HERO

# Kurzfassung

HERO ist ein selbstentwickelter Roboterhund in der Lage sehbehinderten Personen beim Navigieren zu helfen.

Mit HERO haben wir uns das Zeil gesetzt Menschen zu helfen und sie bei alltäglichen Situationen zu unterstützen. Dafür haben wir den Roboter selbst Design, gebaut und programmiert.

Zentrale Themen unseres Projektes sind:

* Bewegung des Hundes
* autonomes Laufen
* mit Kameras und Sensoren die Umgebung wahrnehmen
* auf Veränderung in der Umgebung reagieren und eigenständig Entscheidungen zu treffen.

BILD

Inhalt

[Kurzfassung 2](#_Toc187330586)

[1. Einführung 4](#_Toc187330587)

[1.1 Ideenfindung 4](#_Toc187330588)

[1.2 Ideen zur Umsetzung 4](#_Toc187330589)

[1.3 Fragestellung 5](#_Toc187330590)

[2. Vorgang, Methode, Material 5](#_Toc187330591)

[2.1 Design des Roboters 5](#_Toc187330592)

[2.1.2 Die Beine 5](#_Toc187330593)

[2.2 Elektronik und Hardware 6](#_Toc187330594)

[2.3 Kinematik 7](#_Toc187330595)

[2.4 Gehbewegung 7](#_Toc187330596)

[2.5 Kommunikation mit dem Nutzer 8](#_Toc187330597)

# 1. Einführung

## 1.1 Ideenfindung

Wir kamen auf unsere Idee einen Roboterhund zu bauen bei der Teilnahme and der Jugend-Forscht AG am Gymnasium Hochrad. Ein Teilnehmer unseres Teams hat bereits Erfahrung mit Robotik, so entschieden wir uns einen Roboter zu bauen.

Nur was für einen? Wir bekamen mit wie Nachbarn von uns, sie sind blind, mitten auf der Straße gingen, da es der einfachste Weg für sie ist einem bestimmten Weg zu folgen - nicht gerade das sicherste Unternehmen.

So beschlossen wir einen Roboter zu bauen, der das Navigieren sehbehinderter Personen zu erleichtern.

## 1.2 Ideen zur Umsetzung

Um die Navigation im Freien zu vereinfachen, entwickelten wir ein System, das mithilfe von Sensoren und Kameras die Umgebung detailliert erfasst und diese Daten zur Planung von Aktionen verwendet. Ziel war es, der Person basierend auf den vom Roboter gesammelten Informationen Anweisungen zu geben, um einen reibungslosen Ablauf zu gewährleisten.

Was uns noch fehlte, war die Kommunikation zwischen Menschen und Roboter. Der Roboterhund sollte zudem in der Lage sein, haptische Signale zu nutzen, um dem Nutzer Richtungsänderungen, Hindernisse und das Erreichen von Zielen mitzuteilen. Durch den Einsatz von Mikrofonen, zusätzlichen Sensoren, einem Computer und Lautsprechern konnten wir auch dieses Problem lösen.

Die Person trägt während des Ablaufs ein Armband, das ein eingebautes Mikrofon, einen Computer und einen Motor enthält.

* Das Mikrofon ermöglicht es der Person, dem Roboter Anweisungen zu geben, um an ihr gewünschtes Ziel zu gelangen.
* Der Computer sorgt dafür, dass die Kommunikation zwischen der Person und dem Roboter über ein Kommunikationssystem ablaufen kann.
* Der Motor wird verwendet, um der Person Signale zu senden, zum Beispiel wie weit sie sich drehen muss, um sicher im vorgesehenen Bereich und auf dem Gehweg zu bleiben.
* Die Sensoren des Roboters erfassen die Person hinter ihm und senden ihr Anweisungen, wie sie sich bewegen muss, um im sichern Bereich zu bleiben.

Für die Navigation an unterschiedlichen Orten ist der Roboterhund mit einem GPS-Modul ausgestattet. Dieses ermöglicht es ihm, seine aktuelle Position präzise zu bestimmen und Routen zu definierten Zielen zu berechnen. Das integrierte GPS-System erleichtert nicht nur die Orientierung in bekannten Umgebungen, sondern ermöglicht auch die problemlose Navigation in völlig neuen und unbekannten Gebieten. Durch die Verbindung mit Navigationsdiensten kann der Roboterhund zudem problemlos auf Veränderungen in der Umgebung reagieren, beispielsweise auf Baustellen oder Straßensperrungen, und alternative Routen vorschlagen.

## 1.3 Fragestellung

Können Roboter in eine gute Alternative zum konventionellen Blindenstock sein?

# 2. Vorgang, Methode, Material

## 2.1 Design des Roboters

Für das Model und Design des Roboters entschieden wir alle Teile von Grund auf selbst zu entwickeln, um die gesamte Kontrolle über das Projekt zu haben.

Wir fingen an mit Stift und Papier grobe Skizzen des Hundes zu erstellen, um Ideen zum Design zu finden. Die nächsten zwei Wochen verbrachen wir daran mit dem CAD (Computer-Aided Design) Programm Solidworks ein exaktes 3D Modell des Roboters zu erstellen.

Davor erstellten wir Kriterien, die die Mechanik erfüllen muss.

* Alle Teile müssen so geformt sein, dass sie einfach mit einem 3D-Drucker von zu Hause hergestellt werden können, um die Kosten des Roboters niedrig zu halten.
* Der Roboter sollte genug Platz für große Batterien bieten, um eine möglichst lange Batterielaufzeit zu ermöglichen.
* Jedes Teil sollte einfach zu erreichen sein, um spätere Aufrüstungen einfach zu machen.
* Platz für alle möglichen Sensoren und Kameras muss gegeben sein.

### 2.1.2 Die Beine

Das wichtige eines Navigierenden Roboters ist die Fortbewegung, in diesem Fall also die Beine.

Sie müssen robust sein, um Stürze und kraftvolle Fremdeinwirkungen standzuhalten.

Beim 3D-Drucken ist es wichtig auf die Druckorientierung zu achten. Fürs Drucken wird jedes Teil, welche aus einem STL-Mesh bestehen, von einem Programm in viele Schichten „zerschnitten“, die dann von 3D-Drucker übereinander gedruckt werden.

Die Übergänge zwischen den Druckschichten sind Schwachpunkte in dem jeweiligen Teil.

Alle Teile der Beine müssen also von Drucker horizontal und nicht vertikal gedruckt werden, um das Bein widerstandsfähiger zu machen.

Jedes Bein wird von drei Servos angetrieben, diese werden häufig beim Modellbau benutzt. Wir entschieden uns Servos mit einem Drehmoment von 40kg/cm zu nutzen, um das Gewicht des Hundes halten zu können.

Jeder Motor wiegt um die 70g. Für den Unterschenkel eines Beines entschieden wir uns dazu den Motor nicht an der Rotationsachse des Kniegelenkes zu montieren, sondern die Masse des Motors möglichst nahe an das Obere Gelenk zu befestigen, um das Trägheitsmoment des Beines zu verringern. So kann das Bein schneller und dabei mit weniger Drehmoment der Motoren sich bewegen. Stellen sie sich vor sie schnüren Gewichte an ihre Schuhe und versuchen dann schnell zu gehen.

Die „Schienbeine des Hundes haben eine Krümmung, um sich einfacher zu verbiegen (ohne zu brechen) und somit Stürze besser abzudämpfen.

Ein Bild, das Büroausstattung, Stift, Schreibwaren enthält.

Automatisch generierte BeschreibungEin Bild, das Screenshot, Zahnbürste, Pinsel, Werkzeug enthält.

Automatisch generierte Beschreibung

Am Anfang haben wir die Füße auch mit dem 3D-Drucker gedruckt. Für den ganzen Roboter wurde ausschließlich PLA benutzt, weil es sehr fest und günstig ist. Noch dazu ist es sehr einfach zu drucken, hat allerdings auch einen Nachteil. Wir haben festgestellt, dass die harten Plastikenden der Beine keinerlei Grip hatten, und auf allen Oberflächen einfach nur rumgerutscht sind. Wenn sich die Beine bewegten, die Füße aber nur über den Untergrund rutschten, bewegte sich der Hund nicht fort.

Die Lösung schien uns klar zu sein. Wir müssen die Füße aus irgendeinem weichen Gummi machen. Unser erster Prototyp sind in zwei geteilte Flummis gewesen, die mit Sekundenkleber and die Beine geklebt wurden.

Nicht nur sieht das nicht gut aus, war aber auch nicht gut. Die Flummis in der Hälfte zu zerschneiden war schwierig und zeitaufwendig. Außerdem hatten sie nicht genug Haftung und vielen einfach von den Beinen ab.

Unsere letzte Idee ist, dass wir die Füße aus Gummi selbst herstellen. Mit dem 3D-Drucker haben wir Schalen gedruckt, in die wir flüssiges Silikon gießen konnte, und so die Beine mit einer Gummischicht umschließen konnten. Dies hat sehr gut funktioniert. Die Beine hatten am Ende einen griffigen Gummiteil, der der Form des Beines folgt und sehr rutschfest ist.

Bild

## 2.2 Elektronik und Hardware

Der Roboter erkennt automatisch mithilfe von Kameras Objekte und Aktionen, braucht daher einen Leistungsfähigen Computer an Board, um den Video-Feed zu verarbeiten (mehr dazu ########).

Anfangs nutzten wir einen Raspberry Pi 4b, einen kleinen Computer oft benutzt für Robotik oder Smart-Home Systeme. Für seine Größe ist ein Raspberry Pi sehr leistungsfähig, so dachten wir wäre es eine gute Idee solch einen zu Nutzen.

Die Objekterkennung mit dem Raspberry Pi lief nicht gut. Selbst bei sehr schlechter Kamera Qualität schaffte es der Pi eine Objekterkennungssoftware nicht mit mehr als 5 Bildern die Sekunde zu laufen. Wir stiegen also um, auf einen 8x mal so teureren NVIDIA Jetson Orin Nano, ein sehr ähnliches Produkt mit sehr viel mehr Leistung. Der Jetson wurde uns kostenlos von MyBotShop.de zur Verfügung gestellt. Mit aktivierter Objekterkennung schafft dieser mindestens 15 Bilder die Sekunde.

Der Roboter besitzt fünf HC-SR04 Ultraschallsensoren, die Distanzen messen. Außerdem sind zwei Laser Distanzsensoren verbaut, jeweils einer vorne und hinten. Mit ihnen können z.B. Wände erkannt werden.

Eine Inertiale Messeinheit (ICM-20948) misst Beschleunigung in sechs Richtungen. Mit Hilfe eines Kalman-Filters werden Rotationswinkel und Himmelsrichtung ermittelt.

Alle Sensoren sind mit einem ESP-32 verbunden. Ein ESP-32 ist ein sehr kostengünstiger Mikrokontroller mit dem Sensoren sehr einfach auszulesen sind.

Der ESP32 kommuniziert mit dem UART-Serial Protokoll mit dem Jetson und sendet die Sensordaten.

## 2.3 Kinematik

Um Gehbewegungen mit den Beinen auszuführen, muss sich der Fuß jeden Beines frei in einem 3D Koordinatensystem bewegen können. Die Servo-Motoren lassen sich nur mit Winkeln steuern.

Um das Ende des Fußes millimetergenau im 3D-Raum bewegen zu können muss inversive Kinematik benutzt werden.

Gegeben sind drei Werte, X, Y und Z in einem virtuellen Koordinatensystem. Die Werte geben die Position des Fußes relativ zur der des ersten Gelenkes (0|0|0). Anhand dieser Werte müssen alle Winkel der Motoren berechnet werden, sodass der Fuß relativ vom Startpunkt X, Y und Z Millimeter weit entfernt ist.

Hierfür wurde einfache Trigonometrie mit Sinus- und Kosinussatz angewendet.

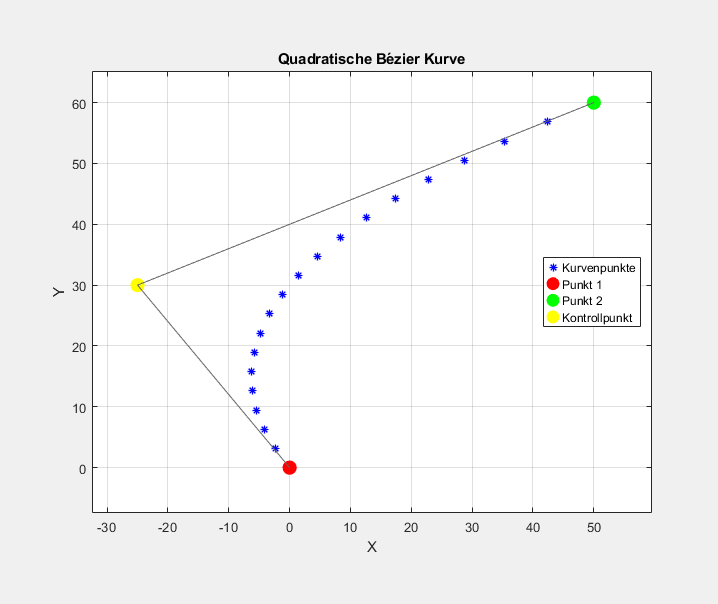
Am Ende erhält man drei Formeln, die bei eingesetzten X, Y und Z Werten drei Winkel entsprechen. Bewegt man das zugehörige Gelenk zu dem Winkel, befindet sich der Fuß im 3D-Raum relativ zum Startpunkt X, Y und Z Millimeter entfernt.

Anschließend wurde jedes Bein mithilfe der Python-Library Matplotlib simuliert.

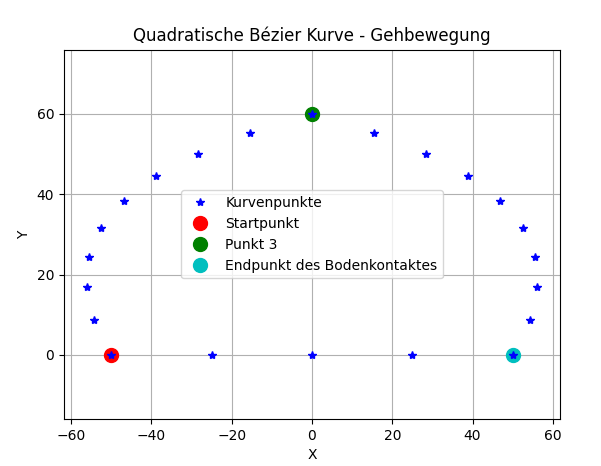
## 2.4 Gehbewegung

Mit der Kinematik ausgerechnet, kann sich jeder Fuß nun präzise im 3D-Raum bewegen. Damit sich der Roboter nach vorne bewegen kann, mussten wir zuerst eine Gehbewegung erstellen. Unsere erste Idee ist es gewesen, jedes Bein vier Punkte in Form eines Trapezes zu geben (siehe Abbildung). Das funktioniert zwar allerdings sind die Bewegungen sehr zackig und abrupt. Vor allem gegen Ende der Bodenberührung muss das Bein schnell in die komplett andere Richtung bewegen. Schon nach wenigen Durchläufen würden die Motoren aufgrund der schnellen Geschwindigkeits- und Richtungsänderung schaden nehmen. Wir haben uns online nach Funktionen angeguckt, die eine „weiche“ Kurve erstellen können.

Wir entschieden uns den Anfang und das Ende der Bodenberührung mit einer „Quadratischen Bezier-Kurve“ zu vollenden. Eine Quadratische Bezier-Kurve erstellt zwischen zwei Punkten P1 und P2 t-viele Punkte, deren Position von einem Kontrollpunkt gelenkt wird.

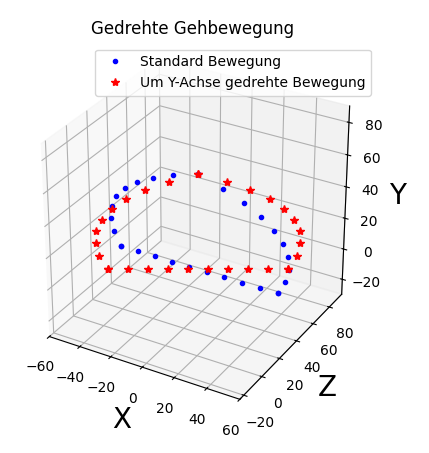


Eine zweite Kurve wird gespiegelt und flache Punkte hinzugefügt. Die Punkte auf Höhe 0 sind die, wo der Fuß jeweils den Boden berührt und den Roboter nach vorne „drückt“. Die Distanz der gespiegelten Bezier-Kurven ist die Schrittweite des Hundes und ist variabel.



## 2.4.2 Kurven

Damit der Roboter Kurven gehen kann, muss die Gehbewegung bei den Vorderbeinen gedreht werden. Jeder Punkt in der Gehbewegung befindet sich in einem 3D-Koordinatensystem. Wenn der Roboter eine Kurve gehen soll, berechnen wir für jeden Punkt in der Gehbewegung neue Koordinaten, die um den Winkel α um die Y-Achse gedreht sind. Dabei erhalten wir anhand von simplem Kalkulationen und den Ursprünglichen X und Z Werten neue X und neu Z Werte:



Nachdem wir die Punkte nun drehen konnten, muss jeder Punkt für jedes Bein neu berechnet werden. Dank unseres Jetson Nanos (und weil dies einfache Rechnungen sind) geschieht dies in Echtzeit.

Wenn wir den Roboter sagen, er solle nach links gehen, dann werden die Wegpunkte für die vorderen Beine neu berechnet.

Guckt man von oben auf den Bodenkontakt jedes Beins, sieht das beim Kurvenlaufen ungefähr so aus. Der Winkel der Vorderbeine variiert je nach Winkel α.

Ein Bild, das Reihe, Text, Screenshot, Diagramm enthält.

Automatisch generierte Beschreibung

Ein Bild, das Reihe, Design enthält.

Automatisch generierte Beschreibung

## 2.5 Kontrolle und Software

Die Softwareentwicklung ist der aufwendigste Teil unseres Projektes. Wir entschieden uns sie Schritt für Schritt anzugehen. Das Endziel unseres Projektes ist die autonome Navigation, komplett isoliert von externen Computern.

Um dieses Ziel schneller zu erreichen, ließen wir den autonomen Aspekt unseres Projektes erstmal aus und konzentrierten uns auf die wesentlichen Aspekte der Steuerung von einem Benutzter mithilfe eines GUIs (Graphical User Interface).

Die gesamte Software lässt sich aktuell in drei Bereiche aufteilen:

1. Code auf dem Mikrocontroller (ESP32)
2. Code und Firmware auf dem Jetson Orin Nano (unserem On-Board Computer)
3. Grafische Benutzeroberfläche.

### 2.5.2 Code des Mikrokontrollers

Unser Roboter besitzt viele Sensoren. Diese müssen ausgelesen und deren Daten teilweise verarbeitet werden. Wir nutzten nicht den Jetson Nano um Sensoren auszulesen, aus dem einfachen Grund, weil es mit einem ESP32 einfacher ist und bereits viele Code-Libraries existieren, die die Auslesung einfacher machen.

Wir benutzen eine Inertiale Messeinheit die Beschleunigung misst. Mithilfe eines Kalman-Filters werden Gyroskop- und Beschleunigunsdaten kombiniert und zukünftige Daten vorhergesagt. Mithilfe dieser Mathematischen Methode erhalten wir präzise Neigungswinkel in X- und Y-Achse.

Der Code beinhaltet eine Initialisierungsphase, in der alle Sensoren, gestartet initialisiert und geprüft werde. Darauf folgt eine nicht endende Schleife. In der Schleife werden alle Sensoren ausgelesen, formatiert und in einen String gepackt. Der String (Zeichenfolge) wird mit Hilfe des UART-Protokolls an den Jetson Orin Nano geschickt. Der Vorgang wiederholt sich so lange, bis der Roboter ausgeschaltet ist.

### 2.5.3 Firmware des Jetson

Die Firmware des Jetson haben wir in Python geschrieben und ist in verschiedene Teile aufgebaut, um das Debuggen und Wartung zu vereinfachen.

Roboter ist aktuell nicht autonom und kommuniziert noch mit einer Benutzerfläche auf einem externen Laptop. Wir benutzen Threading in unserem Code. Kurz gesagt ist der Code in Threads aufgeteilt. Threads sind Codeabschnitte, die parallel und asynchron auf jeweils einem CPU-Kern laufen.

Am Anfang werden startet der Jetson einen WLAN-Hotspot, mit dem sich der Computer mit der Benutzeroberfläche verbinden. Wir hosten gleichzeitig auch zwei TCP-Server in dem vom Jetson erstellten Netzwerk. Die Beiden TCP Verbindungen erlauben der Benutzeroberfläche Daten mit hoher Bandbreite zu empfangen und zu senden. Danach wird dauerhaft nach eintreffenden Befehlen geachtet. Trifft ein Befehl ein, wird dieser individuell verarbeitet.

Bei dem Befehl Kurven zu gehen, werden alle Wegpunkte jeder Beinbewegung neu berechnet. Je nach Einstellung besteht ein Gehzyklus aus 20 bis 50 Wegpunkten. Würde man dem Bein sagen, jedem dieser Wegpunkte zu folgen, wäre die Bewegung sehr „zackig“.

Mit Hilfe eines Interpolationsalgorithmus werden 10 bis 100 Punkte nach linearem Schema berechnet und zwischen die ursprünglichen Punkte gesetzt.

## 2.6 Kamerasystem

HERO ist bei Einreichung dieser Arbeit noch nicht in der Lage sich autonom fortzubewegen. Allerdings haben wir bereits mit dem Kamerasystem angefangen.

HERO benutzt eine kleine, nach vorne gerichtete Kamera. Das Kamerabild wird dann vom Jetson verarbeitet.

Ein Bild, das Text, Screenshot, Reihe, Straße enthält.

Automatisch generierte Beschreibung

Die oben abgebildeten Bilder sind nur Beispiele. Die Kameraperspektive wir im Betrieb anders sein und der Code dementsprechend angepasst.

Anhand dieser Linien können wir eine Mitte ziehen, die der Hund folgen soll. Unser oben gezeigte, vereinfachte Algorithmus funktioniert auch mit einem Video, oder im Betrieb mit dem echt zeit Kamerabild. Dafür haben wir eine Schleife erstellt, die bei jedem Durchlauf das nächste Bild eines Kamerabildes nimmt und die Linien neu ermittelt.

Aktuell arbeiten wir an einer Objekterkennung. HERO wird in der Lage sein Straßenschilder, Ampeln, Treppen und Autos zu erkennen und darauf Entscheidungen zu treffen. Steht eine Ampel auf Rot, so bleibt HERO stehen und gibt dies and den Nutzer weiter. Mit Hilfe der Open Source Library OpenCV und der YOLO-Architektur von Ultralytics lässt sich so etwas mit moderatem Aufwand umsetzen.

Das Kamerasystem ist auch der Grund gewesen, warum wir uns für einen NVIDIA Jetson Orin Nano anstelle des Raspberry Pis entschieden haben. Die Videoverarbeitung benötigt viel Rechenleistung, um das Kamerabild oft in einer Sekunde verarbeiten kann.

Berechnung

Wir haben berechnet, dass ein Auto, welches mit 30km/h 5 Meter vom Roboter, am Roboter vorbeifährt mit der Leistung des Jetson mehr als 4-mal so viel sichtbar ist. Uns ist aufgefallen, dass das sehr wichtig ist, da wenn ein Auto noch schneller fährt, es teilweise gar nicht vom Raspberry Pi erkannt werden kann, da das Bild der Kamera nur ein paarmal jede Sekunde verarbeitet werden kann. Es müssen also möglichst oft Bilder der Kamera verarbeitet werden, um schnell fahrende Autos erkennen zu können und der Roboter darauf reagiert (z.B. könnte der Roboter nur das Überqueren an Ampel zulassen oder seine Geschwindigkeit verringern, um das Navigieren sicherer zu gestalten

Für x = 3 (Raspberry Pi)

Für x = 12 (Jetson Orin Nano)

Wir nutzen aktuell eine Kamera mit einem schmalen Sichtfeld. Eine Kamera mit einem breiterem Sichtfeld haben wir nicht.

## 2.6 Kommunikation mit dem Nutzer

Die Kommunikation zwischen unserem Roboterhund und dem Nutzer ist ein zentraler Aspekt unseres Projekts, der die sichere Interaktion ermöglicht. Unser Ziel war es, eine Kommunikationsschnittstelle zu schaffen, die es dem Nutzer erlaubt, dem Roboter Ziele vorzugeben, während der Roboter gleichzeitig dem Nutzer assistiert und ihm Sicherheit auf dem Weg vermittelt.

Funktionen der Kommunikation:

* Drahtlose Kommunikation: Nutzer und Roboter können Informationen austauschen.
* Zielvorgabe durch den Nutzer: Der Nutzer kann dem Roboter per Sprachbefehl oder über eine andere Eingabemethode ein Ziel vorgeben.
* Assistenz und Sicherheitsvermittlung durch den Roboter: Der Roboter unterstützt den Nutzer aktiv während der Navigation, indem er beispielsweise haptische Signale sendet oder den Weg weist. Er soll dem Nutzer das Gefühl von Sicherheit vermitteln, indem er Hindernisse erkennt und umgeht oder den Nutzer vor Gefahren warnt.

Die Idee, ein Armband mit Mikrofon zu verwenden, kam auf, um eine bequeme und unauffällige Möglichkeit zur Audioaufnahme zu schaffen. Dieses Armband sollte als zentrale Schnittstelle für die Kommunikation dienen.

Komponenten des Kommunikationssystems:

Um die Funktionalität des Armbands zu gewährleisten, waren folgende Komponenten notwendig:

* Mikrofon: Zur Aufnahme der Sprachbefehle des Nutzers.
* Mikrocontroller: Zur Verarbeitung der Audiodaten, zur Kommunikation mit den anderen Komponenten und zur Weiterleitung der Befehle an den Roboter.
* Batterie: Zur Stromversorgung des Armbands.
* Micro-SD-Karte: Zur temporären Speicherung von Daten, falls erforderlich.
* Button: Zur Aktivierung/Deaktivierung bestimmter Funktionen.
* Kompass: Zur Unterstützung der Navigation.

Es ist wichtig zu betonen, dass diese Komponenten in einem voneinander abhängenden System funktionieren. Keine der Komponenten kann isoliert betrieben werden; sie sind aufeinander angewiesen, um die gewünschte Funktionalität zu erreichen.

Umsetzung der Kommunikation:

//