Entwicklung und Untersuchungen zur Nutzbarkeit eines Rollstuhls als Eingabegerät zur Navigation im virtuellen Raum

Benedikt Christian Beigang

14. September 2022

Fakultät Informatik und Medien
HTWK Leipzig

Abstract

Inhaltsverzeichnis

Kapitel 1

Einleitung

Mit dem Aufkommen des modernen Computers sind parallel die unterschiedlichsten Geräte und Methoden entwickelt worden, mit denen man Anweisungen an die Rechenmaschine übermitteln kann. Anfangs noch mit Lochkarten, entwickelten sich die unterschiedlichsten Eingabegeräte und Methoden, um der Maschine präzise zu vermitteln, was sie tun soll. Mit dem Aufkommen des Bildschirms und der Verbreitung des Computers in der Breite der Bevölkerung, entstanden immer mehr virtuelle Umgebungen und gleichzeitig der Wunsch, in diesen möglichst barrierefrei zu navigieren. So wurden bekannte Eingabegeräte aus anderen Bereichen, in die Welt des Computers übersetzt, wie zum Beispiel die Schreibmaschine hin zur Tastatur. Jedoch wurden auch neue Formen und Funktionen gefunden, um den Ansprüchen einer komplexeren digitalen Umgebung gerecht werden zu können. So entstand zum Beispiel die Computer-Maus, um in einer zweidimensionalen Umgebung navigieren zu können. Der Spielcontroller hingegen entpuppte sich nicht nur als gutes Eingabegerät im zweidimensionalen Raum, sondern auch im dreidimensionalen.
Heute erlauben eingebettete Systeme theoretisch jeden Gegenstand in ein Eingabegerät zu verwandeln. Solange ein Gegenstand unterschiedliche Zustände abbilden kann, ist es möglich, über die Manipulation dieser Zustände dem Computer Anweisungen zu übermitteln. Begrenzt wird die Anzahl unterschiedlicher Anweisungen nur durch die Anzahl der verschiedenen Zustände, die das Eingabegerät annehmen kann. Jedoch muss gleichzeitig auch gewährleistet sein, dass jeder Nutzer einfach, präzise und bequem die Eingaben tätigen kann.
Ein Gegenstand, der wenig Beachtung als Eingabegerät erhalten hat, ist der Rollstuhl. Dieser besitzt zwei Räder, die, mit unterschiedlicher Geschwindigkeit, in jeweils zwei Richtungen gedreht werden können. Es stellt sich die Frage, inwieweit man mit einem Rollstuhl im virtuellen Raum sich bewegen und mit ihm interagieren kann und was nötig ist, um ein solches System zu entwickeln. Durch die geringe Anzahl von Eingaben, also den zwei Rädern, ist der Nutzer vermutlich eingeschränkter als wenn er zum Beispiel direkt einen Spielcontrol-

ler verwenden würde. Jedoch birgt die Reduktion von Eingaben und die sehr mechanische Art und Weise, wie man mit dem Rollstuhl interagiert, auch Chancen. So wäre denkbar, das entwickelte System in der Stadtplanung zu verwenden, um leichter die Barrierefreiheit von zukünftigen Bauprojekten virtuell zu testen. Auch wäre denkbar, das System in virtuellen Welten zu nutzen. Drehen sich die Räder des Rollstuhls frei, so kann der Nutzer sich frei im Raum bewegen. Dabei muss er seine Position in der echten Welt nicht verändern und ist nicht limitiert durch die Größe des Raums, in dem er sich befindet.
Diese Arbeit widmet sich also der Frage, mit welchen Mitteln ein Rollstuhl als Eingabegerät umfunktioniert werden kann und wie die Bewegungen der Räder des Rollstuhls in sinnvolle Anweisungen für den Computer abgebildet werden können. Dabei wird ein eingebettetes System entwickelt, dass die Rotationsdaten der Räder an eine Software auf dem Computer übermittelt (Im Folgenden Rollstuhl-Software genannt). Danach werden Verfahren untersucht, mit dem die empfangenen Daten auf Eingaben eines herkömmlichen Eingabegeräts abgebildet werden können, damit die Anweisungen auch von anderer Software gelesen werden kann.

Kapitel 2

Begriffsklärung

Im Folgenden sollen nicht triviale Begriffe erklärt werden die für das Verständis dieser Arbeit notwendig sind.

2.1 Eingebettetes System

Ein eingebettetes System besteht aus einem programmierbaren Mikroprozessor und wird meistens umgeben von Sensoren und Aktoren. Das gesamte System bildet damit eine Schnittstelle zwischen physikalischen Prozessen und einem elektrischen Gerät.

Marwedel definiert ein eingebettetes System wie folgt:

> „Eingebettete Systeme sind informationsverarbeitende Systeme, die in umgebende Produkte integriert sind.“ (S. 2)[1]

Gessler definiert eingebettete Systeme wie folgt:

> „Das Eingebettete Systeme sind Rechenmaschinen, die in elektrischen Geräten „eingebettet“ sind, z. B. in Kaffeemaschinen, CD-, DVD-Spielern oder Mobiltelefonen. Unter eingebetteten Systemen verstehen wir alle Rechensysteme außer den Desktop-Computern.“ (S. 7)[2]

Da für das Steuern von Aktoren und auslesen von Sensoren, sowie das Empfangen und Übertragen von Daten aus heutiger Sicht keine große Rechenleistung benötigt wird, sind die verwendeten Mikroprozessoren verglichen mit modernen Prozessoren nicht Leistungsstark. Das hat zur Folge, dass Randbedingungen entstehen, welche bei der Entwicklung beachtet werden müssen. Dazu zählen Rechenleistung, Verlustleistung und Ressourcenverbrauch (S. 233)[2]. Jedoch sind grade deshalb solche Systeme erschwinglich, da keine Hardware benötigt wird, welche auf maximale Rechenleistung und Speicherverbrauch ausgelegt ist.

Heute sind eingebettete Systeme in den unterschiedlichsten Anwendungsgebieten zu finden, wie zum Beispiel: Autos, Schienenfahrzeuge, Flugzeugen, in der Telekommunikation und bei der Fertigungsautomatisierung (S. 2)[1]. Eingebettete Systeme werden meist mit Hardware-nahen Sprachen programmiert wie C und C++, da diese durch ihre Nähe zur Hardware die höchste Effizienz versprechen (S. 159)[2].

2.2 Virtueller Raum

Im Rahmen dieser Arbeit wird der Begriff virtueller Raum wie folgt definiert: Ein virtueller Raum ist ein durch einen Computer erzeugte Umgebung mit unterscheidbaren Objekten in ihm, welche auf eine zweidimensionale Projektionsfläche abgebildet wird. Es ist kein im physikalischen Sinne real existierender Raum mit messbaren Abmessungen. Die Wahrnehmung als Raum entsteht erst durch die Interpretation des Nutzers als diesen. Ein Raum zeichnet sich dadurch aus, dass man durch ihn navigieren kann. So hat der Nutzer eine Position in diesem Raum, sowie die Möglichkeit diese Position gezielt zu verändern. In der Praxis kann zwischen zweidimensionalen und dreidimensionalen Räumen unterschieden werden. Anders als zweidimensionale Räume können dreidimensionale Räume nicht 1:1 auf die Projektionsfläche abgebildet werden. Es ist eine Funktion notwendig, die alle Punkte des Raums auf die Projektionsfläche abbildet. Ein Beispiel für zweidimensionale Räume sind gängige GUIs, der Nutzer kann innerhalb dieser zwischen verschiedenen Tabs, Seiten oder Bereichen navigieren. Ein Beispiel für dreidimensionale Räume sind 3D-Computerspiele. In ihnen wird eine virtuelle Welt berechnet, in der der Spieler selbstbestimmt umherwandern kann.

2.3 Navigation

Der Duden definiert das Verb *navigieren* wie folgt:

> 1. „den Standort eines Schiffes oder Flugzeugs bestimmen und es auf dem richtigen Kurs halten“
> 2. „(z. B. bei der Suche nach Informationen im Internet) [gezielt] ein Programm oder einen Programmpunkt nach dem anderen aktivieren“ [3]

Navigation ist also die Gesamtheit der Mittel, die nötig sind, um navigieren zu können. Ursprünglich kommt das Wort aus der Seefahrt. Früher wurden „*Landmarken, die Küste, Meeresströmungen, Lotungen der Wassertiefen, jahreszeitliche regelmäßige Winde, Wolkenansammlungen und der Flug der Zugvögel*“ (S. 17)[4] genutzt, um die aktuelle Position des eigenen Schiffes bestimmen zu können. Somit konnte sichergestellt werden, dass das Schiff sein Ziel erreicht. Im Laufe der Zeit kamen immer mehr Methoden und Werkzeuge zum Einsatz, um immer präziser die eigene Position ermitteln und den optimalen Weg bestim-

men zu können. Jedoch beschränkt sich das Wort nicht auf das Herausfinden der eigenen Position, sondern schließt die Tätigkeit der Wegfindung, sowie das Kurshalten eines vorher festgelegten Pfades mit ein. Mit dem Aufkommen neuer Verkehrsmittel wie Flugzeuge und neuen Technologien wie GPS wurde der Begriff der Navigation weiter gefasst. So definiert der Medienwissenschaftler Florian Sprenger Navigation wie folgt: „*... eine Praxis des Umgangs mit Relationen.*“ (S. 1)[5]. Das navigierende Individuum oder Objekt entscheidet auf Grundlage von „*medial ver- oder kulturtechnisch ermittelten Verhältnis zu anderen Objekten [...] oder durch Repräsentationen dieser Relationen auf imaginären, geographischen oder digitalen Karten*“ in welche Richtung sich bewegt werden muss (S. 1)[5]. Im Kontext dieser Arbeit navigiert der Nutzer auf einer zweidimensionalen Ebene, in einem dreidimensionalen virtuellen Raum. Er steuert dabei gezielt seine Fortbewegung und interagiert mit der virtuellen Umgebung, indem die Räder eines Rollstuhls, in dem er sitzt, gedreht werden. Es soll im Rahmen dieser Arbeit zusätzlich hervorgehoben werden, dass Navigation im Gegensatz zum reinen Fortbewegen nicht passiv ist. Das Objekt oder Individuum, das navigiert, hat einen Einfluss auf die Richtung der Fortbewegung.

2.4 Gyroskop

Ein Gyroskop ist ein Sensor, der genutzt wird, um „*die Winkelgeschwindigkeit um eine feste Achse*“ zu messen (S. 1)[6]. Es gibt verschiedene Arten von Gyroskopen, die Messungen mit unterschiedlichen Methoden vornehmen. So wird zwischen drei Arten von Gyroskopen unterschieden: optische, vibrierende und welche, bei denen eine Masse rotiert. Mithilfe von Micro Eelectro Mechanical Systems (MEMS) konnten Trägheitssensoren miniaturisiert und in Massen produziert werden. In der Elektrotechnik sind diese beliebt, da sie klein und kostengünstig sind[7]. Diese messen meist den Coriolis-Effekt. Die Kraft, die der Effekt wirken lässt, entsteht durch die Rotation um eine der vorher festgelegten Achsen. Der Sensor kann diese Kraft messen und als elektrisches Signal weitergeben, sodass anschließend das Signal digitalisiert werden kann[8]. Heute sind sie in durch ihre Verfügbarkeit in den verschiedensten Anwendungsgebieten zu finden, wie dem Auto, der Medizin oder der Unterhaltungselektronik (S. 1)[6].

2.5 Eingabegerät

Im Buch *Virtual und Augmented Reality (VR/AR)* von 2013 werden Eingabegeräte wie folgt beschrieben:

> „Eingabegeräte dienen der sensorischen Erfassung von Nutzerinteraktionen.“ (S. 97)[9]

Das heißt, dass Eingabegeräte eine Schnittstelle von Mensch zu Maschine darstellen.

Kapitel 3

Stand der Forschung

Das Verwenden eines Rollstuhls als Eingabegerät ist zu großen Teilen in der Forschung zu finden, die sich damit beschäftigt, beeinträchtigten Menschen zu helfen. So gibt es bislang Studien, die sich mit der Frage beschäftigen, inwieweit Virtual Reality genutzt werden kann, um Menschen, die seit kurzem einen Rollstuhl verwenden müssen, bei der Eingewöhnung zu helfen[10]. Jedoch ist in diesen Fällen der Rollstuhl meist nur ein passives Eingabegerät. Der Rollstuhl ist nicht aufgebockt, sondern die Bewegung im virtuellen Raum findet durch das Bewegen der VR-Brille statt. Dadurch wird eine Rollstuhl-Simulation gespart, jedoch ist der Nutzer in der Fortbewegung im virtuellen Raum begrenzt, durch die physischen Begrenzungen im echten Raum.

Das im Jahr 2000 erschiene Paper: *Development of a wheelchair virtual reality platform for use in evaluating wheelchair access* geht hingegen der Frage nach, inwieweit ein realer Rollstuhl im Virtuellen simuliert werden kann[11]. Dazu wurde eine „*Motion-Platform*“ konstruiert, auf den ein handelsüblicher Rollstuhl platziert werden kann. Die Konstruktion simuliert über die reine Fortbewegung hinaus, auch die Krafteinwirkung bei Schrägen und unebenen Böden. Im Paper wird jedoch nicht ausführlich auf die hard- oder softwareseitige Modellierung des virtuellen Rollstuhls oder der Plattform eingegangen. Zudem ist die Konstruktion entsprechend groß, unpraktikabel und teuer. Für mögliche Endkunden würde eine so aufwändige Konstruktion einen Sinn ergeben.

Ebenfalls im Jahr 2000 erschien das Paper: *Simulation of the behaviour of a powered wheelchair using virtual reality*[12]. In diesem wird beschrieben, wie ein elektrischer Rollstuhl in der virtuellen Realität simuliert werden kann. Jedoch beschränkt sich die Eingabe auf einen Thumbstick, der häufig an elektrischen Rollstühlen zu finden ist. In Ansätzen wird schon hier beschrieben, welche Auswirkungen die Rotation der Räder des Rollstuhls, auf dessen Rotation haben. Jedoch beschränkt sich die Simulation auf ideale Bewegungen und es gibt keine weiteren Interaktionsmöglichkeiten außer die Bewegung im Raum.

Diese Arbeit geht einen Schritt weiter und versucht ein System zu entwickeln, bei dem ein Rollstuhl im virtuellen Raum simuliert werden kann. Dabei soll, wie bei Harrision und Co., auf aufwändige Konstruktionen verzichtet werden, sodass das System kostengünstig und praktikabel für mögliche Endkunden ist. Trotzdem soll der simulierte Rollstuhl, anders als bei Niniss und Nadif, eine realistische Simulation der Räder beinhalten und möglichst viele Interaktionen über die reine Fortbewegung beinhalten.

Kapitel 4

Entwicklung des eingebetteten Systems

Damit der Rollstuhl-Software bekannt ist, mit welcher Geschwindigkeit sich welches Rad in welche Richtung dreht, ist Hardware notwendig. Um dies zu bewerkstelligen wurde ein eingebettetes System entwicklet. Dieses muss die Rotationsdaten messen und an die Software übermitteln. Dabei kommt ein ESP32-Mikrocontroller zum Einsatz der ein Gyroskop ausliest und anschließend die Daten mithilfe eines Übertragungsprotokolls an die Rollstuhl-Software überträgt. Näher soll in einem Vergleich beleuchtet werden, ob WiFi und ESP-Now das geeignetere Protokoll ist.

4.1 PlatformIO

Zur Entwicklung der eingebetteten Software, die auf den Mikrocontrollern läuft, wurde PlatformIO verwendet. Dies ist eine Framework-Erweiterung für Visual Studio Code, bei der die benötigten Bibliotheken, die für jeden Mikrocontroller und jedes Board notwendig sind, automatisch heruntergeladen und eingerichtet werden. Ebenfalls lassen sich über das User Interface (UI), Bibliotheken, die für das jeweilige Projekt notwendig sind, hinzufügen. Zusätzlich zur Entwicklungsumgebung von Visual Studio Code gibt es Funktionalitäten einen Chip zu flashen und anschließend im seriellen Monitor die Ausführung zu beobachten. Im Gegensatz zu Umgebungen wie der Arduino Integrated Development Environment (IDE) wird Zeit gespart, da dort zunächst manuell Treiber heruntergeladen werden müssen. Zusätzlich kann man nicht von den Vorteilen einer modernen IDE profitieren.

4.2 Messung der Raddaten

Um die Rotation der Räder des Rollstuhls messen zu können, wird ein Sensor benötigt. Dabei wurde sich für ein Gyroskop entschieden, da diese verfügbar, kostengünstig und leicht integrierbar sind[1]. Jedoch erfordert die Verwendung eines Gyroskops eine Voreinstellung und Kalibrierung, welche in diesem Unterkapitel erörtert werden.

4.2.1 Gyroskop

Im Zuge dieser Arbeit wurde sich für das Motion-Tracking-Device GY-521 MPU-6050 entschieden. Dieses ist klein (mit Pins: 20mm x 15mm x 11mm), kostengünstig zu erwerben (ca. 4 Euro) und verfügt unter anderem über 3-Achsen-Gyroskop-Sensoren, mit denen die Rotation gemessen werden kann. Der Chip besitzt folgende 8 Anschlüsse (S.7)[13]:

Tabelle 4.1: Pins des GY-521 MPU-6050

Anschluss	**Funktion**	**Notwendig**
VCC	Power-Supply	Ja
GND	Ground	Ja
SCL	I2C Serial-Clock Line	Ja
SDA	I2C Serial-Date Line	Ja
XDA	Auxiliary Serial Data	Nein
XCL	Auxiliary Serial Clock	Nein
ADO	I2C Address Select	Ja
INT	Interrupt Digital Output	Nein

Die Daten können mithilfe eines angeschlossenen Mikrocontrollers (ESP32) ausgelesen werden. Jede Achse wird auf zwei 8-Bit-Register abgebildet (S. 31)[14]. Zusammen ergibt das einen Wertebereich von 65.536 unterscheidbaren Zuständen. Mit der Drehrichtung rückwärts halbiert sich dieser Wertebereich, da ein Bit für das Verschieben des Wertebereichs ins Negative benötigt wird. Das Gyroskop des MPU-6050 kann in vier verschiedenen Konfigurationen betrieben werden (S. 31)[14]. Damit wird festgelegt, wie klein der Winkel zwischen zwei verschiedenen Zuständen ist; mit anderen Worten, wie viele Stufen pro Grad unterschieden werden können. Da der Wertebereich konstant ist, bedeutet eine empfindlichere Messung, dass das Gyroskop bei einer geringeren Geschwindigkeit das Ende des Wertebereichs erreicht. Angewendet auf den Rollstuhl hat das zur Folge, dass das rotierende Rad bei niedrigeren Geschwindigkeiten seine maximal messbare Geschwindigkeit erreicht. Es gilt folgender Zusammenhang mit welchem die aktuelle Winkelgeschwindigkeit errechnet werden kann:

[1] Es wurde sich gegen die Verwendung einer Lichtschranke entschieden, da bei dieser Art von Messung keine kleinen Geschwindigkeiten möglich sind.

$$z: \text{ Gemessene Stufen} \tag{4.1}$$
$$z_m: \text{ Maximale Anzahl von Stufen} \tag{4.2}$$
$$p: \text{ Stufen pro Grad [s/°]} \tag{4.3}$$
$$\omega: \text{ Aktuelle Winkelgeschwindigkeit [°/s]} \tag{4.4}$$
$$\omega_m: \text{ Maximal messbarere Winkelgeschwindigkeit [°/s]} \tag{4.5}$$

$$z_m = 32768 \tag{4.6}$$
$$z_m = \omega_m \cdot p \tag{4.7}$$
$$\omega = \left(\frac{z}{p}\right) \tag{4.8}$$

Tabelle 4.2: Einstellbare Modi des Gyroskops mit ihren resultierenden Eigenschaften

	Modus 0	**Modus 1**	**Modus 2**	**Modus 3**
Maximal messbarere Winkel-geschwindigkeit [°/s]*	250	500	1000	2000
Stufen pro Grad [1/°]*	131	65,5	32,8	16,4
Maximale Umdrehungszahl pro Sekunde [1/s]	0,69	1,39	2,78	5,56
Maximale Radianten pro Sekunde [rad/s]	4,36	8,73	17,47	34,93
Zurückgelegte Distanz pro Stufe [mm]**	0,04	0,08	0,16	0,32

*(S. 31)[14]
**Werte bei einem Raddurchmesser von 60 cm

Es stellt sich die Frage, welcher der optimale Modus für das hier entwickelte System ist. Um dem Nutzer ein möglichst störungsfreies Erlebnis zu bieten, muss gewährleistet sein, dass das Gyroskop so empfindlich wie möglich eingestellt ist. Das bedeutet, dass der Wertebereich maximal ausgereizt werden muss. Ist der Modus nicht empfindlich genug, so bemerkt der Nutzer möglicherweise das Springen der Bitwerte in Form eines Vorspringens in der Bewegung. Allerdings muss ein Modus gewählt werden, welcher dazu führt, dass der Nutzer nicht schneller als die maximale Gradzahl pro Sekunde drehen kann, da es sonst zu

einem Zahlenüberlauf kommt und zu einer fehlerhaften Weiterverarbeitung der Daten führt. Der Zahlenüberlauf kann zwar abgefangen werden, jedoch sollte bei Bedarf einer maximale Geschwindigkeit diese programmgesteurert festgelegt werden. Dies birgt den Vorteil den maximalen Wert flexibler setzen zu können. Der Modus muss also so empfindlich sein, dass der Nutzer nicht den Übergang von einem Zustand in den nächsten registriert. Gleichzeitig darf er nicht in der Lage sein, die Räder schneller als die maximale Gradzahl pro Sekunde zu drehen.

4.2.2 Idealer Gyroskop-Modus

Wie zuvor beschrieben, muss der Frage nachgegangen werden, in welchem Modus das Gyroskop betrieben werden sollte. Hierfür wurde eine Datenreihe gemessen, mit der Gradzahl pro Sekunde im Verlauf der Zeit. Eine Testperson hat dabei versucht, ein Rad so schnell wie möglich zu drehen.

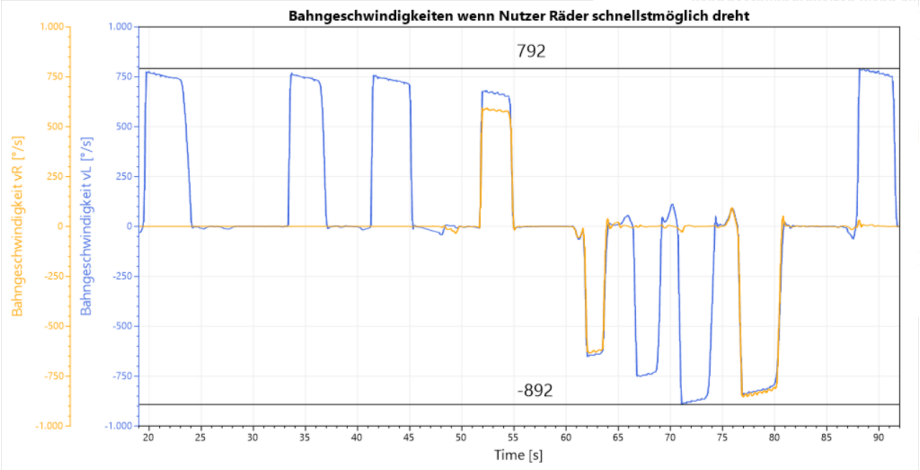

Abbildung 4.1: Winkelgeschwindigkeiten der Räder im Verlauf der Zeit, bei dem die Testperson ein Rad, mit der dominanten Hand, so schnell wie möglich dreht

Abbildung 4.1 zeigt bei einer Roation nach vorne einen maximal erreichten Ausschlag um 800 $°/s$. Bei Rotationen nach hinten ist der Ausschlag etwas höher, übersteigt jedoch nicht -900 $°/s$. Daraus folgt, dass Gyroskop-Modus 2 für dieses Szenario der Ideale ist. Bei diesem Modus ist die Maximale Gradzahl pro Sekunde 1000 $°/s$. Der Nutzer erreicht nicht die maximal Geschwindigkeit und reizt trotzdem den Wertebereich ca. 80 % – 90 % aus. Deshalb wird dieser Modus im System verwendet.

In vielen Anwendungen ist es von Vorteil oder angenehmer für den Nutzer, sich schnellstmöglich mit der maximalen Fortbewegungsgeschwindigkeit zu bewegen. Auf Dauer ist es ermüdend, die Räder möglichst schnell drehen zu müssen. Um den Nutzer zu entlasten, kann entweder der Fortbewegungsvektor $\vec{f}$ immer auf einen maximalen Thumbstick-Ausschlag abgebildet werden oder es

wird ein weiterer Schwellenwert eingeführt, ab dem alle Eingaben als maximaler Thumbstick-Ausschlag abgebildet werden.

4.2.3 Verbesserung der Rohdaten

Die ausgelesenen Werte des Gyroskops sind nicht automatisch kalibriert. Sie besitzen einen konstanten Offset. Deshalb muss beim Start des Systems eine Kalibrierungssequenz gestartet werden. Diese errechnet aus einer Reihe ausgelesener Werte einen Mittelwert, der anschließend von allen zukünftigen Werten abgezogen wird. Dazu dürfen die Räder nicht bewegt werden, da dies das Ergebnis der Kalibrierung unbrauchbar machen würde.

Darüber hinaus hat das Gyroskop-Signal ein Rauschen. Bei hoher Umdrehungszahl ist das Rauschen irrelevant, da es nur einen kleinen Anteil der Gesamtrotation ausmacht. Steht das Rad still, ist das Rauschen jedoch störend, da in diesem Fall nicht erkennbar ist, ob es tatsächlich stillsteht oder eine geringe Rotation gegeben ist. Aus diesem Grund wird ein Schwellenwert bestimmt, welcher der Gyroskop-Wert überschreiten muss, um als Bewegung erkannt werden zu können. Damit ist sichergestellt, dass es sich um eine tatsächliche Rotation handelt.

4.3 ESP32

Um den MPU-6050 betreiben und dessen Daten an die Rollstuhl-Software übermitteln zu können, wird ein Mikrocontroller-Board benötigt. Es muss die entsprechenden Register auslesen und mittels drahtloser Kommunikation versenden. Auf dem Markt ist eine große Anzahl von Produkten für die verschiedensten Anwendungsgebiete erhältlich. Im Rahmen dieser Arbeit wurde der Mikrocontroller ESP32 verwendet, das aktuelle Modell der Firma *Espressif*. Boards mit diesem Chip sind kostengünstig (ca. 8 Euro). Zudem ist der ESP32 mit WiFi (802.11 b/g/n), Bluetooth (v4.2) und ESP-Now Unterstützung ausgestattet (S. 8-9)[15]. Verbaut wurde ein Xtensa® 32-bit LX6 Mikroprozessor, mit 240MHz Taktfrequenz, 448 KB Read Only Memory (ROM) und 520 KB Static Random Access Memory (SRAM). (S. 32)[15] Als Entwicklungsboard wurde das ESP32 Dev Kit C V4 verwendet.

Der MPU-6050 muss wie folgt an das Entwicklungsboard angeschlossen werden:

4.4 3D gedruckte Box

Damit Entwicklungsboard, Gyroskop und Akku zusammengehalten werden, geschützt sind und am Rad befestigt werden können, wird eine Box benötigt, die alle Komponenten aufnehmen kann und diese trägt. Aufgrund dessen wurde – mithilfe von Blender – eine entsprechend seinen Anforderungen konstruierte Box entworfen und mittels 3D-Druckers gedrucht.

Tabelle 4.3: Zuweisungs der jeweiligen Pins

ESP32	**MPU-6050**
3.3V	VCC
GND	GND
GPIO22 (I2C CL)	SDA
GPIO21 (I2C DA)	SCL
ADO	GND

4.5 Vergleich zwischen WiFi und ESP-Now

Für die Übermittlung der Sensordaten an die Rollstuhl-Software auf einem Personal Computer (PC), stehen verschiedene Möglichkeiten zur Verfügung. In dieser Arbeit sind zwei verschiedene Protokolle getestet worden: WiFi und ESP-Now. Die Protokolle müssen dabei leicht in das System integrierbar sein. WiFi ist ein weit verbreiteter Standard, sodass entsprechende Bibliotheken schon existieren, um das Protokoll einbinden zu können[16]. ESP-Now ist weniger verbreitet, da aber der Chip vom selben Hersteller kommt, existieren auch hier schon Bibliotheken, beziehungsweise ist die benötigte Bibliothek schon im Entwicklungspaket des Chips schon enthalten[17]. Ein weiteres verfügbares Protokoll ist Bluetooth. Jenes muss jedoch aufgrund des zeitlichen Rahmens dieser Arbeit, an anderer Stelle beleuchtet werden.

4.5.1 WiFi und WebSockets

WiFi ist eine Kommunikationstechnologie, die durch die WiFi-Alliance entstanden ist und bis heute von ihr gepflegt wird[18]. Sie ermöglicht drahtlose Kommunikation mit jedem Gerät, welches diese Technologie implementiert. Inzwischen ist WiFi ein weit verbreiteter Standard, welcher von den meisten mobilen Geräten unterstützt wird[19]. Ein solches Gerät ist der ESP32.

Zunächst muss eine Verbindung zwischen dem ESP32 und dem lokalen Netzwerk mittels WiFi aufgebaut werden. Damit die Zugangsdaten nicht fest in den Code geschrieben werden müssen, wird die Bibliothek *WiFi Manager*[20] verwendet. Diese baut selbstständig eine Verbindung mit einem Netzwerk auf, nachdem die Zugangsdaten über ein Gerät wie zum Beispiel einem Smartphone übergeben wurden. Dazu wird ein Web-Konfigurations-Portal auf dem ESP32 gehostet, auf das ein nahes WiFi-fähiges Gerät zugreifen kann.

Für die eigentliche Übertragung der Daten können verschiedene Protokolle verwendet werden. Ein klassischer Vertreter ist Hyper Text Transfer Protocol (Secure) (HTTP(S)). Dieses wurde für die vorliegende Arbeit jedoch nicht verwendet, da das Protokoll auf Hypertext ausgelegt ist. Es wird für jede Abfrage von Daten eine neue Transmission Control Protocol (TCP)-Verbindung mit dem Server aufgebaut, der die Daten nach Eingang der Anfrage zurückschickt. Will der

Client neue Daten empfangen, so muss dieser erneut eine TCP-Verbindung mit dem Server aufbauen (S. 4)[21]. Zusätzlich enthält jedes Paket, welches vom Clienten kommt, viel Overhead, da jede dieser Nachrichten einen HTTP(S)-Header besitzt (S. 4)[21]. Da es sich bei den Gyroskop-Daten jedoch um Echtzeitdaten handelt, wäre dieses Vorgehen ineffizient. Viel Zeit und Bandbreite würde für das Übertragen von nicht benötigte Daten verwendet werden.

Eine Alternative ist die Verwendung von einem WebSocket. Das Protokoll wurde entwickelt, um die Nachteile von HTTP(S) bei Echtzeitdaten zu umgehen und wird heute breit unterstützt. Das WebSocket-Protokoll wurde in seiner finalen Form 2011 von der Internet Engineering Task Force entwickelt und veröffentlicht[21]. Dabei wird analog zu HTTP(S) zu Beginn ein TCP Handshake durchgeführt. Der Client stellt an den Server eine Verbindungsanfrage, welcher dieser bestätigt. Ab diesem Zeitpunkt sendet der Server unaufgefordert die vom Client abonnierten Daten, bis die Verbindung vom Client beendet wird (S. 5)[21]. Somit lassen sich höhere Datenraten erzielen, die für Echtzeitanwendungen notwendig sind. Das für die vorliegende Bachelor-Thesis entwickelte System setzt auf einen vom ESP32 gehosteten WebSocket-Server, der von der Rollstuhl-Software auf dem PC abonniert wird. Dabei kommt aufseiten des ESP die Bibliothek *arduinoWebSockets*[22] zum Einsatz, und auf der Client Seite die Bibliothek *websocket-client*[23].

Zusätzlich zur eigentlichen Übertragung der Daten ist es notwendig, dass die Software auf dem PC den Internet Protocol (IP)-Endpunkt des WebSockets auf dem ESP32 kennt. Dazu sendet der Mikrocontroller ebenfalls über WiFi einen User Datagram Protocol (UDP)-Broadcast ins Netzwerk. Neben dem IP-Endpunkt werden auch Informationen über das Gerät mitgesendet, damit die Rollstuhl-Software weiß, um welches Gerät es sich handelt. Nach dieser Bekanntmachung kann der WebSocket abonniert und die Daten übertragen werden.

4.5.2 ESP-Now und Serieller Port

ESP-Now ist ein vom Unternehmen *Espressif* selbst entwickeltes Übertragungsprotokoll, mit dem Mikrocontroller von *Espressif* wie zum Beispiel der ESP8266 (Vorgänger des ESP32) und der ESP32 direkt miteinander Daten austauschen können. Dabei verwendet das Protokoll die Media Access Control (MAC)-Adressen zur Identifikation der Geräte. Es wird jedoch nur eine Verbindung in eine Richtung aufgebaut. Ein großer Vorteil dieses Protokolls ist die unkomplizierte Einbindung in das System. Anders als WiFi muss nicht zunächst eine Verbindung zu einem Netzwerk aufgebaut werden, sondern dem Gerät muss lediglich die MAC-Adresse des Zielgeräts vorliegen. Da die Kommunikation jedoch nur unter Mikrocontrollern stattfindet, muss das Gerät, welches die Sensor-Daten entgegennimmt, diese Daten mittels seriellen Ports an die Rollstuhl-Software übertragen. Damit steigt die Anzahl der Verbindungen, an denen die Übertragung scheitern kann. Jedoch erleichtert es die Verwendung für den Endbenutzer, da dieser kein WiFi-Netzwerk benötigt, um die Geräte mit der Software auf dem PC zu verbinden. Eine Verbindung per Universal Serial Bus (USB)-Kabel ist

ausreichend.

4.6 Analyse der Messungen

Damit der Nutzer eine präzise Eingabe tätigen kann, ist es notwendig, dass möglichst schnell und kontinuierlich neue Pakete empfangen werden. Um die Datenrate zu ermitteln, mit der beide Nodes (eingebettete Systeme an den Rädern) die Gyroskop-Werte verschicken, wird clientseitig alle 250ms die aktuelle Datenrate errechnet. Dazu zählt die Software seit der letzten Messung die eingegangenen Pakete und multipliziert diese mit 4, um die Datenrate pro Sekunde zu erhalten. Zusätzlich wird bei jedem Datensatz ermittelt, wie viel Zeit zwischen den letzten beiden Paketen vergangen ist[2].

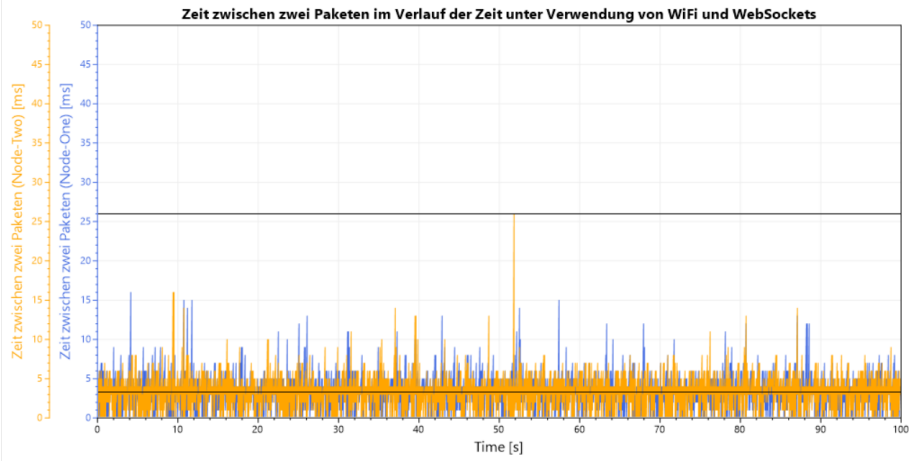

Abbildung 4.2: Intervall zwischen Paketen im Verlauf der Zeit bei WiFi

Den Abbildungen 4.2 und 4.3, sowie der Tabelle 4.4 ist zu entnehmen, dass das Interval zwischen zwei Paketen, bei beiden Mehtoden, im Schnitt im niedrigen einstelligen Millisekundenbereich ist. ESP-Now mit seriellem Port schafft dabei im Durchschnitt 80 Pakete mehr als WiFi mit einem WebSocket. Die Verbindung mit WiFi ist jedoch deutlich störungsfreier. So ist aus Abbildung 4.3 abzulesen, dass entweder ESP-Now oder der serielle Port regelmäßiger und höhere Ausreißer erzeugt, bei denen die Zeit zwischen zwei Paketen über 15 Millisekunden ist. WiFi hingegen hat im kompletten Datensatz nur einen deutlichen Ausreißer (Abbildung 4.2) und hat ansonsten selten Zeiten über 15 Millisekunden. Es kann geschlussfolgert werden, dass WiFi vorzuziehen ist. Jedoch kann auf ESP-Now mit seriellem Port zurückgegriffen werden, wenn nur ein USB-Anschluss und kein WiFi Netzwerk verfügbar ist. Anzumerken ist jedoch, dass die Messungen stark abhängig davon sind, welche USB-Anschlüsse und -Protokolle verwendet

[2]Auf die Messung der Latenz wird im Hinblick auf den Umfang der Arbeit verzichtet, da dies erfodert hätte beide Seiten der Verbindung zu synchronisieren.

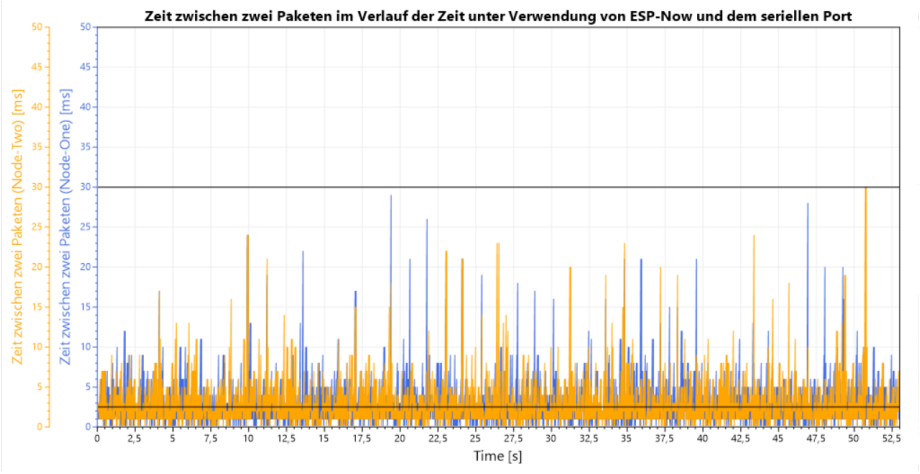

Abbildung 4.3: Intervall zwischen Paketen im Verlauf der Zeit bei ESP-Now

Tabelle 4.4: Messungen der getesteten Verbindungsmethoden

	WiFi mit WebSocket	**ESP-Now mit seriellem Port**
Pakete pro Sekunde Durchschnitt	250,16	330,43
Pakete pro Sekunde Minimum	220	204
Pakete pro Sekunde Maximum	284	440
Paketintervall Durchschnitt [ms]	3,09	2,52
Paketintervall Minimum [ms]	<1	<1
Paketintervall Maximum [ms]	26	30

wurden, sowie welche Datenraten der Router unterstützt. Ein weiterer Einflussfaktor ist die Verbindung zwischen Client und dem Router. Sind diese kabellos verbunden, erhöht sich zusätzlich die Zeit, die ein Paket zur Übertragung benötigt, im Gegensatz zur kabelgebundener Übertragung. Deshalb sind die erhobenen Messwerte nur begrenzt aussagekräftig. Trotzdem lässt sich erkennen, dass beide Methoden genug Pakete verschicken können, um eine flüssige Bewegung nativ aus den Daten berechnen zu können. Es sind keine Interpolationstechniken notwendig, um die Bewegung flüssig erscheinen zu lassen.

Tabelle 4.5: Die Verbindungsmethoden im Vergleich

	WiFi mit WebSocket	**ESP-Now mit seriellem Port**
Anzahl Mikrocontroller	2	3
Daten-Pfad	ESP32 ↓ Router ↓ Rollstuhl-Software	ESP32 ↓ ESP32 ↓ Serieller-Port ↓ Rollstuhl-Software
"Hardcoding" von Verbindungs-Informationen	nicht notwendig	Ziel-MAC-Adressen
Vorteile	• Stabilere Verbindung • Nur eine Übertragungs-technologie notwendig	• Kein Verbindungs-aufbau mit lokalem Netzwerk notwendig • Kein WiFi-Netzwerk notwendig, ein USB-Anschluss genügt

Kapitel 5

Abbildung der empfangenen Daten auf ein Eingabegerät

Nachdem die Daten des eingebetteten Systems zur Rollstuhl-Software übermittelt wurden, müssen diese nun auf auf ein herkömmliches Eingabegerät abgebildet werden. Es wird sich der Frage gewidmet, welches Eingabegerät in Frage kommt und wie möglichst viele, präzise Eingaben auf dieses Eingabegerät abgebildet werden kann. Im letzten Teil des Kapitels werden aufgenommene Datenreihen untersucht, um fehlerhafte Eingaben zu minimieren.

5.1 Interface zur Nutzung der Rad-Daten in externer Software

Die Raddaten, welche die Rollstuhl-Software empfangen hat, müssen nun zu externer Software gelangen. Dazu ist eine vorhandene Schnittstelle notwendig (Tastatur, Maus, Spielcontroller, ...), denn es kann nicht davon ausgegangen werden, dass jede externe Software eine neue Schnittstelle implementiert. Jedoch muss in diesem Fall, in der Rollstuhl-Software eine Abbildung auf eine vorhandene Schnittstelle durchgeführt werden.

5.1.1 Vergleich zwischen Tastatur und Spielcontroller

Tastaturen und Spielcontroller werden von den meisten Anwendungen unterstützt und eignen sich unterschiedlich gut für die Zwecke des hier entwickelten Systems. Tastatureingaben bieten den Vorteil, dass sie von fast jeder erdenklichen Software unterstützt werden. Jedoch lassen nur binäre Eingaben, durch das Drücken von Tasten tätigen. Da die Rollstuhl-Eingaben jedoch unterschiedliche Werte innerhalb eines Wertebereichs darstellen, wird bei einer Tastatureingabe die Interaktionsmöglichkeit stark eingeschränkt. Zudem ist oft das zusätzliche

Verwenden einer Maus erforderlich. Dies verkompliziert das Abbilden zusätzlich. Die Alternative ist ein Mapping auf eine Spielcontroller-Eingabe. Hier gibt es ebenfalls Knöpfe, beziehungsweise binäre Eingaben, aber auch Eingaben entlang von Achsen innerhalb eines Wertebereichs. Überwiegend werden die Achsen in Form eines Thumb-Sticks oder Knopfes mit mehreren Stufen realisiert. Auf der einen Seite könnten so Rollstuhl-Eingaben, wie das Fortbewegen, einfacher auf das Gerät abgebildet werden. Andererseites unterstützt nicht jede Software Eingaben mittels eines Spielcontrollers. Es wurde sich im Rahmen dieser Arbeit für das Abbilden auf, und emulieren von, einem Spielcontroller entschieden. Grund dafür ist, dass die meiste Software in der Fortbewegung eine Rolle spielt (meist Computerspiele oder andere 3D-Räume), diese unterstützen.

5.1.2 Emulation des Spielcontrollers

Um die Eingaben des Rollstuhls in tatsächliche Spielcontroller-Eingaben umzuwandeln, die vom Betriebssystem auch als Controller-Eingabe verstanden werden, ist eine Emulation eines Controllers notwendig. Ziel ist es, programmgesteuert Controller-Eingaben an den Rechner zu senden. Um sich den Aufwand des Schreibens eines neuen Treibers zu ersparen, wird an dieser Stelle auf das *Virtual Gamepad Emulation Framework*[24] zurückgegriffen. Dies ist eine Bibliothek, welche in bestehende Software integriert werden kann und einen virtuellen Controller mit dem Rechner verbindet. Über Befehle lassen sich anschließend Controller-Eingaben tätigen. Das Framework unterstützt Xbox 360-, sowie DualShock 4-Controller[25].

Die Internet-Vertriebsplattform Steam, welche hauptsächlich Computerspiele vertreibt, hat eine Umfrage veröffentlicht, über die Verteilung von Spielcontrollern auf ihrer Platform. Der Abbildung 4.4 ist zu entnehmen, dass 45 % aller Controller ein Xbox 360 Controller sind[26]. Damit sind sie mit großem Abstand am verbreitetsten. Aufgrund dessen wurde sich im Rahmen der vorliegenden Arbeit für die Emulation eines Xbox 360 Controllers entschieden, da anzunehmen ist, dass dieser Controller am wahrscheinlichsten unterstützt wird.

5.2 Algorithmen zur Abbildung der Rad-Daten in Controller-Eingaben

Die Sensor-Daten der Gyroskope liefern die Winkelgeschwindigkeiten der Räder des Rollstuhls. Diese sollen – wie bereits in Kapitel 5.3.1 beschrieben – auf die Eingabemöglichkeiten eines Spielcontrollers abgebildet werden, um sich im virtuellen Raum bewegen oder andere Eingaben tätigen zu können. Die Abbildung erfolgt dabei auf einen Xbox360 Controller. Somit sind die abgebildeten Eingaben von jeder Software lesbar, die eine Xbox360 Controllerunterstützung implementiert haben. Die in diesem Kapitel werden gelgentlich Aussagen getroffen die mit dem Emfinden des Nutzers zu tun haben oder von ihm abhängig sind. Da subjektive Eindrücke nur schwer und unpräzise gemessen werden können,

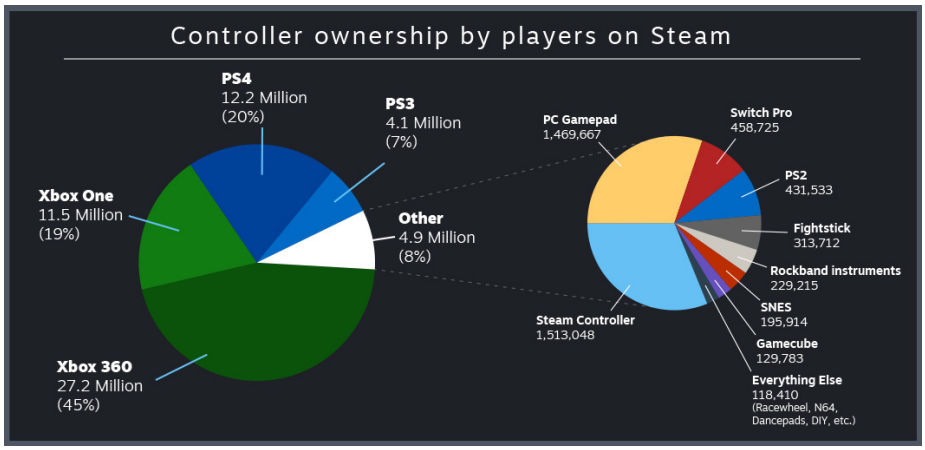

Abbildung 5.1: Verteilung von Besitz von verschiedenen Spielcontrollern auf der Plattform Steam[26]

leidet dadurch an manchen Stellen die Aussagekraft. Darüber hinaus konnten nur wenige Personen im Umkreis des Erstellers dieser Arbeit als Testperson verwendet werden, was zusätzlich einzelne Aussagen abschwächt. Eine zukünftige Arbeit muss sich ausführlicher an diesen Stellen mit den Betreffenden Fragen befassen, und einen größeren diverseren Pool aus Testpersonen verwenden, um die Aussagekraft zu erhöhen.

5.2.1 Abbildung auf einen Thumbstick

Der direkte Weg die Raddaten in eine Eingabe umzuwandeln ist, diese auf jeweils eine Achse eines Thumbsticks abzubilden. Dabei wird die x-Achse mit dem einen, die y-Achse mit dem anderen Rad dargestellt. Vorteilig ist dabei, dass beide Achsen gleichzeitig angesprochen werden können. Jedoch ist es schwieriger, die x-Achse zu bewegen, da sie anders ausgerichtet ist als das Rad, das gedreht wird. Alternativ kann das Ansprechen einer Achse auch durch beide Räder passieren. Dabei wird die x-Achse dann angesprochen, wenn sich die Räder gegeneinander drehen und die y-Achse, wenn sich die Räder miteinander drehen. Damit wird eine intuitive Nutzung angestrebt. Jedoch ist es dabei nicht mehr möglich, gleichzeitig den Cursor entlang beider Achsen zu bewegen, da sich die Räder nicht gleichzeitig mit- und gegeneinander drehen können.

5.2.2 Abbildung auf einen simulierten Rollstuhl

Da das im Rahmen dieser Arbeit entwickelte System darauf abzielt, in einem dreidimensionalen virtuellen Raum, auf einer Ebene zu navigieren, wird eine Abbildung benötigt, die die Position des Nutzers im virtuellen Raum verändert. Da die Daten ohnehin von einem Rollstuhl kommen, liegt die Abbildung auf einen simulierten Rollstuhl nahe. Um die Raddaten der zwei Räder auf eine

Bewegung und Rotation eines Rollstuhls umzurechnen, muss erst festgestellt werden, welche Drehbewegungen zu welchen Rollstuhlbewegungen führt. Dabei können vier vereinfachte Fälle unterschieden werden:

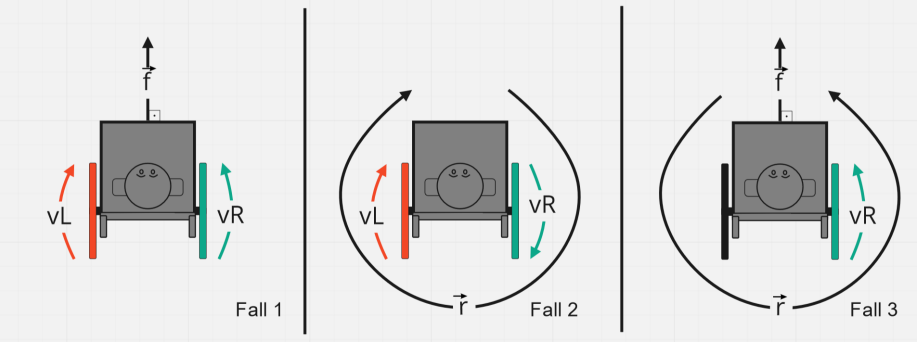

Abbildung 5.2: Die Bewegungs-Fälle des Rollstuhl aus der Vogelperspektive

Fall 1: Drehen sich die Räder mit gleicher Geschwindigkeit in dieselbe Richtung, so ruft dies eine Bewegung nach vorne oder hinten aus (Abbildung 5.2, Fall 1).

Fall 2: Drehen sich die Räder mit gleicher Geschwindigkeit gegeneinander, so ruft dies eine Rotation um die eigene Achse hervor (Abbildung 5.2, Fall 2).

Fall 3: Dreht sich nur ein Rad, so dreht sich dieses um das Stehende (Abbildung 5.2, Fall 3).

Fall 4: Drehen sich die Räder unterschiedlich schnell, so muss die Bewegung zusammengesetzt werden aus den Bewegungskomponenten einer der ersten beiden Fälle und dem dritten Fall.

Im Folgenden wird die Berechnung der Bewegungsanteile aufgezeigt, bestehend aus Bewegung nach vorne/hinten und der Rotation um die eigene Achse:

ω_L : Winkelgeschwindigkeit des linken Rades [°/s] (5.1)

ω_R : Winkelgeschwindigkeit des rechten Rades [°/s] (5.2)

m : Winkelgeschwindigkeit Minimum [°/s] (5.3)

o : Overshoot [°/s] (5.4)

d : Abstand Der Räder [m] (5.5)

$\vec{f}$: Fortbewegungsvektor [m/s] (5.6)

$\vec{r}$: Rotationsvektor [m/s] (5.7)

$\vec{f_{1,2}}$: Fortbewegungsvektor Fall1 oder Fall2 [m/s] (5.8)

$\vec{r_{1,2}}$: Rotationsvektor Fall1 oder Fall2 [m/s] (5.9)

$\vec{f_3}$: Fortbewegungsvektor Fall3 [m/s] (5.10)

$\vec{r_3}$: Rotationsvektor Fall3 [m/s] (5.11)

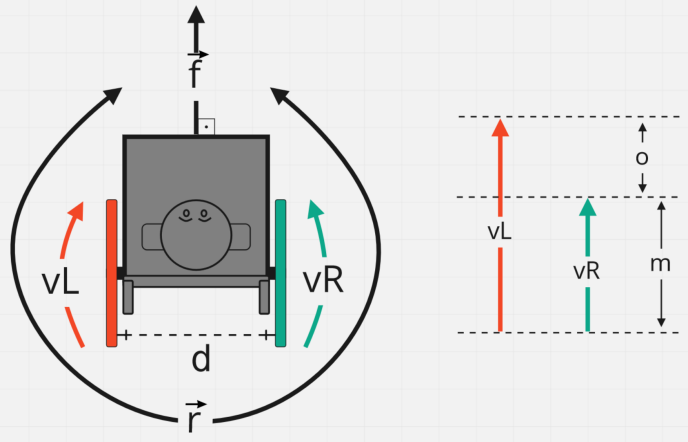

Abbildung 5.3: Skizze des Rollstuhls aus der Vogelperspektive mit Bewegungsvektoren

Zunächst müssen die Rotationen der Räder dekonstruiert werden. Dabei lässt sich die Rotation eines Rades in zwei Komponenten aufspalten (siehe Abbildung 5.3). Zum einen in den Minimum-Anteil m, den sich beide Räder drehen,

$$m = \min\left(|\omega_L|, |\omega_R|\right) \tag{5.12}$$

zum anderen in den Overshoot-Anteil o, den sich ein Rad schneller dreht als das andere.

$$o = ||\omega_L| - |\omega_R|| \tag{5.13}$$

Fall 1: Die Bewegung nach vorne oder hinten (Fortbewegungsvektor Fall1 oder Fall2 $f_{1,2}$) ergibt sich in diesem Fall aus dem Anteil der Geschwindigkeit, mit denen sich beide Räder drehen. Dabei dreht sich jedoch der Rollstuhl nicht.

$$\vec{f_{1,2}} = m \tag{5.14}$$

$$\vec{r_{1,2}} = 0 \tag{5.15}$$

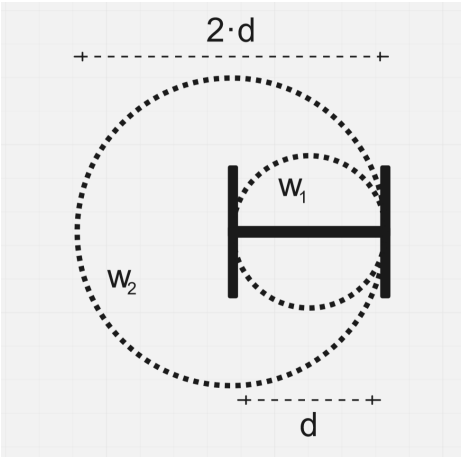

Abbildung 5.4: Skizze des Rollstuhls aus der Vogelperspektive mit Wendekreisen

Fall 2: Zur Berechnung der Rotation um die eigene Achse wird zunächst der Wendekreis w_1 benötigt. Dieser Wendekreis ist abhängig vom Abstand der Räder d und dessen Mittelpunkt liegt im Mittelpunkt zwischen den Rädern (siehe Abbildung 5.4). Anschließend wird mithilfe des Minimums m, das Verhältnis von m zu w_1 errechnet, also wie viel vom Wendekreis gedreht wird. Dieses Verhältnis muss zum Schluss mit 360 multipliziert werden, um den resultierenden Winkel, beziehungsweise Rotationsvektor Fall1 oder Fall2 $r_{1,2}$, zu berechnen. Bei dieser Bewegung verändert der Rollstuhl jedoch nicht seine Position.

$$w_1 = d \cdot \pi \tag{5.16}$$

$$\vec{r_{1,2}} = \left(\frac{m}{w_1}\right) \cdot 360 \tag{5.17}$$

$$\vec{f_{1,2}} = 0 \tag{5.18}$$

Fall 3: Bei diesem Fall gibt es einen Fortbewegungs- und einen Rotationsvektor ungleich null. Da sich nur ein Rad bewegt, hat sich der Wendekreis vergrößert zu w_2. Der Durchmesser von w_2 ist nun doppelt so groß wie von w_1, da das stehende Rad nun der Mittelpunkt des Wendekreises ist. Jetzt wird der Overshoot o (also der Anteil der Bewegung des Rades, das sich mehr als das andere dreht) ins Verhältnis gesetzt mit w_2 und erhält dadurch Θ. Verrechnet man Θ mit dem

inneren Wendekreis w_1, so erhält man den Fortbewegungsvektor Fall3 f_3.

$$w_2 = 2 \cdot d \cdot \pi \tag{5.19}$$

$$\Theta = \frac{o}{w_2} \tag{5.20}$$

$$\vec{f_3} = \Theta \cdot w_1 \tag{5.21}$$

Um den Rotationsvektor Fall3 r_3 berechnen zu können, muss Θ mit 360 multipliziert werden.

$$\vec{r_3} = \Theta \cdot 360 \tag{5.22}$$

Da die Berechnungen mit den absoluten Rotationswerten errechnet wurden, ist es notwendig anhand der Drehrichtungen beider Räder zu bestimmen, ob sich der Rollstuhl vor- oder zurückbewegt und ob er sich dabei nach links oder rechts dreht. Die Drehrichtung ist immer dann links, wenn:

$$\omega_L < \omega_R \tag{5.23}$$

sonst ist sie rechts. Es handelt sich um eine Vorwärtsbewegung, wenn:

$$\omega_L + \omega_R > 0 \tag{5.24}$$

sonst ist es eine Rückwärtsbewegung. Folgende Bedingungen können vernachlässigt werden, da sie in der Realität nur Auftreten wenn der Overshoot $o = 0$ ist und damit $\vec{f_3} = 0$ und $\vec{r_3} = 0$:

$$\omega_L = \omega_R \tag{5.25}$$

$$\omega_L + \omega_R = 0 \tag{5.26}$$

Fall 4: Bevor die zusammengehörenden Vektoren zusammengerechnet werden können, muss überprüft werden, ob $s_{1,2}$ und $r_{1,2}$ mithilfe des ersten oder zweiten Falls berechnet werden müssen. Gilt folgende Bedingung, drehen sich die Räder gegeneinander und es wird Fall 2 benötigt. Andernfalls gilt Fall 1.

$$(\omega_L > 0) \oplus (\omega_R > 0) \tag{5.27}$$

Anschließend können die Bewegungskomponenten addiert werden:

$$\vec{s} = \vec{f_{1,2}} + \vec{f_3} \tag{5.28}$$

$$\vec{r} = \vec{r_{1,2}} + \vec{r_3} \tag{5.29}$$

Sind die Bewegungskomponenten berechnet muss final eine Abbildung dieser auf den Spielcontroller erfolgen. Dabei bietet sich die y-Achse des linken-Thumbsticks

an für den Fortbewegungsvektor $\vec{s}$ und die s-Achse des rechten Thumbsticks für den Rotationsvektor $\vec{r}$. In vielen Anwendungen, im besonderen 3D-Videospielen, ist der linke Thumbstick für das Bewegen im Raum zuständig und der rechte für das Umschauen.

Berechnet man auf Grundlage der oben genannten Formeln die Bewegung des Rollstuhls, so stößt man auf folgendes Problem: Da die Räder bei einer beabsichtigten Bewegung nach vorne meist mit leicht unterschiedlicher Geschwindkeit drehen, erfährt der simulierte Rollstuhl eine Ablenkung nach rechts oder links. Eine Ausrichtung des Rollstuhls, die der Nutzer gezielt vorgenommen hat, um ein bestimmtes Ziel im virtuellen Raum zu erreichen, ist damit nichtig, da der Rollstuhl sein Ziel verfehlt. Bildet man den Durchschnitt v der Rotationen beider Räder und nutzt ihn anstelle vom $Minimum\ m$, so verhindert man die unbeabsichtigte Ablenkung.

$$\vec{v} = \frac{(\omega_L + \omega_R)}{2} \tag{5.30}$$

Verwendet man den Durchschnitt v müssen die verschiedenen Fälle distinkt sein, da sonst das Wenden mit einem Rad nicht mehr möglich wäre. Grund dafür ist der zusammenaddierter Mittelwert der Geschwindigkeiten des stillstehenden und des sich drehenden Rades. Dieser würde anstelle einer Drehung zu einer Vorwärtsbegung führen. Das Zusammenrechnen von Fall 1 oder Fall 2, mit Fall 3 darf also nicht geschehen. Um dies zu realisieren, sind Bewegungszustände notwendig. Auf Basis dieser kann entschieden werden, welcher Fall genutzt werden muss, um die resultierende Bewegung zu errechnen. Im nächsten Kapitel wird darauf eingegangen, welche Bewegungszustände existieren und wie diese erkannt werden können.

5.2.3 Bewegungszustände

Um alle Bewegungszustand-Permutationen ermitteln zu können, muss festegestellt werden, dass sich ein Rad in drei Zuständen befinden kann: Still stehend, nach vorne drehend und nach hinten drehend. Zwei Räder mit jeweils drei Zuständen ergeben dabei neun Bewegungszustände ($3^2 = 9$), welche in Abbildung 5.5 skizziert sind.

Jedoch lässt sich die Menge von Bewegungszuständen in folgende disjunkte Teilmengen unterteilen:

- *Ruhezustand*: kein Rad dreht sich (5)
- *Rotation um die eigene Achse*: Räder drehen sich gegeneinander (4, 6)
- *Einzelradbewegung*: ein Rad steht still und ein Rad dreht sich (1, 3, 7, 9)
- *Sichtachsenbewegung*: Räder drehen sich in dieselbe Richtung (2, 8)

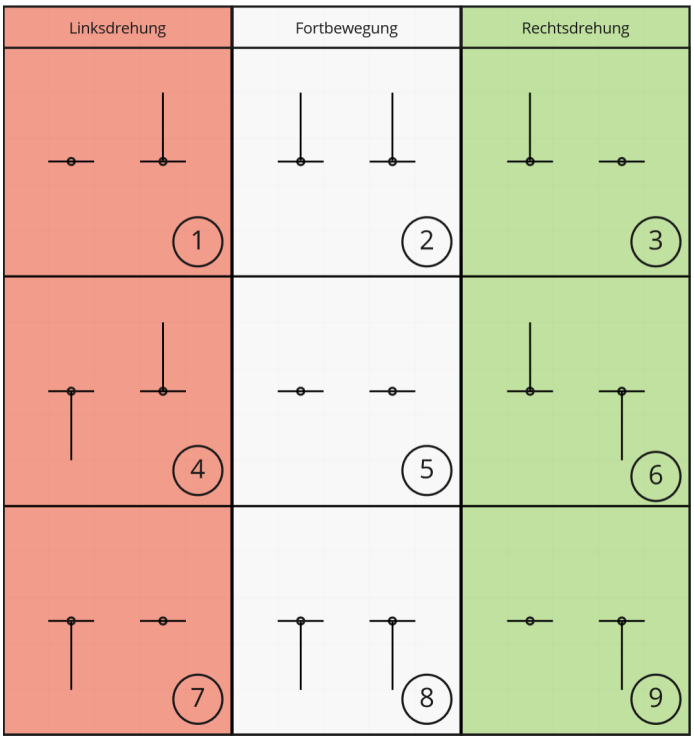

Abbildung 5.5: Die neun Bewegungszustände eines Rollstuhls

Im Folgenden wird darauf eingegangen, wie diese Teilmengen-Zustände erkannt werden:

$$\omega_L : \text{ Winkelgeschwindigkeit des linken Rades} \tag{5.31}$$

$$\omega_R : \text{ Winkelgeschwindigkeit des rechten Rades} \tag{5.32}$$

$$s : \text{ Schwellenwert} \tag{5.33}$$

Ruhezustand

Der *Ruhezustand* wird erreicht, wenn kein anderer Zustand zutrifft oder sich kein Rad dreht. Da das oben beschriebene Rauschen abgeschnitten wurde, gilt der Ruhezustand wenn:

$$(\omega_L = 0) \wedge (\omega_R = 0) \tag{5.34}$$

Einzelradbewegung

Wie beim Ruhezustand ist es auch hier möglich, darauf zu prüfen, ob ein Wert 0 ist. Die einfachste Art und Weise dies zu prüfen, ist folgende Bedingung:

$$(\omega_L = 0) \oplus (\omega_R = 0) \tag{5.35}$$

Der Nutzer hat jedoch Schwierigkeiten, ein Rad vollständig ruhig zu halten. Die Gyroskop-Werte überschreiten selbst bei kleinen Handbewegungen den Schwellenwert für die Rauschunterdrückung. Dies führt zu unbeabsichtigten Eingaben. Deshalb ist diese Methode unzureichend. Führt man einen Schwellenwert t ein, wird eine vom Nutzer unbeabsichtigte *Einzelradbewegung* zuverlässiger unterdrückt:

$$(|\omega_L| < s) \oplus (|\omega_R| < s) \tag{5.36}$$

Rotation um die eigene Achse
Die Bedingung, die gelten muss, wenn sich beide Räder gegeneinander drehen, ist identisch mit der Bedingung, welche schon im Kapitel *Abbildung auf einen realistisch simulierten Rollstuhl* aufgestellt wurde. Diese gilt nur, wenn *Ruhezustand* und *Einzelradbewegung* ausgeschlossen werden konnten, da nicht die Fälle abgedeckt werden, wenn ω_L oder ω_R 0 sind:

$$(\omega_L > 0) \oplus (\omega_R > 0) \tag{5.37}$$

Sichtachsenbewegung
Der Zustand der Sichtachsenbewegung gilt dann, wenn sich die Räder in dieselbe Richtung drehen. Damit ist dieser Zustand das logische Gegenteil der Bedingung *Rotation um die eigene Achse*, vorausgesetzt es wurden *Ruhezustand* und *Einzelradbewegung* ausgeschlossen:

$$(\omega_L > 0) \Leftrightarrow (\omega_R > 0) \tag{5.38}$$

5.2.4 Abbildung auf einen simulierten Rollstuhl mit zusätzlichen Interaktionen

Werden, wie in Kapitel *Abbildung auf einen simulierten Rollstuhl* erläutert, die Eingaben ausschließlich auf eine Rollstuhlbewegung abgebildet, so ist der Nutzer eingeschränkt in seinen Interaktionsmöglichkeiten. Aktionen, wie ein Tastendruck auf dem emulierten Spielcontroller sind in diesen Fällen nicht möglich. Jedoch können bestimmte Bewegungsmuster, die nicht zwangsläufig notwendig sind, genutzt werden, um weitere Interaktionen abzubilden. Werden für das Drehen des Rollstuhls nur die Zustände genutzt, bei denen sich die Räder gegeneinander drehen (Abb. 5.5 Zustand 4 und 6), so bleiben vier Zustände übrig, die mit anderen Interaktionen belegt werden können (Abb. 5.5 Zustand 1, 3, 7, 9). Bei diesen vier Mustern handelt es sich, um die Einzelradbewegungs-Zustände. Diese können dann beispielsweise für das Drücken der Interaktionstasten des Spielcontrollers (A, B, X, Y) genutzt werden.

Bislang wurde die Anzahl der möglichen Interaktionen, durch die Anzahl der Bewegungszustände begrenzt. Will man weitere Interaktionen abbilden, so bleibt nur die Geschwindigkeit mit der sich die Räder drehen oder die Zeit, um Anweisungen zu kodieren. Bezieht man die Zeit in der sich Räder drehen mit ein,

so könnte man das Morse-Alphabet abbilden. Dazu wäre nötig zwei Zustände zu definieren, wie zum Beispiel ein Rad kurz oder lange drehen. Jedoch verkompliziert dies die Eingabe erheblich für den Nutzer, sodass diese Methode nicht weiter verfolgt wird. Eine Alternative dazu ist die Geschwindigkeit mit einzubeziehen. Teilt man den Wertebereich der Geschwindigkeiten eines Rades, so lässt sich in der Theorie die Anzahl der Bewegungszustände verdoppeln. Dazu wird der Wertebereich zunächst mithilfe des Schwellenwertes $s = 100$ geteilt. [1] Die Unterscheidung zwischen langsamer und schneller Rotation ist jedoch intuitiv von jedem Nutzer begreifbar und umsetzbar. Im Wertebereich der langsamen Bewegungen können nun Bewegungen, wie das Neigen des Kamerawinkels, abgebildet werden. Dabei wurde sich für die Bewegungszustände 2 (Neigung nach oben) und 8 (Neigung nach unten) entschieden. Für das Detektieren dieses neuen Teilmengen-Bewegungszustandes wird folgende Bedingung benötigt:

Neigen

$$((\omega_L > 0) \Leftrightarrow (\omega_R > 0)) \wedge (|\omega_L| < s) \wedge (|\omega_R| < s) \tag{5.39}$$

Unweigerlich geht dabei die Möglichkeit verloren, seinen Fortbewegungsvektor s feiner einzustellen. Es sind also keine langsamen Bewegungen nach vorne und hinten möglich. Dafür hat der Nutzer die Möglichkeit, sich frei im Raum umschauen zu können.

5.3 Optimierung der Detektion von Bewegungszuständen

Für den Nutzer ist die korrekte Detektion von Bewegungszuständen entscheidend. Werden unerwünschte Zustände ermittelt, führt dies zu fehlerhaften Eingaben, welche der Nutzer als störend empfindet. Je mehr verschiedene Bewegungszustände voneinander unterschieden werden müssen, desto höher ist die Gefahr der Missinterpretation. Abgesehen davon ist es nicht immer sinnvoll für alle Bewegungsmuster den Wertebereich zu teilen (wie in Kapitel *Abbildung auf einen simulierten Rollstuhl mit zusätzlichen Interaktionen*). Bewegungen, wie die *Rotation um die eigene Achse*, wollen vom Nutzer entweder langsam und präzise oder schnell durchgeführt werden. Nur in bestimmten Fällen, wie bei der Fortbewegung, kann es sinnvoll sein, den Wertebereich zu teilen. So ist die Anzahl der tatsächlich sinnvollen Bewegungsmuster kleiner als die theoretisch denkbaren. Es muss bei jedem Zustand und jeder Interaktion abgewogen werden, ob eine Teilung des Wertebereichs sinnvoll erscheint, oder den Nutzer behindert. Trotz der verringerten Anzahl an Bewegungsmustern, sind beim Testen der *Abbildung auf einen simulierten Rollstuhl mit zusätzlichen Interaktionen* zwei primäre Probleme beobachtet worden, bei denen fehlerhafte Eingaben

[1] Von einer weiteren Teilung ist abzuraten, da es sonst für den Nutzer schwierig wird, die Räder mit den gewünschten beziehungsweise notwendigen Geschwindigkeiten zu drehen.

getätigt werden. Zum Testen des Systems und Aufnehmen der Daten wurde *Counter-Strike: Global Offensive* verwendet.

5.3.1 Unbeabsichtigtes Betätigen von Interaktionstasten

In den ersten Testreihen wurde für die Detektion von einer *Einzelradbewegung* und dem Teilen des Wertebereichs in schnelle und langsame Bewegungen derselbe Schwellwert $s = 100$ verwendet. Unter Verwendung dieser Methode kommt es beim Anfahren oder Bremsen (*Sichtachsenbewegung*) zum unbeabsichtigten Betätigen von Interaktionstasten. Da sich die Räder nicht mit derselben Geschwindigkeit drehen, gibt es ein kurzes Zeitintervall, in dem ein Rad unter dem Schwellwert und ein Rad über dem Schwellenwert liegt. Aus der Abbildung 5.6 ist abzulesen, dass für dieses Zeitintervall die Bedingung der *Einzelradbewegung* gilt, sodass eine Interaktionstaste betätigt wird.

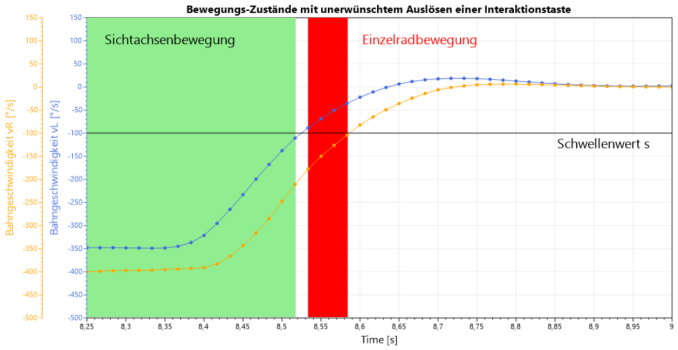

Abbildung 5.6: Bewegungs-Zustände mit unerwünschter *Sichtachsenbewegung*

Dieses Problem lässt sich über das Einführen eines neuen *Einzelradbewegung* Schwellenwertes beheben. Wählt man für diesen einen geringeren Wert $s_1 = 25$ als für den Schwellwert für das Teilen des Wertebereichs s_2, entsteht eine Pufferzone. Beim Beschleunigen überschreiten die Gyroskop-Werte zunächst nacheinander s_1. Anschließend überschreiten die Werte nacheinander s_2. Solange beide Werte in der Pufferzone sind, kann weder eine *Einzelradbewegung* noch eine *Sichtachsenbewegung* detektiert werden (siehe Abbildung 5.7).

Durch das Einführen der neuen Schwellenwerte müssen folgende Teilmengen-Bewegungszustände erweitert werden:

Einzelradbewegung

$$((|\omega_L| < s_1) \oplus (|\omega_R| < s_1)) \wedge ((|\omega_L| > s_2) \oplus (|\omega_R| > s_2)) \tag{5.40}$$

Sichtachsenbewegung

$$((\omega_L > 0) \Leftrightarrow (\omega_R > 0)) \wedge ((|\omega_L| \geq s_2) \wedge (|\omega_R| \geq s_2)) \tag{5.41}$$

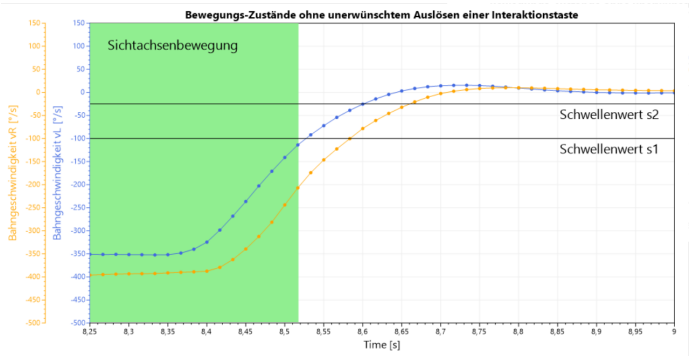

Abbildung 5.7: Bewegungs-Zustände ohne unerwünschter *Sichtachsenbewegung*

Neigen

$$((\omega_L > 0) \Leftrightarrow (\omega_R > 0)) \wedge (|\omega_L| < s_2) \wedge (|\omega_R| < s_2) \tag{5.42}$$

5.3.2 Unbeabsichtigtes Neigen beim Anfahren

Beim Anfahren oder Bremsen (*Sichtachsenbewegung*) wurde beobachtet, dass für ein kurzes Zeitintervall der Kamerawinkel unbeabsichtigt geneigt wird. Ähnlich wie beim vorangegangenen Problem wird auch hier beim Übergang von einem Zustand zum Nächsten ein unerwünschter Zwischenzustand erreicht. In diesem Fall gibt es beim Beschleunigen einen kurzes Zeitinterval in dem die Rollstuhl-Software im *Neigen*-Zustand ist, da $|\omega_L|$ und $|\omega_R|$ beide unter dem Schwellenwert s_2 sind (siehe Abbildung 5.8).

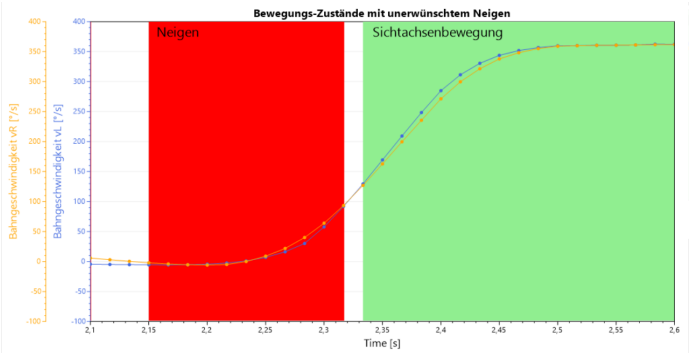

Abbildung 5.8: Bewegungs-Zustände mit unerwünschtem *Neigen*

Da die Fehldetektion immer dann auftritt, wenn sich die Geschwindigkeit der Räder ändert, ist der hier verwendete Lösungsansatz, die Bedingung des Neigungs-Zustandes zu erweitern. In dieser wird nun auch abgefragt, ob die Summe der

Änderungsraten a der Gyroskope unter einem neuen Schwellenwert s_3 liegt. Haben die Räder ihre Zielgeschwindigkeit erreicht, fällt die Änderungsrate unter s_3, sodass der nächste korrekte Zustand detektiert werden kann. In den Tests hat sich $s_3 = 15$ als ein akzeptabler Wert herausgestellt, wie aus den Diagrammen abzulesen ist. Für die Berechnung der Änderungsrate wird folgende Berechnung verwendet:

$$a = |(\omega_{L,[1]} - \omega_{L,[0]})| + |((\omega_{R,[1]} - \omega_{R,[0]})| \tag{5.43}$$

Die Bedingung des Teilmengen-Zustands *Neigen* muss wie folgt ergänzt werden:

Neigen

$$((\omega_L > 0) \Leftrightarrow (\omega_R > 0)) \wedge (|\omega_L| < s_2) \wedge (|\omega_R| < s_2) \wedge (a < s_3) \tag{5.44}$$

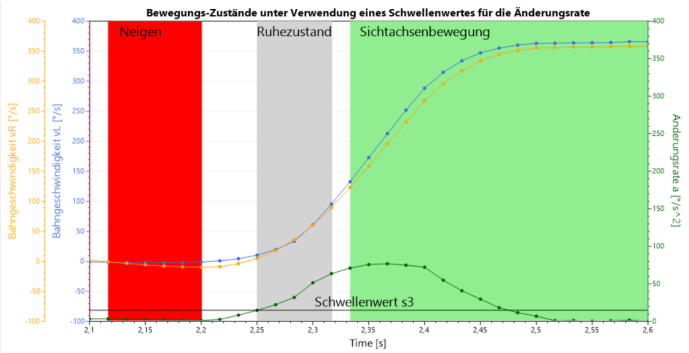

Abbildung 5.9: Bewegungs-Zustände unter Verwendung eines Schwellenwertes für die Änderungsrate ohne unerwünschtem *Neigen*

Aus der Abbildung 5.9 ist abzulesen, dass zwischen dem *Sichtachsenbewegung*-Zustand und dem *Neigen*-Zustand nun ein kurzer Ruhezustand existiert. Diese Verzögerung bei der Fortbewegung ist vom Nutzer kaum bis gar nicht wahrnehmbar. Im gezeigten Beispiel beträgt diese nur etwa 70ms. Jedoch ist anzumerken, dass diese Methode vor allem bei ruckartigen Bewegungen funktioniert, da besonders dann die Änderungsrate schnell einen registrierbaren Ausschlag hat. Verändert sich die Geschwindigkeit nicht schnell genug, kann es immer noch zu registrierbaren Neigungen kommen.

Kapitel 6

Ausblick

Kapitel 7

Fazit

Literatur

[1] Peter Marwedel. *Eingebettete Systeme: Grundlagen Eingebetteter Systeme in Cyber-Physikalischen Systemen*. Wiesbaden: Springer Fachmedien Wiesbaden, 2021. ISBN: 978-3-658-33436-9 978-3-658-33437-6. DOI: 10.1007/978-3-658-33437-6. URL: https://link.springer.com/10.1007/978-3-658-33437-6 (besucht am 05. 08. 2022).

[2] Ralf Gessler. *Entwicklung Eingebetteter Systeme*. Wiesbaden: Springer Fachmedien Wiesbaden, 2014. ISBN: 978-3-8348-1317-6 978-3-8348-2080-8. DOI: 10.1007/978-3-8348-2080-8. URL: http://link.springer.com/10.1007/978-3-8348-2080-8 (besucht am 06. 09. 2022).

[3] *Duden | navigieren | Rechtschreibung, Bedeutung, Definition, Herkunft*. URL: https://www.duden.de/rechtschreibung/navigieren (besucht am 07. 09. 2022).

[4] Gudrun Wolfschmidt. *Navigare necesse est - Geschichte der Navigation: Begleitbuch zur Ausstellung 2008/09 in Hamburg und Nürnberg*. BoD – Books on Demand, 2008. 578 S. ISBN: 978-3-8370-3260-4. Google Books: BYyLsHineFAC. URL: https://books.google.de/books?hl=de&lr=&id=BYyLsHineFAC&oi=fnd&pg=PA17&dq=die+geschichte+der+navigation&ots=9oklRNCwAS&sig=or4MD1oQ5_OmJzaw7zNEs_D08Jw&redir_esc=y#v=onepage&q=die%20geschichte%20der%20navigation&f=false.

[5] Florian Sprenger. "Navigationen und Relationen. Eine medientheoretische Skizze und ein interplanetarisches Beispiel". In: *Navigationen - Zeitschrift für Medien- und Kulturwissenschaften* 22.1 (2022), S. 243–254. DOI: 10.25969/mediarep/18785. URL: https://mediarep.org/handle/doc/19933 (besucht am 28. 08. 2022).

[6] Mario N. Armenise u. a. *Advances in Gyroscope Technologies*. Berlin: Springer, 2010. 117 S. ISBN: 978-3-642-15493-5.

[7] Kazusuke Maenaka. "MEMS Inertial Sensors and Their Applications". In: *2008 5th International Conference on Networked Sensing Systems*. 2008 5th International Conference on Networked Sensing Systems. Juni 2008, S. 71–73. DOI: 10.1109/INSS.2008.4610859.

[8] Utmel. *MPU6050 Module: Datasheet, Alternative, Comparison*. URL: https://www.utmel.com/components/mpu6050-module-datasheet-alternative-comparison?id=414 (besucht am 30. 08. 2022).

[9] Ralf Dörner u. a., Hrsg. *Virtual und Augmented Reality (VR / AR)*. eXamen.press. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. ISBN: 978-3-642-28902-6 978-3-642-28903-3. DOI: 10.1007/978-3-642-28903-3. URL: http://link.springer.com/10.1007/978-3-642-28903-3 (besucht am 05. 08. 2022).

[10] Sara Arlati u. a. "Virtual Reality-Based Wheelchair Simulators: A Scoping Review". In: *Assistive Technology* 32.6 (1. Nov. 2020), S. 294–305. ISSN: 1040-0435, 1949-3614. DOI: 10.1080/10400435.2018.1553079. URL: https://www.tandfonline.com/doi/full/10.1080/10400435.2018.1553079 (besucht am 05. 08. 2022).

[11] C S Harrision u. a. "Development of a Wheelchair Virtual Reality Platform for Use in Evaluating Wheelchair Access". In: Proceedings of the 3rd International Conference on Disability, Virtual Reality and Associated Technologies. The University of Reading, 2000, S. 8. ISBN: 978-0-7049-1142-0. URL: https://centaur.reading.ac.uk/19120/ (besucht am 14. 09. 2022).

[12] H Niniss und A Nadif. "Simulation of the Behaviour of a Powered Wheelchair Using Virtual Reality". In: Proceedings of the 3rd International Conference on Disability, Virtual Reality and Associated Technologies. The University of Reading, 2000, S. 6. ISBN: 978-0-7049-1142-0. URL: https://centaur.reading.ac.uk/19120/ (besucht am 14. 09. 2022).

[13] *GY-521_ 6-Achsen_ Gyroskop_ und_ Beschleunigungssensor_ Datenblatt*. URL: https://cdn.shopify.com/s/files/1/1509/1638/files/GY-521_6-Achsen_Gyroskop_und_Beschleunigungssensor_Datenblatt_AZ-Delivery_Vertriebs_GmbH_b35673f3-1fb8-423c-a533-524c216d92dd.pdf?v=1627564500 (besucht am 30. 08. 2022).

[14] *MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.2*. 2013. URL: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf (besucht am 30. 08. 2022).

[15] *ESP32 Datasheet*. 2022. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (besucht am 28. 07. 2022).

[16] *WiFi - Arduino Reference*. URL: https://www.arduino.cc/reference/en/libraries/wifi/ (besucht am 30. 08. 2022).

[17] *Espressif IoT Development Framework*. Espressif Systems, 30. Aug. 2022. URL: https://github.com/espressif/esp-idf/blob/5c1044d84d625219eafa18c24758d9f0e4006b2c/components/esp_wifi/include/esp_now.h (besucht am 30. 08. 2022).

[18] *Who We Are | Wi-Fi Alliance*. URL: https://www.wi-fi.org/who-we-are (besucht am 28. 07. 2022).

[19] *Discover Wi-Fi | Wi-Fi Alliance*. URL: https://www.wi-fi.org/discover-wi-fi (besucht am 30. 08. 2022).

[20] tzapu. *WiFiManager*. 27. Juli 2022. URL: https://github.com/tzapu/WiFiManager (besucht am 28. 07. 2022).

[21] IETF, Ian Fette und Alexey Melnikov. *RFC 6455 - The WebSocket Protocol*. The WebSocket Protocol. URL: https://datatracker.ietf.org/doc/html/rfc6455 (besucht am 27. 07. 2022).

[22] Markus. *WebSocket Server and Client for Arduino*. 30. Aug. 2022. URL: https://github.com/Links2004/arduinoWebSockets (besucht am 30. 08. 2022).
[23] Mariusz Kotas. *Websocket-Client*. 29. Aug. 2022. URL: https://github.com/Marfusios/websocket-client (besucht am 30. 08. 2022).
[24] *Virtual Gamepad Emulation Framework*. URL: https://vigem.org/ (besucht am 30. 08. 2022).
[25] *Virtual Gamepad Emulation Framework (GitHub)*. GitHub. URL: https://github.com/ViGEm (besucht am 30. 08. 2022).
[26] *Steam :: Steam News :: PC-Gaming mit einem Controller*. 25. Sep. 2018. URL: https://store.steampowered.com/news/group/27766192/view/ (besucht am 23. 08. 2022).

Abbildungsverzeichnis

Abkürzungsverzeichnis

PC Personal Computer

TCP Transmission Control Protocol

UDP User Datagram Protocol

IP Internet Protocol

MAC Media Access Control

HTTP(S) Hyper Text Transfer Protocol (Secure)

UI User Interface

GUI Graphical User Interface

IDE Integrated Development Environment

MEMS Micro Eelectro Mechanical Systems

ROM Read Only Memory

SRAM Static Random Access Memory

USB Universal Serial Bus