BAND_21_HZ:  
73         Serial.println("21 Hz");  
74         break;  
75     case MPU6050_BAND_10_HZ:  
76         Serial.println("10 Hz");  
77         break;  
78     case MPU6050_BAND_5_HZ:  
79         Serial.println("5 Hz");  
80         break;  
81     }
```

```
82
83     Serial.println("");
84     delay(100);
85 }
86 void loop() {
87     /* Get new sensor events with the readings */
88     sensors_event_t a, g, temp;
89     mpu.getEvent(&a, &g, &temp);
90
91     /* Print out the values */
92     Serial.print("Acceleration X: ");
93     Serial.print(a.acceleration.x);
94     Serial.print(", Y: ");
95     Serial.print(a.acceleration.y);
96     Serial.print(", Z: ");
97     Serial.print(a.acceleration.z);
98     Serial.println(" m/s^2");
99
100    Serial.print("Rotation X: ");
101    Serial.print(g.gyro.x);
102    Serial.print(", Y: ");
103    Serial.print(g.gyro.y);
104    Serial.print(", Z: ");
105    Serial.print(g.gyro.z);
106    Serial.println(" rad/s");
107
108    Serial.print("Temperature: ");
109    Serial.print(temp.temperature);
110    Serial.println(" degC");
111
112    Serial.println("");
113    delay(500);
114 }
```

Listing 3.1: Auslesen von Sensordaten

### 3.5.1 Pins

Wie in einem oberen Kapitel erwähnt, ist die richtige Verbindung zwischen Sensor und Microcontroller essenziell.

In diesem Projekt werden in jedem Fall, sei es das Beispielprogramm oder das Hauptprogramm, vier von insgesamt acht Pins vom Sensormodul verwendet. Diese sind VCC, GND, SDA und SCL.

Wird der Sensor nun falsch angeschlossen, können folgende Fehler auftreten.

Der Microcontroller

- findet den Sensor nicht.
- bekommt falsche Daten übermittelt.
- bekommt keine Daten übermittelt.

Diese Fehler beziehen sich auf das falsche Anschließen der Datenleitungen (SDA und SCL). Ein Defekt des Sensors kann nur bei falscher Verbindung der Spannungsversorgung (VCC) und Ground (GND) auftreten. Da diese beiden Anschlüsse auf dem Board des nun ausgewählten Controllers lesbar aufgedruckt sind, sollte es dazu im Normalfall aber nicht kommen.

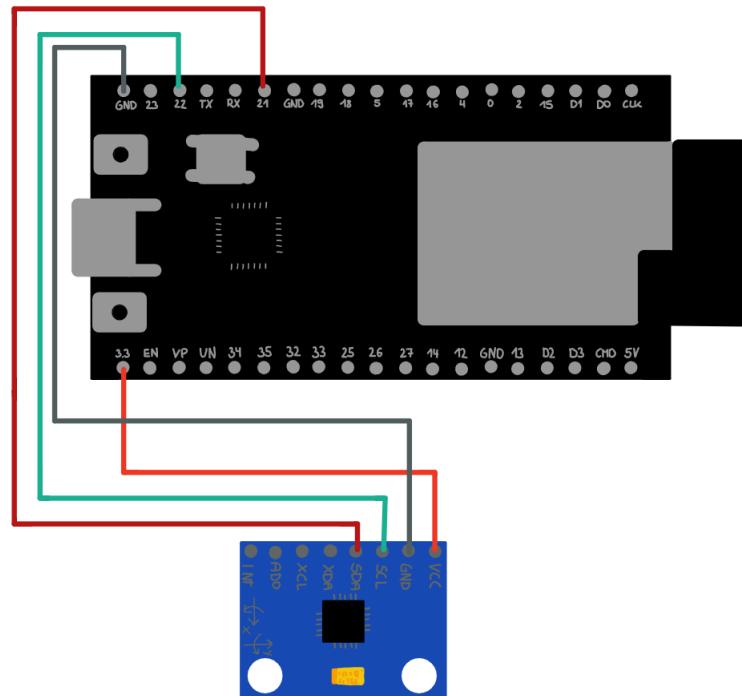


Abbildung 3.8: Sensor am Microcontroller anschließen

Pins am GY-521 MPU-6050	Pins am Microcontroller (ESP32)
VCC	3V3 (oder 5V)
GND	GND
SDA	21
SCL	22

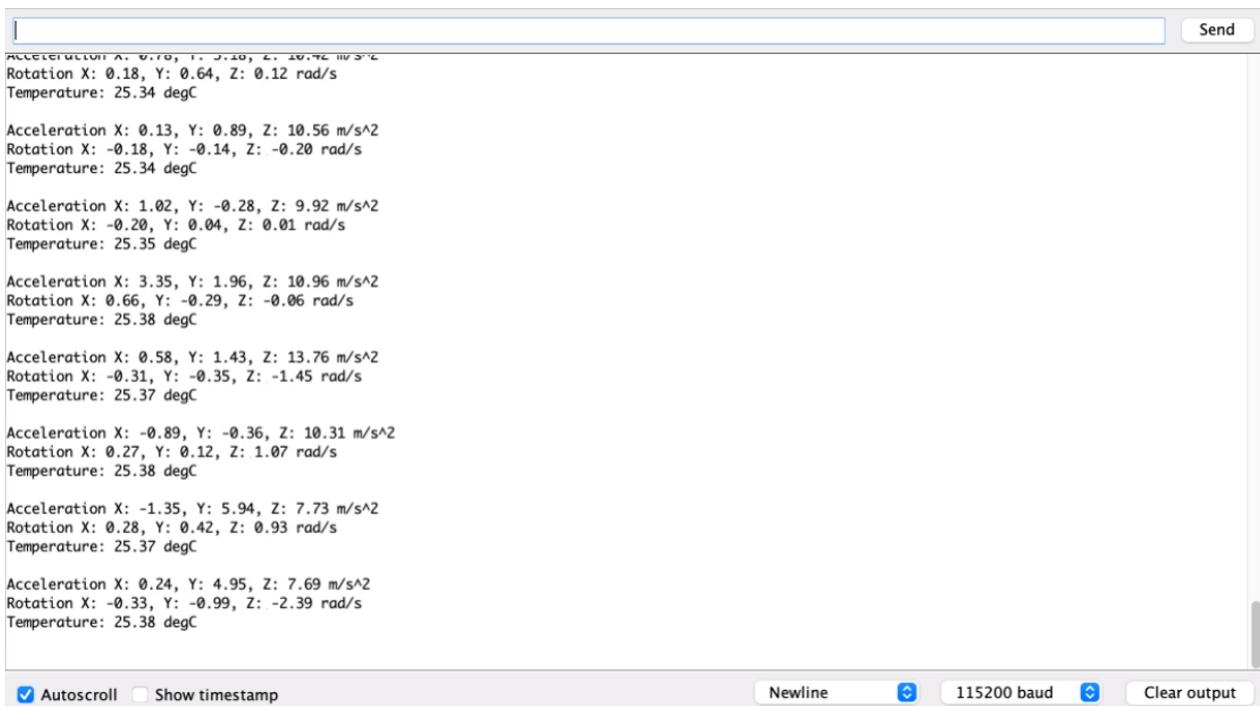
Tabelle 3.3: Sensor am Microcontroller anschließen

### 3.5.2 Daten auswerten

Nachdem das Sensormodul angeschlossen ist, kann man das Programm auf den Controller laden.

Um nun Messdaten lesen zu können, muss im Bereich *Tools* der *Serial Monitor* aktiviert werden.

Wichtig dabei ist, die richtige Baud-Rate (Übertragungsgeschwindigkeit) auszuwählen. Ohne diese Vorkehrung werden keine Daten übermittelt. In diesem Programm ist sie 115200.



The screenshot shows the Arduino Serial Monitor window. At the top, there is a text input field and a 'Send' button. Below the input field, the text area displays the following sensor data:

```

Rotation X: 0.18, Y: 0.64, Z: 0.12 rad/s
Temperature: 25.34 degC

Acceleration X: 0.13, Y: 0.89, Z: 10.56 m/s^2
Rotation X: -0.18, Y: -0.14, Z: -0.20 rad/s
Temperature: 25.34 degC

Acceleration X: 1.02, Y: -0.28, Z: 9.92 m/s^2
Rotation X: -0.20, Y: 0.04, Z: 0.01 rad/s
Temperature: 25.35 degC

Acceleration X: 3.35, Y: 1.96, Z: 10.96 m/s^2
Rotation X: 0.66, Y: -0.29, Z: -0.06 rad/s
Temperature: 25.38 degC

Acceleration X: 0.58, Y: 1.43, Z: 13.76 m/s^2
Rotation X: -0.31, Y: -0.35, Z: -1.45 rad/s
Temperature: 25.37 degC

Acceleration X: -0.89, Y: -0.36, Z: 10.31 m/s^2
Rotation X: 0.27, Y: 0.12, Z: 1.07 rad/s
Temperature: 25.38 degC

Acceleration X: -1.35, Y: 5.94, Z: 7.73 m/s^2
Rotation X: 0.28, Y: 0.42, Z: 0.93 rad/s
Temperature: 25.37 degC

Acceleration X: 0.24, Y: 4.95, Z: 7.69 m/s^2
Rotation X: -0.33, Y: -0.99, Z: -2.39 rad/s
Temperature: 25.38 degC

```

At the bottom of the window, there are several buttons: 'Autoscroll' (checked), 'Show timestamp' (unchecked), 'Newline' (dropdown menu), '115200 baud' (dropdown menu), and 'Clear output'.

Abbildung 3.9: Beispielhafte Ausgabe der Programms

Mit einem anderen Beispielprogramm (*plotter*) aus der selben Bibliothek, wurde zu Testzwecken ein Schritt simuliert. Dafür muss der *Serial Plotter* aktiviert werden.



Abbildung 3.10: Simulation eines Schrittes

## 3.6 Hauptprogramm

### 3.6.1 Sensorposition

Richtige Messungen zu erzielen ist die Grundvoraussetzung für eine einwandfreie Funktion. Vor der Programmierarbeit muss deshalb die Ausrichtung des Sensors klar definiert werden. Die Position des Sensors wurde am Handstück der Gehilfe festgelegt. Folgende Schemaskizzen dienen zur Veranschaulichung.

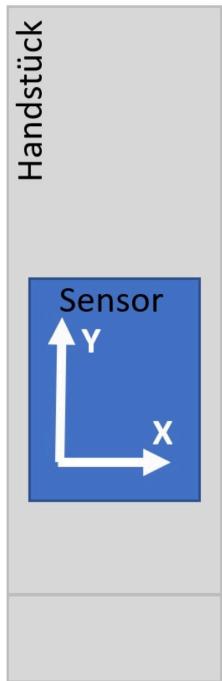


Abbildung 3.11: Ansicht von Oben



Abbildung 3.12: Ansicht von Links

### 3.6.2 Überlegungen zum Schrittzähler

Es sollen Schritte detektiert und anschließend gezählt werden. Es wurden zwei Messungen durchgeführt, um einen Wertebereich zu definieren. Sobald Sensorwerte innerhalb dieses Bereiches liegen wird um **eins** hochgezählt.

#### 3.6.2.1 Messdaten

Mit dem bereits erwähnten Beispielprogramm *basic\_readings* wurden die Messungen durchgeführt.

Daten, welche beim Simulieren eines Schrittes aufgezeichnet werden, dienen als Basis für den Wertebereich.

Bezeichnung	X-Achse	Y-Achse	Z-Achse	Einheit
Accelerometer	-1,6 - 1,5	-1,3 - 5,0	9,9	m/s <sup>2</sup>
Gyroskop	-0,3 - 0,7	-0,2 - 0,7	-0,6 - 0,6	rad/s

Tabelle 3.4: Simulation eines Schrittes

Anschließend wurden Daten in der Ruheposition des Sensors ermittelt.

Bezeichnung	X-Achse	Y-Achse	Z-Achse	Einheit
Accelerometer	0,6	1,1	9,9	m/s <sup>2</sup>
Gyroskop	0,03	0,04	-0,01	rad/s

Tabelle 3.5: Ruheposition des Sensors

#### 3.6.2.2 Definition Wertebereich

Während eines Schrittes überschneiden sich die Messdaten mit denen der Ruheposition und Schritte würden falsch gezählt werden. Um diesem Verhalten entgegenzuwirken, wurde ein klarer Wertebereich festgelegt. Außerdem sind sowohl die Werte des Gyroskops als auch die Beschleunigungsmessung der Z-Achse, für das Detektieren eines Schrittes irrelevant.

Der festgelegte und im Programmcode verwendete Wertebereich lautet daher wie folgt:

- Accelerometer X-Achse: 0,8 - 1,5 m/s<sup>2</sup>
- Accelerometer Y-Achse: 1,4 - 5,0 m/s<sup>2</sup>

### 3.6.3 Programmcode Schrittzähler

```
1 //Bibliotheken
2 #include "math.h"
3 #include "Wire.h"
4 #include <Adafruit_MPU6050.h>
5 #include <Adafruit_Sensor.h>
6
7 //Variablen
8 int Steps = 0;
9 Adafruit_MPU6050 mpu;
10
11 void setup() {
12     //Einstellung der Uebertragungsgeschwindigkeit
13     Serial.begin(9600);
14     // Initialisierung MPU:
15     if (!mpu.begin()) {
16         Serial.println("Failed to find MPU6050 chip");
17         while (1) {
18             delay(10);
19         }
20     }
21     Serial.println("MPU6050 Found!");
22
23     //Einstellungen MPU
24     mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
25     mpu.setGyroRange(MPU6050_RANGE_500_DEG);
26     mpu.setFilterBandwidth(MPU6050_BAND_44_HZ);
27 }
28
29 void loop() {
30     //Aufrufen der Funktion
31     steps();
32
33     delay(500);
34 }
```

```
35
36 void steps() {
37
38     //Sensorwerte erhalten
39     sensors_event_t a, g, temp;
40     mpu.getEvent(&a, &g, &temp);
41
42     //Wertebereiche
43     double gyro_x = g.gyro.x;
44     double gx = 0;
45     if(gyro_x > 0.1 && gyro_x < 0.7) {
46         gx = 1;
47     }
48     double gyro_y = g.gyro.y;
49     double gy = 0;
50     if(gyro_y > 0.1 && gyro_y < 0.7) {
51         gy = 1;
52     }
53     double gyro_z = g.gyro.z;
54     double gz = 0;
55     if(gyro_z > 0.1 && gyro_z < 0.6) {
56         gz = 1;
57     }
58
59     double accel_x = a.acceleration.x;
60     double ax = 0;
61     if(accel_x > 0.8 && accel_x < 1.5) {
62         ax = 1;
63     }
64     double accel_y = a.acceleration.y;
65     double ay = 0;
66     if(accel_y > 1.4 && accel_y < 5) {
67         ay = 1;
68     }
69     double accel_z = a.acceleration.z;
70     double az = 0;
```

```
71     if(accel_z > 6 && accel_z < 9) {  
72         az = 1;  
73     }  
74  
75     //Abfrage der relevanten Sensordaten und hochzaehlen von Schritten  
76     if(/*gx == 1 && gy == 1 && gz == 1 &&*/ ax == 1 && ay == 1) {  
77         Steps++;  
78     }  
79     Serial.println("Schritte:");  
80     Serial.println(Steps);  
81 }
```

Listing 3.2: Schrittzähler

### 3.6.4 Überlegung zur Sturzerkennung

Hier sollen Stürze einer Person erkannt und ebenfalls gezählt werden. Diese Funktion ist mit der Kraftmessung (**4.3 Kraftmessung**) gekoppelt. Als Sturz gewertet werden Sensordaten der zu Boden gefallenen Gehhilfe. Grundvoraussetzung ist, dass der Kraftsensor keine Belastung wahrnimmt. Die auftretenden Beschleunigungen in Richtung links oder rechts, sowie nach vorne und hinten, werden als Anhaltspunkte herangezogen. Bei Bewegungen in diese Richtungen sind die X- und Y-Achse am meisten beansprucht.

Da mittels der Sturzerkennung Hilfe angefordert werden kann, darf es nicht zu falschen Zählungen kommen. Demnach wird die Funktion im Programmcode in einem bestimmten Zeitintervall abgefragt, welcher sich je nach Anforderung verändern lässt. Fällt die Gehhilfe nun zum Beispiel aus der Hand und wird sofort wieder aufgehoben, wird kein Sturz detektiert.

### 3.6.4.1 Messdaten

Bei Bewegungen nach links oder rechts spricht die X-Achse an. Für die Messung wurde die Gehhilfe um 90° nach links gekippt. Eine Messung in rechter Lage ist nicht nötig, da sich nur das Vorzeichen (X-Achse) ändert.

Bezeichnung	X-Achse	Y-Achse	Z-Achse	Einheit
Accelerometer	10,5	0,5	0,5	m/s <sup>2</sup>

Tabelle 3.6: 90° nach Links gekippt

Kippt man die Gehhilfe um 90° nach vorne, treten Beschleunigungen an der Y-Achse auf. In gegenüberliegender Lage dreht sich auch hier das Vorzeichen (Y-Achse) einfach um.

Bezeichnung	X-Achse	Y-Achse	Z-Achse	Einheit
Accelerometer	0,5	-10,0	1,0	m/s <sup>2</sup>

Tabelle 3.7: 90° nach Vorne gekippt

### 3.6.4.2 Analyse

Aufgrund der aufgezeichneten Werte wurde für den Programmcode eine Beschleunigung von ±9,5 m/s<sup>2</sup> für die Sturz-Abfrage festgelegt. Der Grenzwert wurde bewusst niedriger als die bei den Testungen erzielten Beschleunigungen angesetzt. Grund dafür ist, dass die Gehhilfe möglicherweise nicht genau im 90° Winkel am Boden liegen bleibt und somit die auftretenden Beschleunigungen der Testungen nicht erreicht.

### 3.6.5 Programmcode Sturzerkennung

```

1 //Bibliotheken
2 #include "BluetoothSerial.h"
3 #include "math.h"
4 #include "Wire.h"
5 #include <Adafruit_MPU6050.h>
6 #include <Adafruit_Sensor.h>
7
8 #if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)

```

```
9 #error Bluetooth is not enabled! Please run 'make menuconfig' to and
10 enable it
11
12 //Variablen
13 int Falls = 0;
14 int Weight = 0;
15 const long interval = 30000;      //Zeitintervall der Abfrage (30s);
16           veraenderlich
17 unsigned long ptime = 0;
18 BluetoothSerial SerialBT;
19
20 void setup() {
21     //Einstellung der Uebertragungsgeschwindigkeit
22     Serial.begin(9600);
23     // Initialisierung MPU:
24     if (!mpu.begin()) {
25         Serial.println("Failed to find MPU6050 chip");
26         while (1) {
27             delay(10);
28         }
29     }
30     Serial.println("MPU6050 Found!");
31
32     //Einstellungen MPU
33     mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
34     mpu.setGyroRange(MPU6050_RANGE_500_DEG);
35     mpu.setFilterBandwidth(MPU6050_BAND_44_HZ);
36 }
37 void loop() {
38     //Aufrufen der Funktion
39     fall();
40
41     delay(500);
42 }
```

```
43
44 void fall() {
45     //Sensorwerte erhalten
46     sensors_event_t a, g, temp;
47     mpu.getEvent(&a, &g, &temp);
48
49     //millis() zaehlt die Laufzeit des Programmes; ist fuer den
50     //Zeitintervall noetig
51     unsigned long t = millis();
52
53     //Abfrage der Sturzerkennung in einem bestimmten Zeitintervall; hier 30s
54     //Volle Funktion nur gemeinsam mit dem Code der Kraftmessung
55     if(t - ptime >= interval) {
56         if((a.acceleration.x < -9.5 || a.acceleration.x > 9.5 ||
57             a.acceleration.y < -9.5 || a.acceleration.y > 9.5) && Weight < 1)
58         {
59             Falls++;
60             SerialBT.println("HELP ME");
61         }
62     }
63     Serial.println("Stürze:");
64     Serial.println(Falls);
65 }
```

Listing 3.3: Sturzerkennung



## **4 Teil D - Sensorik**

## 4.1 Aufgabenstellung

Die zu untersuchende Thematik ist die Entwicklung des Akkusystems und der Kraftanalyse.

Die Kraftanalyse dient zur Belastungsanalyse. Für den Genesungsverlauf ist es wichtig, dass man die Krücke nur als Unterstützung nutzt. Dies kann mit einem Diagramm sehr gut abgebildet und beobachtet werden.

Das Akku-System soll allen Ansprüchen entsprechen und sollte nur einmal pro Woche geladen werden müssen.

## 4.2 Projektplanung

### 4.2.1 Projektdefinition

Bevor man mit der Planung eines Projektes beginnt, sollte man das Projekt definieren und sich folgende Fragen stellen.

**Kann man bei einer Diplomarbeit von einem Projekt sprechen? Welche Art von Projekt wäre dies?**

Man kann bei der Diplomarbeit von einem Projekt sprechen. Es handelt sich um ein Forschungs- und Entwicklungsprojekt.

In einem Forschungs- und Entwicklungsprojekt gewinnt man neue Erkenntnisse oder Innovationen. Man gliedert es zu den „klassischen“ Projektarten. Grund dafür ist der Innovationsgrad und das Projektrisiko.

**Welchen Nutzen hat das Projekt?**

Das Projekt „iWalk“ soll betroffenen Menschen, die körperlich eingeschränkt sind, schneller Hilfe bei Stürzen garantieren. Auch die gesundheitliche Tätigkeit soll via Schrittzähler und Kraftanalysen aufgezeichnet werden.

## Ist das Projekt wichtig?

Diese Frage ist vor allem dann wichtig, wenn mehrere Projekte gleichzeitig bearbeitet werden.

Das Diplomprojekt ist wichtig, da es in das Maturazeugnis einfließt.

## Gibt es Risiken?

Bei jedem Projekt gibt es Risiken und daraus folgende Erkenntnisse.

Bei einer Diplomarbeit sind die größten Risiken, eine verspätete Abgabe oder das nicht Erfüllung eines geforderten Ziels. Da der Lerneffekt im Vordergrund steht, minimieren sich die Risiken.

## 4.2.2 Projektorganisation

Organisiert wurde das Projekt mit *Microsoft Project*. Dazu muss man folgende Ressourcen anlegen, um ihnen Tätigkeiten zuweisen zu können.

Ressourcename	Art	Materialbeschreibung	Knoten	Gruppe	Position	Max.	Standardzeit	Überzeit-Satz	Kosten/Freitags	Fällig	Basisleiter	Code	E-Mail-Adresse	Projekt
Paul Resch	Arbeit		P	Elektrotechnik	Projekteam	100%	17,76 €/Std.	26,64 €/Std.	0,00 € Anteilig	Standard	040203	respm17@htl-kaindorf.at		
Christoph Sebernegg	Arbeit		C	Informatik	Projekteam	100%	17,76 €/Std.	26,64 €/Std.	0,00 € Anteilig	Standard	240703	sebchm17@htl-kaindorf.at		
Kilian Wade	Arbeit		K	Mechanik	Projekteam	100%	17,76 €/Std.	26,64 €/Std.	0,00 € Anteilig	Standard	010603	wadkim17@htl-kaindorf.at		
Georg Kaufmann	Arbeit		G	Elektrotechnik	Projektleiter	100%	19,43 €/Std.	29,15 €/Std.	0,00 € Anteilig	Standard	121102	kaugen17@htl-kaindorf.at		

Abbildung 4.1: Mitglieder der DA

Anschließend wurde eine Organisationsform mit den entsprechenden Kommunikationswegen erstellt. Die Kommunikationswege sind wichtig, damit der Projektablauf störungsfrei funktioniert. Jedes Projektmitglied sollte wissen, wer sein Ansprechpartner ist.

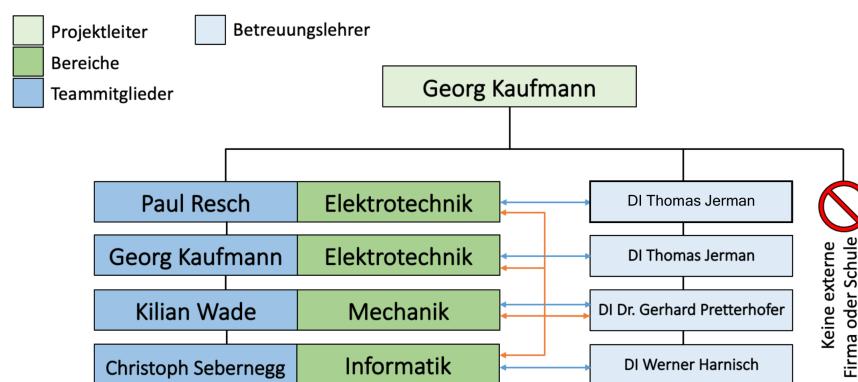


Abbildung 4.2: Organisationsform

Darauf wurden folgende Dinge definiert:

## **Wann und wie finden Projektbesprechungen statt?**

Jeden Mittwoch, situationsbedingt persönlich oder via Discord.

## **Wie werden Informationen untereinander ausgetauscht?**

Informationen werden persönlich, via Discord oder in der von uns erstellten WhatsApp Gruppe ausgetauscht.

## **In welcher Art und Form werden Statusberichte erstellt?**

Jeden Mittwoch werden Statusberichte in Form von Protokollen erstellt.

## **Wie werden Informationen geteilt und abgelegt?**

Dokumente, Berichte und andere Informationen werden im OneDrive und einer angelegten Mappe abgelegt.

## **Wo geschieht die Projektdokumentation**

Die Diplomarbeit wird in Overleaf - Latex verfasst.

### **4.2.3 Projektphasen**

Das Ziel ist es, Stürze zu erkennen und Betroffenen schnelle Hilfe zu gewährleisten. Zudem erleichtert es die Feststellung der gesundheitlichen und körperlichen Verfassung durch Schritt- und Kraftanalysen. Die gesammelten Daten können einfach in der Smartphone App abgerufen werden.

Das Ziel ist es, das Projekt mit einem geschätzten Kostenaufwand von 200 € bis Februar 2022 fertigzustellen

Anbei befindet sich der Zeitplan.

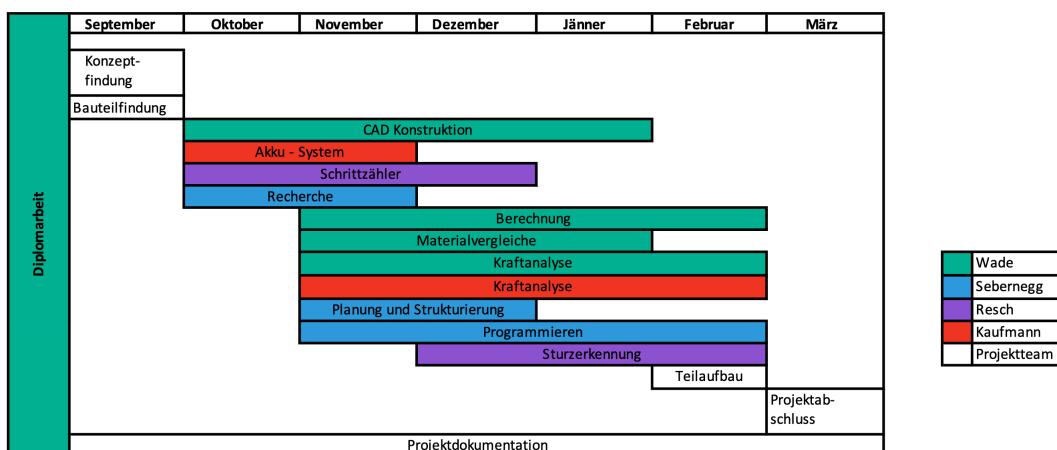


Abbildung 4.3: Zeitpläne

## 4.3 Kraftmessung

### 4.3.1 Anforderungen an den Kraftsensor

Der Kraftsensor sollte eine Belastung in vertikaler Richtung aushalten.

Zur Feststellung wie hoch die Belastung bei einer Person mit zum Beispiel 100 kg ist, wurden Messungen mit einer handelsüblichen Waage angestellt. Dabei stellte sich heraus das sich die Masse halbiert. Somit wirken bei einer Person mit 100 kg, 50 kg auf eine einzelne Gehhilfe.

Um einen passenden Sensor auswählen zu können, wurde das minimale und das maximale Gewicht recherchiert.

Das Normalgewicht einer Frau bzw. eines Mannes, das am häufigsten Auftritt, wird in der unteren Normalverteilungen dargestellt.

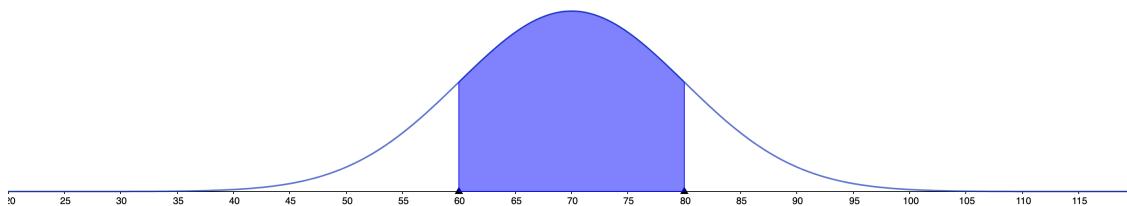


Abbildung 4.4: Normalgewicht einer Frau

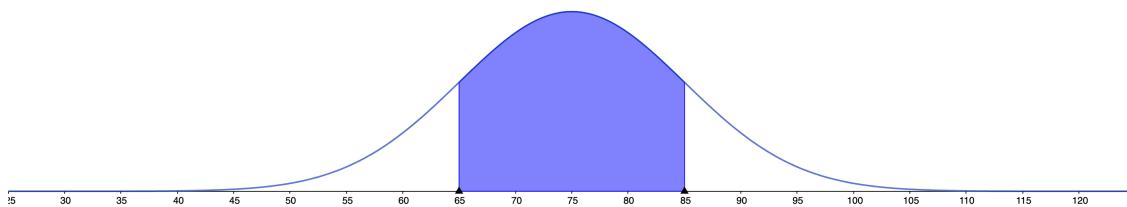


Abbildung 4.5: Normalgewicht eiens Mannes

Die Anforderungen die von dem Kraftsensor zu erfüllen sind, werden in der untenstehenden Tabelle dargestellt.

Geschlecht	max. Gewicht	min. Gewicht
Frau	95 kg	45 kg
Mann	100 kg	50 kg

Tabelle 4.1: Übersicht Gewicht

Der minimale Wert lautet 45 kg und der maximale Wert lautet 100 kg.

Somit lauten die Anforderungen an den Kraftsensor wie folgt.

Maximaler Wert: 50 kg + Sicherheit von 10 kg = 60 kg

Minimaler Wert: 22,5 kg – Sicherheit von 10 kg = 12,5 kg

Die Sicherheit wird addiert bzw. subtrahiert, um einen größeren Messbereich abdecken zu können.

Die Werte müssen jetzt noch in Newton umgerechnet werden, da die Hersteller von Kraftsensoren den Messbereich in der Einheit Newton angeben.

Maximaler Wert: 60 kg \* 9,81 N = 588,6 N

Minimaler Wert: 12,5 kg \* 9,81 N = 122,62 N

min. Anforderung		max. Anforderung	
kg	N	kg	N
12,5	122,6	60	588,6

Tabelle 4.2: Anforderungen Kraftsensor

## 4.3.2 Arten von Kraftsensoren<sup>1</sup>

### 4.3.2.1 Piezo-Sensor

Bei einem Piezo-Sensor wird ein piezoelektrischer Kristall verwendet. Dieser piezoelektrische Kristall sorgt für eine Ladungsverschiebung auf molekularer Ebene. Mittels eines Ladungsverstärkers kann diese Ladungsverschiebung in ein Spannungssignal umgewandelt werden. Anschließend kann man dieses Signal in einen Messwert umwandeln.

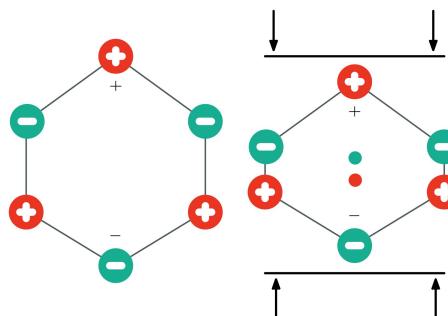


Abbildung 4.6: Piezo-Prinzip

<sup>1</sup>vgl. Kistler, (2021)

**Vorteil:**

Die Verformung dieses Sensors ist äußerst gering und aus diesem Grund können extrem steife Sensor-Aufbauten mit hoher Eigenfrequenz realisiert werden. Zudem ist es ideal für hochfrequente Messereignisse.

**Nachteil:**

Die Ladung geht verloren wenn es keine gute Isolation gibt. Aus diesem Grund sind langzeitstabile Messungen sehr schwer umsetzbar. Die Messung von kleinen Kräften ist auch sehr schwer umsetzbar. Der Faktor Temperatur ist auch eine wesentliche Störquelle und muss beachtet werden.

**4.3.2.2 Dehnmessstreifen**

Dehnmessstreifen sind Federkörper. Dabei handelt es sich um elastisch verformbare Widerstände. Die Verformung durch Längung oder Stauchung und die daraus resultierende Veränderung der elektrischen Widerstände sorgt für die Gewinnung eines elektrischen Spannungssignales. Dieses Spannungssignal kann durch eine Messbrückenschaltung generiert werden.

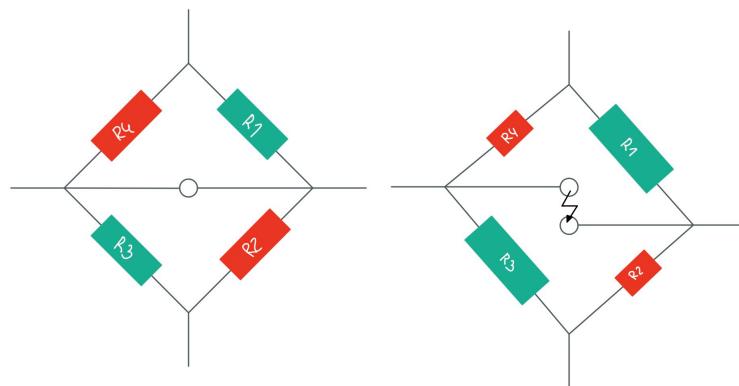


Abbildung 4.7: DMS-Prinzip

**Vorteil:**

Im Gegensatz zu der Variante mit dem Piezo-Sensor sind diese Messungen sehr langzeitstabil. Auch die Störgröße Temperatur kann besser kompensiert werden. Sensoren die mit dieser Methode umgesetzt werden sind sehr genau.

### Nachteil:

Diese Struktur wird für weiche Aufbauten eingesetzt. Je ausgeprägter die elastische Verformung, desto besser ist die Signalerfassung. Aus diesem Grund spielt das Thema Materialermüdung und Materialüberdehung eine kritische Rolle. Schnelle und hochfrequente Messungen sind aufgrund der niedrigen Eigenfrequenz schwer umsetzbar.

### 4.3.2.3 Fazit

Der Kraftsensor wäre in der Gehhilfe an zwei Positionen denkbar. Im untenstehenden Bild wird dargestellt, wo welcher Sensor geeignet wäre.

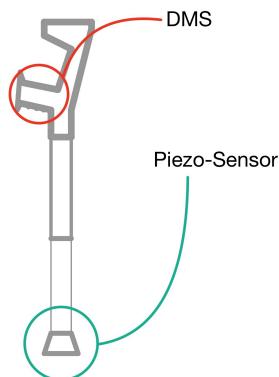


Abbildung 4.8: Position des Sensors

Der Dehnmessstreifen würde im Griff angebracht werden, dort hat der Sensor am meisten Verformung und liefert die besten Messungen.

Der Piezo-Sensor würde im Fuß der Gehhilfe montiert werden, weil dort am meisten Druck entsteht und man die besten Ergebnisse erreichen würde.

Die Entscheidung ist auf den Dehnmessstreifen gefallen. Ein DMS ist kostengünstiger und die Messungen sind langzeitstabil. Auch Temperaturunterschiede nehmen keinen großen Einfluss auf die Messung. Da man sich mit einer Gehhilfe sowohl im Freien als auch in Innenräumen bewegt, spielt die Temperatur eine Rolle.

Gegen den Piezo-Sensor haben die hohen Kosten und die Anfälligkeit auf Temperaturschwankungen gesprochen. Zudem sind diese Sensoren für hohe Kräfte besser geeignet.

### 4.3.3 Gewählter Sensor<sup>2</sup>

Der gewählte Sensor heißt Joy-it SEN-Pressure20 Berührungs-Sensor.

Die technischen Daten dieses Sensors basierend auf dem Prinzip eines DMS, werden unten aufgelistet.

- Ausgabe des Messergebnisses als Analoger Spannungswert
- Ansprechsensibilität ab 20 Gramm
- Genauigkeit  $\pm 2.5$
- Antwortzeit kleiner als 1 ms
- Pins: + / I / S
- Stromversorgung 3,3 V – 5 V
- Betriebstemperatur -20 – 60 °C
- Lebensdauer ca. 100.000 Messungen

### 4.3.4 DMS - Brückenschaltung<sup>3</sup>

Eine Brückenschaltung wird mit elektrischen Widerständen umgesetzt. Dabei werden diese Widerstände gemessen und mit bekannten Größen verglichen.

Der DMS formt Dehnung in eine proportionale Widerstandsänderung um.

Die Wheatstonesche Brückenschaltung wird in der DMS-Technik angewandt. Es gibt zwei Darstellungsoptionen, das linke Bild bzw. die linke Darstellungsart wird häufiger angewandt. Die rechte Darstellung ist vom Schaltauflbau identisch.

Eine Brückenschaltung kann in verschiedenen Arten umgesetzt werden. Man unterscheidet zwischen Viertelbrücke, Halbbrücke und Vollbrücke.

---

<sup>2</sup>vgl. Kraftsensor, (2021)

<sup>3</sup>vgl. Anwendung der Wheatstoneschen Brückenschaltung, (o. D.)

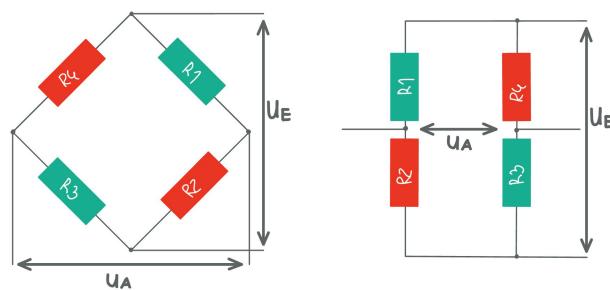


Abbildung 4.9: Brückenschaltung

#### 4.3.4.1 Viertelbrücke

Diese Art von Brückenschaltung ist die häufigste und einfachste DMS-Schaltung. Der ausgewählte Sensor verwendet ebenfalls diese Art von Schaltung. Das kann man aus der Beschriftung des Sensors schließen, denn es wird nur ein Widerstand beschrieben.

Die elektrischen Widerstände haben alle den selben Wert.

$$R = R_1$$

#### 4.3.4.2 Halbbrücke

Die Halbbrücke beteiligt beide Brückenarme an der Messung.

$$R_{Brueckenarm1} = R_1$$

$$R_{Brueckenarm2} = R_2$$

Der Vorteil ist, dass sowohl positive als auch negative Widerstandswerte gleichzeitig gemessen werden können. Dadurch ergibt sich ein genaueres Ergebnis und Störquellen werden minimiert.

#### 4.3.4.3 Vollbrücke

Bei einer Vollbrücke sind alle Brückenarme beteiligt.

Der Vorteil ergibt sich durch genauere Messungen und die Verminderung von Störquellen.

### 4.3.5 Code zur Umsetzung

```
1 int KraftSensorPin = 0;  
2  
3 void setup() {  
4     Serial.begin(9600);  
5 }  
6  
7 void loop() {  
8     val = analogRead(25); // Auslesen des Sensorwertes (zwischen 0 und 4095)  
9     Weight = map(val,0,4095,0,20); // "mappen", also ins Verhältnis setzen  
    des Sensorwertes (0-4095) zu 0-20kg  
10  
11    Serial.println("gemessene Kraft:");  
12    Serial.println(val);  
13    Serial.println(Weight);  
14 }
```

Listing 4.1: Kraftsensor

### 4.3.6 Anschluss des Sensors

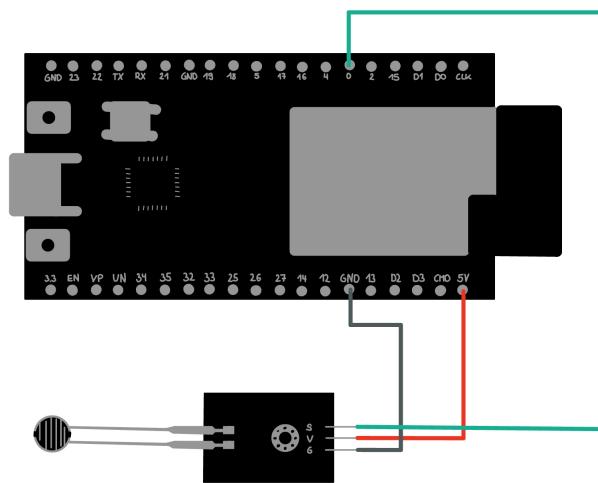


Abbildung 4.10: Anschluss des Sensors

## 4.4 Kommunikation mit der App

### 4.4.1 Anforderungen an die Kommunikation

Die Gehhilfe soll in der Lage sein Schrittzählerdaten, Kraftdaten und Sturzdaten an die Handy App senden zu können. Geplant ist, dass die Handy App eine Anfrage sendet und die Gehhilfe anschließend Daten zurücksendet.

### 4.4.2 Arten der Kommunikation

Es gibt verschiedene Arten wie ein Microcontroller mit einem anderen Microcontroller oder einem Mobiltelefon kommunizieren kann.

#### 4.4.2.1 NFC - Near Field Communication<sup>4</sup>

Wie man aus der Überschrift bereits entnehmen kann, handelt es sich bei NFC um eine Kommunikation, die eine Übertragungsreichweite von ca. fünf bis zehn Zentimeter ermöglicht. Die Daten werden kontaktlos übertragen. Dies funktioniert, weil über eine Radio Frequency Identification Technologie gearbeitet wird. Die Standardfrequenz beträgt 13,56 MHz.

##### Vorteil:

Eine NFC Schnittstelle findet man in allen neuen Mobiltelefonen. Es werden keine eigenen Apps zur Kommunikation benötigt und NFC ist weltweit standardisiert.

##### Nachteil:

Um eine Verbindung herstellen zu können muss das Gerät mindestens zehn Zentimeter neben der Gehhilfe positioniert werden.

#### 4.4.2.2 WLAN - Wireless Local Area Network<sup>5</sup>

Die WLAN-Technik basiert auf einem kabellosen Netzwerk. Benötigt wird ein Router.

Wenn der Router aktiviert ist, Funkt er permanent. Die Frequenz die der Router bzw. die Technik

---

<sup>4</sup>vgl. NFC, (2022)

<sup>5</sup>vgl. WLAN, (2022)

verwendet liegt zwischen 2400 und 5725 MHz.

**Vorteil:**

Mit diesem System ist man an kein Kabel gebunden und es funktioniert über eine beliebig große Distanz.

**Nachteil:**

Dieses System funktioniert nur in Reichweite eines Routers. Wenn das Haus verlassen wird, werden keine Daten von der Gehhilfe an die Handy App gesendet. Zudem benötigt der Konsument einen Router der laufend Kosten verursacht.

#### 4.4.2.3 Bluetooth<sup>6</sup>

Bluetooth überträgt Daten via Funk. Es können verschiedene Reichweiten durch verschiedene Arten von Bluetooth-Geräten erzielt werden. Die Reichweite kann zwischen einem und hundert Meter variieren.

Der Name Bluetooth stammt von einem dänischen König namens Harald Blauzahn. Er verbündete Norwegen und Schweden und stellte somit eine Verbindung her. Erfunden hat der König die Bluetooth-Technik nicht, aber sie wurde trotzdem nach ihm benannt.

**Vorteil:**

Bluetooth weist einen niedrigen Energieverbrauch und eine niedrige Sendeleistung auf. Zudem ist dieses System nur gering Störempfindlich und der Aufenthaltsort spielt keine Rolle. Die Distanz kann mehrere Meter betragen.

**Nachteil:**

Die Nachteile halten sich in Grenzen, die Reichweite kann sowohl als Pro oder als Kontra angesehen werden und die Anzahl der Netzteilnehmer ist begrenzt.

---

<sup>6</sup>vgl. Bluetooth, (2022)

#### 4.4.2.4 Fazit

Wie in den Anforderungen bereits erwähnt spielen Ortsunabhängigkeiten und eine Distanz zwischen einem oder zwei Meter eine Rolle. Aus diesem Grund wurde Bluetooth ausgewählt. Dieses System ist weder Ortsabhängig, noch ist es auf einem Router angewiesen. Zudem wird die Vorgabe der Distanz erfüllt.

### 4.4.3 Arten von Bluetoothmodulen

Es gibt verschiedene Hersteller und Arten von Bluetoothmodulen. Da wir mit der Arduino IDE arbeiten, haben wir die gängigsten Module für diese Programmierumgebung verglichen. Für beide Module müssen Bibliotheken in der IDE eingebunden werden.

#### 4.4.3.1 nRF52840

Der nRF52840 befindet sich auf dem *ItsyBitsy* Board. Dieses Board beinhaltet einen Microcontroller.

In der Arduino Bibliothek gibt es Beispiel Programme um eine Kommunikation via Bluetooth herstellen zu können.

Der Aufbau eines solchen Modules ist immer gleich. Es wird eine Antenne genutzt und anschließend via Funkwellen gesendet.

#### 4.4.3.2 ESP32

Das ESP32 Modul ist gleich aufgebaut. Auch hier findet man einen integrierten Microcontroller. Auch die Beispiel Programme kann man in der Arduino IDE finden.

Der Aufbau bzw. das Prinzip der Funktionsweise ist ebenfalls gleich.

#### 4.4.3.3 Fazit

Die Entscheidung ist auf den ESP32 gefallen. Dieses Modul beinhaltet den besseren Microcontroller und ist kostengünstiger. Zudem wurde zu Beginn mit dem nRF52480 gearbeitet und es sind technische Probleme aufgetreten. Programme konnten nur einmal ausgeführt werden.

#### 4.4.4 Code zur Kommunikation

```
1 #include "BluetoothSerial.h"
2 #include "math.h"
3 #include "Wire.h"
4
5 #if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
6 #error Bluetooth is not enabled! Please run 'make menuconfig' to and
    enable it
7 #endif
8
9 char buf[25]; // Mit diesem String getestet: "180#"
10 char* ptr = buf;
11 byte j;
12
13 BluetoothSerial SerialBT;
14
15 void setup() {
16     Serial.begin(9600);
17     SerialBT.begin("ESP32test"); //Bluetooth device name
18     Serial.println("Bluetooth Modul kann verbunden werden!");
19 }
20
21 void loop() {
22     // Eingang vom Bluetooth Modul auslesen
23     if (SerialBT.available()) {
24         buf[j] = SerialBT.read();
25         j++;
26     }
27
28
29     // Bei # in der Anfrage, TestString zurück senden
30     //if (buf[j-1]=='#') {
31     //    SerialBT.println("SR70;KR50;ST0;#");
32     //    // Format des Strings: Schritte
33     //    // 70 Trennzeichen Maximale Kraft 50kg ; 0 Stürze ; # zur Prüfung ob
34     //    // ganzer String eingeganen ist
```

```
32     // }  
33  
34     // Bei # in der Anfrage, Werte zurück senden  
35     if (buf[j-1]=='#') {  
36         // SerialBT.println("SR" + Steps + ";" + "KR" Krafteinwirkung + ";" +  
37         // "ST" + Falls + ";" + "#");  
38         SerialBT.print("SR");  
39         SerialBT.print(Steps);  
40         SerialBT.print(";");  
41         SerialBT.print("KR");  
42         SerialBT.print(Krafteinwirkung);  
43         SerialBT.print(";");  
44         SerialBT.print("ST");  
45         SerialBT.print(Falls);  
46         SerialBT.print(";");  
47         SerialBT.print("#");  
48         Steps = 0; // Schritte wieder auf 0 setzen, um von vorne zu Zählen  
49     }  
50 }
```

Listing 4.2: Bluetooth Kommunikation

#### 4.4.5 App zum Testen

Für das Projekt wurde das ESP32 Bluetooth Modul verwendet. Zum Testen gibt es eine eigene App mit dem Namen *Serial Bluetooth Terminal*. Am unteren Bild sieht man, dass eine Anfrage mit einem # zum Schluss versendet wurde und der gewünschte String zurückgesendet wurde. Zudem wurden alle Daten anschließen auf null zurückgesetzt.

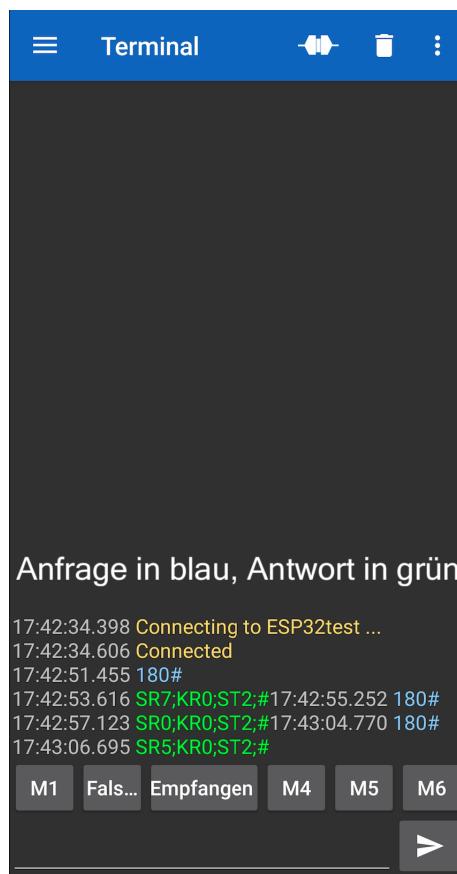


Abbildung 4.11: Serial Bluetooth Terminal

## 4.5 Microcontroller

### 4.5.1 Was ist ein Microcontroller?<sup>7</sup>

Ein Microkontroller ist ein geschlossenes System, das auf einen Chip aufgebracht wird. Es kommen Mikroprozessoren zum Einsatz, die wesentlich weniger komplex und dynamisch sind, als sie in einer CPU vorkommen.

Dieses System führt ein Basisprogramm periodisch aus. Daher eignen sich solche MCUs, wie sie auch genannt werden, perfekt für vorprogrammierte und automatisierte Aufgaben. Diese Wiederholungen werden üblicherweise mit einer zeitgesteuerten Schleife umgesetzt.

#### 4.5.1.1 Arten

Im Wesentlichen kann man MCUs in drei Gruppen aufteilen. Es gibt zwar dutzende Hersteller, trotzdem werden nur drei Arten unterschieden.

<sup>7</sup>vgl. Microkontroller, (2022)

1. 8-Bit-Microkontroller
2. 16-Bit-Microkontroller
3. 32-Bit-Microkontroller

Der größte Unterschied ist die mathematische Genauigkeit und Geschwindigkeit. Ein 8-Bit-Microkontroller benötigt eine größere Anzahl an Buszugriffen und mehr Anweisungen, um eine Berechnung durchzuführen. 16-Bit-Microkontroller bzw. 32-Bit-Microkontroller kommen schneller zum Ausgangsverhalten.

#### **4.5.1.2 Architekturen**

Es gibt zwar nur die drei beschriebenen Kernarten, die Auswahl in Bereich Architekturen und Marken ist jedoch sehr groß.

Beim Kauf sollte man auf die Maschinensprache und auf das Kernbuild achten.

#### **4.5.1.3 Funktionsweiße**

Ein Microkontroller ähnelt einem kleinen Computer, der weniger anspruchsvoll ist.

MCUs erkennen externe Signale über Kommunikationprotokolle und reagieren darauf. Es kann sich um Umweltsensoren, wie den Kraftsensor, USB oder eine Touch Response handeln.

MCUs reagieren, bei richtiger Programmierung, schnell auf bestimmte Eingaben oder Signaldetektionen. Daraus folgt eine große Palette an Funktionen und Anwendungen, die ausgeführt werden können.

### **4.5.2 Anforderungen an den Microcontroller**

Benötigt werden zwei Sensoren. Die ersten Sensoren sind ein Gyrosensor und ein Accelerometer (Block 1), diese zwei Sensoren befinden sich im MPU6050. Der zweite bzw. dritte Sensor ist ein DMS (Block 2). Die Sensoren werden mit dem Microcontroller (Block 3) verbunden. Mittels Anweisungen werden die eingegangenen Signale in Schritte, Stürze und Kräfte umgewandelt. Im Block 4 handelt es sich um die Sturzerkennung. Genauereres über die Sturzerkennung und über den Schrittzähler kann man im Teil C – Sensorik nachlesen. Der zweite Wert ist die Kraft- bzw. die

Belastungsanalyse (Block 5). Die letzten Blöcke beinhalten einen Schrittzähler (Block 6) und die Anzahl der Stürze (Block 7). Anschließend können diese Werte über ein Bluetooth-Modul (Block 8) an die Handy-App (Block 9) gesendet werden.

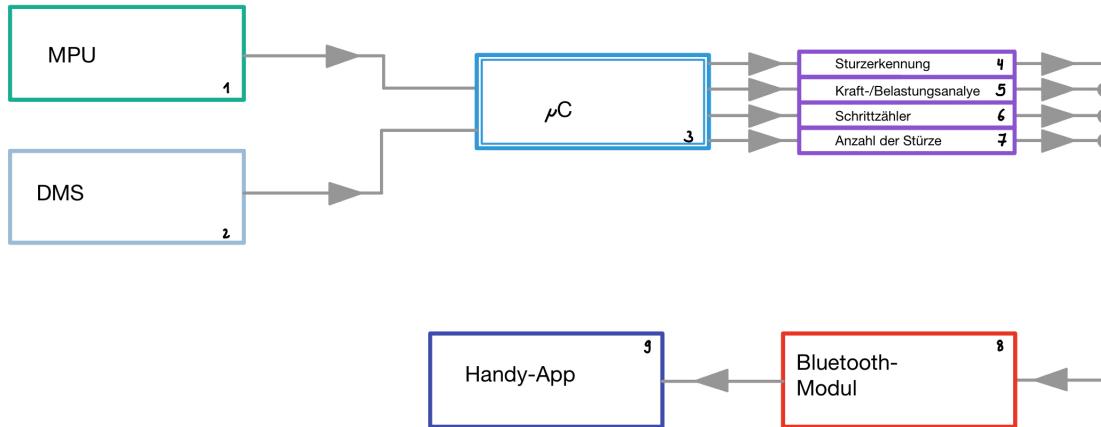


Abbildung 4.12: Anforderung Microcontroller

Um besser erfassen zu können, welche Pins benötigt werden, wurde die nachstehende Tabelle erstellt.

Verwendung	Schnittstelle am $\mu$ C	Kommentar
MPU6050	SCL u. SDA	Schrittzähler u. Sturzerkennung
Piezo-Sensor	Analoger Eingang (AI)	Kraftmessung

Tabelle 4.3: Pins

Somit sind die Mindestanforderungen an unseren  $\mu$ C nicht sehr Anspruchsvoll.

Zusammengefasst ergibt dies folgende Mindesanforderung:

Art	Genaue Bezeichnung	Anzahl Lines
AI	A0	1
SCL	A22	1
SDA	A21	1
Ground	GND	2
VCC	3.3 und 5	2

Tabelle 4.4: Verwendete Pins

### 4.5.3 Vergleich ATmega328P mit ESP32

Im Unterricht wurde das Programmieren des ATmega328P gelehrt. Nun wird verglichen, ob der ESP32 auch verwendet werden kann. Dieser Chip beinhaltet bereits ein Bluetooth Modul und es müsste nicht extern dazu gekauft werden. Die Programmierumgebung Arduino IDE würde auf beiden Chips funktionieren.

Anbei eine Tabelle mit den Daten im Vergleich.

Daten	ATmega328P <sup>8</sup>	ESP32 <sup>9</sup>
<b>Speicher</b>	32 kByte	4 MByte
<b>RAM</b>	2 kByte	512 kByte
<b>EEPROM</b>	1024 Byte	-
<b>I/O Pins</b>	23	39
<b>PWM</b>	6	16
<b>IC</b>	1	2
<b>SPI</b>	2	3
<b>ADC</b>	6	18
<b>UART</b>	1	3
<b>Versorgungsspannung</b>	+1,8 ... +5,5 VDC	+3,3 ... +5,5 VDC
<b>Temperaturbereich</b>	-40 ... +85 °C	-40 ... +125 °C

Tabelle 4.5: C Vergleich

### 4.5.4 Fazit

Der Abgleich mit den Anforderungen zeigt, dass der ATmega328P ausreichen würde und der ESP32 überqualifiziert für diese Aufgabe ist. Da die Kosten des ESP32 fast identisch mit den Kosten des ATmega328P sind und der ESP32 ein Bluetooth Modul beinhaltet, wurde der ESP32 verwendet und für das Diplomprojekt eingesetzt. Die technischen Daten sind deutlich besser und die Anforderungen sind in jedem Fall abgedeckt. Auch die Arduino IDE kann verwendet werden und es muss keine neue Programmierumgebung erlernt werden.

## 4.6 Akkusystem

### 4.6.1 Akku Allgemein<sup>10</sup>

Bei einem Akku handelt es sich um einen elektrischen Energiespeicher. Diesen Speicher kann man wieder aufladen. Bei diesem Ladeprozess wird elektrische in chemische Energie umgewandelt.

Jedes Laden des Akkus nennt man Zyklus, je nach Art überlebt der Akku 100 bis ca. 1000 Zyklen.

<sup>10</sup>vgl. Akku-Grundlagen, (2022)

Das Gegenstück zum Akku ist eine Batterie oder Primärzelle. Diese kann man nicht wieder aufladen. Dementsprechend wird eine Batterie nach Entnahme der Energie weggeworfen. Akkus sind demnach günstiger und umweltfreundlicher.

## **Grundbegriffe**

### **Zelle/Pack**

Die Zelle ist die kleinste Einheit die ein Akku besitzt. Die Reihenschaltung von solchen Zellen nennt man Pack. Eine einzelne Zelle reicht im praktischen Einsatz nicht aus, deswegen werden sie in Reihen geschalten.

### **Kapazität**

Die Einheit der Kapazität wird in Ampere-Stunden (Ah) oder Milliampere-Stunden (mAh) angegeben. Die Kapazität sagt aus, wie lange man etwas mit entsprechendem Strom beliefern kann. Ein Akku mit 3Ah (=3000mAh) kann ein Gerät, das 300mA Strom aufnimmt, zehn Stunden lang betreiben.

### **C**

Als C wird der Strom bezeichnet, welcher der Kapazität dividiert durch eine Stunde gleicht. Diese Angabe benötigt man für Lade- und Entladeströme. Ein Akku mit 3000mAh, könnte mit 4C entladen und mit 2C geladen werden. Dies würde bedeuten das der Akku mit bis zu 12A entladen und mit bis zu 6A geladen werden kann.

(Rechnung:  $C = 3000\text{mAh}/1\text{h} = 3\text{A}$ ;  $3\text{A} * 4\text{C} = 12\text{A}$ ;  $3\text{A} * 2\text{C} = 6\text{A}$ )

### **Spannungen**

Die Nennspannung eines Akkus ist eine nominelle Angabe. Das bedeutet, dass die Spannung bei einem vollgeladenen Akku deutlich höher und bei einem entladenen Akku deutlich geringer ist.

## 4.6.2 Anforderungen an den Akku

Folgende Anforderungen sollte der Akku erfüllen.

<b>Durchmesser</b>	max. 19mm
<b>min. Temperatur</b>	-10 °C
<b>max. Temperatur</b>	+40 °C
<b>Entladespannung</b>	2,0 ... 5 V
<b>Kapazität</b>	min. 1500mAh

Tabelle 4.6: Akku Anforderungen

## 4.6.3 Auswahl des Akkus

Der gewählte Akku heißt Beltrona 18650XH 2.54 Spezial-Akku 18650 Stecker<sup>11</sup>.

Er entspricht allen Anforderungen mit folgenden Werten.

<b>Durchmesser</b>	18,6 mm
<b>min. Temperatur</b>	-10 °C
<b>max. Temperatur</b>	+60 °C
<b>Entladespannung</b>	2,5 V
<b>Kapazität</b>	1500mAh

Tabelle 4.7: Ausgewählter Akku

Um diesen Akku laden zu können, bzw. nutzen zu können, braucht der Benutzer den D1Z Battery

<sup>12</sup> Dieser sorgt dafür, dass man den Akku im normalen Netz laden kann. Dieser sorgt dafür, dass man den Akku im normalen Netz laden kann.

## 4.6.4 Ladeverfahren

### 4.6.4.1 Energy Harvesting

Mittels des Prinzips von Energy Harvesting oder auf Deutsch die Energie Ernte kann man kleine Mengen an Strom durch verschiedenste Verfahren generieren.

Der Vorteil bei diesem Verfahren ist, dass kein Stecker und kein Kabel benötigt wird. Zudem muss der Benutzer nicht an das Laden seines Gerätes denken.

---

<sup>11</sup>vgl. Akku, (2022)

<sup>12</sup>vgl. D1Z BATTERY, (2022)

## Bewegung

Bei diesem Verfahren werden Piezoelektrische Kristalle eingesetzt. Das System ist das Gleiche wie das, des Piezo-Sensors.

Nährer Informationen zum Thema Piezo siehe **4.3.2.1 Piezo-Sensor**.

## Licht

Miniarisierte Solarmodule werden eingesetzt, um aus Licht kleine Mengen Strom zu gewinnen.

## Temperaturdifferenzen

Um aus Temperatur Energie gewinnen zu können muss es eine Temperaturschwankung geben. Diese Technologie ist vielschichtig einsetzbar, der Mensch hat oftmals einen großen Temperaturunterschied zur Raumtemperatur, der Tag hat eine andere Temperatur als die Nacht. Es gibt sehr viele Beispiele für solch ein Verfahren.

## 4.6.4.2 Fazit

Das Prinzip hinter Energy Harvesting ist ein sehr effizientes und durchdachtes. Man könnte es in die Gehhilfe durch die Energiegewinnung aus Bewegungsabläufen sehr gut integrieren. Die Umsetzung ist aber sehr kompliziert und zeitaufwendig.

# 4.7 Speicherung der Daten

## 4.7.1 Anforderungen an die Speicherung

Daten sollen gespeichert werden, damit die Gehhilfe und die Handy-App nicht dauerhaft in Verbindung sein müssen. Diese dauerhafte Bluetooth Verbindung würde sehr viel Akku verbrauchen und diesen Verbrauch sollte man so weit wie möglich eindämmen.

Es werden drei Werte zwischen Handy-App und Gehhilfe gesendet. Das Ziel ist es, dass die Daten nur alle 60 Minuten abgefragt werden. Nun wird zu jedem Wert ein System erstellt, um dieses Ziel erfüllen zu können.

## Schrittzähler

Der Schrittzähler zählt bei jedem Schritt um eine Stelle nach oben. Nach Abfrage wird die Variable, wo dieser Wert gespeichert wird, wieder auf eins zurückgesetzt.

## Stürze

Auch die Stürze werden mittels eines Counters nach oben gezählt und bei Datenaustausch wieder auf eins zurückgesetzt.

## Kraftanalyse

Die Kraftanalyse filtert einen positiven Wert pro Stunde, aus diesem Grund ist die Speicherung schwieriger. Aber es soll das System eines Ringspeichers eingesetzt werden. Genaueres siehe Kapitel Ringspeicher.

### 4.7.2 Ringspeicher

Bei einem Ringspeicher werden Daten über einen gewissen Zeitraum abgespeichert.

Es gibt zwei Pointer, einen Read und einen Write Pointer. Der Write Pointer schreibt Werte in den Speicher und der Read Pointer liest Werte aus dem Speicher aus. Wichtig ist, dass sich diese Zeiger/Pointer nicht gegenseitig überschreiben.

Mittels dieses Systems kann ein Speicher 60 Minuten lang befüllt und anschließend wieder erneut belegt werden. Der Datenspeicher kann die Werte jederzeit auslesen und den höchsten Kraftwert an die Handy-App senden.

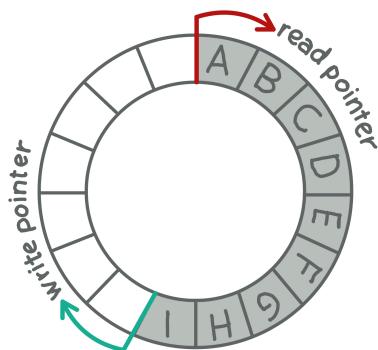


Abbildung 4.13: Ringspeicher

## 5 Anhang

## 5.1 Gesamter Programmcode

```
1 // -----
2 // Diplomarbeit: iWalk
3 // -----
4 // Erstellt von Georg Kaufmann u. Paul Resch
5 // -----
6
7 #include "BluetoothSerial.h"
8 #include "math.h"
9 #include "Wire.h"
10 #include <Adafruit_MPU6050.h>
11 #include <Adafruit_Sensor.h>
12
13 #if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
14 #error Bluetooth is not enabled! Please run 'make menuconfig' to and
15     enable it
16
17 BluetoothSerial SerialBT;
18
19 char buf[25]; // Mit diesem String getestet: "180#"
20 char* ptr = buf;
21 byte j;
22 int Steps = 0;
23 int Falls = 0;
24 int Weight = 0;
25 const long interval = 30000;
26 unsigned long ptime = 0;
27
28 Adafruit_MPU6050 mpu;
29
30 int val;
31
32 void setup() {
33     Serial.begin(9600);
```

```
34     SerialBT.begin("ESP32test"); //Bluetooth device name
35     Serial.println("Bluetooth Modul kann verbunden werden!");
36
37     // MPU:
38     if (!mpu.begin()) {
39         Serial.println("Failed to find MPU6050 chip");
40         while (1) {
41             delay(10);
42         }
43     }
44     Serial.println("MPU6050 Found!");
45
46     mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
47     mpu.setGyroRange(MPU6050_RANGE_500_DEG);
48     mpu.setFilterBandwidth(MPU6050_BAND_44_HZ);
49
50     pinMode(25, INPUT);
51 }
52
53 // the loop routine runs over and over again forever:
54 void loop() {
55     // -----
56     // Schritte:
57     // -----
58     steps();
59
60     // -----
61     // Kraftmessung:
62     // -----
63     weight();
64
65     // -----
66     // Stürze:
67     // -----
68     fall();
69
```

```
70     delay(500);
71
72     // -----
73     // Bluetooth Connection:
74     // -----
75     bluetooth();
76
77 }
78
79 void steps() {
80
81     // Get new sensor events with the readings
82     sensors_event_t a, g, temp;
83     mpu.getEvent(&a, &g, &temp);
84
85     double gyro_x = g.gyro.x;
86     double gx = 0;
87     if(gyro_x > 0.1 && gyro_x < 0.7) {
88         gx = 1;
89     }
90     double gyro_y = g.gyro.y;
91     double gy = 0;
92     if(gyro_y > 0.1 && gyro_y < 0.7) {
93         gy = 1;
94     }
95     double gyro_z = g.gyro.z;
96     double gz = 0;
97     if(gyro_z > 0.1 && gyro_z < 0.6) {
98         gz = 1;
99     }
100
101    double accel_x = a.acceleration.x;
102    double ax = 0;
103    if(accel_x > 0.8 && accel_x < 1.5) {
104        ax = 1;
105    }
```

```
106     double accel_y = a.acceleration.y;
107     double ay = 0;
108     if(accel_y > 1.4 && accel_y < 5) {
109         ay = 1;
110     }
111     double accel_z = a.acceleration.z;
112     double az = 0;
113     if(accel_z > 6 && accel_z < 9) {
114         az = 1;
115     }
116
117     if(/*gx == 1 && gy == 1 && gz == 1 &&*/ ax == 1 && ay == 1) {
118         Steps++;
119     }
120     Serial.println("-----");
121     Serial.println("Schritte:");
122     Serial.println(Steps);
123 }
124
125 void weight() {
126     val = analogRead(25); // Auslesen des Sensorwertes (zwischen 0 und 4095)
127     Weight = map(val,0,4095,0,20); // "mappen", also ins Verhältnis setzten
128         des Sensorwertes (0-4095) zu 0-20kg
129     Serial.println("gemessene Kraft:");
130     Serial.println(val);
131     Serial.println(Weight);
132 }
133
134 void fall() {
135     // -----
136     // Stürze:
137     // -----
138
139     sensors_event_t a, g, temp;
140     mpu.getEvent(&a, &g, &temp);
```

```
141
142     unsigned long t = millis();
143
144     if(t - ptime >= interval) {
145         if((a.acceleration.x < -9.5 || a.acceleration.x > 9.5 || a.
146             acceleration.y < -9.5 || a.acceleration.y > 9.5) && Weight < 1) {
147             Falls++;
148             SerialBT.println("HELP ME");
149             ptime = t;
150         }
151
152         Serial.println("Stürze:");
153         Serial.println(Falls);
154         Serial.println("-----");
155     }
156
157 void bluetooth() {
158     // Eingang vom Bluetooth Modul auslesen
159     if (SerialBT.available()) {
160         buf[j] = SerialBT.read();
161         j++;
162     }
163
164
165     // Bei # in der Anfrage, TestString zurück senden
166     //if (buf[j-1]=='#') {
167     //    SerialBT.println("SR70;KR50;ST0;#"); // Format des Strings: Schritte
168
169     //    70 Trennzeichen Maximale Kraft 50kg ;
170
171     //    // . . . . . . . . . 0 Stü
172     //    rze ; # zur Prüfung ob ganzer String eingeganen ist
173     //}
174
175
176     // Bei # in der Anfrage, Werte zurück senden
177     if (buf[j-1]== '#' ) {
```

```
173     // SerialBT.println("SR" + Steps + ";" + "KR" Krafteinwirkung + ";" +
174     // "ST" + Falls + ";" + "#");
175     SerialBT.print("SR");
176     SerialBT.print(Steps);
177     SerialBT.print(";");
178     SerialBT.print("KR");
179     SerialBT.print(Weight);
180     SerialBT.print(";");
181     SerialBT.print("ST");
182     SerialBT.print(";");
183     SerialBT.print("#");
184
185     Steps = 0;
186     Falls = 0;
187 }
188 }
```

Listing 5.1: Gesamter Programmcode

## 5.2 Anschluss der Sensoren

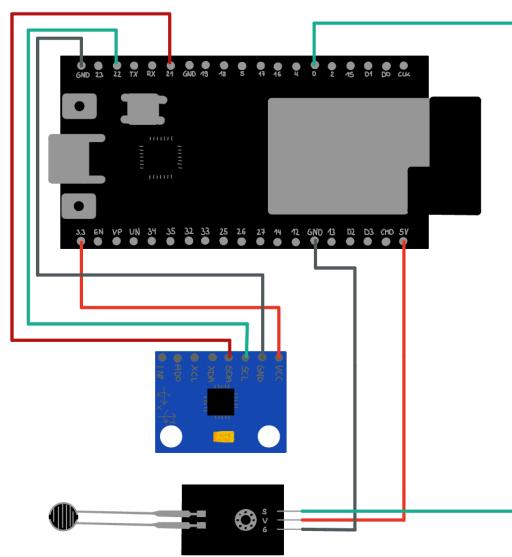


Abbildung 5.1: Gesamter Anschlussplan

## 5.3 Teil A - Mechanik

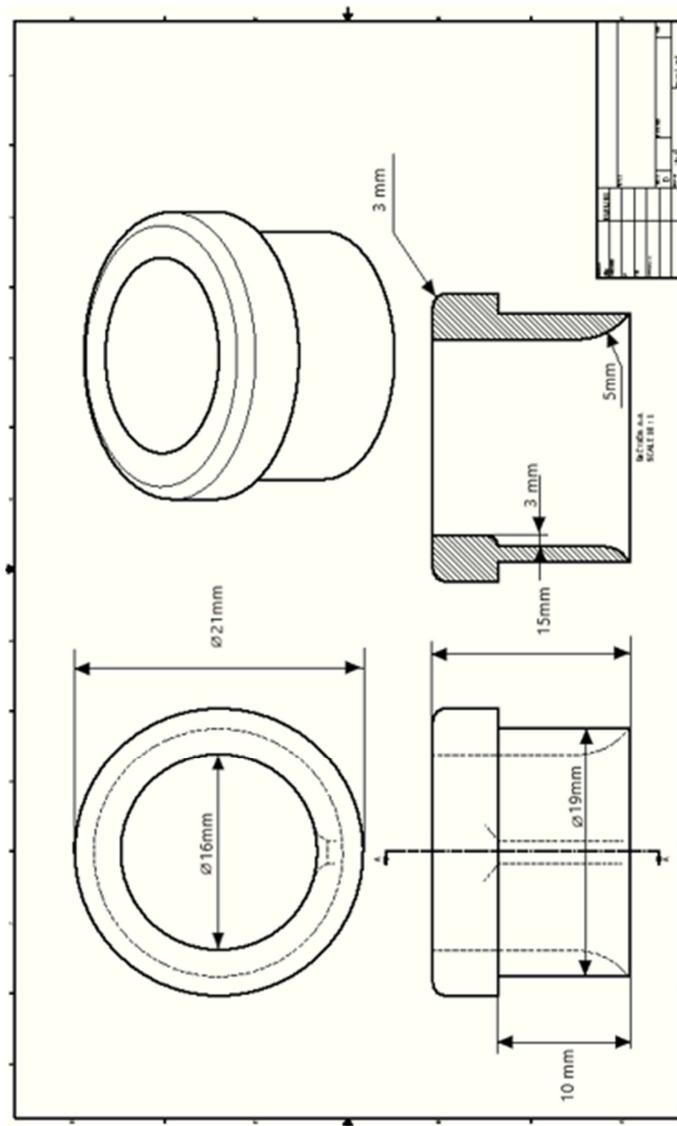


Abbildung 5.2: Werkstättenzeichnung Akku Halterung

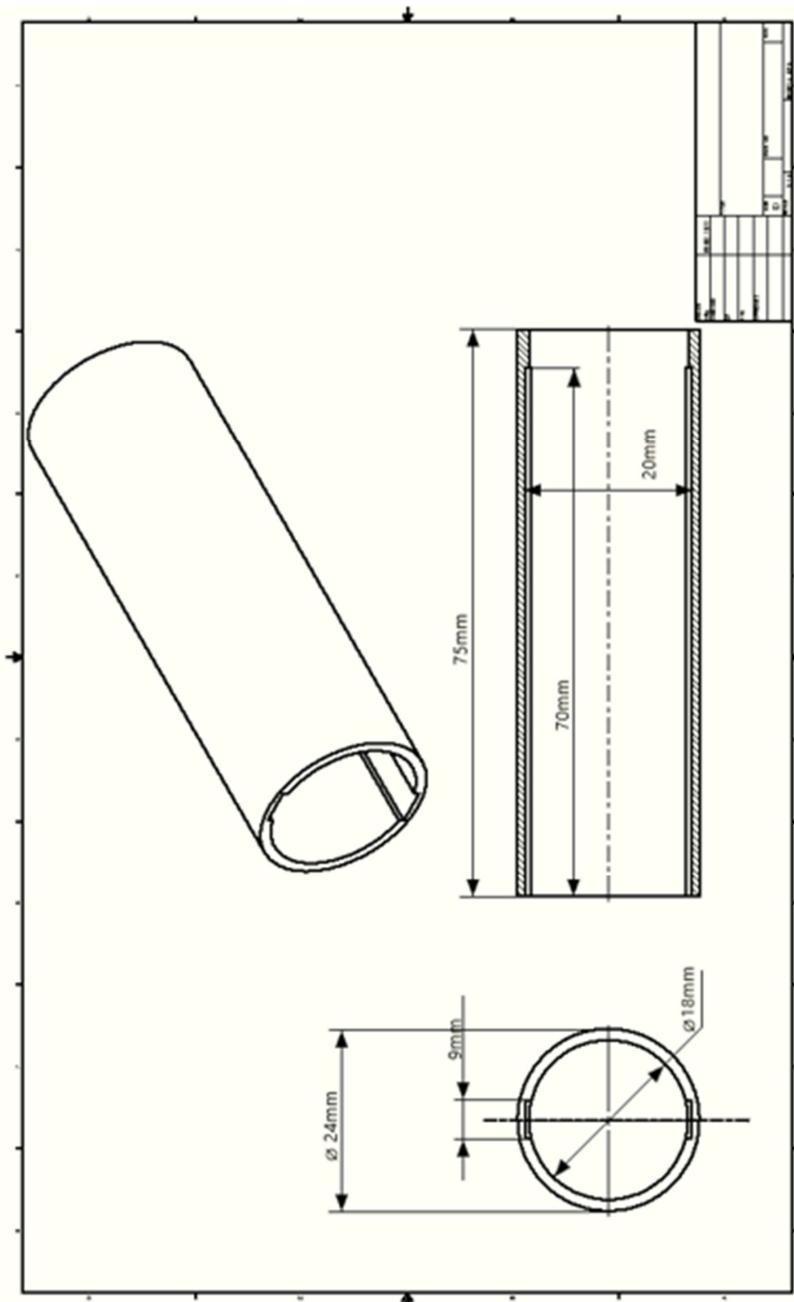


Abbildung 5.3: Werkstättenzeichnung ESP32 Halterung

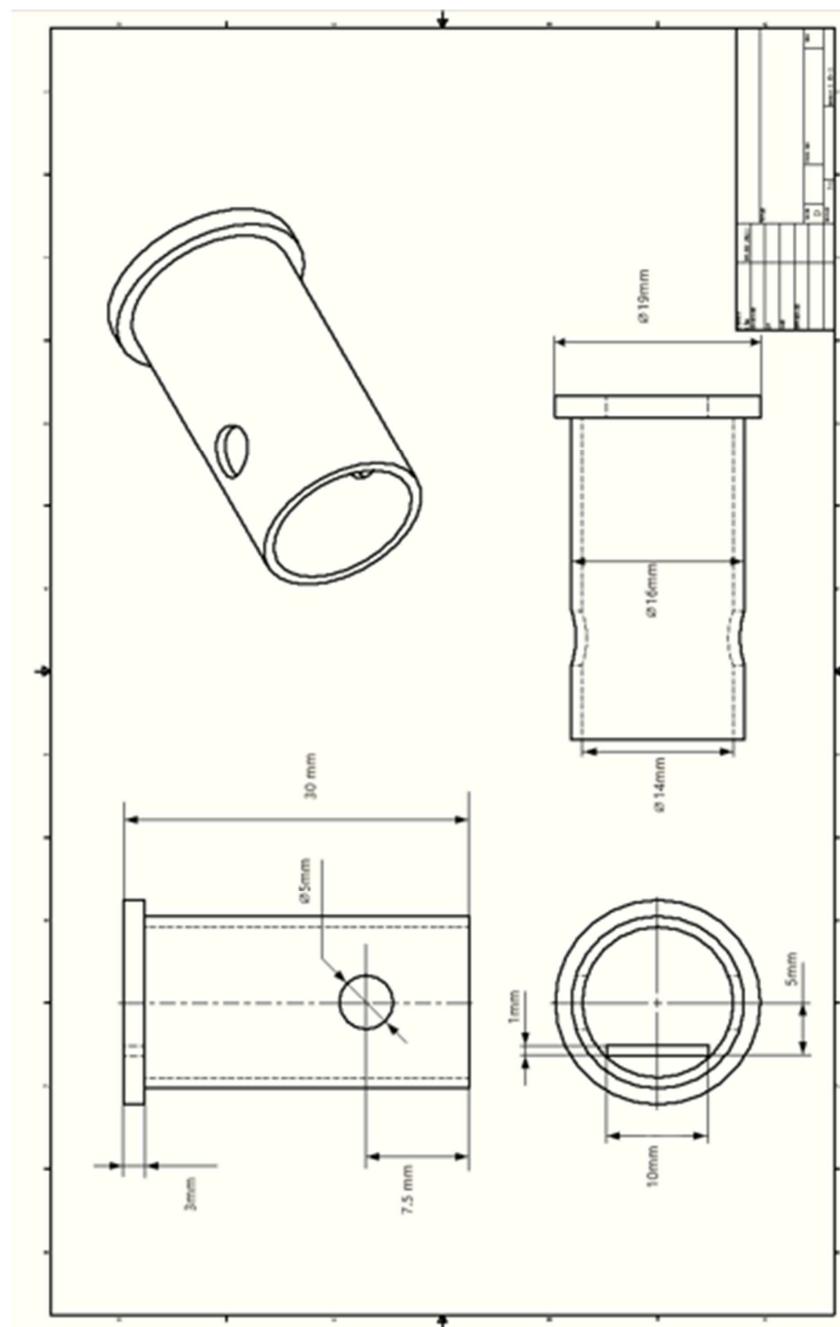


Abbildung 5.4: Werkstättenzeichnung Stoppel Rohr 2



## 6 Zeitaufzeichnung

## 6.1 Kilian Wade - Teil A

Datum	Stunden	Tätigkeit
06.09.21 - 10.09.21	6	Konzeptfindung
11.09.21	2	Besprechung der Arbeitsbereiche
10.10.21 - 28.02.22	102	CAD Konstruktion
17.11.21 - 26.02.22	19	Berechnung
20.09.21 - 01.03.22	57	Projektdokumentation
20.09.21 - 01.03.22	12	Besprechungen
07.03.22	3	Abschlussbesprechung
<b>Summe:</b>	<b>198 h</b>	

Tabelle 6.1: Zeitaufzeichnung - Wade

## 6.2 Christoph Sebernegg - Teil B

Datum	Stunden	Tätigkeit
06.09.21 - 10.09.21	6	Konzeptfindung
11.09.21	2	Besprechung der Arbeitsbereiche
20.09.21	4	Recherchieren Kommunikation Arduino mit App
22.09.21	3	Recherchieren der Latexformatierung
15.10.21 - 17.10.21	5	Design der APP
29.10.21 - 25.02.22	91	Programmierung der App
20.09.21 - 01.03.22	57	Projektdokumentation
20.09.21 - 01.03.22	12	Besprechungen
07.03.22	3	Abschlussbesprechung
<b>Summe:</b>	<b>183 h</b>	

Tabelle 6.2: Zeitaufzeichnung - Sebernegg

## 6.3 Paul Resch - Teil C

Datum	Stunden	Tätigkeit
06.09.21 - 10.09.21	6	Konzeptfindung
11.09.21	2	Besprechung der Arbeitsbereiche
08.10.21 - 10.10.21	12	Bauteilfindung
09.12.21 - 22.12.21	12	Recherche Schrittzähler
04.12.21 - 06.01.21	10	Recherche Sturzerkennung
16.10.21 - 22.02.22	21	Recherche Sensorik
20.10.21	6	Arduino DIE einrichten
07.01.22 - 07.01.22	16	Programmieren Schrittzähler
07.01.22 - 13.01.22	18	Programmieren Sturzerkennung
20.09.21 - 01.03.22	64	Projektdokumentation
20.09.21 - 01.03.22	12	Besprechungen
07.03.22	3	Abschlussbesprechung
<b>Summe:</b>	<b>182 h</b>	

Tabelle 6.3: Zeitaufzeichnung - Resch

## 6.4 Georg Kaufmann - Teil D

Datum	Stunden	Tätigkeit
06.09.21 - 10.09.21	6	Konzeptfindung
11.09.21	2	Besprechung der Arbeitsbereiche
08.10.21 - 10.10.21	12	Bauteilfindung
22.12.21 - 14.02.22	11	Recherche Bluetooth
13.10.21 - 15.01.22	11	Recherche Kraftmessung
16.10.21 - 22.02.22	19	Recherche Sensorik
20.10.21	6	Arduino DIE einrichten
21.12.21 - 14.01.22	17	Kommunikation via Bluetoothmodul
23.12.21 - 28.12.21	19	Programmieren Kraftmessung
20.09.21 - 01.03.22	63	Projektdokumentation
20.09.21 - 01.03.22	12	Besprechungen
07.03.22	3	Abschlussbesprechung
<b>Summe:</b>	<b>181 h</b>	

Tabelle 6.4: Zeitaufzeichnung - Kaufmann



# 7 Abbildungsverzeichnis

0.1	Übersicht Gehhilfe	3
0.2	Übersicht Zeitpläne	5
1.1	Die Schemaskizze der gesamten Krücke mit Einteilung in Teile Eins bis Vier	9
1.2	Schemaskizze Krückenkopf Teil Eins	10
1.3	Schemaskizze Rohr Eins Teil Zwei	11
1.4	Schemaskizze Rohr Zwei Teil Drei	12
1.5	Schemaskizze Gumminoppe Teil Vier	13
1.6	Logo Inventor	14
1.7	Gesamte Krücke mit Inventor 1.0	15
1.8	Krückenkopf mit Inventor	16
1.9	Rohr Eins mit Inventor	17
1.10	Rohr Zwei mit Inventor	18
1.11	Gumminoppe mit Inventor	19
1.12	Rohrverbinder mit Inventor	20
1.13	Gesamte Krücke 2.0	21
1.14	Microcontroller ESP 32	23
1.15	Microcontroller Halterung	23
1.16	Microcontroller Halterung mit ESP32	23
1.17	Beltrona 18650XH 2.54 Spezial-Akku	24
1.18	Akkuhalterung	24
1.19	Akku und Akkuhalterung	25
1.20	Kraftsensor	25
1.21	Rohr Zwei mit Noppenhalterung und Berührungs-sensor	26

1.22 Stoppel an Rohr Zwei . . . . .	26
1.23 Gumminoppe Hülse Einkerbung . . . . .	27
1.24 Gumminoppe Plateau . . . . .	27
1.25 Rohr Zwei mit Gumminoppe . . . . .	28
1.26 Krückenkopf sowie Rohr Eins mit Innenleben . . . . .	29
1.27 Gumminoppe und Rohr Zwei mit Innenleben . . . . .	29
1.28 dynamische Gewichtsmessung mit einer Waage, Marke Korona . . . . .	30
1.29 Belastung Krückenkopf . . . . .	31
1.30 Belastung der zehn Bohrungen . . . . .	32
1.31 belastete Bohrung . . . . .	33
1.32 Belastung Rohr Zwei Unterseite . . . . .	34
1.33 Belastung Gumminoppe Rohr Zwei . . . . .	35
1.34 Belastung Gumminoppe Boden . . . . .	36
1.35 Freimachskizze Krückenkopf Biegung . . . . .	37
1.36 Querschnitt des Rohres an der Bohrung . . . . .	40
1.37 Verbindung Rohr Eins, Zwei und Verbinder . . . . .	44
1.38 Verbindung Rohr Eins, Zwei und Verbinder mit Projektion . . . . .	45
1.39 Krückenkopf Polyamid belastet mit 55 Kilogramm . . . . .	47
1.40 Krückenkopf Polyoxymethylen belastet mit 55 Kilogramm . . . . .	47
 2.1 Android Studio . . . . .	52
2.2 AsyncTask . . . . .	55
2.3 Activity Bluetoothverbindung . . . . .	57
2.4 Logo . . . . .	58
2.5 Startbildschirm . . . . .	59
2.6 Schritte . . . . .	60
2.7 Stürze Activity . . . . .	61
2.8 Belastungsanalyse Activity . . . . .	62
2.9 Notfallkontakt Dialogfenster . . . . .	63
2.10 Screenshot Ladebildschirm . . . . .	64

2.11 Screenshot Startbildschirm . . . . .	65
2.12 Screenshot Schritte Activity . . . . .	66
2.13 Screenshot Stürze Activity . . . . .	67
2.14 Screenshot Belastungsanalyse Activity . . . . .	68
2.15 Screenshot Dialogfenster . . . . .	77
3.1 Darstellung piezoresistiver Beschleunigungssensor . . . . .	86
3.2 vereinfachte Darstellung eines Kondensators . . . . .	87
3.3 Aufbau und Prinzipschaltung . . . . .	88
3.4 Funktionsprinzip . . . . .	89
3.5 GY-521 MPU-6050 . . . . .	91
3.6 Veranschaulichung der Übersetzung in Maschinensprache . . . . .	95
3.7 Einstellungen für das ursprünglich ausgewählte Board . . . . .	97
3.8 Sensor am Microcontroller anschließen . . . . .	101
3.9 Beispielhafte Ausgabe der Programms . . . . .	102
3.10 Simulation eines Schrittes . . . . .	103
3.11 Ansicht von Oben . . . . .	104
3.12 Ansicht von Links . . . . .	104
4.1 Mitglieder der DA . . . . .	115
4.2 Organisationsform . . . . .	115
4.3 Zeitpläne . . . . .	116
4.4 Normalgewicht einer Frau . . . . .	117
4.5 Normalgewicht eiens Mannes . . . . .	117
4.6 Piezo-Prinzip . . . . .	118
4.7 DMS-Prinzip . . . . .	119
4.8 Position des Sensors . . . . .	120
4.9 Brückenschaltung . . . . .	122
4.10 Anschluss des Sensors . . . . .	123
4.11 Serial Bluetooth Terminal . . . . .	129
4.12 Anforderung Microcontroller . . . . .	131

4.13 Ringspeicher . . . . .	136
5.1 Gesamter Anschlussplan . . . . .	144
5.2 Werkstättenzeichnung Akku Halterung . . . . .	145
5.3 Werkstättenzeichnung ESP32 Halterung . . . . .	146
5.4 Werkstättenzeichnung Stoppel Rohr 2 . . . . .	147

# 8 Tabellenverzeichnis

1.1	Messwerte . . . . .	30
1.2	Materialien und Eigenschaften . . . . .	46
1.3	Wichtigste Daten der Kunststoffe . . . . .	46
1.4	Daten der Metalle . . . . .	48
1.5	Wichtigste Daten von TPU und TPR . . . . .	49
2.1	Android Versionen . . . . .	53
3.1	Pins . . . . .	93
3.2	Veranschaulichung der Zahlensysteme . . . . .	94
3.3	Sensor am Microcontroller anschließen . . . . .	102
3.4	Simulation eines Schrittes . . . . .	105
3.5	Ruheposition des Sensors . . . . .	105
3.6	90° nach Links gekippt . . . . .	109
3.7	90° nach Vorne gekippt . . . . .	109
4.1	Übersicht Gewicht . . . . .	117
4.2	Anforderungen Kraftsensor . . . . .	118
4.3	Pins . . . . .	131
4.4	Verwendete Pins . . . . .	131
4.5	C Vergleich . . . . .	132
4.6	Akku Anforderungen . . . . .	134
4.7	Ausgewählter Akku . . . . .	134

6.1	Zeitaufzeichnung - Wade . . . . .	150
6.2	Zeitaufzeichnung - Sebernegg . . . . .	150
6.3	Zeitaufzeichnung - Resch . . . . .	151
6.4	Zeitaufzeichnung - Kaufmann . . . . .	151

## 9 Literaturverzeichnis

**Decker, K.-H. Kabus, K. (2018). Maschinenelemente (20., neu bearbeitete Auflage).** Carl Hanser Verlag.

**Gehhilfen Krücken | Mobilität. (o. D.). Rebotek.** Abgerufen am 29. Jänner 2022, von  
<https://rehashop.at/mobilitaet/gehhilfen-kruecken/rebotec-unterarm-gehstuetzen-extra-lang-blau>

**Inventor Professional 2022 | 3D-Software Inventor | Autodesk.** (2021, 24. November). Inventor. Abgerufen am 28. Jänner 2022, von

<https://www.autodesk.de/products/inventor/overview?term=1-YEAR&tab=subscription#what-is-inventor>

**Becker, A. Pant, M. (2009). Android Grundlagen und Programmierung.** dpunkt.verlag.

**Wikipedia-Autoren. (2013, 24. Mai). Android Studio.** Android Studio. Abgerufen am 28. Dezember 2021, von

[https://de.wikipedia.org/wiki/Android\\_Studio](https://de.wikipedia.org/wiki/Android_Studio)

**Wikipedia-Autoren. (2007, 5. November). Android (Betriebssystem).** Android Studio. Abgerufen am 28. Dezember 2021, von

[https://de.wikipedia.org/wiki/Android\\_\(Betriebssystem\)](https://de.wikipedia.org/wiki/Android_(Betriebssystem))

**A. (o. D.). GitHub - AnyChart/AnyChart-Android: AnyChart Android Chart is an ama-**

**zing data visualization library for easily creating interactive charts in Android apps. It runs on API 19+ (Android 4.4) and features dozens of built-in chart types. GitHub.** Abgerufen am **7. März 2022, von**

<https://github.com/AnyChart/AnyChart-Android>

**Aufbau MEMS Beschleunigungssensor. (o. D.). [Illustration]. Elektronik-Kompendium.de.**

<https://www.elektronik-kompendium.de/sites/bau/1503041.htm>

**Ewald, W. (2021, Jänner 16). MPU6050 Beschleunigungssensor und Gyroskop. Wolles Elektronikkiste.** Abgerufen am **9. Februar 2022, von**

<https://wolles-elektronikkiste.de/mpu6050-beschleunigungssensor-und-gyroskop#anker6>

**Funktion MEMS. (o. D.). [Illustration]. Elektronik-Kompendium.de.**

<https://www.elektronik-kompendium.de/sites/bau/1503041.htm>

**Deimel, F. Hasenzagl, A. (2018). Grundlagen der Elektrotechnik (3. Auflage (2018) Aufl.). Vertias.**

**Grieger, C. (2018a, Mai 7). 3-Achsen Beschleunigungs- und Lagesensor mit Arduino. Pins.** Abgerufen am **9. Februar 2022, von**

<https://elektro.turanis.de/html/prj075/index.html>

**Grieger, C. (2018b, Mai 7). 3-Achsen Beschleunigungs- und Lagesensor mit Arduino. Pins.** Abgerufen am **9. Februar 2022, von**

<https://elektro.turanis.de/html/prj075/index.html>

**GY-521 MPU-6050. (o. D.). [Illustration]. Wolles Elektronikkiste.**

<https://wolles-elektronikkiste.de/mpu6050-beschleunigungssensor-und-gyroskop#anker6>

**MEMS - Micro-Electro-Mechanical Systems. (o. D.-a). Elektronik Kompendium Website.**

**Abgerufen am 16. Jänner 2022, von**

<https://www.elektronik-kompendium.de/sites/bau/1503041.htm>

**MEMS - Micro-Electro-Mechanical Systems. (o. D.-b). Elektronik Kompendium. Abgerufen am 16. Jänner 2022, von**

<https://www.elektronik-kompendium.de/sites/bau/1503041.htm>

**Transfer Multisort Elektronik. (2020, 9. Oktober). Wie funktioniert und was ist ein Beschleunigungssensor? TME Website. Abgerufen am 16. Jänner 2022, von**

<https://www.tme.eu/de/news/library-articles/page/22568/Wie-funkti oniert-und-was-ist-ein-Beschleunigungssensor/>

**What is Arduino? (o. D.). Arduino. Abgerufen am 14. Februar 2022, von**

<https://www.arduino.cc/en/Guide/Introduction/>

**Akku. (o. D.-a). RN-Wissen. Abgerufen am 21. Februar 2022, von**

<https://rn-wissen.de/wiki/index.php/Akku-Grundlagen>

**Akku. (o. D.-b). Conrad. Abgerufen am 21. Februar 2022, von**

[https://www.conrad.at/de/p/beltrona-18650xh2-54-spezial-akku-18650-stecker-lifepo-4-3-2-v-1500-mah-1923284.html?utm\\_source=google&utm\\_medium=organic&utm\\_campaign=shopping&ef\\_id=CjwKCAiAsNKQBhAPEiwAB-I5zRLzlizd7Jfb-tmVXJHoDqh\\_YtnmIMqDSgQBZwu6WxdOVRN5qJlyvRoCMbsQAvD\\_BwE#productTechData](https://www.conrad.at/de/p/beltrona-18650xh2-54-spezial-akku-18650-stecker-lifepo-4-3-2-v-1500-mah-1923284.html?utm_source=google&utm_medium=organic&utm_campaign=shopping&ef_id=CjwKCAiAsNKQBhAPEiwAB-I5zRLzlizd7Jfb-tmVXJHoDqh_YtnmIMqDSgQBZwu6WxdOVRN5qJlyvRoCMbsQAvD_BwE:G:s&gclid=CjwKCAiAsNKQBhAPEiwAB-I5zRLzlizd7Jfb-tmVXJHoDqh_YtnmIMqDSgQBZwu6WxdOVRN5qJlyvRoCMbsQAvD_BwE#productTechData)

**Aschermann, T. (2020, 30. März). WLAN. CHIP. Abgerufen am 21. Februar 2022, von**

[https://praxistipps.chip.de/wie-funktioniert-wlan-verstaendlich-erklärt\\_50572](https://praxistipps.chip.de/wie-funktioniert-wlan-verstaendlich-erklärt_50572)

**Geiger, J. (2018, 28. September). Bluetooth-FAQ. CHIP 365. Abgerufen am 21. Februar 2022, von**

[https://www\(chip.de/artikel/Bluetooth-FAQ-16\\_140320047.html](https://www(chip.de/artikel/Bluetooth-FAQ-16_140320047.html)

**Hoffmann, K. (o. D.). Anwendung der Wheatstoneschen Brückenschaltung [E-Book]. HBM.**

**Kraftsensor. (o. D.). Conrad. Abgerufen am 22. Dezember 2021, von**

<https://www.conrad.at/de/p/joy-it-sen-pressure20-beruehrungs-sensor-1-st-passend-fuer-entwicklungskits-arduino-raspberry-pi-micro-bit-2309327.html?searchType=SearchRedirect>

**Microcontroller. (o. D.). RS. Abgerufen am 21. Februar 2022, von**

<https://at.rs-online.com/web/generalDisplay.html?id=ideen-und-tipps/Microcontroller-leitfaden>

**NFC-Technologie ihre Vorteile |smart-TEC. (o. D.). Smart-Tec. Abgerufen am 21. Februar 2022, von**

<https://www.smart-tec.com/de/auto-id-welt/nfc-technologie>

**Piezo vs. DMS. (o. D.). Kistler. Abgerufen am 22. Dezember 2021, von**

<https://www.kistler.com/de/glossar/begriff/piezo-vs-dms/>

**reichelt elektronik GmbH Co. KG Internet Team (webmaster@reichelt.de). (o. D.). D1Z BATTERY - D1 Shield - Batterie / LiPo. Elektronik und Technik bei reichelt elektronik günstig bestellen. Abgerufen am 21. Februar 2022, von**

[https://www.reichelt.at/at/de/d1-shield-batterie-lipo-d1z-battery-p266068.html?PROVID=2807&gclid=CjwKCAiAsNKQBhAPEiwAB-I5zZiardXH7f7\\_xeFDthhPpugujfL5mYesiX0G9LDbnhGixYbD13Yb7xoC1tsQAvD\\_BwE](https://www.reichelt.at/at/de/d1-shield-batterie-lipo-d1z-battery-p266068.html?PROVID=2807&gclid=CjwKCAiAsNKQBhAPEiwAB-I5zZiardXH7f7_xeFDthhPpugujfL5mYesiX0G9LDbnhGixYbD13Yb7xoC1tsQAvD_BwE)

Anstelle eines USB-Sticks wurde ein GitHub Repository erstellt. Dieses finden Sie unter folgendem QR-Code. Dort finden Sie alle Daten zu unserem Diplomprojekt.

