

Diplomarbeit

Matteo
Die Matte für Balanceübungen

ausgeführt an der
Höheren Abteilung für Mechatronik

im Schuljahr 2025/2026

durch

Laurenz Wimmer

Nandor Bagari

Mika Böhm

Lisa Pripfl

unter Anleitung von

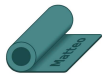
Ewald Cekan

Ewald Cekan

Martin Meschik

Richard Drechsler

Wien, Mai 2026



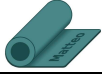
Kurzfassung

Matteo ist eine Gymnastikmatte, die älteren blinden und sehbeeinträchtigten Menschen die Möglichkeit bieten soll, sich eigenständig körperlich zu betätigen. Da im Alter die Bewegungsfreiheit grundsätzlich abnimmt und Einschränkungen der Sehfähigkeit in dieser Hinsicht noch eine weitere Hürde darstellen, setzt Matteo bei einer der wichtigsten Grundlagen an, der Balance. Mithilfe dieses Projekts sollen betroffene Personen eigenständig Balanceübungen durchführen können, ohne auf die Hilfe anderer angewiesen zu sein. Dadurch soll einerseits Motivation zu Aktivitäten gefördert und andererseits eine Grundlage für andere Betätigungen geschaffen werden.

Das Projekt basiert auf in der Matte verbauten Drucksensoren, die die Belastung der einzelnen Füße messen. Pro Fuß gibt es sechs Sensoren, die in Dreiergruppen verbaut sind. Jede dieser Gruppen ist in Kombination mit einer Platte verbaut, die das Gewicht möglichst gleichmäßig auf die drei Sensoren verteilt. Die Platten sind so positioniert, dass jeweils eine Platte die Belastung im vorderen und hinteren Bereich des Fußes misst. Im Anschluss werden linke und rechte Belastung verglichen und die Gewichtsverteilung in Prozent berechnet.

Die Matte bietet die Möglichkeit, sich eigenständig Belastungsziele zu setzen. Unter Berücksichtigung dieser Ziele wird die Gewichtsverteilung ausgewertet. Wenn beispielsweise als Ziel „Rechtes Bein 70%“ gewählt wird, wird die Übung ausgeführt, bis 10 Sekunden lang 70% des Körpergewichts auf dem rechten Bein gemessen werden. Das Auswählen der Übung geschieht über eine integrierte Sprachsteuerung. Auch alle anderen Einstellungen wie Lautstärkenänderungen werden so vorgenommen. Beim Erreichen des Ziels wird via Sprachausgabe das Ende der Übung ausgegeben. Außerdem gibt es zusätzlich visuelles Feedback in Form einer mehrfärbigen LED für nur teilweise sehbeeinträchtigte Personen.

Aufgrund der zu berücksichtigenden Seheinschränkungen hat Matteo einige Funktionen, die zugunsten der Barrierefreiheit und Sicherheit gewählt wurden. Dazu zählen unter anderem Akkus als Stromversorgung, um die Stolpergefahr zu minimieren und ein Schalter, der mit dem Fuß betätigt werden kann, damit Bücken vermieden werden kann. Alle Ideen wurden vor der Umsetzung mit Verantwortlichen des Universitätsportinstituts abgeklärt, da die Sportuniversität Kontakte zu potenziellen Nutzergruppen hat.



Abstract

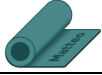
Matteo is a mat designed to enable blind and visually impaired elderly people to exercise independently. As freedom of movement generally decreases with age and visual impairments pose a further obstacle in this regard, Matteo focuses on one of the most important fundamentals: balance. This project aims to enable affected individuals to perform balance exercises independently, without having to rely on the help of others. The aim is to promote motivation to engage in activities and to create a basis for other activities.

The project is based on pressure sensors built into the mat that measure the load on each foot. There are six sensors per foot, which are installed in groups of three. Each of these groups is installed in combination with a plate that distributes the weight as evenly as possible across the three sensors. The plates are positioned so that one plate measures the load in the front and rear areas of the foot. The left and right load are then compared and the weight distribution is calculated as a percentage.

The mat allows you to set your own weight-bearing goals. Weight distribution is evaluated based on these goals. For example, if you select 'Right leg 70%' as your goal, the exercise is performed until 70% of your body weight is measured on your right leg for 10 seconds. The exercise is selected via integrated voice control. All other settings, such as volume changes, are also made in this way. When the target is reached, a voice output announces the end of the exercise. There is also additional visual feedback in the form of a multi-coloured LED for people with partial visual impairments.

Due to the visual impairments that need to be taken into account, Matteo has a number of features that have been chosen for accessibility and safety reasons. These include rechargeable batteries as a power source to minimise the risk of tripping and a switch that can be operated with the foot to avoid bending down. All ideas were clarified with those responsible at the University Sports Institute prior to implementation, as the sports university has contacts with potential user groups¹.

¹ Übersetzt mit [DeepL.com](https://www.deepl.com) am 4.März 2026



Ehrenwörtliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen Hilfsmittel als die angegebenen benützt habe. Die Stellen, die anderen Werken (gilt ebenso für Werke aus elektronischen Datenbanken oder aus dem Internet) wörtlich oder sinngemäß entnommen sind, habe ich unter Angabe der Quelle und Einhaltung der Regeln wissenschaftlichen Zitierens kenntlich gemacht. Diese Versicherung umfasst auch in der Arbeit verwendete bildliche Darstellungen, Tabellen, Skizzen und Zeichnungen.

Für die Erstellung der Arbeit habe ich auch folgende Hilfsmittel generativer KI-Tools zu folgendem Zweck verwendet: DeepL zum Übersetzen des Abstracts. Die verwendeten Hilfsmittel wurden vollständig und wahrheitsgetreu inkl. Produktversion und Prompt ausgewiesen.

Wien, am

<eigenhändige Unterschriften aller Teammitglieder>



Präambel

Die Inhalte dieser Diplomarbeit entsprechen § 7(1) und § 24 der Verordnung des Bundesministers für Bildung über die abschließenden Prüfungen in den berufsbildenden mittleren und höheren Schulen (Prüfungsordnung BMHS) vom 30.5.2012 (BGBl. Nr. II 177/2012) in der derzeit geltenden Fassung.

Liste der betreuenden Lehrer:

Prof. Dipl.-Ing. Dr. techn. Ewald Cekan, Hauptbetreuer

Prof. Dipl.-Ing. Martin Meschik, stv. Hauptbetreuer

Prof. Dipl.-Ing. Richard Drechsler, Individueller Betreuer

Liste der Kooperationspartner:

Universität Wien, Universitätssportinstitut



Inhaltsverzeichnis

1	ALLGEMEINES	2
1.1	Projektteam	2
1.1.1	Laurenz Wimmer	3
1.1.2	Nandor Bagari	4
1.1.3	Mika Böhm	5
1.1.4	Lisa Pripfl	6
1.2	Ausgangssituation	7
1.3	Projektidee	7
1.4	Beschreibung der Idee	7
1.5	Projektmanagement	8
1.5.1	Projektorganisation	8
1.5.2	Objektstrukturplan	8
1.5.3	Projektstrukturplan	9
1.5.4	Zieldefinition	10
1.5.5	Risikoanalyse	15
1.5.6	Stakeholder-Analyse	17
1.5.7	Projektfinanzen	19
1.5.8	Meilensteine	21
1.5.9	Projektmarketing	21
1.5.10	Team-Coaching	24
1.5.11	Meetings	24
1.5.12	Arbeitszeiten	25
2	ELEKTRONIK	26
2.1	Allgemeines Konzept	26
2.2	Stromversorgung	27
2.2.1	Akkus	28
2.2.2	Schalter	28



2.2.3 Akku-Lademodul.....	29
2.2.4 Step-Up-Converter	30
2.2.5 Step-Down-Converter	31
2.2.6 Schaltplan	32
2.3 Messschaltung.....	34
2.3.1 Bauteile Messschaltung.....	34
2.3.2 Erklärung der Schaltung.....	36
2.3.3 Bild Messplatine	39
2.3.4 Vorversionen Messplatine	40
2.4 Platine	42
2.4.1 Bauteile.....	42
2.4.2 Schaltplan	48
2.4.3 Erklärung Platine.....	49
2.4.4 Nachträgliche Verbesserungen	54
2.4.5 Bild gesamte Platine	54
2.4.6 Vorversionen	55
2.5 Akkustand Anzeige	56
2.5.1 Verwendete Bauteile.....	56
2.5.2 Schaltplan	58
3 KONSTRUKTION	59
3.1 Gehäuse	60
3.1.1 Boden.....	61
3.1.2 Deckel	64
3.1.3 Bild Gehäuse	67
3.1.4 Vorversionen	67
3.2 Matte.....	68
3.3 Platten zur Messung.....	70
3.3.1 Material.....	71
3.3.2 Unterlagen	73



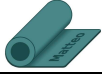
3.3.3 Vorversion	74
3.4 Platine	75
4 PROGRAMMIERUNG	77
4.1 Website	77
4.1.1 Visual Studio Code.....	77
4.1.2 Aufbau der Website	78
4.1.3 HTML-Markup.....	81
4.1.4 CSS	84
4.2 Übersicht des Microcontrollersystems	86
4.3 Firmware des PIC	86
4.3.1 MPLABX IDE	87
4.3.2 H-Term.....	87
4.3.3 ADC (Analog-to-Digital Converter Module)	88
4.3.4 Timer1 (Timer1 Module).....	92
4.3.5 GPIO-Pins (General Purpose Input Output Pins)	93
4.3.6 UART (Universal Asynchronous Receiver Transmitter)	94
4.3.7 Hilfsfunktionen.....	106
4.3.8 Hauptprogramm	107
4.4 Kommunikation zwischen den Mikrocontrollern.....	108
4.4.1 PIC16F1827 zu ESP32	108
4.4.2 Arduino Nicla Voice zu ESP32	113
4.5 Sprachausgabe	116
4.5.1 User Interface.....	116
4.5.2 Theorie.....	117
4.5.3 Grundbegriffe der Audioverarbeitung	120
4.5.4 Gesamtfunktion	120
4.5.5 Lautsprecher ansteuern.....	121
4.6 Spracheingabe.....	125
4.6.1 Edge Impulse	125



4.6.2 Firmware.....	137
4.6.3 Bibliotheken und Funktionen	139
5 LESSONS LEARNED.....	142
5.1 Laurenz Wimmer	142
5.2 Nandor Bagari	144
5.3 Mika Böhm.....	146
5.4 Lisa Pripfl	148
6 LITERATURVERZEICHNIS	150
7 TABELLENVERZEICHNIS	156
8 ABBILDUNGSVERZEICHNIS	157
9 ANHANG.....	163
9.1 Abnahmeprotokoll	164
9.2 Meeting-Protokolle.....	170
9.3 Begleitprotokolle	191
9.4 Jugend Innovativ Bericht.....	195
9.5 Konstruktionszeichnungen:	219
9.6 Schaltpläne	221
9.6.1 Messplatine	221
9.6.2 Hauptplatine Seite 1 (5 V)	222
9.6.3 Hauptplatine Seite 2 (3,3 V)	223
9.6.4 Akkuladestand Anzeige	224
9.7 Datenblätter.....	225
9.7.1 CD74HC4067 [4]	225
9.7.2 BSS138 [5]	230
9.7.3 MCP6002 [6].....	233
9.7.4 Akku-Lademodul [7].....	238



9.7.5 SD-Karten-Modul [8]	241
9.7.6 Andere Datenblätter	243
9.8 Programmcode	244



1 Allgemeines

1.1 Projektteam



Abbildung 1 Diplomarbeitsteam

Das Projektteam von Matteo besteht aus (von links nach rechts):

- Laurenz Wimmer, Projektleiter
- Nandor Bagari, stv. Projektleiter
- Mika Böhm, Projektmitarbeiterin
- Lisa Pripfl, Projektmitarbeiterin

Die Mitglieder werden auf den folgenden Seiten genauer vorgestellt.

1.1.1 Laurenz Wimmer

Rolle:

Projektleiter

Kontakt:

laurenz.wimmer@htl.rennweg.at

Aufgabengebiete:

- Projektmanagement
- Projektmarketing
- Elektronik



Abbildung 2 Laurenz Wimmer

Arbeitszeit: 215 Stunden

Individuelle Aufgabenstellung:

Ich war für die Stromversorgung der Matte sowie die visuelle Rückmeldung in Form einer LED zuständig. Außerdem habe ich in meiner Rolle als Projektleiter die Organisation aller Meetings, die Kommunikation mit Sponsoren, die Bestellungen sowie einen großen Teil des Projektmarketings übernommen.

Die entsprechenden Ziele sind:

- Ziel-H 1 B: Spannungsversorgung
- Ziel-H 7: Projektmanagement
- Ziel-O 1 A: Lichtanzeige Hardware

1.1.2 Nandor Bagari

Rolle:

Stellvertretender Projektleiter

Kontakt:

nandor.bagari@htl.rennweg.at

Aufgabengebiete:

- Konstruktion
- Elektronik
- Projektmarketing



Abbildung 3 Nandor Bagari

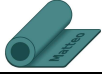
Arbeitszeit: 250 Stunden

Individuelle Aufgabenstellung:

Ich war für die Elektronik und Konstruktion zuständig. Meine Aufgaben umfassten verschiedene Bereiche wie das Entwerfen einer Messschaltung, das Designen einer Platine, um alle Bauteile zu verbinden, sowie die Konstruktion der Matte und eines Gehäuses für die Platine.

Die entsprechenden Ziele sind:

- Ziel-H 1 A Platine & Sensoren
- Ziel-H 1 C Peripherie
- Ziel-H 2 Konstruktion und Aufbau
- Ziel-O 2 Social Media
- Ziel-O 5 A Haptisches Feedback Hardware



1.1.3 Mika Böhm

Rolle:

Projektmitarbeiterin

Kontakt:

mika.boehm@htl.rennweg.at

Aufgabengebiete:

- Programmierung Website
- Programmierung des Messprogramms
- Kommunikation zwischen Microcontrollern

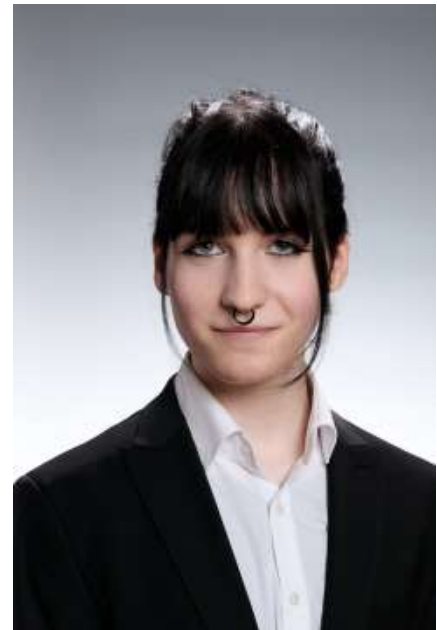


Abbildung 4 Mika Böhm

Arbeitszeit: 230 Stunden

Individuelle Aufgabenstellung:

Ich war für die Programmierung der Firmware zur Gewichtsmessung zuständig. Darunter fallen die Datenverarbeitung und die Logik der Matte. Meine zweite Aufgabe war das Erstellen einer Website, die allgemeine Informationen und den Fortschritt des Projekts öffentlich zugänglich macht.

Die entsprechenden Ziele sind:

- Ziel-H 3 Microcontrollerprogrammierung
- Ziel-H 6 Website
- Ziel-O 1 B Lichtanzeige zur Positionierung der Gliedmaßen (Programmierung)
- Ziel-O 4 Englische Version der Website
- Ziel-O 5 B Haptisches Feedback Software

1.1.4 Lisa Pripfl

Rolle:

Projektmitarbeiterin

Kontakt:

lisa.pripfl@htl.rennweg.at

Aufgabengebiete:

- Sprachsteuerung
- Sprachausgabe
- Kommunikation zwischen Microcontrollern



Abbildung 5 Lisa Pripfl

Arbeitszeit: 280 Stunden

Individuelle Aufgabenstellung:

Ich war für die Spracheingabe und Sprachausgabe zuständig. Darunter fallen das Training des Sprachmodells zur Erkennung der Befehle, die Ausgabe der erkannten Befehle zur Bestätigung und die Ausgabe der aktuellen Belastung.

Die entsprechenden Ziele sind:

- Ziel-H 4 Spracheingabe
- Ziel-H 5 Sprachausgabe
- Ziel-O 3 Messung ohne Zielsetzung

1.2 Ausgangssituation

Mit steigendem Alter wird es immer schwieriger und anstrengender, sich körperlich zu betätigen. Zusätzliche Hindernisse in Form von Blindheit oder Sehbeeinträchtigungen sind dann ein weiterer Faktor, der körperliche Aktivitäten erschwert. Vor allem die Balance wird durch Sehschwächen beeinträchtigt, die einen wichtigen Teil zu körperlichen Aktivitäten beiträgt.

1.3 Projektidee

Im Rahmen des Projektes „Sinnvoll Aktiv“, das die Universität Wien durchführt, ist in der Untergruppe „Active Aging“ die Idee entstanden, eine Matte zur eigenständigen Ausübung von Balanceübungen zu entwickeln. Diese Idee wurde von Prof. Meschik an die Klassen weitergegeben und nach Rücksprache mit den Verantwortlichen von „Sinnvoll aktiv“ haben wir entschieden, dieses Projekt als Diplomarbeit umzusetzen.

1.4 Beschreibung der Idee

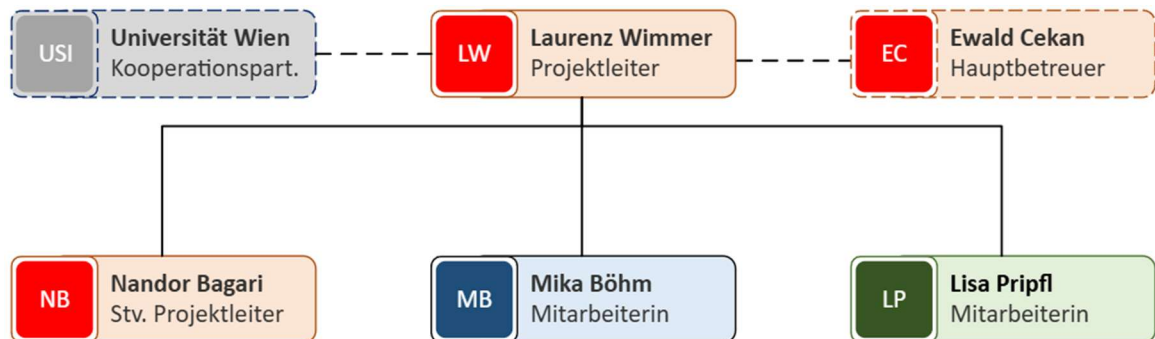
Über die Dauer des letzten Schuljahres wurde eine Matte mit integrierten Drucksensoren entwickelt, die die Belastung der Füße der Benutzerinnen und Benutzer misst und miteinander vergleicht. Über eine Sprachsteuerung mit Sprachausgabe können Übungen eingestellt werden, die auf die Verbesserung der Balance abzielen und einen Anreiz zur körperlichen Betätigung bieten sollen.



Abbildung 6 3D-Modell Matte Idee

1.5 Projektmanagement

1.5.1 Projektorganisation



Hauptbetreuer ... Ewald Cekan
 Stv. Hauptbetreuer ... Martin Meschik
 Individueller Betreuer ... Richard Drechsler

Abbildung 7 Projektorganisation

In Abbildung 7 sind alle Beteiligten angeführt, die aktiv am Projekt mitgewirkt haben. Die individuellen Betreuer sind den Projektmitarbeiterinnen und -mitarbeitern farblich zugeordnet. Die Universität Wien als Auftraggeberin und das Universitätssportinstitut (USI) als Ansprechpartner und Berater haben ebenfalls eine wichtige Rolle bei der Umsetzung gespielt.

1.5.2 Objektstrukturplan

Matteo				
Matte	Elektronik	Steuereinheit	Dokumente	Marketing
Material	Spannungsversorgung	Microcontroller	Präsentationen	Website
Sensoren	Platine	Software	Diplomarbeitsbuch	Werbeplakat
Schutz für Sensoren	Ein-/Ausschalter	Sprachsteuerung	Projekteinreichungen	Visitenkarten
Gehäuse	Taster	Sprachausgabe		Social Media
Lichtanzeige	Infos-LEDs			
	Lautsprecher			
	Mikrofon			
	Lichtanzeige			

Tabelle 1: Objektstrukturplan

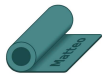
Der Objektstrukturplan zeigt, welche Arbeitspakete für den Verlauf des Projekts geplant wurden und hat als Leitfaden für die Durchführung gedient. Die spezifischen Arbeitsschritte, die im nachfolgenden Projektstrukturplan erarbeitet wurden, sind vom Objektstrukturplan abgeleitet.

1.5.3 Projektstrukturplan

1. Matteo									
Projektdurchführung						1.3		Abschluss	
1.1	Planung	1.2	Technische Planung	1.2.2	Umsetzung	1.2.3	Testphase	1.3.1	Projektbuch fertigstellen
1.1.1	Grobkonzept erstellen	1.2.1	Recherchieren	1.2.2.1	Platinen bestellen	1.2.3.1	Sensorik testen	1.3.2	In Betrieb nehmen
1.1.2	Ziele definieren	1.2.1.1	Erstes Modell erstellen	1.2.2.2	Platinen bestücken	1.2.3.2	Stromversorgung testen	1.3.3	Präsentationen halten
1.1.3	Risikoanalyse erstellen	1.2.1.2	Bauteile aussuchen	1.2.2.3	Gehäuse herstellen	1.2.3.3	Sprachsteuerung testen	1.3.4	Defensio halten
1.1.4	OPS erstellen	1.2.1.3	Blockschaltbild erstellen	1.2.2.4	Software programmieren	1.2.3.4	Software und Hardware testen	1.3.5	Projekt abgenommen
1.1.5	PSP erstellen	1.2.1.4	Schaltplan erstellen	1.2.2.5	Sensoren in die Matte einbauen	1.2.3.5	Testphase abgeschlossen		
1.1.6	Antrag erstellen	1.2.1.5	Platinenlayout erstellen	1.2.2.6	Sprachsteuerung trainieren				
1.1.7	Planung abgeschlossen	1.2.1.6	Flussdiagramme erstellen	1.2.2.7	Software und Hardware implementieren				
		1.2.1.7	Technische Planung abgeschlossen	1.2.2.8	Umsetzung abgeschlossen				
		1.2.1.8							

Tabelle 2: Projektstrukturplan

Der Projektstrukturplan teilt die Durchführung des Projektes in einzelne Phasen ein und zeigt, welche Arbeitsschritte und Meilensteine bis zur Fertigstellung des Projektes umzusetzen sind. Die rot hinterlegten Kästchen stellen diese Meilensteine dar und markieren jeweils das Ende einer bestimmten Phase im Projekt. Aufgrund der unterschiedlichen Aufgabenverteilung sind die Phasen teils parallel zueinander und nicht nacheinander abgelaufen, auch wenn sie hier untereinander dargestellt sind. Es wurde beispielsweise gleichzeitig mit der Erstellung der finalen Platine die Sprachsteuerung trainiert und die Software für alle Mikroprozessoren programmiert.



1.5.4 Zieldefinition

1.5.4.1 Hauptziele

Ziel-H 1 Elektronik

Die Elektronik wird über Akkus versorgt und verbindet Microcontroller, Sensoren, Spannungsversorgung und diverse Schalter, Taster und LEDs über eine zentrale Platine.

a. Platine & Sensoren

Eine Platine verbindet alle elektronischen Komponenten, Sensoren und Schnittstellen mit dem Mikrocontroller. Drucksensoren sind in der Matte verbaut und sind funktionsfähig mit der Platine verbunden, sodass mithilfe einer Messschaltung auf der Platine Messungen vorgenommen werden können.

b. Spannungsversorgung

Die Elektronik wird durch Akkus mit Spannung versorgt, die eine Mindestbetriebszeit von 1 Stunde gewährleisten. Die Akkus werden über einen USB-C-Adapter mithilfe eines Netzgeräts geladen, das direkt an eine Steckdose angeschlossen werden kann.

c. Peripherie

Es sind mind. eine LED zur Anzeige des Akkuladestands, ein Schalter zum Ein- und Ausschalten der Matte sowie ein Taster zum Resetten der Matte verbaut.

Ziel-H 2 Konstruktion und Aufbau

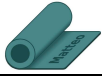
Die Matte besteht aus zwei Teilen, einer mind. 900 x 600 mm großen Physiomatte und einem Gehäuse für die Elektronik an einer Seite der Matte.

a. Matte

Die mind. 900 x 600 mm große Matte hat ein passendes Material für die Oberseite und Unterseite, um eine angenehme Oberfläche und Schutz für die Sensoren zu bieten.

b. Gehäuse

Das Gehäuse für die elektronischen Bauteile ist an einer Seite der Matte angebracht und schützt die Elektronik vor Schäden. Es beinhaltet die Platinen, die Akkus und die Peripherie.



Ziel-H 3 Microcontroller-Programmierung

Es werden die Sensordaten von einem Microcontroller eingelesen und überprüft, ob die eingestellten Trainingsziele erreicht wurden. Anschließend werden die Daten aufbereitet, um sie an einer Schnittstelle auszugeben.

Ziel-H 4 Spracheingabe

Es können diverse Einstellungen mithilfe von Spracheingabe mit max. 20 Befehlen vorgenommen werden.

a. Belastungsziele

Es können Belastungsziele für den linken und rechten Fuß in Form von prozentueller Belastung gesetzt werden. Die Abstufung erfolgt in 10% Schritten.

b. Einstellungen

Es gibt Befehle zum Starten und Stoppen der Übung sowie zum Einstellen der Lautstärke der Sprachausgabe.

Ziel-H 5 Sprachausgabe

Die Sprachausgabe gibt die aktuelle Belastung sowie die aktuellen Einstellungen aus.

a. Feedback während Übungen

Während eine Übung aktiv ist, wird frühestens alle 4 Sekunden die aktuelle Belastung ausgegeben. Es wird außerdem ausgegeben, ob das Ziel erreicht wurde.

b. Rückmeldung nach Einstellungen

Das Starten und Stoppen von Übungen wird ausgegeben. Nachdem Einstellungen vorgenommen wurden, wird der erkannte Befehl ausgegeben, um Missverständnisse zu vermeiden.

Ziel-H 6 Website

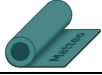
Die Website dient dem Marketing. Sie macht die Projektidee, den Projektfortschritt in Form der Meilensteine und Kontaktinformationen über die Teammitglieder öffentlich zugänglich. Die Website ist aus 6 verschiedenen Seiten aufgebaut.

a. Home

Die Homepage beinhaltet Name, Logo, Sponsoren und eine Abbildung des 3-D-Modells der Matte nach aktuellem Planungsstand

b. Idee

Die Seite „Idee“ beinhaltet Informationen über die Ziele des Projekts, die (geplanten) Kooperationen und das Konzept inklusive einer Skizze zur Veranschaulichung



c. Meilensteine

Der Projektfortschritt wird in Form der erreichten Meilensteine dargestellt.

d. Team

Kontaktdaten des Teams sowie die Aufgabenverteilung sind auf der Seite „Team“ zu finden.

e. Impressum

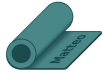
Das Impressum beinhaltet die Adresse der Schule und die Verantwortlichen für die Haftung für Inhalte und Links auf der Website

f. Datenschutz

Die Seite „Datenschutz“ zeigt alle datenschutzrechtlichen Angaben.

Ziel-H 7 Projektmanagement

Das Projektmanagement wird gemäß der Projektmanagement-Vorgaben der HTL-Rennweg durchgeführt. Durch regelmäßige Meetings werden die Betreuer über den Fortschritt informiert.



1.5.4.2 Optionale Ziele

Ziel-0 1 Lichtanzeige

a. Hardware

Eine auf der Matte sichtbare Lichtanzeige in Form einer LED zeigt an, ob das eingestellte Ziel erreicht wurde.

b. Programmierung

Die Lichtanzeige ist so programmiert, dass sie anzeigt, ob das Ziel erreicht wurde.

Ziel-0 2 Social Media

Das Projekt hat eine Social-Media-Präsenz auf mind. 1 Social-Media-Plattform.

Ziel-0 3 Messung ohne Zielsetzung

Es gibt eine Einstellung, bei der kein Belastungsziel gesetzt wird, sondern dauerhaft die Belastung gemessen und frühestens alle 4 Sekunden ausgegeben wird.

Ziel-0 4 Englische Version der Website

Auf der Website kann zwischen Deutsch und Englisch gewechselt werden.

Ziel-0 5 Haptisches Feedback

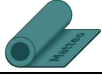
Die Matte gibt haptisches Feedback, ob das Ziel erreicht wurde.

c. Hardware

In der Matte wird ein Vibrationsmotor verbaut.

d. Programmierung

Der Vibrationsmotor wird so programmiert, dass er dann vibriert, wenn das Ziel erreicht wurde.



1.5.4.3 Nicht-Ziele

Ziel-N 1 Messung auf der ganzen Matte

Die Sensoren sind auf der ganzen Matte verteilt, sodass lückenlos die Belastung gemessen wird.

Ziel-N 2 Aktive Ausbalancierung

Die Matte wirkt dagegen, wenn Benutzer:innen im Ungleichgewicht stehen.

Ziel-N 3 Unebene Oberflächen

Die Matte funktioniert auf unebenen Oberflächen.

Ziel-N 4 Verbesserungsvorschläge

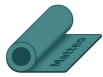
Die Matte gibt Vorschläge zur Verbesserung des Gleichgewichtes und/oder medizinische Empfehlungen.

Ziel-N 5 Ein- und Ausschaltfeedback

Die Matte gibt über die Sprachausgabe in Form von Wörtern aus, wenn die Matte ein- bzw. ausgeschaltet wird.

Ziel-N 6 Andere Übungen

Die Matte ist für Übungen ausgelegt, die nicht auf Gewichtsverteilung beruhen.



1.5.5 Risikoanalyse

#	RISIKO		AUSWIRKUNG				MASSNAHMEN	
	Bezeichnung	Beschreibung	P	A	RF	Verzögerung	... zur Reduktion v. P bzw. A oder für den Fall des Risikoeintritts einen Plan B überlegen	Kosten
						(in Wochen)		
1	Motivation	Eines oder mehrere Team-Mitglieder sind nicht motiviert	40	70	2800	3-4	Regelmäßige Gespräche und frühzeitige Unterstützung bei Überforderung	0 €
2	Zeitmanagement	Zeitprobleme aufgrund von Tests, Schularbeiten etc.	40	60	2400	3-4	Im Voraus Zeitplan definieren und einhalten, genügend Puffer einplanen	0 €
3	Materialsuche	Die Materialsuche dauert länger als ursprünglich gedacht	40	50	2000	2-3	Frühzeitig anfangen, um auch bei Verzögerungen nicht weit zurückzufallen	0 €
4	Diebstahl	Bauteile oder Prototypen werden unrechtmäßig entwendet	20	90	1800	3-4	Bauteile und Prototypen immer gut verstauen und verschließen	500 €
5	Unfall	Ein Mitarbeiter verletzt sich	20	90	1800	6-8	Aufgaben gleichmäßig auf andere PMA verteilen	0 €
6	Lieferverzögerung	Bauteile verspäten sich	50	30	1500	1-2	Alternative Lieferanten bereithalten	0 €
7	Erfahrung	Fehlende Erfahrung in benötigten Bereichen	20	40	800	1	Im Voraus prüfen, welches Wissen gebraucht wird	0 €
8	Verantwortlichkeit	Niemand fühlt sich für bestimmte Aufgaben zuständig	20	40	800	1	Frühzeitig Zuständigkeiten abklären	0 €
9	Betreuerausfall	Betreuer fallen aus	30	20	600	1	Andere Lehrer als Vertretung bereithalten	0 €
10	Defekt	Defekte Bauteile	10	40	400	1	Mehr bestellen oder nachbestellen	100 €

Tabelle 3: Risikoanalyse

In der Risikoanalyse werden alle möglichen Risiken festgehalten, die im Verlauf des Projekts auftreten können. Sie werden nach Eintrittswahrscheinlichkeit (P) und Auswirkung auf das Projekt (A) eingeteilt, die multipliziert den Risikofaktor (RF) ergeben:

$$RF = P * A$$

Die erwartete Verzögerung wird anhand des Risikos abgeschätzt. Außerdem werden schon vorsorglich Gegenmaßnahmen formuliert, die sofort umgesetzt werden können, sobald das entsprechende Risiko eintritt.

Während der Umsetzung sind nur Risiko Nr. 1 und Nr. 10 aufgetreten, weswegen eine ungefähre Verzögerung von ca. 5 Wochen und Mehrkosten von ca. 21,- € entstanden sind.

1.5.5.1 Risikoportfolio

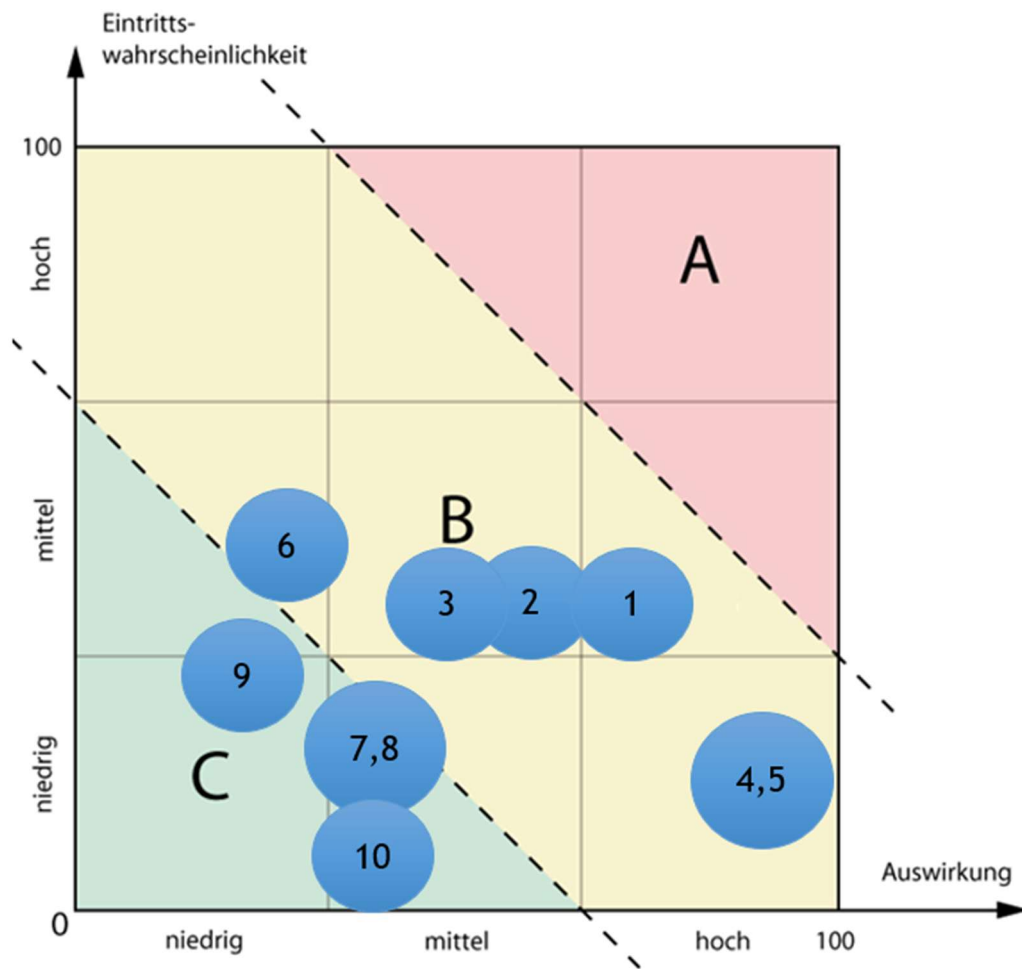


Abbildung 8 Risikoportfolio

Im Risikoportfolio werden die Risiken grafisch dargestellt. So ist auf einen Blick ersichtlich, welche Risiken am gefährlichsten für den Verlauf des Projektes sind. Die dargestellten Bereiche A, B und C kennzeichnen die Relevanz der Risiken.

Risiken im Bereich A werden „Showstopper“ (Projekt kann nicht weiter durchgeführt werden) genannt, da sie zu große Auswirkungen auf das Projekt und/oder eine zu hohe Eintrittswahrscheinlichkeit hätten. Deswegen wird ein Projekt nicht umgesetzt, sollten einer oder mehrere Showstopper identifiziert werden.

Risiken im Bereich B sind zwar immer noch problematisch, können aber gut durch Gegenmaßnahmen ausgeglichen werden.

Risiken im Bereich C sind nicht sehr relevant, da sie sehr geringe Auswirkungen haben und einfach gegengewirkt werden kann.



1.5.6 Stakeholder-Analyse

In der Stakeholder-Analyse sind alle Personen und Gruppen angeführt, die Interesse am Projekt haben könnten. Sie dient dazu, Maßnahmen zu planen, mit denen die Interessen aller Beteiligten bestmöglich abgedeckt werden können.

1.5.6.1 Stakeholder-Tabelle

Nr. (Ext./Int)	Bezeichnung des Stakeholders (SH)	Beschreibung des Einflusses Wie kann der SH das Projekt ev. unterstützen, wie behindern?	Einfluss auf Projekt/Macht (gering/hoch)	Einstellung zum Projekt (positiv/negativ)	Maßnahmen
I-01	Team	Durchführung	hoch	Positiv	Motivieren, unterstützen, zusammenarbeiten.
I-02	Lehrer	Betreuung und Unterstützung bei Problemen	hoch	Positiv	2-wöchentliches Meeting, über Fortschritt informieren
I-03	Sportuniversität Wien	Auftraggeber gibt Vorgaben	hoch	Positiv	Monatliches Update zum Projektfortschritt, Rücksprache über Umsetzung
E-04	Sponsoren	Finanzierung	hoch	Positiv	Auf Plaketen und Webseiten erwähnen, Auf Anfrage über Projektfortschritt informieren
E-05	Hilfsgemeinschaft der Blinden und Sehgeschwachen Österreichs	Feedback zu Ideen, Testen	niedrig	Positiv	Auf Anfrage auf dem Laufenden halten, Feedback einarbeiten, Ggf. Prototyp zum Testen zur Verfügung stellen
E-06	Versportssportklub ASVÖ-Wien	Feedback zu Ideen, Testen	niedrig	Positiv	Auf Anfrage auf dem Laufenden halten
E-07	Blinden und Sehbehindertenverba nd Wien, NÖ und Bgld	Feedback zu Ideen, Testen	niedrig	Positiv	Auf Anfrage auf dem Laufenden halten

Tabelle 4: Stakeholder-Tabelle

Matteo hat ausschließlich positive Stakeholder, weswegen keine Probleme durch unzufriedene Personen bzw. Personengruppen entstanden sind.



1.5.6.2 Stakeholder-Portfolio

Einfluss/Macht	viel	Zufriedenstellen	Eng managen (hoher Aufwand) I-01 Team I-02 Lehrer I-03 Sportuniversität Wien E-04 Sponsoren
	wenig	Überwachen (wenig Aufwand betreiben)	Informieren/auf dem Laufenden halten E-05 Hilfgemeinschaft der Blinden und Sehschwachen Österreichs E-06 Versehrtensportklub ASVÖ-Wien E-07 Blinden und Sehbehindertenverband Wien, NÖ und Bgld
		Einstellung zum Projekt	
		- +	

Abbildung 9 Stakeholder-Portfolio

Das Stakeholder-Portfolio dient der Visualisierung der Stakeholder-Analyse und zeigt auf einen Blick das Interesse und den Einfluss der jeweiligen Stakeholder.



1.5.7 Projektfinanzen

1.5.7.1 Kostenschätzung

Pos.	Bezeichnung des Aufwands	Kosten
1	Elektronik	€ 1 500
2	Material der Matte	€ 300
3	Gehäuse für Elektronik	€ 200
4	Microcontroller	€ 100
5	Serverkosten für Website	€ 100
	Gesamtkosten	€ 2 200

Tabelle 5:Kostenschätzung

Die im Rahmen der Projektplanung durchgeführte Kostenschätzung zeigt grob die einzelnen Bereiche, in denen Kosten erwartet werden. Darin sind auch zusätzliche Kosten berücksichtigt, die durch Risiken entstehen könnten.

1.5.7.2 Projektkosten

Pos.	Bezeichnung des Aufwands	Kosten
1	Elektronik	€ 494,83
2	Material der Matte	€ 182,65
3	Gehäuse für Elektronik	€ 79,64
4	Microcontroller	€ 77,18
5	Serverkosten für Website	€ 62,33
	Gesamtkosten	€ 896,63

Tabelle 6: Tatsächliche Kosten

Die tatsächlichen Projektkosten sind wesentlich geringer ausgefallen als ursprünglich geschätzt. Das liegt daran, dass im Laufe des Projektes wenig Probleme aufgetreten sind und deswegen fast keine Kosten durch Nachbestellungen angefallen sind.

1.5.7.3 Kostenvergleich

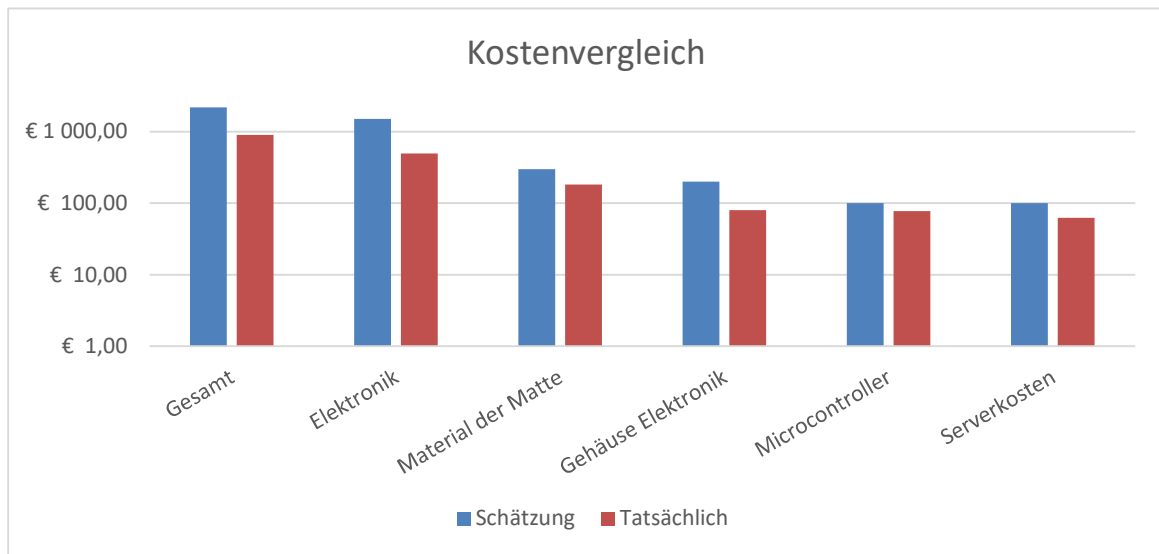


Abbildung 10 Kostenvergleich

Abbildung 10 zeigt einen Vergleich der fünf geschätzten Bereiche und der Gesamtkosten mit den tatsächlichen Kosten. Zur besseren Sichtbarkeit wurde die Y-Achse logarithmisch formatiert, da sonst die einzelnen Teilbereiche nur sehr klein dargestellt worden wären.

1.5.7.4 Kostendeckung

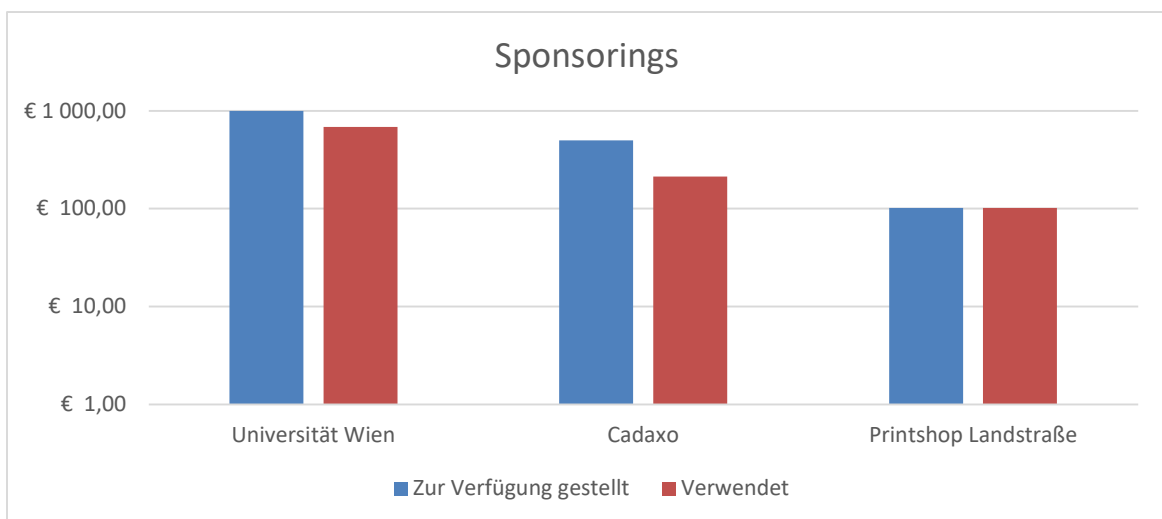


Abbildung 11 Sponsorings

Die gesamten Bauteilkosten des Projektes konnten durch Sponsoring von Universität Wien und Cadaxo abgedeckt werden. Dabei wurde nur ca. die Hälfte des zur Verfügung gestellten Sponsor-Geldes verwendet. Außerdem hat Printshop Landstraße Visitenkarten für den Tag der offenen Tür im Wert von 101,83 € gesponsert. Abbildung 11 zeigt die gesponserten Kosten pro Firma verglichen mit dem zugesagten Kapital. Außerdem wurden Bauteile verwendet, die von Projektmitgliedern von früheren Projekten zur Verfügung gestellt wurden. Diese Bauteile sind nicht in der Kostenrechnung inkludiert.

1.5.8 Meilensteine

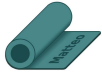
#	Meilenstein	SOLL-Termin	IST-Termin
1	Website Launchen	12.Sep.25	10.Sep.25
2	Planung abgeschlossen	16.Sep.25	15.Sep.25
3	Abgabe Antrag	19.Sep.25	17.Sep.25
4	Technische Planung abgeschlossen	17.Okt.25	14.Okt.25
5	Sprachausgabe	03.Nov.25	03.Nov.25
6	Erster Prototyp	07.Nov.25	23.Jän.26
7	Zwischenpräsentation	25.Nov.25	25.Nov.25
8	Umsetzung abgeschlossen	23.Dez.25	03.Mär.26
9	Testen abgeschlossen	28. Jan 2026	03.Mär.26
10	Abnahme durchgeführt	13.Mär.26	03.Mär.26
11	Endpräsentation abgehalten	17.Mär.26	17.Mär.26

Tabelle 7: Meilensteine

Die anfangs geplanten Meilensteine konnten zu einem großen Teil rechtzeitig abgeschlossen werden. Die Termine für den ersten Prototyp und die abgeschlossene Umsetzung wurden während der Planung am Projektbeginn bewusst sehr früh gesetzt, um einen Ansporn für einen früheren Projektabschluss zu bieten. Dabei wurde unter Rücksprache mit den Betreuern in Kauf genommen, dass wahrscheinlich Verzögerungen auftreten werden, da die Abnahme bis spätestens 13. März 2026 als finales Ziel betrachtet wurde.

1.5.9 Projektmarketing

Im Laufe des Projektes und vor allem im Rahmen der Tage der offenen Tür wurde Projektmarketing betrieben. Darunter fällt die Erstellung von Visitenkarten, des Logos, eines Plakats und der Website sowie das Instagram-Profil. Außerdem wurde das Projekt im Rahmen der Tage der offenen Tür Interessierten vorgestellt. In diesem Teil werden die Visitenkarten, das Plakat und das Instagram-Profil beschrieben, die Website und das Logo werden weiter unten behandelt.



1.5.9.1 Visitenkarten

Die Visitenkarten wurden eigenständig designed und von Printshop Landstraße gedruckt. Dazu wurden verschiedene Vorschläge von VistaPrint in Betracht gezogen und anhand des besten Designs die Visitenkarten erstellt. Wichtig war dabei das Format von 85 x 55 mm, das wir von Printshop Landstraße als Vorgabe erhalten haben. Abbildung 12 und Abbildung 13 zeigen die Vorder- und Rückseite einer Visitenkarte:



Abbildung 12 Visitenkarte vorne

Abbildung 13 Visitenkarte hinten

1.5.9.2 Instagram-Auftritt

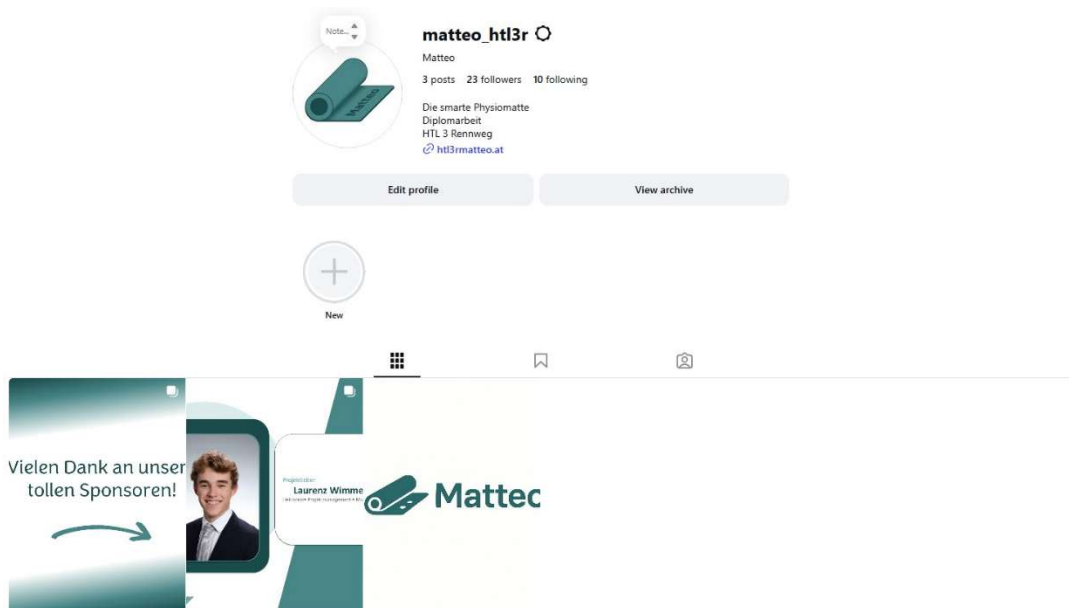
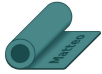


Abbildung 14 Instagram-Profil

Das von Nandor Bagari geführte Instagram-Profil hat im Laufe des Projektes als Informationsquelle für Sponsoren und Interessierte gedient.



1.5.9.3 Plakat



Mechatronik

Matteo

Die Matte für Balanceübungen

Aufgabenstellung

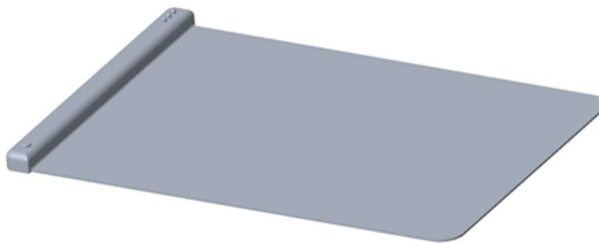
Das Ziel von Matteo ist es, eine Gymnastikmatte zu entwickeln, mit der sehbeeinträchtigte ältere Menschen eigenständig Balanceübungen durchführen können. Dazu messen Sensoren die Belastung der einzelnen Gliedmaßen, woraufhin die Gewichtsverteilung in Prozent ausgegeben wird.

Die Bedienung der Matte erfolgt über eine Sprachsteuerung, die Ausgabe der Belastung und der vorgenommenen Einstellungen über eine Sprachausgabe. Dadurch wird sichergestellt, dass die Matte trotz Sehschwächen einfach benutzbar ist.

Matteo entsteht im Rahmen des Projektes „Sinnvoll Aktiv“ in der Untergruppe „Active Aging“ und wird in enger Kooperation mit der Universität Wien durchgeführt.

INFO

Schuljahr	2025/26
Projektleitung	Laurenz Wimmer
Stv. Projektleitung	Nandor Bagari
Rolle PMA	Mika Böhm
Rolle PMA	Lisa Pripfl
Auftraggeber	Universität Wien
Gesamtstunden	850 Stunden



universität
wien

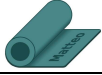
cadaxo



PRINTSHOP®
LANDSTRASSE

Abbildung 15 Plakat

Für die Tage der offenen Tür an der Schule wurden von allen Diplomarbeitsteams je ein Plakat für einen Gesamtüberblick über alle Projekte erstellt. Das Plakat im A1-Format wurde anhand der Plakatvorlage für Diplomarbeiten der Schule erstellt und sind in den Gängen ausgestellt. JLCPCB ist hier noch als Sponsor angeführt, weil sie Rabatte für Bestellungen zugesagt haben, die aber nicht in Anspruch genommen wurden, da noch genügend Sponsor-Geld verfügbar war.



1.5.9.4 Projektwettbewerb Jugend-Innovativ

Das Projektteam hat entschieden, am 39. Jugend-Innovativ-Wettbewerb teilzunehmen, bei dem die besten Projekte von Jugendlichen zwischen 15-20 Jahren ausgezeichnet und der Titel der „Innovativsten Schule“ des jeweiligen Bundeslandes vergeben werden. Dazu wurde ein Projektbericht² verfasst, der einen Überblick über die Idee gibt, die technische Umsetzung erklärt und die Ergebnisse zum Zeitpunkt der Einreichung zusammenfasst. Folgende Termine waren für die Einreichung wichtig³:

- Projektanmeldung von **1. Oktober bis 30. November 2025**
- Konzepte für Projekt-Bonus einreichen bis **30. November 2025**
- Pre-Check aller Anmeldungen & Mitteilung über das Ergebnis bis spätestens **19. Dezember 2025**
- Projektbericht-Abgabe bis **30. Jänner 2026**

Der Projektbericht wird dann von der entsprechenden Fachjury bewertet und gegebenenfalls für die Aufstieg ins Bundesfinale ausgewählt, das vom 27.-29.5.2026 in Wien stattfindet.

Matteo wurde in der Kategorie Engineering I eingereicht, in der Maschinenbau, Mechatronik und Industrial Engineering enthalten sind.

Ebenfalls wurde ein Antrag für den Eco-Bonus gestellt, bei dem es um soziale, ökologische und wirtschaftliche Nachhaltigkeit geht. Die Ergebnisse wurden allerdings noch nicht ausgesendet.

Weitere Informationen zum Wettbewerb sind auf der Jugend-Innovativ-Website zu finden: <https://www.jugendinnovativ.at/>

1.5.10 Team-Coaching

Im Oktober und November wurde im Rahmen eines Freifachs ein Team-Coaching in Anspruch genommen. Dabei wurden Probleme bei der Kommunikation im Team aufgearbeitet, Verbesserungsvorschläge gesucht und ein Fragebogen zur Arbeit im Team ausgefüllt. Die Ergebnisse wurden teamintern besprochen und Probleme in weiteren Projektverlauf berücksichtigt.

1.5.11 Meetings

Während der Durchführung wurden alle 14 Tage Jour-Fixe-Meetings mit allen Betreuern durchgeführt. Dabei wurden alle aktuellen Entwicklungen, Ergebnisse und Probleme besprochen. Außerdem wurden die wichtigsten Schritte festgelegt, die bis zum nächsten Jour-Fixe-Meeting zu erledigen waren.

² Projektbericht im Anhang zu finden (9.4)

³ Quelle: [28]

1.5.12 Arbeitszeiten

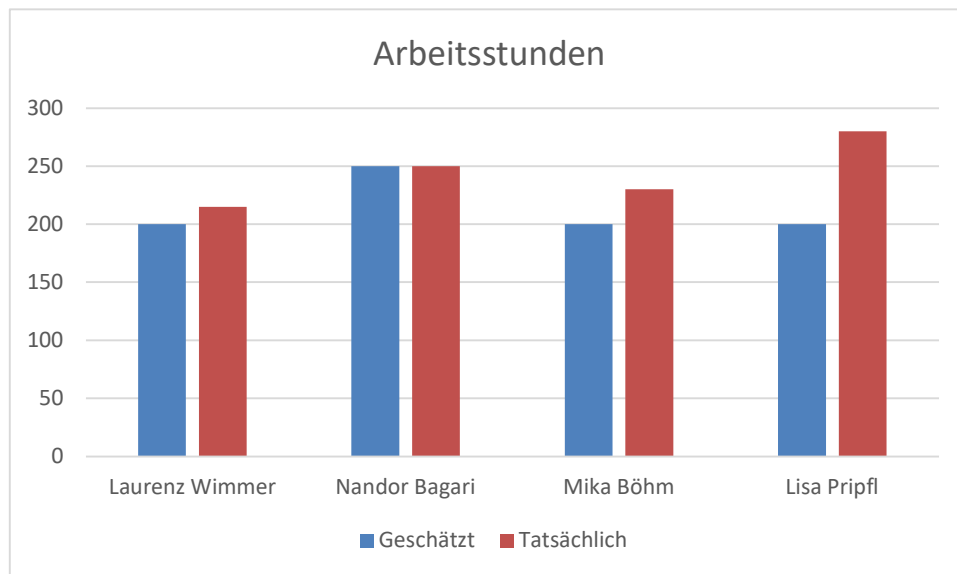


Abbildung 16 Arbeitsstunden

Die Arbeitsstunden wurden von jedem Projektmitglied am Anfang anhand des vermuteten Aufgabenumfangs geschätzt. Diese Schätzungen haben summiert einen Arbeitszeitaufwand von 850 Stunden ergeben.

Die tatsächliche, für das Projekt aufgewendete Arbeitszeit, beträgt ca. 975 Stunden. Die zusätzlichen 125 Arbeitsstunden waren nötig, um unerwartete Probleme zu lösen.

2 Elektronik

2.1 Allgemeines Konzept

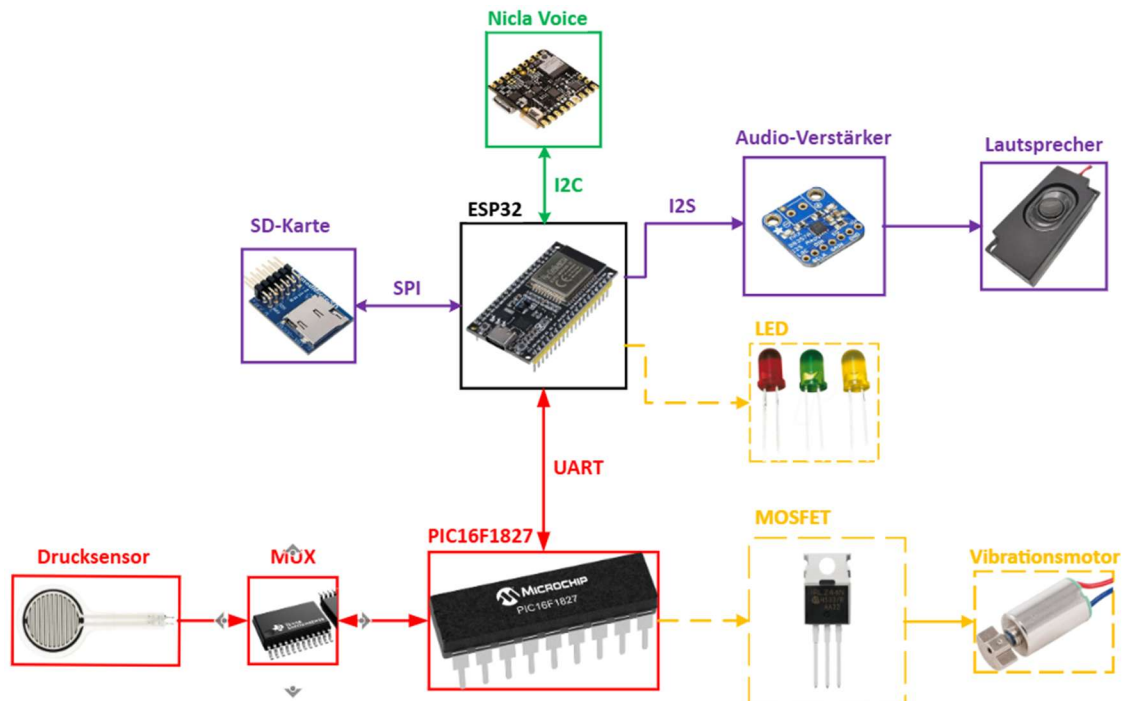


Abbildung 17 Blockbild

Abbildung 17 zeigt alle Komponenten der Elektronik, die auf der Platine eingeplant wurden. Außerdem sind alle Kommunikationswege mit den entsprechenden Datenübertragungsprotokollen darin inkludiert.

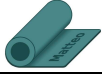
Rot umrandet sind die Komponenten, die für die Messung verwendet werden. Die 12 verbauten Sensoren sind über einen Multiplexer (MUX) mit dem Microcontroller PIC16F1827 verbunden und werden von diesem gemessen.

Violett umrandet sind alle Komponenten, die mit der Sprachausgabe zu tun haben. Die auf der SD-Karte sind die Audiodateien abgespeichert. Die benötigte Datei wird vom Microcontroller ESP32 ausgewählt und über den Audioverstärker an den Lautsprecher gesendet, der sie dann ausgibt.

Grün umrandet ist der Arduino Nicla Voice. Mit diesem Microcontroller wird die Sprachsteuerung umgesetzt.

Gelb strichliert umrandet sind alle Komponenten, die zur Erfüllung der optionalen Ziele benötigt werden. Es wurden alle Komponenten auf der Platine vorgesehen, der MOSFET und der Vibrationsmotor wurden allerdings nicht bestückt.

Das Herzstück der Steuerung ist der ESP32-Microcontroller. Er ist schwarz umrandet und wurde gewählt, weil er alle 4 Übertragungsprotokolle unterstützt, die benötigt werden: I2C, I2S, SPI und UART. Alle Bauteile und Protokolle werden in den folgenden Kapiteln genauer beschrieben.



2.2 Stromversorgung

Bei der Stromversorgung der Matte waren einige Herausforderungen zu lösen:

- **Zwei unterschiedliche Spannungslevel:** Dadurch, dass auf der Platine Microcontroller mit 5 V- und 3,3 V-Versorgung verwendet werden, müssen beide Spannungslevel erzeugt werden. Außerdem ist es wichtig, die beiden Spannungslevel strikt zu trennen, da andernfalls die Microcontroller mit 3,3 V-Versorgung durch Überspannung beschädigt werden.
- **Versorgung über Akkus:** Aufgrund dessen, dass die Verletzungsgefahr auch bei Sehbeeinträchtigungen möglichst geringgehalten werden muss, wurden Akkus als Versorgung für die Matte gewählt. So entsteht keine Stolpergefahr durch eine Versorgungsleitung, die an einer Steckdose angesteckt ist. Diese Akkus müssen allerdings geladen werden und ändern ihre Spannung abhängig vom Ladestand. Bei einer Versorgung über ein handelsübliches Netzteil wäre der 5 V-Spannungsspiegel bereits gegeben, muss nun aber ebenfalls über ein Modul erzeugt werden.
- **Schalter zur Aktivierung der Matte:** Um ein Bücken vermeiden zu können, ist es notwendig, die Matte mit einem Fuß ein- und ausschalten zu können. Deshalb wurde ein entsprechender Schalter gewählt, der durch Drücken betätigt werden kann.
- **Schaltplan erstellen:** Über die verwendeten Module gibt es nur sehr wenig Dokumentation, weswegen für die Erstellung des Schaltplans eine eigene Fusion-Bibliothek angelegt werden musste.

Die einzelnen Komponenten sowie die Schaltung der Stromversorgung werden anschließend genauer beschrieben.

2.2.1 Akkus

Abbildung 18 zeigt die gewählten 18650-Lithium-Ionen-Akkus. Die Bezeichnung gibt dabei die Dimensionen der Akkus in mm an:

Die Akkus haben einen Durchmesser von $\varnothing 18$ mm und eine Länge von 65 mm. Der 0er am Ende der Nummer steht für die runde Bauform, Lithium-Ionen ist der Akku-Typ. Andere Typen sind beispielsweise Lithium-Polymer-Akkus oder Bleiakkus.

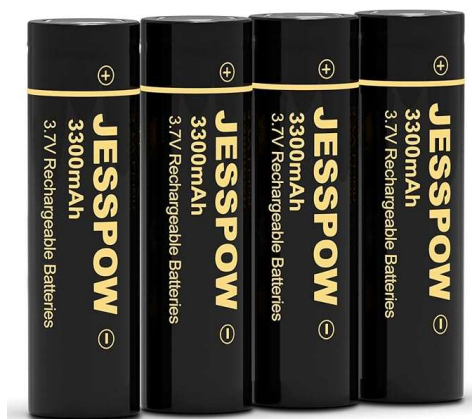


Abbildung 18 Akkus [29]

Die Wahl ist auf diese Akkuzelle gefallen, weil sie weit verbreitet und dementsprechend einfach verfügbar und austauschbar ist. Sie gehört zu den Standard-Baugrößen und wird unter anderem in Powerbanks verwendet. Außerdem hat sie eine Nennspannung von 3,7 V mit Schwankungen von $\pm 0,5$ V, abhängig vom Ladestand. Dieser Spannungsbereich passt gut zu den benötigten Spannungslevels.

Das eigens erstellte Fusion-Schaltsymbol bezieht sich nicht direkt auf die Akkus, sondern auf die Akku-Halterungen, die auf die Platine gelötet werden. Im 3D-Modell der Platine ist die Halterung allerdings mit Akkus modelliert.

2.2.2 Schalter

Der in Abbildung 19 gezeigte Schalter wird über Drücken aktiviert und deaktiviert. Er hat zwei verschiedene Endpositionen, eine für den eingeschalteten und eine für den ausgeschalteten Zustand.

Gewählt wurde diese Art des Schalters, weil er sehr einfach mit dem Fuß zu bedienen ist. Außerdem ist es ohne viel Aufwand möglich, die Druckfläche des Schalters im Rahmen der Konstruktion des Gehäuses zu vergrößern.

Ein Kippschalter wäre zwar auch leicht ertastbar gewesen, ist aber meist sehr klein. Dementsprechend wäre es schwierig, auf die richtige Seite des Kippschalters zu drücken, ohne gleichzeitig auf der anderen Seite Druck auszuüben. Eine Vergrößerung der Fläche ist aufgrund des Kippens keine Option.



Abbildung 19 Schalter

2.2.3 Akku-Lademodul

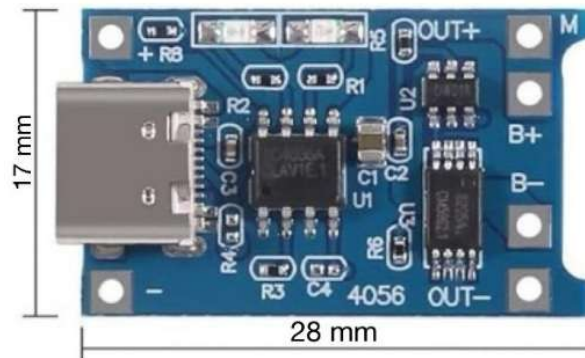


Abbildung 20 Lademodul Maße [30]

Abbildung 20 zeigt das Akku-Lademodul. Es basiert auf dem Chip TP4056, der dafür designed ist, den Ladevorgang einer Lithium-Ionen-Zelle zu steuern und zu überwachen. Das Modul lädt die Akkus mit einer Spannung von 4,2 V auf und stoppt den Ladevorgang, sobald der Ladestrom niedrig genug ist. Der Ladestrom ist am Anfang des Ladevorgangs am höchsten, da der Spannungsunterschied zu diesem Zeitpunkt am größten ist. Wenn die Akkuzelle eine Spannung nahe an 4,2 V erreicht, ist der Spannungsunterschied so gering, dass praktisch kein Strom mehr fließt.

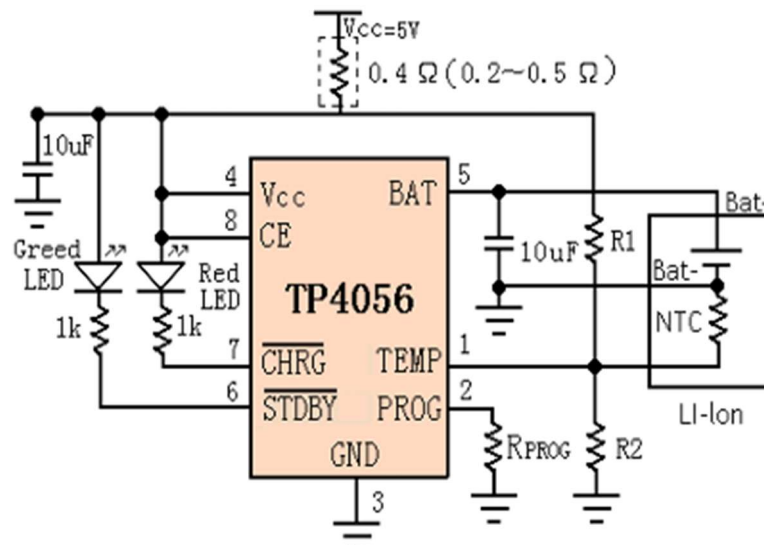


Abbildung 21 Datenblattauszug Beschaltung Chip [34]

Die übliche Beschaltung des Chips ist in Abbildung 21 zu sehen. R_{PROG} ist der Widerstand, über den der maximale Ladestrom festgelegt wird. Das verwendete Modul ist auf 1000 mA eingestellt⁴.

Das Modul hat zusätzliche Schutzfunktionen wie einen Überstromschutz und zeigt über eine rote LED an, ob gerade geladen wird sowie über eine blaue LED, dass der Akku voll ist.

⁴ Eine Tabelle mit allen Werten für R_{prog} ist im Datenblatt im Anhang zu finden: Abschnitt 9.7.4

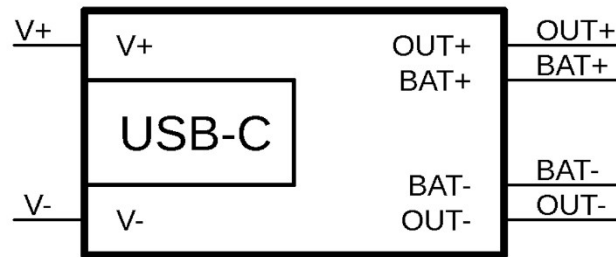


Abbildung 22 Schaltsymbol Lademodul

Abbildung 22 zeigt das erstellte Schaltsymbol für das Lademodul. Es hat 6 Pins mit folgenden Funktionen:

- **V+/V-**: Zwischen diesen Pins liegt im Falle eines angesteckten USB-C-Kabels eine Spannung von 5 V an, solange die Versorgung über ein entsprechendes Netzteil erfolgt.
- **BAT+/BAT-**: Dieses Pins führen zu den Polen des Akkus und werden zum Auf- und Entladen verwendet.
- **OUT+/OUT-**: Output-Pins des Moduls, bei angestecktem Kabel liegen hier 4,2 V an, andernfalls die aktuelle Batteriespannung.

2.2.4 Step-Up-Converter

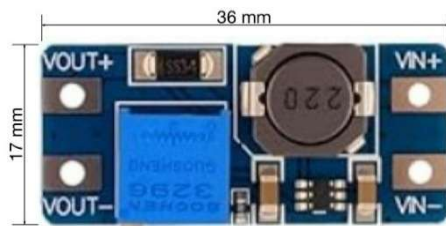


Abbildung 24 Step-Up Maße [31]

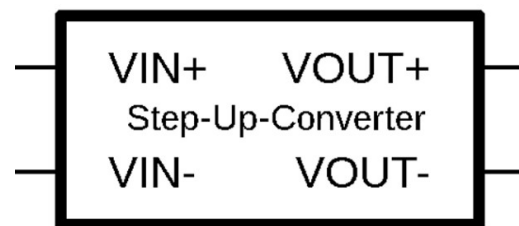


Abbildung 23 Step-Up Schaltsymbol

Abbildung 24 zeigt den verwendeten Step-Up-Converter. Er arbeitet mit einer Eingangsspannung zwischen 2-24 V und wandelt sie auf Ausgangsspannungen zwischen 5-28 V. Die Spannung wird über das Potentiometer eingestellt. Um die Ausgangsspannung zu erhöhen, muss die Schraube des Potentiometers im Uhrzeigersinn gedreht werden, zum Senken der Spannung gegen den Uhrzeigersinn.

Dieses Modul wurde bereits beim FIV-Projekt in der 4. Klasse verwendet und hat gut funktioniert. Der maximale Output-Strom von 2 A ist auch ausreichend für diese Anwendung, weswegen das Modul für dieses Projekt gewählt wurde.

Abbildung 23 zeigt das erstellte Schaltsymbol. Die Pins entsprechen den Anschlüssen des Moduls, sind allerdings zur besseren Darstellung im Schaltsymbol anders angeordnet als auf dem Modul. Der Footprint wurde mit der richtigen Anordnung erstellt.

2.2.5 Step-Down-Converter

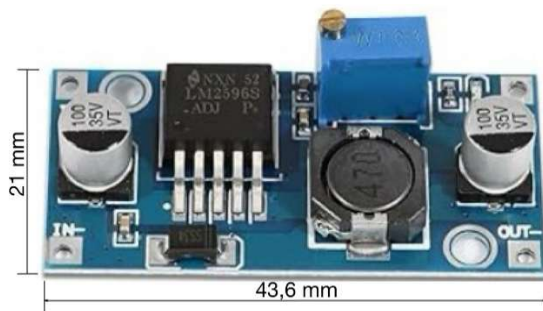


Abbildung 25 Step-Down Maße [32]

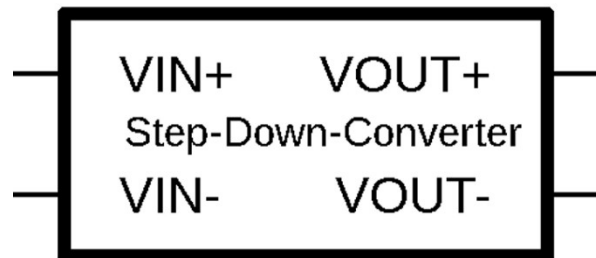


Abbildung 26 Step-Down Schaltsymbol

Abbildung 25 zeigt den verwendeten Step-Down-Converter. Er arbeitet mit einer Eingangsspannung zwischen 3,2-40 V und hat eine Ausgangsspannung zwischen 1,25-35 V. Tests haben allerdings ergeben, dass eine stabile Ausgangsspannung von 3,3 V erst ab einer Eingangsspannung von 4 V möglich ist.

Dieses Modul wurde gewählt, da es genug Ausgangsstrom liefern kann (max. 3 A auf kurze Zeit, im Dauerbetrieb max. 2,5 A) und einige zusätzliche Schutzfunktionen wie Überspannungs- und Überstromschutz verbaut hat.

Das in Abbildung 26 dargestellte Schaltsymbol ist den Anschlüssen des Moduls entsprechend erstellt worden. Auch die Pins sind richtig sortiert.

Sowohl für das Step-Up- als auch das Step-Down-Modul wird vom Verkäufer kein Datenblatt bereitgestellt, es gibt nur jeweils eine kurze Bedienungsanleitung in der Produktbeschreibung auf der Website des Händlers.

2.2.6 Schaltplan

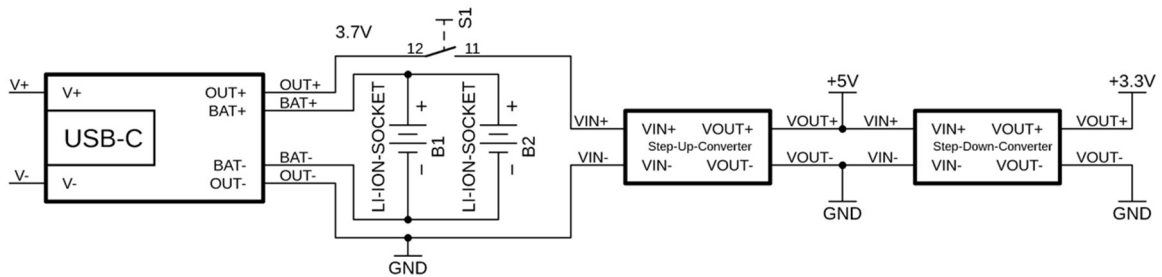


Abbildung 27 Schaltplan Stromversorgung

Abbildung 27 zeigt den Schaltplan der Stromversorgung. Von links nach rechts sind das Akku-Lademodul, die Akkus und die beiden Spannungswandler verbaut. Genau wie abgebildet, ist auch die Anordnung auf dem Platinenlayout, siehe Abbildung 28.



Akku-Lademodul Akkus mit Halterung Step-Up- (L) / Step-Down-Converter (R)

Abbildung 28 Stromversorgung Platine

2.2.6.1 Spannungskonverter

Aufgrund des oben beschriebenen Spannungsabfalls des Step-Down-Converters war die einzige Möglichkeit, den Step-Down-Converter über den 5 V Output des Step-Up-Converters zu versorgen. Diese Lösung ist nicht optimal, weil so die Wirkungsgrade beider Konverter ausgehend von der nachfolgenden Schätzung auf 99,3% des Stromes angewendet werden müssen und sich in Kombination nur ein Wirkungsgrad von ca. 65% bei einem angenommenen Wirkungsgrad von 80% pro Konverter ergibt. Bei dieser Schätzung ist jedoch ein gewisser Sicherheitsfaktor miteinbezogen worden. Der tatsächliche Wirkungsgrad der beiden Spannungswandler könnte bis zu 90% betragen, dazu gibt es allerdings keine Angaben des Herstellers.



2.2.6.2 Geschätzter Stromverbrauch

Zur Auslegung der Akkulaufzeit wurde eine Schätzung des maximalen Stromverbrauchs der Bauteile anhand von Daten aus Datenblättern und Herstellerangaben durchgeführt.

Geschätzter Stromverbrauch pro Spannungslevel			
3,3 V-Seite	I in mA	5 V-Seite	I in mA
ESP32	500	PIC16F1827	5
Adruino Nicla Voice	5	Sensoren	1
Lautsprecher	300		
2 LEDs	30		
Summe 3.3 V	835	Summe 5 V	6

Tabelle 8: Stromverbrauch Schätzung

Daraus wurde im Anschluss der Stromverbrauch der gesamten Schaltung berechnet:

Berechnung des gesamten Verbrauchs		
Erklärung	Rechnung	Ergebnis
Geschätzter Wirkungsgrad Step-Down		80%
Geschätzter Wirkungsgrad Step-Up		80%
Einberechnung Wirkungsgrad Step-Down, Input Step-Down in mA	Summe 3,3 V / 0,8	1043,75
Addition Summe 5 V, Output Step-Up in mA	Input Step-Down + Summe 5 V	1049,75
Einberechnung Wirkungsgrad Step-Up, Input Step-Up, Gesamtstrom in mA	Output Step-Up / 0,8	1312,1875

Tabelle 9: Stromverbrauch Berechnung

Auf Basis dieser Berechnung wurde entschieden, zwei Akkus mit jeweils 3,3 Ah zu verwenden, da die Akkus zur Verfügung standen und so ein großzügiger Puffer zu dem Minimum von einer Betriebsstunde vorhanden ist.

Bei der Berechnung wurde eine maximale Belastung zugrunde gelegt, weswegen die tatsächliche Betriebsdauer bei ca. 8-10 Stunden deutlich höher liegt, als gefordert ist. Die längere Betriebsdauer hat auch den Vorteil, dass Matteo länger genutzt werden kann, ohne Hilfe von anderen Personen zum Anstecken und Aufladen der Akkus in Anspruch nehmen zu müssen. Dadurch wird die Unabhängigkeit der Zielpersonen weiter gefördert.

2.3 Messschaltung

Die Messschaltung ist zu Testzwecken als separate Platine designed worden, auf der nur die zur Messung der Belastung notwendigen Bauteile verbaut wurden. Hiermit konnten die Drucksensoren (Siehe 2.4.1.1) ausgetestet und eine Möglichkeit der frühzeitigen Programmierung des Messprogramms geboten werden, ohne die finale Platine fertiggestellt zu haben.

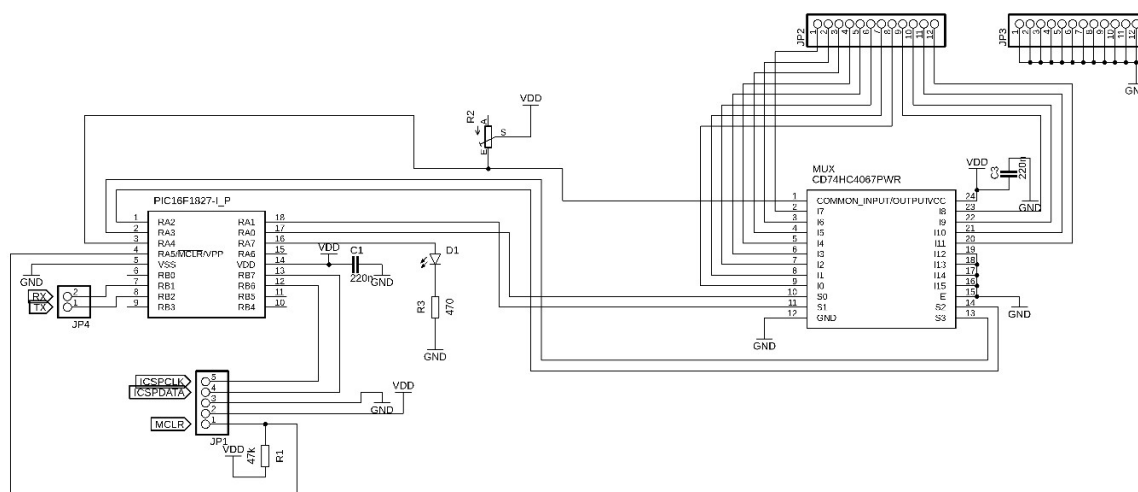


Abbildung 31 Schaltplan Messschaltung

2.3.1 Bauteile Messschaltung

2.3.1.1 Multiplexer

Der CD74HC5067 (Siehe 9.7.1) ist ein 16-Channel analoger Multiplexer (MUX) von Texas Instruments. Hiermit kann man 16 verschiedene Channels mit einem Ausgangs-Channel verbinden. Mittels vier Steuereinsparungen kann zwischen den einzelnen Channels gewählt werden. Der MUX ermöglicht es, dass die Sensoren alle mit derselben Messschaltung und demselben ADC verbunden sind. Beim Testen wurde festgestellt, dass die Verbindung einen Innenwiderstand von circa $50\ \Omega$ hat. Auf Sensoren wie den hier verwendeten, je selben Widerstand haben und deshalb erhöht.

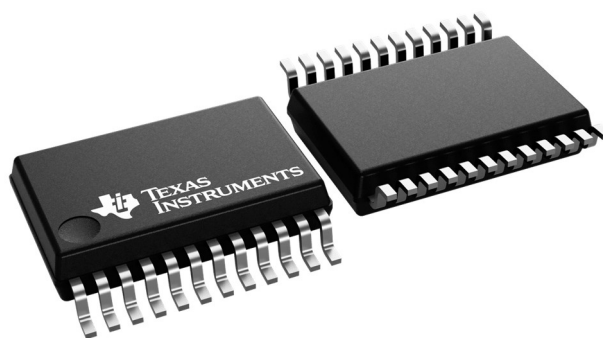


Abbildung 32 MUX (CD74HC4067) [11]

nenwiderstand von circa $50\ \Omega$ hat. Auf den ersten Blick ist das schlecht bei resistiven Sensoren wie den hier verwendeten, jedoch ist das kein Problem da alle Channels den selben Widerstand haben und deshalb es nur den seriellen Messwiderstand um $50\ \Omega$ erhöht.

2.3.1.2 Microcontroller

Der PIC16F1827 (nachfolgend als PIC bezeichnet) ist ein 8-Bit Microcontroller (μC) von Microchip. Er verfügt über 16 Ein-/Ausgänge mit jeweils unterschiedlichen Funktionen (Siehe 9.7). Er ist für das Messen und Auswerten der Daten der Sensoren zuständig. Gewählt wurde dieser Microcontroller aus mehreren Gründen:

- Niedriger Verbrauch: Der durchschnittliche Stromverbrauch von rund $100\ \mu\text{A}$ ist praktisch für den Akkubetrieb, da eine möglichst lange Betriebszeit garantiert werden kann.
- Einfaches Design: Der PIC beinhaltet alle Funktionen, die wir benötigen, ohne überflüssige Module wie Wifi oder Bluetooth bereitzustellen. Dadurch werden die Baugröße, der Stromverbrauch in die Kosten im Vergleich zu anderen Produkten reduziert.
- Erfahrungswerte: In der 4. Klasse wurde bereits ein Projekt mit diesem Prozessor durchgeführt, weswegen mögliche Probleme und Funktionsweisen bekannt waren. So konnte der Programmieraufwand minimiert werden.
- Bauform: Die 18-Pin-PDIP-Bauform des Gehäuses ist vereinfacht den Einbau und das Anlöten an die Platine.

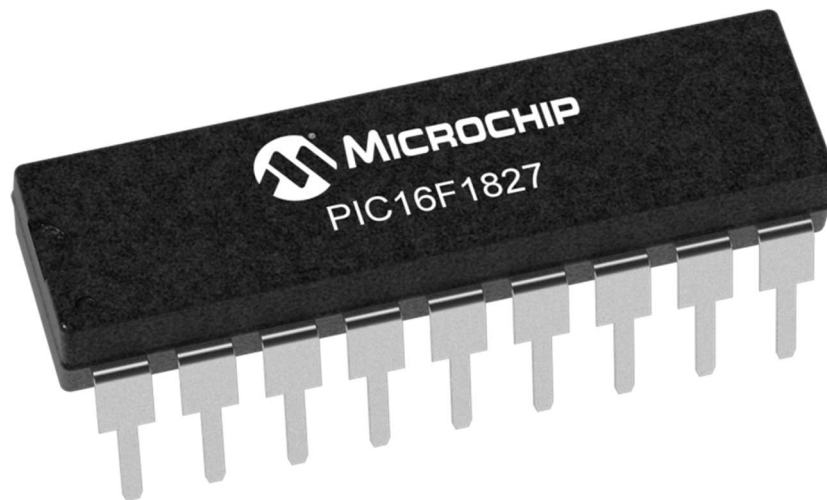


Abbildung 33 μC (PIC16F1827) [9]

Um diesen Idealwert herauszufinden wurden verschiedene Widerstandswerte getestet:

In Abbildung 37 kann man die Messergebnisse bei $1\text{ k}\Omega$ sehen. Man konnte sehr viel Kraft aufragen, die Spannung ist jedoch nie unter 4 V gefallen, was in puncto Auflösung nicht ideal ist.

In Abbildung 36 sieht man die Messergebnisse bei $47\text{ k}\Omega$. Die Auflösung ist zwar sehr gut, jedoch ist die Spannung bereits bei geringer Belastung ganz bis 0 V runtergegangen.

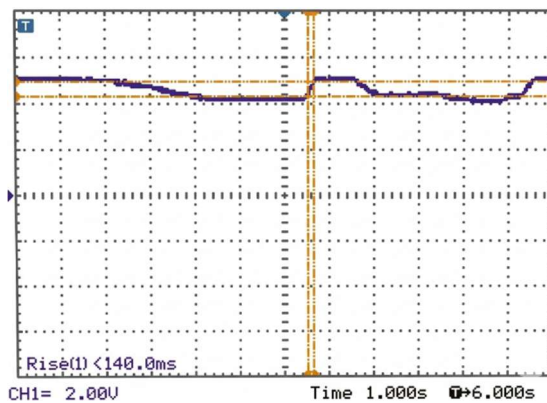


Abbildung 37 Oszi $1\text{ k}\Omega$

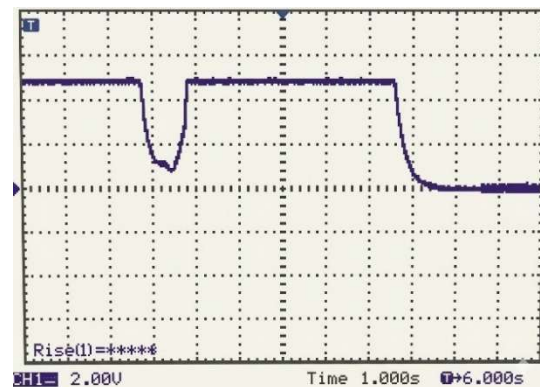


Abbildung 36 Oszi $47\text{ k}\Omega$

Einen guten Kompromiss kann man bei $10\text{ k}\Omega$ finden, wie man an Abbildung 38 erkennen kann. Hier sind die maximale Belastung und die Auflösung in einem guten Zusammenspiel.

Da bei diesen Tests die Sensoren nur mit der Hand gedrückt wurden und damit wenig belastet waren, wurde entschieden, einen noch kleineren Widerstandswert zu wählen: **$4,7\text{ k}\Omega$** . Bei späteren Tests hat sich ergeben, dass dieser auch sehr gut passt.

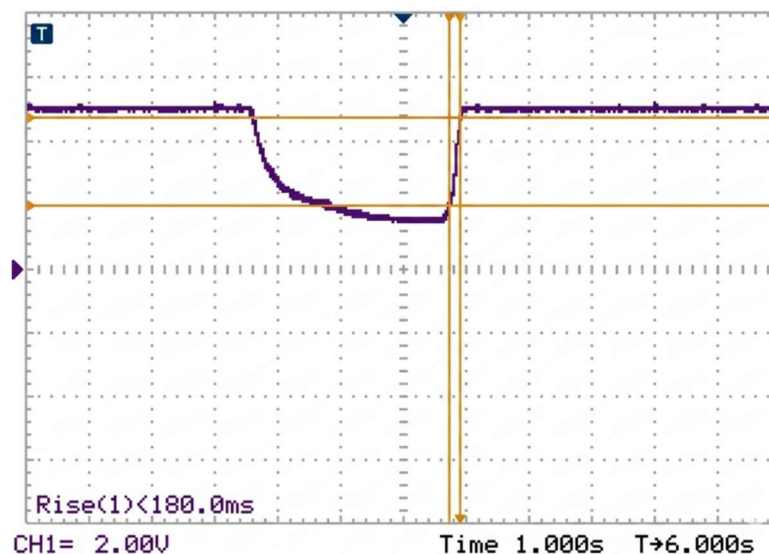


Abbildung 38 Oszi $10\text{ k}\Omega$



Der PIC wird für die Messung der Sensordaten und Steuerung des MUX verwendet. Zur Programmierung wird eine Schnittstelle benötigt, die man in Abbildung 39 unten sehen kann. Die fünf Leitungen zur Programmierung sind:

- Um die Messdaten auch sehen zu können, ist eine UART-Schnittstelle an den Pins RB1 und RB2 konzipiert.

Um eine sichtbare Rückmeldung von der Platine zu bekommen, ist eine LED verbaut.

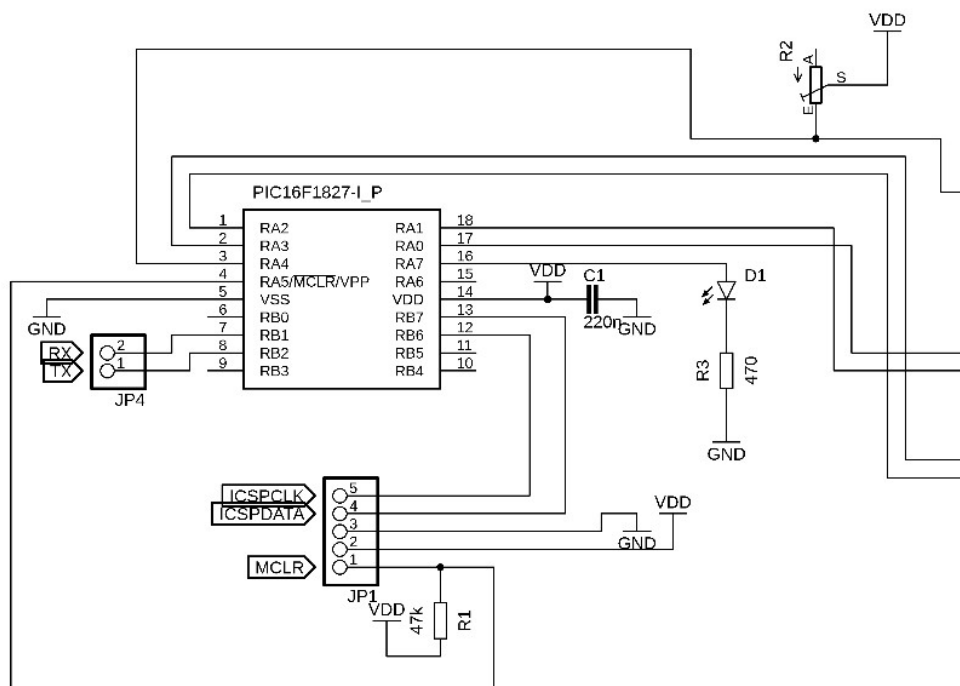


Abbildung 39 Messschaltung PIC

2.3.3 Bild Messplatine

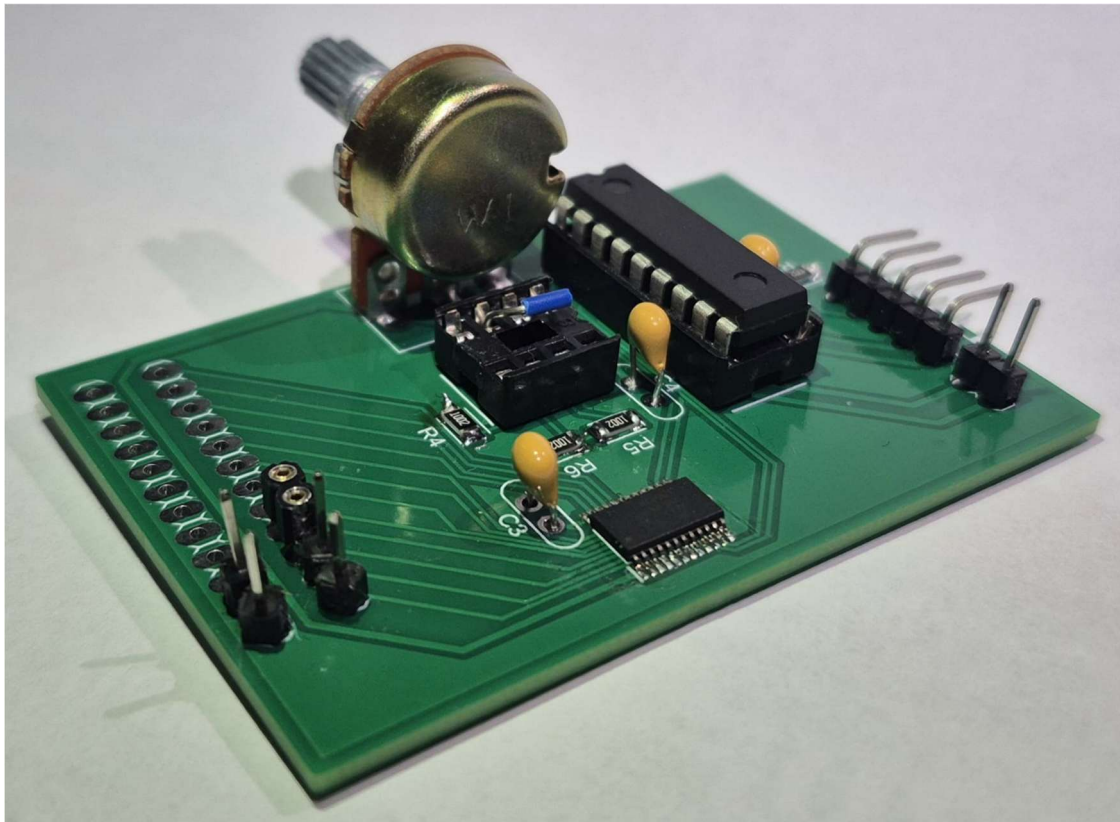


Abbildung 40 Bild Messplatine

Abbildung 40 zeigt den Prototyp der Messschaltung, mit dem die ersten Tests mit Sensoren durchgeführt wurden. Auf der linken Seite sind die Anschlüsse für die Sensoren, der MUX ist im Vordergrund und die Schnittstellen für den PIC sind auf der rechten Seite platziert.

2.3.4 Vorversionen Messplatine

2.3.4.1 Version 1

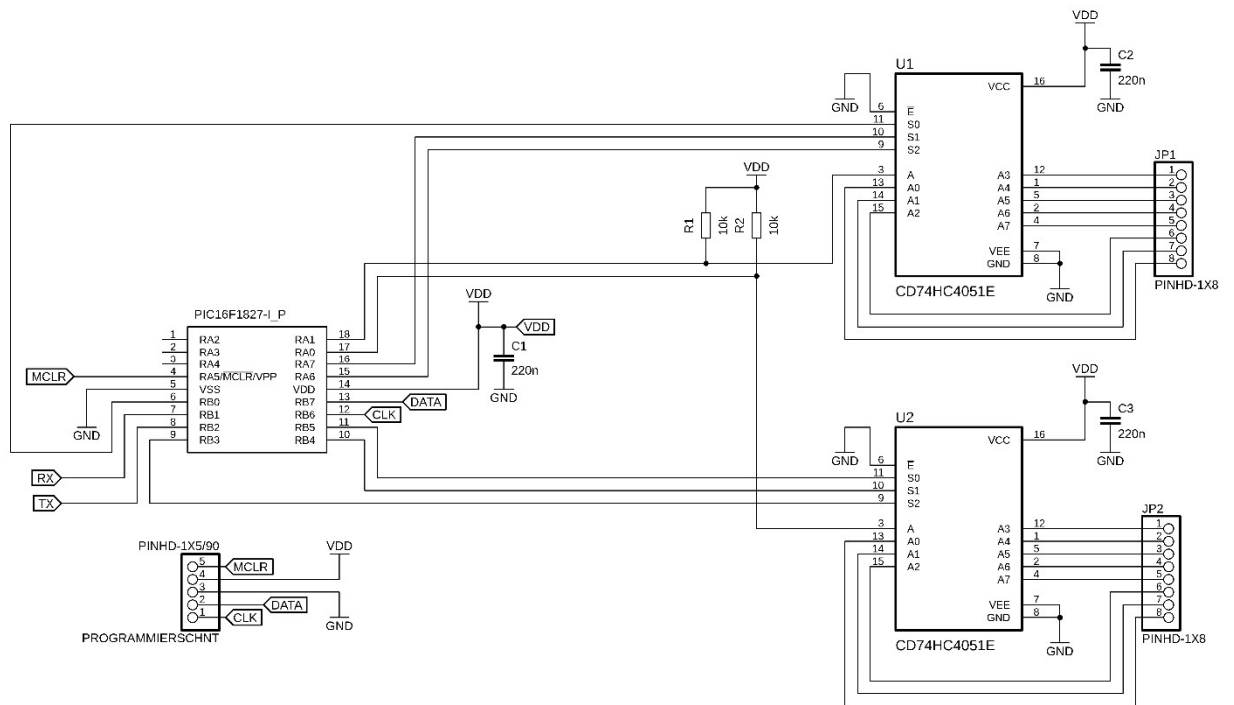


Abbildung 41 Messplatine Vorversion 1

In der ersten Version der Platine waren noch andere MUX eingeplant. Hier wurde der CH74HC4051 analoge Multiplexer verwendet. Dieser hat nur acht Kanäle, daher hätten wir für die 12 Sensoren zwei von ihnen gebraucht. Ein Vorteil wäre gewesen, dass diese MUX in PDIP-Bauform existieren, was um einiges einfacher zu Löten ist als die SOIC-Bauform des anderen. Nachteile wären einerseits, dass diese Variante drei weitere Ausgänge des Microcontrollers verwendet hätte. Andererseits könnten Messabweichungen entstehen, da die zwei MUX mit anderen ADCs und anderen Messwiderständen (mögliche kleine Abweichungen) mit anderen Leiterbahnen verbunden gewesen wären.

Die Schaltung in Abbildung 41 ist nicht vollständig (fehlende LED, 10 kΩ), da diese nicht weiterentwickelt wurde, nachdem die Entscheidung für den 16-Kanal-MUX gefallen ist.

2.3.4.2 Version 2

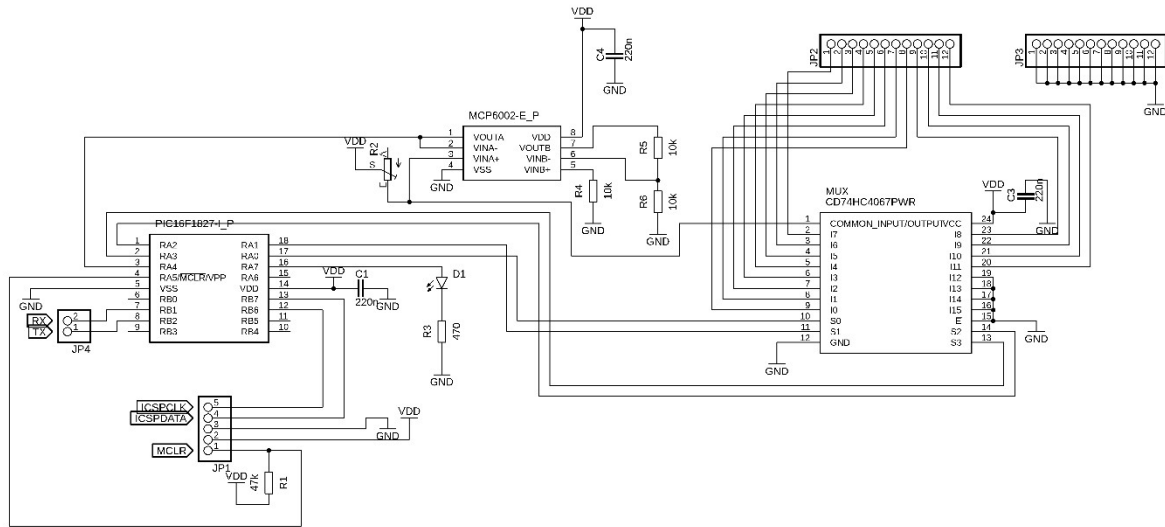


Abbildung 42 Messplatine Vorversion 2

Zuerst wurden als Messeinrichtung ein Operationsverstärker als Spannungsfolger eingebaut. Da die Belastung der Messschaltung mit den Sensoren durch den ADC Messungenauigkeiten verursachen könnte, wurde entschieden diesen einzubauen. Bei den ersten Tests ist das Problem aufgetreten, dass der OPV Störungen verursacht und Differenziator-Verhalten gezeigt hat. Folgend wurde der OPV entfernt und die Messschaltung direkt mit dem Microcontroller verbunden. Weil der ADC-Strom gering genug ist, um vernachlässigt werden zu können, sind ohne die OPV-Entkopplung keine Probleme bei der Messung aufgetreten. Die Anschlüsse VINA+ und VOUTA wurden kurzgeschlossen (Verbindung vom MUX-Ausgang zum ADC-Eingang).

2.4 Platine

Die Platine ist das Rückgrat des Projekts, das alle elektronischen Bauteile miteinander verbindet. Sie wurde möglichst schmal und lang entworfen, um gut an einer Seite der Matte im Gehäuse befestigt werden zu können.

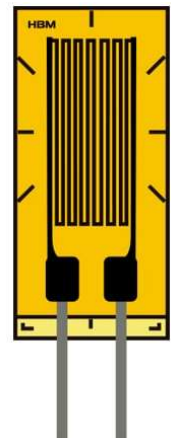
2.4.1 Bauteile

Auf dieser Platine sind alle Bauteile verbaut. Die bereits vorhin beschriebenen Bauteile (Kapitel 2.2 und 2.3) werden hier nicht noch einmal aufgelistet.

2.4.1.1 Drucksensoren

Zur Messung der Balance muss die Belastungen der Füße einzeln gemessen und verglichen werden. Zur Auswahl standen zwei verschiedene Arten von Sensoren:

- **Dehnungsmessstreifen (DMS):** Dehnungsmessstreifen sind die klassische Wahl, wenn es zu Kraftmessung kommt. Hierbei handelt es sich um einen Sensor in Dünnschichttechnik mit zwei Anschlüssen. (Siehe Abbildung 43 DMS). Um Kräfte messen zu können, wird die Widerstandsänderung der dünnen Leiterbahnen bei Dehnung oder Stauchung ausgewertet. Durch die Meandrierung der Leiterbahnen im Sensor wird dieser Effekt vervielfacht. Wird der Sensor auseinandergezogen, so wird die Leitung länger und dünner und somit erhöht sich der Widerstand. Das Gegenteil passiert beim zusammen Drücken des DMS.

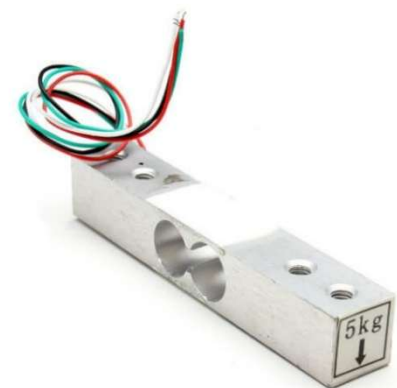


Durch den in Abbildung 44 gezeigten Aufbau, einer *Load Cell*, können senkrechte Kräfte in Biegung und Dehnung umgewandelt werden.

Abbildung 43 DMS [1]

Auf einer Load Cell (auf Deutsch Wägezelle), einer oder mehrere DMS montiert werden

DMS und Load Cells gibt es in vielen Varianten, für unterschiedliche Anwendungsfälle und unterschiedliche Belastungen. Leider sind diese Sensoren auch besonders temperaturabhängig, weshalb man sie oft zusammen mit Wheatstone-Brücken verwendet, um diesen Effekt auszugleichen.



Aufgrund einiger Nachteile haben wir uns nicht für DMS entschieden. Die zur Messung notwendigen Wheatstone-Brücken sind umständlich zu bauen, besonders für unseren Anwendungsfall, da wir mehrere von ihnen brauchen würden. Auch sind Load Cells relativ dick, was ein Hindernis wäre, um die Matte so dünn wie möglich zu machen.

Abbildung 44 Load Cell [38]

- **Force-sensing Resistor (FSR):** FSR sind die Alternative zu DMS. Hier handelt es sich ebenfalls um einen dünnen Sensor mit zwei Anschlüssen und einer dünnen Leitung (hellgrau) innen, die mehrfach von Seite zu Seite gelegt wird (Siehe Abbildung 46 und einer zum Teil leitfähigen Paste. Die Funktionsweise ist etwas anders als die von den DMS. Der FSR besteht aus mehreren Schichten wie auf Abbildung 45 ersichtlich. Hierbei sind die mittleren zwei von Relevanz. Die leitfähigen Silberbahnen (Conductive Silver Traces) haben zwei äußere Ringe mit langen Bahnen, die sich fast zur anderen Seite Strecken. Die zwei Ringe sind dabei nicht direkt verbunden, sondern nur sehr knapp nebeneinander. Daher ist im unbelasteten Zustand auch keine Leitfähigkeit vorhanden, der Sensor ist sehr hochohmig. Beim Drücken auf die Sensorfläche rücken die leitfähigen Partikel in der Tinte (Resistive Carbon Elements) näher zusammen, wodurch der Widerstand in den $k\Omega$ -Bereich sinkt.



Abbildung 46 FSR [39]

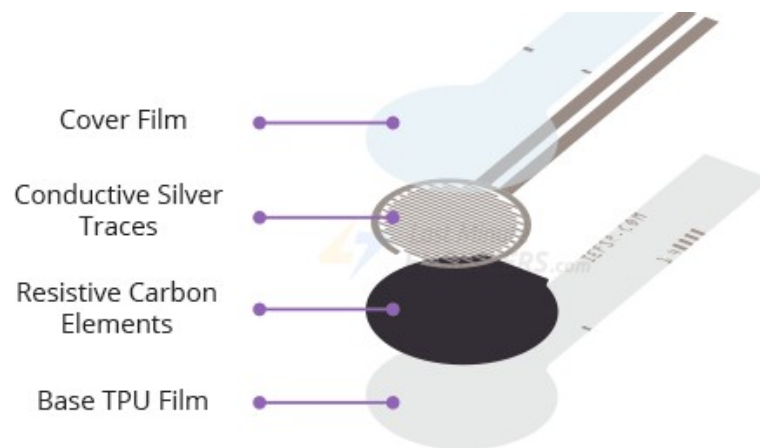


Abbildung 45 FSR Aufbau [40]

An Abbildung 47 ist abgebildet, wie sich der Widerstand bzw. die Leifähigkeit mit der Kraft ändert.

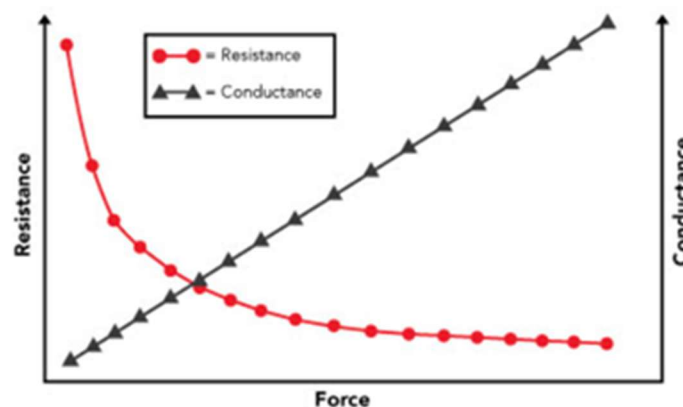


Abbildung 47 FSR Kraft-Widerstand [41]

Im Projekt wurden 12 FSR-Drucksensoren verwendet, sechs pro Fuß. Diese sechs Sensoren wurden weiter in zwei Dreier-Gruppen unterteilt. Diese drei sind gemeinsam auf eine Aluminiumplatte (siehe 3.2) geklebt. Jeweils ein Anschluss der Sensoren sind gemeinsam mit Ground verbunden, die jeweils anderen werden mit einem Kabel an die Platine angeschlossen. Von jeder dieser Dreier-Gruppen gehen somit vier Leitungen zur Platine. Die Leitungen wurden dabei sicher in der Matte verbaut und sind beim Stehen auf der Matte nicht spürbar.

Um die Gewichtskräfte vollständig auf den Sensoren zu verteilen, wurden dünne Unterlagen für die Sensoren konstruiert und 3D-gedruckt. (Siehe Kapitel 3.3.2)

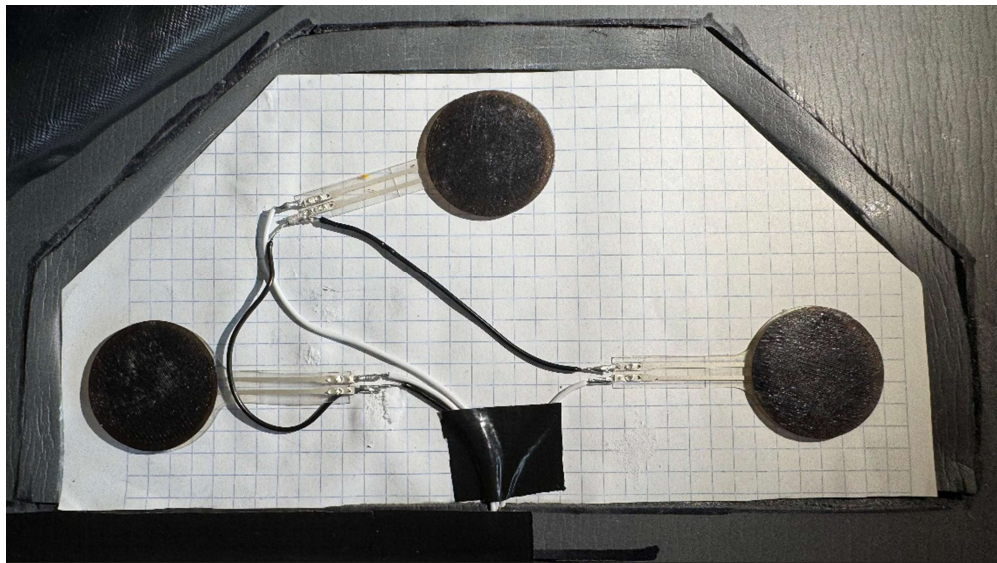


Abbildung 48 Sensoren Lage

2.4.1.2 Microcontroller

Der ESP32 ist ein Microcontroller, der von *Espres-sif Systems* entwickelt wurde. Er ist ein vielseitiger leistungsfähiger Microcontroller mit Funktionen wie Wifi und Bluetooth. Für unsere Zwecke ist er gut geeignet, da er über alle vier Kommunikationsschnittstellen verfügt, die wir benötigen (I²C, I²S, SPI, UART). Der Microcontroller ist besonders einfach zu programmieren. Den Chip kann man bereits verbaut auf einer Platine in Form der „Devkits“ kaufen, wodurch es besonders einfach an der Platine zu befestigen ist.

Der ESP32 ist einerseits für die Steuerung der Abläufe (Verbindungsstelle zwischen Arduino Nicla Voice und PIC) und andererseits für die Sprachausgabe zuständig.



Abbildung 49 ESP32 [17]

2.4.1.3 Arduino Nicla Voice

Der Arduino Nicla Voice ist ein Microcontroller, der speziell für Spracherkennungen entwickelt wurde. Ein „neuronaler Entscheidungsprozessor“ auf dem Board hilft bei gleichzeitiger Ausführung von mehreren KI-Algorithmen und ermöglicht die Verwendung von KI und Machine Learning auf kleinem Raum. Dafür wird üblicherweise besonders viel Rechenleistung und Platz benötigt. Der Microcontroller verfügt bereits über ein Mikrofon, weswegen kein weiteres auf der Platine benötigt wird. Die Kommunikation erfolgt über I²C mit dem ESP32.



Abbildung 50 Arduino Nicla Voice [42]

2.4.1.4 SD-Karten Modul

Das SD-Karten Modul wird benötigt, um die Audiofiles für die Sprachausgabe zu speichern und zu lesen. Auf einer MicroSD-Karte werden die gewünschten Audiodateien gespeichert. Diese können dann vom SD-Karten Modul gelesen und mittels SPI an den ESP32 weitergesendet werden. Die Schnittstelle ist hierbei wichtig, da man für Ton besonders schnelle Übertragungen braucht, die SPI (Serial Peripheral Interface) unterstützt.

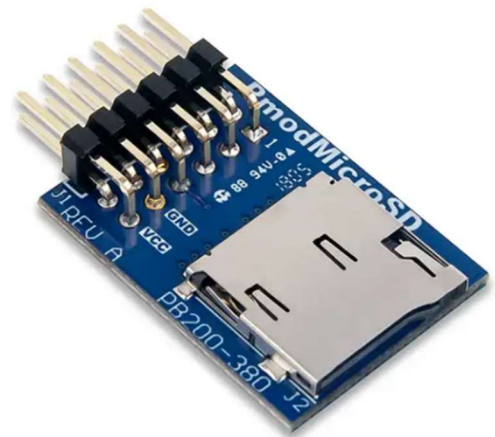


Abbildung 51 SD-Karten Modul [43]

2.4.1.5 Audioverstärker und Lautsprecher

Um die in 2.4.1.4 erwähnten Sounds auszugeben, wird ein MAX98357 Audioverstärker und ein 3 W-8 Ω -Lautsprecher benutzt. Der Audioverstärker kommuniziert mit dem ESP32 mittels I²S-Schnittstelle (Inter-IC Sound). Dieser wandelt dann die digitalen Audio-Daten in sinusförmige für den Menschen hörbare Wellen um und verstärkt diese.

Der Lautsprecher ist an die Plus- und Minus-Kontakte des Audioverstärkers angeschlossen (Siehe Abbildung 52).

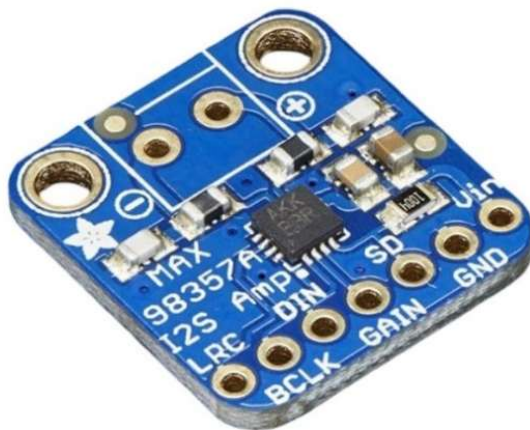


Abbildung 52 Audioverstärker [45]



Abbildung 53 Lautsprecher [44]

2.4.1.6 Logikpegelwandler

Da bei dem Projekt mehrere Logikpegel verwendet werden (ESP32 und Arduino mit 3,3 V und PIC mit 5 V), benötigen wir einen Logikpegelwandler. Hier wurde ein MOSFET-basierter bidirektionaler Wandler benutzt. Von den vier Kanälen auf dem Modul wurden nur zwei tatsächlich verwendet und ein weiterer für ein optionales Ziel vorgesehen.

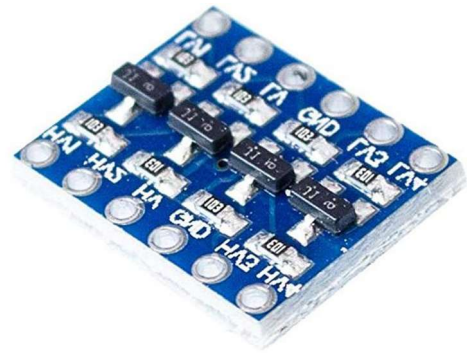


Abbildung 54 Logikpegelwandler [46]

Die Schaltung des Wandlers ist sehr einfach, wie auf Abbildung 55 zu sehen ist. Von einem MOSFET des Typen BSS138 (Siehe 9.7.2) ist das Gate ständig mit +3,3 V verbunden und ist daher normalerweise im leitfähigen Zustand. Ist eine Seite auf logisch 0, so wird über den MOSFET die andere Seite auch auf 0 gezogen. Wird die 3,3 V-Seite auf logisch 1 gezogen, so ist keine Spannung mehr zwischen Gate und Source und der MOSFET hört auf zu leiten. Über den Pull-up Widerstand (R_3) wird die 5 V-Seite auch auf logisch 1 gezogen. Ist auf der 5 V-Seite ein logisch 1, so wird die 3,3 V-Seite über den MOSFET auf logisch 1 gezogen. Die Schaltung kann auch mit anderen Spannungen betrieben werden, sofern sie nicht die Grenzen des MOSFETs übersteigen.

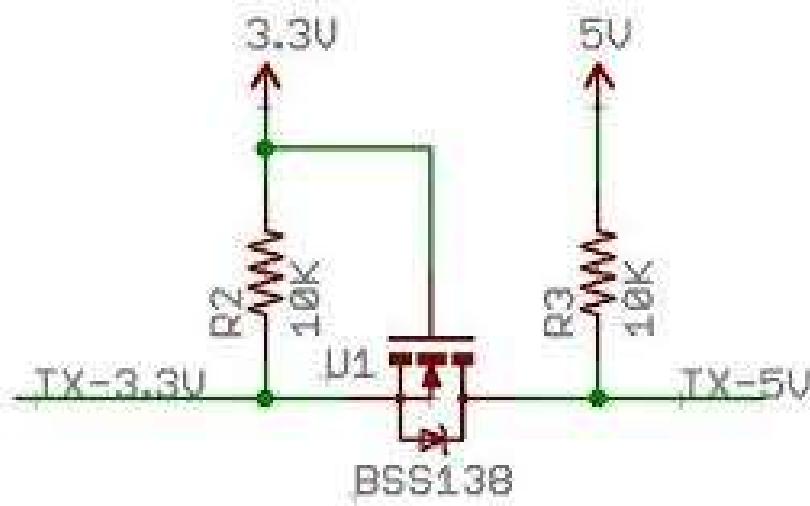


Abbildung 55 Logikpegelwandler Schaltung [12]

2.4.1.7 LEDs

Um das Erreichen des Zieles neben den auditiven Ausgaben auch visuell zu zeigen, wurden LEDs eingeplant. Auf der Platine gibt es Platz für zwei einfarbige (rot und grün zum Beispiel) sowie eine zweifarbige LED. Im endgültigen Produkt wurde nur eine zweifarbige LED verwendet.



2.4.1.8 Mosfet für Vibrationsmotor

Abbildung 56 LEDs [47]

Um die Möglichkeit einer haptischen Rückmeldung zu bieten, wurde ein externer Vibrationsmotor eingeplant, den man in der Hand halten kann. Dieser sollte von einem MOSFET angesteuert werden.

Dieses optionale Ziel wurde letztendlich nicht umgesetzt. Daher ist zwar Platz für die Bauteile auf der Platine vorgesehen, jedoch wurden sie nicht bestückt.

2.4.2 Schaltplan

Der vollständigen Gesamtschaltplan ist im Anhang (Kapitel 9.6) zu finden. Dieser besteht aus zwei Seiten, einer mit allen Bauteilen die mit 5 V versorgt sind und einer bei denen sie mit 3,3 V versorgt sind. Die Spannungsversorgung ist auch auf der ersten Seite zu finden.

2.4.3 Erklärung Platine

Um die Elektronik möglichst unauffällig in der Matte zu verbauen und die Stolpergefahr für die Benutzer zu reduzieren, wurde eine ungewöhnliche Form für die Platine gewählt. Wie in Abbildung 57 gezeigt, wurde die Platine so schmal wie möglich gestaltet, weshalb sie auch sehr lang ist. Die Längen der Seiten sind 533 x 31,8 mm, was circa ein Seitenverhältnis von 17:1 ergibt. Mit dieser Form kann man die Platine entlang einer der kürzeren Seiten der Matte platzieren und so das Gehäuse auch schmal und lang machen.



Abbildung 57 Platinendesign gesamt

In den folgenden Unterkapiteln sind alle mit Ground verbundenen Anschlüsse nicht verbunden, da die GND-Fläche zu Zwecken der Übersichtlichkeit entfernt wurde.

2.4.3.1 Stromversorgung

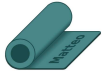
An Abbildung 58 und Abbildung 59 ist die Stromversorgung zu sehen, wie sie in Kapitel 2.2 erklärt ist. Auf der linken Seite ist das USB-C-Akku-Lademodul. Rechts daneben sind die zwei Akkus verbaut. Auf der rechten Seite befinden sich der Step-up- und Step-down-Converter.



Abbildung 58 Platinendesign Stromversorgung



Abbildung 59 Platine Stromversorgung



2.4.3.2 Messschaltung und Extras

Auf Abbildung 61 und Abbildung 60 ist die Messschaltung (Kapitel 2.3), der Platz für den MOSFET, der Platz für den Vibrationsmotor und der Logikpegelwandler zu sehen.

Die Messschaltung ist ähnlich zur Beschreibung in Kapitel 2.3, jedoch wurden ein paar Kleinigkeiten geändert.

- Die Sensoren sind in einer anderen Reihenfolge an den MUX angeschlossen als auf dem Prototyp. Die Anordnung dort hat zu Verwirrungen unter den Teammitgliedern geführt, weshalb hier die Kanäle von rechts nach links geordnet sind (0 ganz rechts, 11 ganz links).
- Die 12 separaten Ground-Anschlüsse wurden auf einen gemeinsamen Anschluss zusammengefasst.
- Die Pin-Leisten zum Anschluss der Sensoren wurden durch Lötflächen ersetzt. Wie an Abbildung 61 ersichtlich, sind die Leitungen der Sensoren an die Lötflächen angelötet.
- Im Vergleich zur Messplatine ist hier für den Spannungsteiler statt einem Potentiometer ein 4,7 k Ω Widerstand verwendet wurde. Als zweiter Widerstand wirkt der interne Widerstand des Sensors. Wie in Kapitel 2.3.2.1 beschrieben, wurde ermittelt, dass dieser Widerstandswert am besten funktionieren würde, daher ist auch die Einstellbarkeit des Potentiometers nicht mehr nötig.

Neben dem PIC ist ein Stecker für dessen Programmierschnittstelle. Um die gesendeten Daten vom PIC auch am Computer lesen zu können, wurde ein Stecker mit der Versorgungsspannung und der UART-Schnittstelle eingebaut. Im späteren Verlauf des Projekts wurden die GND und 5 V Anschlüsse verwendet, um die in Kapitel 2.5 beschriebene Platine zur Anzeige des Akkuladestandes mit Spannung zu versorgen.

In der rechten oberen Ecke sieht man ein MOSFET, eine Diode und einen Anschluss. Dieser war für den Vibrationsmotor gedacht, jedoch wurde dieses Ziel nicht erfüllt. Daher sind außer der Diode die Bauteile auch nicht bestückt.

Rechts unten in Abbildung 61 kann man den Logikpegelwandler sehen. Bei den Tests sind beim Schalten ungewöhnliche Spannungsspitzen aufgetreten, deshalb wurde sicherheitshalber Platz für einen Widerstand und einen Kondensator zum Glätten des Signales vorgesehen (rechts daneben). Glücklicherweise lag das Problem wahrscheinlich in den vielen Kabeln beim Versuchsaufbau, die Störungen eingefangen haben. Daher wurde der Kondensator nicht benötigt und für den Widerstand wurde eine Drahtbrücke eingebaut. Links neben dem Wandler sind drei Anschlüsse, die für ein LED-Streifen gedacht waren, jedoch wurden diese nicht verwendet.

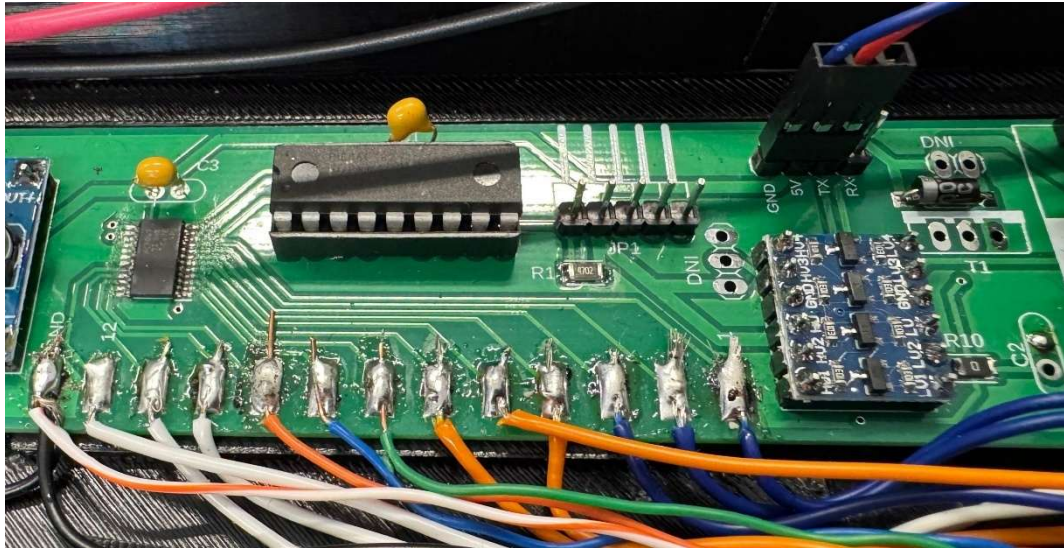


Abbildung 61 Platine Messschaltung

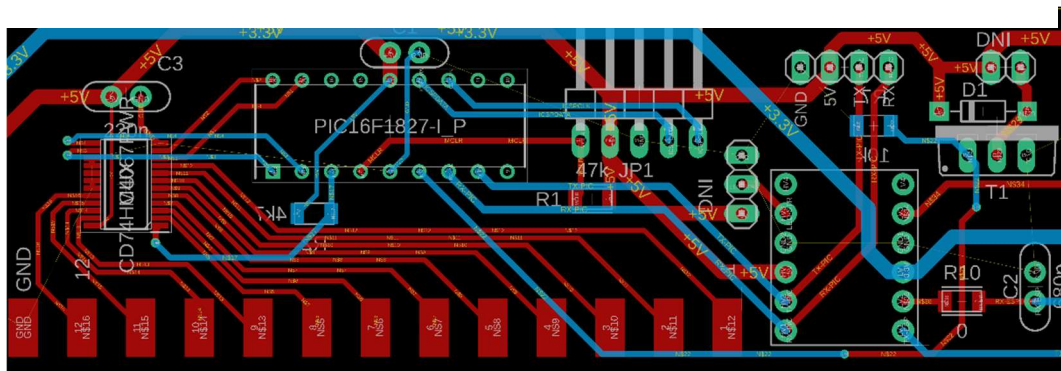


Abbildung 60 Platinendesign Messschaltung

2.4.3.3 Sprachsteuerung und -ausgabe

Am Ende der Platine sind die Sprachsteuerung und Sprachausgabe auf der Platine untergebracht. Hier sind die meisten Leiterbahnen für die verschiedenen Kommunikationsschnittstellen vorgesehen (SPI, I²C und I²S). Auf der linken Seite sieht man den ESP32. Rechts daneben sind die Anschlüsse für die diversen LEDs. Im endgültigen Entwurf wurden nur die Anschlüsse für die zweifarbige Led verwendet. Daneben ist der Audioverstärker, der mit einem kleinen 3D-gedrucktem Zylinder stabilisiert wird. Rechts davon ist der Arduino Nicla Voice und ganz am Ende ist das microSD-Karten-Modul, das auch von zwei Zylindern stabil gehalten wird.

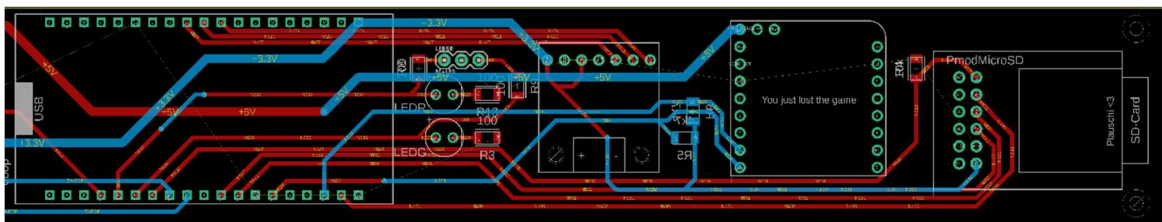


Abbildung 63 Platinendesign Sprachsteuerung

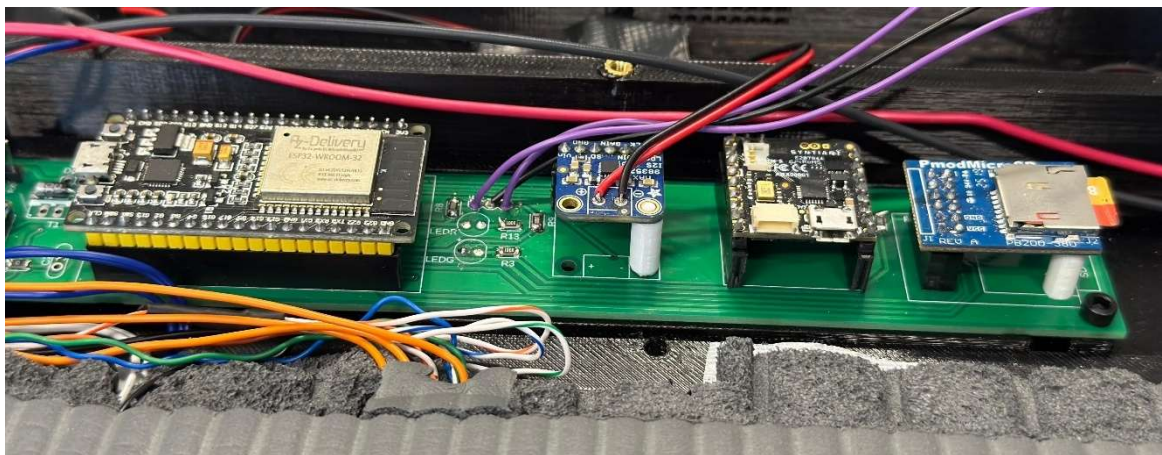


Abbildung 62 Platine Sprachsteuerung

2.4.3.4 Externe Bauelemente

Manche Bauteile mussten aus funktionalen Gründen an bestimmten Stellen im Gehäuse platziert werden und konnten daher nicht direkt an die Platine gelötet werden.

- Der Lautsprecher ist an den Audioverstärker angeschlossen, ist aber im Deckel verschraubt.
- Die LED zur Anzeige des erreichten Zieles (Abbildung 65) ist ebenfalls im Deckel neben dem Lautsprecher eingebaut. Dieser wurde mit Kabeln an die Platine verbunden.
- Die Akkustandsanzeige ist ebenfalls im Deckel verbaut. Drei Leitungen, jeweils eine für Ground (Blau), +5 V (Rot) und Akkuspannung (Braun), verbinden sie mit der Platine
- Der Schalter ist ebenfalls mit zwei Leitungen mit der Platine verbunden. Er ist am anderen Ende des Gehäuses verbaut.

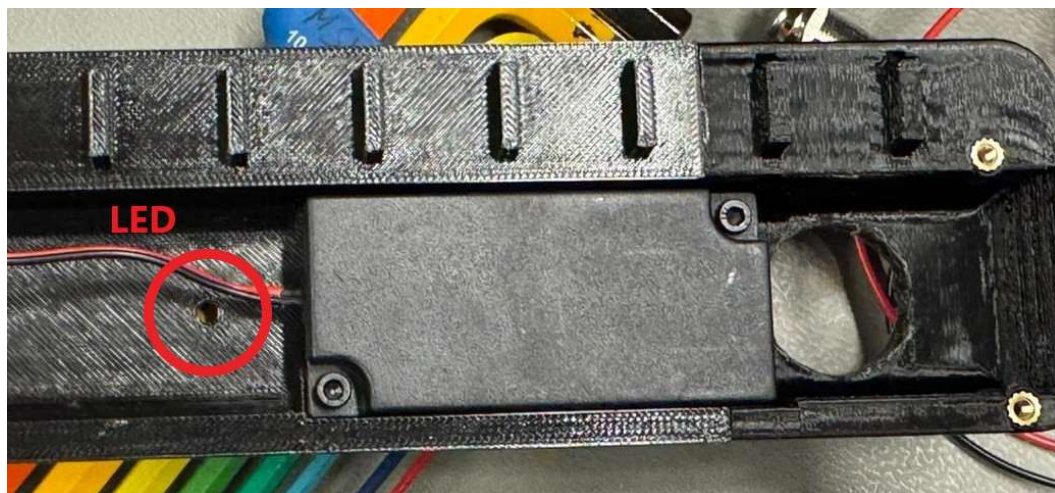


Abbildung 65 Lautsprecher verbaut



Abbildung 64 Akkustandsanzeige verbaut

2.4.4 Nachträgliche Verbesserungen

Auf der Platine musste nachträglich noch eine Sache verbessert werden. Die grüne LED war mit einem Ausgang des ESPs verbunden, der vom ESP beim Starten verwendet wird, wodurch dieser nicht mehr starten konnte. Also Lösung wurde die Leiterbahn aufgetrennt und mit Lackdraht mit einem anderen Ausgang des ESP verbunden.

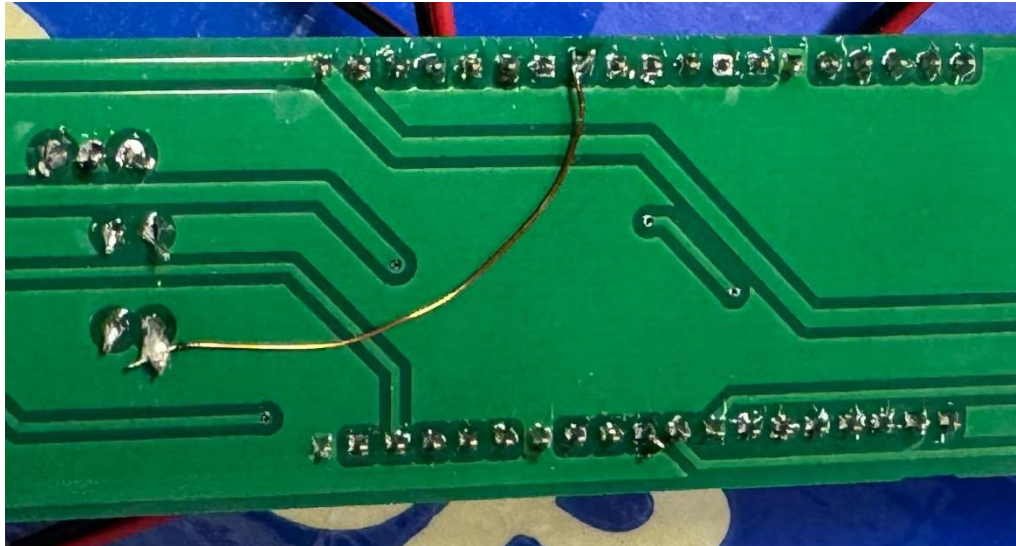
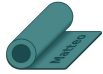


Abbildung 66 Platine nachträgliche Verbesserung

2.4.5 Bild gesamte Platine



Abbildung 67 Gesamte Platine



2.4.6 Vorversionen

Es wurden drei Varianten von der Platine bestellt, wobei die zwei ersten bis auf eine vergessene Leiterbahn deckungsgleich waren und nur bestellt wurden, weil die erste von dem österreichischen Zoll aufgehalten wurde.

Die Änderungen auf der endgültigen Platine waren wie gefolgt:

- Einige Durchgangslöcher wurden zu klein gemacht, diese wurden ausgebessert.
- Der Arduino Nicla Voice wurde mit 3,3 V anstatt mit 5 V versorgt. Da der Microprozessor des Arduino mit 3,3 V arbeitet, wurde fälschlicherweise angenommen das an den Versorgungsanschluss die 3,3 V Leiterbahn angeschlossen gehört.
- Die Step-Up und Step-down Converter waren vorher an derselben Stelle, jedoch einer auf der Unterseite und der andere auf der Oberseite der Platine. Da die Länge der Platine genügend Platz bietet ist und die vertikale Höhe beschränkt ist, wurden sie beide auf die Oberseite nebeneinander angeordnet.

2.5 Akkustand Anzeige

Laut Hauptziel H1c muss eine LED den Akkuladestand anzeigen. Um das zu realisieren, wurde eine Schaltung gebaut die anhand der Spannung des Akkus eine bestimmte Farbe anzeigt:

Akku-V	%	Farbe
~3 - 3.4 V	0 - 20%	Rot
3.4 - 3.8 V	20 - 80%	Gelb
3.8 - 4.2 V	80 - 100%	Grün

Tabelle 10: Akkustand Anzeige



Abbildung 68 Akkustand Anzeige Diagramm

2.5.1 Verwendete Bauteile

2.5.1.1 Rot-Grün-LED

Um die Farben zeigen zu können, wurde eine Rot-Grün-LED verwendet. Wie schon der Name sagt, hat dieser eine rote und eine grüne LED verbaut. Damit kann man alle Farbe von #XXXX00 zeigen, was genau dem benötigten Bereich entspricht (Rot= #FF0000, Gelb=FFFF00, Grün=00FF00). Die LED hat einen gemeinsamen Ground-Anschluss (Mitte) und jeweils einen Anschluss für rot und grün.



Abbildung 69
Rot-Grün-LED
[49]

2.5.1.2 Operationsverstärker

Der MCP6002 (Siehe 9.7.3) ist ein Low-Power 1 MHz Rail-to-Rail OPV (Operationsverstärker) von Microchip. Verbaut sind zwei dieser OPVs in einem 8-Pin PDIP Gehäuse. Dieser wurde eingebaut, damit ADC des Microcontrollers von der Schaltung der Sensoren nicht belastet wird und so die Messung genau bleibt.

Die OPVs wurden als Komparatoren verwendet. Wie in Abbildung 71 ersichtlich, vergleichen die OPVs zwei Spannungen und geben abhängig von der Differenz entweder die positive oder die negative Versorgungsspannung aus.



Abbildung 70 OPV [10]

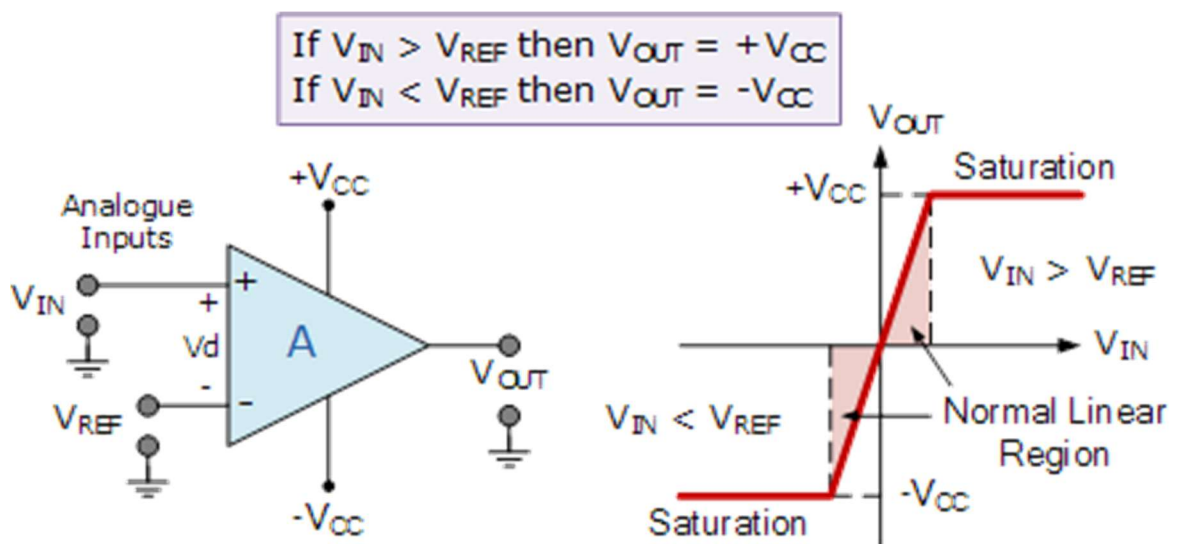


Abbildung 71 Komparator Erklärung [48]

2.5.2 Schaltplan

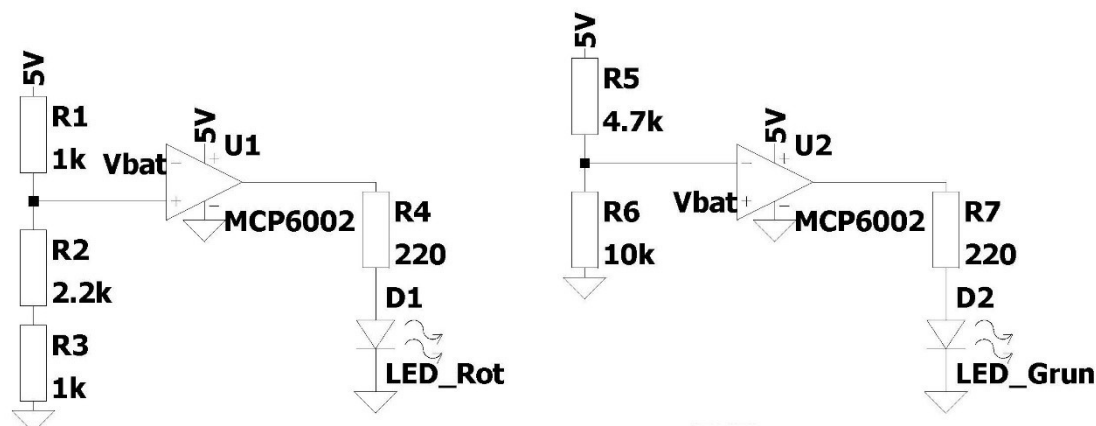


Abbildung 72 Schaltplan Akkustand Anzeige

Zum Vergleichen der Akku-Spannung braucht man Referenzspannungen, hier 3,4 V und 3 V. Diese werden über einfache Spannungsteiler erzeugt.

In der linken Schaltung wird der Spannungsteiler aus 1 k Ω und (zusammengefügt) 3,2 k Ω gebildet, womit eine rechnerische Referenzspannung von 3,81 V zur Verfügung steht. Wird die Akkuspannung größer als dieser, so schaltet der OPV auf Ground → Rot ist aus → Die LED leuchtet grün.

In der rechten Schaltung besteht dieser aus 4,7 k Ω und 10 k Ω , womit eine Spannung von 3,4 V zur Verfügung steht. Sinkt die Akkuspannung unter diesen Wert, schaltet der OPV auf Ground → Grün ist aus → Die LED leuchtet rot.

In der Zone zwischen 3,4 V und 3,8 V leuchten beide Dioden und erzeugen somit ein gemischtes Gelb.

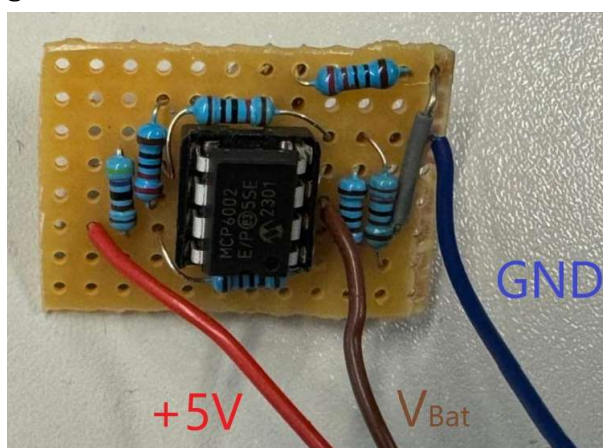


Abbildung 73 Platine Akkustand Anzeige oben

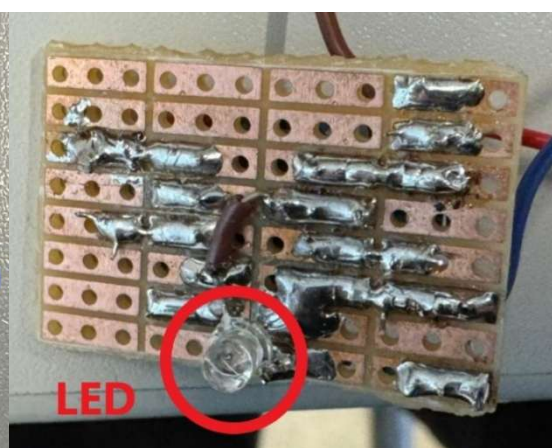


Abbildung 74 Platine Akkustand Anzeige unten

3 Konstruktion

Die Konstruktion kann grob in drei Teile eingeteilt werden:

- **Gehäuse:** Das Gehäuse schützt die Platine vor Schäden. Es ist so schmal und niedrig gemacht, wie es technisch umsetzbar ist, um die Stolpergefahr zu reduzieren und der Matte eine gute Ästhetik zu geben. Das Gehäuse ist an einer der kurzen Seiten der Matte montiert.
- **Matte:** Die Matte ist der größte, aber auch einfachsten Teil der Konstruktion. Die Matte muss per Definition mindestens 900 x 600 mm groß sein und die Sensoren beinhalten.
- **Platten zur Messung:** Die Platten sind die Bauteile, auf denen die Belastung aufliegt. Diese müssen groß genug sein, um genügend Spielraum zur Positionierung der Füße zu bieten, aber nicht so groß, dass die Platten aufgrund von Durchbiegung den Boden berühren und somit die Kraft nicht auf die Sensoren drücken.

In Abbildung 75 ist ein Render des 3D-Modells der gesamten Konstruktion zu sehen. Alle Modelle bis auf das 3D-Modell der Platine (Siehe Kapitel 3.4) und alle Renderings wurden in Creo Parametric erstellt.



Abbildung 75 Gesamte Matte Render

3.1 Gehäuse

Das Gehäuse dient zum Schutz der Elektronik und stellt sicher, dass sich die Matte und die Platine nicht voneinander trennen können. Sie besteht aus zwei Teilen, dem Boden und dem Deckel, die mit Schrauben zusammen verbunden sind. Das Gehäuse hat Außenmaße von 700 x 60 x 38 mm.

Das gesamte Gehäuse ist 3D-gedruckt, da dies die einfachste, schnellste und kostengünstigste Art der Herstellung ist. Das gewählte Filament war schwarzes PETG, da dieses sehr einfach und ungefährlich zu Drucken ist, aber trotzdem die notwendige Robustheit bietet. Für das Gehäuse wurden schätzungsweise 300 g Filament verbraucht, was einem Preis von ca. €7,50 entspricht.



Abbildung 76 Gesamtes Gehäuse Render

3.1.1 Boden

Der Boden dient als Unterlage für die Platine, sodass sie stabil sitzt und nicht beschädigt wird. Um die Platine auf dem Boden zu montieren, sind sechs Löcher auf diesem vorgesehen, die als Durchgangslöcher für M3-Schrauben fungieren. Diese werden mit Einschmelzgewinden verschraubt, wie beschrieben in 3.1.1.1. Als Schrauben wurden M3 x 6 verwendet (DIN 7984). Damit die Platine sich nicht durchbiegt, wurden an allen in Frage kommenden Stellen zusätzliche Auflagen konstruiert. Diese Flächen sind in Abbildung 78 rot markiert.



Abbildung 77 Boden + Platine Render



Abbildung 78 Boden Auflageflächen Render

In den Wänden des Bodens sind acht Löcher, die als Durchgangs- und Senklöcher für die Schrauben verwendet werden. Mit diesen kann von der Unterseite der Boden an den Deckel des Gehäuses geschraubt werden.



Abbildung 79 Boden Durchgangslöcher Render

Auf der rechten Seite des Bodens ist eine Halterung, in der der Schalter montiert wurde. Dieser wurde dann über zwei Leitungen mit der Platine verbunden und dort an den für ihn vorgesehenen Lötstellen festgelötet.

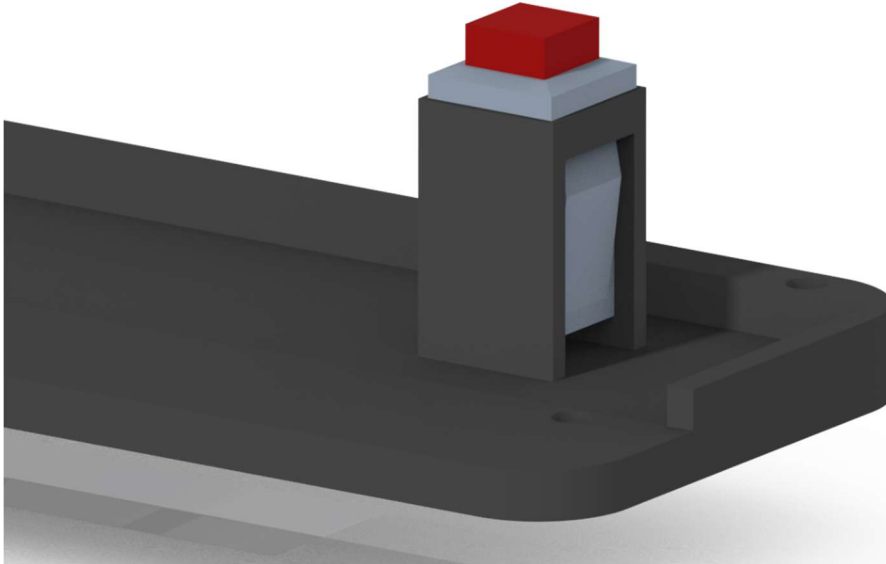


Abbildung 80 Schalter eingebaut Render

Auf der Vorderseite des Bodens ist eine 21 mm lange Lasche. Um die Matte fest mit dem Gehäuse zu verbinden, wird sie zwischen Deckel und Boden des Gehäuses eingeklemmt. Diese Lasche ist dafür die Unterseite.

3.1.1.1 Einschmelzgewinde

Für Gewinde zur Befestigung der Platine gibt es mehrere Lösungen.

Man könnte in den 3D-Druck kleine Löcher mit dem Kerndurchmesser der Schrauben vorsehen und dann spitze Schrauben, wie zum Beispiel Holzschrauben, verwenden, um manuell bei jedem Einschrauben ein neues Gewinde in das Material zu schneiden. Diese Lösung ist der Industriestandard, da die Kosten dafür sehr sind und die Umsetzung vergleichsweise einfach ist. Die Nachteile hier sind aber, dass die nur eine limitierte Anzahl von neuen Gewinden im Material möglich ist, bevor das Material versagt. Dadurch wäre auch limitiert, wie oft das Gehäuse geöffnet werden kann. Bei einem Produkt für Konsumenten ist das kein Problem, weil diese nicht zur Technik gelangen müssen. Bei einem Prototyp wie diesem muss das Gehäuse jedoch recht oft geöffnet werden, um Veränderungen und Verbesserungen vornehmen zu können, weswegen diese Option nicht geeignet ist.

Aus diesem Grund ist die Wahl auf Einschmelzgewinde gefallen. Wie in Abbildung 81 ersichtlich, handelt es sich hier um kleine Bauteile aus Metall, meist aus Messing, die innen ein Gewinde und außen Rillen mit verschiedenen Richtungen haben. Diesen können bis zum Schmelzpunkt des Plastiks erhitzt und dann in das Plastik gedrückt werden. Hierbei schmilzt das Plastik in die Rillen ein und wenn es ausgehärtet ist, hält es die Gewinde fest im Bauteil. Da die Einschmelzgewinde aus Metall sind, gibt es keine Limitierung bei der Anzahl von An- und Abschrauben.



Abbildung 81 Einschmelzgewinde [50]

Auf Abbildung 82 sind zwei Einschmelzgewinde in dem Deckel zu sehen.



Abbildung 82 Einschmelzgewinde eingeschmolzen



Abbildung 83 Gesamt Boden Render

3.1.2 Deckel

Der Deckel bildet die obere Seite des Gehäuses. Nicht nur bietet er Schutz für die Oberseite der Platine, sondern dient auch als Befestigung für mehrere Bauteile. Im Vergleich zum Boden gibt es hier mehrere Details, die zum Laden der Akkus und für die Sprachsteuerung und -ausgabe benötigt werden.



Abbildung 84 Deckel Gesamt Render

Auf der linken Seite der Platine befindet sich das USB-C-Akkulademodul, das von außen über ein USB-C Ladekabel mit einem 5 V-Netzgerät verbunden werden muss. Dafür wurde in den Deckel eine Ausnehmung konstruiert, die der Form eines USB-C-Steckers entspricht. Um das Anstecken für sehbeeinträchtigte Personen zu vereinfachen, wurde das diese Ausnehmung etwas größer gestaltet, als benötigt gewesen wäre, und zusätzlich mit großen 30° Fasen versehen, die den Stecker zur USB-C-Buchse des Lademoduls führen sollen.

Ebenso kann man auf Abbildung 85 erkennen, dass die gesamte obere Seite des Gehäuses mit einer großen Fase versehen wurde. Ursprünglich war dieser eine Rundung, die aber bei der Herstellung im 3D-Drucker zu Problemen geführt hätte.

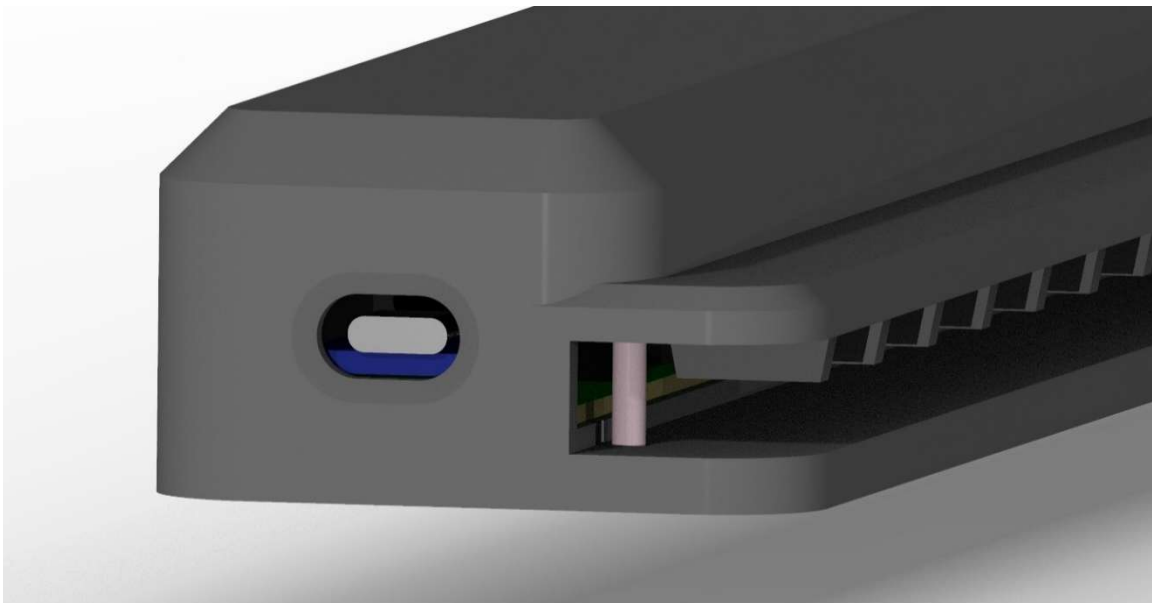


Abbildung 85 USB-C Detail Render

Wie schon in 3.1.1 erwähnt, wird zum Fixieren die Matte zwischen Boden und Deckel des Gehäuses geklemmt. Um sicherzustellen, dass sie auch eingeklemmt bleibt und nicht rausfallen kann, wurde die Unterseite der Lasche zum Klemmen mit vielen Zacken versehen. Diese dringen beim Klemmen in die Matte ein und geben ihr somit mehr Halt. In Abbildung 86 sind die Zacken rot markiert

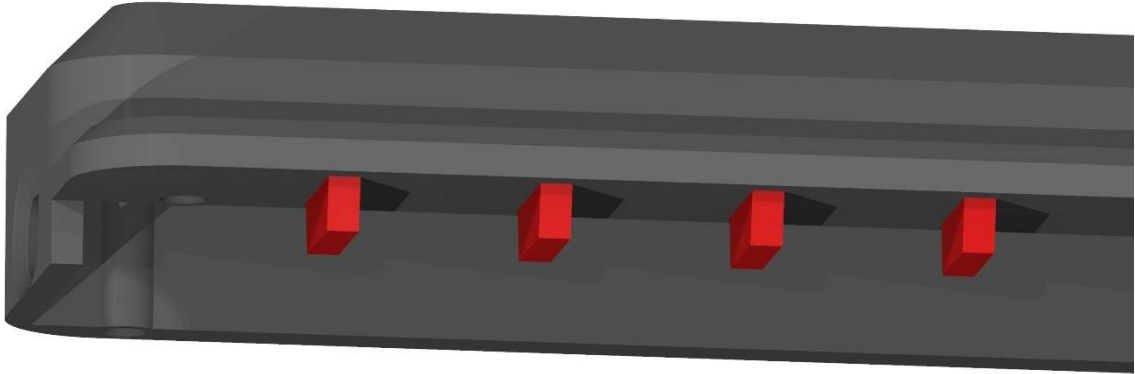


Abbildung 86 Zacken Detail Render

Auch auf der linken Seite befindet sich auch die LED zur Akkustandanzeige, wie in Kapitel 2.5 beschrieben. Da diese Änderung nachträglich zum Drucken erfolgte, wurde hierfür ein Loch für die LED in den Deckel gebohrt und die in Kapitel 2.5.22.5 beschriebene Platine mit Heißkleber in den Deckel geklebt.

Auf der linken Seite des Deckels befinden sich die Bauteile für die Sprachsteuerung und -ausgabe, sowie der Schalter zum Ein- und Ausschalten und die LED zum Anzeigen des aktuellen Übungszustandes.

In Abbildung 87 ist die Druckflächenerweiterung des Druckschalters zu sehen, der ganz rechts im Deckel verbaut ist. Diese wurde aus weißem Filament gedruckt, um für Personen, die nur teilweise sehbeeinträchtigt sind, einen visuellen Anhaltspunkt beim Einschalten der Matte zu bieten.

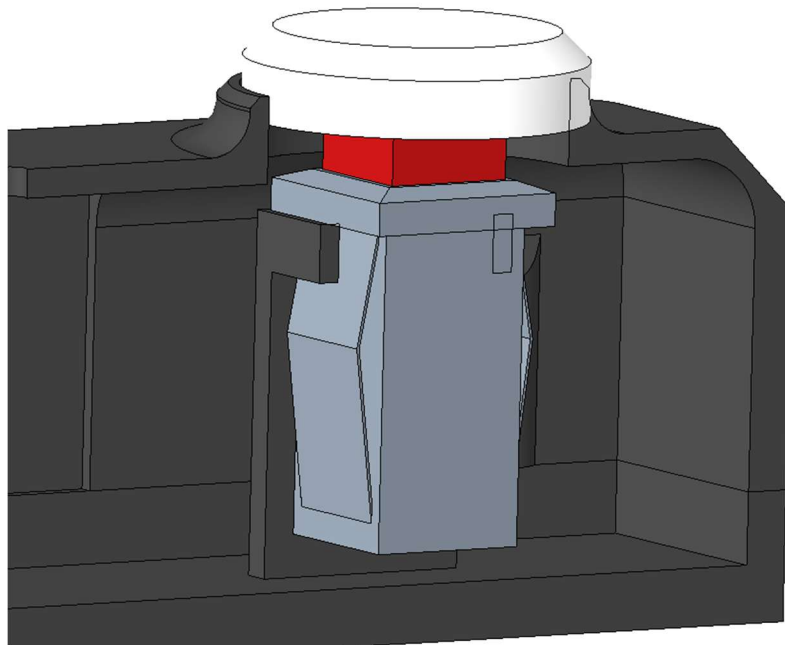


Abbildung 87 Schalter Detail Render

Links neben dem Schalter befinden sich in einem Raster orientierte Löcher. Sie werden benötigt, um eine klare Audioausgabe vom darunter verbauten Lautsprecher zu gewährleisten. Der Lautsprecher ist mit zwei Schrauben montiert (Abbildung 88).



Abbildung 88 Lautsprecher eingebaut

Im einzelnen Loch links neben dem Lautsprecher befindet sich die zweifarbige LED, die den aktuellen Übungsstatus anzeigt. Wie in 2.4.3.4 erwähnt ist diese über Leitungen an die Platine angeschlossen.

Auf der linken Seite von Abbildung 89 sieht man Löcher in der Seite des Deckels, die als Einlass für die Sprachsteuerung dienen.

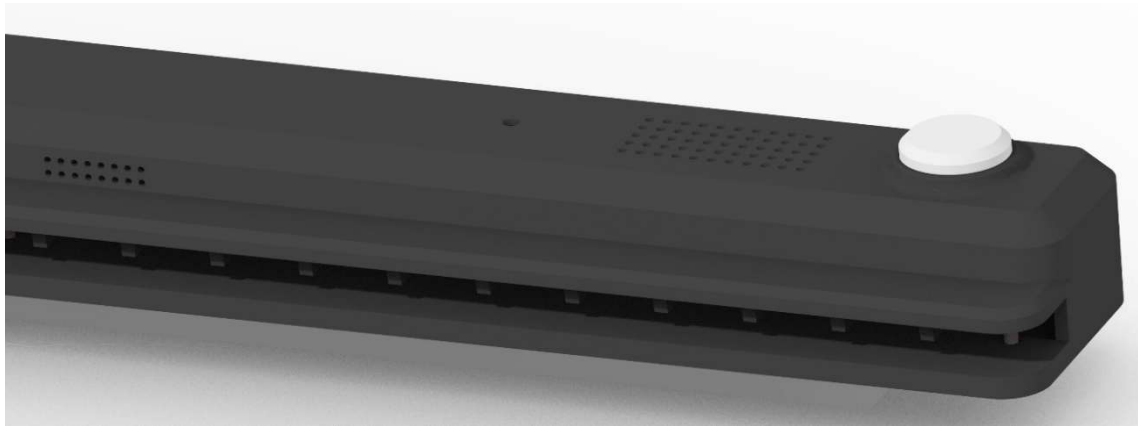


Abbildung 89 Details Gehäuse rechts Render

3.1.3 Bild Gehäuse

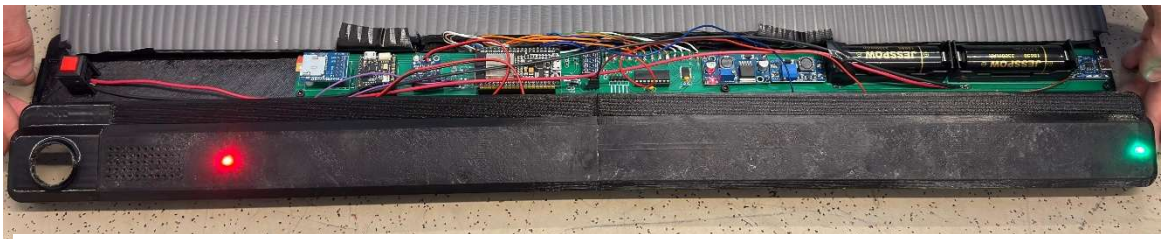


Abbildung 90 Gehäuse mit Platine

3.1.4 Vorversionen

Während der Testphase wurde eine Version des Gehäuses gedruckt, bei der das Einklemmen der Matte nicht möglich war, weil der Abstand zu klein modelliert. Statt einer Mattendicke von 1,5 **cm** wurde eine Dicke von 1,5 **mm** eingeplant.



Abbildung 91 Gehäuse Vorversion

3.2 Matte

Die Matte wurde aus einer Gymnastikmatte ausgeschnitten, wie sie in Abbildung 92 zu sehen ist. Mithilfe eines Skalpells wurde die äußere Kontur ausgeschnitten, die in der Werkstättenzeichnung der Matte im Anhang (Kapitel 9.5) zu sehen ist. Um die Platten zur Verteilung der Belastung (Kapitel 3.3) in der Matte zu versenken, wurden die trapezförmigen Formen mit dem Skalpell zur Hälfte der Matte eingeschnitten und der Stoff entfernt. Als nächstes wurden die Platten mit Kleber in die Ausschnitte geklebt.



Abbildung 92 Gymnastikmatte [33]

Die Leitungen, die von den Sensoren zur Platine führen, mussten auch in die Matte eingebunden werden. Dazu wurden zwei Kanäle in die Matte geschnitten, in denen jeweils die Leitungen von zwei Sensorgruppen zur Matte geführt wurden. Nachdem die Leitungen verlegt waren, wurden die Kanäle mit Klebeband verschlossen.

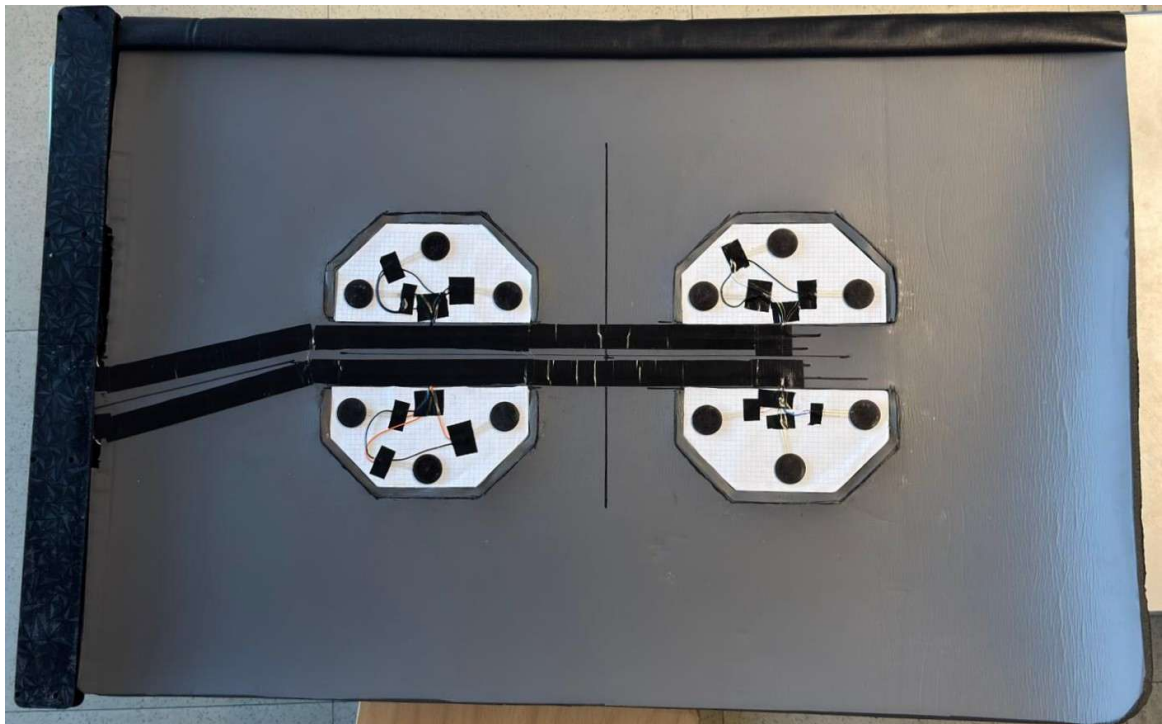
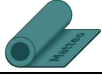


Abbildung 93 Matte von unten offen



Um die Sensorik und die Platten vor Schäden zu schützen, wurde eine Teichfolie auf die untere Seite der Matte geklebt. Im Moment ist nur eine Seite festgeklebt, sodass die Sensorik für etwaige Reparaturen erreichbar ist.



Abbildung 94 Matte gesamt

3.3 Platten zur Messung

Um die zu messende Belastung auf mehrere Sensoren aufzuteilen, wurden Aluminiumplatten hergestellt. Jede der Platten hat drei Drucksensoren auf der Unterseite befestigt. In Abbildung 95 ist zu sehen, dass die Sensoren möglichst nahe am Rand der Trapezform angebracht wurden. So hat die Platte unabhängig von der Positionierung der Füße immer drei stabile Auflagepunkte, gekennzeichnet durch die Kreuze.

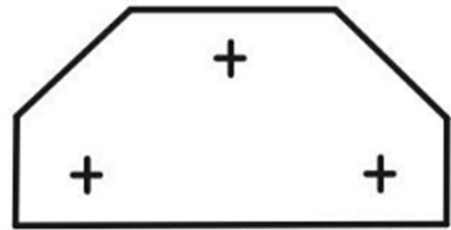


Abbildung 95 Platte Konzept

Die Platen wurden aus 3 mm dicken Aluminiumblechen hergestellt. Dieses Material bietet bei der verwendeten Materialstärke ein ausreichend hohes Widerstandsmoment, sodass die volle Gewichtskraft einer Person auf einer einzigen Platte ohne Durchbiegung aufgenommen werden kann. So können keine Messfehler dadurch entstehen, dass bei zu starker Durchbiegung Kontakt mit dem Boden entsteht und ein Teil der Belastung auf den Untergrund und nicht auf die Sensoren wirkt. Außerdem konnte die Matte durch die dünne Materialstärke möglichst dünn gehalten werden.

Weiters wurde eine Papierschicht zwischen den Sensoren und den Platten als Isolation integriert, da die Sensoren andernfalls über die Aluminiumplatten kurzgeschlossen worden wären.

Die Maße der Platten sind im Anhang (Kapitel 9.5) auffindbar.

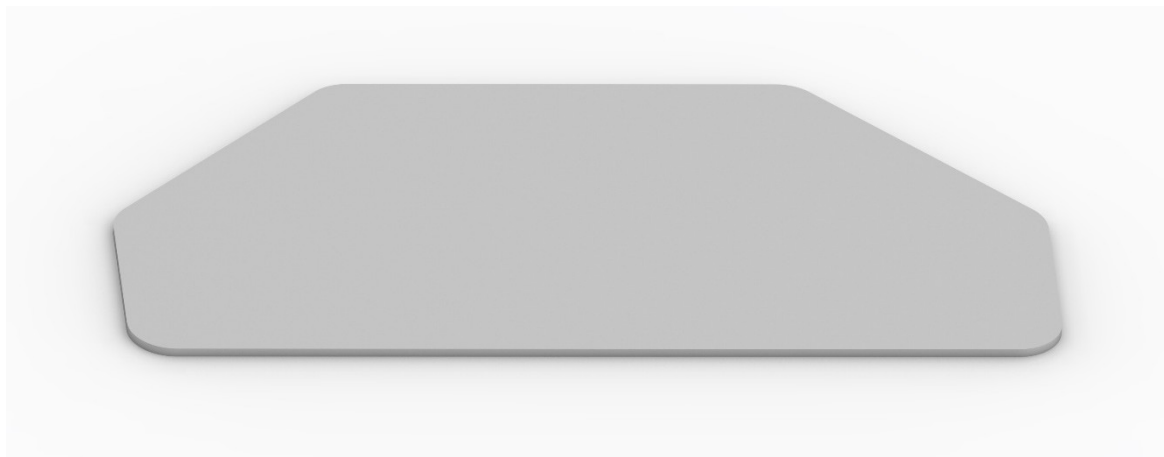
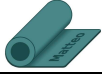


Abbildung 96 Platte Render



3.3.1 Material

Mehrere Materialien standen im Laufe des Projektes zur Diskussion. Das Ziel war es, ein Material zu finden, mit dem die gesamte Dicke (Dicke der Platten, Sensoren und Unterlagen der Sensoren [Kapitel 3.3.2]) zu minimieren. Hierfür spielt die Biegefestigkeit eine große Rolle. In Betracht gezogen wurden:

- **Modellbauholz mit Glasfaser:** Laut Empfehlung eines Betreuers wäre dies eine sehr stabile Option, die auch bei dünnem Material sehr biegesteif sei. Es sollten zwei Modellbauholz-Platten mit Epoxidharz verbunden und eine dünne Glasfaserschicht dazwischen gelegt werden. Angeblich bietet diese Kombination einen besonders starken Widerstand gegen Biegungen. Aufgrund der Komplexität der Herstellung dieser Platten und des fehlenden Knowhows wurde jedoch entschieden, eine andere Option zu wählen.
- **Stahl:** Stahl ist eine naheliegende Option, wenn es zu Biegefestigkeit kommt. Das Problem liegt hier bei der Dichte des Materials. Stahl ist ein eher schweres Material, wodurch die Sensoren bereits mehr belastet wären und daher die maximale Nutzungskraft verringert wird.
- **Aluminium:** Gewählt wurde Aluminium. Es ist einfach zu bearbeiten, biegesteif und im Vergleich zu Stahl sehr leicht. Durch die große Verfügbarkeit des Materials ist es kostengünstig und einfach zu besorgen. Von den vielen Verschiedenen Legierungen ist die Wahl auf ENAW-6061 T6 gefallen. Diese gehört zu den stärkeren Aluminium-Varianten.

3.3.1.1 Creo Simulation

Es wurde eine Simulation mit Creo über die Durchbiegung der Platten durchgeführt. Die Platten wurden dabei mit 80 kg belastet. Das Ergebnis dieser Simulation ist hier zu sehen:

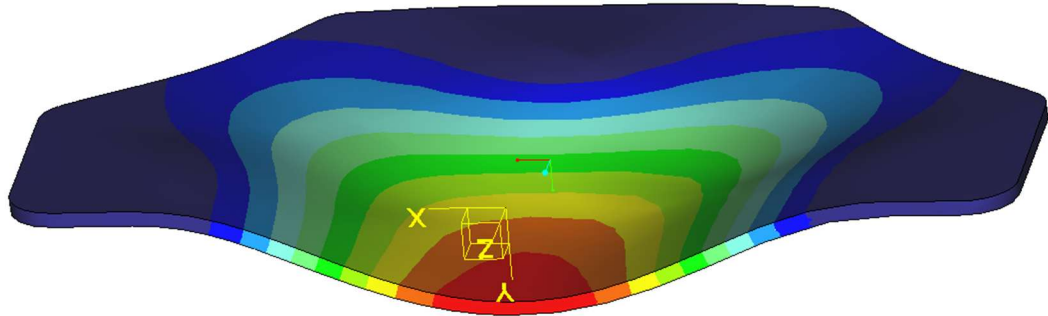


Abbildung 97 Platte Creo Simulation

Die Simulation zeigt, dass die maximale Durchbiegung rund 0,25 mm beträgt. Die Achsenbeschriftung konnte nicht vergrößert werden, da die Schriftgröße von Creo vorgegeben ist.

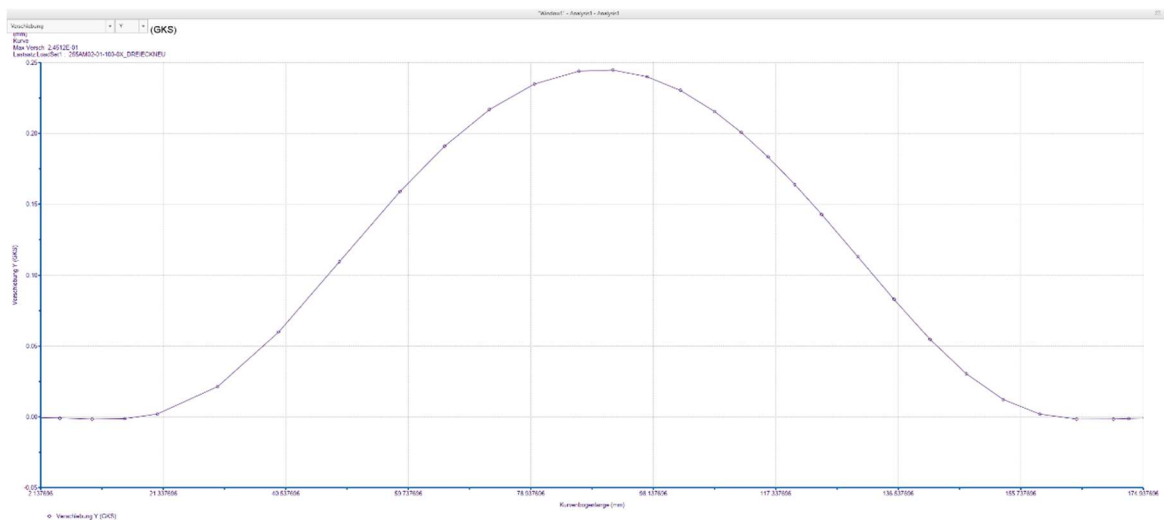


Abbildung 98 Platte Creo Simulation Graph

3.3.2 Unterlagen

Damit die gesamte Kraft an den Sensoren aufliegt, müssen Sensoren und Platte vom Boden distanziert werden. Hierfür wurde eine Unterlage konstruiert und für jeden Sensor jeweils eine 3D-gedruckt. Die Unterlage ist ein flacher Zylinder mit einem Durchmesser von 20 mm und einer Höhe von 2 mm. Mit diesen Maßen konnte sichergestellt werden, dass die gesamte Kraft nur auf den messenden Flächen der Sensoren aufliegt. Die Höhe bietet genügend Raum, um die leichte Durchbiegung der Platten zu erlauben, ohne dass diese den Boden berühren und somit die Messung verfälschen. Auf der Unterseite der Unterlage ist außerdem eine dünne Schicht mit größerem Durchmesser ($\varnothing 30 \times 1 \text{ mm}$) um den Unterlagen Stabilität zu geben.

Im Zusammenbau wurden die Sensoren mit einer Seite auf die Unterlage und mit der anderen Seite auf die Papierschicht auf den Platten geklebt.

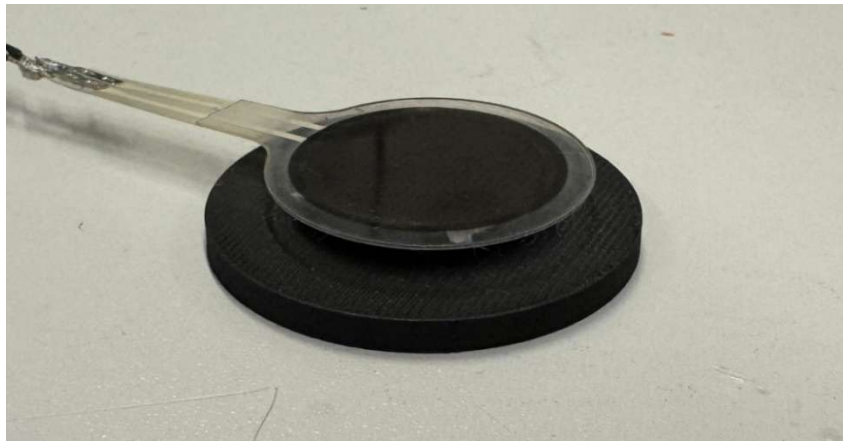


Abbildung 99 Unterlage mit Sensor

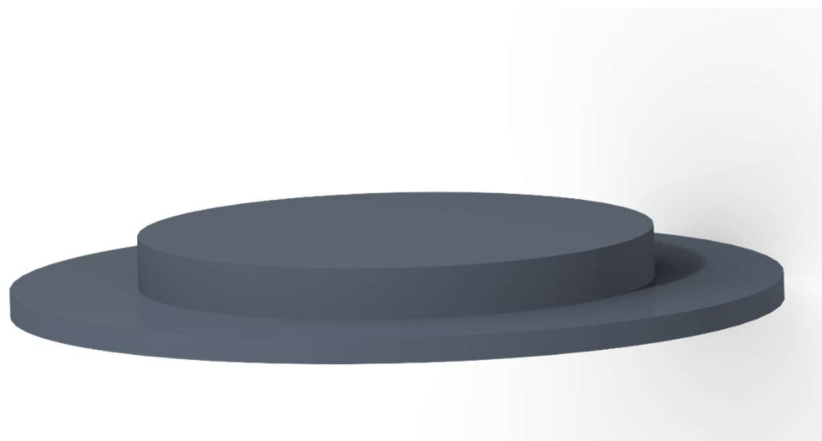


Abbildung 100 Unterlage Render

3.3.3 Vorversion

Eine vorherige Version der Platten hatte eine andere Form. Es wurde eine einfachere dreieckige Form gewählt. Um die benutzbare Fläche zu erhöhen wurde, die Entscheidung getroffen, die oben gezeigte Trapezform zu wählen. Ebenso wurde die Länge der Platten erhöht.



Abbildung 101 Platte Vorversion

3.4 Platine

Die Platine wurde so erstellt, dass gleichzeitig ein 3D-Modell generiert werden konnte. Mit der „3D-PCB“-Funktion von Fusion 360⁵ konnten 3D-Modelle zu den einzelnen elektronischen Bauteilen hinzugefügt werden, woraufhin das Programm sie automatisch auf die 3D-Platine platziert. In den folgenden Abbildungen kann man Bilder von der Platine sehen.

Die 3D-Modelle wurden zum Teil selber konstruiert und zum Teil vom Hersteller der Bauteile zur Verfügung gestellt.

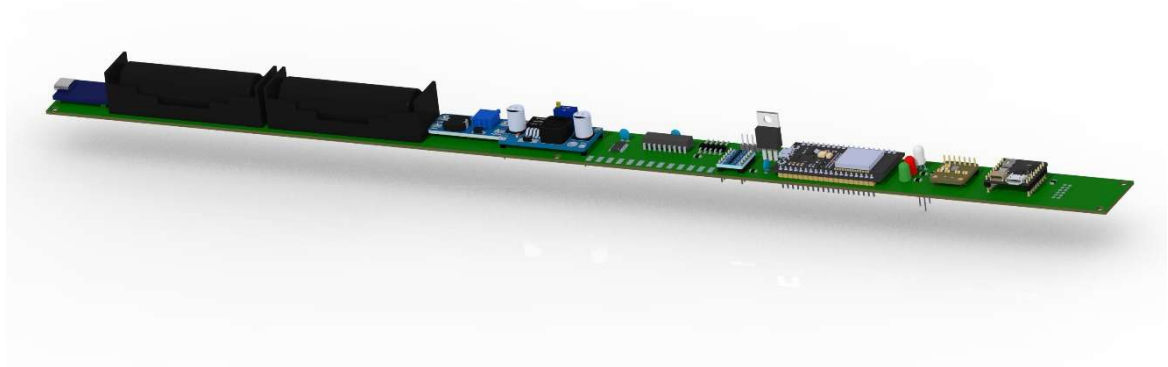


Abbildung 103 Platine Render

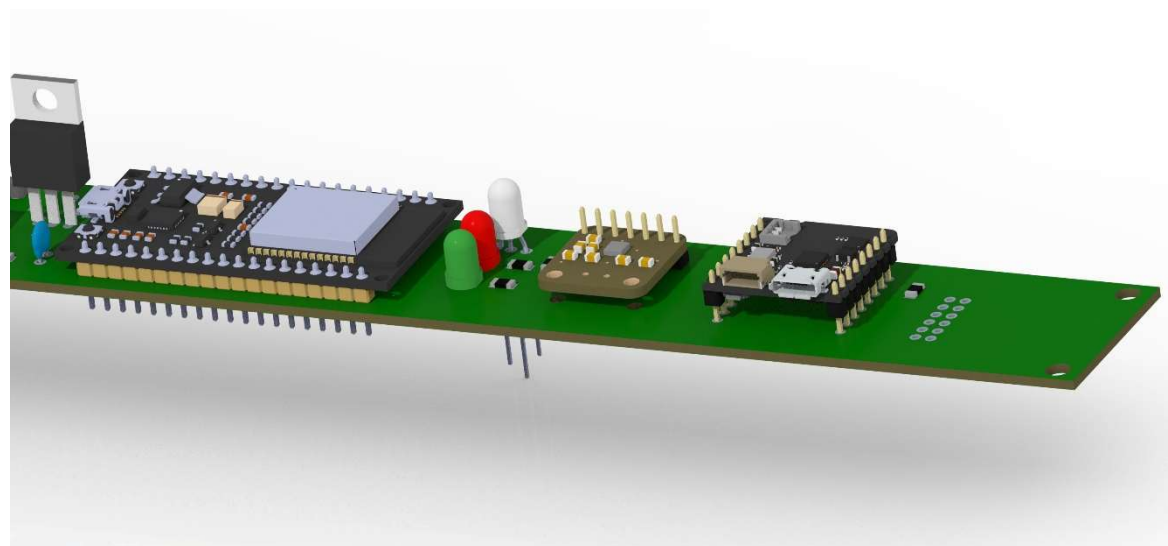


Abbildung 102 Platine Render Sprachsteuerung und -ausgabe

⁵ Für Schaltplan- und Platinen-Entwurf verwendetes Programm

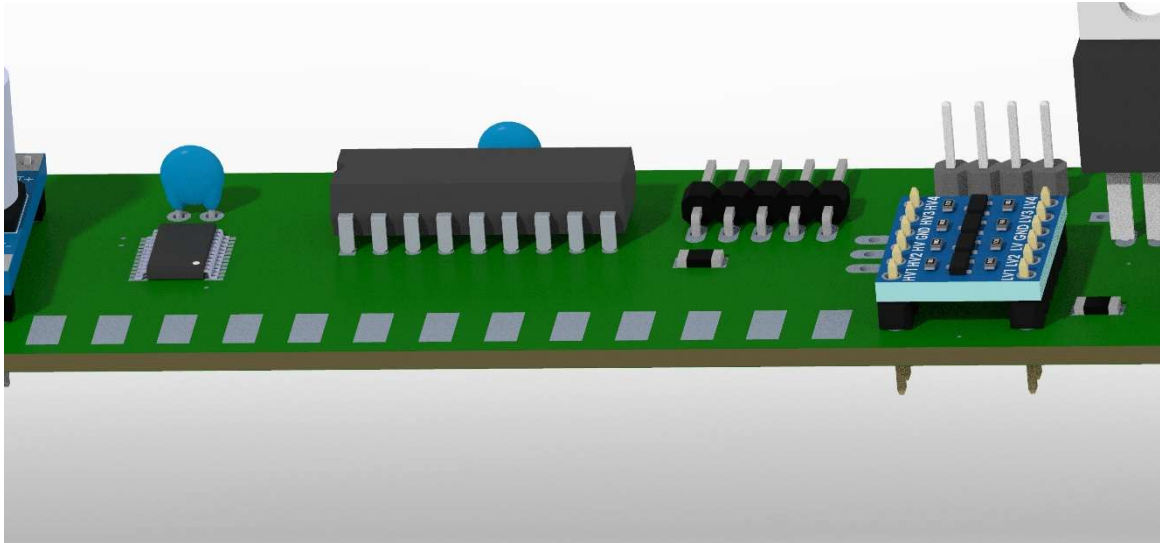


Abbildung 104 Platine Render Messschaltung

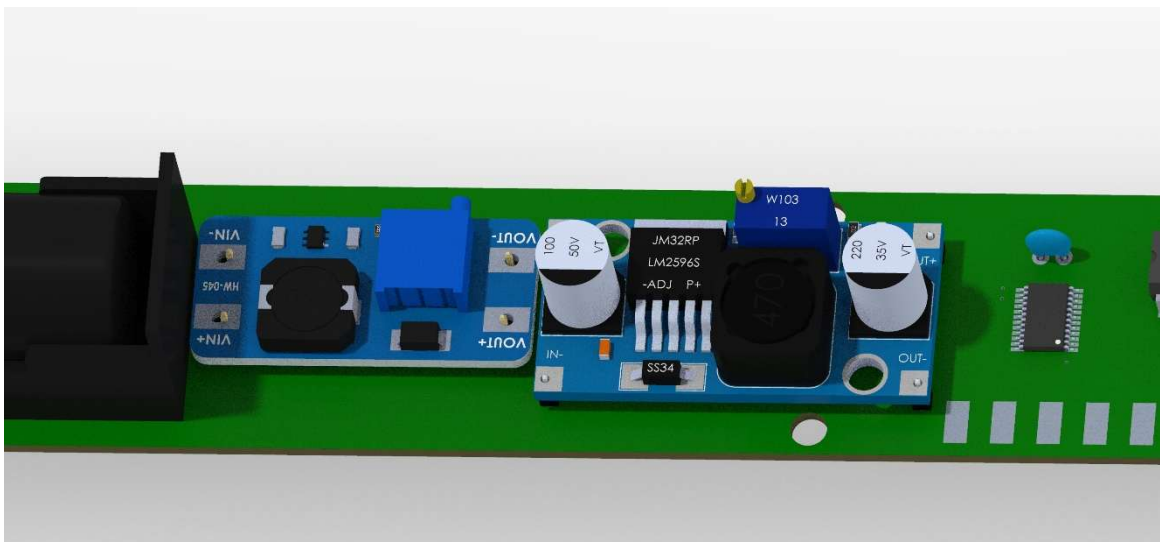


Abbildung 105 Platine Render Stromversorgung

4 Programmierung

4.1 Website

Zu Beginn wurde ein Konzept entwickelt, das festlegt, wie die Website strukturiert ist, wie sie aussieht und was sie anzeigt. Das Aussehen und die Struktur wurden per Hand skizziert, um während der Umsetzung als Vorlage zu dienen. Zwei Programmiersprachen wurden für das Erstellen dieser Website benötigt: HTML als Auszeichnungssprache zur Festlegung der Struktur und CSS als Stylesheet-Sprache um die Darstellung zu definieren.

4.1.1 Visual Studio Code

Visual Studio Code ist ein Texteditor, der ein leichtes Dateimanagement und eine hervorragende Entwicklungsumgebung bietet. Weiters lässt sich das Programm durch die Verwendung von *Extensions* sehr gut auf den eigenen Verwendungszweck anpassen. Visual Studio Code ist geeignet für alle oben genannten Sprachen. Die *Extension* Live-Server wurde verwendet, um auf dem verwendeten Gerät einen lokalen Server zu simulieren. Dadurch kann jede Änderung einer Datei sofort eingesehen werden.

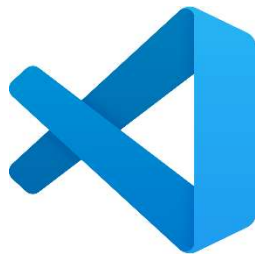


Abbildung 107: Visual Studio Code
[35]



Abbildung 106: Live-Server
[36]

4.1.2 Aufbau der Website

4.1.2.1 Header und Footer

Der Header und der Footer enthalten die Navigationslinks zu den einzelnen Seiten und sind auf jeder Seite gleich. Daher ist das HTML-Markup für alle Seiten für diese beiden Abschnitte gleich. Das Logo gemeinsam mit dem Namen ist in beiden Fällen ein Link, der zurück zur Home-Page führt. Im Header findet man alle Inhaltlichen Seiten. Der Footer enthält das rechtlich erforderliche Impressum sowie die Datenschutzerklärung.

HTML-Markup wird fälschlicherweise oft als HTML-Code bezeichnet.



Abbildung 108: Header



Abbildung 109: Footer

Smartphone Layout

Da ein Smartphone andere Formatierungen voraussetzt, ist der Header platzsparender gestaltet. Die Navigationslinks zu anderen Seiten sind in einem Menü zu finden, das über die drei in Abbildung 110 gezeigten Balken geöffnet werden kann. Abbildung 111 zeigt den ausgeklappten Header, man hat nun dieselben Navigationsmöglichkeiten wie in der Desktop-Version.



Abbildung 110: Smartphone Header

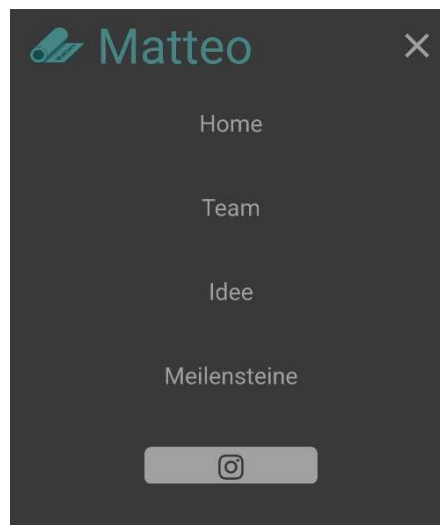


Abbildung 111: Smartphone Header ausgeklappt

4.1.2.2 Home

Wenn ein Zugriff auf die Website über die Domain www.htl3rmatteo.at erfolgt, öffnet sich zuerst die Home-Page. Diese bietet Möglichkeiten zur Navigation auf andere Seiten. Ebenso stellt sie ein 3D-Modell der Matte, den Slogan des Projektes und die Sponsoren dar.

4.1.2.3 Team

Auf der Team-Seite befinden sich Information zu den Mitarbeiter:innen, deren Aufgaben und Kontaktinformationen. Auch die Instagram Seite ist hier verlinkt.

4.1.2.4 Idee

Die Seite enthält eine abstrakte Erklärung des Projekts. Es werden die Ziele sowie die geplanten Kooperationen aufgezählt. Am Ende der Seite befindet sich eine Konzeptskizze der Matte, um ersichtlich zu machen, wie diese aussehen soll und wo sich die Sensoren befinden.

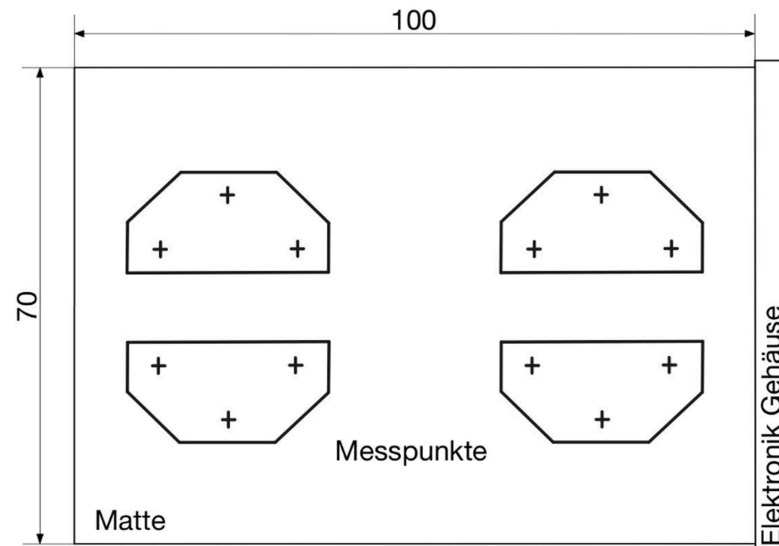


Abbildung 112: Konzeptskizze

4.1.2.5 Meilensteine

Auf dieser Seite wird der Projektfortschritt laufend veröffentlicht. Die Meilensteine sind in Form von Kästchen der Reihe nach untereinander auf der Seite positioniert und haben eine kleine farbliche Kennzeichnung, je nach aktuellem Status.

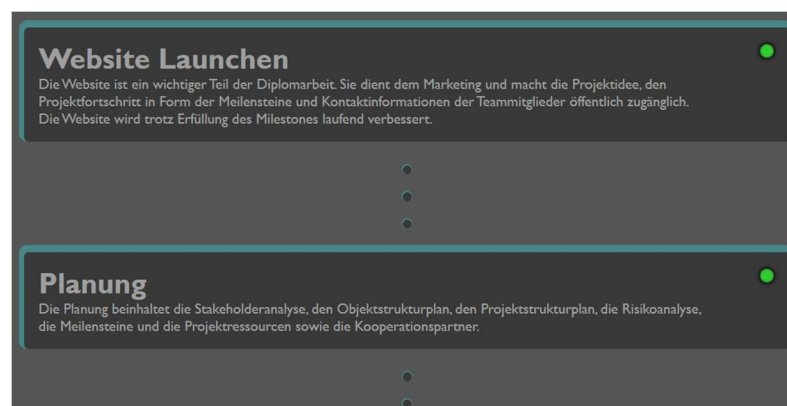
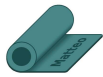


Abbildung 113: Meilensteine



4.1.2.6 Impressum und Datenschutz

Ein Impressum sowie Datenschutz-Informationen sind rechtliche Vorgaben und müssen auf jeder Website abrufbar sein.

Das Impressum beinhaltet alle Angaben gemäß § 5 E-Commerce-Gesetz, § 25 Mediengesetz und § 14 Unternehmensgesetzbuch. Darunter fallen Kontaktinformationen und Adresse der Verantwortlichen, sowie die Zwecke der Website, ein Haftungsausschluss und Urheberrechtsangaben.

Die Datenschutzerklärung macht Informationen zu der Datenverarbeitung auf der Website öffentlich zugänglich. Sie gibt an, wie und in welcher Form bestimmte Daten gespeichert werden.

4.1.3 HTML-Markup

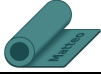
Das HTML-Markup bestimmt die Struktur der Website. Sogenannte *Element-References* oder *Tags* werden verwendet, um eine Seite in verschiedene Abschnitte mit verschiedenen Inhalten und Funktionen zu unterteilen. Diese *Tags* können mit verschiedenen Attributen genauer konfiguriert werden, um das Verhalten der Seite wie gewünscht festlegen zu können.

4.1.3.1 Syntax

Die allgemeine Schreibweise eines HTML-Elements sieht wie folgt aus:

```
<Tag Attribut="Attributwert">Inhalt</Tag>
```

Der Abschnitt vor dem Inhalt ist der sogenannte *Opening-Tag*, der den Beginn festlegt. Der darauffolgende Abschnitt ist der *Closing-Tag*, der das Ende festlegt. Die meisten Tags können sowohl mehrere als auch keine Attribute enthalten. Viele *Tags* haben für sie spezifische Attribute. Es gibt auch Attribute, die jedem *Tag* zugewiesen werden können. Ein Attribut hat immer einen Attributwert, der die Eigenschaft festlegt. Der Inhalt kann auch weitere *Tags* mit eigenen Attributen und Inhalten sein.



4.1.3.2 HTML-Tags

Die aktuelle Version von HTML unterscheidet zwischen 145 verschiedenen *Tags*. In der folgenden Tabelle werden die wichtigsten der verwendeten *Tags* beschrieben. Eine Auflistung aller *Tags* mit genauer Erklärung ist auf der W3Schools-Website [1] zu finden.

Tag	Beschreibung
<html>	Definiert das gesamte HTML-Dokument
<head>	Definiert Metainformationen
<body>	Definiert den Textkörper des Dokuments
<div>	Definiert einen Abschnitt in dem Dokument
<a>	Definiert einen Hyperlink
	Definiert ein Bild
<p>	Definiert einen Absatz
<link>	Verknüpft das Dokument mit einer externen Datei, wie z.B. einem CSS-Stylesheet

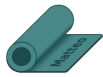
Tabelle 11: HTML-Tags

Metainformationen inkludieren Informationen wie den Titel, die Beschreibung, die Zeichencodierung, verlinkte Stylesheets und vieles mehr.

Ein *Hyperlink* ist der vollständige Begriff für Link. Er verknüpft z.B. ein HTML-Dokument mit einem anderen und leitet, wenn angeklickt auf diese andere Seite weiter.

Hyperlink: [Matteo](#)

Adresse: www.htl3rmatteo.at

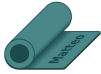


4.1.3.3 HTML-Attribute

Die Attribute eines Elements legen fest wie sich dieses Element zu verhalten hat bzw. welche Eigenschaften es aufweisen soll. Die wichtigsten verwendete Attribute werden in der folgenden Tabelle aufgelistet. Eine Auflistung aller Attribute mit genauer Erklärung ist auf der W3Schools-Website [1] zu finden.

class	Elemente einer Klasse können über den Namen dieser Klasse angesprochen werden
href	Der Attributwert ist ein Link auf eine andere Seite. Der <a> Tag benötigt immer dieses Attribut
src	Legt z.B. für Bilder fest, welche Bilddatei dargestellt werden soll
alt	Kann z.B. eine Bilddatei nicht geladen werden, legt dieses Attribut einen alternativen Text fest
style	Definiert für diesen Tag einen bestimmten Stil

Tabelle 12: HTML-Attribute



4.1.4 CSS

Das CSS ist eine Datei, die festlegt wie die einzelnen in einem HTML-Dokument definierten Elemente aussehen sollen. Eine Website ohne *Stylesheet* ist zwar durch HTML strukturiert, aber nicht formatiert. Das *Stylesheet* legt z.B. die Farben, Positionierung, Anordnung, Größe, Schriftart, und vieles mehr fest. Das C in CSS steht für *Cascading*, also Kaskadierung. Dies bedeutet, dass ein Stil, der auf ein übergeordnetes Element angewendet wird, auch für alle untergeordneten Elemente gilt. Dies macht einheitliches formatieren sehr einfach, kann allerdings bei unvorsichtigem Arbeiten zu Fehlern führen, die schwer zu finden sind.

4.1.4.1 Syntax

Für die Website wurde eine externe CSS-Datei geschrieben, die den Stil jeder Seite festlegt. In jedem HTML-Dokument wurde diese Datei mit dem `<link>` Tag verknüpft.

Die allgemeine Schreibweise einer Stildefinierung sieht wie folgt aus:

```
Selektor {  
    Eigenschaft: Eigenschaftswert;  
}
```

Der Selektor gibt an, welches Objekt auf einer Website mit einem bestimmten Stil ausgestattet werden soll. Klassen werden z.B. mit „Klassenname“ und IDs mit „#ID-Name“ angesprochen werden. Andere Selektoren sind Elemente, untergeordnete Elemente, Elemente mit bestimmten Attributen oder Zustände bestimmter Elemente. Für einen *Hyperlink*, der seine Farbe ändern soll, wenn man mit der Maus darüberfährt, wird folgende Eigenschaft festgelegt:

```
a {  
    color: black;  
}  
a:hover {  
    color: blue;  
}
```

Der erste Selektor legt für alle Elemente, die mit dem `<a>` Tag definiert sind, die Schriftfarbe schwarz fest. Der zweite Selektor spricht ebenfalls alle *Hyperlinks* an. Die Farbe wird allerdings nur auf blau geändert, wenn für den Link der Zustand *hover* gilt. Dies ist der Fall, wenn sich die Maus wie in Abbildung 114 über dem Link befindet.

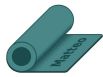


Abbildung 115: Link



Abbildung 114: Link hover

4.1.4.2 CSS-Eigenschaften

CSS unterscheidet zwischen mehr als 200 verschiedenen Eigenschaften. In der folgenden Tabelle sind einige wichtige der verwendeten Eigenschaften angeführt und erklärt. Eine Auflistung aller Eigenschaften mit genauer Erklärung ist auf der W3Schools-Website [1] zu finden.

Eigenschaft	Beschreibung
display	Legt fest, wie der Inhalt dargestellt wird
justify-content	Legt fest, wie die enthaltenden Elemente entlang der Hauptachse ausgerichtet sind
align-items	Legt fest, wie die enthaltenden Elemente entlang der Querachse ausgerichtet sind
color	Legt die Schriftfarbe fest
background-color	Legt die Hintergrundfarbe eines Elements fest.

Tabelle 13: CSS-Eigenschaften

4.2 Übersicht des Microcontrollersystems

Die Matte wird von drei verschiedenen Microcontrollern gesteuert. Jeder Controller hat eine bestimmte Aufgabe. Der ESP32-Wroom-32 ist für das UI (*User Interface*) zuständig. Gemeinsam mit dem Arduino Nicla Voice verarbeitet er die erkannten Sprachbefehle und sendet diese, je nach Befehl, an den PIC16F1827. Der PIC16F1827 regelt die Messung der Drucksensoren und verarbeitet diese Daten. Die Logik der Matte wird ebenfalls von dem PIC16F1827 übernommen. Informationen die an die Benutzer:innen vermittelt werden sollen, werden als Befehle an den ESP32-Wroom-32 weitergeleitet, welcher anschließend bestimmte Audiodateien von einer SD-Karte über einen Lautsprecher ausgibt.

Der ESP32-Wroom-32 wird in den folgenden Kapiteln ESP32 genannt.

Der PIC16F1827 wird in den folgenden Kapiteln PIC genannt.

Der Arduino Nicla Voice wird in den folgenden Kapiteln Arduino genannt.

Ein „0b“, gefolgt von einer Reihe Nullen und Einsen, gibt an, dass es sich um eine Binärzahl handelt.

Blockbild

Abbildung 116 zeigt, wie die einzelnen Prozessoren miteinander kommunizieren und welche Aufgaben sie haben.

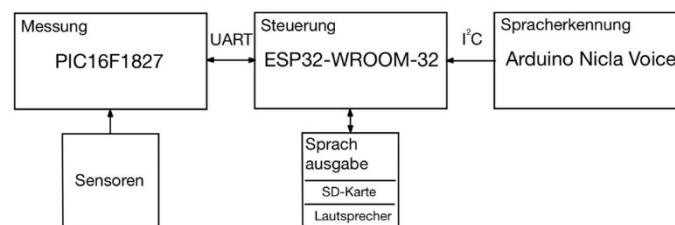


Abbildung 116: Blockbild

4.3 Firmware des PIC

Der PIC ist zuständig für das Einlesen der Drucksensordaten. Diese werden mithilfe des ADC-Moduls (*Analog-to-Digital Converter Module*) in einen Digitalen Wert umgewandelt. Nachdem die Daten ausgewertet und verarbeitet wurden, wird ein Befehl erstellt, der notwendige Informationen an die Sprachausgabe liefert, um auditive Rückmeldung an die Benutzer:innen zu geben. Ebenso können Befehle von der Sprachsteuerung empfangen werden. Diese beinhalten Anweisungen zu verschiedenen Einstellungen, die bei der Datenverarbeitung berücksichtigt werden.

4.3.1 MPLABX IDE

MPLABX ist ein Integrated Development Environment von Microchip, dem Hersteller des PICs. Es eignet sich hervorragend für die Entwicklung, da es In-Circuit-Debugging unterstützt. Außerdem bietet die Umgebung viele Werkzeuge, die das Entwickeln und *Debuggen* deutlich vereinfachen.

Eines dieser Werkzeuge ist der MCC (*MPLAB Code Configurator*), welcher Header-Dateien zu den ausgewählten und konfigurierten Modulen erstellt. Diese Dateien enthalten Funktionen, die aus dem Hauptprogramm aufgerufen werden können. Der MCC wurde verwendet, um grundlegende Einstellungen für das System, wie z.B. die *System-Clock*, den *Watch-Dog-Timer*, vorläufige Pinbelegung, etc., zu generieren.

Der benötigte Compiler ist der MPLAB XC8 Compiler. Während dem Installationsprozess der Entwicklungsumgebung, kann dieser gewählt und ebenso installiert werden. Dieser *Compiler* ist für 8Bit-Prozessoren von Microchip vorgesehen.

4.3.2 H-Term

H-Term ist ein *Terminal*-Programm, welches verschiedene Ports eines Computers auslesen kann. Es wird verwendet, um mithilfe der UART-Schnittstelle mit dem PIC zu kommunizieren. Das Programm erlaubt unzählige verschiedene Einstellungen und Konfigurationen, doch für diesen Verwendungszweck müssen folgende Parameter eingestellt werden.

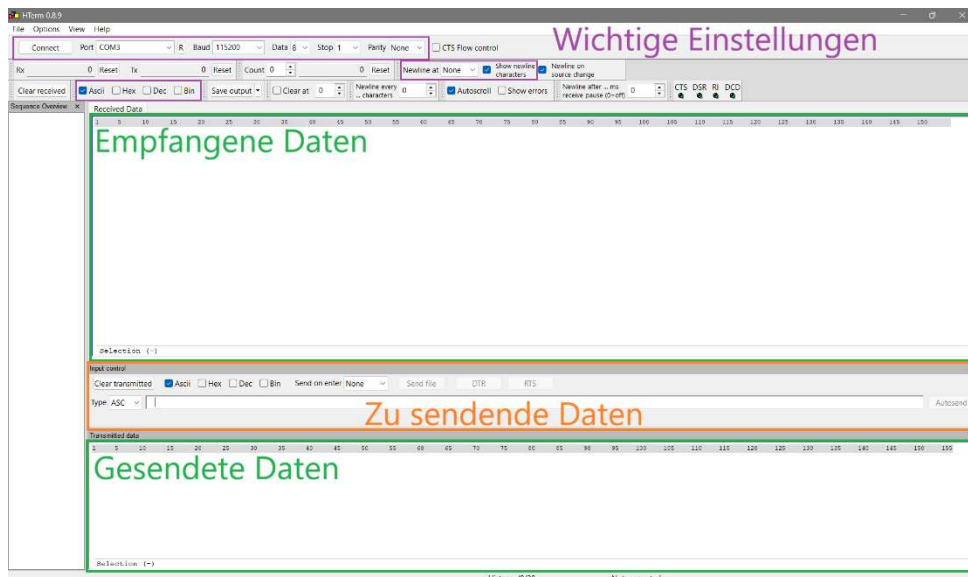


Abbildung 117: H-Term [37]

In Abbildung 117 sind die wichtigsten Bereiche und wofür sie genutzt werden, markiert. Nachdem mit einem UART-zu-USB-Adapter eine Verbindung zwischen dem PIC und dem Gerät auf dem H-Term aktiv ist, hergestellt wurde, muss der *Port* des Adapters im passenden Feld oben links in der Anwendung ausgewählt werden. Welcher *Port* das genau ist, kann im Windows-Gerätemanager nachgesehen werden. Anschließend wird im Feld rechts davon die gewünschte Baudrate eingestellt. Der PIC arbeitet mit 8 Daten-Bits, einem Stopp-Bit und ohne *Parity*. Zu empfehlen ist, unter der Einstellung „Newline at“ „LF“ (Line Feed) zu wählen und „Show newline character“ für bessere Lesbarkeit auszuschalten. Bevor sich nun oben rechts mit der Schnittstelle verbunden wird, muss noch das Format gewählt werden, wie die Daten angezeigt werden sollen.

Um Daten zu senden, wird die gewünschte Zeichenabfolge in die Zeile „Type“ in der Mitte vom Bildschirm im orangen markierten Feld geschrieben. Ein Dropdown-Menü erlaubt das Einstellen des Formats, wie die eingegebenen Daten interpretiert werden sollen. Standardgemäß ist „ASC“ (ASCII) ausgewählt. Im Feld „Send on enter“ kann eingestellt werden, dass jede gesendete Nachricht mit einem LF beendet wird.

Dieses Programm eignet sich hervorragend für das Debuggen des PIC.

4.3.3 ADC (Analog-to-Digital Converter Module)

Der ADC wird verwendet, um die analogen Sensordaten in digitale Werte umzuwandeln. Das Modul basiert auf dem Prinzip der sukzessiven Annäherung, eine sehr weit verbreitete Methode der analog zu digital Wandlung. Das Modul wandelt ein analoges Signal in ein 10Bit binären Wert um. Es bildet das Signal auf einen Wert zwischen 0 und 1023 ab.

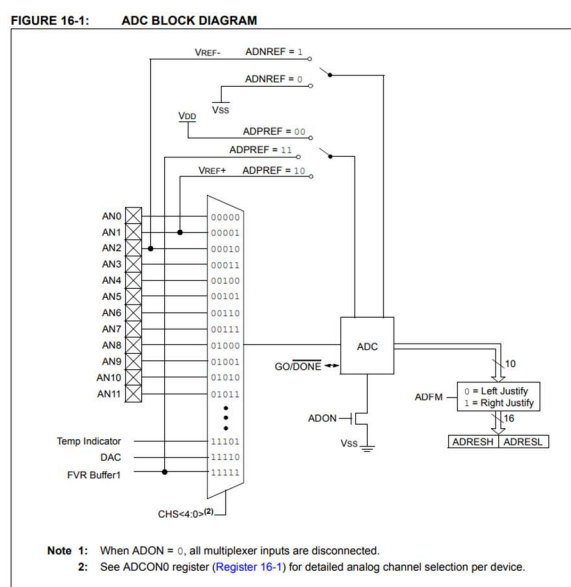


Abbildung 118: ADC Block Diagramm [3]

4.3.3.1 Wandelabweichung und Spannungsreferenzen

Da das analoge Eingangssignal auf einen 10Bit Wert abgebildet wird, gibt es eine bestimmte Wandelabweichung. Der exakte Wert liegt innerhalb einer Toleranz über oder unter dem gewandelten Wert. Diese Toleranz hängt vom Messbereich ab und wird Auflösung genannt. Das Modul bietet, wie in Abbildung 118 zu sehen, verschiedene Möglichkeiten die Referenzspannungen zu wählen.

Wenn das Eingangssignal sich beispielweise nur zwischen 0 V und 4 V befinden kann, kann eine höhere Auflösung verwendet werden. Wird als negative Referenz der VSS-Pin, also 0 V und als positive Referenz der VDD-Pin, also 5 V gewählt, ergibt das eine Auflösung von 4,88 mV pro digitalen Wert. Wird nun ein Signal gewandelt mit dem Ergebnis 450, wird dieser Wert mit 0,00488 V multipliziert und entspricht 2,196 V. Der exakte Wert kann aber um die halbe Schrittbreite über oder unter diesem Wert liegen, zwischen 2,198 und 2,194. Die Abweichung ist somit $\pm 2,44$ mV

Wird nun die FVR (*Fixed Voltage Reference*) mit 4-facher Verstärkung, also 4,096 V verwendet, verringert sich die Abweichung auf $\pm 2,00$ mV. Der Wertebereich hat sich nicht geändert, aber das gewandelte Signal ist, nachdem 5 Bits in den SFR (*Special Function Register*) gesetzt wurden, um 22% genauer. Für die Konfiguration des Moduls und der FVR siehe Kapitel 16.0 und 14.0 des PIC Datenblattes [2].

In Abbildung 119 ist eine vereinfachte Version der Messschaltung zu sehen, der Drucksensor wird als Potentiometer dargestellt. Ist der Sensor unbelastet, hat er einen Widerstandswert von einigen M Ω . In dieser Schaltung wirkt das wie eine Unterbrechung, somit liegen am ADC-Eingang 5 V an. Bei steigender Belastung nähert sich dieser Widerstandswert immer mehr 0 Ω an. Das Eingangssignal kann somit Werte zwischen 0 V und 5 V annehmen. Als positive Referenz wird der VDD-Pin und als negative Referenz wird der VSS-Pin verwendet. Der gewandelte Wert hat somit eine Auflösung von $\pm 2,44$ mV.

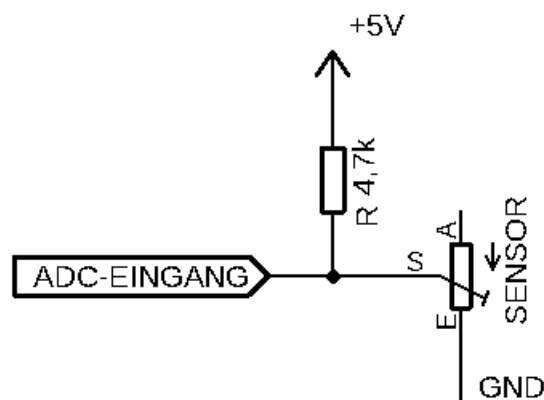


Abbildung 119: Messschaltung

4.3.3.2 ADC-Kanal

Der ADC kann, für den zu wandelnden Signalpegel, aus 15 Möglichkeiten wählen. 12 davon sind GPIO-Pins (*General Purpose Input Output*) und somit für diese Anwendung relevant. Es wäre somit möglich 12 Sensoren direkt mit diesen Pins zu verbinden. Dies setzt allerdings voraus, dass diese Pins für nichts anderes verwendet werden müssen, was nicht der Fall ist. Um die benötigte Anzahl an GPIO-Pins zu verringern und trotzdem 12 Sensoren verwenden zu können wird der MUX (Multiplexer, siehe Kapitel 2.3.1.1) verwendet. Dieser wird mit 4 Eingängen, die eine 4Bit Adresse darstellen gesteuert. 4 Bits können eindeutig 16 Kanäle ansprechen. Der Signalausgang des MUX benötigt einen weiteren GPIO-Pin. Dieser wird, je nach angesteuertem MUX-Kanal, mit einem der Sensoren verbunden. Der Signalausgang kann also mit einem Pin des PICs verbunden werden, der auch als ADC-Eingang verwendet werden kann. Gewählt wurde RA4. Die Adresspins sind RA0 bis RA3. Der erste MUX-Kanal hat die Adresse 0. Um den Zweiten anzusprechen, muss RA0 auf HIGH geschaltet werden.

Abbildung 121 zeigt 4 Zeilen Code die den MUX ansteuern. Sie befinden sich in einer Funktion, die einen 8Bit unsigned *Integer* namens *Adress* übernimmt. *Adress* wird zuerst gemeinsam mit einer Maske einer Bitwise-AND-Operation unterzogen. Der „?“ Operator ist ein so genannter *Conditional Operator*. Gemeinsam mit dem „:“ ist dies eine einzeilige Möglichkeit, eine IF-Bedingung festzulegen. Abbildung 120 zeigt die allgemeine Schreibweise. Ist die Bedingung TRUE, wird in *Variable* der Wert *wenn_TRUE* gespeichert. Ist die Bedingung FALSE, der Wert *wenn_FALSE*. Hat *Adress* nun den Wert 13, bzw. 0b00001101, entspricht die Bedingung in der ersten Zeile einer 1, bzw. 0b00000001. In der Programmiersprache C ist jeder Integer, der nicht 0 ist, TRUE. Somit ist diese Bedingung TRUE und der RA0 Pin wird auf HIGH geschaltet. In der zweiten Zeile entspricht die Bedingung einer 0, der Pin bleibt somit LOW. Die Bedingungen in Zeile 3 und 4 entsprechen den Werten 4 und 8, und sind aus oben genanntem Grund TRUE. Die Vier Pins haben nun die Werte 1, 0, 1 und 1 was wiederum 13 entspricht.

```
252 | LATAbits.LATA0 = (0b00000001 & Address) ? 1 : 0;  
253 | LATAbits.LATA1 = (0b00000010 & Address) ? 1 : 0;  
254 | LATAbits.LATA2 = (0b00000100 & Address) ? 1 : 0;  
255 | LATAbits.LATA3 = (0b00001000 & Address) ? 1 : 0;
```

Abbildung 121: MUX ansteuern

```
Variable = Bedingung ? wenn_TRUE : wenn_FALSE;
```

Abbildung 120: IF-Anweisung

4.3.3.3 Initialisierung

Um den ADC verwenden zu können, müssen bestimmte SFR konfiguriert werden. Abbildung 122 zeigt die Funktion, die dies zu Beginn des Programmstarts durchführt.

```
125 void ADC_init(void)
126 {
127
128     ADCON0 = 0b00010001;
129     ADCON1 = 0b01010000;
130     PIR1bits.ADIF = 0;
131
132     ADRESH = 0x0;
133     ADRESL = 0x0;
134     PIR1bits.ADIF = 0;
135     ADCON1bits.ADFM = 1;
136
137 }
```

Abbildung 122: ADC-Initialisierung

4.3.3.4 Sensordatenerfassung

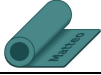
Die Sensordatenerfassung erfolgt mithilfe zweier Funktionen.

4.3.3.4.1 Erfassung eines einzelnen Sensorwertes

Diese Funktion übernimmt eine 8Bit *Integer* Adresse, die angibt welcher Sensor gemessen werden soll. Anschließend wird wie in Kapitel 4.3.3.2 der jeweilige MUX-Steuerpin ein- oder ausgeschaltet. Um nun die ADC-Wandlung zu starten, wird das GO_nDONE Bit des ADCON0 Registers auf 1 gesetzt. Dieses Bit bleibt so lange auf 1, bis die Wandlung abgeschlossen wird. Es kann also ohne mehr Verzögerung als notwendig, gewartet werden, bis das Ergebnis bereitsteht. Die benötigte Zeit liegt im Bereich von 10 μ s.

Das Ergebnis wird aus dem ADRESH und ADRESL Registerpaar ausgelesen und für Debugging-Zwecke gespeichert. Für die Sensordaten wurde ein globales *Array* definiert. Dieser hat die Größe 12 mal 16Bit Integer und kann somit die Ergebnisse aller 12 Sensoren speichern. Der Sensorwert wird an derselben Adresse im *Array* gespeichert, mit der er über den MUX angesteuert werden kann.

Da die Widerstände der Drucksensoren im unbelasteten Zustand sehr hoch sind und sich bei Belastung immer weiter an 0 annähern, ist der Messwert im unbelasteten



Zustand hoch und im belasteten Zustand niedrig. Um dies umzudrehen, wird als Sensorwert die Differenz des Maximalwert 1023 und dem Messwert gespeichert.

Der Widerstandsabfall bei steigender Belastung ist allerdings kein linearer Zusammenhang. Wenn eine Person von 20 % Belastung auf 30 % Belastung des rechten Fußes steigern will, benötigt dies deutlich weniger Druckunterschied als von 70 % auf 80 %. Sowohl die Messwerte als auch die Verhältnisse verlaufen nicht linear. Wird von dem Messwert nun die Wurzel gezogen, verlaufen diese Werte zwar noch immer nicht linear, doch die Verhältnisse der durchschnittlichen Messwerte sind annähernd linear. Wird die Wurzel gezogen, kann eine Präzision erreicht werden, die für das Menschliche Gleichgewichts Gefühl nicht erkennbar und somit vollkommen ausreichend ist.

4.3.3.4.2 Erfassung aller Sensorwerte

Diese Funktion ruft die Funktion aus dem vorigen Kapitel insgesamt 12-mal auf, um alle Sensoren der Reihe nach auszumessen. Weiters wird nur noch ein durchschnittlicher Messwert in einer globalen Variable gespeichert. Ein Aufrufen dieser Funktion aktualisiert alle globalen Sensordatenvariablen innerhalb von 300 μ s auf den aktuellen Belastungswert.

4.3.4 Timer1 (Timer1 Module)

Um bestimmte Prozesse im Programm nach einem festgelegten Zeitintervall zu wiederholen, wird eine globale Variable benötigt, die Sekunden zählt. Hierfür wird der Timer1 verwendet. Dieser ist ein 16Bit Zähler mit vielen Funktionen. Für diesen Verwendungszweck wird er so konfiguriert, dass er jeden 8. *Instruction Cycle* zählt. Der *Instruction Cycle* ist der Takt, an dem eine *Assembly*-Anweisung ausgeführt wird. Dieser Takt ist um das Vierfache langsamer als die *System-Clock*. Der PIC arbeitet mit einem System Takt von 16 MHz. Der Timer1 zählt also in einem Takt von 500 kHz hoch.

4.3.4.1 Sekunden zählen

Der Timer1 enthält ein 16-Bit Registerpaar welches den Zählwert enthält. Es kann Werte von 0 bis 65535 halten. Wenn der Timer1 den maximalen Wert enthält und hochzählt, wird er automatisch auf 0 gesetzt und beginnt im nächsten Takt wieder mit 1. Wenn dieses so genannte *Rollover* stattfindet, wird, wenn konfiguriert, ein Interrupt ausgelöst. Mit einem Takt von 500 kHz passiert dies alle 131,06 ms.

Timer1 Registerpaar beschreiben

Mit einem *Rollover*-Takt von rund 7,63 Hz können nur mit ungenauen Berechnungen die Sekunden gezählt werden. Der Timer1 lässt jedoch zu, dass sein Registerpaar, dass den Zählwert enthält, beschrieben wird. Der größtmögliche Wert, der 500

kHz ohne Rest teilen kann, ist 50 000. Damit der Timer1 genauso oft hochzählt und somit jede zehntel Sekunde ein *Rollover* auftritt, das einen Interrupt auslöst, muss nach jedem *Rollover* der Wert 15536 in das Registerpaar geschrieben werden. Der Timer1 beginnt nun nicht von 0 zu zählen, sondern von diesem Wert. nach 49 999 Takten ist der maximale Wert erreicht. Im nächsten Takt wird das *Rollover* ausgelöst. Dies passiert nun in einem Takt von 10Hz. Nach 10 Interrupts ist also eine Sekunde vergangen und es wird eine Variable inkrementiert. Diese Variable hat den Datentyp *unsigned long* und kann somit einen Wert zwischen 0 und 4294967296 annehmen. Das entspricht ungefähr 136,2 Jahren und wird keine Probleme durch Erreichen des maximalen Wertes aufweisen.

4.3.4.2 Interrupt-basierte Zeitmessung

Wie im vorigen Kapitel erklärt, funktioniert die Sekundenzählung mithilfe der Interrupts, die nach einem *Rollover* ausgelöst werden. Damit der Timer1 den oben genannten Anforderungen entspricht, muss er wie folgt initialisiert werden. Der Timer1 kann mit einem so genannten *Prescaler* konfiguriert werden. Um einen Prescaler von 8 einzustellen, muss das T1CON Register mit 0b00110000 beschrieben werden. T1GCON kann mit 0 initialisiert werden. Die TMR1IF (Timer1 *Interrupt-Flag*) wird auf 0 gesetzt und es wird zugelassen, dass diese Flagge Interrupts auslöst. Zuletzt wird das Modul eingeschalten.

In der ISR (*Interrupt Service Routine*), die Funktion, die aufgerufen wird, wenn ein Interrupt jeglicher Art ausgelöst wird, muss überprüft werden, ob die TMR1IF Grund für das Auslösen ist. Wenn die Flagge den Wert 1 enthält, wird eine globale Zählvariable inkrementiert und der Wert 15536 in das Timer1H/L Registerpaar geschrieben. Erreicht die Zählvariable den Wert 10, wird sie auf 0 gesetzt und die Sekundenvariable wird inkrementiert. Nach jedem TMR1IF *Interrupt* muss diese Flagge wieder zurückgesetzt werden.

Das Zählen der Sekunden passiert somit vollkommen unabhängig vom restlichen Programm.

4.3.5 GPIO-Pins (General Purpose Input Output Pins)

Diese Pins und deren Funktionen können für verschiedene Verwendungszwecke durch das Programm konfiguriert werden. Diese Funktionen sind z.B. ADC-Eingänge, UART-Ein- und Ausgänge, in der Lage *Interrupts* auszulösen und einige mehr, siehe Tabelle 1 des PIC Datenblattes [2]. Der PIC hat insgesamt 18 Pins, 16 davon sind GPIO-Pins und die anderen sind der VSS- und der VDD-Pin. Die GPIO-Pins sind in zwei Gruppen aufgeteilt, die jeweils von identischen Registergruppen gesteuert werden. Die Pin-Gruppierungen heißen PORTA und PORTB, sie enthalten jeweils 8 Pins von RA0 bzw. RB0 bis RA7 bzw. RB7.

PORTX-Initialisierungen

Die Bezeichnung PORTX wird verwendet für sowohl PORTA als auch PORTB. PORTX sind 5 Register zugeordnet, die die allgemeinen Funktionen der 8 Pins festlegen. Es wird durch TRISX konfiguriert, ob die Pins als Ein- oder Ausgänge verwendet werden, durch ANSELX, ob sie analoge oder digitale Signale empfangen und durch WPUX, ob die internen *weak-Pull-Up* Widerstände verwendet werden. Das PORTX Register enthält Informationen ob die Pins HIGH oder LOW sind. Es kann beschrieben werden, allerdings ist durch die Hardware geregelt, dass dadurch das LATX geschrieben wird, welches den Status der Pins ändert, und dadurch PORTX ändert. LATX kann direkt geschrieben werden. Allgemein kann gesagt werden, PORTX wird gelesen und LATX wird geschrieben.

Die insgesamt 10 Register müssen alle initialisiert werden. Zu Beginn werden jeweils beide PORTX und LATX-Register auf 0 gesetzt. Wird ein Pin in einem *Active-LOW* Zustand verwendet, sollten die dazugehörigen PORT und LAT-Bits auf 1 initialisiert werden. Da keine *Pull-Up* Widerstände verwendet werden, werden auch die WPUX-Register mit 0 beschrieben. Der RX-Pin RB1, der für das Empfangen der Daten mittels des UART-Moduls zuständig ist, als auch der Eingang für das ADC-Modul wird als INPUT definiert, indem deren jeweiligen TRISX-Bits auf 1 gesetzt werden. Der ADC-Modul-Eingang muss dazu noch als analoger Pin festgelegt werden. Dazu wird das 5. Bit des ANSELA Registers mit 1 beschrieben. Abbildung 123 zeigt die Initialisierungs-Funktion.

```
114 void GPIO_init(void)
115 {
116     // RB      76543210
117     PORTB = 0b00000000;
118     LATB  = 0b00000000;
119     ANSELB = 0b00000000;
120     WPUB  = 0b00000000;
121     TRISB = 0b00000010; // RB1 RX - Input
122
123     // RA      76543210
124     PORTA = 0b00000000;
125     LATA  = 0b00000000;
126     ANSELA = 0b00010000; // RA4 ADC_IN - Analog ... Sensoreingang
127     WPUA  = 0b00000000;
128     TRISA = 0b00010000; // RA4 ADC_IN - Input ... Sensoreingang
129 }
```

Abbildung 123: GPIO-Initialisierung

4.3.6 UART (Universal Asynchronous Receiver Transmitter)

Der UART, auch RS-232 genannt, ist eine serielle, voll duplexfähige und asynchrone Schnittstelle. Im PIC befindet sich genau genommen ein EUSART-Modul (*Enhanced Universal Synchronous Asynchronous Receiver Transmitter*) welches mehr Funktionen als das UART-Modul hat. Da in diesem Fall allerdings diese Funktionen nicht benötigt werden und das Modul wie ein UART-Modul verwendet wird, wird es in



den folgenden Abschnitten auch so genannt. Der UART wird eingesetzt, um Daten zwischen dem PIC und dem ESP32 auszutauschen. Die Schnittstelle arbeitet mit zwei Datenleitungen und beide Teilnehmer können gleichzeitig senden und empfangen.

4.3.6.1 BRG (Baud Rate Generator)

Die Baudrate des Moduls entspricht der Bitrate und gibt an, wie viele Bits pro Sekunde übertragen werden. Da die Schnittstelle keine *Clock*-Leitung hat, muss bei beiden Teilnehmern dieselbe Baudrate eingestellt werden. Gewählt wurde eine Baudrate von 57600. Der BRG hat ein Registerpaar, welches das Einstellen der gewünschten Baudrate ermöglicht. Es wird jedoch nicht der Wert der Baudrate in diese Register geschrieben da nur Werte von 0 bis 65535 möglich wären. Je nach Konfiguration des BRG, wird eine von 3 Formeln zur Berechnung der Baudrate verwendet. Der UART wird im Asynchronmodus verwendet, somit sind zwei Bits für die Berechnung von Bedeutung, BRG16 und BRGH. Abbildung 124 zeigt die jeweiligen Konfigurationen, das Bit SYNC ist in diesem Fall nicht von Bedeutung, es wird mit 0 initialisiert.

TABLE 26-3: BAUD RATE FORMULAS

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	$F_{osc}/[64 (n+1)]$
0	0	1	8-bit/Asynchronous	$F_{osc}/[16 (n+1)]$
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	$F_{osc}/[4 (n+1)]$
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

Legend: x = Don't care, n = value of SPBRGH, SPBRGL register pair

Abbildung 124: Baud-Rate-Formeln [3]



Da das Registerpaar nur ganze Zahlen darstellen kann, ergibt sich eine gewisse Ungenauigkeit bei der Berechnung der Baudrate. Die exakte Baudrate entspricht in den meisten Fällen nicht der gewünschten und besitzt somit einen Fehler. Durch Umformen der Baudratenformel aus Abbildung 124, lässt sich für eine gewünschte Baudrate die exakte Baudrate, die Abweichung sowie der Registerpaarwert berechnen. Die Konfiguration der Bits BRG16 und BRGH kann dann nach kleinstem Fehler gewählt werden. Folgende Tabelle zeigt die Werte der 3 Möglichkeiten.

BRG16	BRGH	Exakte Baudrate	Fehler	Registerpaarwert
1	1	57971	0,6441%	68
0	1	58824	2,124%	16
1	0	58824	2,124%	16
0	0	62500	8,507%	3

Tabelle 14: Baudrate-Konfiguration

Der BRG wird somit konfiguriert mit BRG16 und BRGH auf 1 und einem Registerpaarwert von 68 oder 0b0000000001000100.

4.3.6.2 Funktionsweise

Das Modul arbeitet mit zwei voneinander getrennten und unabhängigen Untermodulen, wobei eines für das Senden und das andere für das Empfangen zuständig ist. Das Modul arbeitet mit zwei Spannungslevels, einem HIGH-Zustand, auch *mark* genannt, welcher einer 1 entspricht und einem LOW-Zustand, auch *space* genannt, welcher einer 0 entspricht. Eine Datenübertragung hat verschiedene Merkmale. Im unbenutzten Zustand befindet sich die Datenleitung im *mark*-Zustand. Der Übertragungsbeginn wird mit einem Start-Bit signalisiert, darauf folgen 8 oder ggf. 9 Daten-Bits und die Übertragung wird mit einem oder mehreren Stopp-Bits beendet. Das Modul arbeitet mit dem NRZ-Format (non-return-to-zero). NRZ ist implementiert mit den oben genannten Zuständen *mark* und *space*. NRZ legt fest, wenn zwei aufeinander folgende Daten-Bits denselben Wert haben, bleibt die Datenleitung im jeweiligen Zustand und kehrt nicht in einen neutralen Zustand zwischen zwei Bit-Übertragungen zurück. Das Start-Bit ist ein *space* und das oder die Stopp-Bits sind *marks*. Der UART sendet und empfängt das LSB (*Least significant Bit*) zuerst.

4.3.6.2.1 TX (Asynchronous Transmitter)

Der Kern des TX ist das TSR (Transmit Shift Register), ein Schieberegister, welches den zu sendenden Wert, beginnend mit dem LSB in einem bestimmten Takt auf dem

TX-Pin ausgibt. Das TSR kann nicht durch die Software beschrieben werden. Deshalb hat der TX ein weiteres Register, das sogenannte TXREG ist ein Buffer-Register, welches einen Wert von der Software übernimmt und diesen in das vorausgesetzt freie TSR überträgt. Ist das TSR nicht frei, der TX befindet sich also in einem Übertragungsvorgang, bleibt der Wert so lange im Buffer stehen, bis der Vorgang beendet ist. Der TX kann somit 2 Werte halten.

Der TX hat zwei Status-Bits bzw. Flaggen, mit deren Hilfe man eine verzögerungsfreie Übertragung ohne Datenverlust regeln kann. Wird das TXREG beschrieben, während der vorherige Wert noch nicht in das TSR übertragen wurde, geht dieser verloren. Um dies zu verhindern, wird die TXIF (*Transmit Interrupt Flag*) überprüft. Diese enthält den Wert 1, wenn das TXREG bereit ist einen Wert zu übernehmen und kann, wenn zugelassen, einen *Interrupt* auslösen. Eine ähnliche Funktion das TRMT-Bit (*Transmit Shift Register Status*), dieses zeigt, ob das TSR leer ist kann aber keinen *Interrupt* auslösen.

Abbildung 125 zeigt die Übertragung eines einzelnen Bytes. Für eine genaue Erklärung des TX und des Übertragungsvorganges, siehe Kapitel 26.1.1 des PIC-Datenblattes [2].

FIGURE 26-3: ASYNCHRONOUS TRANSMISSION

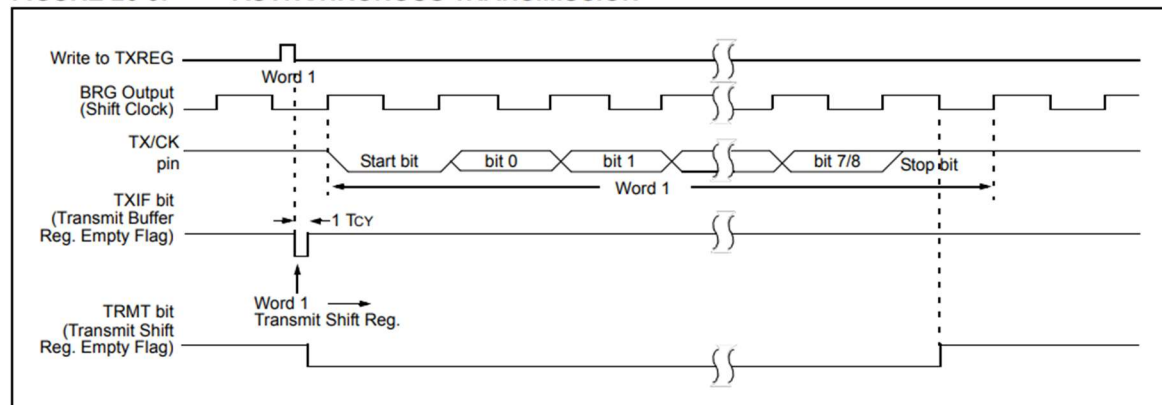


Abbildung 125: Sendevorgang [3]

4.3.6.2.2 RX (Asynchronous Receiver)

Der Kern des RX ist das RSR-Register, welches die Daten einer laufenden Übertragung enthält. Der RX, wenn eingeschaltet, wartet auf einen *space*, also ein Start-Bit. Wurde dieses empfangen, wird es gemeinsam mit den darauffolgenden Daten und dem oder den Stopp-Bits in das RSR geschrieben. Ist die Übertragung abgeschlossen, werden die Daten in das FIFO-Register (*First-in-First-out*) RCREG (*Receive Data Register*) geschrieben. Dieses Register kann 2 Bytes halten. FIFO bedeutet, das erste Byte, das hineingeschrieben wurde, ist auch das erste welches ausgegeben wird. Die empfangenen Daten werden also von vorne ausgelesen und von hinten nachgelegt.

Wurde nach den Daten-Bits ein *space* empfangen, also kein Stopp-Bit, entspricht dies einem sogenannten *Framing Error* und das FERR-Bit des RX wird gesetzt. Dieses Bit muss nicht auf 0 gesetzt werden. Ebenso muss nicht beachtet werden ob so ein Fehler aufgetreten ist, da das Übertragungsprotokoll die Eventualität eines Übertragungsfehlers berücksichtigt und ihn behebt.

Befindet sich ein Wert in dem RCREG, wird die RCIF (*Receive Interrupt Flag*) auf 1 gesetzt und löst, sofern zugelassen, einen *Interrupt* aus. Wird das RCREG nicht ausgelesen und ein dritter Wert empfangen, der nicht in das volle FIFO-Register geschrieben werden kann, löst dies einen sogenannten *Overrun-Error* aus und setzt das OERR-Bit. Um diesen *Error* zu beheben, muss der RX neugestartet werden, indem das CREN-Bit (*Continuous Receive Enable*) zuerst auf 0 und dann wieder auf 1 gesetzt wird. Bevor die Daten aus dem RCREG ausgelesen werden, wird überprüft, ob dieser Error aufgetreten ist.

Abbildung 126 zeigt den Empfangsvorgang eines einzelnen Bytes. Für eine genaue Erklärung des RX und des Empfangsvorganges, siehe Kapitel 26.1.2 des PIC-Datenblattes [2].

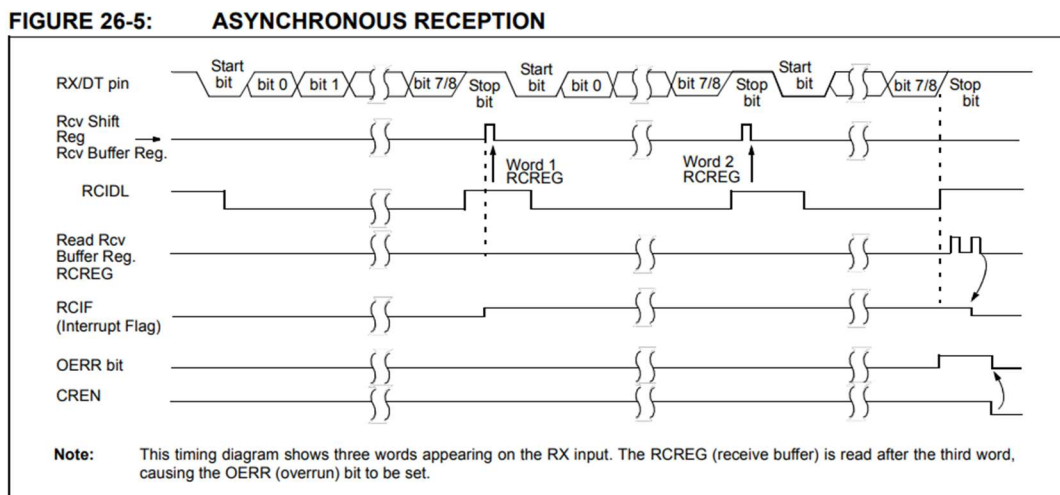


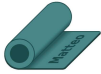
Abbildung 126: Empfangsvorgang [3]

4.3.6.3 Initialisierung

Der UART muss so konfiguriert werden, dass das in Kapitel 4.3.6 genannte EUSART-Modul die Anforderungen erfüllt, die für diesen Anwendungszweck benötigt werden. Abbildung 127 zeigt die Funktion, welche diese Konfiguration nach dem Programmstart ausführt. Zu Beginn wird das BRG-Registerpaar mit dem berechneten Wert beschrieben. Auch die Bits BRG16 und BRGH werden mit dem gewählten Zustand initialisiert. Da das Modul als UART verwendet wird, wird das SYNC-Bit auf 0 gesetzt und somit der asynchrone Betrieb festgelegt. Die Schnittstelle, der Transmitter und der Receiver werden eingeschaltet. Zuletzt wird festgelegt, bei welchen Ereignissen das Modul einen *Interrupt* auslösen darf. Das Kapitel 4.3.6.4 geht genauer auf die Interrupts dieses Moduls und somit auf die Wahl der Zustände dieser 4 Bits ein.

```
132 void UART_init(void)
133 {
134     //Baud Rate and EUSART Mode
135     SPBRGH = 0b00000000; // Baud Rate 57600 > SPBRGH/L = 0b000000001000100
136     SPBRGL = 0b01000100;
137
138     TXSTAbits.BRGH = 1; // High Baud Rate Select Bit = 1 > High Baud Rate
139     BAUDCONbits.BRG16 = 1; // 16bit Baud Rate Generator bit = 1 > 16bit Mode
140     TXSTAbits.SYNC = 0; // EUSART Mode Select bit = 0 > Asynchronous Mode
141     RCSTAbits.SPEN = 1; // Serial Port Enable bit
142     TXSTAbits.TXEN = 1;
143     RCSTAbits.CREN = 1;
144
145     // Interrupts
146     INTCONbits.GIE = 1;
147     INTCONbits.PEIE = 1;
148     PIRbits.TXIE = 0;
149     PIRbits.RCIE = 1;
150
151 }
```

Abbildung 127: UART-Initialisierung



4.3.6.4 Interrupt-basierter UART-Treiber

Ein Treiber ist eine Softwarekomponente, die es dem Prozessorkern erlaubt, mit einem bestimmten Peripheriemodul, in diesem Fall mit der UART-Schnittstelle zu kommunizieren. Der Treiber übernimmt alle modulabhängigen Einstellungen und Vorkehrungen und stellt dem Hauptprogramm verschiedene Funktionen zur Verfügung, um mit dem Modul zu interagieren und um dessen Funktionen zu nutzen.

Die Eigenschaft, dass dieser Treiber Interrupt-basiert ist, erlaubt es dem Hauptprogramm die empfangenen Daten genau dann auszulesen, wenn sie benötigt werden, auch wenn diese bereits einige Zeit früher empfangen wurden. Das Empfangen und Senden passiert im Hintergrund und völlig unabhängig vom Hauptprogramm. Dadurch kommt es weder zu Verzögerungen noch zu Komplikationen während beider Vorgänge. Die Trennung des RX- und TX-Moduls gilt auch für die *Interrupts*.

4.3.6.4.1 TX-Treiber

Um die Unabhängigkeit des Treibers zu gewährleisten, muss ein Buffer für die zu sendenden Daten angelegt werden. Dafür wird eine Art FIFO-Register (siehe Kapitel 4.3.6.2.2) als *Array* angelegt. Wenn das Hauptprogramm einen Wert senden will, wird dieser vorne in das *Array* geschrieben. Sobald die Hardware bereit ist diesen zu senden, wird von hinten aus dem *Array* ausgelesen und gesendet. So einen Speicher nennt man Ring-Buffer.

4.3.6.4.1.1 Ring-Buffer

Ein Ring-Buffer besteht aus drei Komponenten. Einem *Head* (Kopf), einem *Tail* (Schwanz) und dem *Array* selbst. Das *Array* hat eine bestimmte Größe, die Buffer-Größe. Der *Head* und *Tail* sind *Integer*, deren Größe beliebig gewählt werden kann.

Wie der Name vermuten lässt, kann ein Ring-Buffer als ein kreisförmiger Speicher verstanden werden. Ein Element wird einmal in diesen Buffer geschrieben und einmal herausgelesen. Nachdem so ein Buffer initialisiert wurde, zeigen sowohl der *Head* als auch der *Tail* auf das Element 0. Der *Tail* zeigt immer auf das nächste leere Element, welches beschrieben werden kann. Der *Head* zeigt auf das Element, welches sich am längsten im Buffer befindet und als nächstes ausgelesen wird. Durch die beiden Variablen wird ermöglicht, dass der Buffer unterschiedlich viele Daten zu unterschiedlichen Zeitpunkten enthalten kann. Um festzustellen wie viele Daten gehalten werden, wird die Differenz von *Tail* zu *Head* gebildet. Entspricht diese Differenz dem Wert 0, so ist der Buffer leer. Daten können also geschrieben aber nicht gelesen werden, da keine enthalten sind. Ist die Differenz gleich der Buffer-Größe, ist dieser voll. Daten können gelesen werden, doch es können keine weiteren hineingeschrieben werden. Abbildung 128 zeigt einen neu initialisierten Buffer. *Head* und *Tail* sind beide 0, somit ist die Differenz auch 0 und der Buffer ist leer. Es können Daten an der Stelle 0 geschrieben werden, jedoch nicht gelesen.

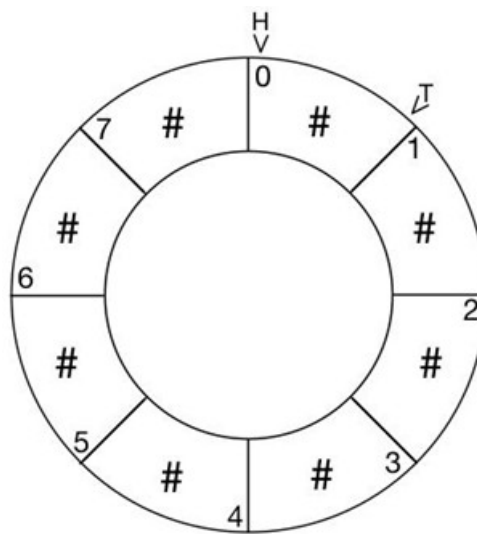


Abbildung 128: Ring-Buffer leer

Als Beispiel wird ein Buffer mit der Größe 8 mal 8Bit Integer definiert. Der *Head* und *Tail* sind ebenfalls 8Bit integer und werden mit dem Wert 0 initialisiert. Zusätzlich werden 3 Makros definiert, die berechnen, wie voll der Buffer ist, ob er leer ist und ob er voll ist. Abbildung 129 zeigt eine solche Initialisierung.

```
40 #define Buffer_Size 8
41
42 volatile uint8_t Buffer[Buffer_Size];
43 volatile uint8_t Head = 0;
44 volatile uint8_t Tail = 0;
45 // Makros
46 #define Buffer_amount() ((uint8_t)(Tail - Head))
47 #define Buffer_Full() (Buffer_amount() == Buffer_Size)
48 #define Buffer_Empty() (Buffer_amount() == 0)
```

Abbildung 129: Ring-Buffer-Initialisierung

Um nun z.B. den Wert 10 zu speichern, wird dieser in das Array an der Stelle *Tail* Modulo Buffer-Größe geschrieben. Warum das Modulo wichtig ist, wird in den kommenden Absätzen erklärt. Nun steht an der Stelle 0 des Buffers der Wert 10. Der *Tail* zeigt aber immer noch auf diese Stelle, wenn man einen weiteren Wert hinzufügen möchte, würde 10 überschrieben werden. Dazu wird der *Tail* inkrementiert und zeigt nun auf das Element 1. Abbildung 130 zeigt den Buffer nach Durchführung dieser Prozesse.

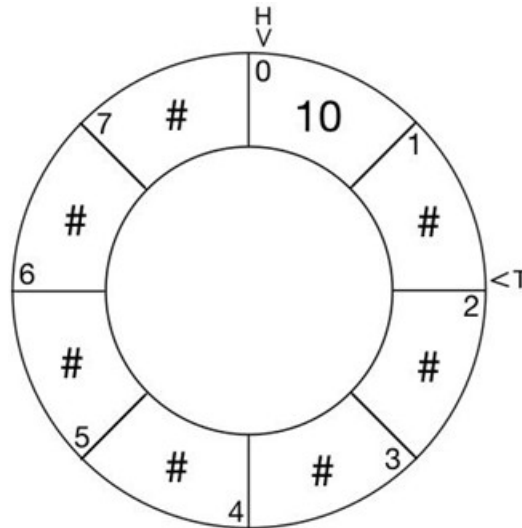


Abbildung 130: Ring-Buffer beschrieben

Das Auslesen eines Wertes erfolgt nach dem gleichen Prinzip, das Element an der Stelle *Head* Modulo Buffer-Größe wird ausgelesen und der *Head* wird inkrementiert. Im nächsten Schritt wird nun ein Wert ausgelesen und zwei weitere gespeichert.

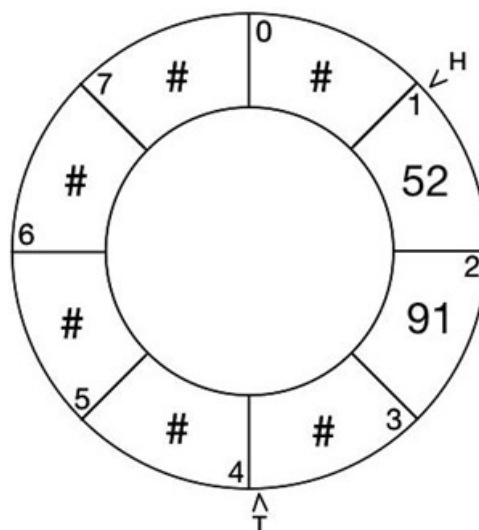


Abbildung 131: Ring-Buffer ausgelesen

Wie in Abbildung 131 zu sehen, wurde der Wert 10 ausgelesen und der *Head* inkrementiert. Ein ausgelesener Wert wird nicht gelöscht, sondern einfach überschrieben. Leere Elemente sind also nicht 0, sondern entsprechen einem Wert, der allerdings nicht mehr von Bedeutung ist und auf den auch nicht mehr zugegriffen wird. Die Werte 52 und 91 wurden in diesem Schritt an die Stellen 1 und 2 geschrieben, der *Tail* ist also um 2 größer als zuvor. Das Makro, das berechnet, wie voll der Buffer ist, würde nun 2 zurückgeben. Der Buffer ist weder voll noch leer und kann somit sowohl beschrieben als auch ausgelesen werden.

Ist der Buffer voll, kann im Gegensatz zu wenn er leer ist, nur noch ausgelesen werden. Das Makro, das die Differenz bildet, gibt in diesem Fall 8, also die Buffer-Größe zurück.

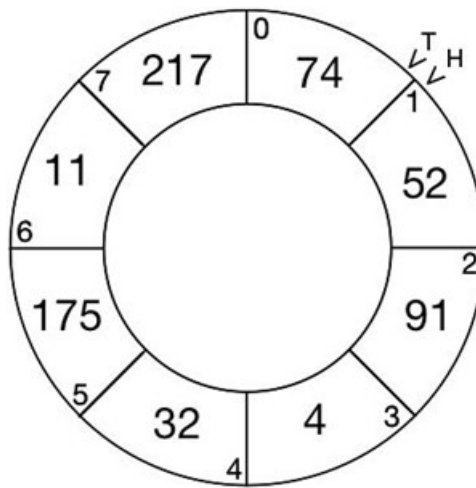
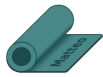


Abbildung 132: Ring-Buffer voll

Bevor aus dem Buffer gelesen wird, wird überprüft, ob dieser nicht leer ist, da sonst nichts gelesen werden kann. Ebenso wenn der Buffer voll ist, darf nicht geschrieben werden. Abbildung 133 zeigt sowohl den Auslese- als auch den Beschreibungsvorgang.

```
53 // Beschreiben
54 if(!Buffer_Full()) // Ist platz um neue Daten zu speichern?
55 {
56     Buffer[Tail % Buffer_Size] = Data; // Daten speichern
57     Tail++;                          // Tail inkrementieren
58 }
59
60 // Auslesen
61 if(!Buffer_Empty()) // Sind Daten zum Auslesen enthalten?
62 {
63     Data = Buffer[Head % Buffer_Size]; // Daten auslesen
64     Head++;                          // Head inkrementieren
65 }
```

Abbildung 133: Ring-Buffer auslesen und beschreiben



Wie bereits erwähnt, kann nicht direkt an der Stelle *Head* ausgelesen werden, sondern an der Stelle *Head* Modulo Buffer-Größe. Dies ist auch in Abbildung 133 zu sehen. Um den Code effizienter und schneller zu gestalten, werden die beiden Zeiger einmal initialisiert und nur inkrementiert. Ihnen werden weder Werte abgezogen, noch werden ihnen neue Werte zugewiesen. In Abbildung 132 hat der *Tail* somit nicht den Wert 0, sondern 8. 8 Modulo Buffer-Größe bzw. ebenfalls 8 ergibt 0. Es muss somit nicht überprüft werden welchen Wert die Zeiger haben, wodurch Rechenleistung eingespart werden kann.

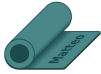
Daraus ergibt sich scheinbar ein weiteres Problem, welches allerdings durch einen kleinen Zusatz bei der Makro-Definierung gelöst werden kann. Wie in Abbildung 129 zu sehen, wird bei der Differenzbildung festgelegt, dass es sich um eine sogenannte *unsigned operation* handelt. Während einer Subtraktion werden zwei 8bit *unsigned integer* zu normalen *signed integers promoted*. Wäre der *Tail* beispielsweise wieder bei 2 nachdem er 258-mal inkrementiert wurde, und der *Head* erst bei 254, so würde bei einer normalen Subtraktion der Wert -252 herauskommen. Da der Buffer keine negative Anzahl an Daten enthalten kann, wird die Rechnung als *unsigned operation* festgelegt. Das Ergebnis entspricht nun dem Wert 4.

Der Buffer ist nun gegen jegliche Eventualitäten gewappnet, und kann ressourcenschonend im Hintergrund verwendet werden

4.3.6.4.1.2 TX-Interrupts

Wie in Kapitel 4.3.6.2.1 erwähnt, löst das TX-Modul einen *Interrupt* aus, indem es die TXIF-Flagge setzt, sobald das TXREG bereit ist einen Wert zu übernehmen. Dies darf nur passieren, wenn Daten zum Senden bereit liegen. Ansonsten kommt es zu einem sogenannten *Interrupt-Storm*. Um zu verhindern, dass das Modul jeden Instruction-Cycle, aufgrund eines leeren TXREG den *Interrupt* triggert, wird ihm die Erlaubnis mit nullsetzen des TXIE-Bits (TX Interrupt Enable) genommen. Wird nun die Funktion zum Senden eines einzelnen Bytes (siehe 4.3.6.4.1.3) aufgerufen, wird das TXIE-Bit wieder auf 1 gesetzt. Daraufhin löst das Modul einen *Interrupt* aus, vorausgesetzt das TXREG ist leer. In der ISR wird überprüft, ob es sich um einen TX-*Interrupt* handelt indem geprüft wird ob die TXIF als auch das TXIE-Bit gesetzt sind.

Wenn nun ein *Interrupt* durch das Modul ausgelöst wird, wird festgestellt, ob der Ring-Buffer einen Wert enthält. Ist das der Fall, wird dieser Wert ausgelesen und in das TXREG übertragen. Sobald dieses Register wieder frei ist, wird erneut ein *Interrupt* ausgelöst. Wenn der Buffer einen weiteren Wert enthält, wird der Vorgang wiederholt. Sollte der Buffer leer sein, wird das TXIE-Bit wieder auf 0 gesetzt und der Übertragungsvorgang wird abgeschlossen. Will man z.B. 10 Zeichen senden, so wird die ISR insgesamt 11-mal von diesem Modul aufgerufen, wobei das 11. Mal ein Kontrolldurchlauf ist, der zukünftige *Interrupts* verhindert.



4.3.6.4.1.3 Sendefunktion

Es wurde eine Funktion benötigt, mit der ein einziges Zeichen, ein Byte gesendet werden kann. Sie übernimmt die zu sendenden Daten als 8Bit-Integer. Bevor dieser in den TX-Ring-Buffer geschrieben wird, wird überprüft, ob der Buffer voll ist und gegebenenfalls gewartet, bis er es nicht mehr ist. Nachdem der Wert in den Buffer übertragen wurde, wird dem Modul erlaubt, *Interrupts* auszulösen, um den Sendevorgang zu starten.

4.3.6.4.2 RX-Treiber

Der RX-Treiber funktioniert auf dieselbe Art und Weise wie der TX-Treiber. Er ist ebenfalls *Interrupt* basiert und verwendet einen eigenen Ring-Buffer um empfangene Daten so lange zu speichern, bis diese im Hauptprogramm verarbeitet werden können.

4.3.6.4.2.1 RX-Interrupts

Das RX-Modul löst einen *Interrupt* aus, sobald ein Wert empfangen wurde und dieser vollständig in das RCREG Register übertragen wurde. Wurde ein *Interrupt* ausgelöst prüft die ISR (*Interrupt Service Routine*), ob die RCIF sowie das RCIE-Bit gesetzt ist. Dadurch kann festgestellt werden, ob der RX diesen *Interrupt* verursacht hat. Wenn das der Fall ist, wird solange der Buffer genügend Platz hat, der empfangene Wert gespeichert. Ist der Buffer voll, gehen die Daten verloren. Die ISR muss so kurz wie möglich sein, um das Hauptprogramm nicht zu lange aufzuhalten. Es kann nicht gewartet werden, bis Platz frei wird, da das Hauptprogramm von der ISR unterbrochen wird und somit keinen Wert aus dem Buffer auslesen kann. Es wäre also überhaupt nicht möglich diese Daten zu speichern. Allerdings werden während des Regelbetriebs nie genug Daten vom ESP32 gesendet, um zu einem *Overflow* des Buffers zu führen. Das Verhalten bei einem solchen Ereignis muss definiert sein, um den Verlust von weiteren Daten zu verhindern.

In der ISR ist ebenfalls geregelt, dass der RX neugestartet wird, wenn ein *Overrun-Error* ausgelöst wurde. Dazu wird das CREN-Bit auf 0 und anschließend wieder auf 1 gesetzt.

4.3.6.5 Funktion zum Senden mehrerer Bytes

Wie in Kapitel 4.3.6.4.1.3 beschrieben, wurde eine Funktion angelegt, um einen 8Bit-Integer zu senden. Um nun beliebig viele Daten mit einem Funktionsaufruf senden zu können, wurde eine Funktion definiert, die einen *Pointer* zu einer Speicheradresse übernimmt. Die Funktion liest den Wert, auf den dieser *Pointer* zeigt, aus und inkrementiert den *Pointer*. Anschließend wird die Treiber-Funktion aufgerufen und der Wert wird als Parameter mitgegeben und wird somit über die Schnittstelle gesendet. Darauf wird der nächste Wert, auf den der *Pointer* nun zeigt, ausgelesen und

der Vorgang beginnt von vorne. Dies wiederholt sich so lange, bis der *Pointer* auf eine so genannte Nullterminierung zeigt.

Wird diese Funktion aufgerufen, muss beachtet werden, dass das übergebene Datenpaket mit einer Nullterminierung endet. Ist dies nicht der Fall, kann ein undefiniertes Verhalten auftreten. Diese Funktion wird ausschließlich für das Senden von vordefinierten *Arrays* oder String-Literalen verwendet.

Ein String-Literal ist eine von Anführungszeichen begrenzte Zeichenabfolge die direkt in das Programm geschrieben wird. Sie ist automatisch nullterminiert. Ein Funktionsaufruf mit einem String-Literal sieht wie folgt aus:

```
UART_Transmit_Word("Hello World!");
```

Abbildung 134: String-Literal

4.3.7 Hilfsfunktionen

Als Hilfsfunktionen werden Funktionen bezeichnet, die bestimmte kleine Aufgaben erledigen wie z.B. das Berechnen der Wurzel einer Zahl oder das Umwandeln eines Datentyps. Für die Firmware wurden drei solcher Funktionen geschrieben.

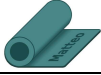
4.3.7.1 Map-Funktion

Eine Map-Funktion ist eine sehr weit verbreitete Methode um einen Wert, der innerhalb eines bestimmten Wertebereichs liegt, auf einen anderen Wertebereich abzubilden. Dieser Vorgang wird *mapping* genannt. Die Funktion übernimmt 5 Parameter, den Ausgangswert (AWert) selbst, die obere (AGO) und untere (AGU) Grenze des Ausgangswertebereiches sowie die obere (ZGO) und untere (ZGU) Grenze des Zielwertebereiches. Die Berechnung erfolgt in 4 Schritten. Zuerst wird die Differenz des Wertes zu seiner unteren Grenze gebildet. Diese wird anschließend durch die Ausgangswertebereichsbreite dividiert und mit der Zielwertebereichsbreite multipliziert. Zuletzt muss noch die untere Grenze des Zielwertebereichs addiert werden und das Ergebnis entspricht dem neuen *gemappten* Wert (ZWert).

$$\frac{AWert - AGU}{AGO - AGU} * (ZGO - ZGU) + ZGU = ZWert$$

Als Beispiel wird, angenommen, ein Wert 10 aus dem Bereich 5 bis 25 soll auf den Bereich 100 bis 300 abgebildet werden. Der *gemappte* Wert entspricht 150.

$$\frac{10 - 5}{25 - 5} * (300 - 100) + 100 = 150$$



4.3.7.2 Rundungsfunktion

Es wird eine Funktion benötigt, die einen Integer auf ein Vielfaches von 5 auf- oder abrundet. Die Einerstelle ist ausschlaggebend wie gerundet wird. 1 und 2 werden auf 0 abgerundet, 3 bis inklusive 7 werden auf 5 gerundet und 8 und 9 werden auf 10 aufgerundet. Diese Funktion wird verwendet, um die momentane Belastung des rechten Fußes so aufzubereiten, dass sie an den ESP32 gesendet werden kann. Sie übernimmt den zu rundenden 8Bit Integer als Parameter und gibt den gerundeten Wert als 8Bit *Integer* zurück.

4.3.7.3 16Bit Integer zu ASCII-String

Diese Funktion wandelt einen 16Bit *Integer* in einen maximal 5-stelligen String um. Dies ist vor allem für Debugging-Zwecke von Vorteil. In H-Term lassen sich Daten zwar sowohl als ASCII-Zeichen als auch als Dezimalzahlen darstellen, allerdings erschwert das die Lesbarkeit, vor allem wenn Daten beide Formate enthalten. Daher ist die Funktion perfekt, um reine Strings auszugeben, die einfach zu lesen und interpretieren sind. Der Maximalwert eines 16Bit *Integers* ist 65535 und somit wird ein 6-stelliger String benötigt. Die letzte Stelle ist eine Nullterminierung, um zu zeigen, dass dies das Ende des Strings ist. Die Funktion übernimmt den 16Bit *Integer* der zu wandeln ist, sowie einen Pointer zu einer Speicheradresse mit genügend Platz für 6 Zeichen. Dazu wird einfach ein temporärer Array mit der Größe 6 mal 8Bit *Integer*, vor Funktionsaufruf angelegt, welcher nach Verwendung des Strings hinfällig wird.

4.3.8 Hauptprogramm

Das Hauptprogramm regelt unter Verwendung aller erwähnten Funktionen, die allgemeine Logik der Matte. Die Funktionsweise wird anhand des Flussdiagramms in Abbildung 135 erklärt.

Nach Programmstart werden alle notwendigen Module und Funktionen des PICs initialisiert. Das Programm hat kein Ende. Es läuft, solange die Matte eingeschaltet ist.

Zu Beginn eines Durchlaufs wird die Empfangsfunktion (siehe Kapitel 4.4.1.1.2) ausgeführt. Dadurch werden Befehle ausgeführt und eventuelle Änderungen in den Statusvariablen vorgenommen.

Je nach Zustand dieser Variablen werden bestimmte Prozesse ausgeführt. Darunter fällt z.B. das erneute Senden eines Befehls nach einem Übertragungsfehler.

Ist die korrekte Kommunikation mit dem ESP32 und allen Peripheriemodulen sichergestellt, wird auf einen Befehl gewartet, der die Übung startet.

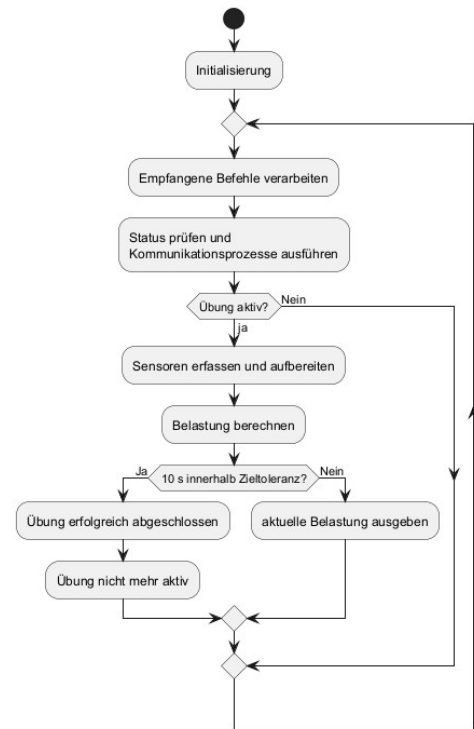


Abbildung 135: Hauptprogramm

Läuft eine Übung, müssen die Sensordaten erfasst und ausgewertet werden. Basierend auf diesen Daten, wird die momentane Belastung berechnet. Wenn ein bestimmtes Ziel eingestellt wurde, muss dieses für 10 s gehalten werden, um die Übung abzuschließen. Eine Zieltoleranz von $\pm 5\%$ erlaubt Schwankungen. Wird diese Toleranz verlassen, wird die Zeit zurückgesetzt und das Ziel muss erneut getroffen werden.

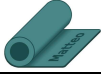
Solange die Übung läuft, wird alle 4 s die Belastung ausgegeben. Dies gilt auch für eine Übung ohne Ziel. Wird diese Einstellung verwendet, wird eine Übung nur durch den Stopp-Befehl beendet.

4.4 Kommunikation zwischen den Mikrocontrollern

Die Matte erfordert ein enges und gut geregeltes Zusammenspiel der 3 Microcontroller. In den folgenden Kapiteln wird näher darauf eingegangen, wie die einzelnen Komponenten untereinander kommunizieren.

4.4.1 PIC16F1827 zu ESP32

Die Kommunikation erfolgt auf Basis eines Übertragungsprotokolls. Dieses Protokoll wurde entwickelt, um den Anforderungen des Projekts zu entsprechen. Kommuniziert wird mithilfe verschiedener Befehle, die aus einer Reihe von ASCII-Zeichen bestehen. Jeder Befehl wird mit einem „>“ angeführt. Dieses Zeichen markiert den Beginn einer Befehlsübertragung. Darauf folgen je nach Befehl zwischen einem



und vier weitere Zeichen. Beendet wird ein Befehl mit einem so genannten LF (Line Feed). Ein LF entspricht einem Dezimalwert von 10 und dem ASCII-Zeichen „\n“.

Jeder empfangene Befehl wird mit einem „<“ vom Empfänger bestätigt. Dadurch erkennt der Sender, ob die Übertragung erfolgreich abgeschlossen wurde.

Folgende Befehle wurden verwendet, um Information vom PIC and den ESP32 zu senden:

- >GR ... *Goal Reached*, Übung erfüllt.
- >DX ... Daten der aktuellen Belastung, X entspricht dem Wert der Belastung.
- >E ... Fehler bei der Übertragung
- >P ... Ping.
- >PA ... Ping-Antwort.

Folgende Befehle wurden verwendet, um Informationen vom ESP32 an den PIC zu senden:

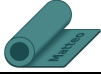
- >KFX ... Ziel für den rechten Fuß. X entspricht einem Wert zwischen 0 und 100 in 10er Schritten.
- >KN ... Kein Ziel.
- >KON ... Start der Übung.
- >KOFF ... Stopp der Übung.
- >E ... Fehler bei der Übertragung.
- >P ... Ping.
- >PA ... Ping-Antwort.

Genauere Informationen zu den Befehlen und wann diese verwendet werden, wird in den folgenden Kapiteln erklärt.

4.4.1.1 PIC-seitig

Der PIC erhält vom ESP32 die benötigten Einstellungen, die durch die Sprachsteuerung festgelegt wurden und sendet Informationen, die mithilfe der Sprachausgabe ausgegeben werden sollen.

Wenn einer der Microcontroller eine Zeichenfolge empfängt, die dem Format eines Befehls entspricht, sie beginnt mit einem „>“ und endet mit „\n“, ist aber nicht in der Liste der Befehle enthalten oder überschreitet die maximale Befehlslänge, meldet dieser einen Empfangsfehler. Durch Störungen können Daten verfälscht werden. Wird so ein Fehlerbefehl empfangen wird der zuletzt fehlerhaft gesendete Befehl noch einmal gesendet. Es muss also immer der letzte Befehl gespeichert werden.



Weiters wird, wenn ein Befehl mit dem korrekten Format erkannt wird, an den Sender zurückgemeldet, dass die Übertragung erfolgreich war. Wenn eine Störung auftritt, die auch das Format des Befehls verfälscht, kann dieser nicht erkannt werden und es wird nicht bestätigt. Wenn ein ausgehender Befehl nicht bestätigt wird, nimmt der PIC an, dass der ESP32 nicht mehr verfügbar ist und startet den Ping-Prozess.

Ein Ping ist ein Befehl, der so lange gesendet wird, bis eine Ping-Antwort empfangen wird. Der Ping muss somit sowohl mit dem „<“ Zeichen bestätigt, als auch mit einem eigenen Befehl beantwortet werden. Diese Antwort wird ebenfalls bestätigt. Es wird also jede notwendige Funktion des Protokolls überprüft. Sobald festgestellt wurde, dass alles funktioniert, wird der Normalbetrieb fortgesetzt, indem der letzte fehlerhafte Befehl erneut gesendet wird.

4.4.1.1.1 Senden

Für das Senden wurde eine Funktion verwendet, die den zuletzt gesendeten Wert speichert, wartet, bis ein Befehl bestätigt wird und den Ping-Prozess regelt.

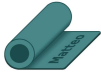
Für die Bestätigung wurde eine globale Variable angelegt. Diese wird von der Funktion für bis zu 2 Sekunden lang abgefragt. Wenn der RX einen *Interrupt* auslöst, wird direkt in der ISR erkannt, ob es sich um ein „<“ Zeichen handelt und gegebenenfalls die Variable auf 1 gesetzt. Wurde nach 2 Sekunden keine Bestätigung erkannt beginnt der Ping-Prozess. Andernfalls wird der Übertragungsprozess als erfolgreich angesehen und die Funktion wird beendet.

Während dem Ping-Prozess wird dem ESP32 jede Sekunde ein Ping zugesendet. Wenn die Funktion, die Befehle verarbeitet (siehe folgenden Kapitel) eine Ping-Antwort erkennt, wird eine Variable auf 1 gesetzt, die angibt, ob der ESP32 verfügbar ist. Sobald diese Variable nicht mehr 0 ist, wird die Funktion beendet.

4.4.1.1.2 Empfangen

Die Funktion, die den Ablauf der Befehlserkennung regelt, überprüft nach Aufruf, ob sich Daten im RX-Buffer befinden. Wenn das der Fall ist, werden so lange Werte ausgelesen, bis ein „>“ gefunden wird. Dieses Zeichen, gemeinsam mit allen darauffolgenden, werden in ein temporäres *Array* gespeichert. Sobald das LF erkannt wird, ist der Befehl vollständig und die Funktion bestätigt diesen. Ist die maximale Befehlslänge erreicht und es wurde kein LF erkannt, ist ein Fehler aufgetreten und ein Error wird gemeldet.

Nach dem Einlesen und Abspeichern eines korrekt formatierten Befehls geht die Funktion zur Erkennung dessen über. Dazu wurden alle Befehle als globale Vergleichsvariablen angelegt. Mithilfe der *memcmp*-Funktion (*Memory Compare*) wird der empfangene Befehl mit jedem möglichen verglichen. Wird keine Übereinstim-



mung gefunden, löst das einen Empfangsfehler aus. Je nach erkanntem Befehl werden bestimmte Konfigurationsvariablen verändert. Diese verändern zum Beispiel wie Belastungsdaten interpretiert werden sollen, oder ob eine Übung gestartet oder gestoppt wurde.

`memcmp()` ist eine Funktion aus der „String.h“-Bibliothek und sie vergleicht zwei Speicherabschnitte miteinander. Sie übernimmt jeweils einen Zeiger auf das erste Element der Abschnitte sowie die Länge dieser Abschnitte in Bytes. Der Rückgabewert entspricht 1, wenn der erste Abschnitt größer ist, -1, wenn der er kleiner ist und 0, wenn beide Abschnitte gleich sind. Für diesen Verwendungszweck reicht die letzte Funktion. Wenn der Rückgabewert ungleich 0 ist, sind es nicht dieselben Befehle.

4.4.1.1.3 Befehlsverarbeitung

Dieses Kapitel beschreibt die Befehle und welche Auswirkungen diese auf das Hauptprogramm haben.

>KFX

Dieser Befehl gibt das Belastungsziel an. Der Wert wird als ASCII-Zeichen empfangen. Um es in Prozent umzuwandeln, wird die Differenz zu dem „0“ Zeichen gebildet und mit 10 multipliziert. Wird eine 3 im ASCII-Format empfangen ergibt das einen Dezimalwert von 30. Die Zielbelastung beträgt somit 30%. Diese ist immer auf den rechten Fuß bezogen. Das Ziel wird in einer globalen Variable gespeichert. Ebenso wird eingestellt, dass es sich um eine Übung mit Belastungsziel handelt, indem die Variable *Target* auf 1 gesetzt wird.

>KN

Soll eine Übung ohne Ziel durchgeführt werden, während der nur die aktuelle Belastung ausgegeben werden soll, ohne diese zu bewerten, muss dieser Befehl empfangen werden. die Variable *Target* wird auf 0 gesetzt.

>KON und >KOFF

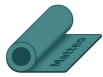
Diese beiden Befehle bestimmen den Status einer Übung. Läuft diese oder soll sie gestoppt werden. „>KON“ setzt die Statusvariable auf 1 und die Übung wird gestartet und „>KOFF“ beendet sie wieder durch Nullsetzen.

>E

Wenn ein *Error* empfangen wird, muss eine Variable gesetzt werden, um dem Hauptprogramm mitzuteilen, dass der letzte Befehl erneut gesendet werden soll.

>P und >PA

Ein Ping wird mit einer Ping-Antwort erwidert und die empfangene Ping-Antwort bestätigt einen funktionierenden Kommunikationsprozess.



4.4.1.2 ESP-seitig

Im folgenden Abschnitt wird ESP32-seitig die Kommunikation mittels UART beschrieben.

4.4.1.2.1 Senden

Für das Übertragen eines Befehls an den PIC gibt es eine Funktion, die den Befehl über I²C (siehe Abschnitt 4.4.2.1) sendet und zu Debug-Zwecken auch am Serial Monitor (siehe Abschnitt 4.5.2) ausgibt. Jeder gesendete Befehl muss vom PIC mit einem Acknowledge bestätigt werden, damit der ESP32 sicherstellen kann, dass der Befehl richtig angekommen ist. Falls der Befehl innerhalb von 2 Sekunden nicht bestätigt wird, wird die Ping-Funktion aufgerufen. Diese sendet jede Sekunde einen Ping an den PIC, bis eine Ping-Antwort erhalten wird. So wird festgestellt, ob der PIC noch erreichbar ist. Wurde die Ping-Antwort erhalten, wird der ursprüngliche Befehl noch einmal gesendet und wieder auf ein Acknowledge gewartet. Wird ein Acknowledge empfangen, war das Senden des Befehls erfolgreich und die Funktion wird verlassen.

4.4.1.2.2 Empfangen

Die eingelesenen Daten werden in ein Array gespeichert. Zuerst wird überprüft, ob das erste Zeichen ein '`>`' ist. Wenn nicht, bedeutet das, dass kein Befehl gesendet wurde, sondern Daten zu Debug-Zwecken. Diese können an der Schnittstelle über H-Term (siehe Abschnitt 4.3.2) ausgelesen werden. In diesem Fall werden die Daten ignoriert.

Jeder Befehl wird mit einem LF (Line feed) beendet. Strings enden allerdings mit einer Nullterminierung, weshalb der LF gegen eine 0 ausgetauscht wird. Eine Nullterminierung bedeutet, dass am Ende des Strings das Zeichen mit dem Wert 0 steht (geschrieben als '`\0`'). Sie signalisiert den Funktionen das Ende eines Strings, selbst wenn die Größe des Arrays größer ist als der eigentliche String. So muss auch keine eigene Variable angelegt werden, in der die Länge explizit gespeichert wird.

Danach wird überprüft, ob der eingelesene String ein gültiger Befehl ist. Mithilfe der Funktion `strcmp()`, näher beschrieben in Abschnitt 4.6.3.3, wird das eingelesene Array mit allen gültigen Befehlen verglichen. Der jeweilige Befehl wird anschließend interpretiert und auditiv ausgegeben. Falls der eingelesene Befehl ungültig ist, wird ein Error an den PIC gesendet, der die Daten dann noch einmal überträgt. So können Fehler in der Übertragung abgefangen werden. Zum Schluss wird das gesamte Array wieder auf 0 gesetzt.

4.4.2 Arduino Nicla Voice zu ESP32

Der Arduino Nicla Voice und der ESP32 kommunizieren über eine serielle I²C Verbindung.

4.4.2.1 I²C

I²C steht für *Inter-Integrated Circuit* und ist eine synchrone, serielle Schnittstelle für die Datenübertragung zwischen mindestens einem Master und einem oder mehreren Slaves. Es ist ein Zweidraht-Bus, der eine Daten- (SDA, Shift Data) und eine Taktleitung (SCL, Shift Clock) besitzt. Die Übertragung erfolgt halbduplex, das heißt, es kann gesendet und empfangen werden, aber nicht gleichzeitig.

Vor der Übertragung sind beide Leitungen auf *HIGH*, das entspricht 3,3 V. Wenn der Master eine Übertragung starten will, zieht er die SDA-Leitung auf Masse (*LOW*), während die SCL-Leitung auf *HIGH* bleibt. Das ist die Start-Bedingung für eine Übertragung, wie in Abbildung 136 dargestellt.

Ein Slave kann nie selbstständig Daten senden. Er muss immer vom Master angesprochen werden. Das geschieht, indem der Master nach der Startbedingung die 7-Bit-Adresse des Slaves sendet, mit dem er kommunizieren möchte und ihm anschließend mit dem *read/write* Bit (R/W) mitteilt, ob er Daten senden oder empfangen möchte. Wird R/W auf *LOW* gesetzt, bedeutet das, dass der Master Daten senden will. Wird es *HIGH* gesetzt, will der Master Daten vom Slave empfangen.

Jedes gesendete Datenbyte wird mit einem Acknowledge-Bit quittiert. Das geschieht, indem der Empfänger die SDA-Leitung auf Masse zieht und der Sender das im neunten SCL-Takt überprüft. Erst wenn die Adresse und R/W bestätigt wurde werden Daten gesendet.

Die Übertragung endet, wenn die SDA-Leitung von *LOW* auf *HIGH* wechselt, während sich die SCL-Leitung auf *HIGH* befindet. Das wird Stopp-Bedingung genannt.

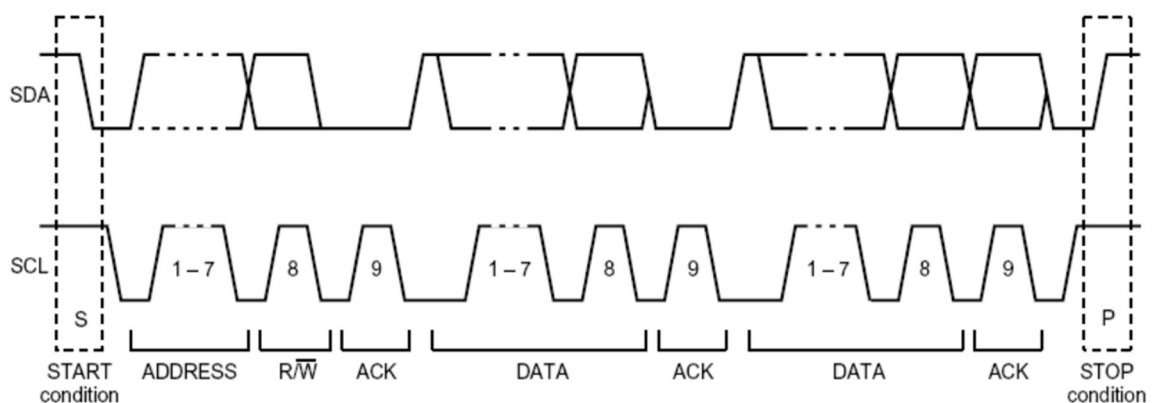
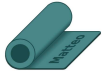


Abbildung 136: I2C Übertragung [26]



4.4.2.2 Übertragungsprotokoll

Zur Übertragung der Daten zwischen dem ESP32 und dem Arduino wurde ein eigenes Datenübertragungsprotokoll erstellt. Der Arduino ist ausschließlich Sender und der ESP32 empfängt.

Jeder Befehl besteht aus einem Buchstaben und einer Zahl. Der Buchstabe gibt die Art des Befehls und die Zahl den Wert an. Es gibt sechs verschiedene Befehlsarten. Die Zahlen bedeuten je nach Art etwas anderes. Bei manchen Befehlen hat die Zahl keine Bedeutung. Sie ist in diesem Fall nur vorhanden, damit alle Befehle gleich aufgebaut sind und damit es leichter ist, neue Befehle hinzuzufügen, falls welche hinzukommen. Bei diesen Befehlen wird standardmäßig 0 gesendet.

- P... Prozent
 - P 1-10... 10 bis 100 % in 10er Schritten
- S... Start/Stop
 - S0... Stopp
 - S1... Start
- H... Hochfahren
 - Zahl nicht von Relevanz
- L... Lautstärke
 - L0... Leiser
 - L1... Lauter
- Z... kein Ziel
 - Zahl nicht von Relevanz
- B... Bein
 - B0... rechtes Bein
 - B1... linkes Bein

4.4.2.3 ESP32-seitig

Der ESP32 arbeitet als Slave. Wenn er Daten empfängt, wird eine Interrupt-Funktion aufgerufen, die die Daten einliest und in zwei Variablen abspeichert.

In der Interrupt-Funktion wird überprüft, ob Daten zum Einlesen vorhanden sind und ob seit dem Beginn der letzten Sprachausgabe über 2 Sekunden vergangen sind.

Wenn weniger als 2 Sekunden vergangen sind, werden die erkannten Befehle ignoriert. Das liegt daran, dass die Spracheingabe die Sprachausgabe erkennen und diese als Befehl interpretieren würde. Diesen Befehl würde die Sprachausgabe dann wieder zur Bestätigung ausgeben und er würde wieder erkannt werden. So würde eine Endlosschleife entstehen. Wenn mehr als 2 Sekunden vergangen sind, werden die Daten

in zwei Variablen eingelesen und die Befehlsart an eine Funktion weitergegeben, die das Zeichen in einen *enum* Datentyp (siehe Abschnitt 4.5.2) wandelt. Der Wert wird innerhalb der einzelnen *Cases* weiterverarbeitet.

Abbildung 137 zeigt diese Funktion. Sie vergleicht mithilfe eines Switches das Zeichen mit mehreren *Cases* und gibt anschließend den zugehörigen Datentyp zurück. Falls ein unbekanntes Zeichen empfangen wurde, gibt es den *Default*. Er wird zurückgegeben, wenn kein *Case* passt. Das könnte zum Beispiel bei einem Fehler in der Übertragung der Fall sein.

```
61  datatypes charToDatatype(char c) {  
62      switch (c) {  
63          case 'P': return Prozent;  
64          case 'S': return Start_Stopp;  
65          case 'B': return Bein;  
66          case 'L': return Lautstaerke;  
67          case 'Z': return kein_Ziel;  
68          case 'H': return Hochfahren;  
69          default: return None;  
70      }  
71  }
```

Abbildung 137: ESP32 Programm char zu Datatype

Interrupt-Routinen sollten eigentlich so kurz wie möglich sein und nur Daten einlesen oder Flags setzen. In dieser wird allerdings eine Berechnung für den Zeitvergleich durchgeführt und eine weitere Funktion aufgerufen. Da das allerdings nie zu Problemen geführt hat, wurde sie vorerst so belassen.

4.4.2.4 Arduino-seitig

Der Arduino ist der Master des Übertragungssystems. Wenn ein Label erkannt wurde, wird mit `strcmp()` das Label durch Vergleiche ermittelt und anschließend der entsprechende Befehl an den ESP32 gesendet. Dafür werden zuerst die Interrupts ausgeschaltet, da I²C ein genaues Timing benötigt und ein Interrupt während einer Übertragung dazu führen kann, dass der Takt zu lange pausiert wird oder ein Acknowledge verpasst wird. Danach werden die Startbedingung und die Adresse gesendet, dann die Daten und die Stopp-Bedingung. Zum Schluss müssen die Interrupts wieder eingeschaltet werden, da sie für die interne Übertragung der Labels benötigt werden. Für die Datenübertragung wird die Bibliothek `Wire.h` verwendet (siehe Abschnitt 4.5.5.4).

```
void sendToESP(char type, int value) {  
    NDP.noInterrupts(); //Interrupts ausschalten  
    Wire.beginTransmission(9); // Übertragung starten  
    Wire.write(type); //Daten übertragen  
    Wire.write(value);  
    Wire.endTransmission(); //Stopp-Bedingung
```

Abbildung 138: I2C Übertragung

4.5 Sprachausgabe

Die Sprachausgabe gibt die aktuelle Belastung auditiv aus. Wenn per Spracheingabe eine Einstellung geändert wurde, wird diese ebenfalls ausgegeben, um zu bestätigen, dass die Änderung erfolgreich war.

4.5.1 User Interface

Wenn die Matte eingeschaltet wird, fährt der Arduino hoch. Anschließend befindet sich die Matte im Startmodus. Hier kann das gewünschte Belastungsziel eingestellt werden. 50 Prozent auf jedem Bein sind voreingestellt. Wenn die Übung anschließend gestartet wird, leuchtet die LED rot. Dann soll das Ziel erreicht werden. Mithilfe der Sprachausgabe wird dem Benutzer mitgeteilt, wie hoch die momentane die aktuelle Belastung des rechten Fußes ist und ob das Ziel erreicht wurde. Wenn das Ziel erreicht wurde, muss es mit einem Spielraum von $\pm 5\%$ 10 Sekunden gehalten werden, um die Übung abzuschließen. Danach beendet sich die Übung automatisch und die LED leuchtet grün. Anschließend befindet sich die Matte wieder im Startmodus.

Während eine Übung läuft, kann das Belastungsziel nicht geändert werden. Die Übung kann aber manuell gestoppt werden. Dann können auch neue Einstellungen getätigt werden. Die Lautstärke kann immer verändert werden.

Falls *Kein Ziel* eingestellt wurde, wird die aktuelle Belastung so lange ausgegeben, bis die Übung manuell beendet wird.

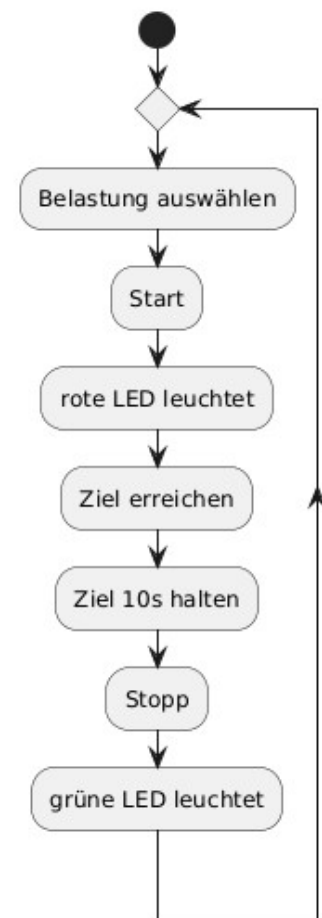
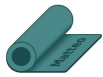


Abbildung 139: User Interface



Die Sprachausgabe erkennt 17 vorgefertigte Befehle.

Tabelle 15: Befehle

Befehl	Beschreibung
Linkes Bein / Rechtes Bein	Die darauffolgende Prozentzahl muss mit dem jeweiligen Bein erreicht werden
10 Prozent bis 100 Prozent in 10er-Schritten	Die jeweiligen Prozent definieren das Ziel
Lauter / Leiser	Verändert die Lautstärke der Sprachausgabe
Start	Startet die eingestellte Übung
Stopp	Beendet die eingestellte Übung
Kein Ziel	Messung ohne Zielsetzung

4.5.2 Theorie

Das Programm basiert auf einer Interrupt-Funktion und einer Switch-Case-Anweisung.

Arduino Programm

Ein Arduino Programm besteht grundsätzlich aus zwei Funktionen. Die `setup()`-Funktion wird zu Beginn einmal aufgerufen. Hier werden Pin-Konfigurationen durchgeführt und Schnittstellen initialisiert. Die `loop()`-Funktion wird dauerhaft wiederholt.

Serieller Monitor oder Serial Monitor

Der Serial Monitor ist ein Debugging-Tool der Arduino IDE. Es tauscht Daten zwischen dem Arduino und dem PC über eine UART-Schnittstelle aus. So können Variablenwerte und Programmabläufe verfolgt werden.

Um beispielsweise *Start* am Serial Monitor auszugeben wird folgende Codezeile benötigt:

```
Serial.println("Start");
```

Switch-Case-Statement

Ein Switch-Case-Statement vergleicht einen Wert mit mehreren *Case*-Marken, ähnlich wie if-Abfragen. Stimmt der Wert mit der *Case*-Marke überein, wird der dazugehörige Codeblock ausgeführt. Jeder dieser Codeblöcke muss mit einem *break* beendet werden, sonst wird der Code aus dem nächsten *Case* auch ausgeführt. Dieser Vergleich funktioniert mit allen ganzzahligen Datentypen (int, enum, char). Zusätzlich zu den *Cases* kann *Default* verwendet werden. Dieser wird dann ausgeführt, wenn kein anderer *Case* passt. Vorteile von Switch-Case gegenüber *if* sind, dass es meist übersichtlicher und schneller ist. Deshalb wird Switch-Case immer dann verwendet, wenn ein Wert mit vielen anderen Werten verglichen werden soll.

Ein Beispiel für eine einfache Switch-Case Anweisung sieht wie folgt aus:

```
int erkannter_Befehl = 2;

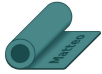
switch (erkannter_Befehl) {
    case 0:
        Serial.println("Start");
        break;
    case 1:
        Serial.println("Stopp");
        break;
    default:
        Serial.println("anderer Befehl");
        break;
}
```

Hier wird die Integer-Variable *erkannter_Befehl* mit den Zahlen 1 und 2 verglichen. Wenn die Variable gleich 1 ist, wird *Start* am Serial Monitor ausgegeben, bei 2 *Stopp* und bei jedem beliebigen anderen Wert *anderer Befehl*.

Enumeration (enum)

Eine Enumeration ist ein vom Benutzer definierter Datentyp. Er wird verwendet, um Integer-Konstanten sinnvolle Namen zuzuweisen, was ein Programm einfacher zu lesen und zu warten macht.

Im Beispiel von Switch-Case kann nicht auf den ersten Blick erkannt werden, welcher *Case* wann ausgeführt wird. Das wird vor allem in realen Anwendungen, in denen der Code der einzelnen *Cases* komplizierter wird, schnell unübersichtlich. Auch mit zunehmender Anzahl an *Cases* wird es immer schwieriger, einen solchen Code zu warten. Daher wird in diesem Zusammenhang oft eine Enumeration verwendet.



Für das Beispiel oben könnte diese wie folgt aussehen:

```
enum Befehle {Start, Stopp, Lauter, Leiser, ...};
```

Standardmäßig wird in diesem Beispiel *Start* der Wert 0 zugewiesen, *Stopp* der Wert 1 und alle nachfolgenden werden um 1 inkrementiert. Es können aber auch manuell Werte zugewiesen werden:

```
enum Befehle {Start = 5, Stopp = 9, Lauter, Leiser, ...};
```

Jetzt wird *Start* der Wert 5 zugewiesen, *Stopp* der Wert 9 und alle darauffolgenden werden ausgehend vom letzten zugewiesenen Wert um 1 inkrementiert. *Lauter* hätte somit den Wert 10 und *Leiser* den Wert 11.

Nachdem eine Enumeration definiert wurde, können Variablen mit dieser Definition angelegt werden:

```
Befehle erkannter_Befehl = Start;
```

Die zugehörige Switch-Case Anweisung würde vereinfacht so aussehen:

```
switch (erkannter_Befehl) {  
    case Start:  
        Serial.println ("Start");  
        break;  
    case Stopp:  
        Serial.println("Stopp");  
        break;  
    default:  
        Serial.println("anderer Befehl");  
        break;  
}
```

Hier wird genau wie vorhin *Start*, *Stopp* und *anderer Befehl* ausgegeben. Allerdings wird so auf den ersten Blick ersichtlich, wann welcher *Case* ausgeführt wird.

Eine enum-Variable kann auch mit den Integer-Werten verglichen werden. Die Definition von Switch-Case wie im vorherigen Beispiel ist somit nach wie vor verwendbar, allerdings würde dadurch der Sinn hinter der Enumeration verloren gehen.

Interrupts

Interrupts werden von einem bestimmten Event getriggert. Sie unterbrechen das Programm kurz, führen die Interrupt-Funktion aus und machen anschließend dort im Programm weiter, wo sie aufgehört haben.

4.5.3 Grundbegriffe der Audioverarbeitung

Abtastrate oder Samplerate: Um ein analoges Signal in ein digitales Signal umzuwandeln wird die Amplitude in einem bestimmten regelmäßigen Abstand abgetastet. Die Abtastrate gibt an, wie oft pro Sekunde abgetastet wird. Ihre Einheit ist Hertz. Eine zu niedrige Abtastrate führt zu Verlusten und Verzerrungen, deshalb muss sie mindestens doppelt so groß sein, wie die höchste im Audiosignal vorkommende Frequenz. Sie sollte allerdings auch nicht zu hoch sein, da das unnötig viel Speicherplatz und Rechenleistung verbraucht.

Bittiefe: Die Bittiefe gibt die Anzahl der Bits an, mit denen der Amplitudenwert eines einzelnen Abtastwertes gespeichert wird. So bestimmt sie die Auflösung und die Präzision. Eine höhere Bittiefe ermöglicht feinere Abstufungen und weniger Verzerrungen.

Sample: Ein *Sample* kann zwei unterschiedliche Bedeutungen haben, je nachdem ob man von Signalverarbeitung oder von Machine-Learning spricht. In der Signalverarbeitung ist ein *Sample* ein einzelner Amplitudenabtastwert, während bei Machine-Learning ein *Sample* ein komplettes Trainingsbeispiel ist.

Features: *Features* sind typische Merkmale, nach denen ein Audiosignal analysiert werden kann. Sie dienen als Eingabe in ein Machine-Learning-Modell. Deep-Learning-Ansätze können auch unstrukturierte Merkmale als Eingabe verwenden, aus denen das Modell die *Features* selbst lernt. Ein Beispiel für einen unstrukturierten *Feature*-Typ ist MFE, siehe Abschnitt 4.6.1.2.2.

Frames: Ein *Frame* ist ein Segment eines Audiosignals, das durch die Unterteilung eines Signals in kurze, sich überlappende Fenster entsteht. Aus ihnen werden dann die *Features* berechnet.

4.5.4 Gesamtfunktion

Abbildung 140 zeigt den vereinfachten Ablauf des Programmes für den ESP32.

Zuerst werden alle benötigten Pins und Module initialisiert. Es wird das SPI-Modul, das I²C-Modul, das I²S-Modul und das UART-Modul benötigt.

Über eine Interrupt-Funktion (siehe Abschnitt 4.5.2) werden die Sprachbefehle vom Arduino empfangen. Je nach Art des Befehls werden sie an den PIC weitergesendet oder über den Lautsprecher ausgegeben. Wenn die Übung bereits gestartet wurde, werden alle Befehle, die die Zieleinstellung betreffen ignoriert.

Falls der PIC Messdaten gesendet hat, werden diese eingelesen und ebenfalls über den Lautsprecher ausgegeben.

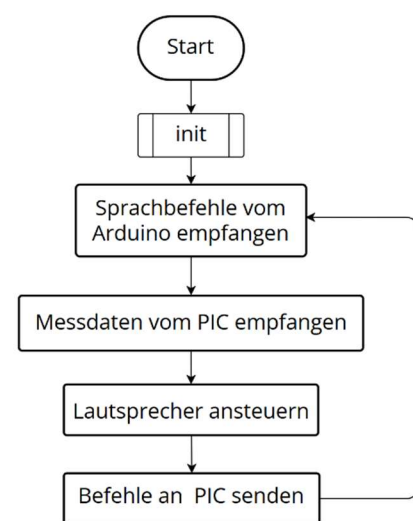


Abbildung 140: Flussdiagramm ESP32

4.5.5 Lautsprecher ansteuern

Im folgenden Abschnitt wird beschrieben, wie die Sprachdateien, die sich auf der SD-Karte befinden, ausgegeben werden.

4.5.5.1 ElevenLabs

Für die Sprachausgabe wurde ElevenLabs verwendet. Das ist eine Website, auf der Audiodateien mit KI-Stimmen generiert werden können. Es kann aus unterschiedlichen Sprechern gewählt werden, um eine für den jeweiligen Zweck passende Betonung bereitzustellen.

II ElevenLabs

Abbildung 141: Eleven Labs [27]

4.5.5.2 SPI

Das SD-Karten-Modul kommuniziert mittels SPI mit dem ESP32. SPI steht für *Serial Peripheral Interface* und ist ein serielles Kommunikationsprotokoll für den Datentransfer zwischen einem Controller und einem oder mehreren Peripheriegeräten. Zur Übertragung werden vier Leitungen benötigt. Eine Taktleitung, eine POCI (Peripheral out/ Controller in, veraltet auch MISO genannt), eine PICO (Peripheral in/ Controller out, veraltet auch MOSI genannt) und eine Chip-Select-Leitung. Die Übertragung erfolgt Vollduplex. Das heißt, es kann gleichzeitig gesendet und empfangen werden. SPI ist ein schnelles Kommunikationsprotokoll, weshalb es oft für SD-Karten und Displays verwendet wird.

Der Master liefert das Taktsignal auf die Taktleitung. Außerdem steuert er über die Chip Select Leitungen die einzelnen Peripheriegeräte an, indem er die jeweilige Leitung auf *HIGH* setzt.

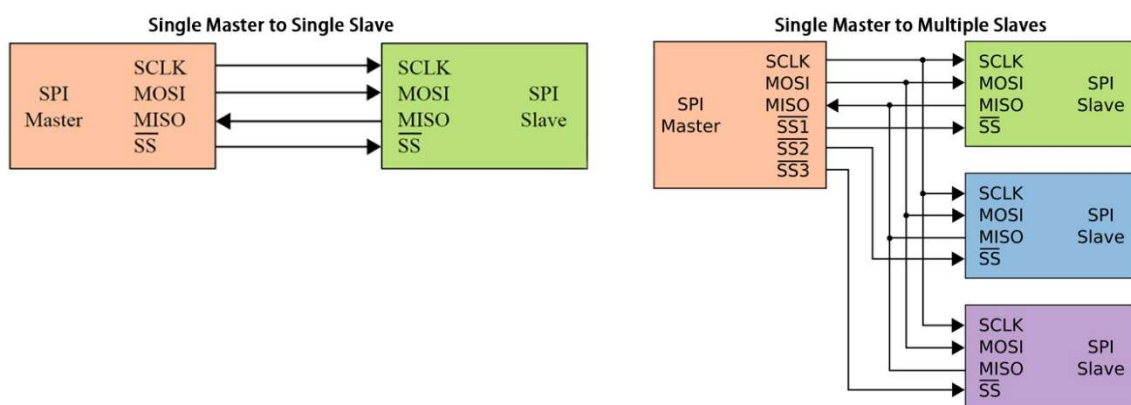


Abbildung 142: SPI Verdrahtung [24]

4.5.5.3 I²S

Der ESP32 kommuniziert mittels I²S mit dem Audioverstärker. I²S steht für *Inter-Integrated Circuit Sound* und ist eine serielle Schnittstelle für die Übertragung von Audiodaten. Es gibt drei Leitungen zur Übertragung. Eine Taktleitung, eine Datenleitung und eine Wortauswahlleitung. Die Wortauswahlleitung bestimmt, ob der linke oder der rechte Audiokanal übertragen wird. Die Aufteilung in einen linken und rechten Audiokanal (auch Stereo genannt) wird benötigt, wenn zwei Lautsprecher verwendet werden. So kann ein räumlicher, realistischer Klang erzeugt werden, bei dem bestimmte Töne stärker von links oder rechts ausgegeben werden können. Da allerdings nur ein Lautsprecher verwendet wurde, wird nur eine Mono-Audioübertragung benötigt. Dafür wird entweder die Wortauswahlleitung auf einen fixen Pegel gelegt (meist GND) oder es wird auf beiden Kanälen das gleiche Signal übertragen. Durch diese Übertragung können DACs (Digital Analog Converter) die Audiodaten in analoge Signale umwandeln.

Im Gegensatz zu I²C gibt es keine Adressierung und auch keine bidirektionale Kommunikation. Start- und Stoppbits sowie Acknowledge-Bits sind ebenfalls nicht vorhanden.

Der Takt und die Wortauswahl können entweder vom Empfänger oder vom Sender vorgegeben werden. Es gibt keine fest vorgegebene Taktfrequenz. Das Signal der Wortauswahl ändert sich, wie in Abbildung 143 dargestellt, immer einen Takt bevor das Datenwort zu Ende ist, um die Datenverarbeitung auf Empfänger- und Senderseite zu vereinfachen. Ein Datenwort ist eine feste Anzahl an Bits, die von einem Prozessor in einem Schritt verarbeitet werden kann. Die einzelnen Datenwörter werden direkt nacheinander gesendet.

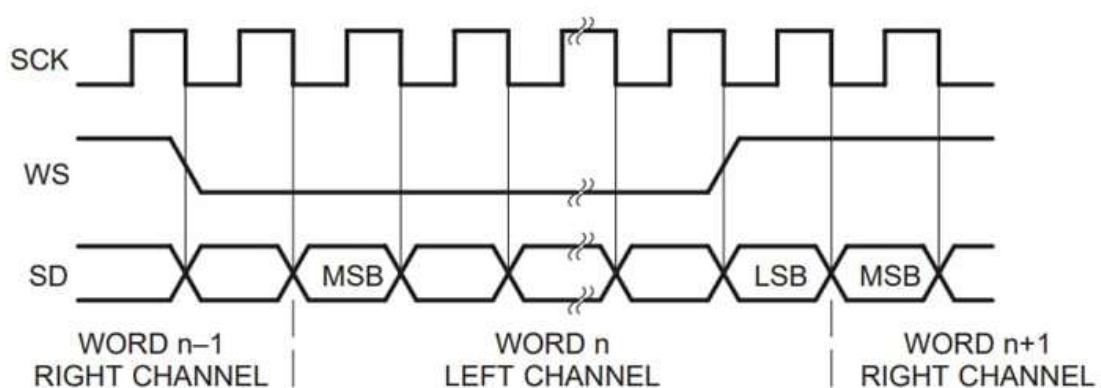


Abbildung 143: I²S Übertragung [25]



4.5.5.4 Verwendete Bibliotheken

Bibliotheken enthalten vordefinierte Funktionen und Klassen. Sie sind wiederverwendbar und werden eingesetzt, um den Aufwand bei der Programmierung zu verringern.

Wire.h

Wire.h ist eine Bibliothek, mit der die Kommunikation über eine I²C Verbindung durchgeführt werden kann. Sie initialisiert die benötigte Hardware und implementiert das I²C Protokoll. Es können Daten gesendet und empfangen werden.

Arduino.h

Arduino.h ist meist automatisch eingebunden und stellt Grundfunktionen wie `pinMode()`, `digitalRead()`/`digitalWrite()` und `delay()` zur Verfügung.

SD.h

SD.h ist eine Bibliothek, mit der auf SD-Karten zugegriffen werden kann. SD-Karten können über SPI gelesen oder beschrieben werden.

AudioFileSourceSD.h

Um Audiodateien, wie beispielsweise WAV-Dateien (Waveform Audio File Format-Dateien), von einer SD-Karte zu lesen, wird die Bibliothek `AudioFileSourceSD.h` verwendet. Diese reicht die Audiodaten anschließend an einen Audiogenerator weiter.

AudioGeneratorWav.h

`AudioGeneratorWav.h` bekommt eine WAV-Datei und wandelt sie in Roh-Audiodaten um. Diese Bibliothek interpretiert den WAV-Header. Der WAV-Header ist der Dateianfang, in dem essenzielle Daten für die Audiowiedergabe enthalten sind, wie beispielsweise die Abtastrate und die Bittiefe.

AudioOutputI2S.h

Um die Roh-Audiodaten mit I²S an den Verstärker zu senden, wird die Bibliothek `AudioOutputI2S.h` verwendet.

4.5.5.5 Funktionen

Dateien lesen

Um die Dateien von der SD-Karte zu lesen, wird die Bibliothek `AudioFileSourceSD.h` verwendet. Folgende Funktionen werden dazu benötigt:

`AudioFileSourceSD()` öffnet eine Datei auf der SD-Karte und gibt die Rohdaten an einen Decoder weiter.

`isOpen()` überprüft, ob eine Datei geöffnet und bereit zum Lesen ist. Gibt *TRUE* zurück, wenn die Datei gefunden und geöffnet wurde und *FALSE*, wenn das nicht der Fall ist.



Decoder

Die Bibliothek `AudioGeneratorWAV.h` wandelt die WAV-Datei in Roh-Audiodateien um. Folgende Funktionen wurden dazu benötigt:

`AudioGeneratorWAV()` ist der Decoder für WAV-Dateien. Er liest eine Datei ein und gibt die im Header enthaltenen Daten an ein Ausgabemodul wie `AudioOutputI2S()` weiter.

`isRunning()` überprüft, ob noch etwas abgespielt wird. Sie gibt *TRUE* zurück, wenn die Wiedergabe läuft und *FALSE*, wenn die Datei beendet ist oder gestoppt wurde.

`loop()` dekodiert bei jedem Aufruf ein Stück Audio und gibt es an die Ausgabe weiter. Sie muss in der `loop()`-Schleife des Programms immer wieder aufgerufen werden.

`stop()` stoppt den Decoder und gibt Ressourcen frei.

`begin()` liest den WAV-Header und verbindet die Audioquelle mit der Ausgabe.

Ausgabe

Die Bibliothek `AudioOutputI2S.h` sendet die Roh-Audiodaten an den Verstärker. Folgende Funktionen wurden dazu benötigt:

`AudioOutputI2S()` initialisiert den I²S Treiber vom ESP32, richtet alle für kontinuierliches Audio benötigten Buffer ein und übernimmt die Abtastrate und Bittiefe vom Decoder.

`SetPinout()` ändert die für I²S verwendeten PINs. Wird diese Funktion nicht aufgerufen werden die Standardpins verwendet.

`SetGain()` stellt die Verstärkung des Audiosignals ein, indem es die digitalen Audiodaten mit einem Faktor multipliziert, bevor sie ausgegeben werden. Eine Verstärkung von 1.0 ist die normale Lautstärke. Alle Werte darunter verringern die Lautstärke und Werte darüber erhöhen sie. Bei einer Verstärkung von über 1 kann es allerdings zu Verzerrungen kommen.

4.5.5.6 Programm

Da immer nur eine Datei nach der anderen abgespielt werden kann, aber keine Informationen verloren gehen sollen, werden die abzuspielenden Dateien in einen Ringbuffer (siehe Abschnitt 4.3.6.4.1.1) geschrieben. Wenn bereits eine Ausgabe läuft, werden die darauffolgenden Dateien dort zwischengespeichert und ausgegeben, sobald die vorherige Datei beendet wurde.

Um eine Ausgabe zu starten, wird zuerst die Datei geöffnet und überprüft, ob das Öffnen erfolgreich war. Wenn das der Fall ist, wird die Wiedergabe gestartet. Wenn nicht, heißt das, dass die Datei vom System nicht gefunden werden konnte. Dann wird zu Debug-Zwecken eine Fehlermeldung am Serial Monitor ausgegeben. Danach wird das Objekt, in dem die nicht gefundene Datei gespeichert wurde, gelöscht.

```
222     else {
223         if (speaker_head-speaker_tail != 0) {
224             //neue Datei öffnen
225             const char* Ausgabe = speaker_buffer[speaker_head % speaker_buffer_size].c_str();
226             file = new AudioFileSourceSD (Ausgabe);
227             speaker_head++;
228
229             if (file->isOpen()) { //Überprüfen ob Datei offen
230                 wav->begin(file, out); //Abspielen starten
231                 t_ausgabe = t_act;
232             }
233             else { //Wenn Datei nicht geöffnet Fehlermeldung
234                 Serial.println("Konnte nicht geöffnet werden!");
235                 delete file;
236                 file = nullptr;
237             }
238         }
239     }
```

Abbildung 144: ESP32 Programm Datei ausgeben

4.6 Spracheingabe

Die Spracheingabe erkennt verschiedene vordefinierte Befehle, mit denen ein Belastungsziel gesetzt werden kann, Übungen gestartet oder gestoppt werden können und die Lautstärke der Sprachausgabe eingestellt werden kann.

4.6.1 Edge Impulse

Edge Impulse ist eine Website, mit der Machine-Learning-Modelle trainiert und anschließend als Bibliothek oder als fertige Binary für den benötigten Mikrocontroller heruntergeladen werden können. Mithilfe von Edge Impulse wurde ein Modell trainiert, das die für die Spracheingabe benötigten Befehle erkennt.



Abbildung 145: Edge Impulse Logo [21]

4.6.1.1 Daten sammeln

Im folgenden Abschnitt wird erklärt, worauf bei der Aufnahme von Trainingsdaten geachtet werden muss und wie die Aufnahme mit Edge Impulse funktioniert.

Trainingsdaten

Jeder zu erkennende Befehl muss mehrfach von unterschiedlichen Personen einge-sprochen werden. Je mehr unterschiedliche Personen einsprechen, desto besser funk-tioniert das Modell schlussendlich, da unterschiedliche Tonhöhen, Lautstärken und verschiedene Sprechgeschwindigkeiten zu einer besseren Erkennung beitragen. Au-ßerdem sollte mit verschiedenen Hintergrundgeräuschen aufgenommen werden.

Zusätzlich zu den Befehlen müssen auch Hintergrundgeräusche und andere Gesprä-che aufgenommen werden. Das ist wichtig, damit das Modell nicht jedes gesprochene Wort als Befehl erkennt. Nimmt man allerdings zu viele Hintergrundgeräusche auf, passiert es immer häufiger, dass auch ein Befehl als solches erkannt und ignoriert wird. Deshalb sollte es ungefähr genauso viele Hintergrundaufnahmen wie tatsächli-che Befehls-Aufnahmen geben. Außerdem ist es wichtig, darauf zu achten, dass unter-schiedliche Arten von Hintergrundgeräuschen aufgenommen werden. Die Hinter-grundgeräusche werden im Label *unknown* gespeichert.

Die aufgenommenen Daten werden in Test und Train aufgeteilt. Ca. 80 % sind Train-Daten. Mit ihnen wird das Machine-Learning-Modell trainiert. Die restlichen ca. 20 % sind die Test-Daten. Diese werden verwendet, um das Modell zu testen. Es können nicht für beides dieselben Daten verwendet werden, da das Modell die Daten beim Testen schon vom Training kennen würde und so immer richtig zuordnen würde. Da-mit würde das Testen sinnlos werden.

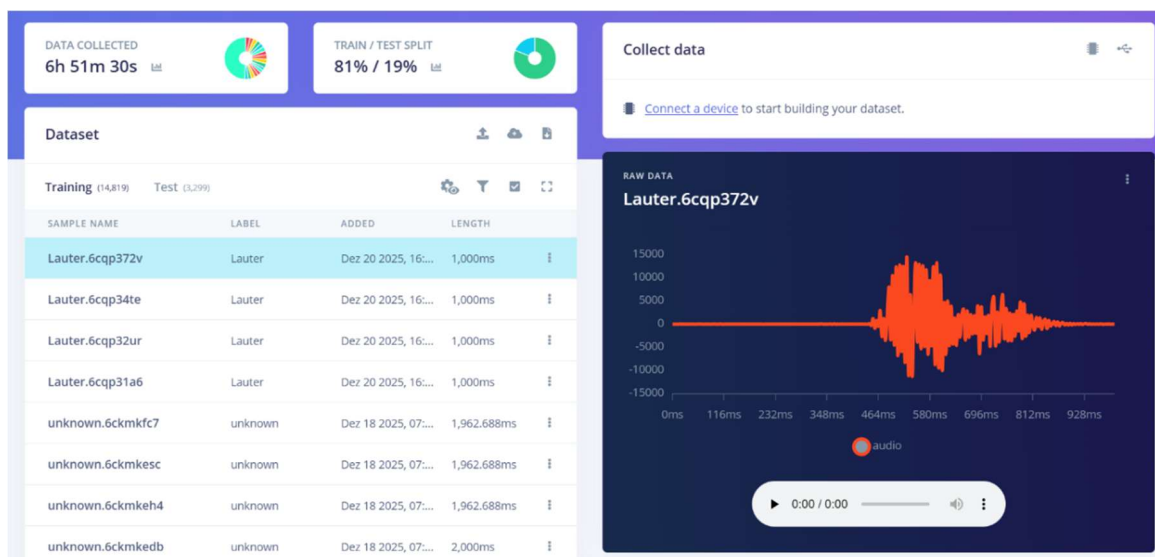


Abbildung 146: Daten aufnehmen

Daten über Arduino aufnehmen

Um den Arduino mit Edge Impulse verbinden zu können und so Daten direkt über das Mikrofon vom Arduino aufnehmen zu können, muss der Arduino Client und der Edge Impulse Client heruntergeladen werden. Dann muss in die Eingabeaufforderung *edge-impulse-daemon* eingegeben werden und der Benutzername, das Passwort, der COM-Port, an dem der Arduino angeschlossen ist und das Edge-Impulse-Projekt ausgewählt werden.

```
C:\Users\lisap>edge-impulse-daemon
Edge Impulse serial daemon v1.36.0
WARN: You're running an outdated version of the Edge Impulse CLI tools
      Upgrade via 'npm update -g edge-impulse-cli'
Stored token seems invalid, clearing cache...
Your JWT token has expired
? What is your user name or e-mail address (edgeimpulse.com)? LisaPripfl
? What is your password? [hidden]
Endpoints:
  Websocket: wss://remote-mgmt.edgeimpulse.com
  API:       https://studio.edgeimpulse.com
  Ingestion: https://ingestion.edgeimpulse.com

? Which device do you want to connect to? (🔍 type to search) (Press <enter> to submit)
> COM3 (Microsoft)
  COM4 (Microsoft)
```

Abbildung 147: edge-impulse-daemon

Anschließend können in Edge Impulse unter *Data acquisition*, die Daten aufgenommen werden. Die Frequenz ist dabei schon voreingestellt und darf nicht verändert werden.

Das Aufnehmen über den Arduino dauert allerdings viel länger als über den Laptop. Das liegt zum einen daran, dass es jedes Mal einige Sekunden dauert, bis das Mikrofon gestartet wird und bis die aufgenommenen Daten

Collect data

Device ?
Arduino Nicla Voice

Label
unknown

Sample length (ms.)
1000

Sensor
Microphone

Frequency
16000Hz

Start sampling

Abbildung 148: Daten über Arduino aufnehmen

hochgeladen wurden. Zum anderen liegt es daran, dass am Laptop nicht genau angezeigt wird, wann tatsächlich aufgenommen wird. Deshalb muss das Timing, wann gesprochen werden muss, selbst ermittelt werden. Das führt zu vielen unbrauchbaren Aufnahmen, da der Befehl abgeschnitten wird oder gar nicht zu hören ist.

Daten über den Laptop aufnehmen

Um Daten über den Laptop aufnehmen zu können, wird zuerst unter *Data acquisition* *Connect a device* und dann *Connect to your computer* ausgewählt. Dann öffnet sich ein Tab, in dem die Daten aufgenommen werden können.

Unter *Label* wird der Befehl ausgewählt, der gerade aufgenommen wird. *Length* gibt die Länge einer Aufnahme an. Unter *Category* kann ausgewählt werden, ob Test- oder Train-Daten aufgenommen werden sollen oder ob automatisch in das empfohlene Verhältnis aufgeteilt werden soll.

Abbildung 149: Daten über den Laptop aufnehmen

4.6.1.2 Impulse erstellen

Nachdem die Daten gesammelt wurden, wird ein *Impulse*, auch *Experiment* genannt, erstellt. Ein *Impulse* besteht aus drei verschiedenen Blöcken (*building blocks*): einem Eingabeblock, einem Verarbeitungsblock und einem Lernblock. Diese Blöcke machen aus den Rohdaten Daten, die das Modell versteht und trainieren anschließend das Modell. Außerdem gibt es die *Output features*. Dort wird angezeigt, welche Befehle erkannt werden sollen.

Abbildung 150: Impulse erstellen

4.6.1.2.1 Eingabeblock (*input block*)

Unter *Time series data* kann die *Window size* und *Window increase* eingestellt werden.

Jedes *Sample*, das aufgenommen wird, wird in kleinere Abschnitte unterteilt und in diesen Abschnitten analysiert. Mit *Window size* stellt man ein, wie lange einer dieser Abschnitte dauert. Verwendet man eine kleine *Window size*, werden schnelle Änderungen im Klang gut erkannt, dafür bringt es eine schlechtere Frequenzauflösung mit sich, also mehr rauschen. Bei einer großen *Window Size* werden die unterschiedlichen vorkommenden Frequenzen besser erkannt, dafür reagiert das Model langsamer.

Mit *Window increase* wird festgelegt um wie viel der nächste Abschnitt zeitlich versetzt ist. Hat man also zum Beispiel 1000 ms *Window size* und 500 ms *Window increase*, beginnt jeder neue Abschnitt 500 ms nach dem vorherigen. Das heißt sie überlappen sich immer um 50 %. Siehe auch Abbildung 151.

Für den Verarbeitungsblock *Syntiant* (siehe Abschnitt 4.6.1.2.2) müssen 968 ms *Window size* und 500 ms *Window increase* eingestellt sein.

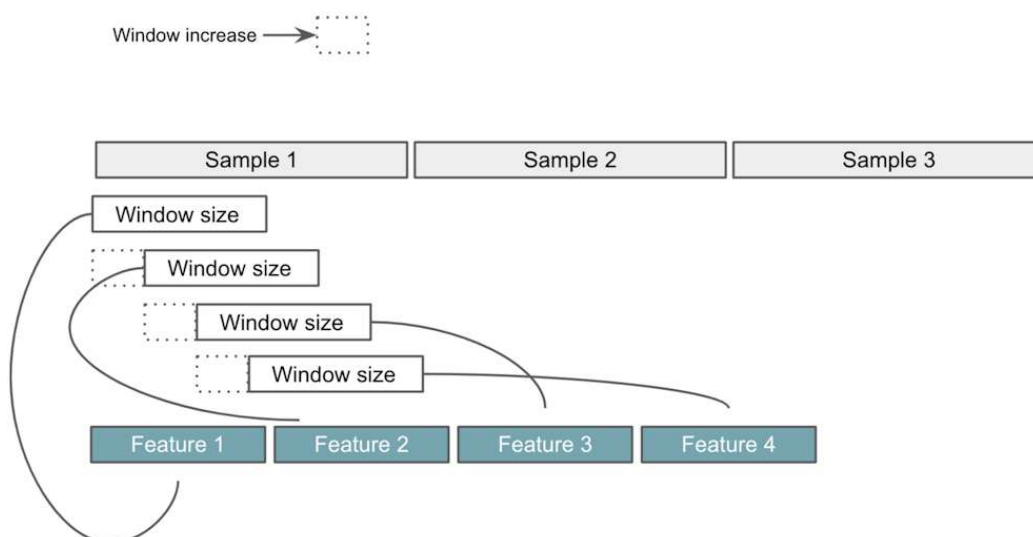


Abbildung 151: Window size und Window increase [23]

4.6.1.2.2 Verarbeitungsblock (*processing block*)

Der zweite Block ist der Verarbeitungsblock. Er extrahiert Features aus den Rohdaten und übergibt diese an das Machine-Learning-Modell. Dabei ist das Ziel, die Datenmenge zu reduzieren und dabei die für die Klassifizierung wichtigen Informationen beizubehalten.

MFE-Block

Um das Modell auf dem Laptop testen zu können, braucht es einen MFE-Block. MFE steht für *Mel-Frequency Energy*. Dieser Block analysiert das Audiosignal im Frequenzbereich und berechnet die Energie von verschiedenen Frequenzbändern.

Dazu wird das Signal zuerst mithilfe einer *FFT* in seine Bestandteile zerlegt. *FFT* steht für *Fast Fourier Transformation*. Sie wird verwendet, um ein Audio-Signal vom Zeitbereich in den Frequenzbereich umzuwandeln. Dadurch kann man erkennen, welche Frequenzen in einem *Frame* des Audiosignals enthalten sind. Dabei wird das Signal nicht in beliebige Frequenzen zerlegt, sondern in viele kleine festgelegte Frequenzbereiche, die *Frequenz-Bins* heißen. Jeder *Frequenz-Bin* enthält Informationen darüber, wie stark eine bestimmte Frequenz im Signal vorkommt. Die *Frequenz-Bins* werden anschließend zu Mel-Filterbändern zusammengefasst.

Wenn ein Frequenzband viel Energie hat, bedeutet das, dass die Frequenzen in diesem Bereich stark vertreten sind, also große Amplituden haben. Wenig bis keine Energie bedeutet, dass die Frequenzen entweder gar nicht vorkommen oder kleine Amplituden haben.

Die Frequenzbänder werden dabei nach der Mel-Skala geteilt. Bei der Mel-Skala werden die für den Menschen hörbaren Frequenzen nicht in gleichmäßige Teile geteilt, sondern tiefe Töne öfter unterteilt und hohe Töne weniger oft. Das liegt daran, dass das menschliche Gehör tiefere Töne feiner unterscheiden kann als hohe. So kann das Modell Details in hohen Frequenzbereichen ignorieren, wodurch es effizienter wird und sich eher an der menschlichen Wahrnehmung orientiert.

MFE ist eine relative einfache und schnelle Variante eines Verarbeitungsblockes und hat oft eine hohe Genauigkeit im Vergleich zu anderen. Dafür werden mehr Features berechnet, wodurch er mehr Speicherplatz benötigt und somit schlecht für Microcontroller geeignet ist.

Mel-Spektrogramm

Die Energie der Mel-Frequenzbänder wird häufig in einem Mel-Spektrogramm dargestellt (siehe Abbildung 152). Auf der x-Achse ist die Zeit in Sekunden aufgetragen und auf der y-Achse die Frequenz in Hertz. Die Farben zeigen an, wie stark die Energie in jedem Frequenzband ist. Je kräftigeres rot, umso mehr Energie, je dunkleres blau umso weniger.

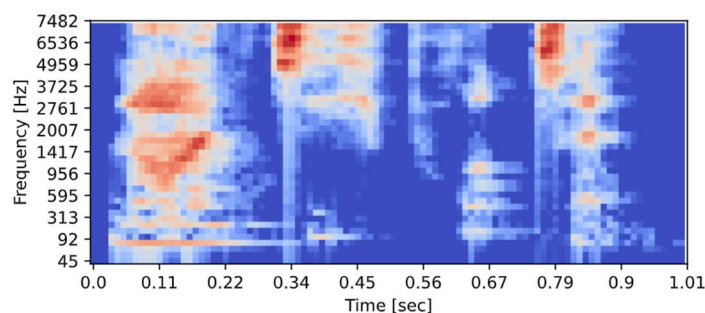


Abbildung 152: Mel-Energie-Diagramm

Syntiant-Block

Für den Arduino wird der Syntiant-Block benötigt. Das ist kein typischer Feature-Verarbeitungsblock, sondern ein spezielles Hardware-Frontend für einen Chip wie zum Beispiel den NDP120, der im Arduino Nicla Voice verbaut ist. Er wird benötigt, weil der NDP120 die Audio-Features direkt auf der Hardware berechnet. Um Labels erkennen zu können, muss Edge Impulse diesen Feature-Typ nachbilden.

Der Chip berechnet dabei eine Darstellung, die einem Log-Mel-Spektrogramm ähnelt. Während ein Mel-Spektrogramm die Energie der Frequenzbänder linear darstellt, wird bei einem Log-Mel-Spektrogramm eine logarithmische Funktion auf die Energiewerte der Frequenzbänder angewendet. Dadurch werden leisere Signale im Verhältnis zu lauterem Signalen sichtbar gemacht. Das ist hilfreich für die Spracherkennung, da wichtige Sprachinformationen, wie beispielsweise Konsonanten, häufig leise sind.

Syntiant-Spektrogramm

Das Syntiant-Spektrogramm ist eine spezielle Form eines Log-Mel-Spektrogramms. Oft sind allerdings einige Parameter der Signalverarbeitung durch die Hardware fest vorgegeben, wie beispielsweise die Anzahl der Filter (*Fiter number*) oder *FFT lenght*. Abbildung 153 zeigt das Syntiant-Spektrogramm für dasselbe *Sample* wie in der vorherigen Darstellung des Mel-Spektrogramms.

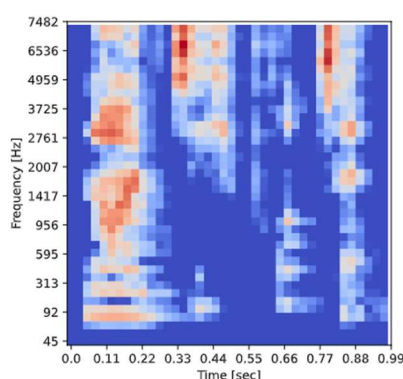


Abbildung 153: Syntiant-Spektrogramm

(Log-)Mel-filterbank energy features

Sowohl für den MFE-Block als auch für den Syntiant-Block können Parameter für die Umwandlung in ein Spektrogramm benutzerdefiniert werden.

Parameters

Log Mel filterbank energy features

Frame length ?	0.032
Frame stride ?	0.024
Filter number ?	40
FFT length ?	512
Low frequency ?	0
High frequency ?	0

Abbildung 154: (Log-)Mel-filterbank energy features

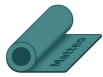
Mit *Frame length* kann angegeben werden, wie lang die *Frames* sind, aus denen dann die *Features* erzeugt werden. Es werden also die Abschnitte, die vorher durch *Window size* erstellt wurden, nochmal aufgeteilt.

Frame stride gibt an um wie viel diese *Frames* zueinander versetzt sind. Also ähnlich wie oben *Window increase*.

Die *Filter number* gibt die Anzahl der Mel-Filterbänder an. Werden mehr Filter gewählt, hat man eine genauere spektrale Auflösung. Das heißt, dass nahe beieinanderliegende Frequenzen präziser unterschieden werden können.

FFT length gibt die Anzahl der Amplitudenmesswerte an, die für die *FFT* genutzt werden. Aus dieser Anzahl ergibt sich die Frequenzauflösung, die mit folgender Formel berechnet werden kann:

$$\text{Frequenzauflösung} = \frac{\text{Samplerate}}{\text{FFT length}}$$



Bei einer Samplerate von 16 kHz und einer *FFT length* von 512 ergibt sich beispielsweise eine Frequenzauflösung von 31,25 Hz. Das heißt, dass die einzelnen *Frequenz-Bins* 31,25 Hz auseinanderliegen. Allerdings ist nur die Hälfte dieser *Frequenz-Bins* nutzbar, was daran liegt, dass ein reales Signal symmetrisch ist. Somit ist die zweite Hälfte des Spektrums nur eine Spiegelung der ersten Hälfte und enthält keine neuen Informationen.

Low frequency steht für die niedrigste Frequenz, der Mel-Filter und *High frequency* für die höchste Frequenz. Ist bei *High frequency* 0 eingetragen ist es die halbe Samplerate.

4.6.1.2.3 Lernblock (*Learning block*)

Der Lernblock bekommt die verarbeiteten Daten aus dem Verarbeitungsblock als Input. Dieser Block analysiert die Daten und lernt daraus welche Muster zu welchen Labels passen. Für das Projekt wird *Transfer Learning (Keyword Spotting)* verwendet, da es darauf optimiert wurde, Gesprochenes zu erkennen. Es benutzt ein bereits vorhandenes Modell und passt es an die eigenen Werte an.

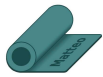
Training settings	
Number of training cycles ?	100
Use learned optimizer ?	☐
Learning rate ?	0.02

Abbildung 155: Transfer Learning Parameter

Auch der *Transfer Learning Block* kann benutzerdefiniert eingestellt werden.

Bei einem Trainingszyklus werden alle Trainingsdaten vom Modell klassifiziert und bei Fehlern die Werte des vortrainierten Modells angepasst, um Fehler in der Zukunft zu vermeiden. *Number of training cycles* gibt die Anzahl dieser Trainingszyklen an.

Die *learning rate* gibt an, wie schnell das vortrainierte Modell seine Werte an das neue Datenset anpasst, um Fehler zu vermeiden. Hat man eine zu hohe *learning rate*, kann das schon Vorgelernte verloren gehen. Ist sie allerdings zu niedrig, dauert das Training sehr lange, da das Modell kaum dazu lernt, wodurch viele Trainings-Zyklen benötigt werden.

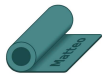


4.6.1.3 Confusion Matrix

Die *Confusion Matrix* zeigt an, wie oft das Modell die Befehle richtig erkannt hat und was es stattdessen erkannt hat, falls es zu einem Fehler gekommen ist. In den Zeilen sieht man die eigentlichen Labels und in den Spalten die vorhergesagten. Oberhalb der Matrix sieht man auch die Gesamtgenauigkeit. In Abbildung 156 liegt sie bei 83,0 %.



Abbildung 156: Confusion Matrix



Ganz unten in der *Confusion matrix* sieht man den *F1 Score*. Mit ihm kann bewertet werden, ob ein Modell gut funktioniert. Er ist ein Wert, der sich aus *Precision* und *Recall* ergibt.

Precision ist der Anteil der richtig vorhergesagten *Keywords* aus allen als *Keyword* vorhergesagten Fällen. Wenn das Modell also zum Beispiel fünf Mal sagt, dass der Befehl *Lauter* erkannt wurde, aber nur drei davon sind tatsächlich *Lauter* ist *Precision* = 60 %.

Recall ist der Anteil der richtig vorhergesagten *Keywords* an allen tatsächlich vorhandenen *Keywords*. Also wenn es den Befehl *Lauter* zum Beispiel fünf Mal gibt und dass Modell drei davon erkennt, ist *Recall* = 60 %.

Aus diesen zwei Werten berechnet sich dann der *F1 Score* mit folgender Formel:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Ist $F1 = 1$ macht das Modell keine Fehler, ist $F1 = 0$ erkennt das Modell die *Keywords* gar nicht.

Außerdem kann anhand der *Confusion Matrix* erkannt werden, welche Befehle das Modell schlecht voneinander unterscheiden kann. In Abbildung 156 beispielsweise erkennt man, dass anstelle von *rechtes Bein* oft *linkes Bein* erkannt wird. Außerdem ist erkennbar, dass die verschiedenen Prozentzahlen manchmal verwechselt werden. Das liegt daran, dass diese Befehle sehr ähnlich klingen und somit schwieriger zu unterscheiden sind.

Dass einem Modell nie Fehler passieren, ist unmöglich zu erreichen, da ein Kompromiss zwischen Genauigkeit und Speicherverbrauch gefunden werden muss. Außerdem können die Trainingsdaten nie absolut alle Hintergrundgeräusche und Betonungen eines Befehls beinhalten und somit kommt es immer zu einem gewissen Fehleranteil.

4.6.1.4 Live Classification

Um ein Modell vom Laptop aus testen zu können, wird in den *Live Classification* Modus gewechselt. Dieser ist besonders dann hilfreich, wenn die benötigte Hardware noch nicht verfügbar ist oder für erste Tests, ob das Modell Befehle richtig erkennt. Im *Live Classification Modus* wird in Echtzeit ausgegeben, zu welchem Label das Modell die gerade vorhandenen Geräusche bzw. das Gesprochene zuordnet. Man sieht immer die fünf zutreffendsten Klassen. Wenn eine davon über 0.7 übereinstimmt, ist höchstwahrscheinlich gerade dieser Befehl zu hören. Man sollte sich allerdings nicht auf die Ergebnisse aus dieser *Classification* verlassen, da das Modell am Laptop meist um einiges besser funktioniert. Das liegt zum einen daran, dass die CPU des Laptops um einiges schneller ist als der Arduino und mehr Speicherplatz vorhanden ist, zum anderen daran, dass mit einem anderen Verarbeitungsblock gearbeitet wird.

TOP 5 RESULTS	
193	Start: 1.00 , Stopp: 0.00, unknown: 0.00, 20_Prozent: 0.00, Lauter: 0.00
192	Start: 1.00 , Stopp: 0.00, 20_Prozent: 0.00, 80_Prozent: 0.00, unknown: 0.00
191	Start: 0.43 , 50_Prozent: 0.31, 80_Prozent: 0.09, 70_Prozent: 0.05, 90_Prozent: 0.03
190	unknown: 0.46 , Linkes_Bein: 0.12, Rechtes_Bein: 0.11, Kein_Ziel: 0.10, 50_Prozent: 0.04
189	unknown: 0.44 , Rechtes_Bein: 0.13, Linkes_Bein: 0.11, Kein_Ziel: 0.10, 50_Prozent: 0.04
188	unknown: 0.44 , Rechtes_Bein: 0.13, Linkes_Bein: 0.11, Kein_Ziel: 0.10, 50_Prozent: 0.04
187	unknown: 0.44 , Rechtes_Bein: 0.13, Linkes_Bein: 0.11, Kein_Ziel: 0.10, 50_Prozent: 0.04

Abbildung 157: Live Classification

4.6.1.5 Deployment

Das fertige Modell kann als Bibliothek oder als fertige Binary heruntergeladen werden, sofern der Microcontroller von Edge Impulse offiziell unterstützt wird. Für den Nicla Voice gibt es eine eigene Binary, die heruntergeladen und auf den Arduino hochgeladen wird.

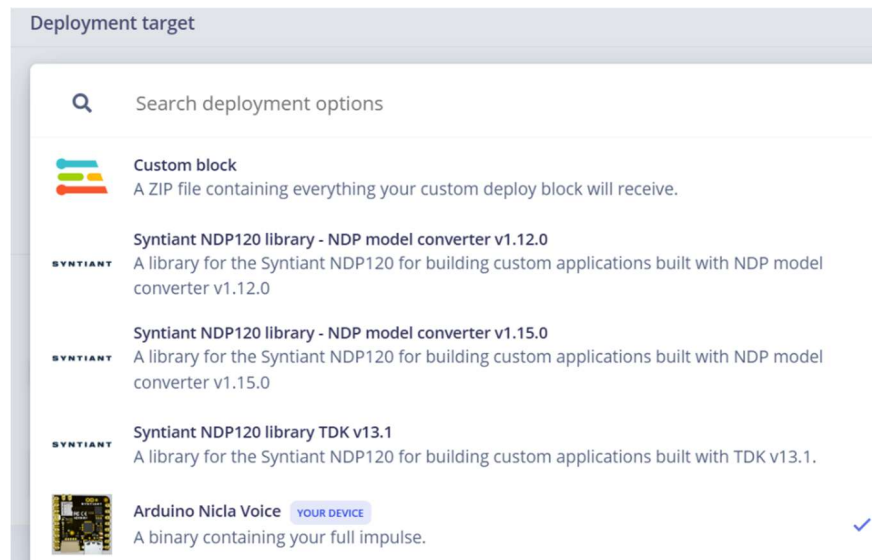


Abbildung 158: Deployment

4.6.2 Firmware

Im folgenden Abschnitt wird die Firmware beschrieben, die auf dem Arduino ausgeführt wird.

4.6.2.1 NDP-Firmware

Jeder Ordner, der beim Exportieren eines Machine-Learning-Modells für den Arduino heruntergeladen wird, enthält unterschiedliche Dateien für die jeweiligen Betriebssysteme des Laptops. Für Windows wird die Datei *flash_windows.bat* verwendet. Jede dieser Dateien für das Flashen, besteht aus drei Teilen, die gemeinsam hochgeladen werden. Diese drei Teile werden auch als Packages bezeichnet.

mcu_fw_120_v91.synpkg ist die Firmware für den NDP120. Sie steuert unter anderem den Speicher, kümmert sich um die Kommunikation zur MCU und lädt und kontrolliert die DSP-Firmware.

dsp_firmware_v91.synpkg ist die DSP- Firmware. DSP steht für *Digital Signal Processing* (digitale Signalverarbeitung). Es wandelt die Mikrofonssignale in digitale Daten um. Außerdem wird Rauschen gefiltert und die Lautstärke normalisiert. Aus diesen Daten werden dann die Syntiant-Features (siehe Abschnitt Verarbeitungsblock (*processing block*) 4.6.1.2.2) erzeugt.

ei_model.synpkg ist das Machine-Learning-Modell. Es bekommt die Daten von der DSP-Firmware und entscheidet, ob gerade ein Befehl gesagt wurde.

Die ersten zwei Dateien sind allerdings immer dieselben, weshalb sie nur beim ersten Mal hochgeladen werden. Danach befinden sich diese bereits am Arduino und werden automatisch übersprungen. Dann wird nur noch das Machine-Learning-Modell überschrieben.

Beim ersten Hochladen ist jedes Mal der Fehler „*Didn't get a nak when expected*“ aufgetreten. *Nak* steht für *negative acknowledgement* und somit bedeutet dieser Fehler, dass der Arduino nicht geantwortet hat, als es vom System erwartet wurde.

Um diesen Fehler zu beheben, muss zuerst einmal kurz der Reset-Button gedrückt werden, nachdem der Arduino vom System gefunden wurde und gefordert wird, dass eine beliebige Taste gedrückt wird, wie in Abbildung 160 abgebildet. Wenn das Hochladen erfolgreich war, sollte es aussehen, wie in Abbildung 159 dargestellt ist.

```
Finding Arduino Mbed Nicla 4.0.8...
arduino:mbed_nicla 4.0.8      4.5.0   Arduino Mbed OS Nicla Boards
Finding Arduino MBED core OK
Finding Nicla Voice...
Finding Nicla Voice OK at COM5
Flashing Arduino firmware...
Open On-Chip Debugger 0.11.0+dev-gab95bac57-dirty (2021-05-11-10:45)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
debug_level: 0

adapter speed: 1000 kHz

target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00006654 msp: 0x20010000
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00006654 msp: 0x20010000
** Programming Started **
** Programming Finished **
shutdown command invoked
New upload port: COM5 (serial)
Drücken Sie eine beliebige Taste . . . |
```

Abbildung 160: Arduino flashen

```
.\syntiant-uploader-win.exe send -m "Y" -w "Y" -p COM5 ei_model.synpkg
2026/02/10 09:35:57 main.go:81: ei_model.synpkg sent successful
Sending 545476 bytes
blocks : 533
```

Abbildung 159: Flashen erfolgreich

4.6.2.2 Microcontroller-Unit-Firmware (MCU)

Zuerst werden die Packages auf den NDP120 geladen und die I²C Schnittstelle initialisiert. Es dauert immer ein bis zwei Minuten, bis die Packages fertig geladen haben, deshalb sendet der Arduino direkt danach eine Meldung an den ESP32, dass dieser nun einsatzbereit ist.

Der NDP120 verarbeitet mit dem Sprachmodell die eingelesenen Audiodaten und sendet erkannte Labels an die MCU. Die MCU arbeitet mit einer Interrupt-Funktion. Immer wenn ein Label erkannt wurde, wird über I²C der entsprechende Befehl an den ESP32 gesendet.

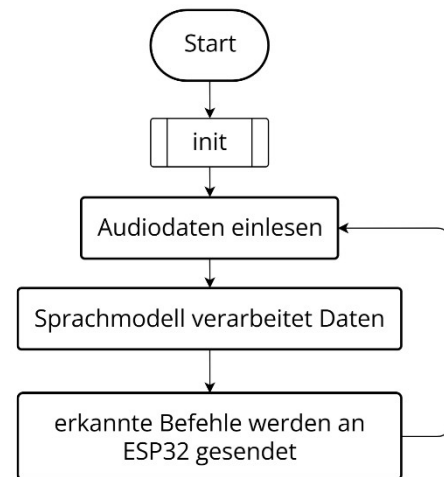


Abbildung 161: Flussdiagramm Arduino

Das Programm basiert auf einem Beispiel von Edge Impulse, bei dem über Sprachsteuerung Relais angesteuert wurden.

4.6.3 Bibliotheken und Funktionen

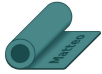
Im folgenden Abschnitt werden die Bibliotheken und Funktionen beschrieben, die für die Firmware der MCU verwendet wurden.

4.6.3.1 NDP.h

Die Bibliothek NDP.h steuert die Kommunikation zwischen dem NDP-Chip und der MCU. Außerdem enthält sie unter anderem Match- und Error-Handler, Status- und Error-Codes und Modellinformationen. Außerdem enthält sie ein globales Objekt NDP.

void onError(mbed::Callback<void(void)> _cb): Der Error-Handler enthält einen Funktionszeiger auf eine Callback-Funktion mit dem Rückgabotyp void, die keine Argumente nimmt. Sie wird aufgerufen, wenn es zu einem Fehler im NDP-System kommt.

Ein Handler ist eine spezielle Funktion, die auf ein bestimmtes Ereignis wartet und dann spezifischen Code ausführt. Der Error-Handler beispielsweise erkennt und verarbeitet Fehler. Häufig protokollieren Error-Handler aufgetretene Fehler, wiederholen den Prozess, in dem ein Fehler aufgetreten ist oder brechen den Vorgang ab. In diesem Fall ruft der Error-Handler eine Callback-Funktion auf. Eine Callback-Funktion ist eine Funktion, die als Argument einer anderen Funktion übergeben wird. Sie kann für verschiedene Anwendungszwecke konfiguriert werden.



void onMatch(mbed::Callback<void(char*)> _cb): Der Match-Handler reagiert auf einen erkannten Befehl. Auch beim Match-Handler gibt es eine Callback-Funktion mit dem Rückgabebetyp void. Diese Funktion nimmt ein Argument vom Typ char*. Der erkannte Befehl wird als Argument an die Funktion übergeben.

int begin(const char* fw1): Mit dieser Funktion wird der NDP-Chip initialisiert und die Kommunikation mit der MCU gestartet. Die Funktion gibt 1 zurück, wenn die Initialisierung erfolgreich war und 0, wenn sie fehlgeschlagen ist. Wird keine Firmware-Datei als Argument an die Funktion übergeben, wird jede passende synpkg-Datei im Root Folder geladen. Der Root Folder ist die oberste Ebene eines Dateisystems, von der alle Unterverzeichnisse abzweigen.

int load(const char *fw, int bootloader = 0): Hier wird eine weitere Firmware-Datei auf den NDP-Chip geladen. Wenn das Hochladen erfolgreich war, gibt die Funktion 1 zurück, ansonsten 0.

int getInfo(): Mit dieser Funktion können strukturierte Informationen über die Firmware (Version und Konfiguration), das Machine-Learning-Modell (Version, Anzahl der Klassen, Modellgröße) und die Audio-Konfiguration (Sample-Rate, Frontend-Parameter) am seriellen Monitor ausgegeben werden. So kann überprüft werden, ob alle Dateien korrekt geladen wurden. Die Funktion gibt 1 zurück, wenn die Daten erfolgreich ausgegeben werden konnten und 0, wenn es zu einem Fehler gekommen ist.

int turnOnMicrophone(): Diese Funktion initialisiert und startet das Mikrofon. Die Funktion gibt den Modus zurück, in dem sich das Mikrofon gerade befindet. Der Modus bestimmt, ob der Takt dauerhaft läuft, oder nur wenn gerade Audio aufgenommen wird.

void noInterrupts(): Diese Funktion deaktiviert alle Interrupts, die vom NDP-Chip kommen.

void interrupts(): Diese Funktion aktiviert alle Interrupts, die vom NDP-Chip kommen.

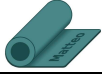
4.6.3.2 Nicla_System.h und RGBled.h

Diese beiden Bibliotheken werden automatisch eingebunden, sobald Arduino Nicla Voice in der Arduino IDE ausgewählt wurde.

bool nicla::begin(): Ist die einzige Funktion, die von Nicla_System.h verwendet wird. Diese Funktion initialisiert den Arduino.

Die **RGBled-Library** steuert die im Arduino verbaute RGB-LED an. Sie hat die vordefinierte Farben Rot, Grün, Blau, Gelb, Cyan, Magenta und Weiß.

void begin(): Diese Funktion initialisiert die RGB-LED.

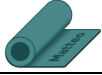


void setColor(Colors color): Eine der vordefinierten Farben wird der Funktion als Argument übergeben. Diese schaltet die LED in der entsprechenden Farbe ein. Man kann der Funktion auch jeweils einen Zahlenwert für Rot, Gelb und Blau übergeben, um eigene Farben anzuzeigen.

void end(): Diese Funktion schaltet die LED aus.

4.6.3.3 String-Compare

int strcmp(char* string1, char* string2): Die String-Compare Funktion vergleicht zwei C-Strings Zeichen für Zeichen. Sie vergleicht so lange, bis entweder ein Zeichen unterschiedlich ist oder sie bei der Nullterminierung angekommen ist. Die Funktion gibt den Wert 0 zurück, wenn beide Strings gleich sind. Wenn sie unterschiedlich sind, wird die Differenz der ASCII-Werte des ersten unterschiedlichen Zeichens ausgegeben. So entsteht ein Wert kleiner null, wenn der erste String lexikographisch kleiner ist als der Zweite und ein Wert größer null, wenn der erste String lexikographisch größer ist als der Zweite. Lexikographisch kleiner bedeutet, dass das erste abweichende Zeichen einen kleineren ASCII-Wert hat, lexikographisch größer, dass es einen größeren ASCII-Wert hat.



5 Lessons Learned

5.1 Laurenz Wimmer

Meine Aufgaben waren die Stromversorgung der Matte und das Projektmanagement. Dazu gehört auch die Kommunikation mit Sponsoren und der Universität Wien.

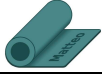
Grundsätzlich hat mir das Projekt von Beginn an Spaß gemacht, auch wenn ich gewusst habe, dass ich nicht viel Technisches zu erledigen haben werde. Rückblicken war die Zielverteilung nicht gut ausgewogen. Ich hätte noch das ein oder andere Elektronik-Ziel übernehmen können, weil sich meine Ziele als weniger umfangreich herausgestellt haben als ursprünglich angenommen. Dafür war Nandors Arbeitsaufwand unverhältnismäßig mehr, weil er einen Großteil der Elektronik und die gesamte Konstruktion machen musste. Dabei habe ich zwar immer mit ihm gemeinsam überlegt, wie wir zu einer guten Umsetzung kommen könnten, er hat aber dennoch einen großen Teil der Arbeit übernommen.

Vor allem, als sich im Dezember immer größer werdende Verzögerungen abgezeichnet haben, hätte ich trotzdem Aufgaben übernehmen können, habe es allerdings nicht getan. Diesbezüglich bin ich zumindest teilweise an den Verzögerungen schuld, die sich durch die gesamte 2. Projekthälfte gezogen haben.

Dennoch hat es sich für mich so angefühlt, als hätten wir laufend Fortschritte gemacht. Ich glaube diesen Eindruck habe ich deswegen gehabt, weil ich auch bei Lisa und Mika im Rahmen meines Verständnisses versucht habe, am aktuellen Stand zu bleiben. Trotzdem haben wir im Laufe des Projektes immer wieder die Situation gehabt, dass sich nicht alle Teammitglieder im Klaren darüber waren, wie weit das Projekt als gesamtes schon ist. Insbesondere der Austausch zwischen Software- und Hardware-Verantwortlichen war nicht immer gegeben, was mir allerdings nicht so deutlich klar war. Einige Missverständnisse haben zwischenzeitlich auch zu Spannungen im Team beigetragen.

Ich habe auch sehr viel Vertrauen in unsere Fähigkeiten gehabt und deswegen nie die Notwendigkeit gesehen, aufgrund von Verzögerungen Projektmitglieder zu motivieren, ihre Aufgaben möglichst bald zu erledigen. Dieses Mal hat es zwar genau so funktioniert, wie ich es mir vorgestellt habe, das hätte aber auch nicht so sein können. Dadurch ist vor allem bei Mika und Lisa gegen Ende des Projektes unnötiger Stress entstanden, der mit ein bisschen Aufwand aller Teammitglieder einfach hätte vermieden werden können.

Die Kommunikation war meines Empfindens grundsätzlich in Bezug auf Aufgaben und Probleme sehr gut, dafür war nicht immer klar, welche persönlichen Bedürfnisse momentan zu kurz kommen. Dadurch und durch den unnötigen Stress am Ende des Projektes sind erneut Spannungen entstanden, die auch zu unnötigen Schuldzuweisungen geführt und das Klima im Team getrübt haben. In einem weiteren Projekt

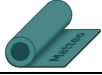


würde ich definitiv mehr Aufwand in Richtung Kommunikation betreiben, um solche Situationen zu vermeiden.

Trotz dieser Probleme bin ich der Meinung, dass wir im Rahmen des Möglichen die Umsetzung des Projektes sehr gut durchgeführt haben. Die Abnahme wurde erfolgreich 10 Tage vor der Deadline durchgeführt und alle anderen Termine eingehalten. Die Kommunikation mit der Universität Wien war sehr gut und hat nie Probleme bereitet. Die einzelnen Projektmitglieder haben ihre Aufgaben letzten Endes sehr gut umgesetzt und ich finde, dass die Zusammenarbeit Spaß gemacht hat.

Für mich war es sehr interessant, den Entstehungsprozess eines Produktes wie diesem mitzuerleben. Ich hätte nicht damit gerechnet, so viele Erfolge feiern zu können, die auch die Rückschläge bei weitem übertrumpfen. Der beste Moment war für mich, als wir die Matte zusammengebaut haben und alle Programme und die Elektronik funktioniert haben, so wie sie es sollen. Auch die Problemlösungsprozesse waren für mich sehr lehrreich und spannend.

Für zukünftige Projekte werde ich mir außerdem mitnehmen, wie wichtig ein engagiertes, interessiertes und vor allem auch kompetentes Team sind. Die Zusammensetzung hat meiner Meinung nach einen wichtigen Faktor gespielt, da alle Teammitglieder Fähigkeiten auf unterschiedlichen Fachgebieten beherrschen, wodurch jeder benötigte Bereich optimal abgedeckt war. Außerdem hat niemand von uns als Ziel nur eine positive Note gehabt, sondern das bestmögliche Endergebnis zu erzielen, was sehr wichtig für eine erfolgreiche Umsetzung ist.



5.2 Nandor Bagari

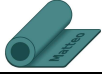
Mein Teil war die Elektrotechnik, mit Fokus auf die Sensoren und der Platine, sowie die Konstruktion. Ich bin sehr glücklich, dass ich diese zwei Themen zugeteilt bekommen habe, da mir das am meisten Spaß macht. Ich bin mir recht sicher, dass mir Projektmanagement, Organisatorisches oder Programmierung ziemlich schnell die Motivation an dem Projekt weiterzuarbeiten genommen hätte.

Ein großes Problem, welches sie so ziemlich durch das gesamte Projekt gezogen hat, war meine Faulheit. Ich tendiere sehr oft (und das nicht nur in der Schule, sondern auch allgemein im Leben) dazu, viel zu Prokrastinieren und Sachen auf den letzten Moment zu lassen. Auch bei dem Projekt war es nicht anders. Die schriftlichen Abgaben sowie die Erstellung von Präsentation habe ich erst wenige Tage, wenn überhaupt vorher begonnen. So geriet ich oft in großen Zeitstress, was für mich und auch meinen schulischen Leistungen nicht guttat. Ein weiteres großes Problem war, dass durch diese Faulheit der Fortschritt am Projekt erschwert wurde. Da die Software nur mit der Hardware rennen kann, habe ich mit meinem Nicht-tun nicht nur mich behindert, sondern auch zwei meiner Teammitglieder. Am Ende hat uns daher die Zeit zum vollständigen Testen des Projektes gefehlt, wodurch möglicherweise kleine unentdeckte Fehler noch vorhanden sind. Ich werde in Zukunft probieren meine Zeit besser einzuteilen, vor allem wenn dadurch andere weitermachen können.

Am meisten Spaß hat mir beim Projekt der elektrotechnische Teil gemacht. Immer wieder bin ich erstaunt über die große Menge an Zeit, was die Wahl der Bauteile beansprucht. Beim Schätzen der Zeit dachte ich, dass sicherlich das Designen der Platine, die meiste Zeit beanspruchen würde. In der Wahrheit war es nur ein bisschen aufwendiger als die Bauteilsuche und Suche nach deren Datenblätter, Platinen-Footprints, 3D-Modelle von den Bauteilen. Da ich diese Suche auch sehr anstrengend finde im Vergleich zum Platinen-designen hat es sich auch viel länger angefühlt.

Auch ist mir aufgefallen, wie sehr ich eigentlich „Platinen auf Platinen“ nicht mag. Damit mein ich Bauteile wie zum Beispiel die Spannungswandler, das Akkulademodul oder alle Module für die Sprachsteuerung und -ausgabe. Irgendwas an diesem Aufbau stört mich. Ich würde viel lieber viele kleine SMD- oder DIP-Bauteile anlöten wollen als Pin-Leisten. Meiner Meinung nach sind diese Platine-Bauformen dafür gedacht mit einer Steckplatine und Jumper-Kabeln genutzt zu werden und haben daher kein Platz auf einer größeren Platine.

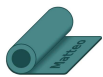
Die Konstruktion war eine Sache, wo ich mir zeitlich besonders schwergetan habe. Ich konnte nur sehr schwer die Hürde des Startens überwinden, weshalb die Konstruktion sehr langsam voranging. Innerhalb der Klasse war es ein wiederkehrender Scherz, dass man noch immer keine Matte sieht. Die endgültige Konstruktion wurde auch erst am Tag der Abnahme zum ersten Mal fertig und das auch nur weil so ziemlich das gesamte Team ab Anfang Februar an der Konstruktion arbeitete, wofür ich auch sehr dankbar bin.



Die Erstellung des Elektronikgehäuses war auch eine besonders große Hürde. Da mit am Anfang nicht einmal ganz klar war, was ich überhaupt konstruieren würde, war es ein großer Trial-and-Error. Die meisten Maße wurden auf Augenmaß gewählt. Das 3D-drucken des Gehäuses war auch eine Herausforderung, da sie so lang ist. Im Endeffekt wurde der Boden in zwei, der Deckel in 3 Teilen gedruckt und dann nachher zusammengeklebt. Auch ist dabei nicht alles glatt gelaufen. Wegen mehrfachen Fehlern von dem Drucker und einem falschen Maß, was erst nach dem Drucken der ersten Teile aufgefallen ist, hat es mehrere Tage gebraucht um das Gehäuse zu drucken.

Am Anfang des Jahres war mein Plan das ich nach der Schule Maschinenbau studieren wollen würde, jedoch hat das Projekt meine Meinung geändert. Mir hat der elektrotechnische Teil viel mehr Spaß gemacht als der Konstruktive und ich habe deshalb auch entschieden eher Elektrotechnik zu studieren.

Insgesamt fand ich das Projekt echt toll und spannend, ich konnte viel für die Zukunft mitnehmen. Auch hätte ich mir kein kompetenteres Team wünschen können. Alle Mitglieder sind echte Expertinnen und Experten in ihrem Themengebiet und ohne dieses hohe Niveau hätte man das Projekt nicht so gut bewältigen können.



5.3 Mika Böhm

Mir hat dieses Projekt von einer technischen Perspektive äußerst viel Spaß gemacht. Ich konnte mich mit so vielen interessanten Dingen auseinandersetzen und habe dabei eine Unmenge an neuem Wissen und Kompetenzen erlangt.

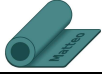
Begonnen habe ich in den Sommerferien mit dem Erstellen der Website. Von Anfang an war für mich klar, dass ich mit so wenig Abstraktion wie möglich arbeiten wollte. Die Website mit einem Framework zu erstellen, kam also nicht infrage. Ich kenne jedes Element und jede Funktion der Website. Ich kann erklären was jeder der verwendeten HTML-Tags und jede verwendete CSS-Eigenschaft tut und wofür diese eingesetzt werden. Ich habe eine sehr steile Lernkurve beobachten können. Mit jeder weiteren Seite, die ich abgeschlossen habe, verbesserte sich mein Können und Wissen und ich änderte die vorigen Seiten ab.

Auch für die Programmierung der Firmware wollte ich möglichst nah mit dem Prozessor zusammenarbeiten. Zu Beginn weigerte ich mich, mit dem MCC zu arbeiten, bis ich gegen Ende des Projekts auf den Geschmack von Assembly gekommen bin. Hätte es mir die Zeit erlaubt, hätte ich das gesamte Programm neu geschrieben. Wie der PIC auf dem Hardware-Level arbeitet, ist mir nun sehr vertraut. Das ist zu einem Großteil dem Datenblatt des Prozessors zu verdanken. Dieses ist mit Abstand die beste Dokumentation eines Produktes mit der ich je arbeiten durfte. Besonders spannend waren die einzelnen Module, mit denen ich zu arbeiten hatte. Um möglichst effiziente Treiber für diese Module zu programmieren, habe ich mich besonders viel mit Interrupts auseinandergesetzt. Auch um die Unabhängigkeit des UART-Treibers zu gewährleisten, spielten die Interrupts eine zentrale Rolle. Besonders gefallen hat mir die Implementierung der Ring-Buffer. Ich habe einige spannende Fakten über verschiedenen Integer-Größen gelernt. Darunter fällt unter anderem das Rechnen mit 8Bit unsigned Integer welche z.B. während einer Subtraktion zu normalen signed Integer angehoben werden.

Auch die Art und Weise wie ich meinen Code strukturiere und kommentiere, hat sich stark verändert. Ich habe einen besseren Umgang mit den verschiedenen Fassetten der Datentypen erlernt. Mein Code ist durchdacht strukturiert und alle Segmente sind so kommentiert, dass ich sie auch in 10 Jahren noch verstehe.

Was ich bei meinen nächsten Projekten jedoch besser machen werde, ist, wo ich bestimmte Dinge teste. Ich habe für die Dokumentation viele Code-Beispiele benötigt und diese in dem Hauptprogramm erstellt. Mir fiel es anschließend schwer alle Änderungen zu finden und diese zu beheben. Ebenso will ich mir in Zukunft, bevor ich mit dem Programmieren beginne, mehr Gedanken machen, wie ich ein bestimmtes Problem löse. Mir ist kurz vor der Abnahme aufgefallen, dass der Aufbau meiner Main-Routine durch Verwendung einer State-Machine deutlich lesbarer und übersichtlicher gemacht hätte werden können.

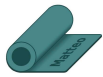
Auch die Dokumentation war eine große Freude zu verfassen. Ich habe mir noch einmal viele Gedanken zu den einzelnen Themen machen können. Dinge, die ich während



der Umsetzung nicht verstanden habe, konnte ich, während dem Schreiben des Buches noch einmal durchdenken und recherchieren. Mir war es ein Anliegen all dieses Wissen, das ich mir über die letzten Monate angeeignet habe, an die nächsten Jahrgänge weiterzugeben. Es fiel mir auch sehr schwer mich nicht in jedem kleinsten Detail zu verlieren. Einen 600 Wörter langen Absatz, den ich schlussendlich wieder entfernt habe, in dem ich den kleinen jedoch höchst spannenden Fakt des Array-to-Pointer-decay's erklärt habe. Ich wollte mich gegen Ende dann etwas zurückhalten, aber der Fakt ist, ich hätte statt 8000 auch 10 bis 15000 Wörter schreiben können und es hätte nur noch mehr Spaß gemacht.

Dieses Projekt war jedoch auch ein weiterer Beweis, dass ich meine Aufgaben so früh wie möglich abschließen sollte. Anfangs hat das auch gut funktioniert, ich war weit voraus mit meinen Aufgaben. Ich muss bei zukünftigen Projekten besser darauf aufpassen das ich den Vorsprung nicht als Ausrede verwende, um nichts zu tun und ihn somit verliere. Die Zeit wurde gegen Ende doch recht knapp.

Gegen Ende des Projektes ist es aufgrund von Stress und mangelnder Kommunikation im Team noch zu Spannungen gekommen. Ich habe gelernt, dass ich besser kommunizieren muss, was ich brauche. Ich habe sowohl mein Fachwissen also auch soziale Kompetenzen und meine Arbeit in einem Team deutlich verbessern können.



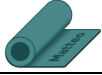
5.4 Lisa Pripfl

Zum größten Teil ist das Projekt gut verlaufen, allerdings lässt das Zeitmanagement sehr zu wünschen übrig. In den Sommerferien wurde so gut wie nichts erledigt, obwohl genügend Zeit gewesen wäre und auch im September wurde abgesehen vom Diplomarbeitsantrag nicht viel weitergebracht. Bis tatsächlich alle Bauteile und das Konzept festgestanden ist war es Ende Oktober. Dadurch wurden die darauffolgenden Monate umso stressiger. Das hätte sich leicht vermeiden lassen, wenn die Anfangszeit gut genutzt worden wäre. Auch das Testen hat viel zu spät stattgefunden, da die erste Platine, auf der sich alle Bauteile befinden erst im Jänner bestellt wurde. Da diese allerdings zuerst im Zoll festhing und dann Fehler beinhaltete war die Endgültige Platine erst zwei Wochen vor Abnahme fertig. So war für die Software viel zu wenig Zeit, um genügend zu Testen und etwaige Fehler auszubessern. Das hat zu unnötigem Stress geführt. Auch dass bei den Sensoren Schwankungen gemessen werden, wurde so erst eine Woche vor Abnahme festgestellt, weshalb sehr kurzfristig eine Lösung hermusste. Die Matte wurde erst am Tag der Abnahme zum allerersten Mal aufgebaut. So wurde rückblickend gesehen das ganze Projekt viel zu wenig getestet und das Risiko, dass während der Abnahme etwas nicht funktioniert, war sehr groß. Wäre rechtzeitig begonnen worden, hätte all das vermieden werden können.

Ein weiteres Problem lag im Projektmanagement. Es wurde nicht darauf geachtet, dass jeder seinen Teil rechtzeitig erledigt. Im Endeffekt wurde jeder sich selbst überlassen und wenn jemand wochenlang nichts getan hat, wurde auch nichts gesagt. Genau das hat schlussendlich zu einem so schlechten Zeitmanagement geführt. So hat sich jeder nur für seinen Teil verantwortlich gefühlt. Dass alle Komponenten auch zusammenspielen müssen und dafür genug Zeit zum Testen und Fehersuchen benötigt wird, wurde teilweise außer Acht gelassen. Daraus habe ich gelernt, dass es wichtig ist nicht nur seinen Teil zu erledigen, sondern auch andere dazu zu bringen ihren Teil zu machen. Außerdem habe ich gelernt, dass nur weil alle Teile einzeln funktionieren, es nicht heißt, dass auch das Gesamte am Ende funktioniert und deshalb immer genug Zeit zum Testen eingeplant werden muss.

Auch die Kommunikation war nicht ideal. Die, die für die Hardware zuständig waren und die, die für die Software zuständig waren haben untereinander viel kommuniziert. Allerdings im gesamten Team wusste man oft nicht, wie es gerade ums Projekt steht. Meist habe ich von der Hardware nur das mitbekommen, was im Jour Fixe besprochen wurde. So war mir auch sehr lange nicht bewusst, dass es in der Hardware zu solchen Verzögerungen gekommen ist.

Aber nicht nur auf das gesamte Team bezogen ist nicht alles gut verlaufen. Auch bei meinem Teil gab es Probleme. Wenn ich das Projekt noch mal machen würde, würde ich nur Mikrocontroller verwenden, die gut dokumentiert sind. Das ist beim Arduino Nicla Voice nicht der Fall. Es gibt ein Datenblatt, das ist aber, wie alle Arduino Datenblätter, sehr kurz und enthält nur sehr grobe Informationen. Zusätzlich ist dieser Mikrocontroller noch relativ neu auf dem Markt, weshalb es wenige Benutzererfahrungen gibt, von denen man profitieren kann. So verbringt man mit eigentlich recht einfachen



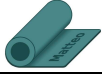
Problemen oft Stunden. Zum Beispiel wurden über 7 Stunden fürs erste Mal flashen benötigt, nur weil man einmal kurz zum richtigen Zeitpunkt den Reset-Button drücken muss, was allerdings nirgends dokumentiert war. Auch der ESP32 hat im Vergleich zum PIC eine sehr schlechte Dokumentation. Allerdings gibt es dort sehr viele Foren, die bei allen möglichen Fragen weiterhelfen können.

Das dadurch oft Probleme aufgetreten sind, zu denen lange keine Lösung gefunden wurde hat sich auch stark auf die Motivation und den Spaß am Projekt ausgewirkt. So sind dann auch immer wieder Flüchtigkeitsfehler passiert, da ich so schnell wie möglich fertig werden wollte. Diese hätten alle leicht vermieden werden können, wenn ich mir die nötige Zeit genommen hätte. Es sind aber nicht nur mehr Fehler passiert, sondern ich habe auch einfach die erste Lösung genommen, die mir eingefallen ist, ohne darüber nachzudenken, ob das eine gute Lösung ist. So musste ich auch am Ende das ganze Programm überarbeiten, da es unnötig kompliziert geschrieben wurde und sehr schlecht zu warten und zu lesen war.

Außerdem werde ich in Zukunft mehr auf die Benennung von Variablen achten. Ich habe oft einfach irgendeine Benennung genommen und bin am Ende draufgekommen, dass es dieselbe Variable zwei Mal gibt, nur einmal mit einem Großbuchstaben und einmal alles klein geschrieben. Das trägt dazu bei, dass einfach Fehler passieren, da man die Variablen vertauscht und der Code schwer zu lesen wird. Außerdem sind einige Variablen so benannt gewesen, dass überhaupt nicht hervorgeht, was sie tun. So habe ich selbst nach den Weihnachtsferien teilweise nicht mehr verstanden, was ich mir beim Code schreiben gedacht habe. Deshalb nehme ich mir auch vor Code gleich zu kommentieren und nicht erst dann, wenn ich es für die Dokumentation brauche.

Wenn ich noch einmal eine Diplomarbeit machen würde, würde ich einen Aufgabenbereich wählen, der mich wirklich interessiert. Ich war von Anfang an nicht begeistert davon eine Sprachsteuerung programmieren zu müssen und so hatte ich nie wirklich Spaß an dem Projekt. Ich habe mich von Beginn an darauf gefreut, wenn es fertig ist. Das hat sich auch sehr negativ auf meine Motivation und meine Stimmung dem Projekt gegenüber ausgewirkt. Dadurch habe ich, wenn etwas nicht nach Plan gelaufen ist, schnell meinen Teamkollegen die Schuld gegeben, obwohl ich auch nicht alles richtig gemacht habe. Das hat vor allem gegen Ende, in Kombination mit einem hohen Stresslevel zu Spannungen im Team geführt.

Zusammenfassend kann man sagen, dass trotz einiger Komplikationen alles gut verlaufen ist und es größtenteils keine Unstimmigkeiten zwischen den Teammitgliedern gab, was die Zusammenarbeit Projektablauf trotz allem sehr angenehm gemacht hat. Um in der Zukunft einen noch besseren zu gewährleisten, sollte allerdings auf ein besseres Zeitmanagement geachtet werden.

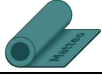


6 Literaturverzeichnis

- [1] „w3schools,“ [Online]. Available:
<https://www.w3schools.com/tags/default.asp>. [Zugriff am 10 3 2026].
- [2] „Datenblatt-PIC16F1827,“ [Online]. Available:
<https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/41391D.pdf>. [Zugriff am 7 10 2025].
- [3] „Datenblatt-16-Channel-MUX,“ [Online]. Available:
<https://www.ti.com/lit/ds/symlink/cd74hc4067.pdf>. [Zugriff am 8 10 2025].
- [4] „Datenblatt-BSS138,“ [Online]. Available:
<https://www.onsemi.com/download/data-sheet/pdf/bss138-d.pdf>. [Zugriff am 16 12 2025].
- [5] „Datenblatt-OPV,“ [Online]. Available:
<https://ww1.microchip.com/downloads/aemDocuments/documents/MSLD/ProductDocuments/DataSheets/MCP6001-1R-1U-2-4-1-MHz-Low-Power-Op-Amp-DS20001733L.pdf>. [Zugriff am 8 10 2025].
- [6] „Datenblatt Lademodul,“ [Online]. Available:
<https://static.chipdip.ru/lib/977/DOC002977110.pdf>. [Zugriff am 29 10 2025].
- [7] „Datenblatt-SD-Karten-Modul,“ [Online]. Available:
<https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/749/ProductDocuments/DataSheets/41363A.pdf>. [Zugriff am 7 11 2025].
- [8] „Datenblatt-PIC16F1827-Kurzfassung,“ [Online]. Available:
<https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/41363A.pdf>. [Zugriff am 17 3 2026].
- [9] „Datenblatt-ESP32-Wroom-32,“ espressif, [Online]. Available:
https://documentation.espressif.com/esp32-wroom-32_datasheet_en.pdf. [Zugriff am 18 11 2025].
- [10] „Datenblatt-Arduino-Nicla-Voice,“ [Online]. Available:
<https://docs.arduino.cc/resources/datasheets/ABX00061-datasheet.pdf>. [Zugriff am 15 10 2025].

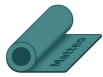
- [11] „Datenblatt-Audioverstärker,“ [Online]. Available: <https://cdn-learn.adafruit.com/downloads/pdf/adafruit-max98357-i2s-class-d-mono-amp.pdf>. [Zugriff am 18 11 2025].
- [12] „Loadcells,“ [Online]. Available: https://en.wikipedia.org/wiki/Load_cell. [Zugriff am 11 3 2026].
- [13] „FSR,“ [Online]. Available: <https://lastminuteengineers.com/fsr-arduino-tutorial/>. [Zugriff am 11 3 2026].
- [14] „DMS,“ [Online]. Available: <https://de.wikipedia.org/wiki/Dehnungsmessstreifen>. [Zugriff am 11 3 2026].
- [15] „Wheatstone-brücken,“ [Online]. Available: <https://www.electronicstutorials.ws/blog/wheatstone-bridge.html#:~:text=The%20Wheatstone%20Bridge%20is%20a,way%20to%20measure%20a%20resistance>. [Zugriff am 11 3 2026].
- [16] „OPV-Komperatorschaltung,“ [Online]. Available: <https://www.electronicstutorials.ws/de/operationsverstarker/opamp-komparator.html>. [Zugriff am 11 3 2026].
- [17] „Bild-PIC16F1827,“ [Online]. Available: <https://media.microchip.com/silicon-devices/medium/pic16f1827-f3x.png>. [Zugriff am 11 3 2026].
- [18] „Bild-MCP6002,“ [Online]. Available: https://media.rs-online.com/image/upload/bo_1.5px_solid_white,b_auto,c_pad,dpr_2,f_auto,h_399,q_auto,w_710/c_pad,h_399,w_710/Y0352239-01?pgw=1. [Zugriff am 11 3 2026].
- [19] „Bild-MUX,“ [Online]. Available: <https://www.ti.com/content/dam/ticom/images/products/package/d/db0024a.png:large>. [Zugriff am 11 3 2026].
- [20] „Bild-Logikpegelwandler-Schaltung,“ [Online]. Available: <https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQP9KGirnmez7B7gb8UyBPBEmeZ68YLB8WYVQ&s>. [Zugriff am 12 3 2026].
- [21] „Bild-ESP32,“ [Online]. Available: : <https://www.az-delivery.de/cdn/shop/files/esp32-nodemcu-module-wlan-wifi-dev-kit-c-development-board-mit-cp2102-und-usb-c-anschluss-esp-32-esp32-wroom-32-kompatibel-mit-arduino-830819.jpg?v=1732745705&width=1200>. [Zugriff am 27 1 2026].

- [22] „Logo Edge Impulse,“ [Online]. Available: https://www.edge-ai-vision.com/wp-content/uploads/2021/05/logo_edgeimpulse_may_2021.png. [Zugriff am 27.1.2026].
- [23] „Logo Arduino IDE,“ [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/7/73/Arduino_IDE_logo.svg. [Zugriff am 27.1.2026].
- [24] „Windows size und Windows increase,“ [Online]. Available: <https://mintcdn.com/edgeimpulse/cMDYCG0TzQNU7yvn/.assets/images/window-size-vs-increase.png?w=1100&fit=max&auto=format&n=cMDYCG0TzQNU7yvn&q=85&s=b666d7026463f0961b1c8e37d0fbbd6e>. [Zugriff am 27.1.2026].
- [25] „SPI,“ [Online]. Available: <https://www.kasuo.com/wp-content/uploads/2025/04/Serial-Peripheral-Interface-Bus.jpg>. [Zugriff am 27.1.2026].
- [26] „I2S,“ [Online]. Available: https://www.allaboutcircuits.com/uploads/articles/introduction-to-the-i2s-interface_rk_aac_image2.jpg. [Zugriff am 28.1.2026].
- [27] „I2C,“ [Online]. Available: https://www.mikrocontroller.net/attachment/372171/I2C_bus.pdf. [Zugriff am 28.1.2026].
- [28] „ElevenLabs,“ [Online]. Available: <https://elevenlabs.io/brand>. [Zugriff am 28.1.2026].
- [29] „Code Arduino Nicla Voice,“ [Online]. Available: https://github.com/Jallson/Voice_Controlled_PowerPlug/blob/main/relayvoice_niclavoiced.ino. [Zugriff am 28.1.2026].
- [30] „Firmware Arduino Nicla Voice,“ [Online]. Available: <https://docs.arduino.cc/tutorials/nicla-voice/user-manual/>. [Zugriff am 10.2.2026].
- [31] „Jugend Innovativ,“ [Online]. Available: <https://www.jugendinnovativ.at/teilnahme/wettbewerb>. [Zugriff am 5.3.2026].
- [32] „Bild Akkus,“ [Online]. Available: Quelle: <https://www.amazon.de/Wiederaufladbarer-Akku%EF%BC%8C3-7V->

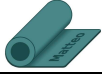


Wiederaufladbare-Scheinwerfer-Taschenlam-
pen/dp/B0DCK4PTXS/ref=sr_1_7?__mk_de_DE=%C3%85M%C3%85%C5%B
D%C3%95%C3%91&crd=3VM7QYNB31ELA&dib=eyJ2IjoiMSJ9.DSxtFaumGE
6cZKTY7hs1cuUDYHhTt84a-xYs8dTW. [Zugriff am 10 3 2026].

- [33] „Bild Akku-Lademodul,“ [Online]. Available: Quelle:
[https://www.amazon.de/Gebildet-Eingangsschnittstelle-Ladeplatine-Lademodul-Schutzfunktio-
on/dp/B0C3V1NC7T/ref=sr_1_1_sspa?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crd=2XIHXY3P9573Z&dib=eyJ2IjoiMSJ9.Yr36Qk2bu28NUqULgl49MyfpBkb2UFJ2gz_ERabnxzx4bd](https://www.amazon.de/Gebildet-Eingangsschnittstelle-Ladeplatine-Lademodul-Schutzfunktio- on/dp/B0C3V1NC7T/ref=sr_1_1_sspa?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crd=2XIHXY3P9573Z&dib=eyJ2IjoiMSJ9.Yr36Qk2bu28NUqULgl49MyfpBkb2UFJ2gz_ERabnxzx4bd). [Zugriff am 10 3 2026].
- [34] „Bild Step-Up-Converter,“ [Online]. Available: Quelle:
<https://www.amazon.de/Converter-Spannungswandler-LAOMAO-Netzteil-Kompatibel/dp/B0B932BR7V>. [Zugriff am 10 3 2026].
- [35] „Bild Step-Down-Converter,“ [Online]. Available: Quelle:
[https://www.amazon.de/BUNTOR-Spannungsregler-
Abw%C3%A4rtswandler-Stromversorgung-Elektronikprojekte/dp/B0F5W7C4KX/ref=sr_1_6?dib=eyJ2IjoiMSJ9.rp7MI-
wyWgQ3Y92L7Updd-psgWZWWYICmLm-T-
JPS7voXvORrh0tY6hYGsePM9yid0jB4aNAbLrcagi6dzCq86l_s2984svt048DXBl
x](https://www.amazon.de/BUNTOR-Spannungsregler- Abw%C3%A4rtswandler-Stromversorgung-Elektronikprojekte/dp/B0F5W7C4KX/ref=sr_1_6?dib=eyJ2IjoiMSJ9.rp7MI- wyWgQ3Y92L7Updd-psgWZWWYICmLm-T- JPS7voXvORrh0tY6hYGsePM9yid0jB4aNAbLrcagi6dzCq86l_s2984svt048DXBl x). [Zugriff am 10 3 2026].
- [36] „Bild-Gymnastikmatte,“ [Online]. Available: [https://m.media-
amazon.com/images/I/810LvciXWsL_AC_SL1500_.jpg](https://m.media-amazon.com/images/I/810LvciXWsL_AC_SL1500_.jpg). [Zugriff am 18 3 2026].
- [37] „Bild - Visual Studio Code,“ [Online]. Available: <https://code.visualstudio.com>. [Zugriff am 18 3 2026].
- [38] „Bild - Live-Server,“ [Online]. Available:
[https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServe
r](https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer). [Zugriff am 18 3 2026].
- [39] „H-Term,“ [Online]. Available: [https://www.der-
hammer.info/pages/terminal.html](https://www.der-hammer.info/pages/terminal.html). [Zugriff am 18 3 2026].
- [40] „Bild-DMS,“ [Online]. Available: [https://midebien.com/wp-
content/uploads/2018/12/serie-C-LC11-min.jpg](https://midebien.com/wp-content/uploads/2018/12/serie-C-LC11-min.jpg). [Zugriff am 13 3 2026].



- [41] „Bild-Load Cell,“ [Online]. Available: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQcpBLTfrCexkv2r_HsK3NfS4_mY_pb8m4dPw&s. [Zugriff am 13 3 2026].
- [42] „Bild-FSR,“ [Online]. Available: https://img.fruugo.com/product/0/99/1737349990_0340_0340.jpg. [Zugriff am 13 3 2026].
- [43] „Bild-FSR-Aufbau,“ [Online]. Available: <https://lastminuteengineers.com/wp-content/uploads/arduino/How-FSRs-Made.png>. [Zugriff am 13 3 2026].
- [44] „Bild-FSR Sensor Graph,“ [Online]. Available: <https://www.tekscan.com/sites/default/files/Figure2-ChangeInResistance.jpg>. [Zugriff am 13 3 2026].
- [45] „Bild-Arduino Nicla Voice,“ [Online]. Available: https://cdn-reichelt.de/resize/600%2F-/web/xxl_ws/R600%2F3840-2160-MAX.png?type=ProductXxl&resize=600%252F-&. [Zugriff am 13 3 2026].
- [46] „Bild-SD-Karten-Modul,“ [Online]. Available: https://cdn-reichelt.de/bilder/web/xxl_ws/A300%2F410-380_1.png?type=ProductXxl&. [Zugriff am 13 3 2026].
- [47] „Bild-Lautsprecher,“ [Online]. Available: https://m.media-amazon.com/images/I/51EjA9akv5L_AC_SL1200_.jpg. [Zugriff am 13 3 2026].
- [48] „Bild-Audioverstärker,“ [Online]. Available: <https://robu.in/wp-content/uploads/2022/06/27.jpg>. [Zugriff am 13 3 2026].
- [49] „Bild-Logikpegelwandler,“ [Online]. Available: <https://etusker.com/wp-content/uploads/2021/04/4-channel-IIC-I2C-Logic-Level-Converter-Bi-Directional-Module-5V-to-3.3V-01.jpg>. [Zugriff am 13 3 2026].
- [50] „Bild-LEDs,“ [Online]. Available: https://m.media-amazon.com/images/I/51Sw7iHSHuL_AC_SL1100_.jpg. [Zugriff am 13 3 2026].
- [51] „Bild-Komperator Erklärung,“ [Online]. Available: <https://www.electronicstutorials.ws/wp-content/uploads/2018/05/opamp-opamp103.gif>. [Zugriff am 10 03 2026].
- [52] „Bild-Rot-Grün-LED,“ [Online]. Available: <https://www.pollin.at/thumbnail/35/g0/96/1700579436/121956-1->



kingbright-duo-led-l-3vegw-rot-gruen-3-mm-t-1-617-568-nm-30-40-mcd-60-rund_800x800.jpg?ts=1700582239. [Zugriff am 13 3 2026].

[53] „Bild-Einschmelzgewinde,“ [Online]. Available: <https://m.media-amazon.com/images/I/91zsEZ6jNEL.jpg>. [Zugriff am 18 3 2026].



7 Tabellenverzeichnis

Tabelle 1: Objektstrukturplan.....	8
Tabelle 2: Projektstrukturplan	9
Tabelle 3: Risikoanalyse.....	15
Tabelle 4: Stakeholder-Tabelle	17
Tabelle 5:Kostenschätzung	19
Tabelle 6: Tatsächliche Kosten	19
Tabelle 7: Meilensteine.....	21
Tabelle 8: Stromverbrauch Schätzung.....	33
Tabelle 9: Stromverbrauch Berechnung.....	33
Tabelle 10: Akkustand Anzeige	56
Tabelle 11: HTML-Tags.....	82
Tabelle 12: HTML-Attribute	83
Tabelle 13: CSS-Eigenschaften.....	85
Tabelle 14: Baudrate-Konfiguration.....	96
Tabelle 15: Befehle	117



8 Abbildungsverzeichnis

Abbildung 1 Diplomarbeitsteam.....	2
Abbildung 2 Laurenz Wimmer	3
Abbildung 3 Nandor Bagari	4
Abbildung 4 Mika Böhm.....	5
Abbildung 5 Lisa Pripfl	6
Abbildung 6 3D-Modell Matte Idee.....	7
Abbildung 7 Projektorganisation	8
Abbildung 8 Risikoportfolio.....	16
Abbildung 9 Stakeholder-Portfolio	18
Abbildung 10 Kostenvergleich.....	20
Abbildung 11 Sponsorings	20
Abbildung 12 Visitenkarte vorne.....	22
Abbildung 13 Visitenkarte hinten	22
Abbildung 14 Instagram-Profil.....	22
Abbildung 15 Plakat.....	23
Abbildung 16 Arbeitsstunden	25
Abbildung 17 Blockbild	26
Abbildung 18 Akkus [29]	28
Abbildung 19 Schalter.....	28
Abbildung 20 Lademodul Maße [30]	29
Abbildung 21 Datenblattauszug Beschaltung Chip [34].....	29
Abbildung 22 Schaltsymbol Lademodul	30
Abbildung 23 Step-Up Schaltsymbol.....	30
Abbildung 24 Step-Up Maße [31]	30
Abbildung 25 Step-Down Maße [32].....	31
Abbildung 26 Step-Down Schaltsymbol.....	31
Abbildung 27 Schaltplan Stromversorgung	32
Abbildung 28 Stromversorgung Platine	32
Abbildung 29 Stromverbrauch Schätzung.....	33



Abbildung 30 Stromverbrauch Berechnung	33
Abbildung 31 Schaltplan Messschaltung	34
Abbildung 32 MUX (CD74HC4067) [11]	34
Abbildung 33 μC (PIC16F1827) [9]	35
Abbildung 34 Messschaltung MUX	36
Abbildung 35 MUX Detail mit Messwiderstand	36
Abbildung 36 Oszi 47 k Ω	37
Abbildung 37 Oszi 1 k Ω	37
Abbildung 38 Oszi 10 k Ω	37
Abbildung 39 Messschaltung PIC	38
Abbildung 40 Bild Messplatine	39
Abbildung 41 Messplatine Vorversion 1	40
Abbildung 42 Messplatine Vorversion 2	41
Abbildung 43 DMS [1]	42
Abbildung 44 Load Cell [38]	42
Abbildung 45 FSR Aufbau [40]	43
Abbildung 46 FSR [39]	43
Abbildung 47 FSR Kraft-Widerstand [41]	43
Abbildung 48 Sensoren Lage	44
Abbildung 49 ESP32 [17]	44
Abbildung 50 Arduino Nicla Voice [42]	45
Abbildung 51 SD-Karten Modul [43]	45
Abbildung 52 Audioverstärker [45]	46
Abbildung 53 Lautsprecher [44]	46
Abbildung 54 Logikpegelwandler [46]	47
Abbildung 55 Logikpegelwandler Schaltung [12]	47
Abbildung 56 LEDs [47]	48
Abbildung 57 Platinendesign gesamt	49
Abbildung 58 Platinendesign Stromversorgung	49
Abbildung 59 Platine Stromversorgung	49
Abbildung 60 Platinendesign Messschaltung	51
Abbildung 61 Platine Messschaltung	51



Abbildung 62 Platine Sprachsteuerung.....	52
Abbildung 63 Platinendesign Sprachsteuerung.....	52
Abbildung 64 Akkustandsanzeige verbaut	53
Abbildung 65 Lautsprecher verbaut	53
Abbildung 66 Platine nachträgliche Verbesserung	54
Abbildung 67 Gesamte Platine.....	54
Abbildung 68 Akkustand Anzeige Diagramm	56
Abbildung 69 Rot-Grün-LED [49]	56
Abbildung 70 OPV [10]	57
Abbildung 71 Komparator Erklärung [48]	57
Abbildung 72 Schaltplan Akkustand Anzeige	58
Abbildung 73 Platine Akkustand Anzeige oben.....	58
Abbildung 74 Platine Akkustand Anzeige unten	58
Abbildung 75 Gesamte Matte Render	59
Abbildung 76 Gesamtes Gehäuse Render.....	60
Abbildung 77 Boden + Platine Render	61
Abbildung 78 Boden Auflageflächen Render	61
Abbildung 79 Boden Durchgangslöcher Render	61
Abbildung 80 Schalter eingebaut Render.....	62
Abbildung 81 Einschmelzgewinde [50]	63
Abbildung 82 Einschmelzgewinde eingeschmolzen	63
Abbildung 83 Gesamt Boden Render	63
Abbildung 84 Deckel Gesamt Render.....	64
Abbildung 85 USB-C Detail Render.....	64
Abbildung 86 Zacken Detail Render.....	65
Abbildung 87 Schalter Detail Render.....	66
Abbildung 88 Lautsprecher eingebaut.....	66
Abbildung 89 Details Gehäuse rechts Render	67
Abbildung 90 Gehäuse mit Platine	67
Abbildung 91 Gehäuse Vorversion.....	67
Abbildung 92 Gymnastikmatte [33]	68
Abbildung 93 Matte von unten offen.....	68



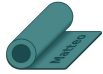
Abbildung 94 Matte gesamt.....	69
Abbildung 95 Platte Konzept.....	70
Abbildung 96 Platte Render.....	70
Abbildung 97 Platte Creo Simulation	72
Abbildung 98 Platte Creo Simulation Graph	72
Abbildung 99 Unterlage mit Sensor.....	73
Abbildung 100 Unterlage Render.....	73
Abbildung 101 Platte Vorversion	74
Abbildung 102 Platine Render Sprachsteuerung und -ausgabe.....	75
Abbildung 103 Platine Render.....	75
Abbildung 104 Platine Render Messschaltung.....	76
Abbildung 105 Platine Render Stromversorgung.....	76
Abbildung 106: Live-Server [36]	77
Abbildung 107: Visual Studio Code [35]	77
Abbildung 108: Header.....	78
Abbildung 109: Footer	78
Abbildung 110: Smartphone Header.....	78
Abbildung 111: Smartphone Header ausgeklappt.....	79
Abbildung 112: Konzeptskizze	80
Abbildung 113: Meilensteine	80
Abbildung 114: Link hover.....	85
Abbildung 115: Link	85
Abbildung 116: Blockbild	86
Abbildung 117: H-Term [37].....	87
Abbildung 118: ADC Block Diagramm [3].....	88
Abbildung 119: Messschaltung.....	89
Abbildung 120: IF-Anweisung	90
Abbildung 121: MUX ansteuern	90
Abbildung 122: ADC-Initialisierung	91
Abbildung 123: GPIO-Initialisierung.....	94
Abbildung 124: Baud-Rate-Formeln [3]	95
Abbildung 125: Sendevorgang [3].....	97



Abbildung 126: Empfangsvorgang [3].....	98
Abbildung 127: UART-Initialisierung	99
Abbildung 128: Ring-Buffer leer	101
Abbildung 129: Ring-Buffer-Initialisierung.....	101
Abbildung 130: Ring-Buffer beschrieben.....	102
Abbildung 131: Ring-Buffer ausgelesen.....	102
Abbildung 132: Ring-Buffer voll	103
Abbildung 133: Ring-Buffer auslesen und beschreiben	103
Abbildung 134: String-Literal	106
Abbildung 135: Hauptprogramm	108
Abbildung 136: I2C Übertragung [26]	113
Abbildung 137: ESP32 Programm char zu Datatype	115
Abbildung 138: I2C Übertragung.....	115
Abbildung 139: User Interface.....	116
Abbildung 140: Flussdiagramm ESP32	120
Abbildung 141: Eleven Labs [27].....	121
Abbildung 142: SPI Verdrahtung [24]	121
Abbildung 143: I2S Übertragung [25]	122
Abbildung 144: ESP32 Programm Datei ausgeben.....	125
Abbildung 145: Edge Impulse Logo [21]	125
Abbildung 146: Daten aufnehmen	126
Abbildung 147: edge-impulse-daemon.....	127
Abbildung 148: Daten über Arduino aufnehmen	127
Abbildung 149: Daten über den Laptop aufnehmen.....	128
Abbildung 150: Impulse erstellen	128
Abbildung 151: Window size und Window increase [23]	129
Abbildung 152: Mel-Energie-Diagramm.....	130
Abbildung 153: Syntiant-Spektrogramm	131
Abbildung 154: (Log-)Mel-filterbank energy features	132
Abbildung 155: Transfer Learning Parameter	133
Abbildung 156: Confusion Matrix.....	134
Abbildung 157: Live Classification.....	136



Abbildung 158: Deployment.....	137
Abbildung 159: Flashen erfolgreich	138
Abbildung 160: Arduino flashen	138
Abbildung 161: Flussdiagramm Arduino	139



9 Anhang

- Abnahmeprotokoll
- Meeting-Protokolle
- Begleitprotokolle
- Jugend Innovativ Bericht
- Konstruktionszeichnungen
- Schaltpläne
- Datenblätter
- Programmcode



9.1 Abnahmeprotokoll

Abnahmeprotokoll

Projekttitel: **Matteo**
Auftraggeber: **Universität Wien**
Auftragnehmer: **Laurenz Wimmer**
Schuljahr: **2025/26** Klasse: **5AM**

VERSION	DATUM	AUTORIN/AUTOR	ÄNDERUNG
V1.0	07.10.2025	Lisa Pripft	Erstellung

1	Grundlage der Abnahme	2
2	Abnahmekriterien.....	2
	Muss-Ziele.....	2
	Optionale-Ziele.....	5

1 Grundlage der Abnahme

Die gegenständliche Abnahme basiert auf dem Vertrag vom 19.09.2025, da_antrag_wimmer und der folgenden Change Requests: (Dokumente alle mit Bezeichnung nennen!)

2 Abnahmekriterien

Muss-Ziele

Ziel	Zielerreichung
Ziel-H 1 Elektronik a. Platine und Sensoren Eine Platine verbindet alle elektronischen Komponenten, Sensoren und Schnittstellen mit dem Mikrocontroller. Drucksensoren sind in der Matte verbaut und sind funktionsfähig mit der Platine verbunden, sodass mithilfe einer Messschaltung auf der Platine Messungen vorgenommen werden können.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Nandor Bagari) Augenscheinliche Kontrolle, dass eine bestückte Platine vorhanden ist. Augenscheinliche Kontrolle, dass Drucksensoren in der Matte vorhanden sind. Messung mit dem Multimeter, dass die Sensoren mit der Messschaltung verbunden sind.	
Ziel-H 1 Elektronik b. Spannungsversorgung Die Elektronik wird durch Akkus mit Spannung versorgt, die eine Mindestbetriebszeit von 1 Stunde gewährleisten. Die Akkus werden über einen USB-C-Adapter mithilfe eines Netzgeräts geladen, das direkt an eine Steckdose angesteckt werden kann.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Laurenz Wimmer) Augenscheinliche Kontrolle, dass mindestens ein Akku und ein USB-C-Adapter vorhanden ist. Messung mit dem Multimeter, dass die Platine mit Spannung versorgt wird. Augenscheinliche Kontrolle, dass Lade-LED beim Ladevorgang leuchtet.	
Ziel-H 1 Elektronik c. Peripherie Es sind mind. eine LED zur Anzeige des Akkuladestands, ein Schalter zum Ein- und Ausschalten der Matte sowie ein Taster zum Resetten der Matte verbaut.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Nandor Bagari) Augenscheinliche Kontrolle, dass mind. eine LED und ein Taster vorhanden ist.	

Ziel-H 2 Konstruktion und Aufbau a. Matte Die mind. 900 x 600 mm große Matte hat ein passendes Material für die Oberseite und Unterseite, um eine angenehme Oberfläche für Benutzer:innen und Schutz für die Sensorik zu bieten.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Nandor Bagari) Augenscheinliche Kontrolle, dass die Matte vorhanden ist.	
Ziel-H 2 Konstruktion und Aufbau b. Gehäuse Das Gehäuse für die elektronischen Bauteile ist an einer Seite der Matte und schützt die Elektronik vor Schäden. Es beinhaltet die Platinen, die Akkus und die Peripherie.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Nandor Bagari) Augenscheinliche Kontrolle, dass ein Gehäuse vorhanden ist, in dem sich die Elektronik befindet.	
Ziel-H 3 Microcontroller-Programmierung Es werden die Sensordaten von einem Microcontroller eingelesen und überprüft, ob die eingestellten Trainingsziele erreicht wurden. Anschließend werden die Daten aufbereitet, um sie an einer Schnittstelle auszugeben.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Mika Böhm) Die eingelesenen und die verarbeiteten Daten sind in MPLAB X einsehbar. Mit einem Oszilloskop können die Daten, die an der Schnittstelle ausgegeben werden, gemessen werden.	
Ziel-H 4 Spracheingabe a. Belastungsziele Es können Belastungsziele für den linken und rechten Fuß in Form von prozentueller Belastung gesetzt werden. Die Abstufung erfolgt in 10% Schritten.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Lisa Pripfl) Nachdem ein Sprachbefehl eingegeben wurde, kann das im Programm ausgelesen werden.	
Ziel-H 4 Spracheingabe b. Einstellungen Es gibt Befehle zum Starten und Stoppen der Übung sowie zum Einstellen der Lautstärke der Sprachausgabe.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: Nachdem der entsprechende Sprachbefehl eingegeben wurde, wird die Übung gestartet oder gestoppt bzw. ändert sich die Lautstärke. Dies kann ebenfalls im Programm ausgelesen werden.	

Ziel-H 5 Sprachausgabe a. Feedback während Übungen Während eine Übung aktiv ist, wird frühestens alle 4 Sekunden die aktuelle Belastung ausgegeben. Es wird außerdem ausgegeben, ob das Ziel erreicht wurde.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Lisa Pripfl) Auditive Kontrolle, dass das Belastungsziel ausgegeben wird.	
Ziel-H 5 Sprachausgabe b. Rückmeldung nach Einstellungen Das Starten und Stoppen von Übungen wird ausgegeben. Nachdem Einstellungen vorgenommen wurden, wird der erkannte Befehl ausgegeben, um Missverständnisse zu vermeiden.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Lisa Pripfl) Auditive Kontrolle, dass nach Änderung der Einstellungen der Befehl wiederholt wird.	
Ziel-H 6 Website Die Website dient dem Marketing. Sie macht die Projektidee, den Projektfortschritt in Form der Meilensteine und Kontaktinformationen über die Teammitglieder öffentlich zugänglich. Die Website ist aus 6 verschiedenen Seiten aufgebaut.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Mika Böhm) Augenscheinliche Kontrolle, dass eine Website mit den verschiedenen Seiten vorhanden ist.	
Ziel-H 7 Projektmanagement Das Projektmanagement wird gemäß der Projektmanagement-Vorgaben der HTL Rennweg durchgeführt. Durch regelmäßige Meetings werden die Betreuer über den Fortschritt informiert.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Laurenz Wimmer) Augenscheinliche Kontrolle, dass Meetingprotokolle, Plakate und Analysen vorhanden sind.	

Optionale-Ziele

Ziel	Zielerreichung
Ziel-O 1 Lichtanzeige a. Hardware Eine auf der Matte sichtbare Lichtanzeige in Form von LEDs zeigt an, ob das eingestellte Ziel erreicht wurde.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Laurenz Wimmer) Augenscheinliche Kontrolle, dass LEDs vorhanden und mit der Platine verbunden sind.	
Ziel-O 1 Lichtanzeige b. Programmierung Die Lichtanzeige ist so programmiert, dass sie anzeigt, ob das Ziel erreicht wurde.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Mika Böhm) In MPLAB X kann der Zustand der Pins ausgelesen werden.	
Ziel-O 2 Social Media b. Programmierung Das Projekt hat eine Social-Media-Präsenz auf mind. 1 Social-Media-Plattform.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Nandor Bagari) Augenscheinliche Kontrolle des Vorhandenseins einer Social-Media-Plattform	
Ziel-O 3 Messung ohne Zielsetzung Es gibt eine Einstellung, bei der kein Belastungsziel gesetzt wird, sondern dauerhaft die Belastung gemessen und frühestens alle 4 Sekunden ausgegeben wird.	☒ erfüllt ☐ teilweise erfüllt ☐ nicht erfüllt
Anmerkung: (Lisa Pripfl) Auditive und augenscheinliche Kontrolle, ob die Einstellung funktionsfähig ist.	
Ziel-O 4 Englische Version der Website Auf der Website kann zwischen Deutsch und Englisch gewechselt werden.	☐ erfüllt ☐ teilweise erfüllt ☒ nicht erfüllt
Anmerkung: (Mika Böhm) Augenscheinliche Kontrolle der englischen Version der Website.	

Ziel-O 5 Haptisches Feedback a. Hardware In der Matte wird ein Vibrationsmotor verbaut.	☐ erfüllt ☐ teilweise erfüllt ☒ nicht erfüllt
Anmerkung: (Nandor Bagari) Augenscheinliche Kontrolle, dass ein Vibrationsmotor vorhanden ist.	
Ziel-O 5 Haptisches Feedback b. Programmierung Der Vibrationsmotor wird so programmiert, dass er dann vibriert, wenn das Ziel erreicht wurde.	☐ erfüllt ☐ teilweise erfüllt ☒ nicht erfüllt
Anmerkung: (Mika Böhm) In MPLAB X kann der Zustand der Pins ausgelesen werden.	

Hiermit bestätigt die/der Unterzeichnende, dass das beauftragte Projekt Matteo vom Projektteam ~~beauftragte Projektleiter/Projektteam~~ an den Auftraggeber übergeben wurde.

PL

Laurenz Wimmer
Unterschrift

PA
Unterschrift

Wien 3.8.2026

Ort und Datum der Unterzeichnung



9.2 Meeting-Protokolle

Protokoll

Projekttitel: **Matteo**
Auftraggeber*in: **Sportuniversität Wien**
Auftragnehmer*in: **Laurenz Wimmer**
Schuljahr: **2025/26** Klasse: **5AM**

VERSION	DATUM	AUTORIN/AUTOR	ÄNDERUNG
v1.0	08.09.2025	Laurenz Wimmer/Nandor Bagari	Erstellung

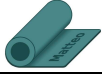
Thema/Ziel(e) des Meetings:	Website, Sensoren und Sprachsteuerung				
Datum:	08.09.2025	Uhrzeit (von-bis):	11:35 - 12:25	Ort:	AUT-Lab.
Teilnehmer*innen (Kürzel, ohne Titel)	Position / Funktion / Aufgabenbereich				
Laurenz Wimmer	Projektleiter				
Nandor Bagari	Mitarbeiter				
Mika Böhm	Mitarbeiterin				
Lisa Pripfl	Mitarbeiterin				
Prof. Drechsler (DRE)	Projektbetreuer				
Prof. Meschik (MSK)	Projektbetreuer				
Prof. Cekan (CEK)	Projektbetreuer				

Verteiler

Tagesordnung

#	Tagesordnungspunkt Beschlüsse TODOs	Erg ²	Verantw.	Deadline
1	Website besprochen - Projektbeschreibung verkürzen: Stichwörter oben, genaue Beschreibung für Interessierte unten - Bild von Konzept einfügen - Sprache vereinheitlichen: zuerst nur englisch, später zusätzlich deutsch - Impressum	TODO TODO TODO TODO	Böhm	12.9.2025
2	Sensoren - Piezoelektrische Sensoren vorläufig ausgeschlossen - Analoge Multiplexer besprochen - Genauigkeit von Positionierung mit Uni klären - Kontakte von Sensoren schützen	E-B E-I TODO E-E	Wimmer	9.9.2025

1) Vorbereitung (Agenda): V-S-Schilderung (mündlich), V-H-Handout, V-P-Präsentation, V-D-Dokument (digital), V-M-Multimedia
2) Ergebnis (Protokoll): E-B-Beschluss, E-E-Empfehlung, E-I-Information, E-O-Offener Punkt, E-A-Abgeschlossen (erledigt), TODO-TODO
Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Projekthandbuch HTL3R

3	Material - Dünne biegesteife Platten als Möglichkeit - Sohlenmaterial als Möglichkeit	E-I E-I		-
4	Sprachsteuerung - ESP32 und fertiges Modul möglich - Microchip fragen - Meiste Module für englische Befehle konzipiert	E-I TODO E-I	MSK	10.9.2025

1) Vorbereitung (Agenda): V-S -Schilderung (mündlich), V-H -Handout, V-P -Präsentation, V-D -Dokument (digital), V-M -Multimedia
2) Ergebnis (Protokoll): E-B -Beschluss, E-E -Empfehlung, E-I -Information, E-O -Offener Punkt, E-A -Abgeschlossen (erledigt), TODO -TODO
Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Protokoll

Projekttitel:	Matteo		
Auftraggeber*in:	Sportuniversität Wien		
Auftragnehmer*in:	Laurenz Wimmer		
Schuljahr:	2025/26	Klasse:	5AM
VERSION	DATUM	AUTORIN/AUTOR	ÄNDERUNG
v1.0	30.08.2018	Laurenz Wimmer	Erstellung

Thema/Ziel(e) des Meetings:	Abklären von Fragen mit Sportuniversität		
Datum:	09.09.2025	Uhrzeit (von-bis):	11:00-11:30
		Ort:	Zoom-Meeting
Teilnehmer*innen (Kürzel, ohne Titel)	Position / Funktion / Aufgabenbereich		
Laurenz Wimmer	Projektleiter		
Lisa Pripfl	Projektmitarbeiterin		
Stefan Andreas Petter	Universität Wien		
Philipp Kornfeind	Universität Wien		

Verteiler
Teilnehmer, Mika Böhm, Nandor Bagari, CEK, MSK, DRE

Tagesordnung				
#	Tagesordnungspunkt Beschlüsse TODOs	Vorb. 1/ Erg ²	Verantw.	Deadline
1	Besprechen der Messpunkte und Sensoren-Anzahl - Zwei 3-Punkt-Auflagen pro Fuß für vorderen und hinteren Teil - 30-50 kg pro Sensor, um Sicherheit zu haben - Deutsche Firma für korrigierte Sensoren (linearerer Verlauf)	E-B E-B E-I		
2	Sprachsteuerung - Arduino <u>Portenta</u> H7 Lite als mögliches Modul für Sprachsteuerung - Ca. 15 Befehle	E-I E-B		
3	Sonstiges - LED-Streifen für optisches Feedback als optionales Ziel - Haptisches Feedback durch Rüttler auf der Matte - Drückschalter zum Einschalten in Kombination mit Summer, kein Kippschalter	E-B E-O E-B		

1) Vorbereitung (Agenda): V-S-Schilderung (mündlich), V-H-Handout, V-P-Präsentation, V-D-Dokument (digital), V-M-Multimedia
 2) Ergebnis (Protokoll): E-B-Beschluss, E-E-Empfehlung, E-I-Information, E-O-Offener Punkt, E-A-Abgeschlossen (erledigt), TODO-TODO
 Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Protokoll

Projekttitel: **Matteo**
Auftraggeber*in: **Universität Wien**
Auftragnehmer*in: **Laurenz Wimmer**
Schuljahr: **2025/26** Klasse: **5AM**

VERSION	DATUM	AUTORIN/AUTOR	ÄNDERUNG
v1.0	07.10.2025	Laurenz Wimmer	Erstellung

Thema/Ziel(e) des Meetings:		Sprachsteuerung, DA-Buch, Abnahme-Protokoll, Jour-Fixe-Termine			
Datum:	07.10.2025	Uhrzeit (von-bis):	14:00-15:00	Ort:	065
Teilnehmer*innen (Kürzel, ohne Titel)		Position / Funktion / Aufgabenbereich			
Laurenz Wimmer (WIM)		Projektleiter			
Nandor Bagari (BAG)		Stv. Projektleiter			
Mika Böhm (BOE)		Projektmitarbeiterin			
Lisa Pripfl (PRI)		Projektmitarbeiterin			
CEK		Hauptbetreuer / Individueller Betreuer			
DRE		Individueller Betreuer			
MSK		Individueller Betreuer			

Verteiler

Teilnehmer

Tagesordnung

#	Tagesordnungspunkt Beschlüsse TODOs	Vorb. ^{1/} Erg. ²	Verantw.	Deadline
1	Sprachsteuerung - Mögliche Hardware: Altes Handy, Microchip, Codec-Modul (in Labor von Prof. Meschik) - Programm für Machine Learning: Edge Impulse - Wählen einer Hardware-Option	E-I E-I TODO	14.10.	PRI
2	Jour-Fixe-Termine - Alle 2 Wochen kurz während Werkstättenunterricht (wenn mit FL. Abgeklärt)	E-B		
3	DA-Buch - Unklarheiten bezüglich des Layouts geklärt - Formeln schreiben in Word erklärt	E-I E-I		

1) Vorbereitung (Agenda): V-S-Schilderung (mündlich), V-H-Handout, V-P-Präsentation, V-D-Dokument (digital), V-M-Multimedia

2) Ergebnis (Protokoll): E-B-Beschluss, E-E-Empfehlung, E-I-Information, E-O-Offener Punkt, E-A-Abgeschlossen (erledigt), TODO-TODO

Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



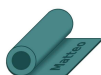
Projekthandbuch HTL3R

4	Abnahmeprotokoll - Ausgefülltes Abnahmeprotokoll besprochen - Überprüfungsmöglichkeiten verbessert	E-I E-I		
5	Stromversorgung - Mit Auslegung auf Sprachsteuerung warten - Module mit <u>Pinleisten</u> auf die Platine löten	E-B E-B		

1) Vorbereitung (Agenda): V-S-Schilderung (mündlich), V-H-Handout, V-P-Präsentation, V-D-Dokument (digital), V-M-Multimedia

2) Ergebnis (Protokoll): E-B-Beschluss, E-E-Empfehlung, E-I-Information, E-O-Offener Punkt, E-A-Abgeschlossen (erledigt), TODO-TODO

Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Agenda/Protokoll

Projekttitel: **Matteo**
Auftraggeber*in: **Universität Wien**
Auftragnehmer*in: **Laurenz Wimmer**
Schuljahr: **2025/26** Klasse: **5AM**

VERSION	DATUM	AUTORIN/AUTOR	ÄNDERUNG
v1.0	22.10.2025	Laurenz Wimmer	Erstellung

Thema/Ziel(e) des Meetings:		Messplatine, Update Sprachsteuerung, EUSART, Stromversorgung, PM			
Datum:	21.10.2025	Uhrzeit (von-bis):	15:15	Ort:	15:45
Teilnehmer*innen (Kürzel, ohne Titel)		Position / Funktion / Aufgabenbereich			
Laurenz Wimmer (WIM)		Projektleiter			
Nandor Bagari (BAG)		Stv. Projektleiter			
Mika Böhm (BOE)		Projektmitarbeiterin			
Lisa Pripfl (PRI)		Projektmitarbeiterin			
CEK		Hauptbetreuer / Individueller Betreuer			
DRE		Individueller Betreuer			
MSK		Individueller Betreuer			

Verteiler

Teilnehmer

Tagesordnung				
#	Tagesordnungspunkt Beschlüsse TODOs	Vorb. ^{1/} Erg. ²	Verantw.	Deadline
1	Messplatine vor Bestellung besprechen - Leiterbahnen breiter machen - Rest ist ok	TODO E-I	BAG	21.10.
2	Stromversorgung - Komponenten getestet - Akku geladen - Schaltplan erstellen	E-I E-I TODO	WIM	4.11.
3	Sprachsteuerung - Gewählter Microcontroller: Arduino Nicla Voice - Kommunikation zwischen Microcontrollern abklären - Schnittstelle: UART	E-I TODO E-I	PRI	4.11.

1) Vorbereitung (Agenda): V-S-Schilderung (mündlich), V-H-Handout, V-P-Präsentation, V-D-Dokument (digital), V-M-Multimedia

2) Ergebnis (Protokoll): E-B-Beschluss, E-E-Empfehlung, E-I-Information, E-O-Offener Punkt, E-A-Abgeschlossen (erledigt), TODO-TODO

Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



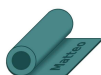
Projekthandbuch HTL3R

4	Projektmanagement - Visitenkarten begonnen - Plakat begonnen	E-I E-I		
5	Sonstiges - MUX testen - Sensor testen - Carbonplatten als Material für Messplatten, scharfkantig bei Verwendung - UART-Schnittstelle fertig - Website fertigstellen	TODO TODO E-I E-I TODO	BAG BAG BOE	22.10. 22.10. 4.11.

1) Vorbereitung (Agenda): V-S -Schilderung (mündlich), V-H -Handout, V-P -Präsentation, V-D -Dokument (digital), V-M -Multimedia

2) Ergebnis (Protokoll): E-B -Beschluss, E-E -Empfehlung, E-I -Information, E-O -Offener Punkt, E-A -Abgeschlossen (ertedigt), TODO -TODO

Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Agenda/Protokoll

Projekttitel: Matteo
Auftraggeber*in: Universität Wien
Auftragnehmer*in: Laurenz Wimmer
Schuljahr: 2025/26 **Klasse:** 5AM

VERSION	DATUM	AUTORIN/AUTOR	ÄNDERUNG
v1.0	07.10.2025	Laurenz Wimmer	Erstellung

Thema/Ziel(e) des Meetings:	TOFT, 1. Demoplatine besprechen, Sprachausgabe und Steuerung, Website			
Datum:	04.11.2025	Uhrzeit (von-bis):	14:15-14:45	Ort: ME-Labor (065)
Teilnehmer*innen (Kürzel, ohne Titel)	Position / Funktion / Aufgabenbereich			
Laurenz Wimmer (WIM)	Projektleiter			
Nandor Bagari (BAG)	Sty. Projektleiter			
Mika Böhm (BOE)	Projektmitarbeiterin			
Lisa Pripfl (PRI)	Projektmitarbeiterin			
CEK	Hauptbetreuer / Individueller Betreuer			
DRE	Individueller Betreuer			
MSK	Individueller Betreuer			

Verteiler

Teilnehmer

Tagesordnung

#	Tagesordnungspunkt Beschlüsse TODOs	Vorb. ^{1/} Erg ²	Verantw.	Deadline
1	Tag der offenen Tür - Demoplatine und Schaltung für Stromversorgung aufbauen - 1. Testprogramm, Sensor drücken -> LED leuchtet - <u>Sprachausgabe</u> wenn fertig herzeigen - Yogamatte zu Demonstrationszwecken mitnehmen	E-B TODO E-B TODO	BOE WIM	7.11.2025 7.11.2025
2	Demoplatine - Vorsicht mit Potentiometer und zu hohem Strom am Sensor - Kurze Erklärung des Aufbaus	E-I E-A		
3	Website - Smartphone-Ansicht bei 2 Seiten fertig - Restliche Seiten fertigstellen	E-I TODO	BOE	7.11.2025

1) Vorbereitung (Agenda): V-S-Schilderung (mündlich), V-H-Handout, V-P-Präsentation, V-D-Dokument (digital), V-M-Multimedia

2) Ergebnis (Protokoll): E-B-Beschluss, E-E-Empfehlung, E-I-Information, E-O-Offener Punkt, E-A-Abgeschlossen (erledigt), TODO-TODO

Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Projekthandbuch HTL3R

4	Sprachein- und -ausgabe - Fall-Back-Lösung für Sprachausgabe: Modul mit SD-Karten-Slot - Fall-Back-Lösung für Spracheingabe: Modul von Prof. Meschik - Sprachmodell einsprechen - Sprachausgabe mit ESP32 fertigstellen	E-I E-I TODO TODO	PRI PRI	1.12.2025 7.11.2025
---	--	----------------------------	------------	------------------------

1) Vorbereitung (Agenda): V-S -Schilderung (mündlich), V-H -Handout, V-P -Präsentation, V-D -Dokument (digital), V-M -Multimedia

2) Ergebnis (Protokoll): E-B -Beschluss, E-E -Empfehlung, E-I -Information, E-O -Offener Punkt, E-A -Abgeschlossen (erledigt), TODO -TODO

Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Agenda/Protokoll

Projekttitel:	Matteo		
Auftraggeber*in:	Universität Wien		
Auftragnehmer*in:	Laurenz Wimmer		
Schuljahr:	2025/26	Klasse:	5AM
VERSION	DATUM	AUTORIN/AUTOR	ÄNDERUNG
v1.0	18.11.2025	Nandor Bagari	Erstellung

Thema/Ziel(e) des Meetings:	Sprachein- und -ausgabe, Pitch-Präsentation, Konstruktion		
Datum:	18.11.2025	Uhrzeit (von-bis):	14:45-15:15
		Ort:	ME-Labor (065)
Teilnehmer*innen (Kürzel, ohne Titel)	Position / Funktion / Aufgabenbereich		
Nandor Bagari (BAG)	Stv. Projektleiter		
Mika Böhm (BOE)	Projektmitarbeiterin		
Lisa Pripfl (PRI)	Projektmitarbeiterin		
CEK	Hauptbetreuer / Individueller Betreuer		
DRE	Individueller Betreuer		
MSK	Individueller Betreuer		

Verteiler
Teilnehmer

Tagesordnung				
#	Tagesordnungspunkt Beschlüsse TODOs	Vorb. ^{1/} Erg. ²	Verant. w.	Deadline
1	Sprachein- und -ausgabe - Soundausgabe funktioniert - Erster Test der Sprachsteuerung am Arduino funktioniert schlecht - Möglicherweise Library statt fertige <u>binary</u>	E-A E-A TODO	PRI	2.12.2025
2	Website - Handyversion online	E-A		
3	Programmierung - Treiber für ADC programmiert	E-A		
4	Pitch-Präsentation - 7 Minuten pro Team - Mikrofone organisieren	E-I TODO	?	25.11.2025

1) Vorbereitung (Agenda): V-S-Schilderung (mündlich), V-H-Handout, V-P-Präsentation, V-D-Dokument (digital), V-M-Multimedia
 2) Ergebnis (Protokoll): E-B-Beschluss, E-E-Empfehlung, E-I-Information, E-O-Offener Punkt, E-A-Abgeschlossen (erledigt), TODO-TODO
 Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



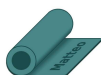
Projekthandbuch HTL3R

	- Raum noch unbekannt	E-O		
5	Platine - Reparatur der schlechten Lötstelle - Entfernung des OPVs	E-A E-A		
6	Konstruktion - Duraluminium - Vorspannung durch Biegung in die andere Richtung - Demo Platte aus Stahl testen im LA-ME(065) mit Werkstoffprüfgerät	E-E E-I TODO	BAG	16.12.2025

1) Vorbereitung (Agenda): V-S -Schilderung (mündlich), V-H -Handout, V-P -Präsentation, V-D -Dokument (digital), V-M -Multimedia

2) Ergebnis (Protokoll): E-B -Beschluss, E-E -Empfehlung, E-I -Information, E-O -Offener Punkt, E-A -Abgeschlossen (erledigt), TODO -TODO

Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Agenda/Protokoll

Projekttitel: **Matteo**
 Auftraggeber*in: **Universität Wien**
 Auftragnehmer*in: **Laurenz Wimmer**
 Schuljahr: **2025/26** Klasse: **5AM**

VERSION	DATUM	AUTORIN/AUTOR	ÄNDERUNG
v1.0	07.10.2025	Laurenz Wimmer	Erstellung

Thema/Ziel(e) des Meetings:	Sprachsteuerung, Sensor einlesen, <u>Failsafe</u>, Platine in Arbeit, JI		
Datum:	02.12.2025	Uhrzeit (von-bis):	15:00-15:40
		Ort:	L.ME(065)
Teilnehmer*innen (Kürzel, ohne Titel)	Position / Funktion / Aufgabenbereich		
Laurenz Wimmer (WIM)	Projektleiter		
Nandor Bagari (BAG)	<u>Stv.</u> Projektleiter		
Mika Böhm (BOE)	Projektmitarbeiterin		
Lisa Pripfl (PRI)	Projektmitarbeiterin		
CEK	Hauptbetreuer / Individueller Betreuer		
DRE	Individueller Betreuer		
MSK	Individueller Betreuer		

Verteiler

Teilnehmer

Tagesordnung

#	Tagesordnungspunkt Beschlüsse TODOs	Vorb. ^{1/} Erg ²	Verantw.	Deadline
1	Sprachsteuerung - Version 1 funktioniert - Befehle werden zu 90-95% erkannt - Background-Noise wird zu 40% auch als Befehl erkannt	E-I E-I E-I		
2	Sensor einlesen - Funktion von ChatGPT erstellen lassen - Bereich für Gewichtswerte angeben - Sensorkontakte mit Gummistreifen festkleben	E-E E-E E-E		
3	<u>Failsafe</u> - Eigener Befehl im Falle eines Fehlers bei einem Sensor - Über Sprachausgabe ausgeben	TODO TODO	BOE PRI	

1) Vorbereitung (Agenda): V-S-Schilderung (mündlich), V-H-Handout, V-P-Präsentation, V-D-Dokument (digital), V-M-Multimedia

2) Ergebnis (Protokoll): E-B-Beschluss, E-E-Empfehlung, E-I-Information, E-O-Offener Punkt, E-A-Abgeschlossen (ertedigt), TODO-TODO

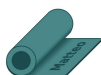
Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Projekthandbuch HTL3R

4	Sonstiges - Finale Platine in Arbeit - Jugend Innovativ und Eco-Bonus beantragt - Übertragungsprotokoll besprochen	E-I E-I E-I		
---	---	-------------------	--	--

1) Vorbereitung (Agenda): V-S -Schilderung (mündlich), V-H -Handout, V-P -Präsentation, V-D -Dokument (digital), V-M -Multimedia
2) Ergebnis (Protokoll): E-B -Beschluss, E-E -Empfehlung, E-I -Information, E-O -Offener Punkt, E-A -Abgeschlossen (erledigt), TODO -TODO
Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Agenda/Protokoll

Projekttitel: **Matteo**
 Auftraggeber*in: **Universität Wien**
 Auftragnehmer*in: **Laurenz Wimmer**
 Schuljahr: **2025/26** Klasse: **5AM**

VERSION	DATUM	AUTORIN/AUTOR	ÄNDERUNG
v1.0	17.12.2025	Laurenz Wimmer	Erstellung

Thema/Ziel(e) des Meetings:	Kommunikation zw. µC, LLC, LED-Streifen, Buck-Conv., Sensor kalibrieren			
Datum:	16.12.2025	Uhrzeit (von-bis):	15:10-15:45	Ort: L.MP (056)
Teilnehmer*innen (Kürzel, ohne Titel)	Position / Funktion / Aufgabenbereich			
Laurenz Wimmer (WIM)	Projektleiter			
Nandor Bagari (BAG)	Stv. Projektleiter			
Mika Böhm (BOE)	Projektmitarbeiterin			
Lisa Pripfl (PRI)	Projektmitarbeiterin			
CEK	Hauptbetreuer / Individueller Betreuer			
DRE	Individueller Betreuer			
MSK	Individueller Betreuer			

Verteiler

Teilnehmer

Tagesordnung				
#	Tagesordnungspunkt Beschlüsse TODOs	Vorb. ^{1/} Erg. ²	Verantw.	Deadline
1	Kommunikation zwischen den Microcontrollern - ESP32 <--> Arduino funktioniert - ESP32-->PIC funktioniert - PIC-->ESP32 funktioniert nicht - Programm kontrollieren	E-I E-I E-I TODO	PRI	17.12.25
2	Logic Level Converter - Nicht ablöten, besser MOSFETs kaufen und auflöten - Modul auch ok	E-B E-I		
3	LED Streifen - Sollten funktionieren, 2. wird getestet - Ersatzweise auch einzelne LEDs vorsehen	TODO E-B	BAG	23.12.25

1) Vorbereitung (Agenda): V-S-Schilderung (mündlich), V-H-Handout, V-P-Präsentation, V-D-Dokument (digital), V-M-Multimedia

2) Ergebnis (Protokoll): E-B-Beschluss, E-E-Empfehlung, E-I-Information, E-O-Offener Punkt, E-A-Abgeschlossen (erledigt), TODO-TODO

Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



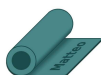
Projekthandbuch HTL3R

4	Buck-Converter - Batteriespannung zu niedrig für Modul - Buck- nach Boost-Converter - Anderes Modul verwenden - 2. Akku vorsehen und ggf. nicht bestücken	E-I E-E E-I TODO	WIM	23.12.25
5	Sensor kalibrieren - Mittelwert auf 3 Sensoren nehmen - Personen mit unterschiedlichem Gewicht zur Messung verwenden - Programm fertig	TODO E-I E-B E-I	BOE	17.12.25

1) Vorbereitung (Agenda): V-S -Schilderung (mündlich), V-H -Handout, V-P -Präsentation, V-D -Dokument (digital), V-M -Multimedia

2) Ergebnis (Protokoll): E-B -Beschluss, E-E -Empfehlung, E-I -Information, E-O -Offener Punkt, E-A -Abgeschlossen (erledigt), TODO -TODO

Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Agenda/Protokoll

Projekttitel: **Matteo**
 Auftraggeber*in: **Universität Wien**
 Auftragnehmer*in: **Laurenz Wimmer**
 Schuljahr: **2025/26** Klasse: **5AM**

VERSION	DATUM	AUTORIN/AUTOR	ÄNDERUNG
v1.0	14.01.2026	Laurenz Wimmer	Erstellung

Thema/Ziel(e) des Meetings:	Finale Platine, Kommunikation zwischen μC, Diplomarbeitsbuch, Platten, <u>TofT</u>				
Datum:	13.01.2026	Uhrzeit (von-bis):	15:15	Ort:	15:45
Teilnehmer*innen (Kürzel, ohne Titel)	Position / Funktion / Aufgabenbereich				
Laurenz Wimmer (WIM)	Projektleiter				
Nandor Bagari (BAG)	<u>Stv.</u> Projektleiter				
Mika Böhm (BOE)	Projektmitarbeiterin				
Lisa Pripfl (PRI)	Projektmitarbeiterin				
CEK	Hauptbetreuer / Individueller Betreuer				
DRE	Individueller Betreuer				
MSK	Individueller Betreuer				

Verteiler

Teilnehmer

Tagesordnung

#	Tagesordnungspunkt Beschlüsse TODOs	Vorb. ^{1/} Erg. ²	Verantw.	Deadline
1	Platine - Finale Version fertiggestellt - Platine bestellt - Bis <u>TofT</u> bestücken	E-I E-I TODO	BAG	23.1.2026
2	Kommunikation zwischen μ C - Kommunikation funktioniert grundsätzlich - Spannungsspitzen sind nicht mehr da - Kleine Fehler im Übertragungsprotokoll, bis <u>TofT</u> ausbessern	E-I E-I TODO	PRI	23.1.2026
3	Diplomarbeitsbuch - Erster Layout-Entwurf fertig	E-I		

1) Vorbereitung (Agenda): V-S-Schilderung (mündlich), V-H-Handout, V-P-Präsentation, V-D-Dokument (digital), V-M-Multimedia

2) Ergebnis (Protokoll): E-B-Beschluss, E-E-Empfehlung, E-I-Information, E-O-Offener Punkt, E-A-Abgeschlossen (erledigt), TODO-TODO

Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



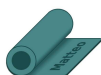
Projekthandbuch HTL3R

	- Team und Kurzfassung fertig - Regeln fürs Verfassen formuliert	E-I E-I		
4	Platten - 4 Dreiecksplatten aus 3 mm Alu ausgesägt - Werden bald mit Sensoren getestet - Finales Material noch nicht fix	E-I TODO E-O	BOE	23.1.2026
5	ToF - Dr. Philipp Kornfeind kommt besuchen - Prototyp sollte fertiggestellt sein	E-I E-I		

1) Vorbereitung (Agenda): V-S -Schilderung (mündlich), V-H -Handout, V-P -Präsentation, V-D -Dokument (digital), V-M -Multimedia

2) Ergebnis (Protokoll): E-B -Beschluss, E-E -Empfehlung, E-I -Information, E-O -Offener Punkt, E-A -Abgeschlossen (erledigt), TODO -TODO

Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Agenda/Protokoll

Projekttitel: Matteo
Auftraggeber*in: Universität Wien
Auftragnehmer*in: Laurenz Wimmer
Schuljahr: 2025/26 **Klasse:** 5AM

VERSION	DATUM	AUTORIN/AUTOR	ÄNDERUNG
v1.0	10.02.2026	Lisa Pripfl	Erstellung

Thema/Ziel(e) des Meetings:	Diplomarbeitsbuch, Jugend Innovativ				
Datum:	27.01.2026	Uhrzeit (von-bis):	15:20-15:50	Ort:	EE-Labor (063)
Teilnehmer*innen (Kürzel, ohne Titel)	Position / Funktion / Aufgabenbereich				
Laurenz Wimmer (WIM)	Projektleiter				
Nandor Bagari (BAG)	Stv. Projektleiter				
Mika Böhm (BOE)	Projektmitarbeiterin				
Lisa Pripfl (PRI)	Projektmitarbeiterin				
CEK	Hauptbetreuer / Individueller Betreuer				
DRE	Individueller Betreuer				
MSK	Individueller Betreuer				

Verteiler

Teilnehmer

Tagesordnung				
#	Tagesordnungspunkt Beschlüsse TODOs	Vorb. ^{1/} Erg. ²	Verantw.	Deadline
1	Diplomarbeitsbuch - Erste Kapitel fertig - Überall deutsche Begriffe verwenden - Oszilloskop Bilder machen	E-I E-I TODO	BOE	20.3.2026
2	Jugend Innovativ - Anwesenheit nur am ersten Tag - Bericht fertigstellen	E-B TODO	WIM	30.1.2026
3	Platine - Fehler bereits ausgebessert - Neue Platine bestellen	E-I TODO	BAG	28.1.2026

1) Vorbereitung (Agenda): V-S-Schilderung (mündlich), V-H-Handout, V-P-Präsentation, V-D-Dokument (digital), V-M-Multimedia

2) Ergebnis (Protokoll): E-B-Beschluss, E-E-Empfehlung, E-I-Information, E-O-Offener Punkt, E-A-Abgeschlossen (erledigt), TODO-TODO

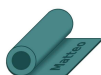
Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Projekthandbuch HTL3R

4	Programmierung - Erste Tests erfolgreich - Gesamttest durchführen	E-I TODO	alle	17.3.2026
5	TOFT - Verpolungsfehler	E-I		

1) Vorbereitung (Agenda): V-S -Schilderung (mündlich), V-H -Handout, V-P -Präsentation, V-D -Dokument (digital), V-M -Multimedia
2) Ergebnis (Protokoll): E-B -Beschluss, E-E -Empfehlung, E-I -Information, E-O -Offener Punkt, E-A -Abgeschlossen (erledigt), TODO -TODO
Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Agenda/Protokoll

Projekttitel: **Matteo**
 Auftraggeber*in: **Universität Wien**
 Auftragnehmer*in: **Laurenz Wimmer**
 Schuljahr: **2025/26** Klasse: **5AM**

VERSION	DATUM	AUTORIN/AUTOR	ÄNDERUNG
v1.0	07.10.2025	Laurenz Wimmer	Erstellung

Thema/Ziel(e) des Meetings:					
Datum:	10.02.2026	Uhrzeit (von-bis):	14:50-15:30	Ort:	065 L.ME
Teilnehmer*innen (Kürzel, ohne Titel)		Position / Funktion / Aufgabenbereich			
Laurenz Wimmer (WIM)		Projektleiter			
Nandor Bagari (BAG)		Sty. Projektleiter			
Mika Böhm (BOE)		Projektmitarbeiterin			
Lisa Pripfl (PRI)		Projektmitarbeiterin			
CEK		Hauptbetreuer / Individueller Betreuer			
MSK		Individueller Betreuer			

Verteiler

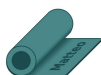
Teilnehmer, DRE

Tagesordnung				
#	Tagesordnungspunkt Beschlüsse TODOs	Vorb. ^{1/} Erg. ²	Verantw.	Deadline
1	Spracherkennung - Weiter trainieren - Befehle funktionieren gut - Hintergrundgeräusche werden noch erkannt	TODO E-I E-I	PRI	13.3.
2	Platine - Demonstration - ESP32 Boot-Pin-Belegung besprochen	E-A E-A		
3	Falsche Messung - Wegen offenen Sensorpads - Terminal-Programm, um Rohdaten anzuzeigen - Vielleicht zu schnelle Messung	E-I E-O E-I		
4	Video von Platine für Abschlusspräsentation erstellen	E-E	ALLE	13.3.

1) Vorbereitung (Agenda): V-S-Schilderung (mündlich), V-H-Handout, V-P-Präsentation, V-D-Dokument (digital), V-M-Multimedia

2) Ergebnis (Protokoll): E-B-Beschluss, E-E-Empfehlung, E-I-Information, E-O-Offener Punkt, E-A-Abgeschlossen (erledigt), TODO-TODO

Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



Agenda/Protokoll

Projekttitel: Matteo
Auftraggeber*in: Universität Wien
Auftragnehmer*in: Laurenz Wimmer
Schuljahr: 2025/26 **Klasse:** 5AM

VERSION	DATUM	AUTORIN/AUTOR	ÄNDERUNG
v1.0	07.10.2025	Laurenz Wimmer	Erstellung

Thema/Ziel(e) des Meetings:	Messung und Messfehler, Aktueller Stand		
Datum:	24.02.2026	Uhrzeit (von-bis):	Ort:
Teilnehmer*innen (Kürzel, ohne Titel)	Position / Funktion / Aufgabenbereich		
Laurenz Wimmer (WIM)	Projektleiter		
Nandor Bagari (BAG)	Stv. Projektleiter		
Mika Böhm (BOE)	Projektmitarbeiterin		
Lisa Pripfl (PRI)	Projektmitarbeiterin		
CEK	Hauptbetreuer / Individueller Betreuer		
DRE	Individueller Betreuer		
MSK	Individueller Betreuer		

Verteiler

Teilnehmer

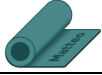
Tagesordnung

#	Tagesordnungspunkt Beschlüsse TODOs	Vorb. ^{1/} Erg. ²	Verantw.	Deadline
1	Messung - Schwankungen bei bestimmten Sensoren - Störungen durch Antennen, GND, gegenseitige Beeinflussung der Leitungen - Geerdete Schirmung, verdrehte einzelne Leitungen bei GND	E-I E-I E-E		
2	Aktueller Stand - Gehäuse wird gedruckt - Matte ist vorbereitet - Kabel auf richtige Länge kürzen - Sensortests werden durchgeführt - Sprachsteuerung fertig, eventuell neue Aufnahmen für Sprachausgabe	E-I E-I TODO TODO	BAG BOE	3.3.2026 3.3.2026

1) Vorbereitung (Agenda): V-S-Schilderung (mündlich), V-H-Handout, V-P-Präsentation, V-D-Dokument (digital), V-M-Multimedia

2) Ergebnis (Protokoll): E-B-Beschluss, E-E-Empfehlung, E-I-Information, E-O-Offener Punkt, E-A-Abgeschlossen (erledigt), TODO-TODO

Abnahme des Protokolls erfolgt automatisch, wenn binnen 48 Stunden kein Einspruch erfolgt.



9.3 Begleitprotokolle

Diplom- und Abschlussarbeiten

Höhere Technische Bundeslehranstalt Wien 3, Rennweg 89b, A - 1030 Wien



Begleitprotokoll						
Schuljahr:		2025/26		Abteilung:		Mechatronik
Name:	Laurenz Wimmer	Rolle:	Projektleitung	Typ:	Diplomarbeit	
Titel der Arbeit:		Indiv. Betreuung:				
Individuelle Themenstellung:		Matteo				
Elektronik u. Projektmanagement						
Datum	TeilnehmerInnen (Kürzel, bei Besprechungen)	Art	Beschreibung der Tätigkeiten, der Arbeiten, der Vereinbarungen, TO Dos	Deadline	Dauer	
08.9.2025	DRE, MSK, CEK	Meeting	Besprechung des aktuellen Stands		01:00	
09.9.2025	S. Andreas Petter, Philipp Kornfeind	Meeting	Klären einiger technischer Fragen und Abklären des Konzepts		00:30	
07.10.2025	DRE, MSK, CEK	Jour Fixe	DA-Buch, Stromversorgung, Sprachsteuerung, Abnahmeprotokoll und Jour-Fixe-Termine besprochen		01:15	
21.10.2025	DRE, MSK, CEK	Jour Fixe	Aktueller Stand, Review Platine, EUSART, Stromversorgung		00:30	
04.11.2025	CEK, DRE, MSK	Jour Fixe	Aktueller Stand und TOFT		00:30	
02.12.2025	DRE, MSK, CEK	Jour Fixe	Sprachsteuerung, Übertragungsprotokoll, Sensor		00:35	
16.12.2025	DRE, MSK, CEK	Jour Fixe	LLC, Step-Down-Conv., LED-Streifen, µC-Kommunikation,		00:40	
13.1.2026	DRE, MSK, CEK	Jour Fixe	Platine, Sprachsteuerung, Kommunikation Microcontroller, ToFT		00:40	
27.01.2026	DRE, MSK, CEK	Jour Fixe	Diplomarbeitsbuch, Jugend Innovativ		00:15	
10.2.2026	DRE, MSK, CEK	Jour Fixe	Spracherkennung, Platine, Flasche Messung, Video von Platine		00:40	
24.2.2026	DRE, MSK, CEK	Jour Fixe	Messfehler		00:30	
Gesamtstunden						07:05

Abschluss des Protokolls	
Datum:	19.03.2026

Unterschrift Schülerin/Schüler:

Diplom- und Abschlussarbeiten

Höhere Technische Bundeslehranstalt Wien 3, Rennweg 89b, A - 1030 Wien

Begleitprotokoll			
Schuljahr:		2025/26	
Name:	Nandor Bagari	Rolle:	Projektleitung Stv.
Titel der Arbeit:		Matteo	
Individuelle Themenstellung:		Elektronik u. Konstruktion	
		Abteilung:	Mechatronik
		Typ:	Diplomarbeit
		Indiv. Betreuung:	Ewald Cekan

Datum	TeilnehmerInnen (Kürzel, bei Besprechungen)	Art	Beschreibung der Tätigkeiten, der Arbeiten, der Vereinbarungen, TO Dos	Deadline	Dauer
08.9.2025	DRE, MSK, CEK	Meeting	Besprechung des aktuellen Stands		01:00
07.10.2025	DRE, MSK, CEK	Jour Fixe	DA-Buch, Stromversorgung, Sprachsteuerung, Abnahmeprotokoll und Jour-Fixe-Termine besprochen		01:15
21.10.2025	DRE, MSK, CEK	Jour Fixe	Aktueller Stand, Review Platine, EUSART, Stromversorgung		00:30
04.11.2025	CEK, DRE, MSK	Jour Fixe	Aktueller Stand und TOFT		00:30
18.11.2025	CEK, DRE, MSK, PMA	Jour Fixe	Projektfortschritt, Präsentation, Material		00:30
02.12.2025	DRE, MSK, CEK	Jour Fixe	Sprachsteuerung, Übertragungsprotokoll, Sensor		00:35
16.12.2025	DRE, MSK, CEK	Jour Fixe	LLC, Step-Down-Conv., LED-Streifen, µC-Kommunikation,		00:40
13.1.2026	DRE, MSK, CEK	Jour Fixe	Platine, Sprachsteuerung, Kommunikation Microcontroller, ToFT		00:40
27.01.2026	DRE, MSK, CEK	Jour Fixe	Diplomarbeitsbuch, Jugend Innovativ		00:15
10.2.2026	DRE, MSK, CEK	Jour Fixe	Spracherkennung, Platine, Flasche Messung, Video von Platine		00:40
24.2.2026	DRE, MSK, CEK	Jour Fixe	Messfehler		00:30
			Gesamtstunden		07:05

Abschluss des Protokolls	
Datum:	19.03.2026

Unterschrift Schülerin/Schüler:

Diplom- und Abschlussarbeiten

Höhere Technische Bundeslehranstalt Wien 3, Rennweg 89b, A - 1030 Wien



Begleitprotokoll					
Schuljahr:		2025/26		Abteilung:	
Name:	Mika Böhm	Rolle:	Projektteammitglied	Typ:	Diplomarbeit
Titel der Arbeit:		Indiv. Betreuung:			
Individuelle Themenstellung:		Matteo			
		Microcontroller-Programmierung u. Website			
		Mechatronik			
Datum	TeilnehmerInnen (Kürzel, bei Besprechungen)	Art	Beschreibung der Tätigkeiten, der Arbeiten, der Vereinbarungen, TO Dos	Deadline	Dauer
08.9.2025	DRE, MSK, CEK	Meeting	Besprechung des aktuellen Stands		01:00
07.10.2025	DRE, MSK, CEK	Jour Fixe	DA-Buch, Stromversorgung, Sprachsteuerung, Abnahmeprotokoll und Jour-Fixe-Termine besprochen		01:15
21.10.2025	DRE, MSK, CEK	Jour Fixe	Aktueller Stand, Review Platine, EUSART, Stromversorgung		00:30
04.11.2025	CEK, DRE, MSK	Jour Fixe	Aktueller Stand und TOFT		00:30
18.11.2025	CEK, DRE, MSK, PMA	Jour Fixe	Projektfortschritt, Präsentation, Material		00:30
02.12.2025	DRE, MSK, CEK	Jour Fixe	Sprachsteuerung, Übertragungsprotokoll, Sensor		00:35
16.12.2025	DRE, MSK, CEK	Jour Fixe	LLC, Step-Down-Conv., LED-Streifen, µC-Kommunikation,		00:40
13.1.2026	DRE, MSK, CEK	Jour Fixe	Platine, Sprachsteuerung, Kommunikation Microcontroller, ToFT		00:40
10.2.2026	DRE, MSK, CEK	Jour Fixe	Spracherkennung, Platine, Flasche Messung, Video von Platine		00:40
24.2.2026	DRE, MSK, CEK	Jour Fixe	Messfehler		00:30
Gesamtstunden					06:50

Abschluss des Protokolls	
Datum:	19.03.2026

Unterschrift Schülerin/Schüler:

Diplom- und Abschlussarbeiten

Höhere Technische Bundeslehranstalt Wien 3, Rennweg 89b, A - 1030 Wien

Begleitprotokoll			
Schuljahr:		2025/26	
Name:	Lisa Pripfl	Rolle:	Projektteammitglied
Titel der Arbeit:		Indiv. Betreuung:	
		Matteo	
Individuelle Themenstellung:		Abteilung:	
		Mechatronik	
		Typ:	
		Diplomarbeit	
		Indiv. Betreuung:	
		Richard Drechsler	
Programmierung von Sprachein- und -ausgabe			

Datum	TeilnehmerInnen (Kürzel, bei Besprechungen)	Art	Beschreibung der Tätigkeiten, der Arbeiten, der Vereinbarungen, TO Dos	Deadline	Dauer
08.9.2025	DRE, MSK, CEK	Meeting	Besprechung des aktuellen Stands		01:00
09.9.2025	S. Andreas Petter, Philipp Kornfeind	Meeting	Klären einiger technischer Fragen und Abklären des Konzepts		00:30
07.10.2025	DRE, MSK, CEK	Jour Fixe	DA-Buch, Stromversorgung, Sprachsteuerung, Abnahmeprotokoll und Jour-Fixe-Termine besprochen		01:15
21.10.2025	DRE, MSK, CEK	Jour Fixe	Aktueller Stand, Review Platine, EUSART, Stromversorgung		00:30
04.11.2025	CEK, DRE, MSK	Jour Fixe	Aktueller Stand und TOFT		00:30
02.12.2025	DRE, MSK, CEK	Jour Fixe	Sprachsteuerung, Übertragungsprotokoll, Sensor		00:35
18.11.2025	DRE, MSK, CEK, PMA	Jour Fixe	Projektfortschritt, Präsentation, Material		00:30
16.12.2025	DRE, MSK, CEK	Jour Fixe	LLC, Step-Down-Conv., LED-Streifen, µC-Kommunikation,		00:40
13.1.2026	DRE, MSK, CEK	Jour Fixe	Platine, Sprachsteuerung, Kommunikation Microcontroller, TorT		00:40
27.01.2026	DRE, MSK, CEK	Jour Fixe	Diplomarbeitsbuch, Jugend Innovativ		00:15
10.2.2026	DRE, MSK, CEK	Jour Fixe	Spracherkennung, Platine, Flasche Messung, Video von Platine		00:40
24.2.2026	DRE, MSK, CEK	Jour Fixe	Messfehler		00:30
Gesamtstunden					07:35

Abschluss des Protokolls	
Datum:	19.03.2026

Unterschrift Schülerin/Schüler:



9.4 Jugend Innovativ Bericht

Diplomarbeit Matteo

2025/2026

HTL Rennweg, 5AM

P2514360-Matteo-Die Matte für Balanceübungen

Engineering I – Mechatronik



Schule

HTBLA Wien 3

Rennweg 89B, 1030 Wien

direktion@htl.rennweg.at

+43 1 24215-10

Projektkoordinator

Laurenz Wimmer

laurenz.wimmer@htl.rennweg.at

+43 677 64857313

Projektbetreuer

Ewald Cekan

Ewald.cekan@htl.rennweg.at

Projektteam

Laurenz Wimmer, 5AM, Projektleiter

Nandor Bagari, 5AM, stv. Projektleiter

Mika Böhm, 5AM, Projektmitarbeiterin

Lisa Pripfl, 5AM, Projektmitarbeiterin

Externe Kooperationspartner

Universität Wien, Universitätssportinstitut



1 Inhaltsverzeichnis

1	Inhaltsverzeichnis.....	2
2	Projektdokumentation.....	3
2.1	Projektentstehung und Planung	3
2.1.1	Erste Ideen.....	3
2.1.2	Ausgearbeitetes Konzept	3
2.1.3	Ziel des Projektes	4
2.1.4	Zeitliche Planung.....	4
2.2	Projektumsetzung und Ausblick.....	6
2.2.1	Allgemeines	6
2.2.2	Was Matteo auszeichnet.....	7
2.2.3	Innovationen.....	7
2.2.4	Umsetzung der Ziele	8
2.2.5	Nutzungsrecht.....	15
2.3	Aktueller Stand und Ergebnisse	16
2.3.1	Elektronik	16
2.3.2	Konstruktion.....	17
2.3.3	Programmierung.....	17
2.3.4	Spracherkennung.....	17
2.3.5	Ausblick	18
2.4	Verwertung nach Abschluss des Projektes	18
2.5	Verbesserungsmöglichkeiten.....	19
2.5.1	Sensorik	19
2.5.2	Erweiterte Übungsmöglichkeiten	19
2.5.3	Erweitertes Feedback.....	19
2.5.4	Übertragen der Daten auf Website	19
2.6	Projektteam und -organisation	20
2.6.1	Motivation.....	20
2.6.2	Ressourcen	22
3	Quellenverzeichnis.....	23
4	Abbildungsverzeichnis.....	24

2 Projektdokumentation

2.1 Projektentstehung und Planung

Die Produktidee „Matteo – Die Matte für Balanceübungen“ ist im Rahmen des Projektes „Sinnvoll aktiv“ in der Untergruppe „Active Aging“ entstanden, das jährlich vom Universitätssportinstitut Wien durchgeführt wird. Im Rahmen von „Sinnvoll aktiv“ besteht die Möglichkeit, sich mit Interessensgruppen auszutauschen und Ideen zu finden. So ist die grundsätzliche Idee einer Matte entstanden, die es blinden und sehbeeinträchtigten Menschen höheren Alters ermöglichen soll, eigenständig Balanceübungen durchzuführen. Über einen unserer Projektbetreuer, Martin Meschik, der schon länger in Kontakt mit der Universität Wien steht, wurde die Idee an uns herangetragen.

Da unser Team von Anfang an ein Projekt zur Unterstützung von blinden und sehbeeinträchtigten Personen als Diplomarbeit durchführen wollte, waren wir schnell von der Idee überzeugt und konnten im März letzten Jahres an einem Workshop teilnehmen, bei dem auch Vertreter diverser Blindenvereine und des ASVÖ-Versehrtenportklubs anwesend waren. Dort konnten wir mit Betroffenen über unsere Idee sprechen und erste Vorschläge einholen, wie das Projekt am besten umzusetzen ist. Außerdem hatten wir die Möglichkeit, durch unterschiedliche Übungen selbst zu erleben, welche Probleme durch Sehschwächen im Alltag entstehen können und unsere Erfahrungen in unseren Überlegungen zur Umsetzung miteinbeziehen.

2.1.1 Erste Ideen

Im Laufe der nächsten Monate bis Ende Mai wurden dann konkrete Lösungsansätze entwickelt, die auch in Meetings mit Verantwortlichen des Universitätssportinstituts besprochen und gegebenenfalls abgeändert wurden. Ende Juni wurde dann ein weiteres Treffen aller „Sinnvoll Aktiv“-Teilnehmer organisiert, bei dem das erarbeitete Konzept mit Vertretern vorher genannter Vereine besprochen werden konnte und weitere Anregungen bezüglich der Umsetzung und insbesondere der Barrierefreiheit eingeholt werden konnten. Anhand dieses Feedbacks wurde über den Sommer das Konzept konkretisiert und die endgültigen Ziele gesetzt.

2.1.2 Ausgearbeitetes Konzept

Das finale Konzept umfasst eine Matte mit integrierten Drucksensoren, die die Belastung der einzelnen Füße miteinander vergleicht und bezogen auf das Körpergewicht als Verhältnis ausgibt. Um eine möglichst hohe Barrierefreiheit zu erzielen, können sämtliche Einstellungen und Aktionen über eine Sprachsteuerung vorgenommen werden. Feedback erfolgt ebenfalls über eine Sprachausgabe, es gibt aber zusätzlich noch visuelles Feedback für nicht vollständig erblindete Personen. Optional könnte außerdem haptisches Feedback hinzugefügt werden, bei dem die Matte durch Vibrationen beispielsweise das Ende der Übung signalisiert.

Die Nutzung der Matte erfolgt dann nach folgendem Schema: Die Matte wird über einen Druckschalter aktiviert, der mit einem Fuß betätigt werden kann. Sobald die Sprachsteuerung aktiviert ist, kann die gewünschte Übung eingestellt werden. Dazu wird zuerst der Fuß und dann die gewünschte Belastung in 10%-Schritten

genannt, also zum Beispiel „rechter Fuß 70%“. Alternativ zu einer Messung mit Belastungsziel gibt es auch die Möglichkeit, eine Messung ohne Ziel mit dem Befehl „Kein Ziel“ auszuwählen. Die Sprachausgabe wiederholt dann die erkannte Einstellung. Wenn der Befehl richtig erkannt wurde, wird mit dem Befehl „Start“ die Übung gestartet. Während der Übung wird im Abstand von 10 Sekunden die aktuelle Belastung in 5%-Schritten ausgegeben. Sobald für die Dauer von 3 Sekunden die eingestellte Belastung erreicht ist, gibt die Matte das Ende der Übung über die verschiedenen Feedbackmöglichkeiten aus. Die Übung kann auch jederzeit mit dem Befehl „Stopp“ abgebrochen werden. Außerdem kann die Lautstärke mit den Befehlen „Lauter“ und „Leiser“ geändert werden.

2.1.3 Ziel des Projektes

Durch diese Balanceübungen soll erreicht werden, dass sich die Balance der Benutzer:innen so weit verbessert, dass sie eventuell auch andere körperliche Aktivitäten ausüben können. Dadurch soll die Motivation zur körperlichen Betätigung erhöht und die Gesundheit gefördert werden. Matteo kann aber auch in anderen Szenarien wie einem Reha-Aufenthalt eingesetzt werden, um die Balance zu trainieren oder Physiotherapeut:innen direktes Feedback zu bieten, wie die Patient:innen ihr Gewicht verteilen. Auch nach Unfällen, bei denen ein Bein verletzt wurde, könnte Matteo zum Einsatz kommen.

2.1.4 Zeitliche Planung

Die zeitliche Einteilung des Projektes wurde durch zwei Faktoren beeinflusst, einerseits die Vorgaben durch die Timeline von „Sinnvoll aktiv“ und andererseits durch die Fristen, die die Schule für die Durchführung der Diplomarbeit gesetzt hat. Um zeitliche Probleme gegen Ende des Projektes zu verhindern, wurden die Meilensteine sehr ambitioniert gesetzt und genügend Pufferzeit eingeplant, um Spielraum bei der Durchführung zu haben. Die Meilensteine sind in folgender Tabelle zu sehen:

#	Meilenstein	SOLL-Termin
1	Website Launchen	12.Sep.25
2	Planung abgeschlossen	16.Sep.25
3	Abgabe Antrag	19.Sep.25
4	Techn. Planung abgeschlossen	17.Okt.25
5	Sprachausgabe	06.Nov.25
6	Erster Prototyp	07.Nov.25
7	Zwischenpräsentation	25.Nov.25
8	Umsetzung abgeschlossen	23.Dez.25
9	Testen abgeschlossen	28. Jan 2026
10	Projekt abgenommen	17.Mär.26



Der 17. März 2026 ist der Tag, an dem laut Vorgaben der Schule die Endpräsentation und Abnahme der Diplomprojekte durchgeführt werden. Februar und März wurden in der Planung dafür vorgesehen, eventuelle Lieferverzögerungen und Fehler bei der Umsetzung auszugleichen. Die Universität Wien hat im Rahmen der Planung ebenfalls diese Meilensteine erhalten und abgesegnet. Für den Verlauf des Projektes wurde vereinbart, dass die Verantwortlichen von „Sinnvoll Aktiv“ laufende Updates zum Projektstand in Form von E-Mails und je nach Bedarf auch Online-Meetings erhalten. Außerdem werden die Professoren, die das Projekt in der Schule betreuen, in 14-tägigen Jour-Fixe-Meetings über den Projektfortschritt informiert.

2.2 Projektumsetzung und Ausblick

2.2.1 Allgemeines

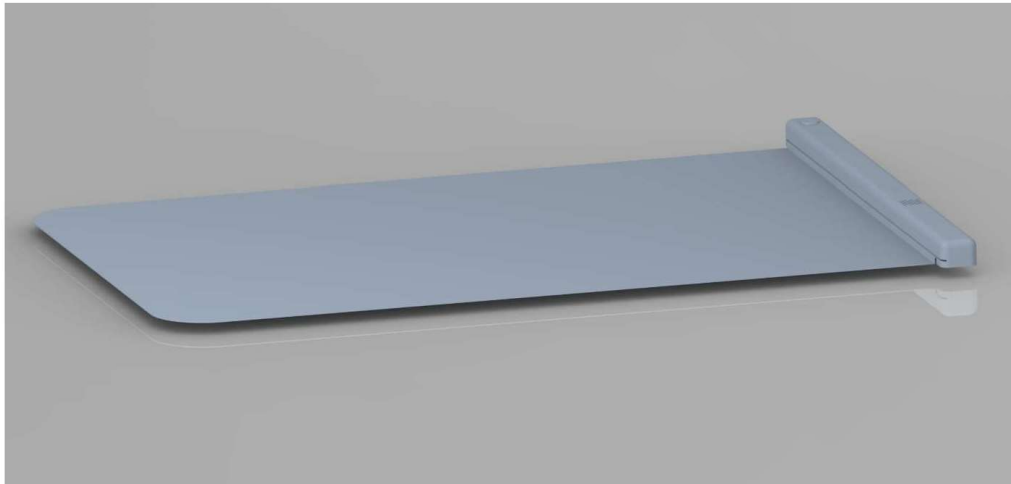


Abbildung 1 3D-Modell Matteo

Abbildung 1 zeigt ein 3D-Modell der Matte nach aktuellem Stand. Die Matte hat ungefähre Dimensionen von 70x100x1,5 cm und hat auf einer der kürzeren Enden das Gehäuse für die Elektronik befestigt. Im Gehäuse befinden sich die drei Prozessoren, die für Sprachsteuerung, Messung und Datenverarbeitung verwendet werden, die Stromversorgung über Lithium-Ionen-Akkus sowie die Messschaltung, mit der die Drucksensoren ausgemessen werden. Die 12 Sensoren sind in der Matte nach dem Schema eingebaut, das in Abbildung 2 gezeigt ist. Jedes X markiert dabei einen Sensor und die Formen, die die Sensoren umschließen, kennzeichnen Platten zur besseren Aufteilung des Drucks.

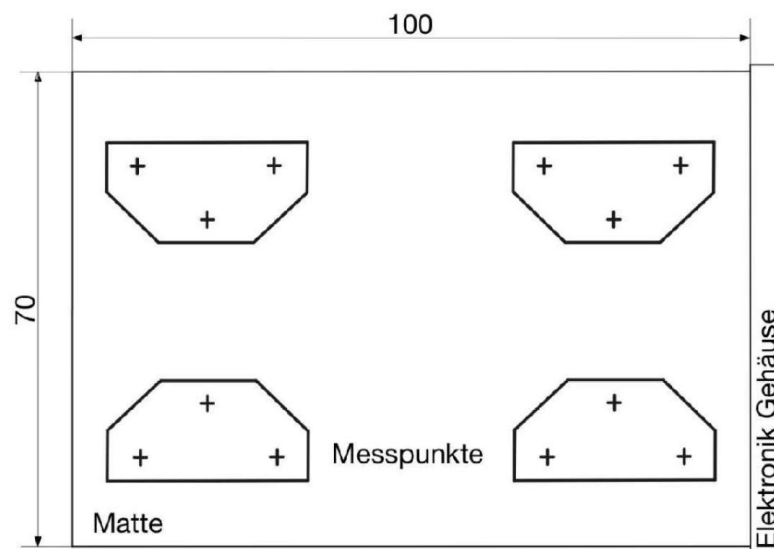


Abbildung 2 Sensorlayout

2.2.2 Was Matteo auszeichnet

Was „Matteo – Die Matte für Balanceübungen“ von anderen Produkten abhebt, ist die Kombination vieler verschiedener Konzepte, die zu einer gänzlich neuen Anwendungsmöglichkeit geführt haben. Was besonders heraussticht ist die Verwendung einer Sprachsteuerung zur Bedienung der Matte, die im weiteren Sinne als Hilfsmittel im Bereich der Gesundheitsvorsorge gesehen werden kann. Außerdem ist Matteo das erste Produkt, dass die Messung der Gewichtsverteilung kompakt in einer Gymnastikmatte unterbringt.

2.2.3 Innovationen

Im Gegensatz zu vergleichbaren Produkten wie dem Wii Fit Balance Board bietet Matteo viele Vorteile. Aufgrund der gewählten Materialien und der Dimensionen ist die Verletzungsgefahr wesentlich geringer, da keine Stolpergefahr durch eine mehrere Zentimeter hohe Stufe entsteht und die Gymnastikmatte selbst aus Schaumstoff und nicht aus Hartplastik besteht. Weiters ist Matteo wesentlich einfacher in der Benutzung, da keine externen Geräte zur Bedienung benötigt werden. Dadurch kann sichergestellt werden, dass fehlende technische Kenntnisse kein Hindernis darstellen und auch ältere, technisch weniger versierte Personen problemlos Matteo benutzen können. Die Aufbewahrung der Matte gestaltet sich auch wesentlich einfacher, da sie zu einem gewissen Grad eingerollt werden kann und somit keine Platzprobleme durch starre Dimensionen bei der Verstauung entstehen, wie es bei anderen Produkten der Fall ist.

Um möglichst genaue Messungen zu erhalten, wurde eigens für dieses Projekt eine Messschaltung entwickelt, mit der die verbauten Drucksensoren ausgelesen und in weiterer Folge ausgewertet wurden. Durch das Multiprozessorsystem gibt es außerdem in Zukunft die Möglichkeit für diverse funktionale Erweiterungen, mit denen die Nutzung von Matteo auf viele andere Bereiche ausgeweitet werden könnte.

2.2.3.1 Das Sprachmodell

Bei der Entwicklung der Sprachsteuerung wurde künstliche Intelligenz eingesetzt, mit der die einzelnen Befehle erstellt wurden. Das verwendete Programm „Edge Impulse“, mit dem ein Sprachmodell erstellt, trainiert und verbessert werden kann, verwendet Machine Learning. Dabei handelt es sich um einen maschinellen Lernprozess, bei dem bereits vorhandene Daten analysiert, geordnet und letztendlich Aktionen zugeordnet werden können. Im Laufe der letzten Monate wurden deshalb viele Stunden lang Sprachaufnahmen von über 30 Personen erstellt, die sich bereiterklärt haben, bei dem Projekt mitzuwirken und in gewisser Weise eine Vorlage für die Sprachsteuerung zu erstellen. Da ein Machine-Learning-Modell darauf beruht, dass bereits bestehende Muster erkannt und verarbeitet werden, war es eine große Hilfe, viele unterschiedliche Stimmen, Tonlagen, Aussprachen und Betonungen als Basis für das Training der Sprachsteuerung zur Verfügung zu haben. Anhand dieser Tonaufnahmen konnten dann die bereits beschriebenen Befehle erstellt werden.



Abbildung 3 Logo Edge Impuls

Edge Impulse Logo:
https://www.edge-ai-vision.com/wp-content/uploads/2021/05/logo_edgeimpulse_may_2021.png

2.2.4 Umsetzung der Ziele

Um alle gesetzten Ziele mit besonderem Augenmerk auf die Barrierefreiheit und Sicherheit zu erfüllen, wurde eine Vielzahl von Maßnahmen im Rahmen der Planung besprochen und nach Rücksprache mit der Universität Wien und unter Berücksichtigung der zur Verfügung stehenden Zeit diejenigen ausgewählt, die am wirksamsten und im Rahmen der Diplomarbeit umsetzbar sind.

2.2.4.1 Sicherheitsaspekte

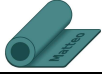
Begonnen wurde bei der Minimierung der Verletzungsgefahr. Dazu wurden anfangs Probleme gesammelt, die bei bereits bestehenden Produkten bestehen. Auch die Hinweise der Vertretungen der Blindenvereine sowie des Universitätsportinstituts wurden dabei berücksichtigt. Auf Basis des bereits vorhandenen Konzepts für Matteo wurden dann Änderungen durchgeführt. So wurde beispielsweise die Idee verworfen, die Matte mit einem Stand-PC via Kabelverbindung zu verbinden und so die Sprachsteuerung umzusetzen. Stattdessen wurde die Sprachsteuerung direkt in die Elektronik integriert, um keine Unsicherheiten durch herumliegende Leitungen zu verursachen. Aus demselben Grund wird die Matte in ihrer finalen Version von einem Akku versorgt und nicht wie ursprünglich geplant über ein Netzgerät, das direkt an einer Steckdose angesteckt wird. Weiters ist die Höhe der Matte von maximal 15 mm und die Höhe des seitlich befestigten Gehäuses von maximal 35 mm weitestgehend unproblematisch, da auch bei der Gestaltung des Gehäuses Formen verwendet werden, die ein Einhängen oder Hängenbleiben verhindern sollen.

2.2.4.2 Nutzung

Als nächstes wurden Überlegungen zum Thema Benutzerfreundlichkeit angestellt. Nach Rücksprache mit der Universität Wien bezüglich der Ausgangssituation wurde die Matte so ausgelegt, dass Seheinschränkungen, körperliche Einschränkungen und Probleme mit neuer Technologie keine Hindernisse darstellen sollten.

Zur Aktivierung der Matte wurden mehrere Optionen in Betracht gezogen. Zuerst war geplant, in dem Gehäuse der Matte einen Schalter zu verbauen, der mit der Hand betätigt wird. Das Feedback der Universität und der kooperierenden Vereine hat allerdings gezeigt, dass das Bücken zur Aktivierung ein Problem darstellen kann und dass sich durch die körperlichen Einschränkungen im Alter ein Einschalten per Hand als sehr schwierig herausstellen könnte. Deswegen wurde der Schalter so konzipiert, dass er mit dem Fuß betätigt werden kann. Dazu wird ein Drückschalter verwendet, der bei einmaligem Drücken aktiviert und bei nochmaliger Betätigung deaktiviert wird. Drückschalter sind allerdings meistens sehr klein und sind schwierig zu ertasten. Deshalb ist im Gehäuse eine große Taste vorgesehen, die leicht durch Fühlen zu finden ist und den Druck auf den eigentlichen Schalter überträgt. Diese Taste wird durch Federn gestützt, die die Taste nach der Betätigung wieder in die Ausgangsposition zurückbewegen. Optional können durch Reliefs auf der Oberseite der Taste zusätzliche Hilfen zum Ertasten hinzugefügt werden.

Für die Bedienung der Matte war vorerst geplant, einen analogen Regler in Kombination mit Tasten im Gehäuse einzubauen, über die alle Einstellungen vorgenommen werden können. Allerdings würde dadurch wieder das Problem auftreten, dass sich die Benutzer:innen zum Gehäuse bücken müssten. Aus diesem Grund war der nächste Lösungsansatz, eine Fernbedienung zu verwenden. Eine zugekaufte Fernbedienung



würde es allerdings schwierig machen, Feedback zu den ausgewählten Einstellungen zu geben und zu zeigen, welche Tasten welche Einstellungen verändern. Die Entwicklung einer eigenen Fernbedienung mit haptischem Feedback und eigens konzipierten Tasten würde aber den Rahmen der Diplomarbeit übersteigen. Aus diesem Grund wurde letztendlich die Sprachsteuerung als Bedienelement gewählt, da sie eine intuitive und einfache Nutzung ermöglicht.

2.2.4.3 Aufladen der Akkus

Eine Herausforderung bleibt allerdings das Aufladen der Akkus, die im Gehäuse der Matte verbaut sind. Damit alle herkömmlichen Smartphone-Ladekabel und -Netzgerät verwendet werden können, wurde ein USB-C-Port zum Aufladen der Akkus gewählt. Dieser ist allerdings sehr klein und vor allem unter Berücksichtigung von Blindheit und Sehschwächen schwierig zu finden und zu verwenden. Um dem entgegenzuwirken, werden die um den USB-C-Port umliegenden Seiten abgeschrägt und als eine Art Trichter verwendet, der beim Einstecken des Netzgerätes helfen soll, indem er den Stecker am Ladekabel direkt zur USB-C-Buchse führt. Was sich hier allerdings nicht verhindern lässt, ist, dass sich die Benutzer:innen zum Anstecken des Ladekabels zum Gehäuse bücken müssen. Die Akkulaufzeit ist aber so ausgelegt, dass die Matte für mehrere Stunden aktiviert sein kann, ohne dass ein Ladevorgang durchgeführt werden muss. Bei Personen mit Bewegungseinschränkungen ist demnach die Idee, dass die Akkulaufzeit der Matte ausreicht, um zwischen Besuchen von Verwandten, Bekannten oder unterstützendem Personal Übungen durchführen zu können und das Aufladen von anderen Personen erledigen zu lassen.

2.2.4.4 Konstruktion

Die Aufgaben im Bereich der Konstruktion teilen sich in 2 Bereiche auf. Einerseits muss das Gehäuse für die Elektronik entworfen werden und andererseits die Platten, die das Gewicht von den Füßen möglichst gleichmäßig auf die 12 verbauten Sensoren aufteilen sollen. Das Gehäuse ist dabei die kleinere Herausforderung, da das Material nicht beansprucht wird und lediglich die Dimensionen an die Größe der Elektronik angepasst werden muss. Beim Design der Platten hingegen haben sich einige Komplikationen ergeben.

Die Funktionsweise der Sensoren, die weiter unten erklärt wird, erfordert es, dass das Gewicht sehr gleichmäßig auf die gesamte Messfläche der Sensoren aufgeteilt wird. Aus diesem Grund ist es wichtig, dass sich die verwendeten Platten nicht durch die Belastung durchbiegen, da sonst ein leichtes Kippen auftreten würde und die Sensoren falsch messen oder gar Schaden nehmen könnten. Aus diesem Grund wurden verschiedene Materialien in Betracht gezogen.

Von Seiten der Universität wurden Karbonplatten vorgeschlagen, die sehr dünn gehalten werden könnten und sich dadurch gut eignen würden. Allerdings ist Karbon schwierig zu bearbeiten und sehr scharfkantig, weswegen als Alternative ein Verbundmaterial aus Modellbaufurnierholz, Epoxidharz und Glasfaser vorgeschlagen wurde. Diese Platten müssten allerdings eigens angefertigt werden und wären vermutlich nicht biegesteif genug gewesen, um sie in der Matte verbauen zu können. Schlussendlich wurden Aluminiumplatten gewählt, deren geeignete Dicke experimentell durch Belastungstests ermittelt wird.

Für die Form der Platten waren ebenfalls mehrere Möglichkeiten im Gespräch. Von Seiten der Universität wären quadratische Platten erwünscht gewesen. Aufgrund der 3-Punkt-Auflage (siehe Abbildung 4 Platten zur Druckverteilung, die Kreise stellen die Sensoren dar) war aber die Befürchtung, dass die Platten kippen könnten. Um diesem Kippen entgegenzuwirken, wurden die ersten Tests mit dreieckigen Platten durchgeführt, die zwar wesentlich stabiler sind, aber auch eine signifikant kleinere Fläche haben, auf der sie die Belastung aufnehmen können. Aus diesem Grund wurde nach Rücksprache mit der Universität die Form gewählt, die auf Abbildung 4 zu sehen ist und als einer Kombination von Dreieck und Quadrat entstanden ist.

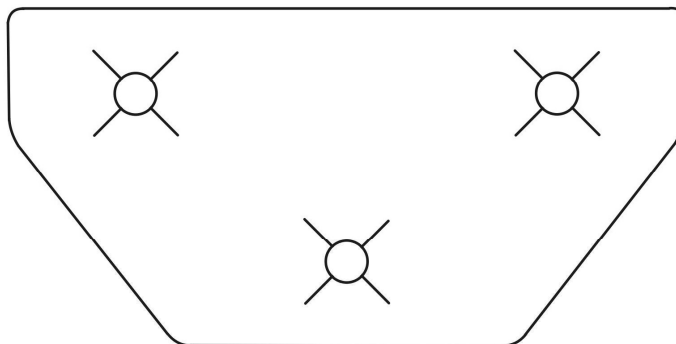


Abbildung 4 Platten zur Druckverteilung

2.2.4.5 Hardware und Funktionsweise

Für die gesamte Elektronik wurde eine Platine erstellt. Sie hat ungefähre Maße von 600x30x2 mm, damit das Gehäuse möglichst schmal und niedrig gehalten werden kann. Sie dient in erster Linie dazu, die drei verwendeten Mikroprozessoren für die Sprachsteuerung, die Messung und die Datenverarbeitung zu verbinden. Abbildung 5 gibt eine grobe Übersicht über das Zusammenwirken der Elemente, die anschließend beschrieben werden.

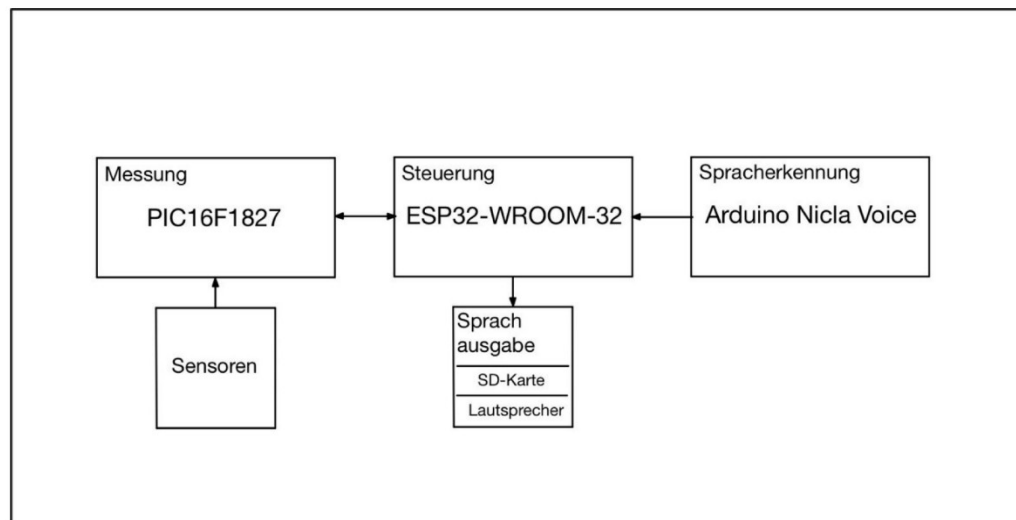


Abbildung 5 Übersicht Microprozessoren

2.2.4.5.1 Spracherkennung

Für die Ausführung der Sprachsteuerung wird ein Arduino-Nicla-Voice-Microcontroller verwendet, der eigens für Spracherkennung entwickelt und ausgelegt wurde. Dieser verwendet einen Prozessor namens NDP120 (Neural Decision Processor), der auf Machine Learning spezialisiert ist. Durch seine Leistungsstärke können viele Befehle problemlos gleichzeitig trainiert und eingespeichert werden. Außerdem ist auf dem Microcontroller ein integriertes Mikrofon verbaut, sodass keine zusätzlichen Bauteile benötigt werden. Dieser Microcontroller gibt allerdings nur Rückmeldung, welcher Befehl erkannt wurde.



Abbildung 6 Arduino Nicla Voice

Bild von: https://cdn-reichelt.de/resize/600%2F-/web/xxl_ws/R600%2F3840-2160-MAX.png?type=ProductXxl&resize=600%252F-

2.2.4.5.2 Ablaufsteuerung

Die Verarbeitung der Befehle übernimmt dann ein ESP32-WROOM-Modul, das via I2C-Schnittstelle mit dem Arduino Nicla Voice verbunden ist. Über diese serielle Schnittstelle werden die erkannten Befehle in Form von so genannten „Labels“ an den ESP32 weitergegeben, der dann entsprechende Aktionen setzt. Zuerst wird überprüft, ob der Befehl gerade ausgeführt werden kann, da nicht alle Befehle während einer aktiven Übung eine Auswirkung haben. Sollte der Befehl ausgeführt werden können, wird der Lautsprecher über die I2S-Schnittstelle des ESP32 aktiviert und die Audiodatei abgespielt, die dem erkannten Befehl entspricht. I2S ist eine serielle Schnittstelle, die speziell für Kommunikation mit Audiogeräten gedacht ist.



Abbildung 7 ESP32 WROOM

Bild von <https://www.az-delivery.de/cdn/shop/files/esp32-nodemcu-module-wlan-wifi-dev-kit-c-development-board-mit-cp2102-und-usb-c-anschluss-esp-32-esp32-wroom-32-kompatibel-mit-arduino-830819.jpg?v=1732745705&width=1200>

Die Tonaufnahmen wurden mithilfe der Website „ElevenLabs“ erstellt, die eine KI-Stimme verwendet, um einen eingegebenen Text in eine gesprochene Audiodatei umzuwandeln. Diese wurden dann auf einer Micro-SD-Karte gespeichert. In Zukunft könnten noch zusätzliche Hinweise ausgegeben werden, da der Speicherplatz der SD-Karte zum jetzigen Zeitpunkt nur zu kleinen Teilen belegt ist. Parallel zum Abspielen der Audiodateien sendet beziehungsweise empfängt der ESP32 Signale zum / vom 3. Mikroprozessor, einem PIC16F1827 von der Firma Microchip, via der seriellen Schnittstelle RS232.

ElevenLabs

Abbildung 8 Logo ElevenLabs

Link zum Logo: <https://elevenlabs.io/brand>

2.2.4.5.3 Messung

Der PIC16F1827 ist für die tatsächliche Gewichtsmessung zuständig und wertet die Daten aus. Dazu werden nacheinander die 12 verbauten Sensoren ausgemessen und durch eine Berechnung annähernd linearisiert. Das ist nötig, weil sich der Widerstand der resistiven Drucksensoren (siehe Abbildung 11) sehr nichtlinear verhält und dadurch ein direkter Vergleich der gemessenen Rohdaten eine Herausforderung darstellt. Diese Sensoren basieren darauf, dass zwei Strom leitende Schichten bei Belastung zusammengedrückt werden, wodurch sich der elektrische Widerstand der Sensoren verringert.



Abbildung 9 PIC16F1827

https://atrs-online.com/web/p/mikrocontroller/6988959?cm_mm_c=AT-PLA-DS3A_-google_-CSS_AT_DE_Pmax_Test_-_-6988959&matchtype=&&gclid=aw.ds&gad_source=1&gad_campaignid=20298205821&gbraid=0AAAAADkeWNMs2adLaX9AT9ivtOIJdIeJ4q&gclid=Cj0KKQcIAYvHLbHdIARjAShXf6xoxcM4ihnuDflobRpwEin9iMt_qeGRGObCUjwqaBh3qk35PUx&gadAgeC7FAlw_wcB



Abbildung 11 Resistiver Drucksensor

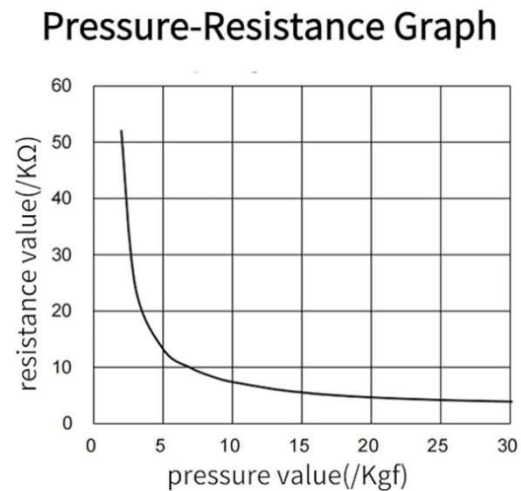


Abbildung 10 Widerstandsverlauf Drucksensor

Bilder von Händlerangaben:

https://www.amazon.de/gp/product/B098WPZJSR/ref=ox_sc_act_title_1?smid=A16GYQDD9HBXX&psc=1

Abbildung 10 zeigt den Verlauf des Widerstandswertes, der vom Hersteller angegeben wurde. Um diese Nichtlinearität auszugleichen wurden mehrere Lösungsansätze getestet. Aufgrund dessen, dass für die verwendeten Sensoren kein Datenblatt zu Verfügung stand, wurde das in Abbildung 10 gezeigte Diagramm an ChatGPT gesendet mit der Aufforderung, aus dem Verlauf eine mathematische Funktion zu erstellen. Allerdings war die Grafik für ChatGPT nicht ausreichend, um eine zutreffende Funktion daraus abzulesen. Aus dem Verlauf des Graphen konnte allerdings abgeleitet werden, dass es sich grundsätzlich um eine Funktion der Form $y(x) = \frac{1}{x}$ handelt.

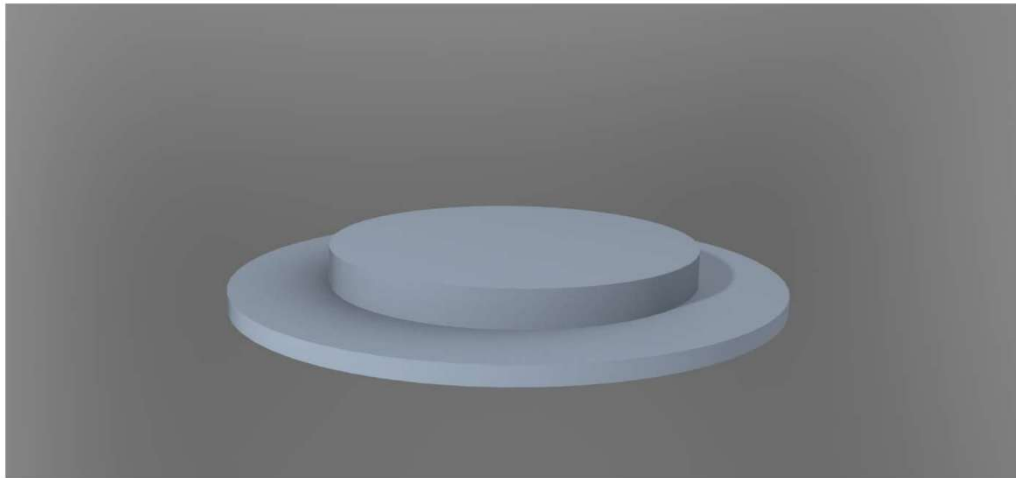


Abbildung 12 3D-Modell Stempel

Auf Basis dieser Annahme war die nächste Idee, einen einzelnen Sensor mit bekannten Gewichten zu belasten und die Messwerte dazu zu verwenden, auf die Funktion rückzuschließen. Dazu wurde der in Abbildung 12 gezeigte Stempel 3D-gedruckt, um das Gewicht eines mit Wasser gefüllten Topfes auf die Messfläche des Sensors zu fokussieren. Die Messwerte haben allerdings trotz gleichbleibendem Gewicht starke Schwankungen aufgewiesen, weswegen auch dieser Versuch zu keiner Lösung geführt hat.

Nach Besprechungen mit den betreuenden Lehrkräften wurde beschlossen, die Wurzel aus den Messwerten zu ziehen, da dadurch kleine Werte vergrößert und große Werte verkleinert werden. Ein Beispiel für eine Linearisierung mit Wurzelberechnung zeigt die rote Kurve in Abbildung 13. Der für die Messung relevante Bereich ist markiert. Außerdem wurde darauf verzichtet, den gemessenen Wert einem Gewicht zuzuordnen, da Matteo nur einen prozentuellen Vergleich der Werte vom rechten und linken Fuß anstellt und das tatsächliche Gewicht daher irrelevant ist.

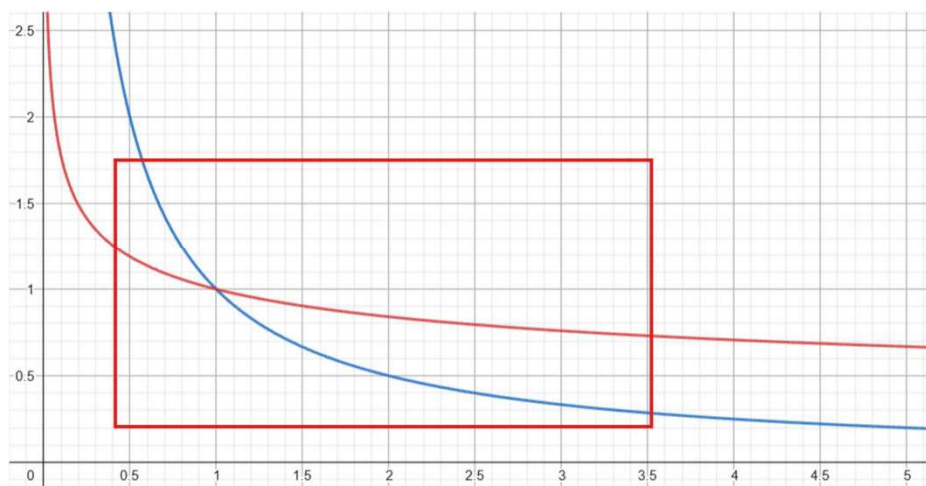
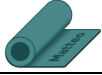


Abbildung 13 Beispiel Linearisierung

Bild erstellt mit GeoGebra: <https://www.geogebra.org/calculator>



Nachdem die Messwerte auf vergleichbare Werte umgerechnet wurden, werden sie addiert und dadurch die gesamte Belastung beider Füße berechnet. Dieser Wert wird dann als Basis für die Berechnung der prozentuellen Gewichtsverteilung verwendet. Es wurden auch schon erste Tests mit dieser Berechnungsmethode durchgeführt, bei denen plausible Ergebnisse für die Gewichtsverteilung gemessen wurden. Das Ergebnis der Belastungsverteilung wird anschließend zum ESP32-Microcontroller gesendet, der dann wieder entsprechende Aktionen setzt, also beispielsweise die Übung beendet und die entsprechende Audiodatei abspielt, falls das eingestellte Ziel erreicht sein sollte.

2.2.5 Nutzungsrecht

Da die Idee von der Universität Wien stammt und das Projekt von ihr in Auftrag gegeben wurde, fällt das Nutzungsrecht des Projektes an die Universität und das Projekt „Sinnvoll Aktiv“. Aus diesem Grund sind auch keine Patentrecherchen oder -anmeldungen durchgeführt worden.

2.3 Aktueller Stand und Ergebnisse

Zum jetzigen Zeitpunkt ist ein großer Teil des Projektes bereits abgeschlossen. Die Sprachsteuerung und Sprachausgabe funktionieren bereits und die Messschaltung wurde getestet. Auch eine Testversion des finalen Elektronik-Designs wurde bereits getestet und kleine Anpassungen vorgenommen. Im Rahmen des Tages der offenen Tür am 23.1.2026 wurden auch zum ersten Mal alle Komponenten gemeinsam erfolgreich in Betrieb genommen, was durch einen Projektbetreuer sowie Vertreter des Universitätsportinstitutes überprüft wurde. Außerdem steht das Design des Gehäuses bereits kurz vor der Finalisierung und wird 3D-gedruckt, sobald der finale Elektronikentwurf getestet ist und gesichert keine Änderungen mehr auftreten.

2.3.1 Elektronik

Die folgenden drei Bilder zeigen das Design der Leiterplatte als 3D-Modell, auf der alle elektronischen Bauteile miteinander verbunden werden. Abbildung 14 zeigt den Abschnitt mit der Stromversorgung, Abbildung 15 die Messschaltung und Abbildung 16 den Abschnitt, der für die Spracherkennung und Steuerung zuständig sind.

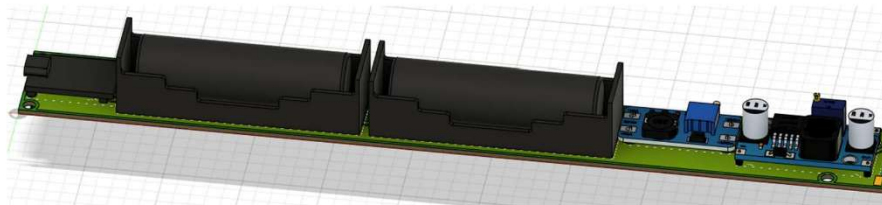


Abbildung 14 Stromversorgung

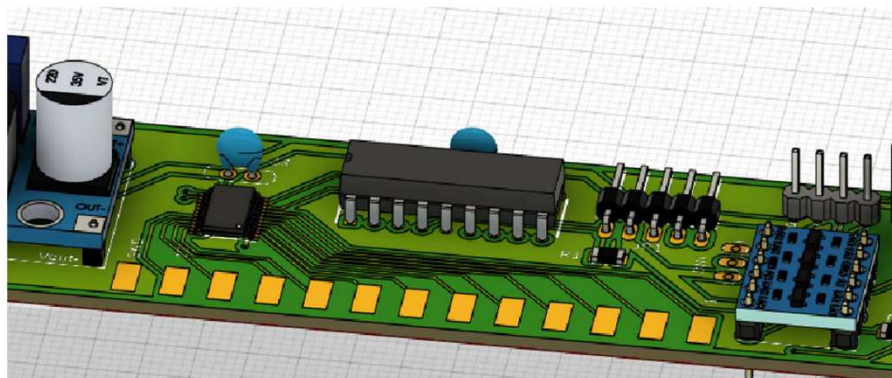


Abbildung 15 Messschaltung

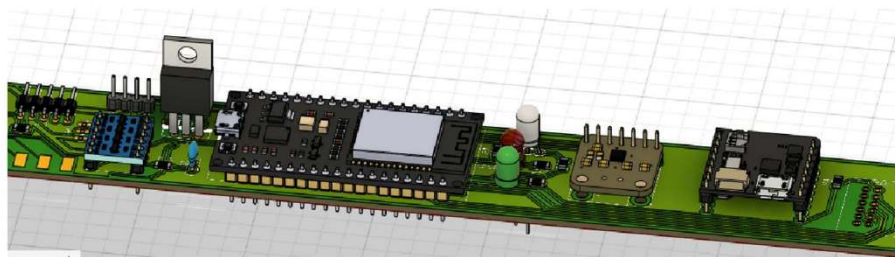


Abbildung 16 Sprachsteuerung

2.3.2 Konstruktion



Abbildung 17 3D-Modell Gehäuse

Abbildung 17 zeigt das Gehäuse der Matte, wie es nach momentanem Stand aussehen wird. Es ist aus mehreren kleinen Teilen aufgebaut, die nach dem 3D-Druck zusammengefügt werden. In Abbildung 18 ist das offene Gehäuse mit dem vorgesehenen Platz für die Elektronik abgebildet.

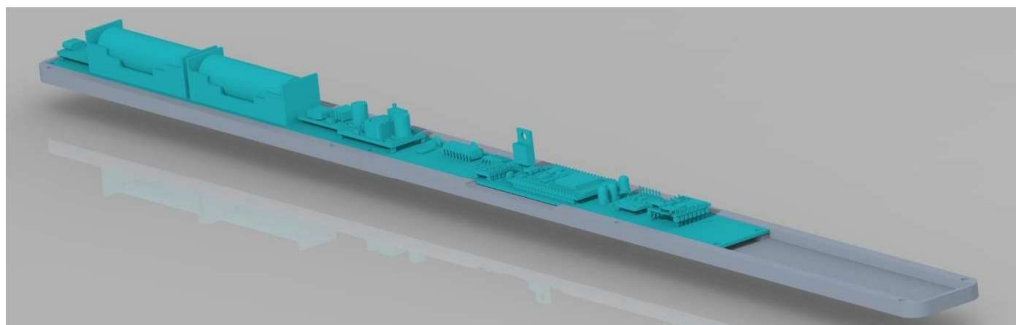


Abbildung 18 3D-Modell Gehäuse offen

2.3.3 Programmierung

Die Programmierung aller Prozessoren ist bereits abgeschlossen und getestet. Darunter fällt das Programm zur Messung des Gewichts und die Ablaufsteuerung. Eventuell können durch optionale Ziele noch kleinere Änderungen auftreten, die integriert werden müssen, was aber kein Problem darstellen sollte.

2.3.4 Spracherkennung

Das Sprachmodell erkennt aktuell über 95% der trainierten Befehle und ordnet sie korrekt zu. Dennoch wird es noch weiter verbessert, um Hintergrundgeräusche besser herausfiltern zu können. In Zukunft könnten auch noch weitere Befehle hinzugefügt werden, da das Sprachmodell im Rahmen der Rechenleistung des verwendeten Microcontrollers erweitert werden kann. In der aktuellen Version gibt es 17 Befehle, die die Sprachsteuerung erkennen kann.

2.3.5 Ausblick

Im Laufe der nächsten 1,5 Monate werden die letzten Schritte zum fertigen Produkt durchgeführt. Dazu gehört die Auswahl einer geeigneten Dicke für die Matte, in der die Sensorik verbaut wird, und für die Platten, die den Druck auf jeweils 3 Sensoren verteilen. Außerdem muss die Elektronik fertiggestellt und in das Gehäuse eingebaut werden, das möglichst bald hergestellt wird. Im Anschluss daran müssen die Sensoren in die Matte eingebaut und mit der Elektronik verbunden werden. Abschließend müssen Schutzvorkehrungen getroffen werden, um eine Beschädigung der Sensorik durch äußere Einwirkungen zu verhindern, zum Beispiel durch einen unebenen Untergrund, der zu einem Kabelbruch führen könnte.

Eventuell wird der fertige Prototyp der Matte zu Testzwecken an einen der kooperierenden Vereine übergeben. Diese sind:

- Blinden- und Sehbehindertenverband Wien, Niederösterreich und Burgenland
- Versehrtensportklub ASVÖ-Wien
- Hilfsgemeinschaft der Blinden und Sehbehinderten Österreichs

Dadurch könnte Feedback eingeholt werden, wie die Matte in Zukunft weiterentwickelt werden kann. Alle Vereine wurden auch bereits im September darüber informiert, dass es die Möglichkeit geben wird, einen Prototyp zu testen und so an der Entwicklung mitzuwirken.

Im März 2026 wird die Projektabnahme durchgeführt und im April die Defensio abgehalten, bei der alle Projektmitglieder von ihren jeweiligen individuellen Betreuern zu ihren Aufgabenbereichen befragt werden. Sobald das Projekt abgeschlossen ist, wird es an die Universität Wien und „Sinnvoll Aktiv“ übergeben.

2.4 Verwertung nach Abschluss des Projektes

Geplant ist, dass Matteo im Rahmen von „Sinnvoll Aktiv“ Verwendung findet. Insbesondere sollen Sportgruppen diverser Vereine für Blinde und Sehschwache davon profitieren, die Matteo auch gemeinsam nutzen können. Sollte sich dieser Plan verwirklichen, hat Matteo potenziell große Auswirkungen auf die Lebensqualität vieler Menschen, wenn sie wieder die Möglichkeit bekommen, zu einem gewissen Grad eigenständig Sport zu betreiben. Auch bei den anderen weiter oben genannten Anwendungsmöglichkeiten wie Physiotherapie und Reha-Aufenthalten könnte Matteo eine Rolle spielen, in dem beispielsweise Patient:innen durch genaue Messungen Rückmeldung bekommen, wie sie ihre Übungen ausführen und dadurch eine Basis für eine korrekte Ausübung geschaffen werden kann.



Abbildung 19 Logo Sinnvoll Aktiv

Logo von Sinnvoll Aktiv:
<https://www.sinnvollaktiv.at/>

Ein weiterer Anwendungsbereich ist auch Nutzung von älteren Personen, die nicht erblindet sind, da die Balance im Alter auch ohne zusätzliche Erschwernisse abnimmt. Dadurch würde sich eine neue Sparte öffnen, über die auch kommerzielle Erfolge erzielt werden könnten. Die Entscheidung darüber liegt allerdings bei den Verantwortlichen der Universität und wurde noch nicht besprochen.

2.5 Verbesserungsmöglichkeiten

Dadurch, dass Matteo im Rahmen eines Diplomprojektes entsteht, ist nur ein kleiner Teil des eigentlichen Potenzials ausgenutzt. Deshalb kann fast jeder Aspekt des von Matteo weiterentwickelt oder erweitert werden.

2.5.1 Sensorik

Im Rahmen der Diplomarbeit werden 12 Sensoren auf einer im Verhältnis zu Mattengröße eher kleinen Fläche verbaut. Für die aktuelle Anwendung ist das völlig ausreichend, es könnte aber in Zukunft die Anzahl der Sensoren erhöht werden. Dadurch müssten die Füße nicht mehr genau auf den Platten positioniert werden, um eine Messung durchführen zu können und die Nutzung würde sich weiter vereinfachen.

2.5.2 Erweiterte Übungsmöglichkeiten

Aufgrund dessen, dass pro Fuß zwei Platten verbaut sind, ergibt sich die Möglichkeit einer Messung der Belastung am vorderen und hinteren Teil des Fußes. Dadurch könnte das Übungsangebot von einfachen Balanceübungen mit Gewichtsverlagerung nach links und rechts darum erweitert werden, dass zusätzlich noch der Belastungsunterschied der vorderen bzw. hinteren Teile der Füße gemessen und in den Übungen gefordert werden könnte. Die Balance könnte so noch weiter verbessert werden und Anwendungen im medizinischen Bereich an Genauigkeit und Informationsgehalt gewinnen.

2.5.3 Erweitertes Feedback

Das visuelle Feedback, das momentan aus einer LED besteht, könnte so erweitert werden, dass durch LEDs nicht nur der Abschluss einer Übung angezeigt wird. Es könnte eine Anzeige des Batteriestands und eine Positionierungshilfe auf der Matte umgesetzt werden. Außerdem könnten Statusanzeigen hinzugefügt werden, die beispielsweise einen Fehler im Programm, den aktuellen Zustand einer Übung, die aktuelle Belastung oder die Lautstärke anzeigen. All diese Optionen könnten auch in der Sprachausgabe Anwendung finden, dazu müssten allerdings auch entsprechende Sprachbefehle hinzugefügt werden.

2.5.4 Übertragen der Daten auf Website

Durch die Verwendung eines ESP32 besteht die Möglichkeit, dass die Messdaten in Echtzeit über WLAN oder Bluetooth an ein Smartphone oder einen Computer übertragen werden. Diese Option könnte vor allem bei Physiotherapie hilfreich sein, da die Therapeut:innen genau sehen könnten, welche Bereiche der Matte besonders belastet werden und daraus Rückschlüsse auf die individuellen Problemstellungen ziehen könnten.

2.6 Projektteam und -organisation

Seit dem Anfang des Projektes vor ungefähr einem Jahr hat sich das Projektumfeld immer weiter vergrößert. Ausgehend vom Kernteam, also den 4 Teammitgliedern Nandor Bagari, Mika Böhm, Lisa Pripfl und mir selbst, Laurenz Wimmer, sind zuerst unsere Betreuer Ewald Cekan, Richard Drechsler und Martin Meschik dazugekommen. Nachdem die Entscheidung gefallen ist, die Idee der Universität umzusetzen, sind Stefan Andreas Petter als Ansprechpartner und Stellvertreter von „Sinnvoll Aktiv“ und der Universität Wien sowie Philipp Kornfeind als technischer Berater der Universität immer mehr involviert gewesen. Im Herbst sind dann noch die Kontakte der möglicherweise kooperationsbereiten Vereine sowie externe Sponsoren dazugekommen.

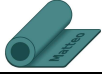
Grundsätzlich ist die Kommunikation mit allen wichtigen Beteiligten sehr gut verlaufen. Vor allem im schulinternen Projektteam ist die Kommunikation sehr gut. Einen wichtigen Beitrag dazu hat auch ein Team-Coaching geleistet, das einer unserer Lehrer angeboten hat. Im Rahmen des Coachings konnten wir Punkte herausarbeiten, die Anfangs zu Missverständnissen geführt haben und haben es geschafft, diese Hindernisse durch regelmäßige Updates zum aktuellen Arbeitsstand jedes einzelnen Teammitgliedes zu beseitigen. Auch die Kommunikation mit den Betreuern ist sehr gut verlaufen. Wir halten alle 14 Tage ein Jour Fixe Meeting ab, bei dem alle aktuellen Entwicklungen, Probleme und Ideen besprochen werden. Wir können auch jederzeit außerhalb dieser Meetings mit den Betreuern reden, falls Fragen oder Probleme aufkommen und erhalten immer konstruktive Antworten und Vorschläge.

Der Austausch mit Stefan Andres Petter und den anderen Sponsoren Cadaxo und Printshop Landstraße erfolgt in erster Linie via E-Mail, wir haben aber auch schon einige Online-Meetings und Treffen in Person im Rahmen vom Tag der offenen Tür und „Sinnvoll Aktiv“-Events abgehalten. Sobald wir eine Bestellliste erstellt hatten oder sonstige Bauteile gebraucht haben, konnten wir damit rechnen, spätestens 2 Tage später eine Antwort von Stefan Andreas Petter zu erhalten mit der Info, dass die Bestellung aufgegeben wurde. Herr Petter hat uns auch sehr damit unterstützt, dass er alle Bauteile persönlich in unserer Schule hinterlegt hat, damit wir möglichst schnell weiterarbeiten können.

2.6.1 Motivation

Am Anfang des Projektes hat ein Teammitglied aufgrund der Aufgabenverteilung wenig Motivation gehabt, am Projekt zu arbeiten. Entstanden ist diese Situation dadurch, dass die Zuteilung der groben Ziele beschlossen wurde, bevor ein konkretes Konzept entstanden ist und dadurch die Erwartungen in Bezug auf die zu erledigenden Arbeiten des Teammitglieds nicht erfüllt wurden. Vor allem aber nach dem Team-Coaching hat sich die Situation gebessert, auch wenn noch immer keine vollständige Zufriedenheit entstanden ist. Die Aufgaben wurden aber trotzdem perfekt umgesetzt und es hat keine weiteren Probleme diesbezüglich gegeben.

Durch die unterschiedlichen großen Umfänge der Ziele haben manche Teammitglieder insgesamt mehr zu tun als andere. Damit aber keine Überbelastung entsteht, versuchen wir, uns gegenseitig bei unseren Aufgaben zu unterstützen. So haben wir es auch geschafft, die Arbeitsmoral grundsätzlich hochzuhalten und alle Aufgaben in einem überschaubaren



zeitlichen Rahmen abzuschließen. Es hat dabei auch geholfen, dass unser Team in einer ähnlichen Zusammensetzung schon in der 4. Klasse ein Projekt durchgeführt hat und eingespielt war. Daher hatten wir auch keine Konflikte, die das Projekt gefährdet haben könnten oder nicht durch ein Gespräch mit den Betreuern geklärt werden konnten.

Zusammenfassend habe ich festgestellt, dass wir seit September sehr harmonisch mit allen Beteiligten zusammengearbeitet haben und uns grundsätzlich in Bezug auf das Arbeiten in einem Team weiterentwickelt haben, weswegen wir auch für zukünftige Projekte wertvolle Erfahrungen mitnehmen können.

2.6.2 Ressourcen

Die bis jetzt eingesetzten finanziellen Ressourcen belaufen sich aufgrund der diversen elektronischen Bauteile insgesamt auf schätzungsweise 830€. Darin sind alle bisher bestellten Materialien enthalten, auch 3D-Druckfilament für das Gehäuse. Abbildung 20 zeigt einen Vergleich der Kosten mit dem Sponsoring.

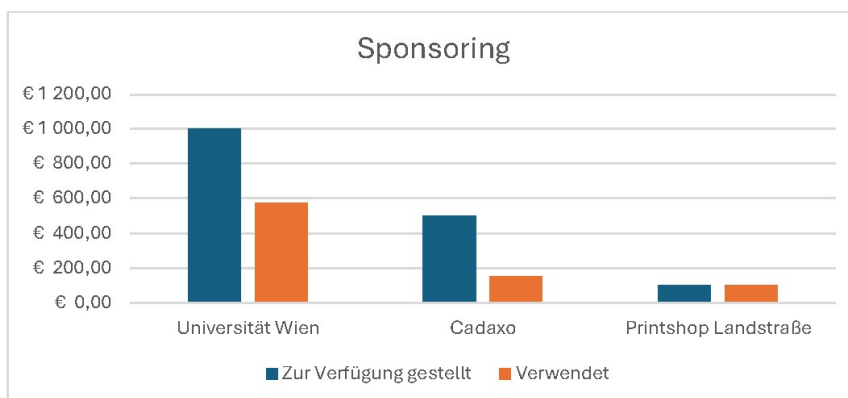


Abbildung 20 Sponsoring

Wie in Abbildung 20 dargestellt ist, stehen uns noch über die Hälfte der zugesagten Sponsorings zur Verfügung.

Die Arbeitszeit der einzelnen Teammitglieder ist momentan noch sehr unterschiedlich, da bei den Hardware-Zielen noch einige Arbeitsschritte zu erledigen sein. Eine ungefähre Stundenanzahl der einzelnen Projektmitarbeiter ist in Abbildung 21 zu sehen.

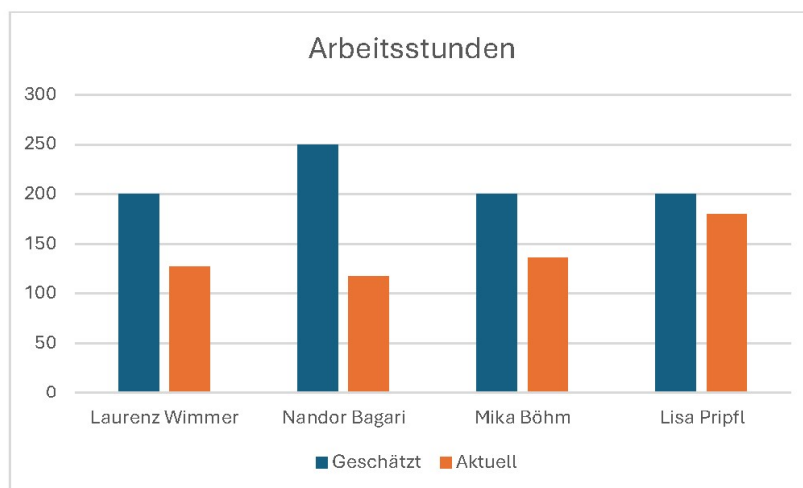
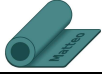


Abbildung 21 Arbeitszeit



3 Quellenverzeichnis

Erklärung zur Erstellung der Impulse mit Edge Impulse:

<https://docs.edgeimpulse.com/studio/projects/impulse-design>

Allgemeine Erklärung, wie der Arduino Nicla Voice mit Edge Impulse verwendet wird:

<https://docs.edgeimpulse.com/hardware/boards/arduino-nicla-voice>

Erklärung des Audio-Blocks von Edge Impulse, mit dem die Audiodaten für den Arduino Nicla Voice verarbeitet werden:

<https://docs.edgeimpulse.com/studio/projects/processing-blocks/blocks/audio-syntiant>

Erklärung des Learning-Blocks in Edge Impulse:

<https://docs.edgeimpulse.com/studio/projects/learning-blocks>

Erklärung für I2C-Schnittstelle des Arduino Nicla Voice:

<https://docs.edgeimpulse.com/projects/expert-network/voice-controlled-power-plugin-nicla-voice>

Beispielprogramm für I2C-Schnittstelle:

https://github.com/Jallson/Voice-Controlled-PowerPlug/blob/main/relayvoice_niclavoice.ino

Datenblatt 16-Kanal-Multiplexer:

https://www.ti.com/lit/ds/symlink/cd74hc4067.pdf?ts=1759843597022&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FCD74HC4067

Datenblatt Operationsverstärker:

<https://www.mouser.at/datasheet/3/282/1/MCP6001-1R-1U-2-4-1-MHz-Low-Power-Op-Amp-DS20001733L.pdf>

Datenblatt 8-Kanal-Multiplexer:

https://www.ti.com/lit/ds/symlink/cd74hc4051-ep.pdf?ts=1757350750317&ref_url=https%253A%252F%252Fwww.google.com%252F

Datenblatt MOSFET im Logic-Level-Converter:

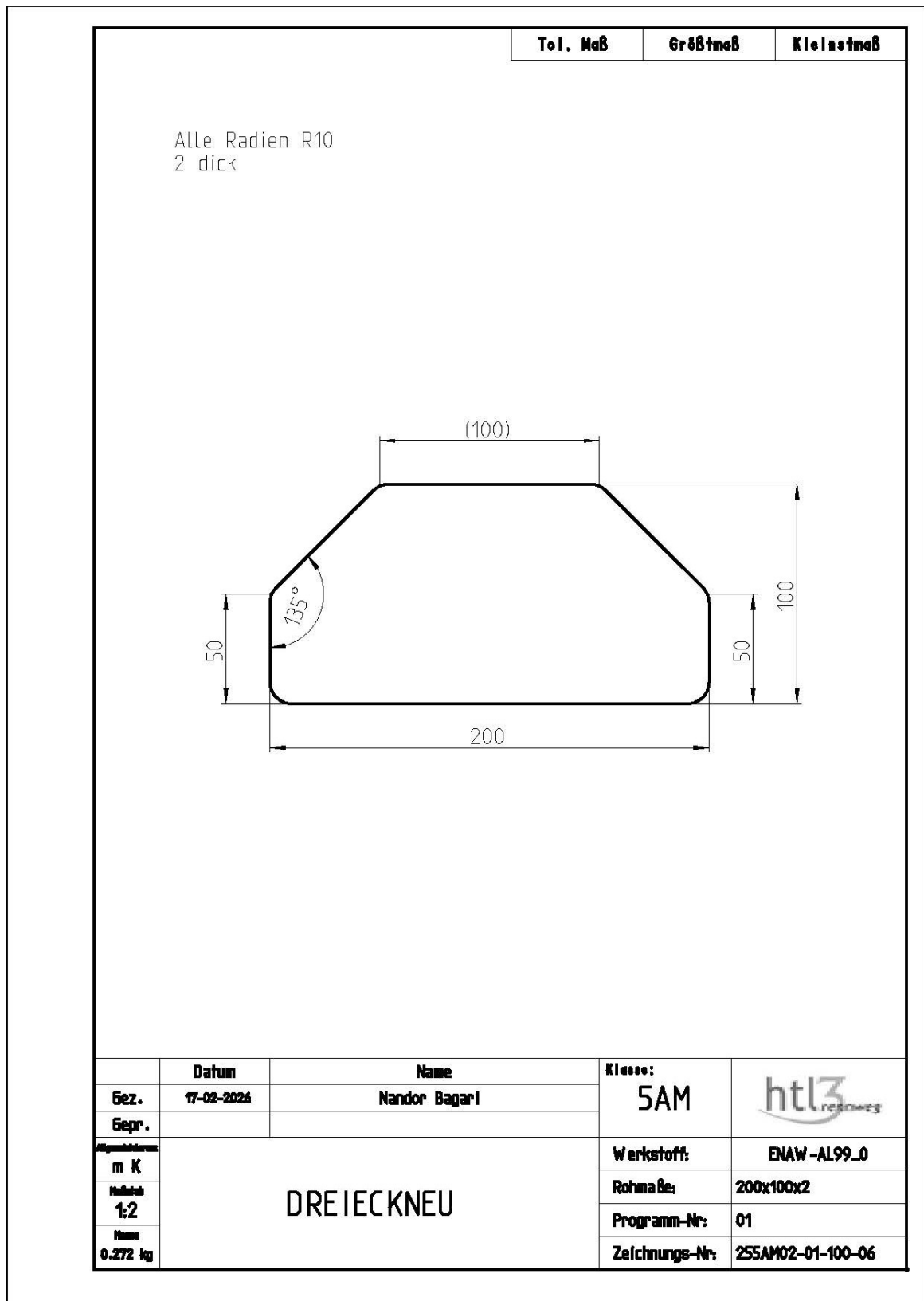
<https://www.onsemi.com/download/data-sheet/pdf/bss138-d.pdf>

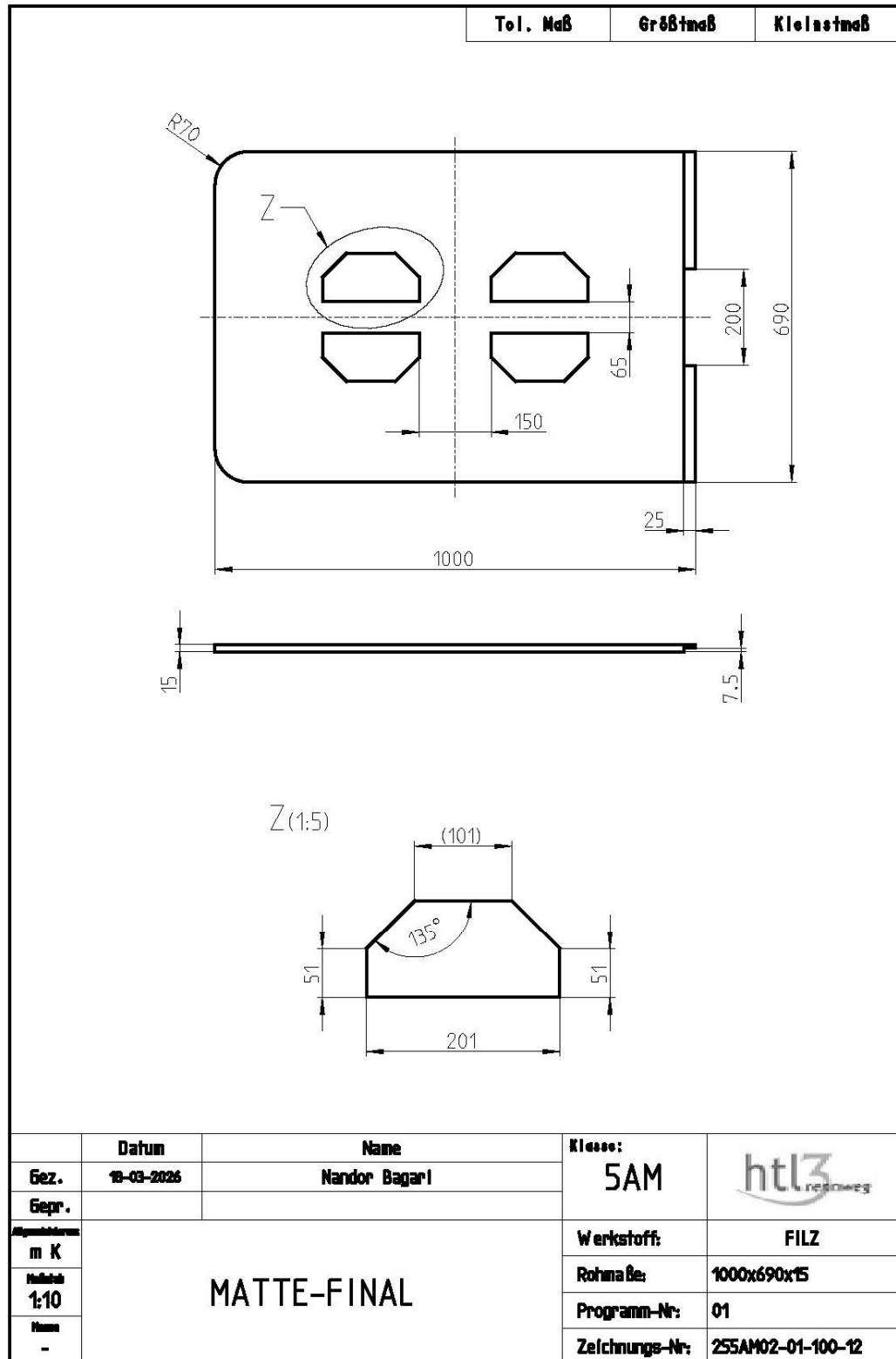


4 Abbildungsverzeichnis

Abbildung 1 3D-Modell Matteo.....	6
Abbildung 2 Sensorlayout	6
Abbildung 3 Logo Edge Impuls	7
Abbildung 4 Platten zur Druckverteilung	10
Abbildung 5 Übersicht Microprozessoren	11
Abbildung 6 Arduino Nicla Voice.....	11
Abbildung 7 ESP32 WROOM	12
Abbildung 8 Logo ElevenLabs.....	12
Abbildung 9 PIC16F1827	13
Abbildung 10 Widerstandsverlauf Drucksensor	13
Abbildung 11 Resistiver Drucksensor	13
Abbildung 12 3D-Modell Stempel	14
Abbildung 13 Beispiel Linearisierung	14
Abbildung 14 Stromversorgung.....	16
Abbildung 15 Messschaltung	16
Abbildung 16 Sprachsteuerung.....	16
Abbildung 17 3D-Modell Gehäuse	17
Abbildung 18 3D-Modell Gehäuse offen.....	17
Abbildung 19 Logo Sinnvoll Aktiv	18
Abbildung 20 Sponsoring	22
Abbildung 21 Arbeitszeit.....	22

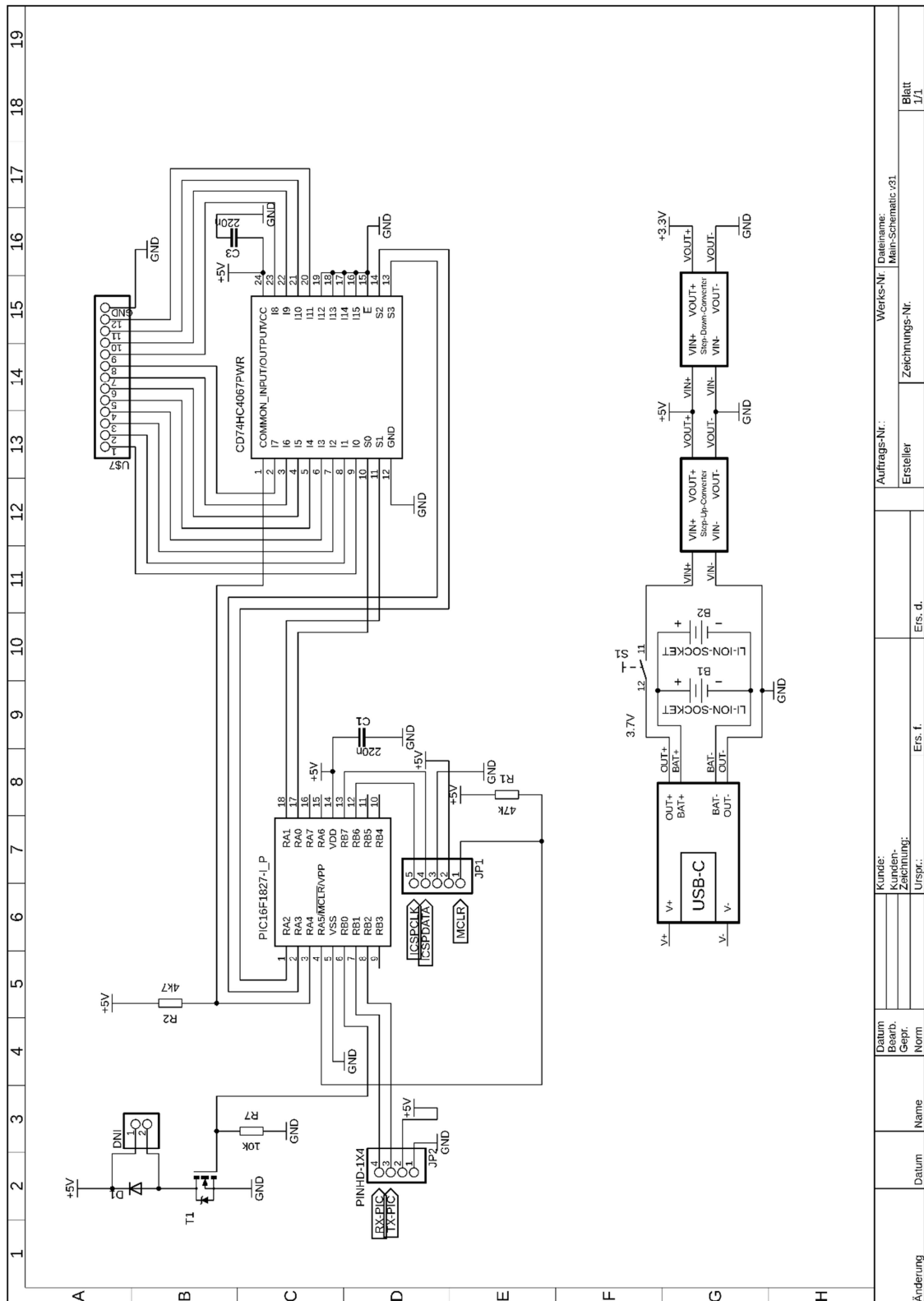
9.5 Konstruktionszeichnungen:







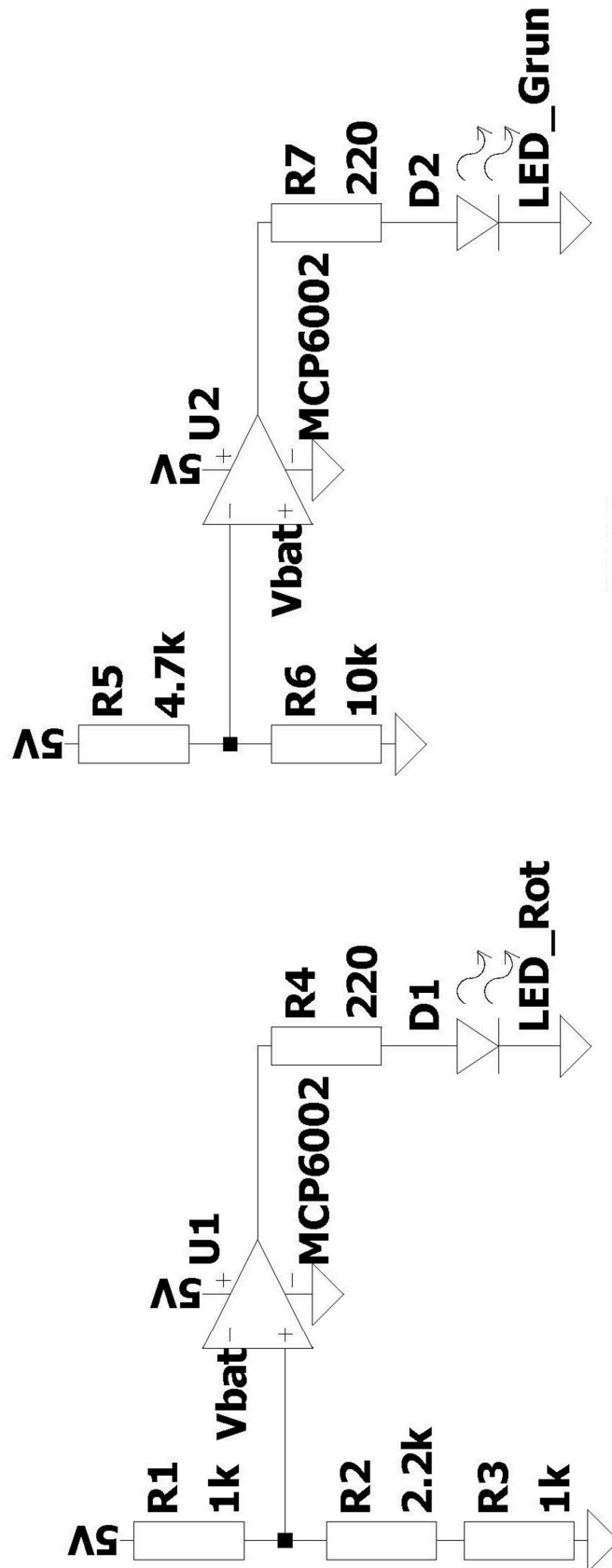
9.6.2 Hauptplatine Seite 1 (5 V)



Hefttrand



9.6.4 Akkuladestand Anzeige





9.7 Datenblätter

9.7.1 CD74HC4067 [4]



CD74HC4067, CD74HCT4067

SCHS209D – NOVEMBER 1998 – REVISED DECEMBER 2024

CD74HCx4067 High-Speed CMOS Logic 16-Channel Analog Multiplexer and Demultiplexer

1 Features

- Wide analog input voltage range
- Low ON resistance
 - $V_{CC} = 4.5V$, 70Ω (typ)
 - $V_{CC} = 6V$, 60Ω (typ)
- Fast switching and propagation speeds
- Break-before-make switching
 - 6ns (typ) at 4.5V
- Available in both narrow- and wide-body plastic packages
- Fanout (over-temperature range)
 - Standard outputs: 10 LSTTL loads
 - Bus driver outputs: 15 LSTTL loads
- Wide operating temperature range: $-55^{\circ}C$ to $+125^{\circ}C$
- Balanced propagation delay and transition times
- Significant power reduction compared to LSTTL logic ICs
- HC types
 - 2V to 6V operation
 - High noise immunity: $N_{IL} = 30\%$, $N_{IH} = 30\%$ of V_{CC} at $V_{CC} = 5V$
- HCT types
 - 4.5V to 5.5V operation
 - Direct LSTTL input logic compatibility, $V_{IL} = 0.8V$ (max), $V_{IH} = 2V$ (min)
 - CMOS input compatibility, $I_I \leq 1\mu A$ at V_{OL} , V_{OH}

2 Applications

- Energy infrastructure
- Building automation
- Wireless infrastructure
- Appliances
- Data center & enterprise computing
- Retail automation & payment

- Signal gating
- Modulators
- Squelch controls
- demodulators
- choppers
- commutating switches
- Analog-to-digital and digital-to-analog conversions
- Digital control of frequency, impedance, phase, and analog-signal gain

3 Description

The CD74HC4067 and CD74HCT4067 devices are digitally controlled analog switches that use silicon-gate CMOS technology to achieve operating speeds similar to LSTTL, with the low power consumption of standard CMOS integrated circuits.

These analog multiplexers and demultiplexers control analog voltages that may vary across the voltage supply range. They are bidirectional switches, thus allowing any analog input to be used as an output and vice-versa. The switches have low *on* resistance and low *off* leakages. In addition, these devices have an enable control that, when high, disables all switches to their *off* state.

Package Information

PART NUMBER	PACKAGE ⁽¹⁾	BODY SIZE (NOM)
CD74HC4067M	SOIC(24)	15.4mm × 10.3mm
CD74HC4067M96	SOIC(24)	15.4mm × 10.3mm
CD74HC4067SM96	SSOP(24)	8.20mm × 7.40mm
CD74HCT4067M	SOIC(24)	15.4mm × 10.3mm
CD74HC4067PW	TSSOP(24)	7.8mm × 6.4mm
CD74HCT4067PW	TSSOP(24)	7.8mm × 6.4mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

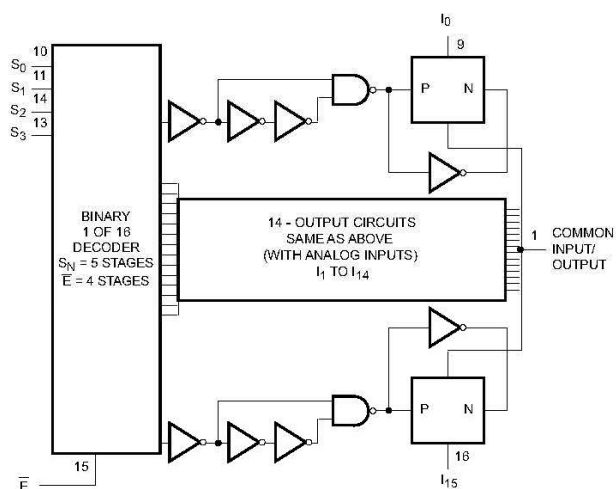


An IMPORTANT NOTICE at the end of this data sheet addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers. PRODUCTION DATA.



CD74HC4067, CD74HCT4067

SCHS209D – NOVEMBER 1998 – REVISED DECEMBER 2024



Functional Block Diagram



CD74HC4067, CD74HCT4067

SCHS209D – NOVEMBER 1998 – REVISED DECEMBER 2024

4 Pin Configuration and Functions

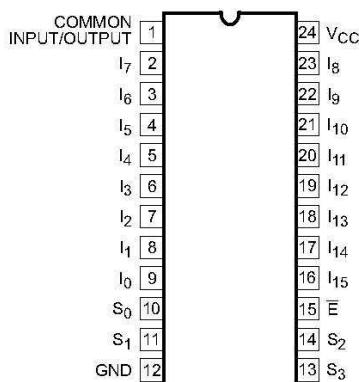


Figure 4-1. N, DW, or DB Packages 24-Pin PDIP, SOIC, or SSOP (Top View)

PIN		TYPE ⁽¹⁾	DESCRIPTION
NAME	NO.		
COMMON INPUT/OUTPUT	1	IO	Common input or output.
I ₇	2	IO	Switch input/output
I ₆	3	IO	Switch input/output
I ₅	4	IO	Switch input/output
I ₄	5	IO	Switch input/output
I ₃	6	IO	Switch input/output
I ₂	7	IO	Switch input/output
I ₁	8	IO	Switch input/output
I ₀	9	IO	Switch input/output
S ₀	10	I	Select/Address pin
S ₁	11	I	Select/Address pin
GND	12	P	Ground pin
S ₃	13	I	Select/Address pin
S ₂	14	I	Select/Address pin
E	15	I	Enable for all switches ON/OFF
I ₁₅	16	IO	Switch input/output
I ₁₄	17	IO	Switch input/output
I ₁₃	18	IO	Switch input/output
I ₁₂	19	IO	Switch input/output
I ₁₁	20	IO	Switch input/output
I ₁₀	21	IO	Switch input/output
I ₉	22	IO	Switch input/output
I ₈	23	IO	Switch input/output
VCC	24	P	Power pin

(1) I = input, O = output, P = Power



4.1 Device Functional Modes

Table 4-1. Truth Table

S0	S1	S2	S3	E	SELECTED CHANNEL
X	X	X	X	1	None
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	0	2
1	1	0	0	0	3
0	0	1	0	0	4
1	0	1	0	0	5
0	1	1	0	0	6
1	1	1	0	0	7
0	0	0	1	0	8
1	0	0	1	0	9
0	1	0	1	0	10
1	1	0	1	0	11
0	0	1	1	0	12
1	0	1	1	0	13
0	1	1	1	0	14
1	1	1	1	0	15

5 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)^{(1) (2)}

		MIN	MAX	UNIT
V _{CC} HC	DC Supply voltage	-0.5	7	V
V _{CC} HCT		-0.5	7	V
I _{IK}	DC input diode current	For V _I < -0.5V or V _I > V _{CC} + 0.5V		20 mA
I _{OK}	DC output diode current	For V _O < -0.5V or V _O > V _{CC} + -0.5V		20 mA
I _{CC}	DC V _{CC} or ground current	-50	50	mA
DC Output Source or Sink Current per Output Pin, I _O	For V _O > -0.5V or V _O < V _{CC} + -0.5V	-25	25	mA
T _{JMAX}	Maximum junction temperature (Plastic Package)		150	°C
T _{stg}	Storage temperature	-65	150	°C

- (1) Stresses beyond those listed under *Absolute Maximum Rating* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Condition*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
- (2) All voltages are with respect to ground, unless otherwise specified.



8 Electrical Characteristics: HC Devices

Over operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS				MIN	TYP	MAX	UNIT
Analog Switch									
		V _{IS} (V)	V _I (V)	V _{CC} (V)	T _A				
High Level Input Voltage	V _{IH}			2	25°C	1.5		V	
					–40°C to +85°C	1.5			
					–55°C to +125°C	1.5			
				4.5	25°C	3.15			
					–40°C to +85°C	3.15			
					–55°C to +125°C	3.15			
				6	25°C	4.2			
					–40°C to +85°C	4.2			
					–55°C to +125°C	4.2			
Low Level Input Voltage	V _{IL}			2	25°C	0.5		V	
					–40°C to +85°C	0.5			
					–55°C to +125°C	0.5			
				4.5	25°C	1.35			
					–40°C to +85°C	1.35			
					–55°C to +125°C	1.35			
				6	25°C	1.8			
					–40°C to +85°C	1.8			
					–55°C to +125°C	1.8			
"ON" Resistance IO = 1mA	R _{ON}	V _{CC} or GND	V _{CC} or GND	4.5	25°C	70	160	Ω	
					–40°C to +85°C	200			
					–55°C to +125°C	240			
				6	25°C	60	140		
					–40°C to +85°C	175			
					–55°C to +125°C	210			
		V _{CC} to GND	V _{CC} to GND	4.5	25°C	90	180	Ω	
					–40°C to +85°C	225			
					–55°C to +125°C	270			
				6	25°C	80	160		
					–40°C to +85°C	200			
					–55°C to +125°C	240			
"ON" Resistance Between Any Two Switches	ΔR _{ON}			4.5	25°C	10	Ω		
				6	25°C	8.5			
Off-Switch Leakage Current	I _Z	E = V _{CC}	V _{CC} or GND	6	25°C	±0.8		μA	
					–55°C to 85°C	±8			
					–55°C to 125°C	±8			
Input Leakage Current (Any Control)	I _{IL}		V _{CC} or GND ⁽¹⁾	6	25°C	±0.1		μA	
					–55°C to 85°C	±1			
					–55°C to 125°C	±1			



9.7.2 BSS138 [4]



DATA SHEET
www.onsemi.com

N-Channel Logic Level Enhancement Mode Field Effect Transistor

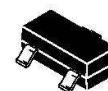
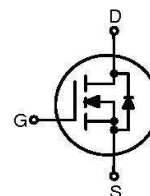
BSS138

General Description

These N-Channel enhancement mode field effect transistors are produced using **onsemi's** proprietary, high cell density, DMOS technology. These products have been designed to minimize on-state resistance while provide rugged, reliable, and fast switching performance. These products are particularly suited for low voltage, low current applications such as small servo motor control, power MOSFET gate drivers, and other switching applications.

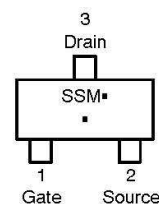
Features

- 0.22 A, 50 V
 - ♦ $R_{DS(on)} = 3.5 \Omega @ V_{GS} = 10 V$
 - ♦ $R_{DS(on)} = 6.0 \Omega @ V_{GS} = 4.5 V$
- High Density Cell Design for Extremely Low $R_{DS(on)}$
- Rugged and Reliable
- Compact Industry Standard SOT-23 Surface Mount Package
- HBM Class 0A, MM Class M2 (Note 3)
- This Device is Pb-Free and Halogen Free



SOT-23-3
CASE 318-08

MARKING DIAGRAM



SS = Specific Device Code
M = Date Code*
▪ = Pb-Free Package

(Note: Microdot may be in either location)

*Date Code orientation and/or position may vary depending upon manufacturing location.

ORDERING INFORMATION

Device	Package	Shipping†
BSS138, BSS138-G	SOT-23-3 (Pb-Free)	3000 / Tape & Reel

†For information on tape and reel specifications, including part orientation and tape sizes, please refer to our Tape and Reel Packaging Specification Brochure, BRD8011/D.



BSS138

ABSOLUTE MAXIMUM RATINGS $T_A = 25^\circ\text{C}$ unless otherwise noted.

Symbol	Parameter	Ratings	Unit
V_{DS}	Drain–Source Voltage	50	V
V_{GS}	Gate–Source Voltage	± 20	
I_D	Drain Current – Continuous (Note 1)	0.22	A
	Drain Current – Pulsed (Note 1)	0.88	
P_D	Maximum Power Dissipation (Note 1)	0.36	W
	Derate Above 25°C	2.8	mW/ $^\circ\text{C}$
T_J, T_{STG}	Operating and Storage Junction Temperature Range	-55 to $+150$	$^\circ\text{C}$
T_L	Maximum Lead Temperature for Soldering Purposes, 1/16" from Case for 10 s	300	

Stresses exceeding those listed in the Maximum Ratings table may damage the device. If any of these limits are exceeded, device functionality should not be assumed, damage may occur and reliability may be affected.

THERMAL CHARACTERISTICS $T_A = 25^\circ\text{C}$ unless otherwise noted.

Symbol	Parameter	Ratings	Unit
$R_{\theta JA}$	Thermal Resistance, Junction–to–Ambient (Note 1)	350	$^\circ\text{C}/\text{W}$

ELECTRICAL CHARACTERISTICS $T_A = 25^\circ\text{C}$ unless otherwise noted.

Symbol	Parameter	Test Conditions	Min	Typ	Max	Unit
--------	-----------	-----------------	-----	-----	-----	------

OFF CHARACTERISTICS

BV_{DS}	Drain–Source Breakdown Voltage	$V_{GS} = 0\text{ V}, I_D = 250\text{ }\mu\text{A}$	50	–	–	V
$\frac{\Delta V_{GS(th)}}{\Delta T_J}$	Breakdown Voltage Temperature Coefficient	$I_D = 250\text{ }\mu\text{A}$, Referenced to 25°C	–	72	–	mV/ $^\circ\text{C}$
I_{DSS}	Zero Gate Voltage Drain Current	$V_{DS} = 50\text{ V}, V_{GS} = 0\text{ V}$	–	–	0.5	μA
		$V_{DS} = 50\text{ V}, V_{GS} = 0\text{ V}, T_J = 125^\circ\text{C}$	–	–	5	
		$V_{DS} = 30\text{ V}, V_{GS} = 0\text{ V}$	–	–	100	nA
I_{GSS}	Gate–Body Leakage	$V_{GS} = \pm 20\text{ V}, V_{DS} = 0\text{ V}$	–	–	± 100	

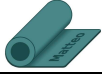
ON CHARACTERISTICS

$V_{GS(th)}$	Gate Threshold Voltage	$V_{DS} = V_{GS}, I_D = 1\text{ mA}$	0.8	1.3	1.5	V
$\frac{\Delta V_{GS(th)}}{\Delta T_J}$	Gate Threshold Voltage Temperature Coefficient	$I_D = 1\text{ mA}$, Referenced to 25°C	–	–2	–	mV/ $^\circ\text{C}$
$R_{DS(on)}$	Static Drain–Source On–Resistance	$V_{GS} = 10\text{ V}, I_D = 0.22\text{ A}$	–	0.7	3.5	Ω
		$V_{GS} = 4.5\text{ V}, I_D = 0.22\text{ A}$	–	1.0	6.0	
		$V_{GS} = 10\text{ V}, I_D = 0.22\text{ A}, T_J = 125^\circ\text{C}$	–	1.1	5.8	
$I_{D(on)}$	On–State Drain Current	$V_{GS} = 10\text{ V}, V_{DS} = 5\text{ V}$	0.2	–	–	A
g_{FS}	Forward Transconductance	$V_{DS} = 10\text{ V}, I_D = 0.22\text{ A}$	0.12	0.5	–	S

DYNAMIC CHARACTERISTICS

C_{iss}	Input Capacitance	$V_{DS} = 25\text{ V}, V_{GS} = 0\text{ V}, f = 1.0\text{ MHz}$	–	27	–	pF
C_{oss}	Output Capacitance		–	13	–	pF
C_{rss}	Reverse Transfer Capacitance		–	6	–	pF
R_G	Gate Resistance	$V_{GS} = 15\text{ mV}, f = 1.0\text{ MHz}$	–	9	–	Ω





BSS138

ELECTRICAL CHARACTERISTICS $T_A = 25^\circ\text{C}$ unless otherwise noted. (continued)

Symbol	Parameter	Test Conditions	Min	Typ	Max	Unit
--------	-----------	-----------------	-----	-----	-----	------

SWITCHING CHARACTERISTICS

$t_{d(on)}$	Turn-On Delay Time	$V_{DD} = 30\text{ V}, I_D = 0.29\text{ A},$ $V_{GS} = 10\text{ V}, R_{GEN} = 6\ \Omega$	–	2.5	5	ns
t_r	Turn-On Rise Time		–	9	18	ns
$t_{d(off)}$	Turn-Off Delay Time		–	20	36	ns
t_f	Turn-Off Fall Time		–	7	14	ns
Q_g	Total Gate Charge	$V_{DS} = 25\text{ V}, I_D = 0.22\text{ A},$ $V_{GS} = 10\text{ V}$	–	1.7	2.4	nC
Q_{gs}	Gate-Source Charge		–	0.1	–	nC
Q_{gd}	Gate-Drain Charge		–	0.4	–	nC

DRAIN-SOURCE DIODE CHARACTERISTICS AND MAXIMUM RATINGS

I _S	Maximum Continuous Drain–Source Diode Forward Current		–	–	0.22	A
V _{SD}	Drain–Source Diode Forward Voltage	V _{GS} = 0 V, I _S = 0.44 A (Note 2)	–	0.8	1.4	V

Product parametric performance is indicated in the Electrical Characteristics for the listed test conditions, unless otherwise noted. Product performance may not be indicated by the Electrical Characteristics if operated under different conditions.

- $R_{\theta JA}$ is the sum of the junction-to-case and case-to-ambient thermal resistance where the case thermal reference is defined as the solder mounting surface of the drain pins. $R_{\theta JA}$ is guaranteed by design while $R_{\theta JA}$ is determined by the user's board design.



- 350°C/W when mounted on a minimum pad.

- Pulse Test: Pulse Width $\leq 300\ \mu\text{s}$, Duty Cycle $\leq 2.0\%$
- ESD between the gate and source serves only, no gate overvoltage rating is implied.

TYPICAL CHARACTERISTICS

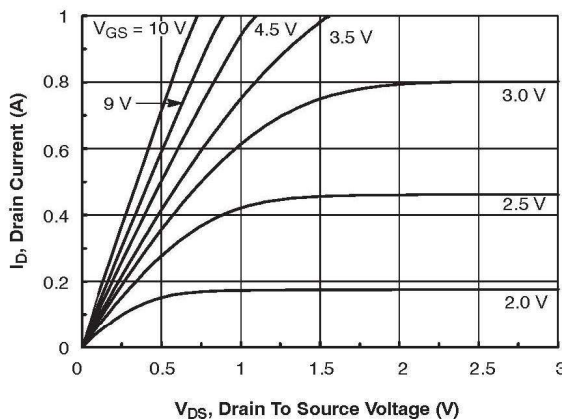


Figure 1. On-Region Characteristics

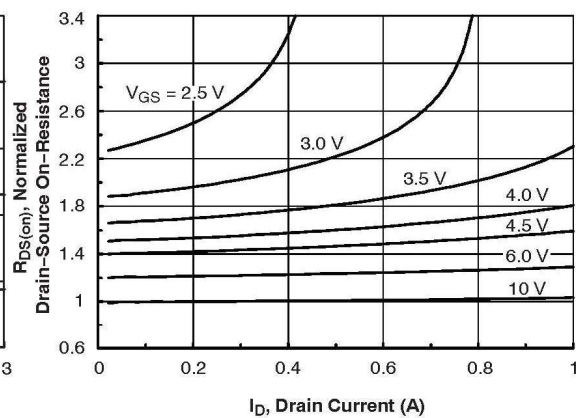


Figure 2. On-Resistance Variation with Drain Current and Gate Voltage



9.7.3 MCP6002 [5]



MCP6001/1R/1U/2/4

1 MHz, Low-Power Op Amp

Features

- Available in 5-Lead SC-70 and 5-Lead SOT-23 Packages
- Gain Bandwidth Product: 1 MHz (typical)
- Rail-to-Rail Input/Output
- Supply Voltage: 1.8V to 6.0V
- Supply Current: $I_Q = 100 \mu\text{A}$ (typical)
- Phase Margin: 90° (typical)
- Temperature Range:
 - Industrial: -40°C to $+85^\circ\text{C}$
 - Extended: -40°C to $+125^\circ\text{C}$
- Available in Single, Dual and Quad Packages

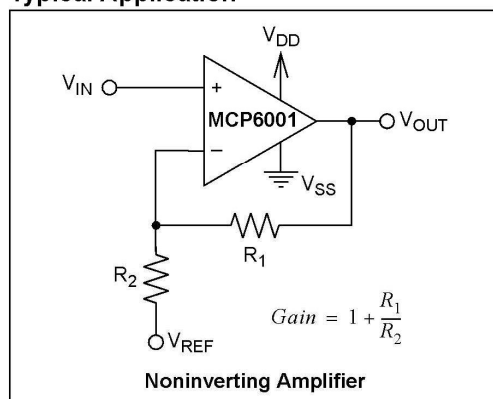
Applications

- Automotive
- Portable Equipment
- Photodiode Amplifier
- Analog Filters
- Notebooks and PDAs
- Battery-Powered Systems

Design Aids

- SPICE Macro Models
- FilterLab® Software
- Mindi™ Circuit Designer and Analog Simulator
- Microchip Advanced Part Selector (MAPS)
- Analog Demonstration and Evaluation Boards
- Application Notes

Typical Application

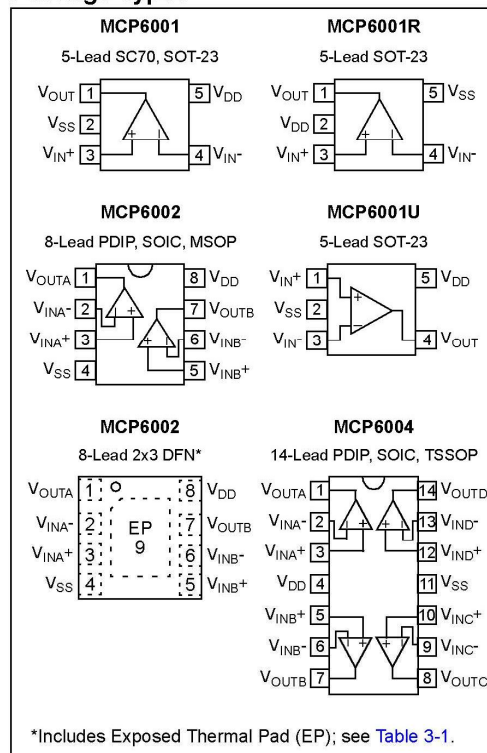


Description

The Microchip Technology Inc. MCP6001/2/4 family of operational amplifiers (op amps) is specifically designed for general purpose applications. This family has a 1 MHz Gain Bandwidth Product (GBWP) and 90° phase margin (typical). It also maintains a 45° phase margin (typical) with a 500 pF capacitive load. This family operates from a single-supply voltage as low as 1.8V, while drawing 100 μA (typical) quiescent current. Additionally, the MCP6001/2/4 supports rail-to-rail input and output swing, with a Common-mode input voltage range of $V_{DD} + 300 \text{ mV}$ to $V_{SS} - 300 \text{ mV}$. This family of op amps is designed with Microchip's advanced CMOS process.

The MCP6001/2/4 family is available in the industrial and extended temperature ranges, with a power supply range of 1.8V to 6.0V.

Package Types





MCP6001/1R/1U/2/4

1.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings†

$V_{DD} - V_{SS}$	7.0V
Current at Analog Input Pins (V_{IN+} , V_{IN-})	± 2 mA
Analog Inputs (V_{IN+} , V_{IN-})††	$V_{SS} - 1.0$ V to $V_{DD} + 1.0$ V
All Other Inputs and Outputs	$V_{SS} - 0.3$ V to $V_{DD} + 0.3$ V
Difference Input Voltage	$ V_{DD} - V_{SS} $
Output Short-Circuit Current	Continuous
Current at Output and Supply Pins	± 30 mA
Storage Temperature	-65°C to +150°C
Maximum Junction Temperature (T_J)	+150°C
ESD Protection On All Pins (HBM; MM)	≥ 4 kV; 200V

† Notice: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

†† See [Section 4.1.2 “Input Voltage and Current Limits”](#).

DC ELECTRICAL SPECIFICATIONS

Electrical Characteristics: Unless otherwise indicated, $T_A = +25^\circ\text{C}$, $V_{DD} = +1.8\text{V}$ to $+5.5\text{V}$, $V_{SS} = \text{GND}$, $V_{CM} = V_{DD}/2$, $V_L = V_{DD}/2$, $R_L = 10\text{ k}\Omega$ to V_L and $V_{OUT} \approx V_{DD}/2$ (refer to [Figure 1-1](#)).

Parameters	Sym	Min	Typ	Max	Units	Conditions
Input Offset						
Input Offset Voltage	V_{OS}	-4.5	—	+4.5	mV	$V_{CM} = V_{SS}$ (Note 1)
Input Offset Drift with Temperature	$\Delta V_{OS}/\Delta T_A$	—	± 2.0	—	$\mu\text{V}/^\circ\text{C}$	$T_A = -40^\circ\text{C}$ to $+125^\circ\text{C}$, $V_{CM} = V_{SS}$
Power Supply Rejection Ratio	PSRR	—	86	—	dB	$V_{CM} = V_{SS}$
Input Bias Current and Impedance						
Input Bias Current:	I_B	—	± 1.0	—	pA	$T_A = +85^\circ\text{C}$ $T_A = +125^\circ\text{C}$
Industrial Temperature	I_B	—	19	—	pA	
Extended Temperature	I_B	—	1100	—	pA	
Input Offset Current	I_{OS}	—	± 1.0	—	pA	
Common-Mode Input Impedance	Z_{CM}	—	$10^{13} \parallel 6$	—	$\Omega \parallel \text{pF}$	
Differential Input Impedance	Z_{DIFF}	—	$10^{13} \parallel 3$	—	$\Omega \parallel \text{pF}$	
Common-Mode						
Common-Mode Input Range	V_{CMR}	$V_{SS} - 0.3$	—	$V_{DD} + 0.3$	V	
Common-Mode Rejection Ratio	CMRR	60	76	—	dB	$V_{CM} = -0.3\text{V}$ to 5.3V , $V_{DD} = 5\text{V}$
Open-Loop Gain						
DC Open-Loop Gain (Large Signal)	A_{OL}	88	112	—	dB	$V_{OUT} = 0.3\text{V}$ to $V_{DD} - 0.3\text{V}$, $V_{CM} = V_{SS}$
Output						
Maximum Output Voltage Swing	V_{OL} , V_{OH}	$V_{SS} + 25$	—	$V_{DD} - 25$	mV	$V_{DD} = 5.5\text{V}$, 0.5V input overdrive
Output Short-Circuit Current	I_{SC}	—	± 6	—	mA	$V_{DD} = 1.8\text{V}$
		—	± 23	—	mA	$V_{DD} = 5.5\text{V}$
Power Supply						
Supply Voltage	V_{DD}	1.8	—	6.0	V	Note 2
Quiescent Current per Amplifier	I_Q	50	100	170	μA	$I_O = 0$, $V_{DD} = 5.5\text{V}$, $V_{CM} = 5\text{V}$

Note 1: MCP6001/1R/1U/2/4 parts with date codes prior to December 2004 (week code 49) were tested to ± 7 mV minimum/maximum limits.

2: All parts with date codes November 2007 and later have been screened to ensure operation at $V_{DD} = 6.0\text{V}$. However, the other minimum and maximum specifications are measured at 1.8V and 5.5V.



MCP6001/1R/1U/2/4

3.0 PIN DESCRIPTIONS

Descriptions of the pins are listed in [Table 3-1](#).

TABLE 3-1: PIN FUNCTION TABLE

MCP6001	MCP6001R	MCP6001U	MCP6002		MCP6004	Symbol	Description
5-Lead SC70, SOT-23	5-Lead SOT-23	5-Lead SOT-23	8-Lead MSOP, PDIP, SOIC	8-Lead 2x3 DFN	8-Lead PDIP, SOIC, TSSOP		
1	1	4	1	1	1	V_{OUT}, V_{OUTA}	Analog Output (Op Amp A)
4	4	3	2	2	2	V_{IN}^{-}, V_{INA}^{-}	Inverting Input (Op Amp A)
3	3	1	3	3	3	V_{IN}^{+}, V_{INA}^{+}	Noninverting Input (Op Amp A)
5	2	5	8	8	4	V_{DD}	Positive Power Supply
—	—	—	5	5	5	V_{INB}^{+}	Noninverting Input (Op Amp B)
—	—	—	6	6	6	V_{INB}^{-}	Inverting Input (Op Amp B)
—	—	—	7	7	7	V_{OUTB}	Analog Output (Op Amp B)
—	—	—	—	—	8	V_{OUTC}	Analog Output (Op Amp C)
—	—	—	—	—	9	V_{INC}^{-}	Inverting Input (Op Amp C)
—	—	—	—	—	10	V_{INC}^{+}	Noninverting Input (Op Amp C)
2	5	2	4	4	11	V_{SS}	Negative Power Supply
—	—	—	—	—	12	V_{IND}^{+}	Noninverting Input (Op Amp D)
—	—	—	—	—	13	V_{IND}^{-}	Inverting Input (Op Amp D)
—	—	—	—	—	14	V_{OUTD}	Analog Output (Op Amp D)
—	—	—	—	9	—	EP	Exposed Thermal Pad (EP); must be connected to V_{SS} .

3.1 Analog Outputs

The output pins are low-impedance voltage sources.

3.2 Analog Inputs

The noninverting and inverting inputs are high-impedance CMOS inputs with low bias currents.

3.3 Power Supply Pins

The positive power supply (V_{DD}) is 1.8V to 6.0V higher than the negative power supply (V_{SS}). For normal operation, the other pins are at voltages between V_{SS} and V_{DD} .

Typically, these parts are used in a single (positive) supply configuration. In this case, V_{SS} is connected to ground and V_{DD} is connected to the supply. V_{DD} will need bypass capacitors.

3.4 Exposed Thermal Pad (EP)

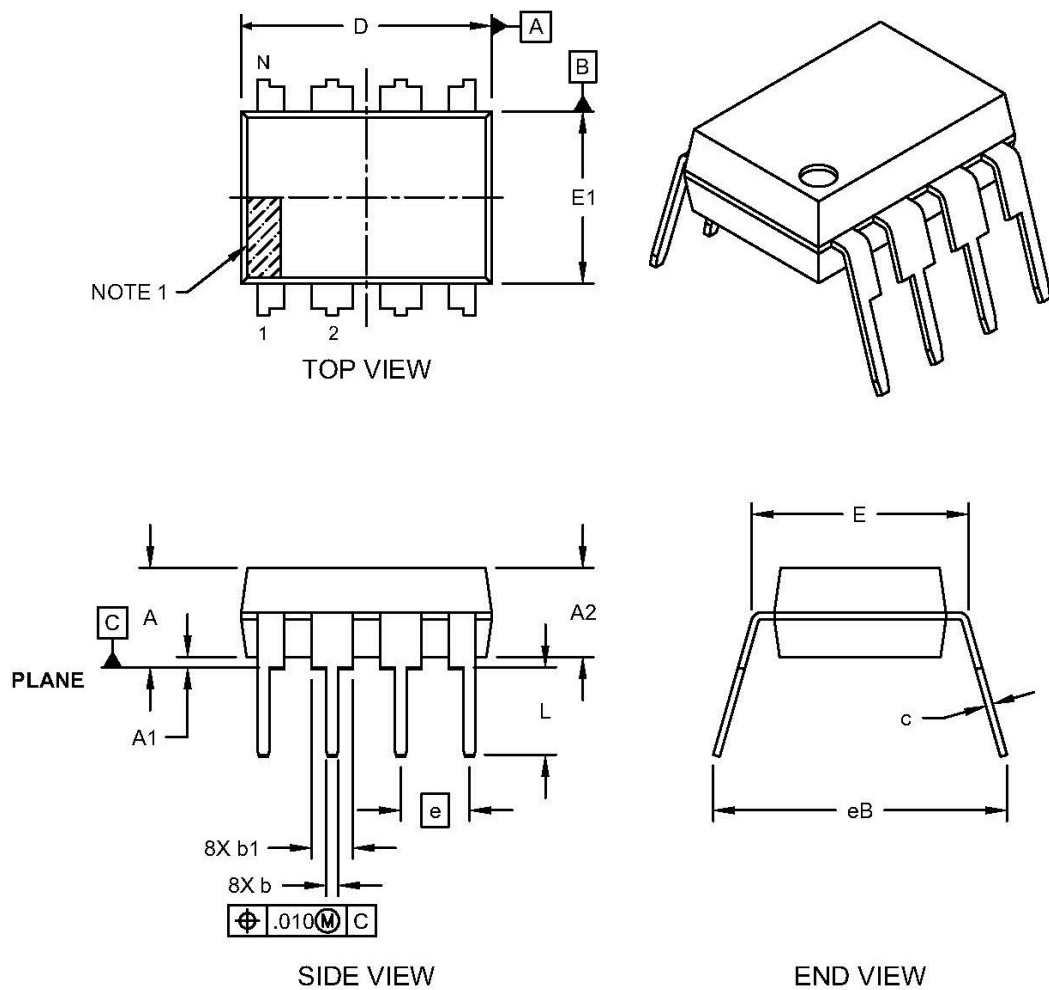
There is an internal electrical connection between the Exposed Thermal Pad (EP) and the V_{SS} pin; they must be connected to the same potential on the Printed Circuit Board (PCB).



MCP6001/1R/1U/2/4

8-Lead Plastic Dual In-Line (P) - 300 mil Body [PDIP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Microchip Technology Drawing No. C04-018-P Rev E Sheet 1 of 2

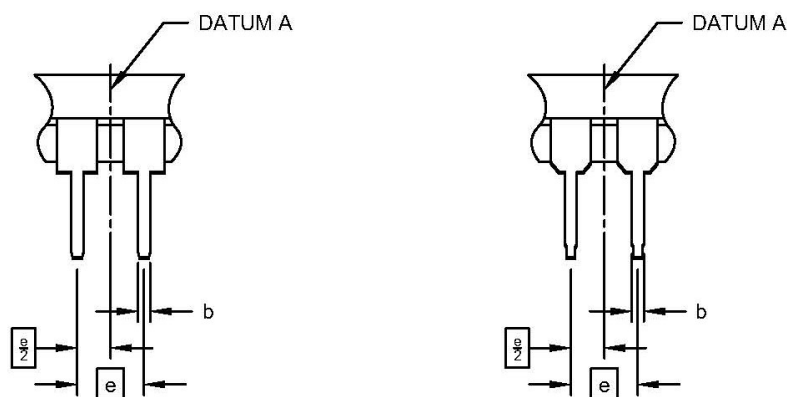


MCP6001/1R/1U/2/4

8-Lead Plastic Dual In-Line (P) - 300 mil Body [PDIP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>

ALTERNATE LEAD DESIGN
(NOTE 5)



Units		INCHES		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N	8		
Pitch	e	.100 BSC		
Top to Seating Plane	A	-	-	.210
Molded Package Thickness	A2	.115	.130	.195
Base to Seating Plane	A1	.015	-	-
Shoulder to Shoulder Width	E	.290	.310	.325
Molded Package Width	E1	.240	.250	.280
Overall Length	D	.348	.365	.400
Tip to Seating Plane	L	.115	.130	.150
Lead Thickness	c	.008	.010	.015
Upper Lead Width	b1	.040	.060	.070
Lower Lead Width	b	.014	.018	.022
Overall Row Spacing	§	eB	-	.430

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M
BSC: Basic Dimension. Theoretically exact value shown without tolerances.
5. Lead design above seating plane may vary, based on assembly vendor.

Microchip Technology Drawing No. C04-018-P Rev E Sheet 2 of 2



9.7.4 Akku-Lademodul [7]



南京拓微集成电路有限公司
NanJing Top Power ASIC Corp.

TP4056 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation in SOP-8

DESCRIPTION

The TP4056 is a complete constant-current/constant-voltage linear charger for single cell lithium-ion batteries. Its SOP package and low external component count make the TP4056 ideally suited for portable applications. Furthermore, the TP4056 can work within USB and wall adapter.

No blocking diode is required due to the internal PMOSFET architecture and have prevent to negative Charge Current Circuit. Thermal feedback regulates the charge current to limit the die temperature during high power operation or high ambient temperature. The charge voltage is fixed at 4.2V, and the charge current can be programmed externally with a single resistor. The TP4056 automatically terminates the charge cycle when the charge current drops to 1/10th the programmed value after the final float voltage is reached.

TP4056 Other features include current monitor, under voltage lockout, automatic recharge and two status pin to indicate charge termination and the presence of an input voltage.

FEATURES

- Programmable Charge Current Up to 1000mA
- No MOSFET, Sense Resistor or Blocking Diode Required
- Complete Linear Charger in SOP-8 Package for Single Cell Lithium-Ion Batteries
- Constant-Current/Constant-Voltage
- Charges Single Cell Li-Ion Batteries Directly from USB Port
- Preset 4.2V Charge Voltage with 1.5% Accuracy
- Automatic Recharge
- two Charge Status Output Pins
- C/10 Charge Termination
- 2.9V Trickle Charge Threshold (TP4056)
- Soft-Start Limits Inrush Current
- Available Radiator in 8-Lead SOP Package, the Radiator need connect GND or impending

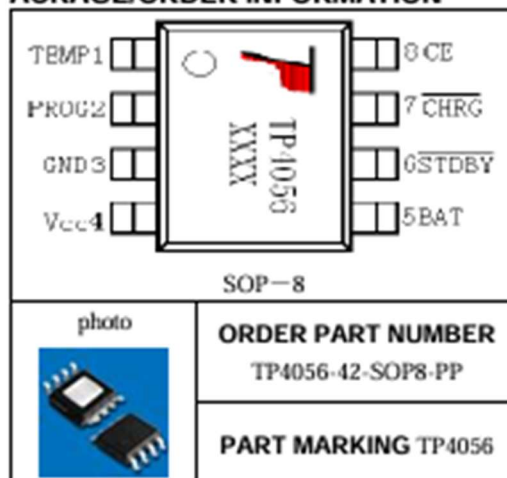
ABSOLUTE MAXIMUM RATINGS

- Input Supply Voltage (V_{CC}): -0.3V~8V
- TEMP: -0.3V~10V
- CE: -0.3V~10V
- BAT Short-Circuit Duration: Continuous
- BAT Pin Current: 1200mA
- PROG Pin Current: 1200uA
- Maximum Junction Temperature: 145°C
- Operating Ambient Temperature Range: -40°C~85°C
- Lead Temp.(Soldering, 10sec): 260°C

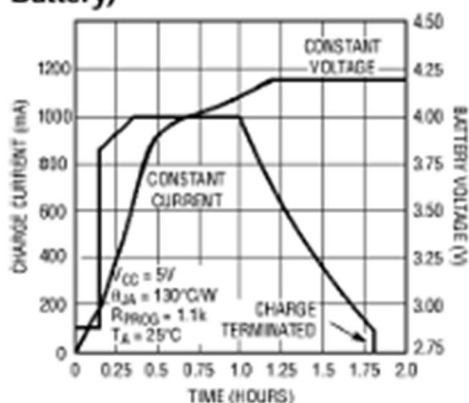
APPLICATIONS

- Cellular Telephones, PDAs, GPS
- Charging Docks and Cradles
- Digital Still Cameras, Portable Devices
- USB Bus-Powered Chargers, Chargers

PACKAGE/ORDER INFORMATION



Complete Charge Cycle (1000mAh Battery)





TEMP(Pin 1): Temperature Sense Input Connecting TEMP pin to NTC thermistor's output in Lithium ion battery pack. If TEMP pin's voltage is below 45% or above 80% of supply voltage V_{IN} for more than 0.15S, this means that battery's temperature is too high or too low, charging is suspended. The temperature sense function can be disabled by grounding the TEMP pin.

PROG(Pin 2): Constant Charge Current Setting and Charge Current Monitor Pin charge current is set by connecting a resistor R_{ISET} from this pin to GND. When in precharge mode, the ISET pin's voltage is regulated to 0.2V. When in constant charge current mode, the ISET pin's voltage is regulated to 2V. In all modes during charging, the voltage on ISET pin can be used to measure the charge current as follows:

$$I_{BAT} = \frac{V_{ISET}}{R_{ISET}} \times 1200 \quad (V_{ISET}=1V)$$

GND(Pin3): Ground Terminal

VCC(Pin 4): Positive Input Supply Voltage V_{IN} is the power supply to the internal circuit. When V_{IN} drops to within 30mV of the BAT pin voltage, TP4056 enters low power sleep mode, dropping BAT pin's current to less than 2uA.

BAT(Pin5): Battery Connection Pin. Connect the positive terminal of the battery to BAT pin. BAT pin draws less than 2uA current in chip disable mode or in sleep mode. BAT pin provides charge current to the battery and provides regulation voltage of 4.2V.

STDBY(Pin6): Open Drain Charge Status Output When the battery Charge Termination, the STDBY pin is pulled low by an internal switch, otherwise STDBY pin is in high impedance state.

CHRG(Pin7): Open Drain Charge Status Output When the battery is being charged, the CHRG pin is pulled low by an internal switch, otherwise CHRG pin is in high impedance state.

CE(Pin8): Chip Enable Input. A high input will put the device in the normal operating mode. Pulling the CE pin to low level will put the YP4056 into disable mode. The CE pin can be driven by TTL or CMOS logic level.

ELECTRICAL CHARACTERISTICS

The ● denotes specifications which apply over the full operating temperature range, otherwise specifications are at $T_A=25^{\circ}\text{C}$, $V_{CC}=5V$, unless otherwise noted.

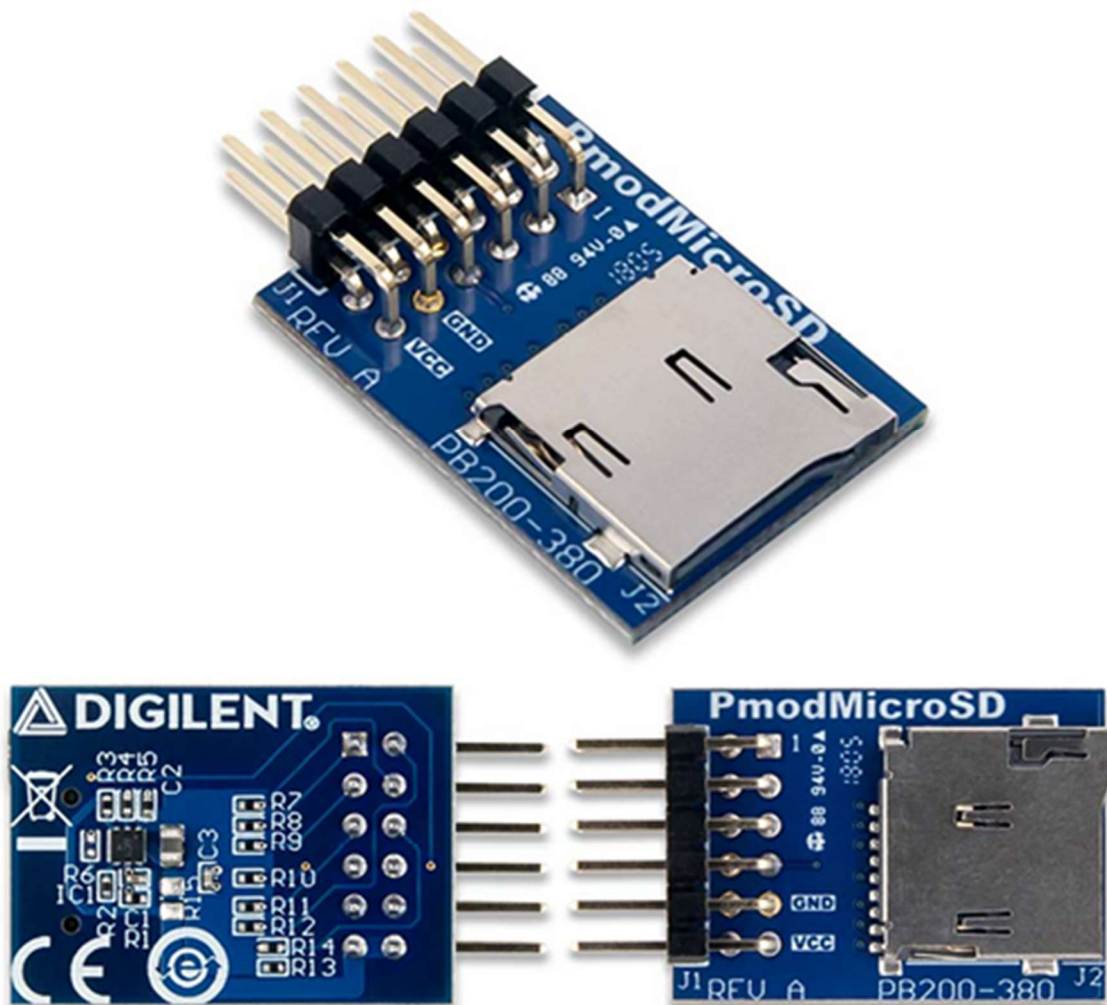
SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
V_{CC}	Input Supply Voltage		● 4.0	5	8.0	V
I_{CC}	Input Supply Current	Charge Mode, $R_{PROG} = 1.2k$ Standby Mode (Charge Terminated) Shutdown Mode (R_{PROG} Not Connected, $V_{CC} < V_{BAT}$, or $V_{CC} < V_{LVL}$)	● ● ●	150 55 55	500 100 100	μA μA μA
V_{FLOAL}	Regulated Output (Float) Voltage	$0^{\circ}\text{C} \leq T_A \leq 85^{\circ}\text{C}$, $I_{BAT}=40mA$		4.137	4.2	4.263 V
I_{BAT}	BAT Pin Current Text condition: $V_{BAT}=4.0V$	$R_{PROG} = 2.4k$, Current Mode $R_{PROG} = 1.2k$, Current Mode Standby Mode, $V_{BAT} = 4.2V$	● ● ●	450 950 0	500 1000 -2.5	550 1050 -6 mA μA
I_{TRKL}	Trickle Charge Current	$V_{BAT} < V_{TRKL}$, $R_{PROG}=1.2K$	●	120	130	140 mA
V_{TRKL}	Trickle Charge Threshold Voltage	$R_{PROG}=1.2K$, V_{BAT} Rising		2.8	2.9	3.0 V
V_{TRHYS}	Trickle Charge Hysteresis Voltage	$R_{PROG}=1.2K$		60	80	100 mV
T_{LIM}	Junction Temperature in Constant Temperature Mode			145		$^{\circ}\text{C}$

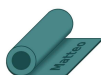
9.7.5 SD-Karten-Modul [8]



Pmod MicroSD Reference Manual

The Digilent Pmod MicroSD (Revision A) allows system boards to read from and write to microSD cards. With no limitation on the file system or memory size of the microSD card, users will be able to store and access large amounts of data from their system board.





Features

- MicroSD card slot
 - Store and access large amounts of data from your system board
 - No limitation on file system or memory size of microSD card used
 - 1-bit and 4-bit communication
 - 12-pin Pmod port with SPI interface
-

Functional Description

The Pmod MicroSD is a great way to store and access information on a device that can then be accessed by outside devices such as a phone or computer.

Interfacing with the Pmod

The Pmod MicroSD is designed to communicate with the host board primarily through the SPI protocol. By default, the microSD card itself is defined to boot up in SD mode, but will enter into SPI mode when the Chip Select line is pulled low. The actual technique of writing to and reading from specific locations on an microSD goes beyond the scope of this document, but the complexity of the process is nicely abstracted away within the chipKIT SD library and Digilent PmodSD IP Core.

A pinout description table for the 2×6 pin header on the Pmod MicroSD can be found below:

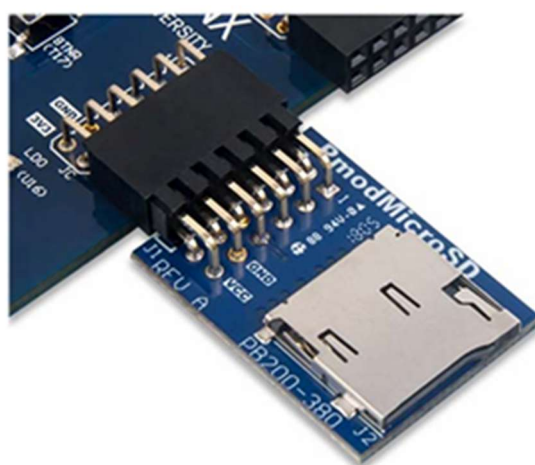
Connector J1- Pin Descriptions						
Pin	Signal	Description		Pin	Signal	Description
1	~CS	Chip Select / Data3		7	DAT1	Data 1
2	MOSI	MOSI / Command		8	DAT2	Data 2

3	MISO	MISO / Data0	9	CD	Card Detect
4	SCK	Serial Clock	10	NC	<i>Not Connected</i>
5	GND	Power Supply Ground	11	GND	Power Supply Ground
6	VCC	Power Supply (3.3V)	12	VCC	Power Supply (3.3V)

Any external power applied to the Pmod MicroSD must be within the specifications of the inserted microSD card. As determined by the SD card association, this voltage range must be within 2.7 to 3.6 volts. The 3.3 operating voltage on Digilent system boards nicely complies with this standard.

Physical Dimensions

The pins on the pin header are spaced 100 mil apart. The PCB is 1.8 inches long on the sides parallel to the pins on the pin header and 1.8 inches long on the sides perpendicular to the pin header.

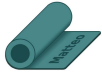


<https://reference.digilentinc.com/reference/pmod/pmodmicrosd/reference-manual> 6-19-18

9.7.6 Andere Datenblätter

Alle anderen verwendeten Datenblätter haben einen zu großen Umfang, um sie hier einzufügen. Links zu diesen sind im Quellenverzeichnis zu finden:

- PIC16F1827 [9]
- ESP32 [10]
- Arduino Nicla Voice [11]
- Audio-Verstärker [12]



9.8 Programmcode

HTML-Markup Home:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>Matteo</title>

    <link          rel="icon"          type="image/x-icon"
href="https://htl3rmatteo.at/images/Matteo_logo.png">

    <link rel="stylesheet" href="https://htl3rmatteo.at/Matteo.css">

    <link          rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">

</head>

<body class="Home">

    <nav class="navbar">

        <div class="navbar_container">

            <a href="https://htl3rmatteo.at/Home.html"
class="navbar_logo">

                Matteo

            </a>

            <div class="navbar_toggle">

                <span class="bar"></span>

                <span class="bar"></span>

                <span class="bar"></span>

                <span class="bar"></span>

                <span class="bar"></span>

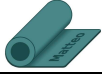
            </div>

        </div>

    </nav>

</body>

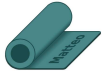
</html>
```

```
<ul class="navbar_menu">
    <li class="navbar_links">
        <a href="https://htl3rmatteo.at/Home.html"
class="navbar_link">Home</a>
    </li>
    <li class="navbar_links">
        <a href="https://htl3rmatteo.at/Team.html"
class="navbar_link">Team</a>
    </li>
    <li class="navbar_links">
        <a href="https://htl3rmatteo.at/Idea.html"
class="navbar_link">Idee</a>
    </li>
    <li class="navbar_links">
        <a
href="https://htl3rmatteo.at/Projektfortschritt.html"
class="navbar_link">Meilensteine</a>
    </li>
    <li class="navbar_btn">
        <a href="https://www.instagram.com/matteo_htl3r"
target="_blank" class="button">
            <i class="fa fa-instagram" style="font-size:
24px;"></i>
        </a>
    </li>
</ul>
</div>
</nav>

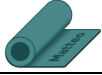
<!--Content-->

<div class="content_container">
    <div class="text_info_container">
        <div class="logo_container">
```

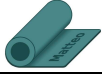
```

    <div class="content_logo">Matteo</div>
</div>
<div class="Slogan_nav_container">
    <div class="Slogan">Sicher stehen, Sichergehen.</div>
    <ul class="nav_cards">
        <li class="nav_card">
            <a href="https://htl3rmatteo.at/Team.html"
class="card">
                <p class="card_text">Team</p>
                
            </a>
        </li>
        <li class="nav_card">
            <a href="https://htl3rmatteo.at/Idea.html"
class="card">
                <p class="card_text">Idee</p>
                
            </a>
        </li>
        <li class="nav_card">
            <a
href="https://htl3rmatteo.at/Projektfortschritt.html" class="card">
                <p class="card_text">Meilen<br>steine</p>
                
            </a>
        </li>
    </ul>
</div>
```



```
        </ul>
    </div>
</div>
<div class="img_container">
    <div class="img_inner_container">
        
    </div>
</div>
</div>

<div class="Sponsoren_container">
    <div class="Sponsor">
        <a href="https://zsu-schmelz.univie.ac.at" target="_blank"
class="Sponsor_Link">
            
            <p class="Sponsor_name">Sportuniversität Wien</p>
        </a>
    </div>
    <div class="Sponsor">
        <a href="https://www.cadaxo.com" target="_blank"
class="Sponsor_Link">
            
            <p class="Sponsor_name">Cadaxo</p>
        </a>
    </div>
    <div class="Sponsor">
        <a href="https://jlcpcb.com" target="_blank"
class="Sponsor_Link">
            
            <p class="Sponsor_name">JLCPCB</p>
        </a>
    </div>
</div>
```



```
        </a>
    </div>
    <div class="Sponsor">
        <p>
            <b>Jetzt Sponsor werden:</b> <br class="desktop_only">
            <b class="desktop_only" >Email:</b> <a href="mailto:
1088@htl.rennweg.at"
            target="_blank"
class="Member_Mail">laurenz.wimmer@htl.rennweg.at</a>
        </p>
    </div>
</div>

<!-- footer -->

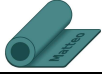
    <div class="footer_container">
        <a href="https://htl3rmatteo.at/Home.html"
class="footer_logo_container">
            
            <div class="footer_Name">Matteo</div>
        </a>
        <ul class="footer_menu">
            <li class="footer_links">
                <a href="https://htl3rmatteo.at/Impressum.html"
class="footer_link">Impressum</a>
            </li>
            <li class="footer_links">
                <a href="https://htl3rmatteo.at/Datenschutz.html"
class="footer_link">Datenschutz</a>
            </li>
        </ul>
        <p class="copyright">© 2025 Matteo</p>
    </div>
```



```
<script src="https://htl3rmatteo.at/Home.js"></script>
</body>
</html>
```

HTML-Markup Team:

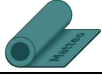
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Team | Matteo</title>
  <link rel="icon" type="image/x-icon"
href="https://htl3rmatteo.at/images/Matteo_logo.png">
  <link rel="stylesheet" href="https://htl3rmatteo.at/Matteo.css">
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
</head>
<body class="T">
  <nav class="navbar">
    <div class="navbar_container">
      <a href="https://htl3rmatteo.at/Home.html"
class="navbar_logo">
        
      <div class="navbar_toggle">
        <span class="bar"></span>
        <span class="bar"></span>
        <span class="bar"></span>
        <span class="bar"></span>
      </div>
    </div>
  </nav>
</body>
</html>
```



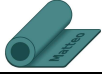
```
<span class="bar"></span>
</div>
<ul class="navbar_menu">
  <li class="navbar_links">
    <a href="https://htl3rmatteo.at/Home.html"
class="navbar_link">Home</a>
  </li>
  <li class="navbar_links">
    <a href="https://htl3rmatteo.at/Team.html"
class="navbar_link">Team</a>
  </li>
  <li class="navbar_links">
    <a href="https://htl3rmatteo.at/Idea.html"
class="navbar_link">Idee</a>
  </li>
  <li class="navbar_links">
    <a
href="https://htl3rmatteo.at/Projektfortschritt.html"
class="navbar_link">Meilensteine</a>
  </li>
  <li class="navbar_btn">
    <a href="https://www.instagram.com/matteo_htl3r"
target="_blank" class="button">
      <i class="fa fa-instagram" style="font-size:
24px;"></i>
    </a>
  </li>
</ul>
</div>
</nav>

<!--Content-->

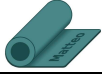
<div class="Team_content_container">
```



```
<div class="Team_container">
  <div class="Member_container">
    <div class="Member_img_container">
      
    </div>
    <div class="Member_info_container">
      <p class="Member_Name">Laurenz Wimmer</p>
      <p class="Member_Role">Projektleiter</p>
      <p class="Member_Assignment">Elektrotechnik und
Projektmanagement</p>
      <a href="mailto:1088@htl.rennweg.at"
class="Member_Mail" target="_blank">1088@htl.rennweg.at</a>
    </div>
  </div>
  <div class="Team_text_container">
    <p class="Team_text">Team</p>
    <a href="https://www.instagram.com/matteo_htl3r"
target="_blank" class="btn">
      <i class="fa fa-instagram" style="font-size:
48px;"></i>
    </a>
  </div>
  <div class="Member_container">
    <div class="Member_img_container">
      
    </div>
    <div class="Member_info_container">
      <p class="Member_Name">Nandor Bagari</p>
      <p class="Member_Role">Stellv. Projektleiter</p>
      <p class="Member_Assignment">Elektrotechnik und
Konstruktion</p>
```



```
<a href="mailto:1062@htl.rennweg.at"
class="Member_Mail" target="_blank">1062@htl.rennweg.at</a>
</div>
</div>
<div class="Member_container">
  <div class="Member_img_container">
    
  </div>
  <div class="Member_info_container">
    <p class="Member_Name">Mika Böhm</p>
    <p class="Member_Role">Projektmitarbeiterin</p>
    <p class="Member_Assignment">Software und
Website</p>
    <a href="mailto:1064@htl.rennweg.at"
class="Member_Mail" target="_blank">1064@htl.rennweg.at</a>
  </div>
</div>
<div class="Member_container">
  <div class="Member_img_container">
    
  </div>
  <div class="Member_info_container">
    <p class="Member_Name">Lisa Pripfl</p>
    <p class="Member_Role">Projektmitarbeiterin</p>
    <p class="Member_Assignment">Software</p>
    <a href="mailto:1081@htl.rennweg.at"
class="Member_Mail" target="_blank">1081@htl.rennweg.at</a>
  </div>
</div>
</div>
</div>
```

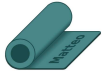
```
<!--footer-->

<div class="footer_container">
    <a href="https://htl3rmatteo.at/Home.html"
class="footer_logo_container">
        
        <div class="footer_Name">Matteo</div>
    </a>
    <ul class="footer_menu">
        <li class="footer_links">
            <a href="https://htl3rmatteo.at/Impressum.html"
class="footer_link">Impressum</a>
        </li>
        <li class="footer_links">
            <a href="https://htl3rmatteo.at/Datenschutz.html"
class="footer_link">Datenschutz</a>
        </li>
    </ul>
    <p class="copyright">© 2025 Matteo</p>
</div>

<script src="https://htl3rmatteo.at/Home.js"></script>
</body>
</html>
```

HTML-Markup Idee:

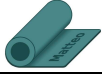
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Idee | Matteo</title>
```



```
<link          rel="icon"          type="image/x-icon"
href="https://htl3rmatteo.at/images/Matteo_logo.png">

    <link rel="stylesheet" href="https://htl3rmatteo.at/Matteo.css">

                                <link          rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
</head>
<body class="Idea">
    <nav class="navbar">
        <div class="navbar_container">
            <a href="https://htl3rmatteo.at/Home.html"
class="navbar_logo">
                                
                                Matteo
            </a>
            <div class="navbar_toggle">
                <span class="bar"></span>
                <span class="bar"></span>
                <span class="bar"></span>
                <span class="bar"></span>
                <span class="bar"></span>
            </div>
            <ul class="navbar_menu">
                <li class="navbar_links">
                    <a href="https://htl3rmatteo.at/Home.html"
class="navbar_link">Home</a>
                </li>
                <li class="navbar_links">
                    <a href="https://htl3rmatteo.at/Team.html"
class="navbar_link">Team</a>
                </li>
                <li class="navbar_links">
```



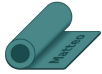
```

                <a href="https://htl3rmatteo.at/Idea.html"
class="navbar_link">Idee</a>
            </li>
            <li class="navbar_links">
                <a
href="https://htl3rmatteo.at/Projektfortschritt.html"
class="navbar_link">Meilensteine</a>
            </li>
            <li class="navbar_btn">
                <a href="https://www.instagram.com/matteo_htl3r"
target="_blank" class="button">
                    <i class="fa fa-instagram" style="font-size:
24px;"></i>
                </a>
            </li>
        </ul>
    </div>
</nav>

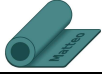
<!-- Content -->

<div class="content_container">
    <div class="text_info_container">
        <div class="Idea_header_container">
            <div class="Idea_header">Idee</div>
        </div>
        <div class="Slogan_nav_container">
            <h1>Konzept</h1>
            <div class="abschnitt">
                <p class="head">
                    Ziele des Projektes
                </p>
                <ul class="SL_Files">

```



```
<li class="SL_File">
    Älteren Menschen mit Sehbeeinträchtigung
    Balanceübungen zu ermöglichen
</li>
<li class="SL_File">
    Matte, die über Sensoren die Belastung
    misst
</li>
<li class="SL_File">
    Auditives Feedback
</li>
<li class="SL_File">
    Sprachsteuerung für diverse Einstellungen
</li>
<li class="SL_File">
    Über integrierte Akkus betreiben
</li>
</ul>
</div>
<div class="abschnitt">
    <p class="head">
        Kooperationen
    </p>
    <ul class="SL_Files">
        <li class="SL_File">
            Sportuniversität Wien
        </li>
        <li class="SL_File">
            Blinden- und Sehbehindertenverband Wien,
            Niederösterreich und Burgenland
        </li>
        <li class="SL_File">
            Versehrtensportklub ASVÖ-Wien
        </li>
    </ul>
</div>
```



<li class="SL_File">

Hilfsgemeinschaft der Blinden und
Sehbehinderten Österreichs

</div>

<p class="Absatz">

Ziel des Projektes Matteo ist es, älteren Menschen mit Sehbeeinträchtigung oder Blindheit dabei zu helfen, Balanceübungen durchzuführen.

Im Alter nimmt die Bewegungsfreiheit grundsätzlich ab und Sehbeeinträchtigungen stellen ein zusätzliches Hindernis dar.

Deshalb ist das Ziel, Betroffenen die Möglichkeit zur eigenständigen Betätigung zu geben.

</p>

<p class="Absatz">

Dazu wird eine Matte entwickelt, die über Sensoren die Belastung der einzelnen Gliedmaßen misst.

Je nach Belastung gibt die Matte dementsprechendes Feedback aus.

Dies soll auditiv erfolgen, wodurch keine Einschränkungen durch Sehschwächen entstehen.

</p>

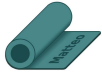
<p class="Absatz">

Die Matte wird ungefähr 100x70 cm haben, wodurch genug Platz zur Ausübung diverser Übungen geboten ist.

Gesteuert wird die Matte über eine Sprachsteuerung.

So können diverse Einstellungen vorgenommen werden.

Unter anderem wird es möglich sein, sich ein Belastungsziel für die einzelnen Gliedmaßen zu setzen.



Beispielsweise 70% des Körpergewichts auf dem rechten und 30% auf dem linken Bein.

</p>

<p class="Absatz">

Die Matte wird über integrierte Akkus versorgt.

So besteht keine Stolpergefahr durch am Boden liegende Kabel.

Das Material der Matte wird so gewählt, dass die verbauten Sensoren nicht oder nur sehr schwach spürbar sind.

So wird das Durchführen der Übungen so angenehm wie möglich gestaltet.

</p>

<p class="Absatz">

Das Projekt wird in Zusammenarbeit mit der Sportuniversität Wien und in enger Kooperation mit dem Blinden-

und Sehbehindertenverband Wien,

Niederösterreich und Burgenland,

dem Versehrten Sportklub ASVÖ-Wien und der Hilfsgemeinschaft der Blinden und Sehbehinderten Österreichs durchgeführt.

Dadurch soll möglichst praxisnahes Feedback zu diversen Ideen eingeholt werden können,

um eine möglichst hohe Benutzerfreundlichkeit zu gewährleisten.

</p>

</div>

</div>

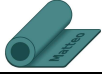
<div class="img_container">

<div class="img_inner_container">

</div>

</div>

</div>



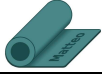
```
<!-- Footer -->

<div class="footer_container">
    <a href="https://htl3rmatteo.at/Home.html"
class="footer_logo_container">
        
        <div class="footer_Name">Matteo</div>
    </a>
    <ul class="footer_menu">
        <li class="footer_links">
            <a href="https://htl3rmatteo.at/Impressum.html"
class="footer_link">Impressum</a>
        </li>
        <li class="footer_links">
            <a href="https://htl3rmatteo.at/Datenschutz.html"
class="footer_link">Datenschutz</a>
        </li>
    </ul>
    <p class="copyright">© 2025 Matteo</p>
</div>

<script src="https://htl3rmatteo.at/Home.js"></script>
</body>
</html>
```

HTML-Markup Meilensteine:

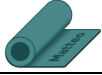
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Meilensteine | Matteo</title>
```

```
<link rel="icon" type="image/x-icon"
href="https://htl3rmatteo.at/images/Matteo_logo.png">

<link rel="stylesheet" href="https://htl3rmatteo.at/Matteo.css">

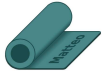
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
</head>
<body class="PF">
    <nav class="navbar">
        <div class="navbar_container">
            <a href="https://htl3rmatteo.at/Home.html"
class="navbar_logo">
                
                Matteo
            </a>
            <div class="navbar_toggle">
                <span class="bar"></span>
                <span class="bar"></span>
                <span class="bar"></span>
                <span class="bar"></span>
                <span class="bar"></span>
            </div>
            <ul class="navbar_menu">
                <li class="navbar_links">
                    <a href="https://htl3rmatteo.at/Home.html"
class="navbar_link">Home</a>
                </li>
                <li class="navbar_links">
                    <a href="https://htl3rmatteo.at/Team.html"
class="navbar_link">Team</a>
                </li>
                <li class="navbar_links">
```



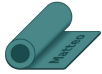
```
<a href="https://htl3rmatteo.at/Idea.html"
class="navbar_link">Idee</a>
</li>
<li class="navbar_links">
<a
href="https://htl3rmatteo.at/Projektfortschritt.html"
class="navbar_link">Meilensteine</a>
</li>
<li class="navbar_btn">
<a href="https://www.instagram.com/matteo_htl3r"
target="_blank" class="button">
<i class="fa fa-instagram" style="font-size:
24px;"></i>
</a>
</li>
</ul>
</div>
</nav>

<!-- Content -->

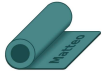
<div class="Milestone_List_container">
<ul class="Milestone_List">
<li class="Milestone_container">
<div class="Milestone_Text_container">
<div class="Milestone_Header">Website
Lauchen</div>
<div class="Milestone_Info">Die Website ist ein
wichtiger Teil der Diplomarbeit.
Sie dient dem Marketing und macht die
Projektidee,
den Projektfortschritt in Form der Meilensteine
und Kontaktinformationen der Teammitglieder öffentlich zugänglich.
Die Website wird trotz Erfüllung des Milestones
laufend verbessert.
```



```
</div>
</div>
<div class="Milestone_Status done"></div>
</li>
<li class="dot"></li>
<li class="dot"></li>
<li class="dot"></li>
<li class="Milestone_container">
  <div class="Milestone_Text_container">
    <div class="Milestone_Header">Planung</div>
    <div class="Milestone_Info">Die Planung beinhaltet
die Stakeholderanalyse, den Objektstrukturplan,
den Projektstrukturplan, die Risikoanalyse,
die Meilensteine und die Projektressourcen sowie die
Kooperationspartner.</div>
  </div>
  <div class="Milestone_Status done"></div>
</li>
<li class="dot"></li>
<li class="dot"></li>
<li class="dot"></li>
<li class="Milestone_container">
  <div class="Milestone_Text_container">
    <div class="Milestone_Header">Diplomarbeitsantrag</div>
    <div class="Milestone_Info">Der Antrag beinhaltet
die Haupt-, Neben- und Nichtziele des Projekts.
Ebenso sind die Planung und alle dazugehörigen
Analysen enthalten.
Der Antrag wird von allen Mitarbeitern sowie
den Betreuern unterschrieben und anschließend im ABA-Portal
eingereicht.
  </div>
</li>
</div>
```



```
        <div class="Milestone_Status done"></div>
    </li>
    <li class="dot"></li>
    <li class="dot"></li>
    <li class="dot"></li>
    <li class="Milestone_container">
        <div class="Milestone_Text_container">
            <div class="Milestone_Header">Technische
Planung</div>
            <div class="Milestone_Info">Die technische Planung
beinhaltet Schaltpläne, erste 3D-Modelle, Auswahl der Bauteile und
Flussdiagramme der Programme.</div>
        </div>
        <div class="Milestone_Status done"></div>
    </li>
    <li class="dot"></li>
    <li class="dot"></li>
    <li class="dot"></li>
    <li class="Milestone_container">
        <div class="Milestone_Text_container">
            <div class="Milestone_Header">Sprachausgabe</div>
            <div class="Milestone_Info">Die Sprachausgabe gibt
Informationen wie die aktuelle Belastung auditiv aus.</div>
        </div>
        <div class="Milestone_Status done"></div>
    </li>
    <li class="dot"></li>
    <li class="dot"></li>
    <li class="dot"></li>
    <li class="Milestone_container">
        <div class="Milestone_Text_container">
            <div class="Milestone_Header">Erster Prototyp</div>
```



```
<div class="Milestone_Info">Der erste Prototyp hat  
grundlegende Funktionen wie das Messen der Belastung.
```

```
Die Daten werden über einen externen Computer  
ausgegeben. Die Sprachein- und ausgabe werden in diesem Prototyp nicht  
enthalten sein.</div>
```

```
</div>
```

```
<div class="Milestone_Status done"></div>
```

```
</li>
```

```
<li class="dot"></li>
```

```
<li class="dot"></li>
```

```
<li class="dot"></li>
```

```
<li class="Milestone_container">
```

```
<div class="Milestone_Text_container">
```

```
<div
```

```
class="Milestone_Header">Zwischenpräsentation</div>
```

```
<div class="Milestone_Info">Am 25.11.2025 wird vor  
den Betreuern des Projektes eine Präsentation gehalten, in der der  
momentane Stand präsentiert wird.</div>
```

```
</div>
```

```
<div class="Milestone_Status done"></div>
```

```
</li>
```

```
<li class="dot"></li>
```

```
<li class="dot"></li>
```

```
<li class="dot"></li>
```

```
<li class="Milestone_container">
```

```
<div class="Milestone_Text_container">
```

```
<div class="Milestone_Header">Umsetzung</div>
```

```
<div class="Milestone_Info">Wenn die Umsetzung  
abgeschlossen ist, kann mit dem Testen der Matte begonnen werden.</div>
```

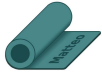
```
</div>
```

```
<div class="Milestone_Status done"></div>
```

```
</li>
```

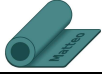
```
<li class="dot"></li>
```

```
<li class="dot"></li>
```



```
<li class="dot"></li>
<li class="Milestone_container">
  <div class="Milestone_Text_container">
    <div class="Milestone_Header">Testen</div>
    <div class="Milestone_Info">Es werden alle
Komponente getestet und eventuell überarbeitet, bis alle Ziele erfüllt
sind und die Matte einwandfrei funktioniert.</div>
  </div>
  <div class="Milestone_Status in_Progress"></div>
</li>
<li class="dot"></li>
<li class="dot"></li>
<li class="dot"></li>
<li class="Milestone_container">
  <div class="Milestone_Text_container">
    <div class="Milestone_Header">Abnahme und
Endpräsentation</div>
    <div class="Milestone_Info">Das Projekt wird von
den Betreuern abgenommen und anschließend der Schule
präsentiert.</div>
  </div>
  <div class="Milestone_Status"></div>
</li>
</ul>
</div>

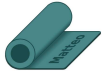
<!-- Footer -->
<div class="footer_container">
  <a href="https://htl3rmatteo.at/Home.html"
class="footer_logo_container">
    
    <div class="footer_Name">Matteo</div>
  </a>
```



```
<ul class="footer_menu">
  <li class="footer_links">
    <a href="https://htl3rmatteo.at/Impressum.html"
class="footer_link">Impressum</a>
  </li>
  <li class="footer_links">
    <a href="https://htl3rmatteo.at/Datenschutz.html"
class="footer_link">Datenschutz</a>
  </li>
</ul>
<p class="copyright">© 2025 Matteo</p>
</div>
<script src="https://htl3rmatteo.at/Home.js"></script>
</body>
</html>
```

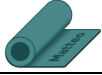
HTML-Markup Impressum:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Impressum | Matteo</title>
  <link rel="icon" type="image/x-icon"
href="https://htl3rmatteo.at/images/Matteo_logo.png">
  <link rel="stylesheet" href="https://htl3rmatteo.at/Matteo.css">
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
</head>
<body class="Impressum">
  <nav class="navbar">
    <div class="navbar_container">
```

```
<a href="https://htl3rmatteo.at/Home.html"
class="navbar_logo">

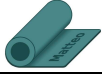
alt=""
Matteo
</a>
<div class="navbar_toggle">
  <span class="bar"></span>
  <span class="bar"></span>
  <span class="bar"></span>
  <span class="bar"></span>
  <span class="bar"></span>
</div>
<ul class="navbar_menu">
  <li class="navbar_links">
    <a href="https://htl3rmatteo.at/Home.html"
class="navbar_link">Home</a>
  </li>
  <li class="navbar_links">
    <a href="https://htl3rmatteo.at/Team.html"
class="navbar_link">Team</a>
  </li>
  <li class="navbar_links">
    <a href="https://htl3rmatteo.at/Idea.html"
class="navbar_link">Idee</a>
  </li>
  <li class="navbar_links">
    <a
href="https://htl3rmatteo.at/Projektfortschritt.html"
class="navbar_link">Meilensteine</a>
  </li>
  <li class="navbar_btn">
```



```
<a href="https://www.instagram.com/matteo_htl3r"
target="_blank" class="button">
    <i class="fa fa-instagram" style="font-size:
24px;"></i>
</a>
</li>
</ul>
</div>
</nav>

<!-- Content -->

<div class="content_container">
    <div class="text_container">
        <div class="abschnitt">
            <p class="head">
                Impressum
            </p>
            <p class="Paragraph">
                <br> <b>Matteo</b> <br> <br>
                Angaben gemäß § 5 E-Commerce-Gesetz (ECG), § 25
                Mediengesetz und § 14 Unternehmensgesetzbuch (UGB): <br> <br>
                <b>Name:</b> Mika Böhm <br> <br>
                <b>Adresse:</b> <br>
                Rennweg 89b <br>
                1030 Wien <br>
                Austria <br> <br>
                <b>E-Mail:</b> <a href="mailto:1064@htl.rennweg.at"
class="Member_Mail" target="_blank">1064@htl.rennweg.at</a> <br> <br>
                <b>Tel.:</b> <a href="tel: +43 677 62403662"
class="Member_Mail" target="_blank">+43 1 242 15-10</a>
            </p>
        </div>
    </div>
</div>
```



```
<div class="abschnitt">
```

```
<p class="head">
```

Zweck der Webseite

```
</p>
```

```
<p class="Paragraph">
```

Das Projekt Matteo ist eine Diplomarbeit an der HTL Rennweg im Bereich Mechatronik.

Die Website dient ausschließlich der Projektpräsentation und möglichen Sponsorings im Rahmen des Schulprojekts.

```
</p>
```

```
</div>
```

```
<div class="abschnitt">
```

```
<p class="head">
```

Haftungsausschluss

```
</p>
```

```
<p class="Paragraph">
```

Alle Inhalte dieser Website wurden mit größtmöglicher Sorgfalt erstellt. Für die Richtigkeit, Vollständigkeit und Aktualität der Inhalte kann jedoch keine Gewähr übernommen werden.

Diese Website enthält Links zu externen Webseiten Dritter, auf deren Inhalte kein Einfluss besteht. Deshalb kann für diese fremden Inhalte auch keine Gewähr übernommen werden.

Für die Inhalte der verlinkten Seiten ist stets der jeweilige Anbieter oder Betreiber verantwortlich.

```
</p>
```

```
</div>
```

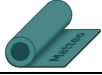
```
<div class="abschnitt">
```

```
<p class="head">
```

Urheberrecht

```
</p>
```

```
<p class="Paragraph">
```

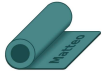


Die auf dieser Website erstellten Inhalte und Werke unterliegen dem österreichischen Urheberrecht. Beiträge Dritter sind als solche gekennzeichnet.

Die Vervielfältigung, Bearbeitung und Verbreitung außerhalb der Grenzen des Urheberrechts bedürfen der schriftlichen Zustimmung des jeweiligen Autors bzw. Erstellers.

```
        </p>
    </div>
</div>
</div>

<!-- Footer -->
    <div class="footer_container">
        <a href="https://htl3rmatteo.at/Home.html"
class="footer_logo_container">
            
            <div class="footer_Name">Matteo</div>
        </a>
        <ul class="footer_menu">
            <li class="footer_links">
                <a href="https://htl3rmatteo.at/Impressum.html"
class="footer_link">Impressum</a>
            </li>
            <li class="footer_links">
                <a href="https://htl3rmatteo.at/Datenschutz.html"
class="footer_link">Datenschutz</a>
            </li>
        </ul>
        <p class="copyright">© 2025 Matteo</p>
    </div>
    <script src="https://htl3rmatteo.at/Home.js"></script>
</body>
</html>
```



HTML-Markup Datenschutz:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>Datenschutz | Matteo</title>

    <link rel="icon" type="image/x-icon"
href="https://htl3rmatteo.at/images/Matteo_logo.png">

    <link rel="stylesheet" href="https://htl3rmatteo.at/Matteo.css">

    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">

</head>

<body class="DS">

    <nav class="navbar">

        <div class="navbar_container">

            <a href="https://htl3rmatteo.at/Home.html"
class="navbar_logo">

                Matteo

            </a>

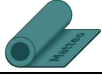
            <div class="navbar_toggle">

                <span class="bar"></span>
                <span class="bar"></span>
                <span class="bar"></span>
                <span class="bar"></span>
                <span class="bar"></span>

            </div>

            <ul class="navbar_menu">

                <li class="navbar_links">
```



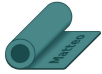
```

        <a href="https://htl3rmatteo.at/Home.html"
class="navbar_link">Home</a>
    </li>
    <li class="navbar_links">
        <a href="https://htl3rmatteo.at/Team.html"
class="navbar_link">Team</a>
    </li>
    <li class="navbar_links">
        <a href="https://htl3rmatteo.at/Idea.html"
class="navbar_link">Idee</a>
    </li>
    <li class="navbar_links">
        <a
href="https://htl3rmatteo.at/Projektfortschritt.html"
class="navbar_link">Meilensteine</a>
    </li>
    <li class="navbar_btn">
        <a href="https://www.instagram.com/matteo_htl3r"
target="_blank" class="button">
            <i class="fa fa-instagram" style="font-size:
24px;"></i>
        </a>
    </li>
</ul>
</div>
</nav>

<!-- Content -->

<div class="content_container">
    <div class="text_container">
        <div class="abschnitt">
            <p class="head">
                Datenschutzerklärung
            </p>
        </div>
    </div>
</div>

```



</p>

<p class="Paragraph">

In folgender Datenschutzerklärung informieren wir Sie über die wichtigsten Aspekte der Datenverarbeitung im Rahmen unserer Webseite.

Wir erheben und verarbeiten personenbezogene Daten nur auf Grundlage der gesetzlichen Bestimmungen (Datenschutzgrundverordnung, Telekommunikationsgesetz 2003).

Sobald Sie als Benutzer auf unsere Webseite zugreifen oder diese besuchen wird Ihre IP-Adresse, Beginn sowie Beginn und Ende der Sitzung erfasst.

Dies ist technisch bedingt und stellt somit ein berechtigtes Interesse iSv Art 6 Abs 1 lit f DSGVO.

</p>

</div>

<div class="abschnitt">

<p class="head">

Kontakt mit uns

</p>

<p class="Paragraph">

Wenn Sie uns per Email kontaktieren, dann werden die von Ihnen an uns übermittelten Daten zwecks Bearbeitung Ihrer Anfrage oder für den Fall von weiteren Anschlussfragen für sechs Monate bei uns gespeichert.

Es erfolgt, ohne Ihre Einwilligung, keine Weitergabe Ihrer übermittelten Daten.

</p>

</div>

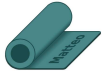
<div class="abschnitt">

<p class="head">

Cookies

</p>

<p class="Paragraph">



Unsere Website verwendet so genannte Cookies. Dabei handelt es sich um kleine Textdateien, die mit Hilfe des Browsers auf Ihrem Endgerät abgelegt werden.

Sie richten keinen Schaden an. Wir nutzen Cookies dazu, unser Angebot nutzerfreundlich zu gestalten. Einige Cookies bleiben auf Ihrem Endgerät gespeichert, bis Sie diese löschen.

Sie ermöglichen es uns, Ihren Browser beim nächsten Besuch wiederzuerkennen.

Wenn Sie dies nicht wünschen, so können Sie Ihren Browser so einrichten, dass er Sie über das Setzen von Cookies informiert und Sie dies nur im Einzelfall erlauben.

Bei der Deaktivierung von Cookies kann die Funktionalität unserer Website eingeschränkt sein.

</p>

</div>

<div class="abschnitt">

<p class="head">

Server-Log Files

</p>

<p class="Paragraph">

Diese Webseite und der damit verbundene Provider erhebt im Zuge der Webseitennutzung automatisch Informationen im Rahmen sogenannter „Server-Log Files“.

Dies betrifft insbesondere:

<ul class="SL_Files">

<li class="SL_File">

IP-Adresse oder Hostname

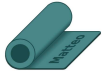
<li class="SL_File">

den verwendeten Browser

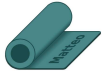
<li class="SL_File">

Aufenthaltsdauer auf der Webseite sowie

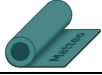
Datum und Uhrzeit



```
<li class="SL_File">
    aufgerufene Seiten der Webseite
</li>
<li class="SL_File">
    Spracheinstellungen und Betriebssystem
</li>
<li class="SL_File">
    „Leaving-Page“ (auf welcher URL hat der
Benutzer die Webseite verlassen)
</li>
<li class="SL_File">
    ISP (Internet Service Provider)
</li>
</ul>
</p>
<p class="Paragraph">
    Wenn sie vermuten, dass im Zuge der Verarbeitung
Ihrer Daten Verstöße gegen das Datenschutzrecht passiert sind,
    so haben Sie die Möglichkeit sich bei uns oder der
Datenschutzbehörde zu beschweren.
</p>
</div>
<div class="abschnitt">
    <p class="head">
        Ihre Rechte als Betroffener
    </p>
    <p class="Paragraph">
        Sie als Betroffener haben bezüglich Ihrer Daten,
welche bei uns gespeichert sind grundsätzlich ein Recht auf:
        <ul class="SL_Files">
            <li class="SL_File">
                Auskunft
            </li>
```



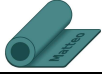
```
<li class="SL_File">
    Löschung der Daten
</li>
<li class="SL_File">
    Berechtigung der Daten
</li>
<li class="SL_File">
    Übertragbarkeit der Daten
</li>
<li class="SL_File">
    Widerruf und Widerspruch zur
Datenverarbeitung
</li>
<li class="SL_File">
    Einschränkung
</li>
</ul>
</p>
<p class="Paragraph">
    Wenn sie vermuten, dass im Zuge der Verarbeitung
Ihrer Daten Verstöße gegen das Datenschutzrecht passiert sind,
    so haben Sie die Möglichkeit sich bei uns oder der
Datenschutzbehörde zu beschweren.
</p>
</div>
<div class="abschnitt">
    <p class="head">
        Kontaktdaten
    </p>
    <p class="Paragraph">
        Webseitenbetreiberin: Mika Böhm <br>
        E-Mail: <a href="mailto:1064@htl.rennweg.at"
class="Member_Mail" target="_blank">1064@htl.rennweg.at</a> <br>
```



```

                                Tel.: <a href="tel: +43 677 62403662"
class="Member_Mail" target="_blank">+43 677 62403662</a>
                                </p>
                                </div>
                                </div>
                                </div>

<!-- Footer -->
<div class="footer_container">
                                <a href="https://htl3rmatteo.at/Home.html"
class="footer_logo_container">
                                
                                <div class="footer_Name">Matteo</div>
                                </a>
                                <ul class="footer_menu">
                                <li class="footer_links">
                                <a href="https://htl3rmatteo.at/Impressum.html"
class="footer_link">Impressum</a>
                                </li>
                                <li class="footer_links">
                                <a href="https://htl3rmatteo.at/Datenschutz.html"
class="footer_link">Datenschutz</a>
                                </li>
                                </ul>
                                <p class="copyright">© 2025 Matteo</p>
                                </div>
                                <script src="https://htl3rmatteo.at/Home.js"></script>
</body>
</html>
```

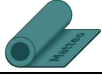


CSS der Website:

```
/*:root {
    --Font_color: #ffaana;
    --Font_color_dark: #370a0a;
    --nav_footer_color: #062121;
    --background_color: #ffaana44;
    --Logo_color: #2e6b6c;
    --border_color: #164949;
    --link_hover_color: #ffe3aa;
}*/

:root {
    --Font_color: #a0a0a0;          /* soft white for good contrast on
dark backgrounds */
    --Font_color_dark: #393939;      /* rich charcoal for light
backgrounds */
    --nav_footer_color: #393939;      /*70a3a3 darker shade of the logo
color for navbar/footer */
    --background_color: #565656;      /* very light teal-tinted gray for
page background */
    --shadow_color: #488687;
    --Logo_color: #a0a0a0;          /* brand teal */
    --border_color: #1c4b4c;          /* medium teal border tone */
    --link_hover_color: #488687;      /* warm gold accent for hover states
*/
    --Status_border_color: #212121;
    --Status_not_started_color: #a0a0a0;
    --Status_in_progress_color: orange;
    --Status_completed_color: limegreen;
}

* {
    box-sizing: border-box;
```



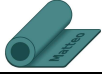
```
margin: 0;
padding: 0;
font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS',
sans-serif;
}

/* Navbar -----
-----*/

html, body {
    overflow-x: hidden;
}

.navbar {
    background-color: var(--nav_footer_color);
    box-shadow: 0rem 0.1rem 0rem 0.1rem var(--shadow_color);
    display: flex;
    justify-content: center;
    align-items: center;
    position: sticky;
    top: 0;
    height: 60px;
    z-index: 999;
}

.navbar_container {
    display: flex;
    justify-content: space-between;
    align-items: center;
    height: 100%;
    width: 100%;
    padding: 2rem;
```



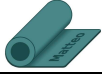
```
}

.navbar_logo {
    color: var(--link_hover_color);
    font-size: 2.5rem;
    text-decoration: none;
    z-index: 900;
}

.navbar_menu {
    display: flex;
    align-items: center;
    list-style: none;
    text-align: center;
}

.navbar_links {
    display: flex;
    justify-content: center;
    height: 100%;
}

.navbar_link {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100%;
    padding: 0 1rem;
    color: var(--Font_color);
    text-decoration: none;
    font-size: 1.2rem;
    transition: all 0.3s ease;
```

```
}

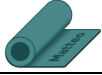
.navbar_link:hover {
    color: var(--link_hover_color);
}

.navbar_links:nth-child(4) {
    padding-right: 1rem;
}

.navbar_btn {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100%;
}

.button {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 30px;
    width: 30px;
    border-radius: 5px;
    border: none;
    background-color: var(--Font_color);
    text-decoration: none;
    color: var(--Font_color_dark);
    transition: all 0.3s ease;
}

.button:hover {
```



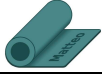
```
        background-color: var(--link_hover_color);
    }

    @media screen and (max-width: 960px) {
        .navbar_container {
            padding: 0;
            width: 100%;
        }

        .navbar_logo {
            margin-left: 1rem;
        }

        .navbar_menu {
            display: grid;
            grid-template-columns: auto;
            justify-content: center;
            width: 100%;
            position: absolute;
            top: -1000px;
            opacity: 0;
            z-index: -1;
            height: 50vh;
        }

        .navbar_menu.active {
            top: 100%;
            opacity: 1;
            z-index: 1;
            height: 50vh;
            width: 100%;
            background-color: var(--nav_footer_color);
        }
    }
}
```



```
        transition: top 0.3s, z-index 0.3s, background-color 0.3s,  
        opacity 0.5s;
```

```
        padding-bottom: 1rem;
```

```
    }
```

```
.navbar_toggle {
```

```
    margin-right: 1rem;
```

```
    z-index: 900;
```

```
}
```

```
.navbar_toggle .bar {
```

```
    display: block;
```

```
    height: 3px;
```

```
    width: 25px;
```

```
    background-color: var(--Font_color);
```

```
    margin: 0 auto;
```

```
    cursor: pointer;
```

```
}
```

```
.navbar_toggle .bar:nth-child(2) {
```

```
    height: 5px;
```

```
    background-color: var(--nav_footer_color);
```

```
}
```

```
.navbar_toggle .bar:nth-child(4) {
```

```
    height: 5px;
```

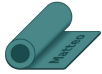
```
    background-color: var(--nav_footer_color);
```

```
}
```

```
.navbar_links:nth-child(4) {
```

```
    padding: 0;
```

```
}
```



```
.navbar_btn {
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 0;
    height: 100%;
}

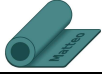
.navbar_btn > .button {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 100%;
}

.navbar_toggle.active .bar:nth-child(1) {
    transform: translateY(8px) rotate(45deg);
}

.navbar_toggle.active .bar:nth-child(5) {
    transform: translateY(-8px) rotate(-45deg);
}

.navbar_toggle.active .bar:nth-child(3) {
    background-color: var(--nav_footer_color);
}

.navbar_toggle .bar {
    transition: 0.3s ease;
}
```



```
.navbar_toggle .bar:nth-child(3) {
    transition: 0.1s;
}

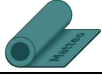
}

/* Home -----
-----*/

.content_container {
    display: flex;
    justify-content: space-evenly;
    align-items: center;
    /*height: 85vh;*/
    width: 100%;
    padding: 4rem;
    background-color: var(--background_color);
}

.text_info_container {
    align-self: stretch;
    display: grid;
    grid-template-columns: auto;
    justify-content: space-between;
    align-content: space-between;
    align-items: space-between;
    margin-right: 2rem;
}

.logo_container {
    display: flex;
    justify-content: space-between;
```

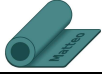


```
        align-items: center;
        width: 100%;
        height: 100%;
    }

    #logo_img {
        height: 100px;
        width: 150px;
        padding: 5px;
        border-radius: 10px;
        background-color: var(--nav_footer_color);
        box-shadow: -0.1rem -0.25rem 0px 0.25rem var(--shadow_color);
    }

    .content_logo {
        display: flex;
        justify-content: center;
        align-items: center;
        width: 90%;
        margin-left: 2rem;
        padding: 5px;
        background-color: var(--nav_footer_color);
        color: var(--Font_color);
        box-shadow: -0.1rem -0.25rem 0px 0.25rem var(--shadow_color);
        border-radius: 10px;
        font-size: 5rem;
    }

    .Slogan_nav_container {
        display: flex;
        flex-direction: column;
        justify-content: space-evenly;
```

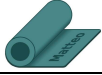


```
    align-items: center;
    max-width: 1000px;
    background-color: var(--nav_footer_color);
    box-shadow: -0.1rem -0.25rem 0px 0.25rem var(--shadow_color);
    padding: 2rem;
    margin-top: 2rem;
    border-radius: 10px;
}
```

```
.Slogan {
    display: flex;
    justify-content: center;
    align-items: center;
    text-align: center;
    height: 5rem;
    font-size: 3rem;
    text-decoration: none;
    color: var(--Font_color);
}
```

```
.nav_cards {
    display: flex;
    justify-content: space-evenly;
    align-items: center;
    list-style: none;
    height: 30vh;
}
```

```
.card {
    display: flex;
    justify-content: center;
    align-items: center;
```



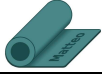
```
    position: relative;
    width: 7rem;
    height: 7rem;
    margin: 0 2rem;
    background: #00000000;
    color: #00000000;
    font-size: 1.7rem;
    text-decoration: none;
    transition: all 0.5s;
    z-index: 1;
}

.Team_img {
    position: absolute;
    justify-self: center;
    align-self: center;
    transition: all 0.5s ease;
    opacity: 1;
}

.Team_img:hover {
    opacity: 0.15;
}

.card_text {
    width: 100%;
    text-align: center;
}

.card:hover {
    color: var(--Font_color_dark);
}
```

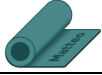



```
.card::before {  
    position: absolute;  
    content: "";  
    top: 0;  
    left: 0;  
    width: 7rem;  
    height: 7rem;  
    border-radius: 5px;  
    background: var(--Logo_color);  
    z-index: -1;  
    transition: all 0.5s;  
}
```

```
.card:hover::before {  
    transform: scale(1.2);  
    border-radius: 15px;  
}
```

```
.img_container {  
    display: flex;  
    justify-content: center;  
    align-items: center;  
    align-self: flex-end;  
    height: 100%;  
    width: 100%;  
    border-radius: 10px;  
    max-width: 840px;  
}
```

```
.img_inner_container {  
    display: flex;
```

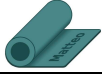


```
        justify-content: center;
        align-items: center;
        align-content: space-around;
        width: 100%;
        background-color: var(--nav_footer_color);
        border-radius: 10px;
    }

    #img {
        width: 100%;
        background-color: var(--nav_footer_color);
        box-shadow: -0.1rem -0.25rem 0px 0.25rem var(--shadow_color);
        border-radius: 10px;
        padding: 2rem;
    }

    .Sponsoren_container {
        display: grid;
        grid-template-columns: 1fr 1fr;
        justify-content: center;
        align-items: center;
        padding: 2rem 5rem;
        padding-bottom: 7rem;
        background-color: var(--background_color);
    }

    .Sponsor{
        display: flex;
        justify-content: center;
        align-items: center;
        background-color: var(--nav_footer_color);
        padding: 1rem;
```



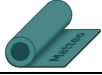
```
margin: 1rem 1rem;
border-radius: 10px;
box-shadow: -0.1rem -0.25rem 0px 0.25rem var(--shadow_color);
}
```

```
.Sponsor_Link {
  display: grid;
  grid-template-columns: 1fr;
  justify-content: center;
  justify-items: center;
  align-items: center;
  padding: 1rem 0.5rem 0 0.5rem;
  font-size: 1.5rem;
  height: 10rem;
  text-decoration: none;
  color: var(--Font_color);
}
```

```
.Sponsor:nth-child(3) {
  grid-column: 1 / span 2;
  width: 48%;
  justify-self: center;
}
```

```
.Sponsor:nth-child(4) {
  grid-column: 1 / span 2;
  width: 98%;
  justify-self: center;
}
```

```
.Sponsor p {
  display: grid;
```



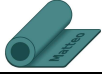
```
justify-content: center;
justify-items: center;
padding: 0.5rem;
font-size: 2rem;
text-decoration: none;
color: var(--Font_color);
}

@media screen and (max-width: 1200px) {

    .content_container {
        display: grid;
        grid-template-columns: 1fr;
        justify-content: center;
        justify-items: center;
        align-items: center;
        /*height: 150vh;*/
        padding: 2rem;
        padding-bottom: 1rem;
    }

    .text_info_container {
        /*height: 75%;*/
        align-content: space-between;
        margin: 0;
        margin-bottom: 1rem;
    }

    .logo_container {
        padding-bottom: 1rem;
    }
}
```



```
.img_container {
    margin-top: 2rem;
    width: 720px;
}

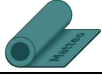
}

@media screen and (max-width: 800px) {

    .content_container {
        /*height: 150vh;*/
        padding: 0.5rem;
        padding-bottom: 1rem;
        width: 100%;
    }

    .text_info_container {
        height: 100%;
        padding: 0;
        width: 90%;
        padding-bottom: 1rem;
        padding-top: 2rem;
    }

    .logo_container {
        display: grid;
        grid-template-columns: 1fr;
        justify-content: center;
        justify-items: center;
        align-content: space-between;
        height: 250px;
    }
}
```



```
        width: 100%;
    }

    .content_logo {
        margin: 0;
        width: 100%;
    }

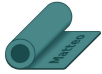
    .Slogan_nav_container {
        width: 100%;
        margin-bottom: 1rem;
        margin-top: 0.5rem;
        justify-self: baseline;
    }

    .Slogan {
        padding-bottom: 2rem;
        height: auto;
        font-size: 2rem;
    }

    .nav_cards {
        flex-direction: column;
        align-items: space-between;
        height: 100%;
    }

    .nav_card {
        padding: 1rem 0;
    }

    .card {
```



```
        color: var(--Logo_color);
    }

    .Team_img {
        opacity: 0.15;
    }

    .card_text {
        color: var(--Font_color_dark);
    }

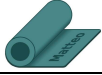
    .card:hover {
        transform: none;      /* keine Skalierung */
        color: inherit;      /* keine Farbänderung */
    }

    .card::before {
        transform: none;      /* keine Animation beim Hover */
    }

    .img_container {
        margin-bottom: 1rem;
        margin-top: 0.5rem;
        width: 90%;
    }

    #img {
        padding: 0.5rem;
    }

    .Sponsoren_container {
        grid-template-columns: 1fr;
```



```
padding: 0.5rem;
padding-bottom: 4rem;
}

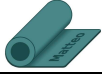
.Sponsor:nth-child(1) {
    grid-column: 1;
    width: 90%;
    justify-self: center;
}

.Sponsor:nth-child(2) {
    grid-column: 1;
    width: 90%;
    justify-self: center;
}

.Sponsor:nth-child(3) {
    grid-column: 1;
    width: 90%;
}

.Sponsor:nth-child(4) {
    grid-column: 1;
    width: 90%;
    font-size: 0.5rem;
    justify-content: center;
}

.Home .Member_Mail {
    font-size: 1rem;
}
```

```
br.desktop_only {
    display: none;
}

p {
    text-align: center;
}

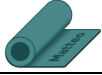
a {
    padding-top: 0.2rem;
}

b {
    font-size: 1.5rem;
}

b.desktop_only {
    display: none;
}
}

/* Team page -----
-----
-----*/

.Team_content_container {
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: var(--background_color);
    padding: 4rem;
    padding-bottom: 5rem;
}
```



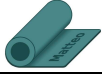
```
}

.Team_container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr;
    height: 100%;
    width: 100%;
    position: relative;
    background-color: var(--nav_footer_color);
    padding: 1rem;
    box-shadow: -0.1rem -0.25rem 0rem 0.25rem var(--shadow_color);
    border-radius: 5px;
}

.Member_container {
    display: grid;
    grid-template-columns: auto;
    align-items: center;
    justify-content: center;
    justify-items: center;
    padding: 0 1rem;
}

.Member_container:nth-child(4) {
    grid-column: 2;
    justify-self: flex-start;
    padding-bottom: 1rem;
    padding-right: 1rem;
}

.Member_container:nth-child(5) {
    grid-column: 3;
```



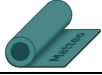
```
        justify-self: flex-end;
        padding-bottom: 1rem;
        padding-left: 1rem;
    }

    .Member_container:nth-child(1),
    .Member_container:nth-child(3) {
        padding-top: 1rem;
        width: 23rem;
    }

    .Member_info_container {
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
        font-size: 1.2rem;
        color: var(--Font_color);
    }

    .Member_Mail {
        text-decoration: none;
        color: var(--Font_color);
    }

    .Team_text_container {
        display: grid;
        grid-column: 2 / span 2;
        grid-template-columns: 1fr 1fr;
        justify-self: center;
        justify-items: center;
        padding: 2rem;
    }
```



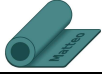
```
        font-size: 3rem;
        color: var(--Font_color);
    }

    .Member_img_container {
        display: flex;

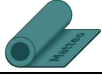
    }

    .Team_text {
        padding-bottom: 1rem;
        font-size: 6rem;
        grid-column: 1 / span 2;
    }

    .btn{
        display: flex;
        justify-content: center;
        align-items: center;
        justify-self: center;
        grid-column: 1 / span 2;
        height: 50px;
        width: 50px;
        border-radius: 10px;
        border: none;
        background-color: var(--Font_color);
        text-decoration: none;
        color: var(--Font_color_dark);
        transition: all 0.3s ease;
        cursor: pointer;
    }
```



```
.btn:hover {  
    background-color: var(--link_hover_color);  
}  
  
.btn i {  
    color: var(--Font_color_dark); /* Iconfarbe */  
}  
  
@media screen and (max-width: 1200px) {  
  
    .Team_content_container {  
        padding: 2rem;  
        padding-bottom: 5rem;  
    }  
  
    .Team_container {  
        grid-template-columns: 1fr 1fr;  
    }  
  
    .Team_text_container {  
        order: 1;  
        grid-column: 1 / span 2;  
    }  
  
    .Member_container:nth-child(1) {  
        order: 2;  
        padding-top: 0;  
    }  
  
    .Member_container:nth-child(3) {  
        order: 3;  
        padding-top: 0;  
    }  
}
```



```
}

.Member_container:nth-child(4) {
    order: 4;
    grid-column: 1;
    justify-self: center;
    padding-top: 2rem;
}

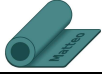
.Member_container:nth-child(5) {
    order: 5;
    grid-column: 2;
    justify-self: center;
    padding-top: 2rem;
}
}

@media screen and (max-width: 800px) {

    .Team_content_container {
        padding: 2rem 1rem 5rem 1rem;
        width: 100%;
    }

    .Team_container {
        grid-template-columns: 1fr;
        padding: 0;
        width: 90%;
    }

    .Member_container {
        padding: 0;
    }
}
```



```
        width: 90%;
    }

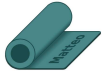
    .Member_container:nth-child(1),
    .Member_container:nth-child(3),
    .Member_container:nth-child(4),
    .Member_container:nth-child(5) {
        width: 90%;
        justify-self: center;
        padding: 0;
        padding-left: 0;
        padding-bottom: 1rem;
        padding-right: 0;
        padding-top: 0;
    }

    .Member_info_container p {
        font-size: 1rem;
    }

    .Member_info_container a {
        font-size: 1rem;
    }

    .Member_info_container .Member_Name {
        font-size: 1.5rem;
        font-style: bold;
    }

    .Team_text_container {
        grid-column: 1;
    }
```



```
.Member_container:nth-child(1) {
    grid-column: 1;
}

.Member_container:nth-child(3) {
    grid-column: 1;
}

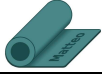
.Member_container:nth-child(4) {
    grid-column: 1;
}

.Member_container:nth-child(5) {
    grid-column: 1;
}
}

/* Idea Page -----
-----
-----*/

.Idea .content_container {
    display: grid;
    grid-template-columns: 1fr;
    justify-items: center;
}

.Idea .Slogan_nav_container {
    padding: 2rem 3rem;
    margin-bottom: 2rem;
    justify-self: center;
}
```

```
}

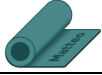
.Idea .img_container {
    margin-bottom: 2rem;
}

.Idea_header_container {
    display: flex;
    justify-self: center;
    justify-content: center;
    align-items: center;
    width: 100%;
    padding: 1rem 2rem;
    background-color: var(--nav_footer_color);
    box-shadow: -0.1rem -0.25rem 0rem 0.25rem var(--shadow_color);
    border-radius: 10px;
}

.Idea_header {
    font-size: 4.5rem;
    color: var(--Font_color);
}

.Idea h1 {
    color: var(--Font_color);
}

.Absatz {
    font-size: 1.5rem;
    color: var(--Font_color);
}
```



```
@media screen and (max-width: 800px) {
```

```
  .Idea .content_container {  
    padding-bottom: 5rem;  
  }
```

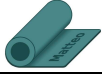
```
  .Idea .text_info_container {  
    justify-items: center;  
    width: 95%;  
    margin-bottom: 0;  
    padding-bottom: 0;  
  }
```

```
  .Idea .Idea_header_container {  
    padding: 0.5rem 0;  
    width: 95%;  
  }
```

```
  .Idea .Idea_header {  
    font-size: 3rem;  
  }
```

```
  .Idea .Slogan_nav_container {  
    width: 95%;  
    margin-bottom: 0;  
    margin-top: 2rem;  
    padding: 1rem 0.5rem;  
    font-size: 1rem;  
  }
```

```
  .Idea .head {  
    font-size: 1.5rem;
```



```
}

.Idea .SL_Files {
    margin-left: 1.5rem;
    font-size: 1rem;
}

.Idea .Absatz {
    font-size: 1rem;
}

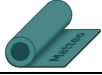
.Idea .img_container {
    align-self: center;
    margin-top: 2rem;
    margin-bottom: 0;
}

}

/* Milestones -----
-----
-----*/

.Milestones h1 {
    font-size: 2.5rem;
    color: var(--Font_color);
}

.Milestones p {
    font-size: 2rem;
    color: var(--Font_color);
}
```

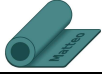


```
.Milestone_List_container {  
    display: flex;  
    justify-content: center;  
    align-items: center;  
    width: 100%;  
    padding: 2rem 2rem 6rem 2rem;  
    background-color: var(--background_color);  
}
```

```
.Milestone_List {  
    display: grid;  
    grid-template-columns: 1fr;  
    justify-items: center;  
    list-style: none;  
}
```

```
.Milestone_container {  
    display: flex;  
    justify-content: space-between;  
    align-items: center;  
    width: fit-content;  
    max-width: 60%;  
    padding: 1rem;  
    margin: 1rem 0;  
    border-radius: 5px;  
    border: none;  
    background-color: var(--nav_footer_color);  
    box-shadow: -0.1rem -0.25rem 0rem 0.25rem var(--shadow_color);  
}
```

```
.Milestone_Text_container {
```

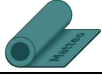


```
display: grid;
grid-template-columns: auto;
justify-content: center;
align-items: center;
max-width: 90%;
margin-right: 1rem;
}

.Milestone_Header {
    align-items: center;
    font-size: 2rem;
    font-weight: bold;
    color: var(--Font_color);
}

.Milestone_Info {
    align-items: center;
    font-family: 1.5rem;
    color: var(--Font_color);
}

.Milestone_Status{
    width: 20px;
    height: 20px;
    border-radius: 50%;
    border: 3px solid;
    box-shadow: 0 0 2px inset var(--Status_border_color), 0 0 5px var(-
-Status_border_color);
    border-color: var(--Status_border_color);
    background-color: var(--Status_not_started_color);
    align-self: baseline;
}
```



```
.done {
    background-color: var(--Status_completed_color);
}

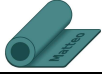
.in_Progress{
    background-color: var(--Status_in_progress_color);
}

.dot {
    display: flex;
    justify-self: center;
    align-items: center;
    width: 10px;
    height: 10px;
    margin: 10px 0;
    border-radius: 50%;
    box-shadow: -0.02rem -0.05rem 0rem 0.05rem var(--shadow_color);
    background-color: var(--nav_footer_color);
}

@media screen and (max-width: 800px) {

    .PF .Milestone_List_container {
        width: 100%;
        padding: 0;
    }

    .PF .Milestone_List {
        width: 100%;
        padding: 2rem 1rem 5rem 1rem;
    }
}
```



```
.PF .Milestone_container {
    max-width: 100%;
    width: 90%;
    margin: 0.5rem 0 0 0;
    padding: 0.5rem;
}

.PF .Milestone_Text_container {
    margin: 0;
}

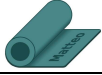
.PF .Milestone_Header {
    font-size: 1.5rem;
}

.PF .Milestone_Status {
    margin: 0;
}

}

/* Datenschutz und Impressum -----
-----
-----*/

.text_container {
    display: grid;
    grid-template-columns: 1fr;
    justify-content: center;
    align-items: center;
    width: 100%;
```



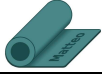
```
    height: 100%;
    padding: 3rem 6rem;
    border-radius: 10px;
    background-color: var(--nav_footer_color);
    box-shadow: -0.1rem -0.25rem 0px 0.25rem var(--shadow_color);
}

.abschluss {
    display: grid;
    grid-template-columns: 1fr;
    align-items: center;
    padding: 2rem;
}

.head {
    font-size: 2rem;
    font-weight: bold;
    color: var(--shadow_color);
}

.Paragraph {
    margin-left: 1rem;
    font-size: 1.5rem;
    color: var(--Font_color);
}

.SL_Files {
    margin-left: 3.5rem;
    font-size: 1.5rem;
    color: var(--Font_color);
}
```

```
@media screen and (max-width: 800px) {
```

```
    .DS .content_container {  
        padding: 2rem 1rem 5rem 1rem ;  
    }
```

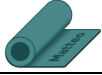
```
    .DS .SL_Files {  
        font-size: 1rem;  
        margin-left: 1.5rem;  
    }
```

```
    .Impressum .content_container {  
        width: 100%;  
        padding: 2rem 1rem 5rem 1rem;  
    }
```

```
    .text_container {  
        width: 95%;  
        padding: 0.5rem;  
        justify-items: center;  
    }
```

```
    .head {  
        font-size: 1.5rem;  
        text-align: left;  
    }
```

```
    .Paragraph {  
        margin: 0;  
        font-size: 1rem;  
        text-align: left;  
    }
```



```
.Impressum .Paragraph b {
    font-size: 1.2rem;
}

.abschnitt {
    width: 90%;
    padding: 1rem 0;
}

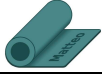
}

/* Under Construction -----
-----
-----*/

.UC .content_container {
    height:75vh;
}

.Info_container {
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 2rem;
    border-radius: 10px;
    background-color: var(--nav_footer_color);
    box-shadow: -0.1rem -0.25rem 0px 0.25rem var(--shadow_color);
}

.text {
    font-size: 1.5rem;
    color: var(--Font_color);
}
```



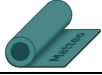
```
}

/* Footer -----
-----*/

.footer_container {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    position: relative;
    width: 100%;
    height: 100%;
    background-color: var(--nav_footer_color);
    box-shadow: 0rem -0.1rem 0px 0.1rem var(--shadow_color);
    padding: 1rem;
}

.footer_logo_container {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    text-decoration: none;
}

.footer_logo {
    position: absolute;
    top: -70px;
    height: 60px;
    width: 90px;
}
```



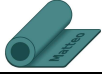
```
border-radius: 5px;
padding: 2px;
}

.footer_Name {
  display: flex;
  justify-content: center;
  align-items: center;
  color: var(--Font_color);
  text-decoration: none;
  font-size: 1.5rem;
}

.footer_menu {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 30%;
  list-style: none;
  padding-top: 1rem;
}

.footer_links {
  display: flex;
  align-items: center;
  width: 10rem;
  padding: 1rem;
}

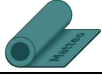
.footer_menu :nth-child(1) {
  justify-content: flex-end;
}
```



```
.footer_link {  
    display: flex;  
    justify-content: center;  
    align-items: center;  
    font-size: 1rem;  
    color: var(--Font_color);  
    text-decoration: none;  
    transition: all ease-in-out 0.3s;  
}
```

```
.footer_link:hover {  
    color: var(--shadow_color);  
    transition: all ease-in-out 0.3s;  
}
```

```
.copyright {  
    display: flex;  
    justify-content: center;  
    align-items: center;  
    width: 100%;  
    font-size: 1rem;  
    text-decoration: none;  
    color: var(--Font_color);  
}
```



Firmware des PIC16F1827:

```
/*
 * Folgendes Programm ist die endgültige Version der PIC16F1827
Firmware für die Diplomarbeit Matteo.
 * Geschrieben von Mika Böhm im Schuljahr 2025/26 in der Klasse 5AM
der HTBLA Rennweg.
*/

// Bibliotheken -----
-----

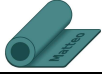
#include <pic16f1827.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include "mcc_generated_files/system/system.h"

// Globale Variablen -----
-----

// Befehle -----
--

#define max_Inst_Length 6
#define D_data_index    2

// Eingehend
const uint8_t KF[max_Inst_Length] = {'>', 'K', 'F',  0, 10, 0};
const uint8_t KN[max_Inst_Length] = {'>', 'K', 'N', 10,  0, 0};
const uint8_t KON[max_Inst_Length] = {'>', 'K', 'O', 'N', 10, 0};
const uint8_t KOFF[max_Inst_Length] = {'>', 'K', 'O', 'F', 'F', 10};
const uint8_t iE[max_Inst_Length] = {'>', 'E', 10,  0,  0, 0};
const uint8_t iP[max_Inst_Length] = {'>', 'P', 10,  0,  0, 0};
const uint8_t iPA[max_Inst_Length] = {'>', 'P', 'A', 10,  0, 0};
```



```
// Ausgehend
const  uint8_t GR[max_Inst_Length]  = {'>', 'G', 'R', 10, 0, 0};
const  uint8_t oE[max_Inst_Length]  = {'>', 'E', 10, 0, 0, 0};
const  uint8_t oP[max_Inst_Length]  = {'>', 'P', 10, 0, 0, 0};
const  uint8_t oPA[max_Inst_Length] = {'>', 'P', 'A', 10, 0, 0};
volatile uint8_t D[max_Inst_Length]  = {'>', 'D', 1, 10, 0, 0};

volatile uint8_t last_outgoing_Inst[max_Inst_Length] = {0, 0, 0, 0, 0,
0};

// UART Buffer -----
--

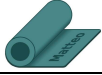
#define Buffer_Size 8

volatile uint8_t Data = 0;

volatile uint8_t Buffer[Buffer_Size];
volatile uint8_t Head = 0;
volatile uint8_t Tail = 0;
// Makros
#define Buffer_amount() ((uint8_t)(Tail - Head))
#define Buffer_Full()   (Buffer_amount() == Buffer_Size)
#define Buffer_Empty()  (Buffer_amount() == 0)

// RX
#define RX_Buffer_Size 20

volatile uint8_t RX_Buffer[RX_Buffer_Size]; // Speichert empfangene
Zeichen bis diese Ausgelesen werden können
volatile uint8_t RX_head = 0;                // Wenn aus dem Buffer
gelesen wird, wird der Head inkrementiert
```



```
volatile uint8_t RX_tail = 0; // Wird in den Buffer
geschrieben, wird der Tail inkrementiert

// Makros
#define RX_amount() ((uint8_t)(RX_tail - RX_head))
#define RX_Full() (RX_amount() == RX_Buffer_Size)
#define RX_Empty() (RX_amount() == 0)

//TX
#define TX_Buffer_Size 15

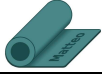
volatile uint8_t TX_Buffer[TX_Buffer_Size]; // Speichert Zeichen bis
diese gesendet werden können
volatile uint8_t TX_head = 0; // Wenn aus dem Buffer
gelesen wird, wird der Head inkrementiert
volatile uint8_t TX_tail = 0; // Wird in den Buffer
geschrieben, wird der Tail inkrementiert

// Makros
#define TX_amount() ((uint8_t)(TX_tail - TX_head))
#define TX_Full() (TX_amount() == TX_Buffer_Size)
#define TX_Empty() (TX_amount() == 0)

// ADC -----
--

volatile uint16_t ADCres; // Zum Speichern des ADC Ergebnis-
Registerpaar
volatile uint8_t ADCresH; // Zum Speichern des ADC Ergebnis-Register
High
volatile uint8_t ADCresL; // Zum Speichern des ADC Ergebnis-Register
Low

// Zeitvariablen -----
--
```

```
#define t_send_RFc 4 // Wie oft wird die aktuelle Belastung gesendet
[s]

#define t_send_Ping 1 // Wie oft wird ein Ping gesendet [s]

#define t_hold_Target 10

volatile uint8_t interrupt_count = 0;

volatile unsigned long t = 0; // Vergangene Sekunden seit
Programmstart

volatile unsigned long t_prev_RFc = 0; // Wann wurde zuletzt die
Belastung gesendet

volatile unsigned long t_prev_Ping = 0; // Wann wurde zuletzt ein
Ping gesendet

volatile unsigned long t_Target_complete = 0;

volatile uint8_t hold_sec_count = 0;

// Sensordaten -----
--

#define Sensor_Amount 12

volatile uint16_t Sensor_Value_raw[Sensor_Amount]; // Rohdaten
volatile uint8_t Sensor_Value[Sensor_Amount]; // Daten nach
Linearisierung

volatile uint8_t Sensor_avg = 0;

volatile uint8_t RF_Sensor_avg = 0; // Durchschnittlicher
Sensor Wert des rechten Fuß

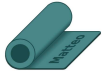
volatile uint8_t LF_Sensor_avg = 0; // Durchschnittlicher
Sensor Wert des linken Fuß

volatile uint16_t RF_Sensor_avg_raw = 0;

volatile uint16_t LF_Sensor_avg_raw = 0;

// Konfigurationsvariablen -----
--

const bool debug = false; // Debug Modus ein oder ausschalten
```



```
volatile bool Target          = true;  // Übung mit Ziel
volatile bool exercise_run    = false; // Übung läuft nicht
volatile bool RX_Error        = false; // Kein Fehler bei der Übertragung
volatile bool ESP_available    = false; // ESP reagiert nicht

volatile uint8_t RFt = 50; // Zielbelastung für den rechten Fuß
volatile uint8_t Tol = 5;

// Laufzeitvariablen -----
--

volatile bool ACK                      = false; // Acknowledge nicht
empfangen

volatile bool resend_Inst              = false; // Letzten Befehl nicht
erneut schicken

volatile bool RFc_inside_Tolerance    = false; // Momentane Belastung
ist nicht innerhalb der Zieltoleranz

volatile bool Target_reached          = false;

volatile uint8_t RFc = 0; // Momentane Belastung
volatile uint8_t LFc = 0;

// State-Machine

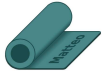
enum States {IDLE, TX_ERROR, RX_ERROR, PING, EXERCISE_RUN,
HOLD_TARGET, TARGET_REACHED};

enum States State;

// Initialisierungen -----
-----

// GPIO-Pins -----
--

void GPIO_init(void)
{
    // RB      76543210
```



```
PORTB  = 0b00000000;
LATB   = 0b00000000;
ANSELB = 0b00000000;
WPUB   = 0b00000000;
TRISB  = 0b00000010; // RB1 RX - Input

// RA      76543210
PORTA  = 0b00000000;
LATA   = 0b00000000;
ANSELA = 0b00010000; // RA4 ADC_IN - Analog ... Sensoreingang
WPUA   = 0b00000000;
TRISA  = 0b00010000; // RA4 ADC_IN - Input ... Sensoreingang

}

// UART-Modul -----
--

void UART_init(void)
{
    //Baud Rate and EUSART Mode

    SPBRGH = 0b00000000; // Baud Rate 57600 > SPBRGH/L =
0b0000000001000100
    SPBRGL = 0b01000100;

    TXSTAbits.BRGH = 1; // High Baud Rate Select Bit = 1 > High
Baud Rate

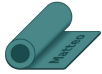
    BAUDCONbits.BRG16 = 1; // 16bit Baud Rate Generator bit = 1 > 16bit
Mode

    TXSTAbits.SYNC = 0; // EUSART Mode Select bit = 0 > Asynchronous
Mode

    RCSTAbits.SPEN = 1; // Serial Port Enable bit

    TXSTAbits.TXEN = 1;

    RCSTAbits.CREN = 1;
```



```
// Interrupts
INTCONbits.GIE    = 1;
INTCONbits.PEIE   = 1;
PIE1bits.TXIE     = 0;
PIE1bits.RCIE     = 1;

}

// ADC-Modul -----
--
void ADC_init(void)
{
    // 0 unwichtig, 00100 Kanalauswahl RA4, 0 unwichtig, 0 ADC
    // ausgeschalten
    ADCON0 = 0b00010000;

    // 1 right justified, 101 1 MHz Arbeitstakt, 0 unwichtig, 0
    // negative Referenz = VSS, 0 positive Referenz = VDD
    ADCON1 = 0b11010000;

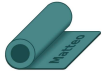
    // ADC Interrupt verhindern
    PIE1bits.ADIE = 0;

    // Ergebnis Register
    ADRESH = 0x0;
    ADRESL = 0x0;

    // ADC Interrupt Flag
    PIR1bits.ADIF = 0;

    // ADC einschalten
    ADCON0bits.ADON = 1;
}

// TMR1-Modul -----
--
```



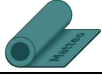
```
void TMR1_init(void)
{
    T1CON = 0b00110000; //Prescaler is 8, increments at 500 000 Hz
    T1GCON = 0b00000000;
    TMR1H = 0;
    TMR1L = 0;
    PIR1bits.TMR1IF = 0;
    PIE1bits.TMR1IE = 1;
    T1CONbits.TMR1ON = 1;
}

// ISR -----
-----

void __interrupt(low_priority) Interrupt()
{
    if(PIR1bits.TMR1IF) // TMR1 Interrupt. Rollover hat stattgefunden
    {
        interrupt_count++;

        if(interrupt_count == 10) // 10 Rollover entsprechen einer
        Sekunde
        {
            interrupt_count = 0;
            t++; // Sekundenzähler inkrementieren
        }

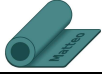
        TMR1H = 0x3C; // TMR1 auf 15536 setzten um ein Rollover jede
        zehntel Sekunde zu erzeugen.
        TMR1L = 0xB0;
        PIR1bits.TMR1IF = 0; // Flag zurücksetzen
    }
}
```



```
    if(PIR1bits.TXIF && PIE1bits.TXIE) // UART TX Interrupt. UART ist
bereit zu senden
    {
        if(!TX_Empty()) // Sind Daten vorhanden, die gesendet werden
sollen
        {
            TXREG = TX_Buffer[TX_head % TX_Buffer_Size]; // TXREG wird
mit Daten beschrieben, diese werden gesendet
            TX_head++;
        }
        else
        {
            PIE1bits.TXIE = 0; // Wenn keine Daten gesendet werden
müssen
        }
    }

    if(PIR1bits.RCIF && PIE1bits.RCIE)
    {
        if(RCSTAbits.OERR) // Ist ein Overflow aufgetreten
        {
            RCSTAbits.CREN = 0; // Schnittstelle neustarten
            RCSTAbits.CREN = 1;
        }

        if(!RX_Full()) // Wenn RX Buffer nicht voll ist
        {
            if(RCREG == '<') // Acknowledge wurde empfangen
            {
                ACK = true;
            }
            else // Daten wurden empfangen
            {
```



```
        RX_Buffer[RX_tail % RX_Buffer_Size] = RCREG; // Daten
in den RX Buffer schreiben
        RX_tail++;
    }
    NOP();
}
else // Wenn RX Buffer voll ist
{
    uint8_t dummy = RCREG; // Daten gehen verloren
}
}

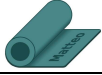
// Funktionsdeklarationen -----
-----

// Hilfsfunktionen

uint8_t map(uint8_t value, uint8_t vUG, uint8_t vOG, uint8_t tUG,
uint8_t tOG) // Überträgt eine Zahl von einem Wertebereich in den
anderen
{
    return ((long)(value - vUG) * (long)(tOG - tUG)) / (vOG - vUG) +
tUG;
}

uint8_t round5(uint8_t value) // Rundet eine Zahl auf ein Vielfaches
von 5
{
    if((value % 5) >= 3)
    {
        return value + (5 - (value % 5));
    }
    else
    {

```



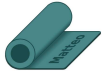
```
        return value - (value % 5);
    }
}

void uint16_ascii(uint16_t value, uint8_t *Str) // Wandelt eine 16bit
Integer in einen Ascii-String um
{
    uint8_t deconst_value[6];
    uint8_t i = 0;
    uint8_t j = 0;

    if(value == 0)
    {
        Str[0] = '0';
        Str[1] = '\\0';
        return;
    }

    while(value > 0)
    {
        deconst_value[i++] = (value % 10) + '0';
        value /= 10;
    }

    while(i > 0)
    {
        Str[j++] = deconst_value[--i];
    }
    Str[j] = '\\0';
}
```

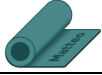



```
// Treiberfunktionen
// UART
void UART_Transmit_Byte(uint8_t Data) // sendet ein Byte Daten
{
    while(1) // warten bis der TX Buffer platz hat
    {
        if(!TX_Full())
        {
            break;
        }
    }
    TX_Buffer[TX_tail % TX_Buffer_Size] = Data;
    TX_tail++;
    PIE1bits.TXIE = 1; // UART TX Interrup zulassen
}

void UART_Transmit_2Byte(uint16_t Byte) // sendet 2 Bytes Daten
{
    uint8_t ByteL = Byte & 0x00ff; // 16bit Integer auftrennen
    uint8_t ByteH = (Byte >> 8) & 0x00ff;
    UART_Transmit_Byte(ByteH);
    UART_Transmit_Byte(ByteL);
}

void UART_Transmit_Data(uint8_t *word, uint16_t arg0, uint16_t arg1,
uint16_t arg2) // Sendet Daten und Strings
{
    int arg_count = 0;
    uint16_t next_arg = 0;

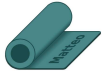
    while(true)
    {
```



```
switch(arg_count)
{
    case 0:
        next_arg = arg0;
        break;
    case 1:
        next_arg = arg1;
        break;
    case 2:
        next_arg = arg2;
        break;
}

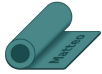
if(*word == '\0') // Wenn das Ende des Strings erreicht ist
{
    return;
}
if(!TX_Full())
{
    if(*word == '\\' && *(word+1) == 'n') // Linefeed erkennen
    {
        UART_Transmit_Byte(10);
        word += 2;
    }
    else if(*word == '%' && arg_count < 3) // % symbolisiert
    die Stelle an der Daten eingefügt werden
    {
        word++;

        if(*word == 'd') // d symbolisiert, dass ein Integer
        eingefügt wird
        {
```



```
        uint8_t arg_to_str[6];
        uint16_ascii(next_arg, arg_to_str); // Daten in
einen String umwandeln
        int i = 0;
        NOP();
        while(arg_to_str[i] != '\0')
        {
            UART_Transmit_Byte(arg_to_str[i]);
            i++;
        }
        arg_count++;
        word++;
    }
}
else
{
    UART_Transmit_Byte(*word);
    word++;
}
}
}
```

```
void UART_Transmit_Word(uint8_t *word) // Sendet einen String
{
    while(true)
    {
        if(*word == '\0') // Wenn das Ende des Strings erreicht ist
        {
            return;
        }
    }
}
```

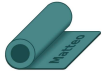


```
        if(!TX_Full())
        {
            if(*word == '\\\\' && *(word + 1) == 'n') // Linefeed
erkennen
            {
                UART_Transmit_Byte(10);
                word += 2;
            }
            else
            {
                UART_Transmit_Byte(*word); // String übertragen
                word++;
            }
        }
    }
}

// ADC
void read_Sensor(uint8_t Address) // speichert den Wert eines
bestimmten Sensors
{
    LATAbits.LATA0 = (0b00000001 & Address) ? 1 : 0; // MUX-
Steuerleitungen umschalten um bestimmten Sensor anzusprechen
    LATAbits.LATA1 = (0b00000010 & Address) ? 1 : 0;
    LATAbits.LATA2 = (0b00000100 & Address) ? 1 : 0;
    LATAbits.LATA3 = (0b00001000 & Address) ? 1 : 0;

    ADCON0bits.GO_nDONE = 1; // ADC Wandlung starten

    while(ADCON0bits.GO_nDONE == 1) // Warten bis die Wandlung
abgeschlossen ist
    {
        NOP();
    }
}
```



```
}

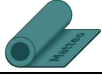
    ADCres = ((uint16_t)ADRESH << 8) | ADRESL; // ADC-Ergebnis
Registerpaar für Debuggingzwecke speichern
    ADCresH = ADRESH;
    ADCresL = ADRESL;

    Sensor_Value_raw[Address] = 1023 - ADCres;           // Rohdaten
speichern
    Sensor_Value[Address] = sqrt(Sensor_Value_raw[Address]); //
Linearisierte Daten speichern
}

void read_all_Sensor(void) // speichert die Werte aller Sensoren
{
    Sensor_avg = 0;
    for(uint8_t i = 0; i < Sensor_Amount; i++)
    {
        read_Sensor(i);
        Sensor_avg += Sensor_Value[i];
    }
    Sensor_avg /= Sensor_Amount;
}

// Befehlsverarbeitung
void execute_Instruction(void) // Führt die empfangenen Befehle aus
{
    uint8_t Inst[max_Inst_Length] = {0, 0, 0, 0, 0, 0};
    uint8_t i = 0;
    bool    Inst_complete = false;

    if(RX_Empty()) {return;} // Wenn kein Befehl empfangen wurde muss
auch keiner ausgeführt werden
```

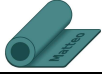


```
while(RX_Buffer[RX_head % RX_Buffer_Size] != '>') // ein Befehl
beginnt immer mit >
{
    RX_head++;
    if(RX_Empty()) {return;}
}
while(!Inst_complete)
{
    RCIE = 0;
    if(RX_Empty() || i == max_Inst_Length) {RX_Error = true; RCIE
= 1; return;}
    // wurde kein Linefeed empfangen und doch die maximale
    Befehlslänge erreicht, ist ein Fehler aufgetreten
    //if(i == max_Inst_Length) {RX_Error = true; return;}

    else if(RX_Buffer[RX_head % RX_Buffer_Size] == 0x0A) // Jeder
    Befehl endet mit einem Linefeed
    {
        Inst_complete = true;
        UART_Transmit_Byte('<'); // Nach einem Befehl im richtigen
        Format wird ein ACK gesendet

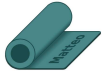
    }
    Inst[i] = RX_Buffer[RX_head % RX_Buffer_Size];
    RX_head++;
    i++;
}
RCIE = 1;

// Da der Befehl aus Daten besteht und somit variiert, muss auf
diese Weise überprüft werden
if(Inst[0] == KF[0] && Inst[1] == KF[1] && Inst[2] == KF[2] &&
Inst[4] == KF[4] && Inst[5] == KF[5])
```



```
{
    Target = true;
    NOP();
    RFt = (Inst[3] - '0') * 10;
    return;
}

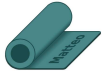
else if(memcmp(Inst, KN, max_Inst_Length) == 0) // In allen anderen
Fällen wird mit memcmp gearbeitet
{
    Target = false; // Übung ohne Ziel
}
else if(memcmp(Inst, KON, max_Inst_Length) == 0)
{
    exercise_run = true; // Übung gestartet
    return;
}
else if(memcmp(Inst, KOFF, max_Inst_Length) == 0)
{
    exercise_run = false; // Übung gestoppt
    return;
}
else if(memcmp(Inst, iE, max_Inst_Length) == 0)
{
    resend_Inst = true; // Error empfangen, der zuletzt gesendete
Befehl wird erneut gesendet
    return;
}
else if(memcmp(Inst, iP, max_Inst_Length) == 0)
{
    UART_Transmit_Word(oPA); // auf ein Ping antworten
    return;
}
```



```
    else if(memcmp(Inst, iPA, max_Inst_Length) == 0)
    {
        ESP_available = true; // Wurde eine Antwort auf einen Ping
        empfangen, ist der ESP verfügbar
        return;
    }
    else
    {
        RX_Error = true; // Hat der Befehl das korrekte Format, ist
        aber unbekannt, wird dies als Übertragungsfehler gesehen
        return;
    }
}

void send_Instruction(uint8_t Inst[max_Inst_Length]) // Diese Funktion
sendet Befehle
{
    UART_Transmit_Word(Inst);
    memcpy(last_outgoing_Inst, Inst, max_Inst_Length); // Der zuletzt
    gesendete Befehl wird gespeichert
    for(int i = 0; i <= 2000; i++) // es wird 2s auf ein ACK gewartet
    {
        if(ACK)
        {
            ACK = !ACK;
            NOP();
            return;
        }
        __delay_ms(1);
    }

    NOP();
```

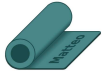
```
    ESP_available = false; // Wird kein ACK empfangen, ist der ESP
    nicht verfügbar
    resend_Inst = true;
    while(!ESP_available)
    {
        execute_Instruction();
        if(t - t_prev_Ping >= t_send_Ping)
        {
            NOP();

            t_prev_Ping = t;
            UART_Transmit_Word(oP); // Der ESP wird solange angepingt
bis dieser antwortet
        }
    }
    ESP_available = true;
}
```

```
uint8_t Standardabweichung(uint8_t *Data, uint8_t size, bool debug)
{
    /*
    * Die Standardabweichung wird mit folgender Formel berechnet:
    *  $\sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2}$ 
    */

    uint16_t avg = 0;

    for(int i = 0; i < size; i++)
    {
        avg += Data[i];
    }
    avg /= size;
```



```
uint8_t SA = 0;

for(int i = 0; i < size; i++)
{
    SA += ((Data[i] - avg) * (Data[i] - avg));
}
SA /= size;
SA = sqrt(SA);
NOP();
if(debug)
{
    UART_Transmit_Data("SA:  %d\nAVG:  %d\n-----
\n", SA, avg, 0);
}
return SA;

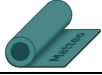
}

void S_avg_valid(uint8_t *avg, uint8_t *Data, uint8_t size)
{
    uint8_t SA = Standardabweichung(Data, size, 0);

    if(*avg <= 3)
    {
        *avg = 0;
        return;
    }
    else {return;}
}
```



```
// main -----  
-----  
  
int main(void)  
{  
    SYSTEM_Initialize();  
    GPIO_init();  
    UART_init();  
    ADC_init();  
    TMR1_init();  
  
    while(!debug)  
    {  
        execute_Instruction();  
  
        if(resend_Inst)  
        {  
            resend_Inst = false;  
            send_Instruction(last_outgoing_Inst);  
        }  
  
        if(RX_Error)  
        {  
            RX_Error = false;  
            send_Instruction(oE);  
        }  
  
        if(!ESP_available)  
        {  
            send_Instruction(oP);  
            NOP();  
        }  
    }  
}
```



```
if(exercise_run)
{
    NOP();
    RF_Sensor_avg = 0;
    LF_Sensor_avg = 0;
    read_all_Sensor();

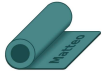
    for(int i = 0; i < 6; i++)
    {
        RF_Sensor_avg += Sensor_Value[i];
        LF_Sensor_avg += Sensor_Value[Sensor_Amount - (i +
1)];
    }

    RF_Sensor_avg /= Sensor_Amount / 2;
    LF_Sensor_avg /= Sensor_Amount / 2;

    S_avg_valid(&RF_Sensor_avg, Sensor_Value,
Sensor_Amount/2);
    S_avg_valid(&LF_Sensor_avg, Sensor_Value + 6,
Sensor_Amount/2);

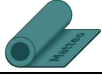
    RFc = (RF_Sensor_avg * 100) / (RF_Sensor_avg +
LF_Sensor_avg);
    RFc = round5(RFc);
    NOP();
    RF_Sensor_avg = 0;
    LF_Sensor_avg = 0;

    if((RFc == RFt) && exercise_run && Target &&
!Target_reached)
    {
```

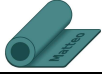


```
        NOP();
        Target_reached = true;
        hold_sec_count = 1;
        t_Target_complete = t;
    }
    else if((RFc >= (RFt - Tol)) && (RFc <= (RFt + Tol)) &&
exercise_run && Target)
    {
        RFc_inside_Tolerance = true;
    }
    else if(exercise_run && Target)
    {
        RFc_inside_Tolerance = false;
    }

    if(Target_reached && (t - t_Target_complete >=
t_hold_Target) && Target && exercise_run)
    {
        send_Instruction(GR);
        Target_reached = false;
        exercise_run = false;
        hold_sec_count = 0;
    }
    else if(Target_reached && (t - t_Target_complete >=
hold_sec_count) && Target && exercise_run)
    {
        hold_sec_count++;
        if(!RFc_inside_Tolerance)
        {
            Target_reached = false;
            hold_sec_count = 0;
        }
    }
}
```

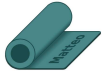


```
if((t - t_prev_RFc >= t_send_RFc) && exercise_run)
{
    NOP();
    D[D_data_index] = RFc + 1;
    send_Instruction(D);
    NOP();
    t_prev_RFc = t;
}
}
}
}
```

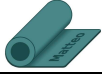


Programmcode Arduino Nicla Voice

```
//Bibliotheken-----  
  
#include "NDP.h"  
#include <Wire.h>  
  
//Daten an den ESP32 senden-----  
  
void sendToESP(char type, int value) {  
    NDP.noInterrupts();  
    Serial.print(type);  
    Serial.println(value);  
    Wire.beginTransmission(9);  
    Wire.write(type);  
    Wire.write(value);  
    Wire.endTransmission();  
    NDP.interrupts();  
}  
  
//Callback-Funktion für erkannte Befehle-----  
  
void sendit(char* label) {  
    Serial.println(label);  
    nicla::leds.begin();  
    nicla::leds.setColor(green);  
  
    if (strcmp(label, "NN0:10_Prozent") == 0) {  
        sendToESP('P', 1);  
    }  
    else if (strcmp(label, "NN0:20_Prozent") == 0) {  
        sendToESP('P', 2);  
    }  
    else if (strcmp(label, "NN0:30_Prozent") == 0) {  
        sendToESP('P', 3);  
    }  
    else if (strcmp(label, "NN0:40_Prozent") == 0) {
```



```
    sendToESP('P', 4);
}
else if (strcmp(label, "NN0:50_Prozent") == 0) {
    sendToESP('P', 5);
}
else if (strcmp(label, "NN0:60_Prozent") == 0) {
    sendToESP('P', 6);
}
else if (strcmp(label, "NN0:70_Prozent") == 0) {
    sendToESP('P', 7);
}
else if (strcmp(label, "NN0:80_Prozent") == 0) {
    sendToESP('P', 8);
}
else if (strcmp(label, "NN0:90_Prozent") == 0) {
    sendToESP('P', 9);
}
else if (strcmp(label, "NN0:100_Prozent") == 0) {
    sendToESP('P', 10);
}
else if (strcmp(label, "NN0:Start") == 0) {
    sendToESP('S', 1);
}
else if (strcmp(label, "NN0:Stopp") == 0) {
    sendToESP('S', 0);
}
else if (strcmp(label, "NN0:Linkes_Bein") == 0) {
    sendToESP('B', 1);
}
else if (strcmp(label, "NN0:Rechtes_Bein") == 0) {
    sendToESP('B', 0);
}
else if (strcmp(label, "NN0:Lauter") == 0) {
    sendToESP('L', 1);
}
```

```
else if (strcmp(label, "NN0:Leiser") == 0) {
    sendToESP('L', 0);
}
else if (strcmp(label, "NN0:Kein_Ziel") == 0) {
    sendToESP('Z', 0);
}
//delay(100);
nicla::leds.end();
}

//Setup-----

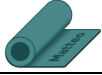
void setup() {
    Wire.begin(); //I2C initialisieren
    Serial.begin(115200); //Serial Monitor initialisieren

    // Nicla System setup-----
    nicla::begin();
    nicla::leds.begin();
    nicla::leds.setColor(blue);

    //Neural Decision Processor callbacks setup-----
    //NDP.onError(ledRedBlink);
    NDP.onMatch(sendit);

    //Neural Decision Processor firmware und ML-Model laden-----
    Serial.println("Loading synpackages");
    NDP.begin("mcu_fw_120_v91.synpkg");
    NDP.load("dsp_firmware_v91.synpkg");
    NDP.load("ei_model.synpkg");
    Serial.println("packages loaded");
    NDP.getInfo();

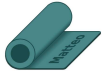
    //Mikrofon initialisieren-----
    Serial.println("Configure mic");
```



```
NDP.turnOnMicrophone();

nicla::leds.end();
sendToESP('H', 0); //ESP melden, dass Arduino hochgefahren ist
}

void loop() {}
```



Programmcode ESP32:

```
//Bibliotheken-----  
  
#include <Wire.h>  
#include "Arduino.h"  
#include "SD.h"  
#include "AudioFileSourceSD.h"  
#include "AudioGeneratorWAV.h"  
#include "AudioOutputI2S.h"  
  
//PINs für die SD-Karte-----  
  
#define SD_CS 4  
#define SPI_MOSI 23  
#define SPI_MISO 19  
#define SPI_SCK 18  
  
//PINs für den Audioverstärker-----  
  
#define I2S_DOUT 27  
#define I2S_BCLK 26  
#define I2S_LRC 25  
  
//I2C PINs-----  
  
#define SDA 21  
#define SCL 22  
#define I2C_ADDRESS 9  
  
//UART Pins-----  
  
#define RX0 16  
#define TX0 17  
  
//LED Pins-----  
  
#define LED_red 2  
#define LED_green 33  
  
//Ringbuffer-Variablen-----  
  
#define speaker_buffer_size 10
```



```
unsigned int speaker_head;
unsigned int speaker_tail;
String speaker_buffer[speaker_buffer_size];

//Variablen für das Einlesen vom Arduino-----
char type = 0;
int value = 0;

//Variablen für das Einlesen vom PIC-----
static char bein;
char PIC_Data[6];
int i = 0;
bool is_started = false;

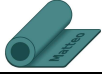
//Variablen für Acknowledge-----
unsigned long t_act = 0;
int t_ack_wait = 2000;
unsigned long t_ack = 0;

//Variablen für Ping-----
unsigned long t_ping = 0;
int t_ping_wait = 1000;

//Variablen für Sprachausgabe-----
int t_ausgabe_wait = 2000;
unsigned long t_ausgabe = 0;

//Letzter gesendeter Befehl-----
String last_sent = "";

//Variablen für switch-----
enum datatypes{Prozent, Start_Stopp, Bein, Lautstaerke, kein_Ziel, None,
Hochfahren, Ausgabe};
static datatypes Datatype = None;
```



```
//Variablen für I2S-----
AudioGeneratorWAV *wav;
AudioFileSourceSD *file;
AudioOutputI2S *out;

float gain = 0.3;

//Char zu Enum datatypes-----

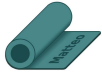
datatypes charToDatatype(char c) {
    switch (c) {
        case 'P': return Prozent;
        case 'S': return Start_Stopp;
        case 'B': return Bein;
        case 'L': return Lautstaerke;
        case 'Z': return kein_Ziel;
        case 'H': return Hochfahren;
        default: return None;
    }
}

//Daten vom PIC empfangen-----

void receiveEvent(int len) {

    if(Wire.available() && t_act-t_ausgabe >= t_ausgabe_wait) { //Wenn Daten
    vorhanden und keine Sprachausgabe aktiv einlesen
        type = Wire.read();
        value = Wire.read();
        Datatype = charToDatatype(type);
    }
}

//Ping senden-----
```

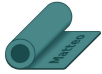


```
void ping() {
    while(1) {
        while (!Serial1.available()) { //solange keine Antwort erhalten wurde
            jede Sekunde Ping senden
            t_act = millis();
            if(t_act-t_ping >= t_ping_wait) {
                Serial1.write(">P\n");
                t_ping = t_act;
            }
        }

        char ping[5];
        for (int n = 0; n<4;) { //vier Zeichen einlesen wenn vorhanden
            if(Serial1.available()) {
                ping[n] = Serial1.read();
                n++;
            }
            t_act = millis();
            if(t_act-t_ping >= 5000) {
                t_ping = t_act;
                break;
            }
        }

        ping[4] = 0;
        if(strcmp(ping, "<>PA") == 0 ) { //Wenn Ping-Antwort erhalten Funktion
            verlassen
            Serial1.write("<");
            return;
        }
    }
}

//Daten an PIC senden-----
```

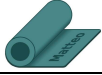


```
void sendToPIC(String befehl) {
    last_sent = befehl;
    const char* send = befehl.c_str();
    Serial1.write(send);
    Serial.print(befehl);

    while(1) {
        t_ack = t_act;
        while(!Serial1.available()) { //auf Acknowledge warten
            t_act = millis();
            if(t_act - t_ack >= t_ack_wait) { //Ping Funktion aufrufen falls kein
Acknowledge
                ping();
                Serial1.write(send);
                break;
            }
        }
        char ack = Serial1.read();
        if(ack == '<') { //Senden erfolgreich wenn Acknowledge empfangen
            return;
        }
    }
}

//Datei in Ringbuffer schreiben-----

void write_buffer (String Ausgabe) {
    if(speaker_tail - speaker_head != speaker_buffer_size) { //Wenn Ringbuf-
fer nicht voll, Datei in Buffer schreiben
        speaker_buffer[speaker_tail % speaker_buffer_size] = Ausgabe;
        speaker_tail++;
    }
}
```



```
//setup-----

void setup() {
  Serial.begin(115200);
  Serial1.begin(57600, SERIAL_8N1, RX0, TX0); //UART initialisieren
  delay(1000); // Testen ob notwendig

  Wire.begin((uint8_t)I2C_ADDRESS, SDA, SCL, 100000); //I2C initialisieren
  Wire.onReceive(receiveEvent);

  pinMode(SD_CS, OUTPUT); //Clock Select Leitung für SPI auf HIGH
  digitalWrite(SD_CS, HIGH);

  SPI.begin(SPI_SCK, SPI_MISO, SPI_MOSI); //SPI initialisieren

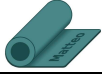
  if(!SD.begin(SD_CS)) //Überprüfen ob SD-Karte vorhanden ist
  {
    Serial.println("Error accessing microSD card!");
    while(true);
  }

  out = new AudioOutputI2S(); //I2S initialisieren
  out->SetPinout(I2S_BCLK, I2S_LRC, I2S_DOUT);
  out->SetGain(gain);

  wav = new AudioGeneratorWAV();

  pinMode(LED_green, OUTPUT); //LED-Pins auf OUTPUT und LOW
  pinMode(LED_red, OUTPUT);
  digitalWrite(LED_green, LOW);
  digitalWrite(LED_red, LOW);
}

//main-----
```

```
void loop() {

    t_act = millis();

    //Sprachausgabe laufen lassen-----

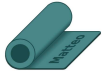
    if (wav && wav->isRunning()) { //WAV-Datei läuft bis ende

        if (!wav->loop()) { //Wenn beendet file löschen
            wav->stop();
            if (file) {
                delete file;
                file = nullptr;
            }
        }
    }

    //Sprachausgabe starten-----

    else {
        if (speaker_head-speaker_tail != 0) {
            const char* Ausgabe = speaker_buffer[speaker_head % speaker_buffer_size].c_str(); //neue Datei öffnen
            file = new AudioSourceSD (Ausgabe);
            speaker_head++;

            if (file->isOpen()) { //Überprüfen ob Datei offen
                wav->begin(file, out); //Abspielen starten
                t_ausgabe = t_act;
            }
            else {
                Serial.println("Konnte nicht geöffnet werden!"); //Wenn Datei nicht
                //geöffnet Fehlermeldung
                delete file;
                file = nullptr;
            }
        }
    }
}
```

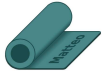


```
}  
}  
}
```

```
switch(Datatype) {
```

```
//Verarbeitet Prozent Einstellungen-----
```

```
case Prozent: {  
    if(!is_started) { //nur wenn Übung noch nicht gestartet  
        int ziel = value;  
        int ziel_r = value;  
  
        if(bein == 'l') { //wenn linkes Bein eingestellt ist auf Rechtes            ziel_r = 10 - ziel;  
        }  
  
        if(bein == 'r' || bein == 'l') {  
            String befehl = ">KF";  
            befehl += ziel_r;  
            befehl += "\n";  
            sendToPIC(befehl);  
  
            String Ziel_Ausgabe = "/";  
            Ziel_Ausgabe += ziel*10;  
  
            if(bein == 'l') { //Zieleinstellung für linkes Bein ausgeben  
                Ziel_Ausgabe += "_Prozent_links.wav";  
            }  
  
            if(bein == 'r') { //Zieleinstellung für rechtes Bein ausgeben  
                Ziel_Ausgabe += "_Prozent_rechts.wav";  
            }  
            write_buffer(Ziel_Ausgabe) ;  
        }  
    }  
}
```



```
    }
}
bein = 0;
Datatype = None;
break; }

//Startet und stoppt die Übung-----

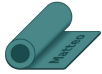
case Start_Stopp: {
    int start_stopp = value;
    if(start_stopp == 1 && !is_started) { //Übung startet wenn noch
nicht gestartet wurde
        sendToPIC(">KON\n");
        digitalWrite(LED_green, LOW);
        digitalWrite(LED_red, HIGH);
        is_started = true;
        write_buffer("/Start.wav");
    }
    else if(start_stopp == 0 && is_started) { //Übung stoppt wenn sie
läuft
        sendToPIC(">KOFF\n");
        digitalWrite(LED_red, LOW);
        is_started = false;
        write_buffer("/Stopp.wav");
    }

    Datatype = None;
    break; }

//Einstellung für welches Bein-----

case Bein: {
    if(!is_started) { //nur wenn Übung nicht gestartet

        if(value == 0) { //setzt Bein auf rechts
```



```
        bein = 'r';
    }
    else if(value == 1) { //setzt Bein auf links
        bein = 'l';
    }
}
Datatype = None;
break; }
```

//Lautstärke einstellen-----

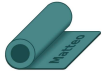
```
case Lautstaerke: {
    if(value == 0) { //Verstärkung wird bei Leiser um 0.2 verringert
        if(gain >= 0.2) {
            gain -= 0.2;
            out->SetGain(gain);
        }
        write_buffer("/Leiser.wav");
    }
}
```

```
if(value == 1) { //Verstärkung wird bei Lauter um 0.2 erhöht
    gain += 0.2;
    out->SetGain(gain);
    write_buffer("/Lauter.wav");
}
```

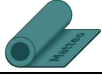
```
Datatype = None;
break; }
```

//Eistellung für Messung ohne Ziel-----

```
case kein_Ziel: {
    if(!is_started) { //funktioniert nur wenn Übung nicht gestartet
        sendToPIC(">KN\n");
        write_buffer("/Kein_Ziel.wav"); //Befehl zur Bestätigung ausgeben
    }
}
```



```
    }  
    Datatype = None;  
    break; }  
  
//Meldung wenn PIC hochgefahren wurde-----  
  
    case Hochfahren: {  
        write_buffer("/Hochfahren.wav");  
        Datatype = None;  
        break; }  
  
//Einlesen der Daten vom PIC-----  
  
    case None: {  
        while(Serial1.available()) { //Wenn Daten vorhanden sind wird einge-  
        lesen  
            PIC_Data[i] = Serial1.read(); //Daten lesen und in Array speichern  
            if(PIC_Data[0] != '>') { //Wenn erstes Zeichen nicht '>' einlesen  
abbrechen  
                PIC_Data[0] = 0;  
                i = 0;  
                break;  
            }  
  
            if(PIC_Data[i] == 10) { //Linefeed beendet Befehl  
                PIC_Data[i] = 0; //Linefeed in Nullterminierung für String umwan-  
deln  
  
                Datatype = Ausgabe;  
                i = 0;  
                break;  
            }  
            i++;  
        }  
        break; }
```



//Sprachausgabe der Daten vom PIC-----

```
case Ausgabe: {

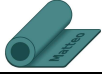
    if(strcmp(PIC_Data, ">GR") == 0) { //Ziel erreicht ausgeben
        write_buffer("/Ziel_erreicht.wav");
        Serial1.write("<");
        digitalWrite(LED_red, LOW);
        digitalWrite(LED_green, HIGH);
        is_started = false;
    }

    else if(PIC_Data[1] == 'D') { // aktuelle Belastung ausgeben
        if(PIC_Data[2] == 0) { //Error senden wenn keine Belastung empfan-
gen wurde
            Serial1.write(">E\n");
        }

        else {
            String Belastung_Ausgabe = "/";
            Belastung_Ausgabe += PIC_Data[2]-1;
            Belastung_Ausgabe += "_Prozent.wav";

            write_buffer(Belastung_Ausgabe);
            Serial1.write("<");
        }
    }

    else if(strcmp(PIC_Data, ">E") == 0) { //Bei Errormeldung letzten Be-
fehl noch einmal senden
        Serial1.write("<");
        sendToPIC(last_sent);
    }
}
```



```
else if(strcmp(PIC_Data, ">P") == 0) { //Ping beantworten
    sendToPIC("<>PA\n");
}

else { //Wenn ungültiger Befehl Error senden
    Serial1.write("<");
    sendToPIC(">E\n");
}

for(int n = 0; n <= i-1; n++) { //Array wieder auf 0 setzen
    PIC_Data[n] = 0;
}

Datatype = None;

break; }
}
}
```