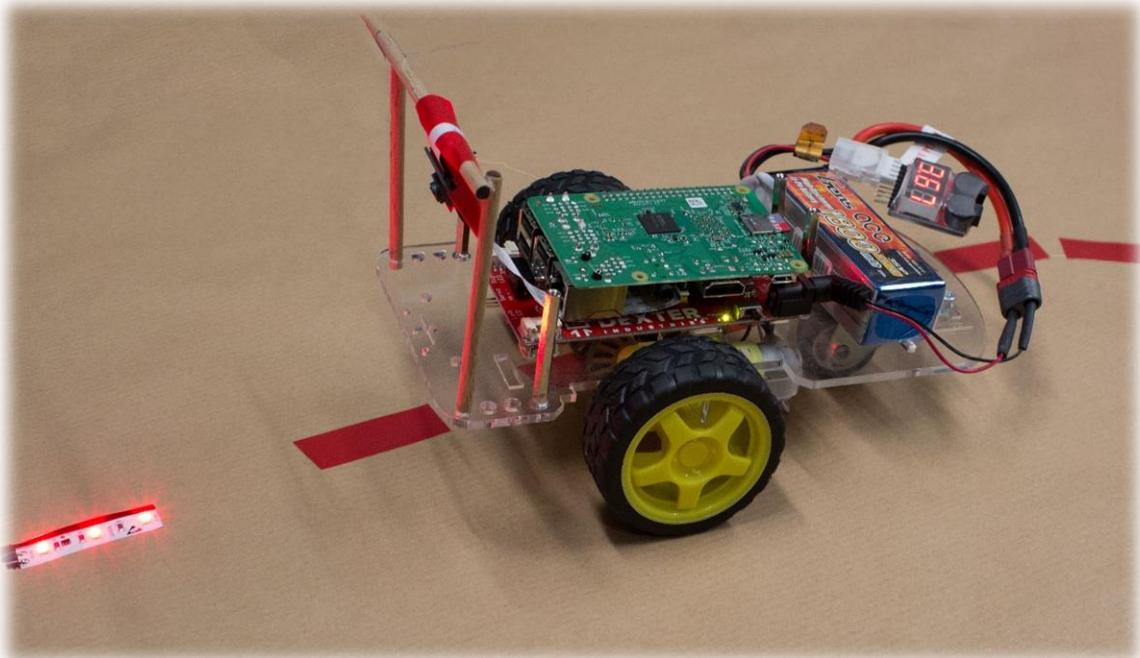


Bachelor Diplomarbeit

Autonomous Rollator

Projektdokumentation



Klassifikation: *Sperre*

Modul	TA.BA_BAA+E.F1701
Projektteam	Silvano Stecher Elektrotechnik Via da Ftan 495E 7550 Scuol silvano.stecher@stud.hslu.ch
Betreuer	Prof. Zeno Stössel, Leiter Kompetenzzentrum CC Electronics
Co-Betreuer	Martin Friedli, Senior Wissenschaftlicher Mitarbeiter iHomeLab
Experte	Thomas Schmidiger, Maxon Motor

Selbständigkeitserklärung

Hiermit erkläre ich, dass die vorliegende Arbeit selbstständig angefertigt und keine anderen als die angegebenen Hilfsmittel verwendet wurden. Sämtliche verwendeten Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Horw, 07. Juni 2017

Silvano Stecher

Abstract

Deutsch

Diese Arbeit beschreibt die Konzeptionierung und Entwicklung eines autonomen Rollators. Es wird der Abteilung Elektrotechnik der Hochschule Luzern Technik & Architektur (HSLU) in Horw als Bachelor Diplomarbeit vorgelegt. Auftraggeber dieser Arbeit ist das iHomeLab der Hochschule Luzern Technik & Architektur in Horw. Diese Arbeit beschreibt die einzelnen Schritte der Evaluation und der Durchführung. Ein Rollator ist ein gutes Hilfsmittel für Personen, die auf eine Gehunterstützung angewiesen sind. Das iHomeLab hat Anpassungen an einem Rollator vorgenommen, um das Gehen im steilen Gelände zu erleichtern. Als nächster Schritt soll der Rollator selbstständig den Weg zur Ladestation finden. Der Weg zur Ladestation wird mit Markierungen am Boden aufgezeigt. Ausgestattet mit einer Kamera und einem Raspberry Pi werden Bilder aufgenommen und verarbeitet. Die benötigten Informationen werden aus den Bildern extrahiert und für die Ansteuerung der Motoren verwendet. Für die Ansteuerung wurden passende Regler entworfen. Für die Bildverarbeitung wird die OpenCV-Bibliothek verwendet. Das ganze Projekt wird auf dem Raspberry Pi mit Python programmiert. Das Ende der Strecke wird mittels einem leuchtenden LED-Stripe markiert. Ein Ausblick mit möglichen Erweiterungen und Ideen wird zum Schluss vorgestellt.

English

This paper describes the evaluation and development of an autonomous rollator. It is realised as a Bachelor Thesis at Lucerne University of Applied Sciences and Arts (HSLU) in Horw for the iHomeLab at the HSLU. The purpose of this paper is to describe the particular steps of the evaluation and execution. A rollator is a helpful tool for people with a disability in walking. The iHomeLab has already made some adapting's to a rollator for supporting the user in case of slopes.

A next step is to get the rollator independent, so that it can find his way to the charging station. The way to the charging station is shown by markings on the ground. Equipped with a camera and a Raspberry Pi pictures are taken. In a further step these pictures are processed and the needed information is extracted. This is achieved with different algorithms of image processing which are part of the library OpenCV. The used implementation language for this project is Python. The extracted information is used for determining the correct parameters for the motors. For a correct line following a regulator is implemented. The end of the road is signed by a LED-Stripe. A prospect to some extensions and ideas is mentioned as a conclusion.

Inhaltsverzeichnis

1. AUSGANGSLAGE	11
2. ANFORDERUNGEN.....	12
2.1. Anforderungsliste	12
2.2. Streckenanforderungen.....	13
3. LÖSUNGSENTWICKLUNG	14
3.1. Recherche	14
3.2. Bewertung der Lösungen.....	16
3.3. Konzept	18
4. ROBOTER GOPIGO	19
4.1. Inbetriebnahme.....	19
4.2. Energieversorgung.....	19
4.3. Anpassungen am GoPiGo	20
5. RASPBERRY PI 3 MODEL B	21
5.1. Inbetriebnahme und Installationen	21
5.2. Programmierung.....	22
5.3. Schnittstelle zum GoPiGo.....	22
6. WEGDETEKTION.....	23
6.1. Bildaufnahme und Vogelperspektive	24
6.2. Canny-Algorithmus.....	24
6.3. Hough-Transformation und Winkelberechnung	25
6.4. Übergabe Parameter an Motoren	26
6.5. Detektion Streckenende.....	27
7. ZENTRIERTES BEFAHREN.....	29
7.1. Möglichkeiten zur Abstandsmessung.....	29

7.2. Wahl der Abstandmessung.....	30
8. REGELUNG UND KORREKTUREN.....	32
8.1. Kurvenregelung	32
8.2. Regelung zur Mittellinie.....	33
8.3. Implementierung der Regler	34
8.4. Automatische Rückfindung	36
8.5. Kontinuierliches Fahren.....	38
9. TESTS UND VERBESSERUNGEN.....	40
9.1. Drahtloser Zugriff auf Raspberry Pi	40
9.2. Verbesserung der Effizienz	41
10. ABSCHLUSS.....	42
10.1. Rückblick	42
10.2. Ausblick	43
10.3. Persönliches Fazit.....	43
11. LITERATUR- UND QUELLENVERZEICHNIS	44
A. ANHANG - ERWEITERTE PROJEKTDOKUMENTE	
B. ANHANG - ZUSÄTZLICHE INHALTLICHE UNTERLAGEN	
C. ANHANG - TESTBERICHTE	

Abbildungsverzeichnis

Abbildung 1: Rollator des iHomeLab.....	10
Abbildung 2: Gesamtkonzept des Projekts	18
Abbildung 3: Akkumulator für Energieversorgung.....	19
Abbildung 4: GoPiGo mit Kamerahalterung.....	20
Abbildung 5: Anpassung an der Lenkrolle	20
Abbildung 6: Raspberry Pi 3 Model B mit Beschreibung der Anschlüsse	21
Abbildung 7: Ausgangslage des Roboters für die Bildaufnahme.....	23
Abbildung 8: Verarbeitetes Bild mit Vogelperspektive.....	24
Abbildung 9: Verarbeitetes Bild mit Kantendetektion	24
Abbildung 10: Verarbeitetes Bild mit Linien und Winkel α	25
Abbildung 11: Konzept für ein Nachfolgeprojekt mit dem Rollator	26
Abbildung 12: Histogramm des aufgenommenen Bildes.....	27
Abbildung 13: Ergebnis der Grenzwertbestimmung.....	28
Abbildung 14: Detektion vom Ende der Strecke.....	28
Abbildung 15: Angaben zur Abstandsmessung.....	29
Abbildung 16: Ermittlung der Mittellinie	30
Abbildung 17: Erstes Beispiel für die Abstandsmessung	30
Abbildung 18: Zweites Beispiel für die Abstandsmessung	31
Abbildung 19: Drittes Beispiel für die Abstandsmessung	31
Abbildung 20: Geschlossener Regelkreis des Systems	32
Abbildung 21: Regelung zur Mittellinie Rechtskurve	33
Abbildung 22: Regelung zur Mittellinie Linkskurve.....	33
Abbildung 23: Regelung zur Mittellinie Gerade	34
Abbildung 24: Automatische Rückfindung Fall 1 und Fall 2	36
Abbildung 25: Automatische Rückfindung Fall 3 und Fall 4	36
Abbildung 26: Detailansicht des Roboters mit Fahrbahn	38
Abbildung 27: Darstellung des Ringbuffers in Kreis- und Matrizenform	38
Abbildung 28: Ausschnitt des RaspiTerminals	40
Abbildung 29: Zugriff auf das Raspberry Pi mittels Putty	40

Tabellenverzeichnis

Tabelle 1: Änderungsprotokoll des Dokumentes	8
Tabelle 2: Begriffe / Abkürzungen.....	9
Tabelle 3: Bewertungstabelle Lösungsansätze.....	14
Tabelle 4: Bewertung Hardware für die Bildverarbeitung	16
Tabelle 5: Bewertung Kamera.....	16
Tabelle 6: Bewertung Software/Bibliothek für die Bildverarbeitung	17
Tabelle 7: Bewertung Programmiersprache	17
Tabelle 8: Automatische Rückfindung mittels der Abstandsmessung	37

Dokumentversion

Version	Änderungsdatum	Autor	Kommentar
0.1	21.02.2017	Silvano Stecher	Erstellung
0.2	21.05.2017	Silvano Stecher	Erste Überarbeitung
1.0	06.06.2017	Silvano Stecher	Endversion

Tabelle 1: Änderungsprotokoll des Dokumentes

Begriffe und Abkürzungen

Begriff / Abkürzung	Erklärung
Raspberry Pi, Raspi	Raspberry Pi 3 Model B
RaspiCam	Raspberry Pi Camera V2
GoPiGo	Ersatzroboter für Rollator
PixyMon	Software für die CMUcam5 Pixy
PixyCam	CMUcam5 Pixy
Open CV	Bibliothek für die Bildverarbeitung
Python Shell	Konsole für die Pythonprogrammierung
cm	Center of Mass (Schwerpunkt eines Körpers)
OS	Operating System (Betriebssystem)
EBV	Modul Echtzeitbildverarbeitung an der HSLU

Tabelle 2: Begriffe / Abkürzungen

Einleitung

Die vorliegende Bachelor Diplomarbeit *Autonomous Rollator* beinhaltet die Umsetzung einer autonomen Wegfindung eines Rollators. Es ist eine Arbeit, die an der Hochschule Luzern Technik & Architektur in Horw durchgeführt wird. Der Auftraggeber dieser Arbeit ist das iHomeLab der Hochschule Luzern in Horw.

Der Rollator findet seinen Einsatz vor allem bei älteren Leuten die eine Gehunterstützung benötigen. Durch mehrere Anpassungen am Rollator kann dieser zu einem höheren Komfort führen.

In der Vergangenheit wurde dies durch den Einbau von Motoren realisiert. Bei einer Steigung unterstützen diese den Anwender.

Eine andere Anpassung ist die Ansteuerung des Rollators mittels eines Joysticks.

Die vorliegende Arbeit stellt den ersten Schritt zur selbständigen Wegfindung des Rollators vom iHomeLab dar. Der Rollator soll dabei selbstständig eine markierte Strecke abfahren und an dessen Ende anhalten. Am Ende der Strecke kann sich eine Ladestation befinden, wo sich der Rollator aufladen kann. In diesem Projekt wird auf die Wegfindung und korrekte Parameterübergabe eingegangen. Zusätzlich sollen passende Regler implementiert werden. Ein Bild des Rollators ist in Abbildung 1 ersichtlich.



Abbildung 1: Rollator des iHomeLab

Da dieses Projekt auf einer vorhandenen Studentenarbeit basiert, wird im Kapitel 1 die Ausgangslage beschrieben. Im Kapitel 2 werden die Anforderungen an das Projekt aufgezeigt. Aus den Anforderungen entstand eine Recherche mit anschliessender Evaluation und Bewertung.

Mittels der Evaluation wurde ein Gesamtkonzept erstellt. Im Kapitel 4 und 5 folgen Beschreibungen der Hardware und deren wichtigsten Angaben. Im Kapitel 6 wird auf die Wegdetektion eingegangen und wie diese realisiert wird. Das Kapitel 7 stellt das zentrierte Befahren der Strecke vor. Damit der Rollator eine Strecke zuverlässig abfahren kann wird noch ein Regler benötigt. Dieser wird im Kapitel 8 beschrieben. Angaben zur Durchführung der Tests und allfällige vorgenommene Verbesserungen, werden im Kapitel 9 besprochen. Der Abschluss bildet ein Rückblick, ein Ausblick und ein persönliches Fazit.

1. Ausgangslage

Als Ausgangslage für dieses Projekt dient das Projekt „Rollator Remote“ vom iHomeLab. Das Projekt „Rollator Remote“ wurde im Herbstsemester 2016 als Industriearbeit von Fabian Niederberger durchgeführt. Auftraggeber war das iHomeLab der Hochschule Luzern Technik & Architektur. Die Aufgabe bestand darin, den Rollator über Bluetooth ansteuern zu können. Während des Projekts „Rollator Remote“ wurde die Regelstrecke aufgenommen, eine passende Hardware erstellt, ein Regler entworfen und die Ansteuerung implementiert.

Mit dem Projekt „Rollator Remote“ als Ausgangslage soll der Rollator autonom einem Weg folgen können. Die Idee besteht darin, dass der Rollator selbstständig den Weg zur Ladestation findet, um seinen Akku dort aufzuladen. Der Weg ist mittels LEDs oder einer rot markierten Strecke vorgegeben. Die Erkennung der Strecke soll mittels einer Kamera realisiert werden. Nach der Auswertung des Bildes werden passende Werte einem Regler übergeben. Dieser sorgt für die korrekte Befahrung der Strecke.

Da das Befahren einer Strecke voraussichtlich im Innenbereich stattfindet, sollen die Messungen und die Funktionstests drinnen stattfinden.

Aufgrund einer technischer Störung am vorhandenem Rollator konnte dieser für dieses Projekt nicht verwendet werden. Ein passendes Ersatzgerät wurde während dem Verlauf des Projekts im Roboter GoPiGo von Dexter Industries gefunden. Mehr zum GoPiGo folgt im Kapitel 4.

2. Anforderungen

Um eine zielorientierte Durchführung des Projekts zu gewährleisten, wurden Anforderungen an das Produkt gestellt. Diese dienen als Richtlinien, um bei allfälligen Unstimmigkeiten darauf zurückgreifen zu können. Zudem sind sie ein Hilfsmittel bei der Evaluierung von möglichen Lösungen. Im Anhang befinden sich die Dokumente *Anforderungsspezifikation* und *Streckenspezifikationen*. Darin sind alle Anforderungen und Masse für die Strecke detailliert aufgeführt.

2.1. Anforderungsliste

Die Anforderungen an das Produkt wurden mit dem Auftraggeber vereinbart und werden hier nun vorgestellt.

Allgemeine Anforderungen:

- vorhandene Hardware nutzen
- Wegmarkierung detektieren und folgen
- im Innenbereich anwendbar
- passende Parameterübergabe für Roboter
- wenn keine Spur: anhalten, sobald eine detektiert; dieser folgen

Roboteransteuerung:

- vorhandenen Roboter verwenden

Schnittstelle:

- Werteübergabe analog oder direkt implementiert

Richtwerte Befahrung:

- stabiles Verhalten
- startet sobald Strecke erkannt wird
- stoppt wenn Ende oder keine Wegmarkierung erkannt wird
- Autokorrektur beim Verlassen der Strecke

Umgebungsbedingungen:

- Steigung <1%
- haushaltliche Bedingungen, keine Stufen oder Hindernisse
- kleinere Störungen/Reflexionen dürfen vorhanden sein

Wegmarkierung:

- LED-Streifen mit roten LEDs markieren das Ende der Strecke
- Max. Länge 10m
- direkter Weg ins Ziel ohne Kreise oder Kreuzungen
- rote Klebestreifen
- maximaler Radius von 35cm

Wunschvorstellung:

- verschiedene Befehle verarbeiten (Ein/Aus, Wegmarkierung suchen)
- Gehgeschwindigkeit und flüssiges Fahren
- Vorhersage Strecke
- Störungen/Fehldetektionen ausblenden
- selbständig nach Spur suchen

2.2. Streckenanforderungen

Die Streckenanforderungen bilden die Richtlinien für die Verlegung der Strecke. Diese Richtwerte wurden in gegenseitiger Absprache mit dem Auftraggeber besprochen.

Es ergeben sich folgende wichtige Punkte:

- Es soll ein direkter Weg befahren werden
- Die Strecke soll keine Kreuzungen und eckige Kurven aufweisen
- Es sollen keine Hindernisse auf der Strecke vorhanden sein
- Die Strecke wird mit roten Klebestreifen markiert
- Als Störungen können Reflexionen auftreten

In Absprache mit dem iHomeLab entstanden folgende Richtwerte:

- Die Klebestreifen sollen rot und mindestens 50mmx18mm gross sein
- In geraden Abschnitten sind die Abstände zwischen den Klebestreifen auf 50mm bis 80mm begrenzt
- In den Kurven soll maximal ein Abstand von 30mm vorhanden sein, empfohlen wird 10mm bis 20mm
- Das Ende der Strecke wird mit einem leuchtendem LED-Stripe markiert
- Der Kurvenradius darf nicht kleiner als 350mm sein

Eine Abbildung mit allen Angaben befindet sich im Anhang A.

3. Lösungsentwicklung

Mit den erstellen Anforderungen wurden potentielle Lösungen evaluiert. Dabei wurde die Problemstellung in vier Teilbereiche unterteilt. In jedem dieser Teilbereiche wurden Recherchen durchgeführt. Die Ergebnisse der Recherche sind im Abschnitt 3.1 aufgeführt. Nach der Recherche wurden die gesammelten Informationen mit Hilfe der Tabelle 3 bewertet. Das Ergebnis der Bewertung befindet sich im Unterkapitel 3.2. Nach der Auswahl der Lösungen für die einzelnen Teilbereiche wurde ein Gesamtkonzept erstellt. Dieser wird im Unterkapitel 3.3 vorgestellt.

3.1. Recherche

Für das Projekt wurde, wie bereits oben genannt, die allgemeine Problemstellung in vier Teilbereiche unterteilt. Diese sind: Hardware für die Bildverarbeitung, Wahl einer passenden Kamera, verwendete Software/Bibliothek und die Programmiersprache. In diesen Teilbereichen wurden Recherchen durchgeführt und die Ergebnisse gesammelt und ausgewertet. Eine grobe Zusammenfassung der Recherche befindet sich im folgenden Abschnitt.

#	Kriterium	Abkürzung	Gewichtung	Beschreibung
1	Machbarkeit	M	4	Aufwand, Ertrag, Einfachheit
2	Kosten	K	1	Höhe der Kosten
3	Flexibilität	F	2	Ausbaubarkeit mit zusätzlichen Produkten/Funktionen
4	Lieferung	L	2	Lieferzeit der Produkte / Installation
5	Innovation	I	3	Wie innovativ ist die Verwendung dieses Produkts

Tabelle 3: Bewertungstabelle Lösungsansätze

Die Bewertung eines Konzepts ist mit folgender Formel definiert:

$$\text{Bewertung} = \sum_{n=1}^{\text{Anzahl Kriterien}} \text{Bewertung n} \cdot \text{Gewichtung}$$

Bewertungskriterien:

Werte von 1-5

1 nicht gut

5 sehr gut

Bevorzugte Lösungsvariante
Ausweichvariante 1
Ausweichvariante 2

Die Ergebnisse der Recherche werden im folgenden Abschnitt präsentiert.

Hardware für Bildverarbeitung

Produkt	Beschreibung	Positives	Negatives
FRDM-KL25Z	Der FRDM-KL25Z ist eine Entwicklungsplattform für Kinetis. Der Mikrocontroller basiert auf einem ARM Cortex-M0+.	+ Echtzeitfähig (FreeRTOS) + Programmierumgebung bekannt + Debuggmöglichkeit	- Schnittstelle muss extra implementiert werden
Arduino Mega	Das Arduino Mega ist ein Entwicklungsboard basierend auf einem ATmega2560. Die Implementierung kann im AtmelStudio oder Arduino IDE stattfinden.	+ kompatibel mit PixyCam + sehr einfach + grosse Community + alles in einem (Motoren & Kamera)	- Keine Debuggmöglichkeit
Raspberry Pi 3 Model B	Der Raspberry Pi 3 ist ein Mini-Computer. Er besitzt einen 1.2GHz Quad-Core-Prozessor ARM Cortex-A53 64-Bit.	+ sehr mächtig + OpenCV + kompatibel Raspberry Pi Kamera V2 + kompatibel mit Pixy	- Aufwendig - wenig Erfahrung - overkill je nach Kamera

Kamera

Produkt	Beschreibung	Positives	Negatives
CMUCam5 Pixy	Ist eine leicht anwendbare Kamera mit der man Farben von Objekten detektieren kann. Ist mit Arduino und Raspberry Pi kompatibel. Kann über verschiedene Schnittstellen angesteuert werden. Output ist 50fps.	+ einfach + gute Community	- beschränkt in Anwendung - nur Farbdetektion
Raspberry Pi Camera V2	Zweite Generation Raspberry Pi Modul mit 8 Megapixel und integrierter Focus Linse. Video mit 1080p30, 720p60 und 640x480p90 möglich.	+ viele Möglichkeiten + gute Community	- Aufwendig - wenig Erfahrung

Software/Bibliothek für Kamera

Produkt	Beschreibung	Positives	Negatives
PixyMon	PixyMon ist die verwendete Software für die Ansteuerung der CMUCam5 Pixy. PixyMon ist Linux, Mac und Windows kompatibel.	+ einfach + keine Implementierung + gute Community	- beschränkt in Anwendung - nur Farbdetektion
OpenCV	Bibliothek für Bildverarbeitung. Wird oft in Kombination mit Raspberry Pi und Raspberry Pi Kamera verwendet.	+ viele Möglichkeiten + gute Community	- Aufwendig - wenig Erfahrung

Programmiersprache

Produkt	Beschreibung	Positives	Negatives
Arduino IDE	Arduino Entwicklungsumgebung. Die Implementierung und Handhabung ist sehr einfach gestaltet.	+ einfach + gute Community + sehr verbreitet	- kein Debugging möglich - Hobbyanwendungen
C/C++	C ist eine weltweit verbreitete Programmiersprache. Der Anwendungsbereich ist sehr unterschiedlich.	+ sehr verbreitet	- kommt nicht mit Raspberry Pi
Python	Python ist eine mächtige Programmiersprache mit effizienten Datenstrukturen. Sie ist automatisch auf jedem Raspberry Pi mit OS Raspbian vorinstalliert.	+ sehr verbreitet	- Aufwendig - wenig Erfahrung - overkill je nach Kamera

3.2. Bewertung der Lösungen

Die Bewertung der möglichen Lösungsansätze wurde mittels der Tabelle 3 durchgeführt. Am Schluss dieses Unterkapitels wird ein kurzes Fazit gezogen, wieso welcher Lösungsansatz gewählt wurde.

Hardware für die Bildverarbeitung

Kriterium	FRDM-KL25Z	Arduino Mega	Raspberry Pi Model 3 B
Machbarkeit	3	4	5
Kosten	5	3	3
Flexibilität	2	4	4
Lieferung	5	4	4
Innovation	3	2	3
Gesamt	39	41	48

Tabelle 4: Bewertung Hardware für die Bildverarbeitung

Kamera

Kriterium	CMUCam5 Pixy	Raspberry Pi Camera V2
Machbarkeit	3	3
Kosten	2	3
Flexibilität	3	3
Lieferung	5	3
Innovation	2	4
Gesamt	36	39

Tabelle 5: Bewertung Kamera

Software/Bibliothek für die Bildverarbeitung

Kriterium	PixyMon	OpenCV
Machbarkeit	3	3
Kosten	5	5
Flexibilität	3	4
Lieferung	5	4
Innovation	2	4
Gesamt	39	45

Tabelle 6: Bewertung Software/Bibliothek für die Bildverarbeitung

Programmiersprache

Kriterium	Python	Arduino IDE	C/C++
Machbarkeit	4	2	4
Kosten	5	5	5
Flexibilität	4	2	4
Lieferung	5	5	4
Innovation	4	2	4
Gesamt	51	33	49

Tabelle 7: Bewertung Programmiersprache

Fazit:

Hardware für die Bildverarbeitung

Das beste Ergebnis für die Hardware erzielte das Raspberry Pi 3 Model B. Durch seine hohe Flexibilität und guter Rechenleistung kann es in vielen Bereichen eingesetzt werden. Es bietet die Möglichkeit, die CMU5cam Pixy zu verwenden oder aber auch die Raspberry Pi Camera V2. Die Kombination aus OpenCV und Raspberry Pi bildet eine gute Hardware für die Bildverarbeitung.

Kamera

Die Wahl für die Kamera fiel auf die Raspberry Pi Camera V2. Die Inbetriebnahme der PixyCamera gestaltete sich als schwierig. Zudem waren die erhaltenen Informationen aus den Bildern sehr beschränkt brauchbar. Aus diesem Grund wird empfohlen, die Raspberry Pi Camera V2 zu verwenden. Zusammen mit dem Raspberry Pi und OpenCV sind die Möglichkeiten sehr gross.

Software/Bibliothek für die Bildverarbeitung

Eines der meist bekannten Bildverarbeitungsbibliotheken ist das OpenCV. Mit den vorhandenen Befehlen dieser Bibliothek lassen sich Bilder effizient und einfach bearbeiten. Im Gegensatz zu PixyMoon, bietet OpenCV ein breites Mass für die Gestaltung und Implementierung an. Darum wird die Verwendung mit OpenCV empfohlen.

Programmiersprache

Bei der Wahl der zu verwendenden Software zeigte sich, dass Python die beste Wahl ist. Diese Hoch-Programmiersprache wird nicht nur bei der Bildverarbeitung verwendet, sondern auch bei vielen anderen Anwendungen. Dadurch das Python auf dem OS Raspbian Pixel schon vorinstalliert ist, besitzt dieser einen minimalen Vorteil gegenüber C/C++.

3.3. Konzept

In den vorangehenden zwei Unterkapitel wurde die Recherche und die Bewertung der Lösungsansätze vorgestellt. Im folgenden Unterkapitel wird das ganze Konzept und alle Zusammenhänge aufgezeigt. In Abbildung 2: *Gesamtkonzept des Projekts* ist das Konzept graphisch dargestellt.

Das aktuelle Konzept gewährleistet, dass nur wenige Anpassungen, für die spätere Verwendung mit dem Rollator, nötig sind.

Genauere Angaben bezüglich der einzelnen Komponenten werden im weiteren Teil des Dokuments beschrieben.

Für die ganze Energieversorgung wird ein Akkumulator verwendet. Die gewählte Kapazität soll den Roboter eine Stunde betreiben können. Das Raspberry Pi kann auf den GoPiGo gesteckt werden. Dabei wird das Raspi automatisch via Steckanschluss des GoPiGo mit Strom versorgt. Die Implementierung auf dem Raspberry Pi wird mit Python vorgenommen. Damit die Bildverarbeitung realisiert werden kann, wird die OpenCV Bibliothek in Python hinzugefügt. Bilder für die Bildverarbeitung erhält das Raspberry Pi von der Raspberry Pi Camera V2. Diese ist mit einem Flachbandkabel mit dem Raspi verbunden. Die DC-Motoren des GoPiGos werden mit dem Raspberry Pi angesteuert.

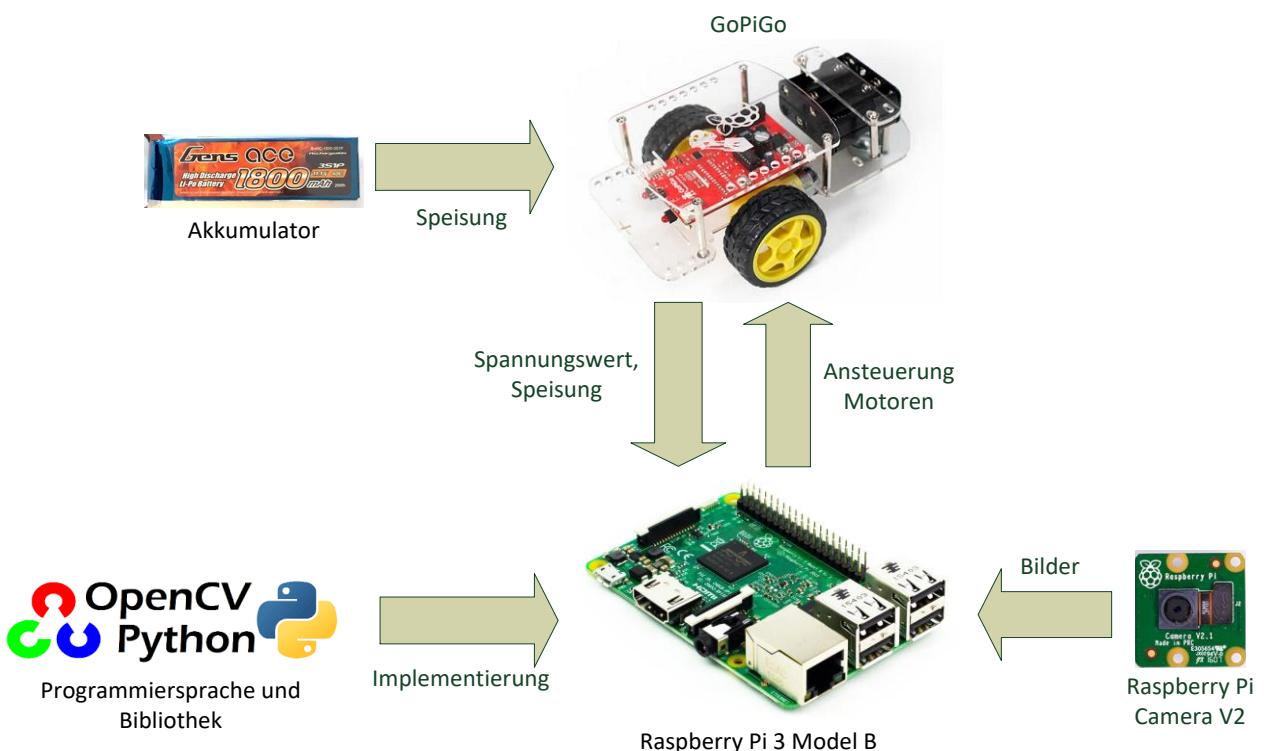


Abbildung 2: *Gesamtkonzept des Projekts*

4. Roboter GoPiGo

Nachdem sich herausstellte, dass der Rollator während des ganzen Projekts ausfällt, wurde ein passender Ersatz im GoPiGo von Dexter Industries gefunden. Durch seinen einfachen Aufbau und seiner leichten Ansteuerung mussten nur wenige Anpassungen vorgenommen werden. Im folgenden Abschnitt wird auf die Inbetriebnahme des GoPiGos eingegangen, wie die Energieversorgung stattfindet und welche Anpassungen durchgeführt wurden.

4.1. Inbetriebnahme

Das GoPiGo musste zu Beginn zuerst zusammengebaut werden. Die Aufbauanleitung findet man im Internet auf der Herstellerseite. Diese liegt in schriftlicher Form vor oder ist über Video abrufbar. Nach dem Aufbau musste auf dem Raspberry Pi die GoPiGo Python Bibliothek installiert werden (Link im Quellenverzeichnis unter *Links für die Installation*).

Nach der Installation der Bibliothek war das GoPiGo fertig für die erste Ansteuerung. Um nun zu überprüfen, ob der Roboter angesteuert werden kann, wurden in der Python Shell folgende Befehle eingegeben: `from gopigo import *, fwd() und stop()`. Nach dieser Eingabe drehten sich die Räder und somit war die Inbetriebnahme erfolgreich.

4.2. Energieversorgung

Das GoPiGo kann mit einer Spannung von 9 bis 12 Volt betrieben werden. Aus diesem Grund versorgt ein 11.1V/1800mAh Lithium-Polymerakkumulator den Roboter. Bei der Auslegung wurde darauf geachtet, dass der Roboter eine Stunde lang betrieben werden kann. Der Akkumulator besitzt 3 Zellen à je 3.7V. Der Akkumulator wird auf der hinteren Seite des Roboters mittels eines Klettverschlusses befestigt. Die Berechnung für die Wahl des Akkumulators befindet sich im Anhang.

Für das Aufladen des Akkus wird ein Ladestrom von 1C bis 3C empfohlen. 1C entspricht den 1800mAh. Vom iHomeLab hat man eine passende Ladestation (*reactor 500*) erhalten. Damit kann der Akku nach Gebrauch wieder aufgeladen werden. Ein Spannungsüberwachungs-Modul am Balancer zeigt die aktuellen Zellenspannungen an. Bei zu niedriger Spannung wird der Buzzer aktiviert und signalisiert, dass die Zellenspannung einen tiefen Spannungswert erreicht hat.

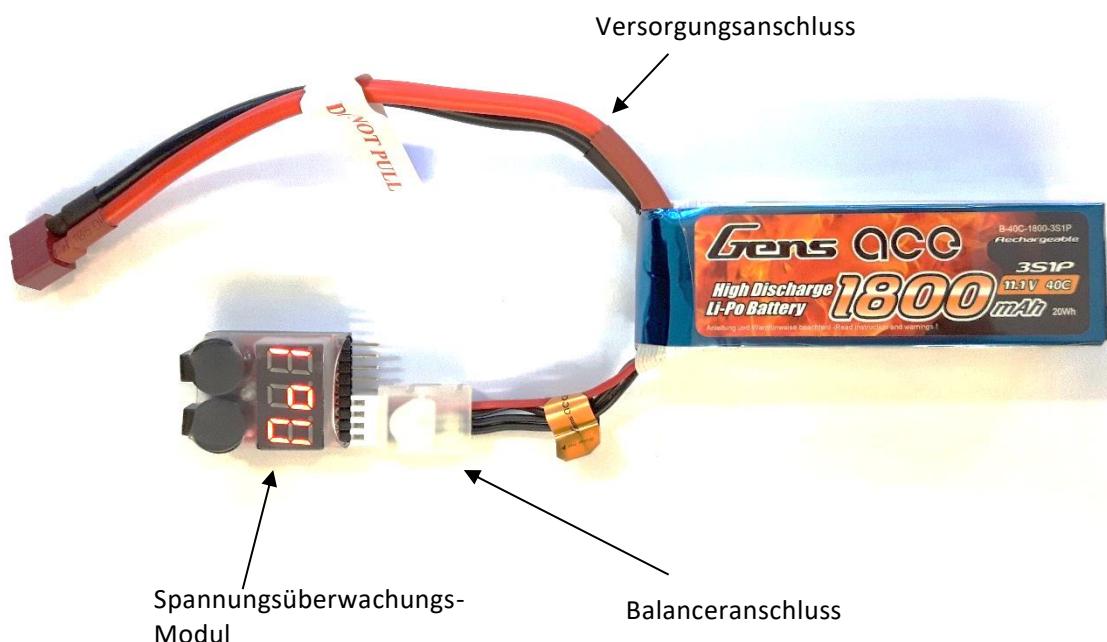


Abbildung 3: Akkumulator für Energieversorgung

4.3. Anpassungen am GoPiGo

Damit der Batterieverbrauch nicht zu hoch ausfällt, wurde beschlossen, die Batteriehalterung durch einen Akku zu ersetzen. Der Akku findet, wie im Bild ersichtlich, auf der hinteren Seite des Roboters Platz. Zudem wird am Balanceranschluss des Akkus ein Spannungsüberwachungsmodul angeschlossen.

Damit die Raspicam einen guten Aufnahmebereich hat, wird eine Halterung erstellt. Diese befindet sich auf der Vorderseite des GoPiGos. Dabei kann die Kamera nicht zentriert montiert werden, da sonst das Flachbandkabel durch ein Knicken beschädigt werden könnte.

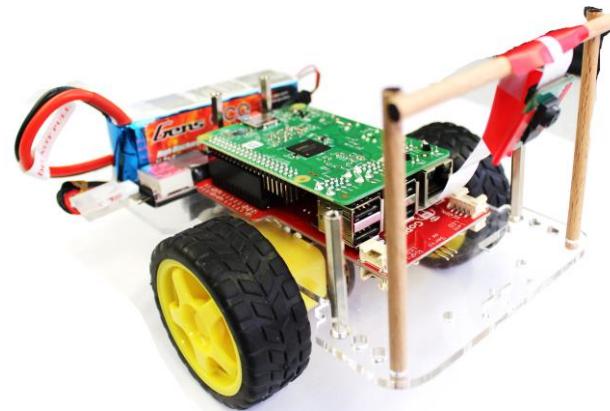


Abbildung 4: GoPiGo mit Kamerahalterung

Die mitgelieferte Kugelrolle, die üblicherweise montiert wird, besitzt eine elektrisch leitende Kugel. Da die Wegmarkierung zum Teil aus LED-Stripes besteht, kann es sein, dass beim Überqueren eines Stripes ein Kurzschluss entstehen kann. Mittels einer Lenkrolle aus Kunststoff wird dieses Problem behoben.

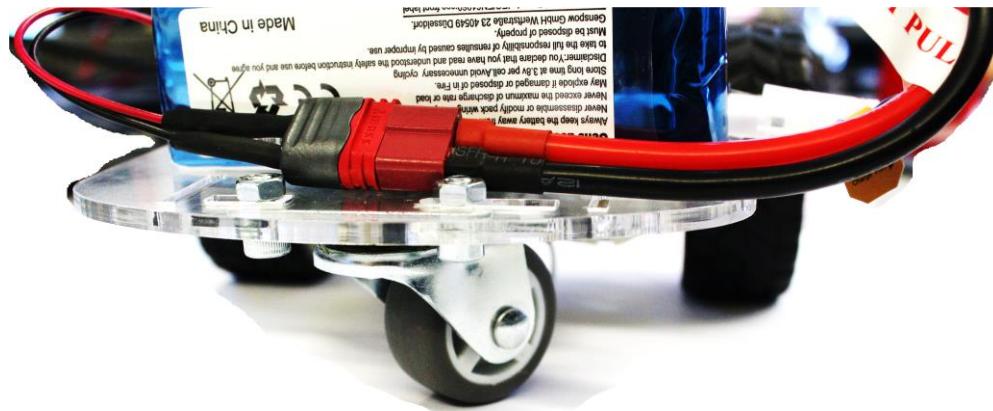


Abbildung 5: Anpassung an der Lenkrolle

5. Raspberry Pi 3 Model B

Als Hardware für das Projekt, wird das Raspberry Pi 3 Model B verwendet. Das Raspberry Pi inklusive einer 16GB SD-Karte wurde vom iHomeLab zur Verfügung gestellt. Das verwendete Betriebssystem ist Raspbian PIXEL. Dieses kann von der offiziellen Raspberry Pi Seite heruntergeladen werden. Das Raspbian PIXEL kommt mit einigen vorinstallierten Programmen. Dazu gehört auch das zum Programmieren benötigte Programm Python mit der Version 3.0.0 und 2.7.9.

5.1. Inbetriebnahme und Installationen

Der Zugriff auf das Pi ist über verschiedene Arten möglich (SSH, Remotedesktop). Während dem Projekt wurde mit einem Monitor, einer Tastatur und einer Maus auf dem Raspberry Pi gearbeitet. Diese Peripherien können an den vorhandenen Anschlüssen angeschlossen werden. Die Internetverbindung kann mittels WLAN oder Ethernet-Anschluss erreicht werden. Die Raspberry Pi Kamera findet an einer der beiden Flachbandkabelanschlüsse mit der Bezeichnung *Camera Platz*. Die SD-Karte mit dem Betriebssystem befindet sich auf der Seite des Raspberry Pi. Die Energieversorgung kann mittels Micro-USB-Anschluss erfolgen. Im Betrieb kann das Raspberry Pi bis zu 1.5A ziehen.

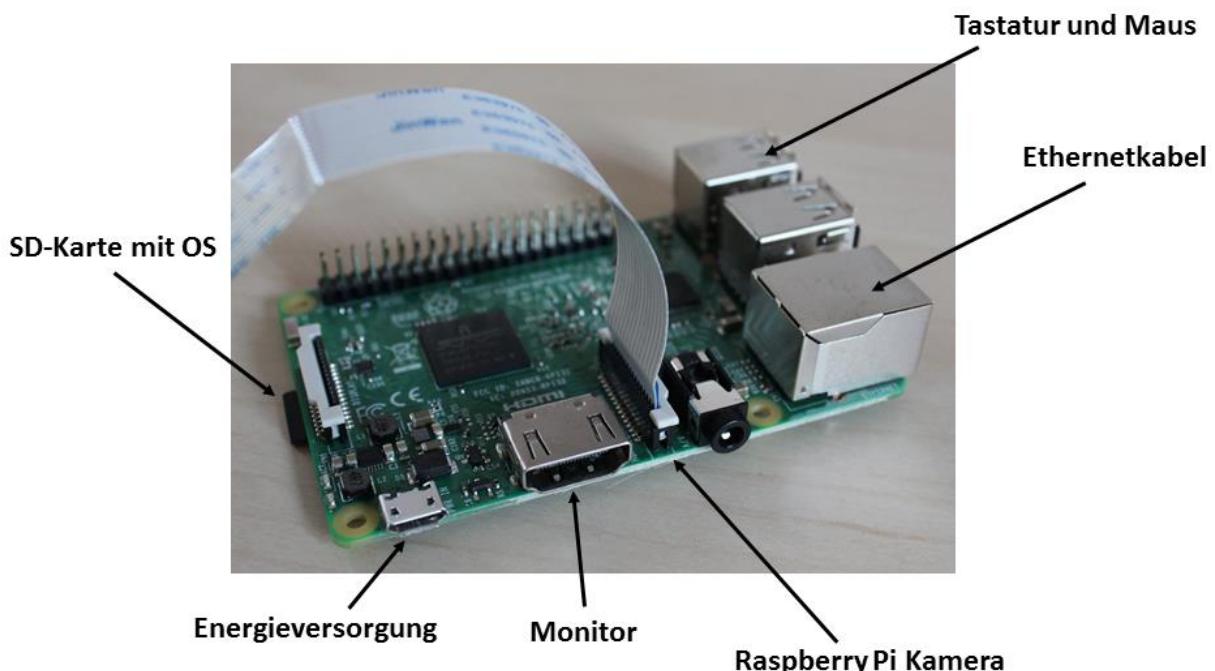


Abbildung 6: Raspberry Pi 3 Model B mit Beschreibung der Anschlüsse

Um auf das Raspberry Pi zugreifen zu können, benötigt man ein Login. Dieses wird beim Hochfahren, also wenn der Bildschirm angeschlossen ist, nicht benötigt. Für den Zugriff via SSH wird das Login benötigt. Das Login lautet wie folgt:

Benutzername: pi
Passwort: BDA17

Für die Bildverarbeitung und Implementierung mussten auf dem Raspi noch zusätzliche Installationen vorgenommen werden. Dazu gehören:

- OpenCV 3.2 für Python 2.7.9, Bibliothek für die Bildverarbeitung
- Numpy, Bibliothek für einfache Vektoren-, Matrizen- und Arrayberechnungen
- Matplotlib 2.0, Bibliothek für die Erstellung von Plots und Figuren
- GoPiGo, Bibliothek für die Ansteuerung des Roboters
- Imutils, Python-Package mit Funktionen für die Bildverarbeitung

Die Installation von OpenCV und Numpy kann mehrere Stunden in Anspruch nehmen. Es ist zu erwähnen, dass bei der Installation von OpenCV die virtuelle Umgebung (virtual Environment) für Python nicht erstellt wurde. Die Programmierung kann ohne diese realisiert werden. Die Links für alle Installationen befinden sich im Quellenverzeichnis.

5.2. Programmierung

Damit nicht immer das Raspberry Pi aufgeschaltet werden muss, wurde Python 2.7.13 inklusive allen Programmen (OpenCV 3.2, Numpy und Matplotlib) auf dem Computer installiert. Durch diese Umsetzung kann ein hohes Mass an Flexibilität erreicht werden. Zuerst wurde der Code auf dem PC erstellt und getestet. Anschliessend wurde dieser auf das Raspberry Pi übertragen.

Da der Verlust von Daten immer wieder vorkommen kann, wurden fortlaufend Backups erstellt. Dafür wurde mit SourceTree und Github gearbeitet. Somit wäre es möglich, auf alte Versionen zurückzugreifen, falls aktuelle Codeteile nicht mehr passen oder nicht mehr vorhanden sind. Zusätzlich wurden Backups der SD-Karte erstellt.

5.3. Schnittstelle zum GoPiGo

Das Raspberry Pi wird mittels der vorhandenen Stiftleiste auf den GoPiGo gesteckt. Über diese Stiftleiste erfolgt die Versorgung und die Kommunikation des Raspberry Pi.

Für die Anwendung der Arbeit auf dem Rollator kann die Kommunikation über die Stiftleisten oder via seriellen Schnittstelle (z.B USB) realisiert werden. Hier befindet sich die Schnittstelle zwischen dem Rollator und der Bildverarbeitung.

6. Wegdetektion

Die Wegdetektion ist ein essentieller Bestandteil dieser Arbeit. Sie ist die Schnittstelle zwischen dem inneren System und den äusseren Umständen. Die äusseren Umstände beinhalten die Strecke mit den vorhandenen Wegmarkierungen und Reflexionen. Das innere System stellt das Raspberry Pi und das GoPiGo dar. Um die benötigten Informationen aus den erhaltenen Bildern zu extrahieren, müssen gewisse Verarbeitungen vorgenommen werden. Diese sehen folgendermassen aus:

- Bild wird aufgenommen
- Ausschnitt und Vogelperspektive
- Canny-Algorithmus um Kanten zu detektieren
- Hough-Transformation um Linien zu bilden
- Winkelberechnung für die Kurvenbefahrung
- Übergabe Parameter an Motoren
- Ende der Strecke ermitteln mittels der Helligkeit

Für die Kurvenbestimmung durch die Kantendetektion und der anschliessenden Linienbildung ergeben sich gewisse Vorteile. Ein Vorteil liegt in der Immunität gegenüber Reflexionen. Ein weiterer Vorteil liegt in der Flexibilität. Grundsätzlich können irgendwelche Streifen detektiert werden. Es müssen einfach klare gerade Kanten vorhanden sein. Das ist auch zugleich der Nachteil. Für die Befahrung sind deutliche Kanten notwendig.

Die im folgenden Abschnitt verwendeten Bilder sind aus folgender Ausgangslage entstanden.

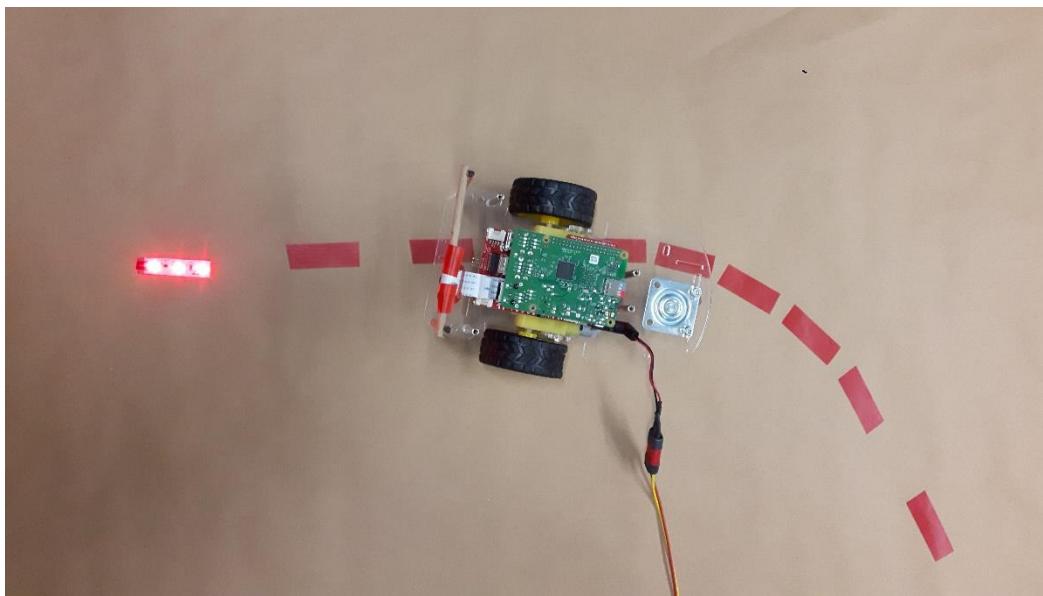


Abbildung 7: Ausgangslage des Roboters für die Bildaufnahme

6.1. Bildaufnahme und Vogelperspektive

Die RaspiCam nimmt Bilder auf und sendet diese am Raspi. Da das aufgenommene Bild unnötige oder sogar störende Informationen beinhalten kann, ist es empfehlenswert, es auf den gewünschten Bereich zuzuschneiden.

Ein weiterer wichtiger Punkt für die wahrheitsgetreue Aussage der Daten ist die Ansicht, respektive die Aufnahme der Strecke. Um dieses Mass hoch zu halten wird eine Ansicht der Strecke von oben erwünscht.

Beide Aufgaben können auf einmal durchgeführt werden. Dies geschieht mit der Perspektivfunktion. Dafür werden im Bild vier Punkte definiert. Aus dem entstandenen Bereich wird dann die Perspektive erstellt. Das Ergebnis ist in *Abbildung 8: Verarbeitetes Bild mit Vogelperspektive* zu sehen. Die Zahlen links und unterhalb des Bilds stellen die X und Y-Koordinaten dar und somit auch die Auflösung.

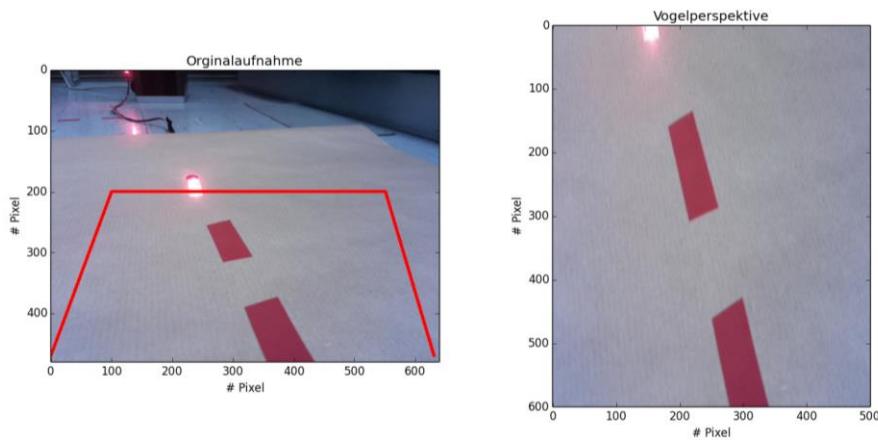


Abbildung 8: Verarbeitetes Bild mit Vogelperspektive

6.2. Canny-Algorithmus

Um später die Hough-Transformation durchführen zu können, müssen zuerst die Kanten der Klebestreifen gefunden werden. Dafür wird der Canny-Algorithmus verwendet. Der Canny-Algorithmus ähnelt sehr einem Gradienten-Filter. Für diesen Algorithmus wird aus dem Originalbild ein graues Bild erstellt. Die Grauwerte im Bild besitzen einen Wert zwischen 0 und 255. 255 entspricht einem weißen Pixel und 0 einem schwarzen. Der Canny-Algorithmus schaut nun wo grosse Sprünge bei den Grauwerten stattfinden und zeigt diese im Bild an. Die Bedingungen für die Kantendetektion werden über einen unteren und oberen Schwellwert für die Hysterese eingestellt. Das Ergebnis des Canny-Algorithmus ist in *Abbildung 9: Verarbeitetes Bild mit Kantendetektion* ersichtlich.

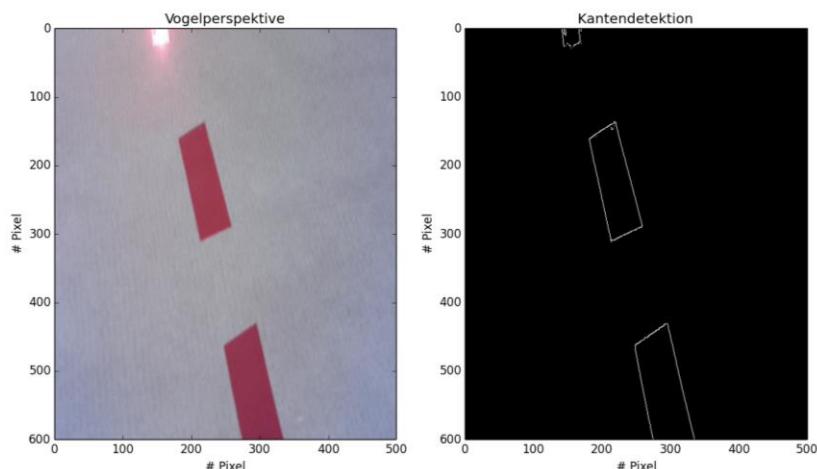


Abbildung 9: Verarbeitetes Bild mit Kantendetektion

6.3. Hough-Transformation und Winkelberechnung

Die Hough-Transformation bildet einen zentralen Schritt für die aktuell ausgewählte Methode. Bei der Hough-Transformation geht der Algorithmus Pixel für Pixel durch und schaut, ob sich eine Linie bilden lässt. Dafür wird das Bild des Canny-Algorithmus benötigt.

Die Hough-Transformation im OpenCV gibt den Start- und Endpunkt der jeweiligen Linien zurück. Mit diesen Angaben wird der Winkel trigonometrisch berechnet. Die Variable h stellt die Gegenkathete und die Variable b die Ankathete des Winkels α dar. Die Hypotenuse wird mit l ausgedrückt.

$$h = y_2 - y_1$$

$$b = x_2 - x_1$$

$$l = \sqrt{b^2 + h^2}$$

Berechnung von α in Grad:

$$\alpha = \arcsin\left(\frac{h}{l}\right) \cdot \frac{180}{\pi}$$

Falls mehrere Linien erkannt werden, wird der Median aller Winkel ermittelt. Dadurch fallen grosse Ausreisser weniger ins Gewicht.

Der erhaltene Winkel α kann ab hier unabhängig vom verwendeten Roboter an die Motorensteuerung weitergeleitet werden. Im weiteren Verlauf der Dokumentation bezieht sich der Winkel α immer auf den Winkel für die Kurvenerkennung.

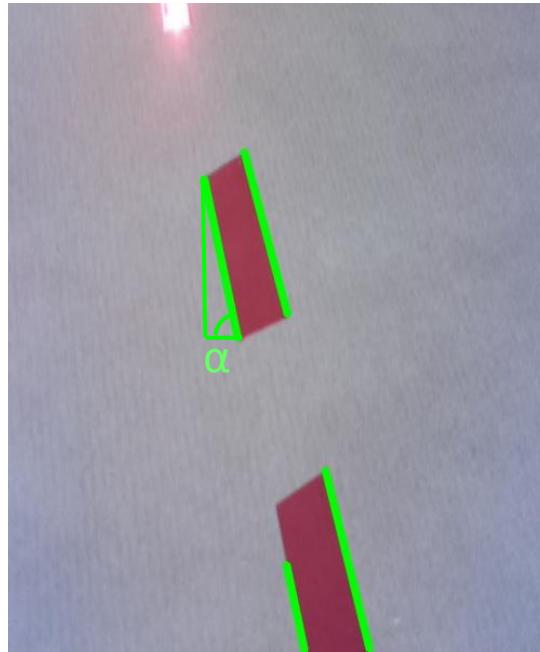


Abbildung 10: Verarbeitetes Bild mit Linien und Winkel α

6.4. Übergabe Parameter an Motoren

Mit dem erhaltenen Winkel α können der Regler und die Motoren passend angesteuert werden. In unserem Fall werden die Motoren direkt via der Dexter Industries Leiterplatte vom Raspberry Pi aus angesteuert. Eine Liste mit den Befehlen für die Ansteuerung des GoPiGos befindet sich im Anhang.

Die α -Winkel bilden die Informationen, welche über der Schnittstelle dem Rollator übergeben werden. Auf dem Rollator, können diese Informationen verwendet werden, um den Regler und die Motoren ansteuern zu können. Dabei ist die Wahl der Hardware für die Ansteuerung frei wählbar. Nachfolgende Abbildung zeigt, wie die ganzen Zusammenhänge für ein Nachfolgeprojekt aussehen könnten.

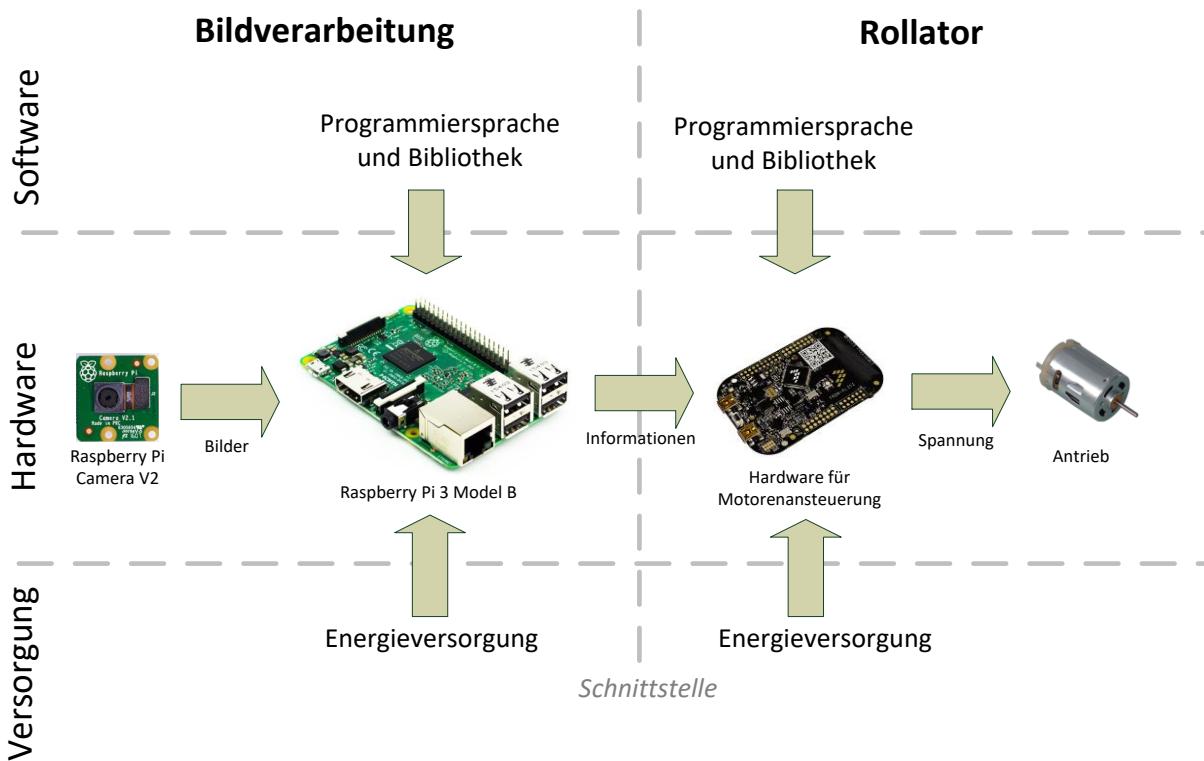


Abbildung 11: Konzept für ein Nachfolgeprojekt mit dem Rollator

6.5. Detektion Streckenende

In diesem Abschnitt wird beschrieben, wie das Ende der Strecke erkannt wird. Dadurch, dass LEDs das Ende der Strecke markieren, wird die Helligkeit im Bild gemessen. Dafür wird ein graues Bild erzeugt. Die vorher farbigen Pixel mit drei Werten (rot, grün und blau) enthalten jetzt neu nur noch einen Wert, den man als Grauwert bezeichnet. Nun muss ein Grenzwert für den Grauwert bestimmt werden. Für die Bestimmung des Grenzwerts wird ein Histogramm des Bildes erstellt. Das Histogramm zeigt, wie oft ein bestimmter Grauwert im Bild vorkommt. In Abbildung 12: *Histogramm des aufgenommenen Bildes* ist ersichtlich, dass vor allem Grauwerte im Bereich von 140 bis 180 vorkommen. Pixel mit einem Grauwert höher als 200 kommen selten vor. Die Messung des Grauwerts im Bild bestätigt die Erwartung, dass der leuchtende LED-Stripe im Bild einen höheren Grauwert als 200, besitzt. Aus diesem Grund wird der Grenzwert auf 200 gesetzt.

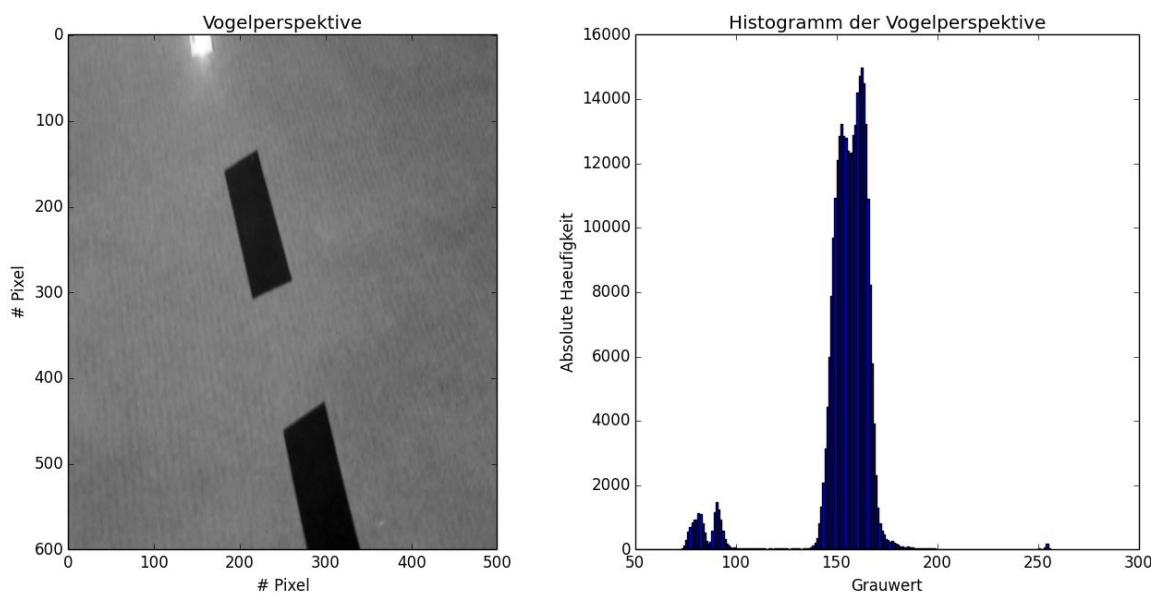


Abbildung 12: *Histogramm des aufgenommenen Bildes*

Nach dem festlegen des Grenzwerts werden Werte die kleiner als 200 sind schwarz dargestellt und höhere Werte weiss. Damit allfällige Löcher im weissen Teil des Bildes geschlossen werden, wird eine Schliessung durchgeführt. Falls kleinere Reflexionen vorhanden sind, werden diese mit einer anschliessenden Öffnung entfernt. Eine Schliessung oder Öffnung wird in der Bildverarbeitung auch als morphologische Operationen bezeichnet. Das Ergebnis nach der Bestimmung des Grenzwerts und den morphologischen Operationen ist in Abbildung 13: *Ergebnis der Grenzwertbestimmung* ersichtlich.

Reflexionen die sehr gross und über dem Grenzwert im Bild vorhanden sind, können nur schwer von der Zielmarkierung unterschieden werden. Aus diesem Grund ist im Programm ein Zähler eingebaut. Sobald vier Mal einen Wert über 200 entdeckt wird, hält der Roboter an. Ein Neustart kann nur noch manuell erfolgen.

Mittels des oben erwähnten Zählers, sollen mögliche Reflexionen während des aktiven Fahrens verschwinden. Der Grund, weshalb vier Durchläufe gewählt wurden, liegt darin, dass der Roboter dann eine Strecke von 200mm zurücklegt. Das entspricht der Distanz vom Roboter bis zum Punkt auf der Strecke. Siehe dafür Seite 38.

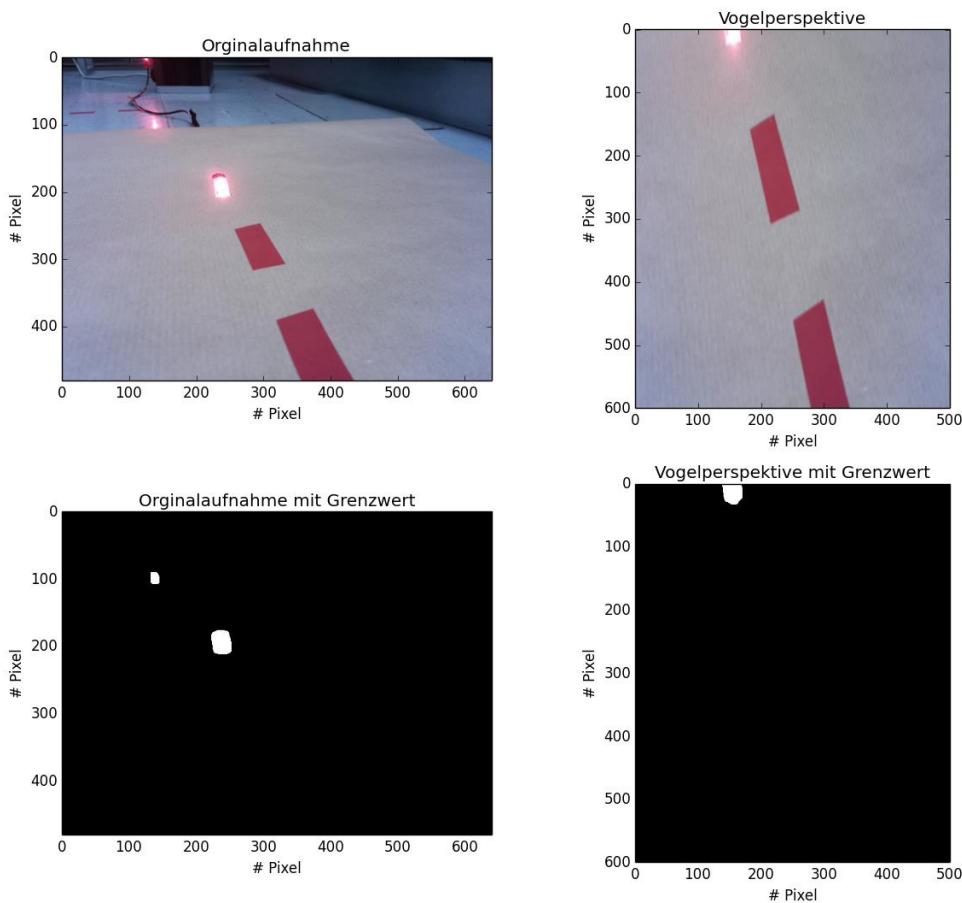


Abbildung 13: Ergebnis der Grenzwertbestimmung

Um nun zu überprüfen, ob die Strecke zu Ende ist, gibt es verschiedene Ansätze. Eine Möglichkeit wäre, die Überprüfung, ob eine genügend grosse weisse Fläche vorhanden ist. Eine andere ist die Bestimmung des Zentrums der weissen Fläche. Falls ein cm (center of mass) im Bild vorhanden ist, wird der Counter erhöht und das Raspberry Pi nimmt an, dass das Streckenende naht. Falls im Bild grosse und starke Reflexionen vorkommen, erhöhen diese ebenfalls den Counter. Man hat sich für die letztere Variante entschieden. In der rechten Aufnahme der nachfolgenden Abbildung markiert der blaue Punkt in der weissen Fläche das cm.

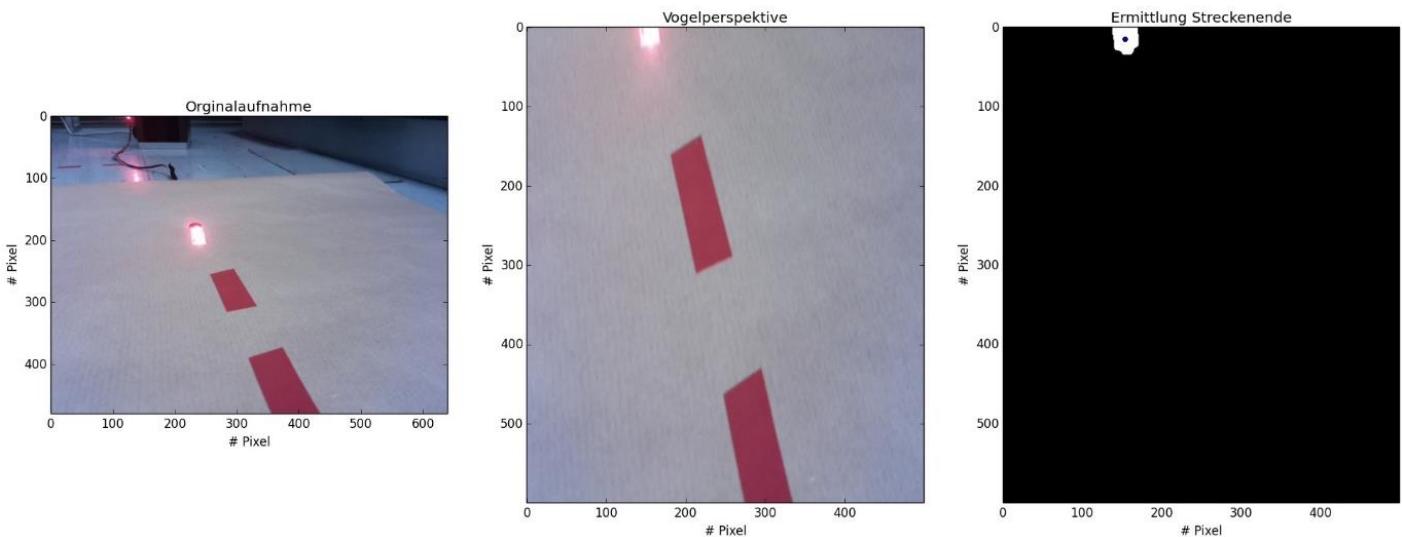


Abbildung 14: Detektion vom Ende der Strecke

7. Zentriertes Befahren

Im vorherigen Kapitel 6 wurde erläutert, wie die Kurvendetektion realisiert wird. Diese Methode gewährleistet aber nicht, dass auch eine zentrierte Befahrung stattfindet. Es kann vorkommen, dass ein Streifen detektiert und eine gute Kurve gefahren wird aber mit einem gewissen Abstand zur Wegmarkierung. Im folgenden Kapitel wird diese Problematik analysiert und passende Lösungen vorgestellt. Später soll die beste Methode für die Messung des Abstands in einen Regler einfließen.

7.1. Möglichkeiten zur Abstandsmessung

Um den Abstand zur Mittellinie zu messen, werden zwei Ideen vorgestellt. Für die folgenden Beispiele spielt es keine Rolle, ob sich die Wegmarkierung links oder rechts der Mittellinie befindet. Für eine bessere Vorstellung wird die Abbildung 15: Angaben zur Abstandsmessung hinzugezogen. Die Abbildung zeigt ein mögliches aufgenommenes Bild. Dabei bildet das rote Rechteck die Wegmarkierung und die blaue Strich-Punkt-Linie die Mittellinie des Roboters. Die Zahlen links und oberhalb des Bildes entsprechen den Koordinaten des Bildes. Das aufgenommene Bild besitzt somit eine Auflösung von 500x600 Pixel. Das entspricht der Auflösung nach der Perspektiven-Transformation.

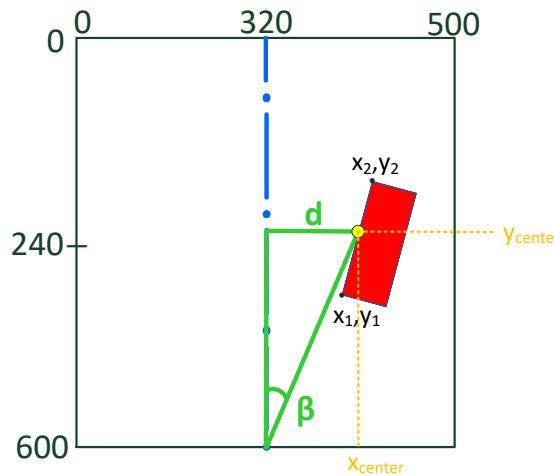


Abbildung 15: Angaben zur Abstandsmessung

Die erste Möglichkeit für die Messung besteht darin, den Abstand d zu messen. Wenn der Roboter nun zentriert fährt, ist d klein und im Idealfall beträgt dieser 0. Um es ganz genau zu nehmen, müsste noch die Breite der Wegmarkierung berücksichtigt werden. Dies wird hier vernachlässigt und man nimmt einen Fehler in Kauf, da das Verhältnis der Markierungsbreite wesentlich kleiner als die Bildbreite ist. Den Mittelpunkt der Länge der Wegmarkierung (gelber Punkt) kann man berechnen, indem man die Linie, die durch die Hough-Transformation resultiert, halbiert.

Die zweite Möglichkeit liegt in der Berechnung des Winkels β . Dabei wird wieder der Mittelpunkt (gelber Punkt) verwendet. Um d zu berechnen wird die X-Koordinate der Mittellinie von der X-Koordinate des Mittelpunkts abgezogen. Da das Resultat somit positive und negative Werte annehmen kann, wird hier der Betrag genommen. Auf welche Seite der Roboter korrigieren muss, wird im Kapitel 8.4 erläutert.

Berechnung des Mittelpunkts und d :

$$x_{Center} = \frac{x_2 - x_1}{2} + x_1$$

$$y_{Center} = \frac{y_2 - y_1}{2} + y_1$$

$$d = x_{Mittellinie} - x_{Center}$$

Berechnung von β in Grad:

$$\beta = \arctan\left(\frac{d}{y_{Center}}\right) \cdot \frac{180}{\pi}$$

Da in unserem Fall die Kamera nicht genau in der Mitte des Roboters liegt, wurde die Mittellinie nicht bei $x_{Mittellinie} = 250$ gesetzt, sondern bei 300. Dieser Wert erhielt man, indem ein Meterstab vor dem Roboter gelegt und ein Bild aufgenommen wurde. Die Breite des GoPiGos beträgt 15cm. Die Mittellinie des Roboters liegt dann bei 7.5cm. Nun muss noch die X-Koordinate im Bild an der Stelle 7.5cm herausgelesen werden und schon kann die korrekte Mittellinie ermittelt werden.

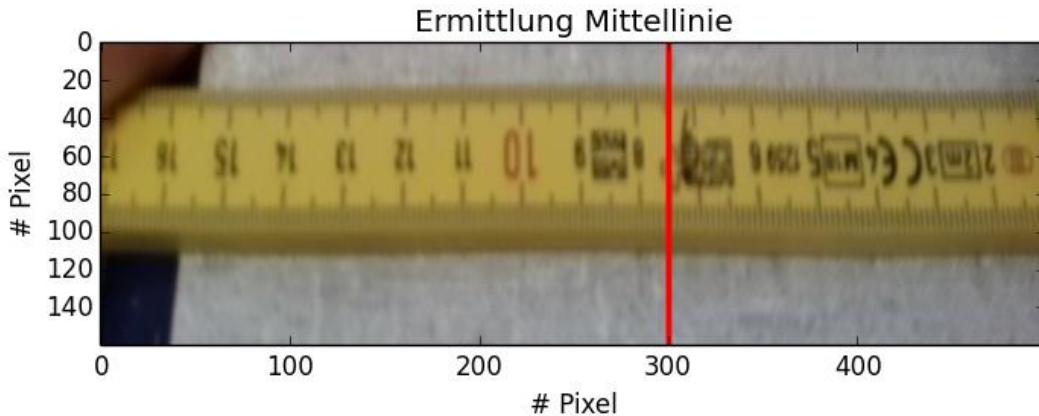


Abbildung 16: Ermittlung der Mittellinie

7.2. Wahl der Abstandsmessung

Um nun zu entscheiden, welcher Ansatz sich am besten eignet, werden zwei mögliche Situationen analysiert.

Der erste Ansatz ist in *Abbildung 17: Erstes Beispiel für die Abstandsmessung* ersichtlich. Dieser zeigt zwei mögliche Aufnahmen der Kamera. In der linken Aufnahme ist die Wegmarkierung weiter entfernt als in der rechten. Dabei ist d_1 gleich weit entfernt von der Mittellinie wie d_2 . Somit können folgende Schlüsse gezogen werden:

$$\beta_1 < \beta_2 \text{ und } d_1 = d_2$$

Würde man nun d als Parameter an den Regler übergeben, kann er nicht unterscheiden, ob die Wegmarkierung nahe ist oder noch weit entfernt. Anders sieht es beim Winkel β aus. Der Winkel enthält auch Informationen, wie weit entfernt die Wegmarkierung ist. Somit würde die Korrektur in der rechten Aufnahme stärker gewichtet werden als in der linken.

Daraus schliesst man, dass β sich in diesem Fall besser eignet als d .

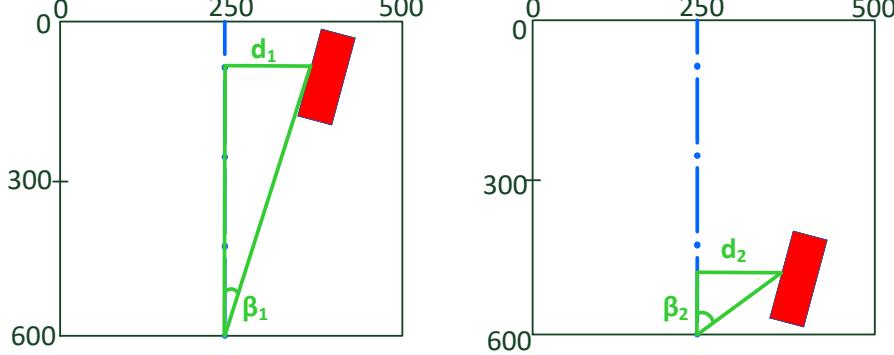


Abbildung 17: Erstes Beispiel für die Abstandsmessung

Für die zweite Möglichkeit wird analysiert, wie sich d und β verhalten, wenn der Abstand zur Mittellinie grösser wird. Die Abbildung 18: Zweites Beispiel für die Abstandsmessung zeigt zwei mögliche Aufnahmen.

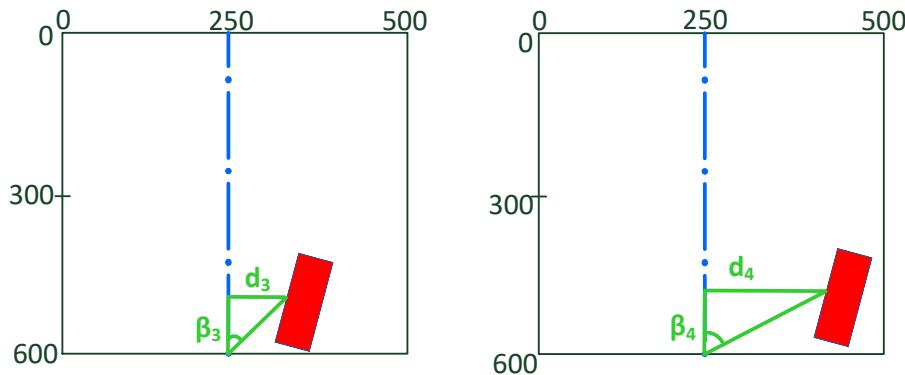


Abbildung 18: Zweites Beispiel für die Abstandsmessung

Für die oben gezeigte Situation können folgende Schlüsse gezogen werden:

$$\beta_3 < \beta_4 \text{ und } d_3 \ll d_4$$

Wenn nun der Abstand zur Mittellinie grösser wird, ändert sich der Winkel β und d . Dabei ändert sich d viel stärker als β . In diesem Fall eignet sich d besser für die Abstandsmessung.

Entscheidung:

Der Vorteil von d liegt in der Einfachheit. Es müssen keine trigonometrischen Rechnungen vollzogen werden und er ist leicht zu berechnen. Dafür wird die Entfernung der Wegmarkierung nicht berücksichtigt. In β stecken Informationen über die Distanz. Bei grosser Distanz zur Mittellinie fällt β aber weniger ins Gewicht als d .

Es ist wünschenswert, dass auch die Entfernung gewichtet wird. Somit kann, wenn die Wegmarkierung nahe ist, schneller die Zentrierung vorgenommen werden und der Regler muss später nicht mehr so stark regeln. Aus diesem Grund wird β für die Abstandsmessung gewählt.

Folgend noch eine Aufnahme, wenn sich die Mittellinie rechts der Wegmarkierung befindet. Dabei wären d_5 und β_5 positiv. In den obigen Beispielen wäre d und β negativ.

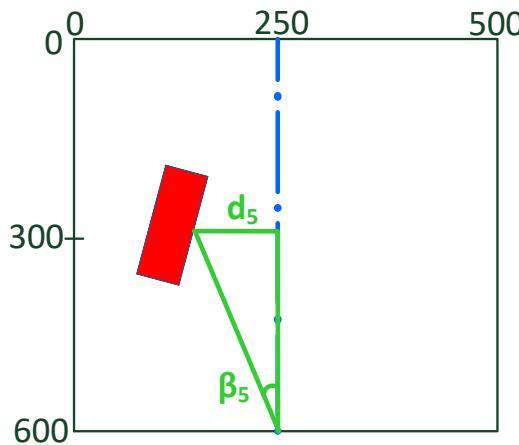


Abbildung 19: Drittes Beispiel für die Abstandsmessung

8. Regelung und Korrekturen

Mittels einer passenden Regelung soll der Roboter die Strecke zuverlässig befahren können. Dabei soll er dies möglichst zentriert bewältigen. Mit dem in Kapitel 6 erhaltenen Kurvenwinkel α wird ein geeigneter Regler implementiert. Da der erhaltene Winkel keine Auskunft darüber gibt, ob der Roboter jetzt in der Mitte fährt, wird noch ein zusätzlicher Wert benötigt. Dieser Wert ist ebenfalls ein Winkel mit der Bezeichnung β . Dieser wurde im Kapitel 7 besprochen.

Abbildung 20: Geschlossener Regelkreis des Systems zeigt den Regelkreis mit allen wichtigen Komponenten. Die Bildverarbeitung extrahiert aus dem Bild die benötigten Winkel α und β . Für α wird ein 90° Winkel angestrebt, da dann der Roboter gerade fährt. Für β wird ein Winkel von 0° bevorzugt, da somit die Zentrierung schon vorhanden ist. Die Kurvenregelung wird mit einem PID-Regler realisiert und die Mittellinienregelung mit einem P-Regler.

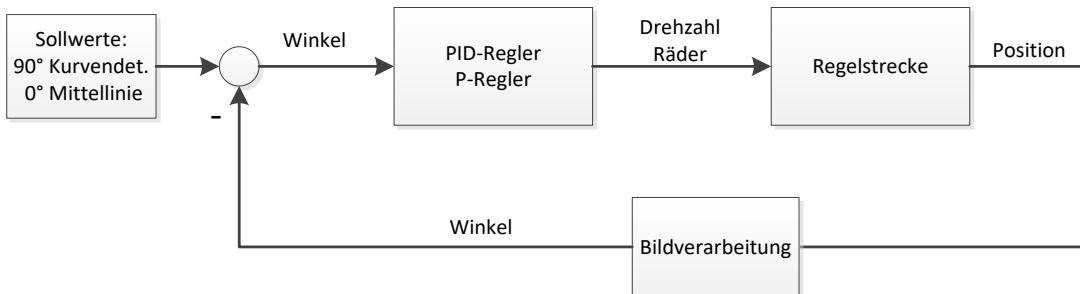


Abbildung 20: Geschlossener Regelkreis des Systems

8.1. Kurvenregelung

Für die Kurvenregelung wird der erhaltene Winkel α benötigt. Das Prinzip dieses Reglers beruht auf einem PID-Regler. Dabei wird die Differenz aus der Führungsgröße und dem Winkel α gebildet. Die Differenz wird mit einem proportionalen, einem integralen und einem differenziellen Anteil multipliziert und dann zusammen addiert. Mehr dazu folgt im Kapitel 8.3.

Für die Regelung müssen die Parameter K_p , K_i , K_d und T_d passend gewählt werden. Da die Regelstrecke nicht bekannt ist, können dort höchstens Annahmen getroffen werden. Durch das DC-Motoren verwendet werden, wäre es möglich, die Regelstrecke mittels einem PT_1 -Glied zu beschreiben. Die Bildverarbeitungszeit wird durch die Totzeit T_t beschrieben. Somit kann das System folgendermassen ausgedrückt werden:

$$G(s) = \frac{K_s}{1 + T_1 s} e^{-sT_t}$$

Die genaue Herleitung für $G(s)$ mit allen Werten befindet sich im Anhang. Da keine Angaben zu den verwendeten Motoren gefunden wurden, hat man Werte eines ähnlichen Motors genommen. Dieser Motor wird auch in der Blockwoche *Regelungstechnik Labor* an HSLU verwendet. Für T_t wird die gemessene Zeit aus den Tests angenommen. T_t ist abhängig von den erhaltenen Bildern. Wenn viele Kanten im Bild ersichtlich sind, so wird auch mehr Bearbeitungszeit benötigt und somit steigt die Totzeit.

Bei der obigen Übertragungsfunktion handelt es sich um ein zeitkontinuierliches System. Würde man das System als diskret beschreiben wollen, müssten noch gewisse Ergänzungen vorgenommen werden. Mit $G(s)$ können mittels der Ziegler-Nichols Methode erste Werte für K_p , T_i und T_d evaluiert werden. Eine andere Möglichkeit besteht in der Verwendung von Matlab und Siso-Tool. Für die erste Dimensionierung des Reglers wurden mit dem Siso-Tool die Parameter bestimmt.

8.2. Regelung zur Mittellinie

Im folgenden Abschnitt wird auf die Regelung zur Mittellinie eingegangen. Wie im Kapitel 7 beschrieben wird β als Abstandsparameter verwendet.

Für die Regelung können sechs Fälle auftreten. Diese werden analysiert und daraus Schlüsse für die Regelung gezogen. Die Entfernung der Wegmarkierung zum Roboter wird hierbei nicht berücksichtigt. Der Winkel β berücksichtigt bereits einen Anteil. Das + symbolisiert, dass eine positive Anpassung vorgenommen werden soll. Das - stellt eine negative Anpassung dar.

Im ersten Fall steht eine Rechtskurve bevor. Dabei kann die Mittellinie rechts oder links der Wegmarkierung liegen. Im Falle, dass die Mittellinie links ist, muss eine stärkere Rechtskurve stattfinden. Falls die Mittellinie rechts ist, muss die Rechtskurve leichter durchgeführt werden, da man automatisch auf die korrekte Spur kommt.

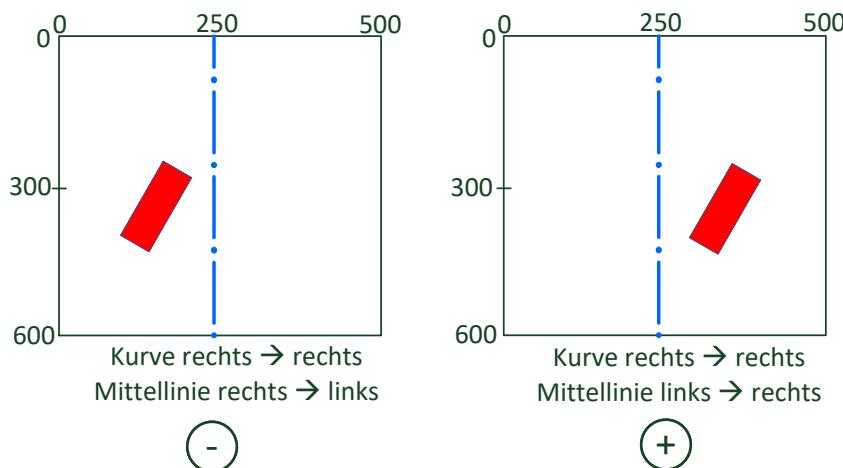


Abbildung 21: Regelung zur Mittellinie Rechtskurve

Im zweiten Fall steht eine Linkskurve bevor. Wieder kann die Mittellinie sich links oder rechts der Wegmarkierung befinden. Es wird wieder der gleiche Gedanke, wie beim ersten Fall, aufgegriffen. Falls die Mittellinie sich rechts befindet, muss die Kurve stärker befahren werden. Bei der rechten Aufnahme muss weniger stark gedreht werden.

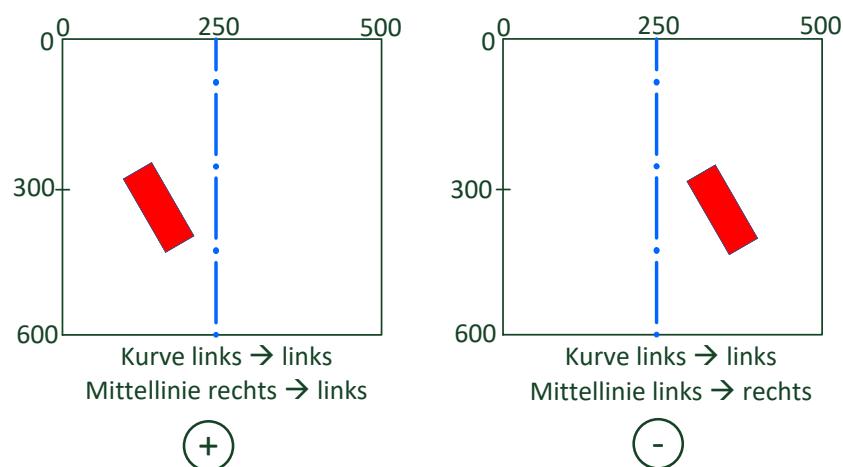


Abbildung 22: Regelung zur Mittellinie Linkskurve

Der letzte Fall beinhaltet das geradeaus Fahren. Falls die Mittellinie sich rechts der Wegmarkierung befindet, soll nach rechts korrigiert werden. Falls die Mittellinie links ist, soll der Roboter sich nach links drehen. Diese Korrektur findet nicht in der PID Funktion statt, sondern direkt im Hauptprogramm.

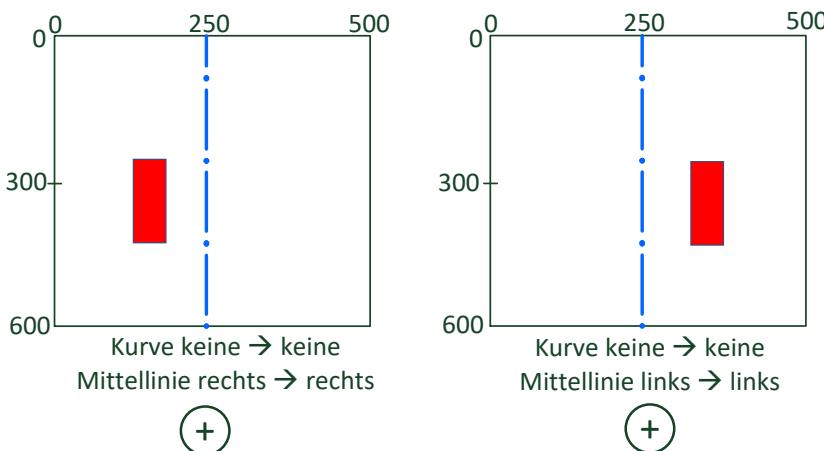


Abbildung 23: Regelung zur Mittellinie Gerade

Die Regelung wird mittels der Drehzahl der einzelnen Motoren durchgeführt. Steht nun eine Rechtskurve an, muss sich das linke Rad schneller drehen als das rechte. Ist nun noch die Mittellinie links der Wegmarkierung, wird der Roboter nochmals mehr nach rechts drehen um die Mittellinie zu greifen.

8.3. Implementierung der Regler

Beide Regler wurden zusammen in einer Funktion implementiert. Diese Funktion erhält als Eingabe die Winkel α , β und d . Mit diesen Parametern werden Werte für die Motoren berechnet, die zwischen 0 und 40 sind. Grundsätzlich sind Werte zwischen 0 und 255 möglich. Würde der Wert zu hoch gewählt werden, wäre der Roboter zu schnell unterwegs und er könnte keine Korrekturen mehr vornehmen.

Wie schon am Anfang erwähnt, werden ein PID- und ein P-Regler verwendet. Die erhaltenen Winkel werden zuerst vom Sollwert abgezogen. Durch diese resultiert ein Fehler, der mit einem Proportionalen, einem Integralen und einem Differentiellen Anteil multipliziert wird. Anschliessend werden diese verschiedenen Anteile zusammen addiert. Der Integrator und der Differenzierer benötigen für die Berechnung die Abtastzeit. Da diese Zeit vom aufgenommenen Bild abhängig ist, je nach Bild werden mehrere Linien und Reflexionen erkannt, wird sie jedes Mal neu gemessen. Nun folgt der Code des PID-Reglers. Der gesamte Code befindet sich im Anhang.

```
def PID(angle, beta, d):
    global time_old, iValue, lastError
    time_new = int(round(time.time() * 1000))
    T = time_new - time_old

    error = 90-angle

    #P-Anteil
    pValue = Kp * error
    #I-Anteil
    iValue += Ki/Ti * (error + lastError)/2 * T
    #D-Anteil
    dValue = Kd * Td * (error - lastError) / T

    #Reset I-Value (Anti-Reset-Windup)
    if angle > 100:
        iValue = 100

    #set new values
    lastAngle = angle
    time_old = time_new
```

```
#calculation for output (es wird durch 180 damit value nicht zu gross)
value = (pValue + iValue + dValue)/180
```

Im Regelkreis wurde nach dem Regler eine Saturation und ein Anti-Reset-Windup implementiert. Die Saturation limitiert die Drehzahl auf eine obere und eine untere Grenze. Der programmierte Anti-Reset-Windup sorgt dafür, dass der summierte Fehler des Integrators einen gewissen Wert nicht übersteigt.

Für die Regelung zur Mittellinie muss zuerst die Kurvenrichtung bestimmt werden. Dies wird mit dem Winkel α realisiert. Die Richtung wird benötigt, um die einzelnen Fälle in Kapitel 8.2 zu unterscheiden. Danach wird der Median des Fehlers und des Abstands d berechnet. Anschliessend werden die Fälle die im Kapitel 8.2 erwähnt werden berechnet. Der Verstärkungsfaktor Kp_Center wurde experimentell mit einem selber geschriebenen Python Skript ermittelt.

```
#Regelung zur Mittellinie
if angle > 0 and abs(angle)< 85:      #Motoren sollen rechts drehen
    detectLeft = None
    detectRight = True
elif angle < 0 and abs(angle)< 85:      #drehe links
    detectLeft = True
    detectRight = None
else:
    detectLeft = None
    detectRight = None

error = np.median(beta*np.pi/180)      #Berechnung Fehler in Bogenmass
d = np.median(d)

if d >= 0 and detectLeft == True:      #Mittellinie rechts und Kurve links
    value += Kp_Center * error
elif d < 0 and detectRight == True:     #Mittellinie links und Kurve rechts
    value += Kp_Center * error
elif d >= 0 and detectRight == True:     #Mittellinie rechts und Kurve rechts
    value -= Kp_Center * error
elif d < 0 and detectLeft == True:       #Mittellinie links und Kurve links
    value -= Kp_Center * error
elif detectLeft == None and detectRight == None and d < 0:  #Mittellinie ist rechts
    value += Kp_Center * error
elif detectLeft == None and detectRight == None and d >= 0: #Mittellinie ist links
    value += Kp_Center * error

#Saturation
if value > 40:
    value = 40
elif value < 0:
    value = 0

return value
```

Da für das vorhandene System eine grosse Totzeit anfällt und die Abtastzeit sehr gross ist, kann der PID-Regler nur schwer seine Aufgabe erfüllen. Aus diesem Grund mussten Anpassungen am Programm vorgenommen werden, damit das GoPiGo in einem guten Rahmen Kurven fahren kann. Die erste Anpassung liegt in der Geschwindigkeitsreduzierung. Die andere Anpassung befasst sich mit der Implementierung eines Gedächtnisses und die langsame stückweise Befahrung der Kurve. Da der Roboter bei der Zentrierung immer in die Saturation läuft, musste eine zusätzliche Anpassung vorgenommen werden. Diese Anpassung wird im Abschlusstest beschrieben.

8.4. Automatische Rückfindung

Es kann nun sein, dass für einige Bilder keine Linien erkannt werden. Ein Grund dafür liegt darin, dass aus den erkannten Kanten sich nicht immer Linien bilden lassen. Es ist auch möglich, dass der Roboter während dem Fahren die Linie verliert. Darum wäre es von Vorteil zu wissen, was die letzte Richtung war, in die er gefahren ist. Aus diesem Grund wurde eine Art Gedächtnis implementiert. Dafür wird der Abstand zur Mittellinie d benötigt. Es werden nun vier Fälle vorgestellt, in denen der Roboter die Linie verloren hat und die Kamera keine Wegmarkierung aufnehmen konnte.

Im ersten Fall ist der Roboter links von der Linie abgekommen und sollte nach rechts korrigieren. Der Abstand d ist dabei negativ, weil die Wegmarkierung rechts lag. Dies ist auch in den Abbildungen 21 bis 23 ersichtlich.

Der zweite Fall beschreibt die Situation, wenn der Roboter zu früh eingelenkt hat. Somit wird der Roboter links die Wegmarkierung finden. Das erhaltene d aus vorherigen Bildern war positiv.

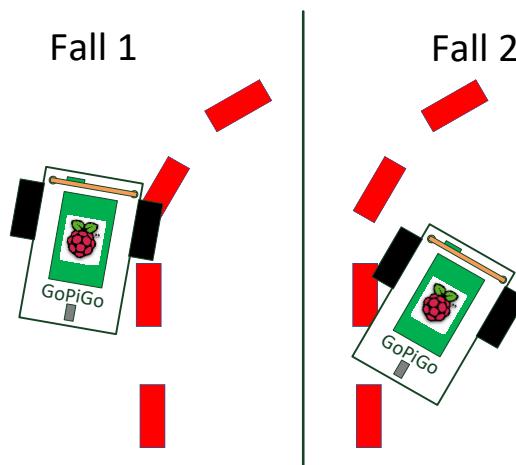


Abbildung 24: Automatische Rückfindung Fall 1 und Fall 2

Im Fall drei steht eine Linkskurve an. Hier hat der Roboter zu früh eingelenkt und findet die Wegmarkierung nicht mehr. Aus diesem Grund soll er nach rechts korrigieren, um wieder die Wegmarkierung zu finden. Das d hierbei war negativ.

Der vierte und letzte Fall befasst sich mit der Situation, in der der Roboter zu spät einlenkt. Hier befindet sich die Wegmarkierung rechts. Somit soll links nach der Wegmarkierung gesucht werden. Das d war für diesen Fall positiv.

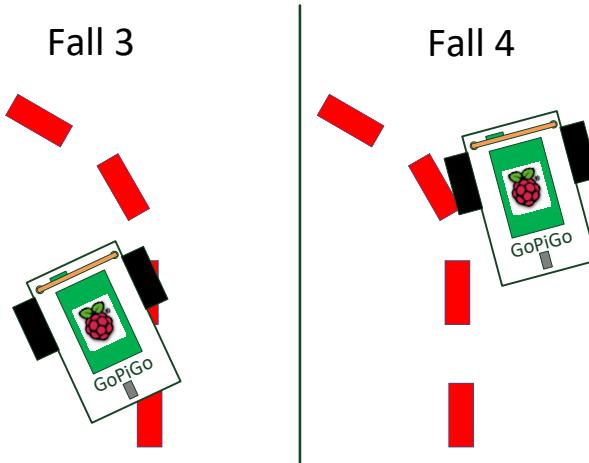


Abbildung 25: Automatische Rückfindung Fall 3 und Fall 4

Aus den vorherigen Aussagen ergibt sich folgende Tabelle:

Fall	erkanntes d	Korrektur nach
Fall 1	negativ	rechts
Fall 2	positiv	links
Fall 3	negativ	rechts
Fall 4	positiv	links

Tabelle 8: Automatische Rückfindung mittels der Abstandsmessung

Das d wird immer erstellt, sobald eine Linie im Bild erkannt wird. Wenn keine Linie erkannt wird, bleibt das alte d erhalten. Aus den obigen Erkenntnissen schliesst man, dass das d für die automatische Rückfindung gut verwendbar ist. Sobald das d positiv ist soll rechts nach der Wegmarkierung gesucht werden. Bei negativem d nach links. Falls während fünf Durchgängen noch immer keine Linie erkannt wird, stoppt der Roboter automatisch. Sobald wieder eine Line erkannt wird fährt er weiter. Wenn keine Linien erkannt werden, blinken die LEDs vorne auf der roten Leiterplatte. Für diese Methode darf die Geschwindigkeit nicht zu gross gewählt werden, da sonst ein falsches d resultiert.

Durch die obigen Fälle wird auch ersichtlich, dass durch die versetzte Positionierung der Kamera Linkskurven leichter durchführbar sind als Rechtskurven. Diesem Problem wird entgegengewirkt indem eine zentrierte Befahrung durchgeführt wird.

8.5. Kontinuierliches Fahren

Im Abschnitt 8.4 wurde die Lösung besprochen, wenn der Roboter die Strecke verliert. Nun kann es auch sein, dass die Wegmarkierungen nicht immer zuverlässig platziert wurden. Bis jetzt kann eine nicht korrekt angebrachte Wegmarkierung das ganze Fahrverhalten beeinflussen.

Ein weiterer Faktor ist, dass die Kamera sehr weitsichtig ist. Dadurch wird ein Richtungswechsel schon früh erkannt. Um nicht zu früh einzulenken, wäre es empfehlenswert, einen Moment zu warten, bis der Roboter sich in die gewünschte Richtung bewegt. Für beide Probleme wurde eine gemeinsame Lösung erarbeitet, die in folgendem Abschnitt besprochen wird.

Nachfolgende Abbildung zeigt die Weitsicht des Roboters mit den dazugehörigen Messwerten.

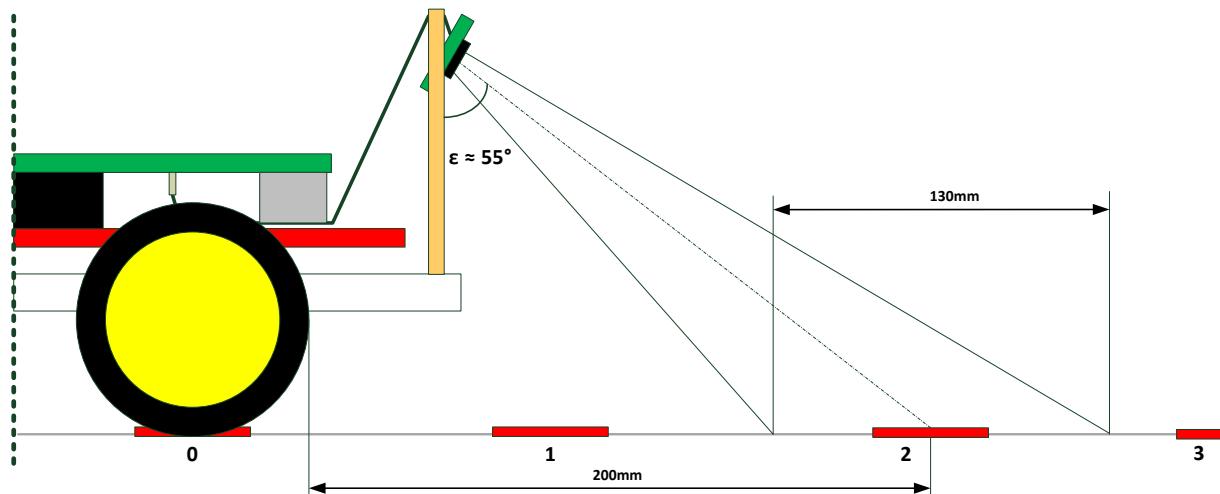


Abbildung 26: Detailansicht des Roboters mit Fahrbahn

Für den nun aufgeführten Lösungsansatz wurden folgende Gedanken gemacht. Wenn die letzten Ergebnisse eine gerade Strecke ergaben, ist es sehr wahrscheinlich, dass der nächste Abschnitt ebenfalls gerade ist. Somit soll für die Wegmarkierung 1 gerade gefahren werden, wenn vorhin auch gerade Wegmarkierungen erkannt wurden. Die Wegmarkierungen 2, 3 und 4 wären Rechtskurven. Dann soll der Roboter spätestens bei der zweiten Wegmarkierung die Rechtskurve anfahren.

Um nun zu entscheiden, welche Richtung eingeschlagen werden soll, wird die Anzahl der erkannten Kurven zusammengezählt. Dabei werden die letzten vier und das aktuelle Ergebnis berücksichtigt. Das Ganze kann man in Form eines Ringbuffers oder einer Matrix ausdrücken.

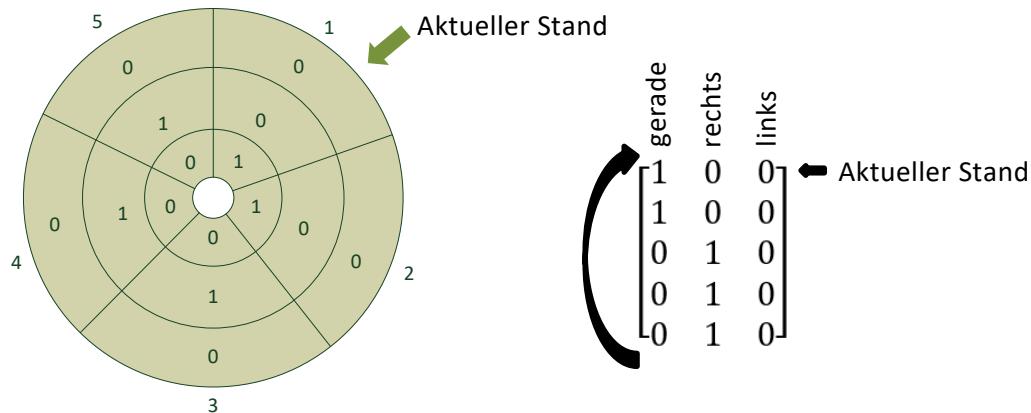


Abbildung 27: Darstellung des Ringbuffers in Kreis- und Matrizenform

Bei der Matrix stellt die linke Spalte die erkannten geraden Teile dar, die mittlere Spalte die rechten Kurven und die rechte Spalte die erkannten linken Kurven. Zählt man nun die Spalten zusammen, kommt man auf folgendes Ergebnis:

Erkannte Geraden: 2
Erkannte Rechtskurven: 3
Erkannte Linkskurven: 0

Dadurch, dass mehr Rechtskurven erkannt worden sind als Geraden wird ab der Wegmarkierung 2 die Rechtskurve angefahren.

Nun besteht die Möglichkeit, vor allem zu Beginn, wenn das Array mit Nullen gefüllt ist, schon nach zwei erkannten Geraden, zwei rechte Kurven folgen. Dann würde die Matrix folgendermassen aussehen:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Wenn der Fall auftritt, dass zwei Spalten die gleiche Summe ergeben, wird in die Richtung gefahren, in der die letzte Markierung erkannt wurde. Im Falle der obigen Matrix somit nach rechts.

Der Grund für die Wahl von fünf Zeilen liegt darin, dass fünf ungerade ist und somit der obig erwähnte Fall nur selten vorkommt. Der andere Grund liegt in der Distanz. Da man 200mm von der Wegmarkierung entfernt ist und nach einem Durchgang 50mm gefahren ist, ergibt das vier Durchgänge und somit vier Zeilen. Um diese Zahl noch auf eine Ungerade zu bekommen wird auf fünf aufgerundet.

Diese Lösung bringt auch gewisse Nachteile mit sich. Für die Wegmarkierung müssen dann folgende Bedingungen erfüllt sein:

- Kurven müssen aus mindestens drei Wegmarkierungen bestehen
- Zu Beginn müssen mindestens 2 gerade Wegmarkierungen vorkommen

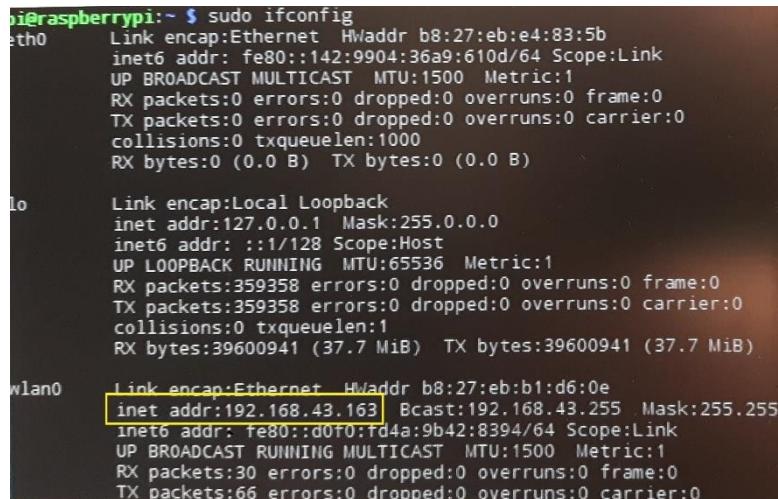
Zudem besteht die Gefahr, dass der Roboter über die Linie fährt. Diese Methode funktioniert nur solange nicht sehr viele fehlerhaft platzierte Wegmarkierungen vorhanden sind. Diese möglichen Probleme nimmt man in Kauf.

9. Tests und Verbesserungen

Zu allen Tests wurde ein Bericht erstellt. Diese sind im Anhang vorzufinden. Für die meisten Tests war eine WLAN Verbindung nötig. Wie man sich verbindet ist im Kapitel 9.1 genauer beschrieben.

9.1. Drahtloser Zugriff auf Raspberry Pi

Um auf das Raspberry Pi drahtlos zugreifen zu können, wird ein mobiler Hotspot via Handy erstellt. Der Computer und das Raspberry Pi müssen mit diesem Hotspot verbunden sein. Um die IP-Adresse des Raspberry Pi herauszufinden, werden ein Desktop und eine Tastatur empfohlen. Mittels dem Befehl `sudo ifconfig` im Terminal des Raspis kann die IP-Adresse herausgefunden werden.



```
pi@raspberrypi:~ $ sudo ifconfig
eth0      Link encap:Ethernet HWaddr b8:27:eb:e4:83:5b
          inet6 addr: fe80::142:9904:36a9:610d/64 Scope:Link
              UP BROADCAST MULTICAST  MTU:1500 Metric:1
              RX packets:0 errors:0 dropped:0 overruns:0 frame:0
              TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
              UP LOOPBACK RUNNING MTU:65536 Metric:1
              RX packets:359358 errors:0 dropped:0 overruns:0 frame:0
              TX packets:359358 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1
              RX bytes:39600941 (37.7 MiB)  TX bytes:39600941 (37.7 MiB)

wlan0    Link encap:Ethernet HWaddr b8:27:eb:b1:d6:0e
          inet addr:192.168.43.163 Bcast:192.168.43.255 Mask:255.255.
          inet6 addr: fe80::d0f0:1d4a:9b42:8394/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST  MTU:1500 Metric:1
              RX packets:30 errors:0 dropped:0 overruns:0 frame:0
              TX packets:66 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:1024 (1.0 B)  TX bytes:1024 (1.0 B)
```

Abbildung 28: Ausschnitt des RaspiTerminals

Mit Putty wird ein SSH-Zugriff auf das Pi erstellt. Mit dem Benutzernamen und dem Passwort muss man sich anschliessend einloggen. Der Nutzername und das Passwort befinden sich im Kapitel 5.1

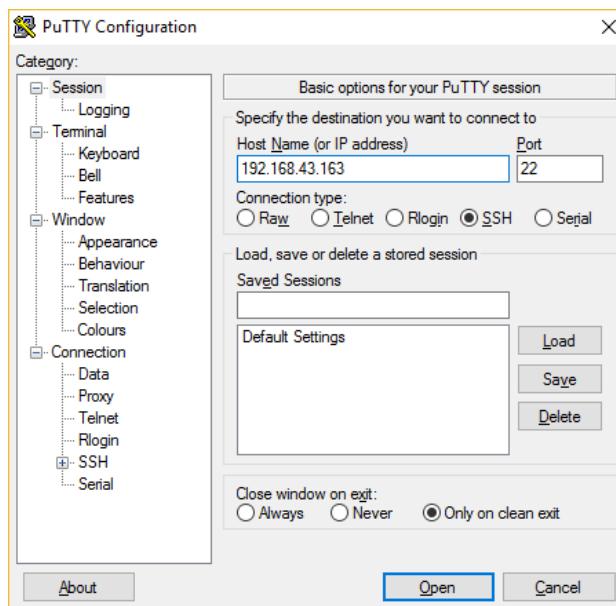


Abbildung 29: Zugriff auf das Raspberry Pi mittels Putty

9.2. Verbesserung der Effizienz

Die ganze Regelung hängt sehr stark von der Bildverarbeitung ab. Wenn nun die Bildverarbeitung sehr lange dauert, entstehen grosse Totzeiten, was das ganze System in eine instabile Lage bringt. Beim ersten Test fiel auf, dass die Regelung sehr schlecht funktioniert und es sehr lange dauert, bis eine Korrektur stattgefunden hat. Aus diesem Grund wurde die Zeit für einen Durchlauf gemessen. Das Ergebnis lag im Bereich von 1.2 und 1.5 Sekunden. Bei dieser Messung war das OpenCV 2.4.9.1 auf dem Raspberry Pi installiert. Zudem betrug die Auflösung des Bildes nach der Perspektive 1000x1250 Pixel.

Damit die Totzeit kleiner wird, wurden folgende Anpassungen vorgenommen:

- OpenCV 3.2 auf das Raspberry Pi installiert
- Auflösung nach der Perspektive auf 500x600 Pixel heruntergesetzt

Nach diesen Anpassungen wurde das gleiche Programm wieder ausgeführt und die Zeit gemessen. Die gemessene Zeit lag im Bereich von 0.5 Sekunden und 0.7 Sekunden. Somit läuft das Programm nun doppelt so schnell, was sich sehr positiv auf die Regelung auswirkt. Lässt man nun noch das Erzeugen von Bildern und die Konsolenausgabe weg, erreicht man 0.4 Sekunden.

10. Abschluss

In diesem letzten Kapitel wird ein Rückblick mit einem Vergleich auf die Anforderungen vollzogen. Für die spätere Übertragung und Einsatz auf dem Rollator wird im Ausblick auf wichtige Punkte hingewiesen. Zum Schluss folgt ein persönliches Fazit.

Der Auftrag einen autonomen Rollator bis Ende Semester zu erarbeiten, konnte nicht erfüllt werden. Trotzdem konnte während dieser BDA ein Roboter entworfen werden, der selbständig einer Strecke folgt und an dessen Ende anhält.

10.1. Rückblick

In diesem Abschnitt wird der Bezug auf die Aufgabenstellung und die selber gestellte Anforderungsliste genommen. Beide befinden sich im Anhang. Im Kapitel 2 wurden die Anforderungen mit den Streckenspezifikationen bereits angesprochen.

- Der Rollator des iHomeLabs konnte aus technischen Gründen nicht verwendet werden. Aus diesem Grund wurde diese Arbeit mit einem externen Roboter durchgeführt.
- Der Roboter folgt einer Strecke und detektiert die Wegmarkierungen mittels der Kanten. Während der Fahrt erfasst der Roboter Bilder und extrahiert aus diesen Winkel und Abstände. Diese Parameter werden den Reglern übergeben. Sobald keine Spur mehr vorhanden ist, sucht der Roboter zuerst nach dieser. Nach mehreren vergeblichen Versuchen wird die Suche abgebrochen und er hält an. Sobald wieder eine Spur erkannt wird, fährt er fort. Es können keine Befehle über eine Fernbedienung durchgeführt werden. Es ist aber möglich über WLAN auf den Roboter zuzugreifen. So können Anpassungen und Befehle übergeben werden.
- Für die Befahrung wurden Regler implementiert und getestet. Eine zuverlässige Befahrung konnte nur mittels Autokorrektur und zusätzlichen Funktionen realisiert werden. Das konstante flüssige Fahren kann nicht gewährleistet werden, da die Regelung zu langsam agiert. Farbstörungen werden vollkommen ignoriert. Die Resistenz auf falsche Kanten oder Helligkeiten konnten nur begrenzt eingebaut werden.
- Als Abschluss wurde eine Teststrecke erstellt. Diese wurde getestet und soll auch an der Abschlusspräsentation und bei der Diplomausstellung vorhanden sein um LIVE-Vorführungen durchführen zu können. Bei der Auslegung der Strecke wurde nicht exakt gearbeitet und die Masse wurden nicht immer eingehalten.

Fazit Erfüllung des Auftrags

Grundsätzlich erfüllt der Roboter die Aufgabe der Streckendetektion und deren Abfuhr. Es sind aber weitere Optimierungen nötig, um die Zuverlässigkeit zu verbessern. Vor allem bei der Dauer der Bildverarbeitung und der Regelung wird eine Nacharbeit empfohlen.

10.2. Ausblick

Bei einer Weiterführung dieses Projektes sind Verbesserungen und Nacharbeiten vorzunehmen. In diesem Unterkapitel werden Verbesserungen und wichtige Hinweise in den Bereichen Hardware, Bildverarbeitung und Reglerimplementierung beschrieben.

Verbesserungen und Hinweise zur Hardware

Zur Hardware gehören das Raspberry Pi, die Kamera und die Motoren. In diesem Projekt wurden die Motoren direkt vom Raspberry Pi angesteuert. Später kann dies über eine andere Hardware durchgeführt werden. Es wird empfohlen, dass der Regler und die Bildverarbeitung eigene Rechenzeiten bekommen und diese nicht voneinander abhängig sind. Aus diesem Grund sollen sie auf unterschiedlicher Hardware arbeiten oder mittels Multithreading voneinander unabhängig sein.

Bei der Kamera sind vor allem die Positionierung und der Anstellwinkel sehr wichtig. Damit ein hohes Mass an Flexibilität für die Ausrichtung erreicht werden kann, soll eine Webcam mit USB-Anschluss verwendet werden. Das zentrierte Befestigen der Kamera ist ebenfalls ein wichtiger Punkt.

In einem guten Rahmen sollen auch die Motoren ansteuerbar sein. Mit der vorliegenden Implementierung und Hardware liegt der Arbeitsbereich nur bei 20 Integer-Werten. Durch einen grösseren Arbeitsbereich kann der Regler mehr Werte anfahren und so optimaler arbeiten.

Verbesserungen und Hinweise zur Bildverarbeitung

Die Bildverarbeitung ist ein wichtiger Aspekt dieser Arbeit. Dadurch, dass diese sehr rechenintensiv ist, sollen vor allem hier Verbesserungen durchgeführt werden. Das Problem liegt vor allem darin, dass viele Pixel verarbeitet werden müssen. Wenn man den Bereich in einer ROI (Region of interest) einschränken kann, müssten weniger Pixel analysiert werden, was die Rechenzeit verkürzt. Zudem kann bei passender Kamerapositionierung auf die Vogelperspektive verzichtet werden. Wie im Abschlusstest angesprochen kann der Roboter schlecht Reflexionen von der realen Endmarkierung unterscheiden. Hier sollte ebenfalls noch Zeit für robuste Lösungen investiert werden.

Verbesserungen und Hinweise zur Regelung

Für eine gute Regelung ist es von zentraler Bedeutung, dass die Regelstrecke bekannt ist. Aus diesem Grund werden die technischen Angaben der Motoren benötigt, damit zumindest eine passende Simulation durchgeführt werden kann. Die Abtastzeit der Regelung wird im Moment von der Bildverarbeitung bestimmt. Dies soll vermieden werden. Durch eine unabhängige Abtastzeit kann die Regelung effizienter agieren und Korrekturen schneller durchführen. Dabei muss die Stabilität des Systems noch immer gewährleistet werden.

10.3. Persönliches Fazit

In dieser Bachelorarbeit flossen verschiedene Bereiche zusammen. Die Kombination aus Bildverarbeitung und Regelung war etwas vollkommen neues, was das Ganze sehr spannend machte. Durch das parallele Besuchen des Moduls EBV konnte das gelernte direkt in der Praxis umgesetzt werden. Dies kam vor allem bei der Bildverarbeitung zum Tragen. Auch die Implementierung passender Regler wurde vom Klassenzimmer in dieses Projekt übertragen.

Vor dem Projekt wurde die Rechenzeit des Raspberry Pi bezüglich der Bildverarbeitung unterschätzt. Es wurde angenommen, dass diese viel schneller durchgeführt werden kann. Dies wirkte sich auf die ganze Regelung aus.

Diese Arbeit hat mir zudem aufgezeigt, wie wichtig Kleinigkeiten sein können. Als Beispiel dafür kann die Positionierung der Kamera genommen werden.

Durch die wöchentlichen Sitzungen mit Hr. Martin Friedli konnte das iHomeLab immer auf dem neusten Stand gehalten werden. Bei der Projektplanung war das Einbauen einer Pufferzeit sehr wertvoll. Einzig die Umsetzung der Regelung nahm mehr Zeit in Anspruch, als geplant. Zudem wäre eine genauere Beschreibung der einzelnen Bereiche im Projektplan von Vorteil gewesen.

Dass der Rollator für dieses Projekt nicht zum Einsatz kommen konnte, wird sehr bedauert. Allgemein konnte ich von dieser BDA sehr viel profitieren und nehme sehr viel Erfahrung mit.

11. Literatur- und Quellenverzeichnis

Literatur

Bradski, G. & Kaehler, A., (2008). *Learning OpenCV – Computer Vision with the OpenCV Library* (1. Aufl.). Sebastopol: O'Reilly Media, Inc.

Zahn, K., (Feb. 2017). *Bildverarbeitung - Vorlesungsskript TA.BA_EBV.F1701*. Hochschule Luzern Technik und Architektur

Lutz, H & Wendt W., (2012). *Taschenbuch der Regelungstechnik mit MATLAB und Simulink* (10. Aufl.). Europa-Lehrmittel

Mann, H. & Schiffelgen, H. & Rainer, F., (2009). *Einführung in die Regelungstechnik* (11. Aufl.). Carl Hanser Verlag München

Niederberger, F., (2016). *Rollator Remote – Industrieprojekt FS16* (1.0 Version). Hochschule Luzern Technik und Architektur

Links für die Installation

Raspberry Pi. (2017). Raspberry Pi Homepage. Verfügbar unter
<https://www.raspberrypi.org/> (29.03.2017)

Second Robotics LLC. (2017). *Installing OpenCV 3.2 Computer Vision*. Verfügbar unter
<http://secondrobotics.com/tutorials/2254-2/> (29.03.2017)

Matplotlib. (2017). Installing. Verfügbar unter
<http://matplotlib.org/users/installing.html> (29.03.2017)

Dexter Industries. (2017). Get started – 2. SD Card. Verfügbar unter
<https://www.dexterindustries.com/GoPiGo/getting-started-with-your-gopigo-raspberry-pi-robot-kit-2/sdcard-2/> (29.03.2017)

Dexter Industries. (2017). Installing the GoPiGo Python Library. Verfügbar unter
<https://www.dexterindustries.com/GoPiGo/programming/python-programming-for-the-raspberry-pi-gopigo/installing-gopigo-python-library/> (29.03.2017)

pyimagesearch. (2015). A series of OpenCV convenience functions. Verfügbar unter
<http://www.pyimagesearch.com/2015/02/02/just-open-sourced-personal-imutils-package-series-opencv-convenience-functions/> (29.03.2017)

Links für die Implementierung

Tutorialspoint. (2017). Python – Tutorial. Verfügbar unter
<https://www.tutorialspoint.com/python/index.htm> (29.03.2017)

pyimagesearch. (2016). OpenCV center of contour. Verfügbar unter
<http://www.pyimagesearch.com/2016/02/01/opencv-center-of-contour/> (12.04.2017)

Open Source Computer Vision. (2017). OpenCV modules. Verfügbar unter
<http://docs.opencv.org/3.1.0/index.html> (29.03.2017)

Allgemeine Quellen

Gensace. (2017). Gens ace 1800mah 3S 40C lipo battery. Verfügbar unter
<http://www.gensace.de/gens-ace-1800mah-11-1v-40c-3s1p-lipo-battery-pack.html> (10.04.2017)

Patrick's Notebook. (2017) Patrick's Notebook – Random Things In Life That Amuse Me.
Verfügbar unter
<http://notebook.chaopricha.com/?p=32> (30.05.2017)

Quelle Bilder

Raspberry Pi. (2017). Camera Module V2. Verfügbar unter
<https://www.raspberrypi.org/products/camera-module-v2/> (11.04.2017)

PIMORONI. (2017). Rasbberry Pi 3. Verfügbar unter
<https://shop.pimoroni.com/products/raspberry-pi-3> (11.04.2017)

Python Programming. (2017). Supplies Needed- Robotics with Python Raspberry Pi and GoPiGo p.2.
Verfügbar unter
<https://pythonprogramming.net/supplies-needed-raspberry-pi-gopigo-robot/> (11.04.2017)

OpenCV Python Tutorials. (2017). Welcome to OpenCV-Python !!! Verfügbar unter
<http://opencvpython.blogspot.ch/2011/04/welcome-to-opencv-python.html> (11.04.2017)

Mouser. (2017). NXP Freedom Boards. Verfügbar unter
<http://eu.mouser.com/new/nxp-semiconductors/freescale-freedom-boards/> (17.05.2017)

Kidmakers. (2017). DC-Motor. Verfügbar unter
<http://www.kidmakers.org/motors-a-guide-to-give-your-projects-go/> (17.05.2017)

A. Anhang - Erweiterte Projektdokumente

Inhalt dieses Abschnitts

- Vorgegebene Aufgabenstellung
- Projektauftrag
- Risikoanalyse
- Projektplan
- Anforderungsspezifikation
- Streckenspezifikationen
- Arbeitsjournal

Horw, 20. Februar 2017
Seite 1/3

Diplomarbeit im Fachbereich Elektrotechnik

Aufgabe für Herrn Stecher Silvano

Autonomous Rollator

Fachliche Schwerpunkte

Automation & Embedded Systems, Energiesysteme & Antriebstechnik

Einleitung

Das iHomeLab der Hochschule Luzern ist Schweizer Denkfabrik und Forschungszentrum für Gebäudeintelligenz. Im Forschungsbereich Ambient Assisted Living (AAL) werden Assistenzsysteme erforscht, welche situationsabhängig und unaufdringlich zu einer Steigerung der Lebensqualität beitragen. Dazu gehört auch ein Rollator mit Elektroantrieb (iWalkActive), der ähnlich einem E-Bike die Nutzenden beim Bergauf- und Bergabfahren unterstützt. Der so ausgerüstete Rollator könnte noch so erweitert werden, dass er selbstständig zur Ladestation oder von der Ladestation wieder zur Person fährt, die ihn benutzen möchte.

Aufgabenstellung

In der PAIND „RollatorRemote“ wurde ein Rollator mit Elektroantrieb so erweitert, dass er mit einem Joystick ferngesteuert werden kann. Die Fernsteuerung hat eine Schnittstelle, welche es ermöglicht, den Rollator auch mit einem Embedded-Computer zu steuern.

Die Pixy CMUcam5 ist eine Kamera die farbige Objekte erkennen kann. Diese Kamera soll nun verwendet werden, um den Rollator z.B. entlang einer roten Linie oder einer Reihe von LEDs fahren zu lassen. So könnte der Rollator auch selbstständig zur Ladestation fahren.

Aufgaben:

- Objekterkennung (rote Linie, LEDs, ...) implementieren resp. in Pixy konfigurieren und testen.
- Regelalgorithmen für die Objektverfolgung erarbeiten.
- Implementieren der Regelalgorithmen, so dass Fahrbefehle für den Rollator erzeugt werden können.
- Programm auf einem Embedded-Computer erstellen, damit Fahrbefehle an die in „RollatorRemote“ entwickelte Fernbedienung/Schnittstelle übermittelt werden können.
- Der Rollator muss stehen bleiben, wenn kein Objekt erkannt wird. Und er muss auf einen Befehl hin losfahren, wenn Objekte erkannt werden.

Erwartete Ergebnisse

- Anforderungen sind mit Betreuer / Co-Betreuer zu erarbeiten
- Vergleich und Bewertung von Varianten geeigneter Regelalgorithmen
- Wahl der bestgeeigneten Variante für den Regelalgorithmus
- Wahl eines geeigneten Embedded Systems
- Konzept der Umsetzung und Tests
- Umsetzung
- Test des Rollators mit Testbericht

Termine

Start der Arbeit:	Montag 20.2.2017
Zwischenpräsentation:	Zu vereinbaren im Zeitraum-.....2017
Abgabe Broschüre-Doku:2017, per Mail an Betreuer und H. R Andrist
Abgabe Schlussbericht:	Montag, 9.6.2017, vor 16:00 im Sekretariat
Abgabe Poster-File:2017 per Mail an Betreuer und H. R. Andrist
Abschlusspräsentation:	Zu vereinbaren im Zeitraum-.....2017

Dokumentation

Der gebundene Schlussbericht ist in 4-facher Ausführung zu erstellen. Er enthält zudem zwingend

- die folgende Selbstständigkeitserklärung auf der Rückseite des Titelblattes:
„Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche verwendeten Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.
Horw, Datum, eigenhändige Unterschrift”
- einen englischen Abstract mit maximal 2000 Zeichen.
- Ein Titelblatt mit: Name des Studierenden, Titel der Arbeit, Abgabedatum, Dozent, Experten, Abteilung, Klassifikation (Einsicht/Rücksprache/Sperre)
- Eine CD-Hülle, innen, auf der Rückseite des Berichtes

Alle Exemplare des Schlussberichtes müssen termingerecht abgeben werden. Zusätzlich muss zu jedem Exemplar eine CD mit dem Bericht (inkl. Anhänge), dem Poster und den Präsentationen, Messdaten, Programmen, Auswertungen, usw. unmittelbar nach der Präsentation abgeben werden.

Ein Poster sowie Unterlagen für eine Diplomarbeitsbroschüre sind gemäss den offiziellen Layout-Vorgaben termingerecht einzureichen.

Fachliteratur/Web-Links/Hilfsmittel

Die Dokumentation zur Ansteuerung des Rollator Elektroantriebs aus der PAIND „Rollator-Remote“ wird abgegeben.

Geheimhaltungsstufe: Sperre

Horw, 10.2.2017
Seite 3/3
Diplomarbeit im Fachbereich Elektrotechnik

Verantwortlicher Dozent/Betreuungsteam, Industriepartner

Dozent	Prof. Zeno Stössel	zeno.stoessel@hslu.ch
Co-Betreuer	Martin Friedli	martin.friedli@hslu.ch
Industriepartner	intern, iHomeLab	
Experte	Thomas Schmidiger Tel. 041 660 01 85	thomas.schmidiger@maxonmotor.com

Hochschule Luzern
Technik & Architektur

Prof. Zeno Stössel

Projektauftrag - Autonomous Rollator

1. Ausgangslage

Am iHomeLab der Hochschule Luzern (HSLU) wurde ein Rollator entwickelt, der die Anwender unterstützt, wenn sich diese den Hang hoch oder hinab bewegen. Im Herbstsemester 2016 wurde während des Projekts "PAINd" der Rollator so modifiziert, dass dieser sich Remotemässig über eine Fernsteuerung ansteuern lässt. Nun soll dieses Projekt weitergeführt werden, so dass der Rollator selbständig und autonom einem Weg folgen kann. Da der Rollator nicht verwendet wird, soll als Ersatz ein passender Roboter gewählt werden. Der Weg ist mittels LEDs oder einer rot markierten Strecke vorgegeben. Die Idee besteht darin, dass der Rollator selbständig den Weg zur Ladestation findet, um sich dort aufzuladen.

Die Erkennung der Strecke soll vorzugsweise mittels einer Kamera realisiert werden. Andere mögliche Varianten um den Weg zu erkennen sollen nicht ausgeschlossen werden. Am Ende soll die bestmögliche Variante umgesetzt werden, damit der Rollator die Strecke korrekt folgen und passende Parameter für die Motorensteuerung übergeben werden. Da das Befahren einer Strecke voraussichtlich im Innenbereich stattfindet, sollen die Messungen und die Funktionstests drinnen stattfinden. Die Länge der Strecke soll auf 10 Meter begrenzt sein.

2. Qualitative Zielsetzung

Der Roboter soll bis Ende des Frühlingssemester 2017 autonom eine Strecke im Innenbereich abfahren können und an dessen Ende anhalten. Die markierte Strecke ist dabei vom Untergrund unterscheidbar.

In einem ersten Schritt soll eine passende Variante evaluiert werden, um die Strecke zu erkennen und auszuwerten. Dazu bieten sich verschiedene Möglichkeiten an. Diese werden bewertet und zum Schluss eine ausgewählt. Die erhaltenen Daten von der Wegerkennung werden verwendet um die Motoren anzusteuern.

In einem zweiten Schritt soll eine passende Schnittstelle für die Steuerung gefunden und implementiert werden.

Die Geschwindigkeit des Rollators soll in einem angemessenen Rahmen stattfinden. Es soll nicht grösser als die Gehgeschwindigkeit eines durchschnittlichen Menschen sein.

3. Projektplan

Ein Projektplan wurde mittels Project Professional erstellt. Die Meilensteine sind mit «Abschluss xxx» benannt. Es sind folgende Meilensteine vorhanden:

- Abschluss Initialisierung
- Abschluss Einarbeitung/Lösungsfindung
- Abschluss Wegdetektion
- Abschluss Schnittstelle
- Abschluss Funktionstest
- Abschluss Projekt

Das PDF des Projektplans wird im Projektordner abgelegt.

Die Zwischenpräsentation findet am 1. Mai 2017 an der HSLU statt.

3.2 Risikobeurteilung

Die Risikobeurteilung wurde ebenfalls im Projektordner abgelegt. Hierbei wurden die Risiken in technische Risiken und organisatorische Risiken unterteilt.

3.3 Mittelplanung

Schon zu Beginn des Semesters soll der Rollator auf seine Funktionsfähigkeit geprüft werden. Es soll ein kleiner Funktionstest vorgenommen werden. Mittels der vorhandenen Dokumentationen soll der Rollator analysiert werden, um ein genaues Bild über die Funktionsweise zu bekommen. Material für die Inbetriebnahme des Rollators wird vom iHomeLab zu Verfügung gestellt.

Zudem soll Kontakt mit Hr. Fabian Niederberger aufgenommen werden, um Unklarheiten zu beseitigen und projektspezifische Daten austauschen zu können.

Bestellungen sollen früh genug in Auftrag gegeben werden. Bei langen Lieferfristen soll falls nötig ein anderer Anbieter gewählt werden. Die Bestellungen werden vom Elektroniklabor der Hochschule Luzern vorgenommen. Bei der Herstellung von Leiterplatten sollen diese früh genug in Auftrag gegeben, da es sehr oft zu Engpässen kommt.

3.4 Schnittstelle

Damit das iHomeLab auf dem aktuellsten Stand ist, werden wöchentlichen Treffen stattfinden. An diesen werden mögliche Probleme, Entscheidungen, der aktuelle Stand oder Beschlüsse besprochen werden.

Um die Motoren ansteuern und die Odometriedaten auslesen zu können muss der CAN-Bus verstanden werden. Hier soll eine genaue Recherche vorgenommen werden, damit die Daten zuverlässig ausgelesen werden können. Mittels Test soll die Verständlichkeit verbessert werden.

3.5 Vorgehensvorschlag 1. Projektetappe

Im ersten Teil wird das vorhandene Produkt analysiert und getestet. Die vorhandene Dokumentation des Industrieprojekts „Rollator Remote“ soll durchgelesen werden und die Funktionsweise verstanden werden. Anschliessend werden mögliche Lösungen erarbeitet, um eine Wegstrecke zu erkennen. Dies kann zum Beispiel mit der Pixy CMUcam5 realisiert werden, mit Infrarotsensoren oder via eine Quelle, die ein Signal sendet. Diese Möglichkeiten sollen bewertet werden und die am besten geeignete wird dann realisiert. Vorzugsweise soll eine Kamera verwendet werden.

Zudem sollen mögliche Regelalgorithmen genauer angesehen werden und ebenfalls bewertet werden.

Falls nötig muss auf eine leistungsfähigere Hardware gewechselt werden. Nach Abschluss dieser Evaluierung wird mit der Implementierung der Wegdetektion begonnen.

Das weitere Vorgehen ist dem Projektplan zu entnehmen.

Risikoanalyse

Um mögliche Risiken die während des Projekts auftreten können zu minimieren, wird eine Risikoanalyse erstellt. Dabei werden die Risiken in organisatorische und technische Risiken unterteilt. Sie werden nach Eintrittswahrscheinlichkeit und Schadenspotential bewertet.

Diese Tabelle ist nicht abschliessend und die Werte der einzelnen Kategorien können sich während des Projekts ebenfalls noch ändern.

Die unten aufgeführte Risikomatrix zeigt den Schaden, welche die aufgelisteten Risiken auf das Projekt haben können. Es wird auch ersichtlich welche Risiken während des Projekts im Auge behalten werden müssen.

Die organisatorischen Risiken sind mit Nummern von 10 bis 19 bezeichnet.

Die technischen Risiken sind mit Nummern von 20 bis 29 bezeichnet.

Eintrittswahrscheinlichkeit	Schadenspotential				
	1 - unbedeutend	2 - gering	3 - spürbar	4 - kritisch	5 - Existenz bedrohend
	5 - gross				
	4 - möglich		13,20,22, 25,26	11,29	
	3 - gering		21	12,14,16, 23,24,26	15
	2 - sehr gering	28			
	1 - unwahrscheinlich		10,17		

Organisatorische Risiken

Nr.	Beschreibung	Massnahmen	Eintrittswahrscheinlichkeit	Schadenpotential
10	Lieferungen von Bauteile nicht möglich/lange Lieferzeiten	Frühzeitig bestellen, Liefertermine berücksichtigen	1	3
11	Unklarheiten bestehendes Produkt, nicht dokumentiert	Kontaktaufnahme mit Vorgänger oder Betreuer	4	4
12	Krankheit/Abwesenheit	Pufferzeit einplanen	3	4
13	Fehlerhafte Planung	Genaues Besprechen mit dem Betreuer über die nächsten Arbeitsschritte	4	3
14	Datenverlust bei der Dokumentation	Ständige Backups machen	3	4
15	Datenverlust der Implementierung	Versionskontrollsystem verwenden (Git) und Backups machen	3	5
16	Festgelegte Lösungsvariante kann nicht gewünscht umgesetzt werden	Projektanforderungen müssen angepasst werden, auf Alternativlösung ausweichen	3	4
17	Engpass bei Produktion	Produktion der Platine früh einplanen	1	3

Tabelle 1: Organisatorische Risiken

Eintrittswahrscheinlichkeit

- 1 – unwahrscheinlich
- 2 – sehr gering
- 3 – gering
- 4 – möglich
- 5 – gross

Schadenpotential

- 1 – unbedeutend
- 2 – gering
- 3 – spürbar
- 4 – kritisch
- 5 – Existenz bedrohend

Technische Risiken

Nr.	Beschreibung	Massnahmen	Eintrittswahrscheinlichkeit	Schadenpotential
20	Lichtverhältnisse ändern sich	Konstante Verhältnisse gewährleisten, Filter implementieren	4	3
21	Variables Umfeld (andere Räumlichkeiten)	Kann je nach Lösungswahl problematisch (Infrarotsensor, Gegenstände auf Strecke)	3	4
22	Bildverarbeitung zu langsam für Regler	Passende Hardware verwenden, optimieren, Schnelligkeit anpassen	4	3
23	Schnittstelle funktioniert nicht	Genaues informieren, nachfragen an Fachleuten, Debugging	3	4
24	Hardware nicht ausreichend	Andere Hardware verwenden, Optimierung an aktueller Hardware	3	4
25	Langsamer und/oder ungenauer Regler	Besseren Regler verwenden, Anpassungen Parameter (P,I,D-Anteil), Schnelligkeit anpassen	4	3
26	Regler wird instabil	Standardregelverfahren, Parameteranpassung	4	3
27	Wechselnder Untergrund	Beschränkten Untergrund festlegen mit erkennbaren Farben für Strecke	3	4
28	Energiespeicher nicht ausreichend	Berechnung benötigte Energie	2	2
29	Probleme bezüglich vorhandener Hardware (Motoren zu wenig Drehmoment, Kabel kaputt etc.)	Andere Motoren, Anforderungen anpassen, Aufbauinformationen Produkt beschaffen	4	4

Tabelle 2: Technische Risiken

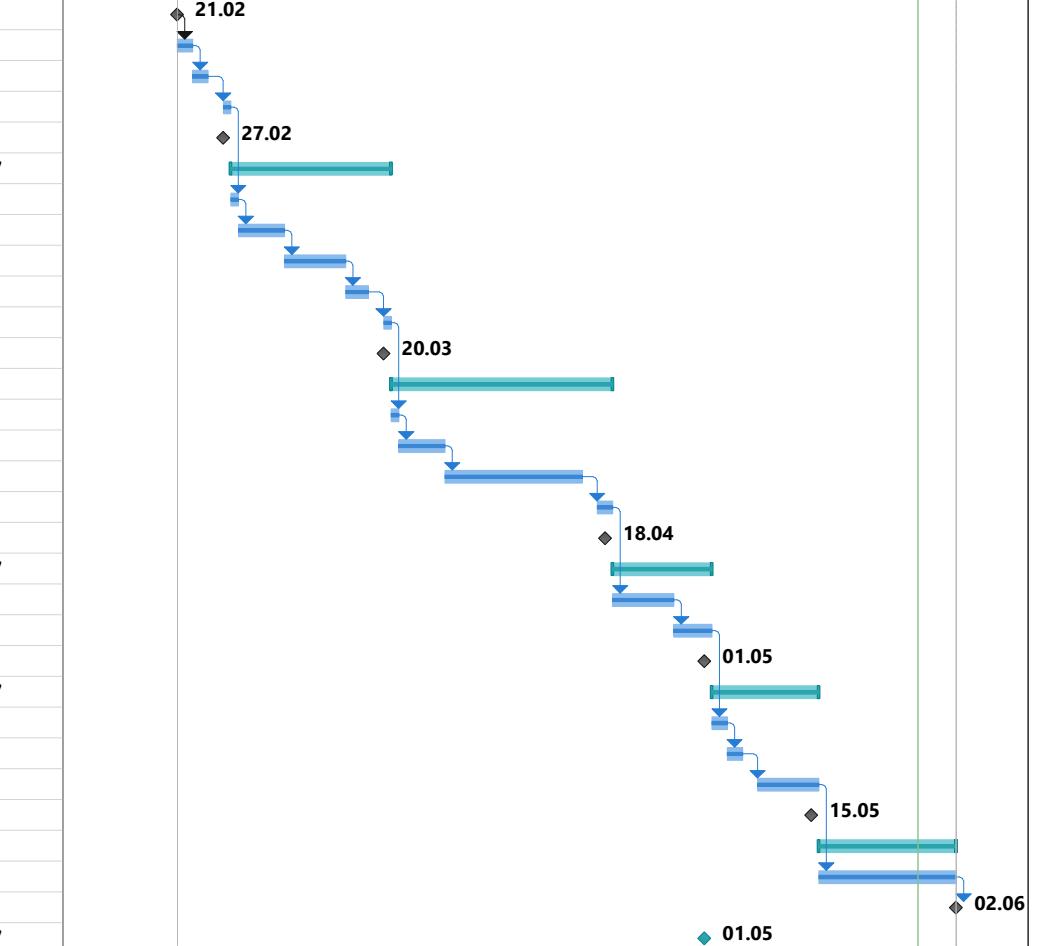
Eintrittswahrscheinlichkeit

- 1 – unwahrscheinlich
- 2 – sehr gering
- 3 – gering
- 4 – möglich
- 5 – gross

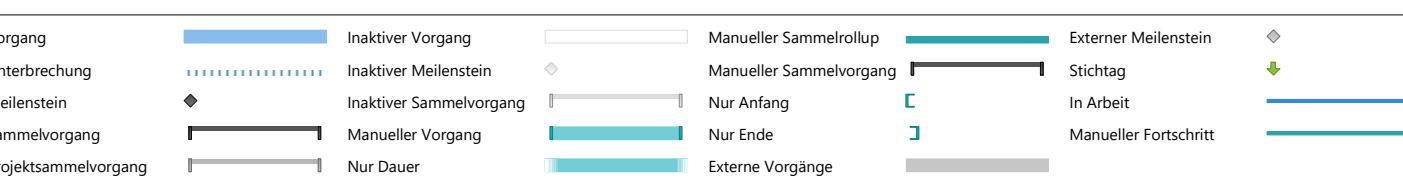
Schadenpotential

- 1 – unbedeutend
- 2 – gering
- 3 – spürbar
- 4 – kritisch
- 5 – Existenz bedrohend

Nr.	Vorgangs	Vorgangsname	Dauer	Anfang	Ende	Jan 2017	06.02	13.02	20.02	März 2017	06.03	13.03	20.03	27.03	03.04	10.04	17.04	24.04	Mai 2017	01.05	08.05	15.05	22.05	Juni 2017	29.05	05.06
1	Start	Dokumentation	74 Tage	Die 21.02.17	Fre 02.06.17																					
2	Start	Initialisierung	5 Tage	Die 21.02.17	Mon 27.02.17																					
3	Start	Kick-Off Gespräch	0 Tage	Die 21.02.17	Die 21.02.17																					
4	Start	Projektplanung	2 Tage	Die 21.02.17	Mit 22.02.17																					
5	Start	Aufgabenstellung definieren	2 Tage	Don 23.02.17	Fre 24.02.17																					
6	Start	Risikobeurteilung	1 Tag	Mon 27.02.17	Mon 27.02.17																					
7	Start	Abschluss Initialisierung	0 Tage	Mon 27.02.17	Mon 27.02.17																					
8	Start	Einarbeitung und Lösungsfund	15 Tage	Die 28.02.17	Mon 20.03.17																					
9	Start	Alte Projektdaten sammeln	1 Tag	Die 28.02.17	Die 28.02.17																					
10	Start	Ansteuerung mit akutellem Stand und Analyse	4 Tage	Mit 01.03.17	Mon 06.03.17																					
11	Start	Lösungsfundung Wegdetektierung	6 Tage	Die 07.03.17	Die 14.03.17																					
12	Start	Lösungsfundung Schnittstelle	3 Tage	Mit 15.03.17	Fre 17.03.17																					
13	Start	Lösungen bewerten/entscheiden	1 Tag	Mon 20.03.17	Mon 20.03.17																					
14	Start	Abschluss Einarbeitung/Lösungsfundung	0 Tage	Mon 20.03.17	Mon 20.03.17																					
15	Start	Wegdetektierung	21 Tage	Die 21.03.17	Die 18.04.17																					
16	Start	Installation Software	1 Tag	Die 21.03.17	Die 21.03.17																					
17	Start	Aufsetzen der Hardware	4 Tage	Mit 22.03.17	Mon 27.03.17																					
18	Start	Implementierung Wegdetektion	14 Tage	Die 28.03.17	Fre 14.04.17																					
19	Start	Test Wegdetektierung	2 Tage	Mon 17.04.17	Die 18.04.17																					
20	Start	Abschluss Wegdetektion	0 Tage	Die 18.04.17	Die 18.04.17																					
21	Start	Schnittstelle/Regelung	9 Tage	Mit 19.04.17	Mon 01.05.17																					
22	Start	Implementierung Schnittstelle/Regelung	6 Tage	Mit 19.04.17	Mit 26.04.17																					
23	Start	Test Schnittstelle	3 Tage	Don 27.04.17	Mon 01.05.17																					
24	Start	Abschluss Schnittstelle	0 Tage	Mon 01.05.17	Mon 01.05.17																					
25	Start	Funktionstest	10 Tage	Die 02.05.17	Mon 15.05.17																					
26	Start	Funktionstest mit Roboter	2 Tage	Die 02.05.17	Mit 03.05.17																					
27	Start	Testbericht erstellen	2 Tage	Don 04.05.17	Fre 05.05.17																					
28	Start	Anpassungen vornehmen	6 Tage	Mon 08.05.17	Mon 15.05.17																					
29	Start	Abschluss Funktionstest	0 Tage	Mon 15.05.17	Mon 15.05.17																					
30	Start	Abschluss	14 Tage	Die 16.05.17	Fre 02.06.17																					
31	Start	Pufferzeit/Fertigstellung/Sonstiges	14 Tage	Die 16.05.17	Fre 02.06.17																					
32	Start	Abschluss Projekt	0 Tage	Fre 02.06.17	Fre 02.06.17																					
33	Start	Präsentation	0 Tage	Mon 01.05.17	Mon 01.05.17																					



Projekt: Projektplan_BDASilvan
Datum: Mon 29.05.17



Anforderungsspezifikation				
Nr.	F M W	Bezeichnung	BAA+E.FS17 Autonomous Rollator	
			Werte Daten Erläuterungen Änderungen	Erfüllt
1		Allgemein		
1.1	F	Vorhadene Hardware nutzen		✗
1.2	F	Wegmarkierung folgen		✓
1.3	F	Wegdetektion		✓
1.4	F	passende Parameterübergabe für Roboter		✓
1.5	F	wenn keine Spur anhalten		✓
1.6	F	sobald Linie detektiert diese folgen		✓
1.7	W	verschiedene Befehle verarbeiten	von Fernbedienung	✗
1.8	F	Anwendungsbereich	Indoor	✓
2		Roboteransteuerung		
2.1	F	Vorhandener Roboter verwenden	iHomeLab Rollator oder externer Roboter	✓
2.2	W	Geschwindigkeit Roboter	Gehgeschwindigt	✗
3		Schnittstelle		
3.1	F	Wertübergabe	analog oder direkt implementiert	✓
4		Richtwerte Befahrung		
4.1	F	Stabiles Verhalten		✗
4.2	F	Start/Stopp	bei Streckendetektion starten, bei Ende Strecke anhalten	✓
4.3	F	Autokorrektur	automatisches Rückfinden auf Strecke	✓
4.4	W	konstantes Fahren		✗
4.5	W	Vorhersage Strecke		✗
4.6	W	Störungen/Fehldetektion ausblenden	Iritierende Objekte im Detektionsbereich	✗
4.7	W	selbstständig nach Spur suchen	wenn keine Spur vorhanden diese suchen (im Kreis drehen)	✓
5		Umgebungsbediengungen		
5.1	F	Untergrund	Steigung < 1%	-
5.2	F	Oberflächenbeschaffenheit	Haushaltliche Bedingungen, keine Stufen oder Hindernisse, kein Teppich (ESD)	-
5.3	F	Kleinere Störungen dürfen vorhanden sein	Lichreflexionen	-
6		Kosten		
6.1	F	Nich definiert, falls möglich mit vorgegebenen Setup arbeiten	nach Absprache mit iHomeLab	-
7		Wegmarkierung	Dokument: <i>Streckenspezifikationen</i>	
7.1	F	Streckenmarkierung	rote Klebestreifen	✓
7.2	F	Abstände einhalten	gemäss Streckenspezifikationen	✓
7.3	F	max. Länge	10m	-
7.4	F	Kurven	direkter Weg zum Ziel ohne Kreise oder überschneidungen	✓
7.5	F	Markierung Ende	rote LEDs	✓
7.6	F	Farbhintergrund	Kanten erkennbar zwischen Markierung und Untergrund	✓
7.7	F	Kurvenradius	max. 35cm	✓

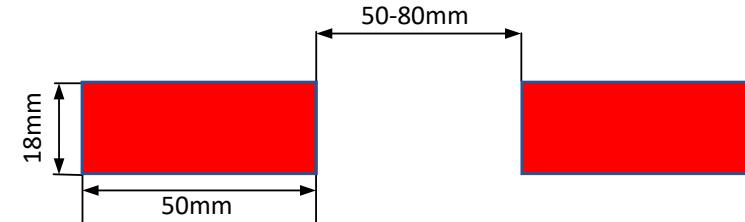
Änderungsjournal

Version	Datum	Änderung	Verantwortlich
v1	22.02.2017	Erstellung	S.Stecher
v1.1	25.03.2017	Anpassungen nachdem Rollator technische Störung erleidet hat	S.Stecher
v1.2	30.04.2017	Anpassungen an Wegmarkierung, Kantenerkennung	S.Stecher
v1.3	22.05.2017	Kurvenradius ergänzt, Erfüllung getestet	S.Stecher

Hinweis: Ergänzungen oder Anpassungen können während des Projekts in Absprache mit dem Co-Betreuer vorgenommen werden. Die Streckenspezifikationen mit Abbildungen werden in einem separatem Dokument aufgeführt.

Streckenspezifikationen

Streckenende

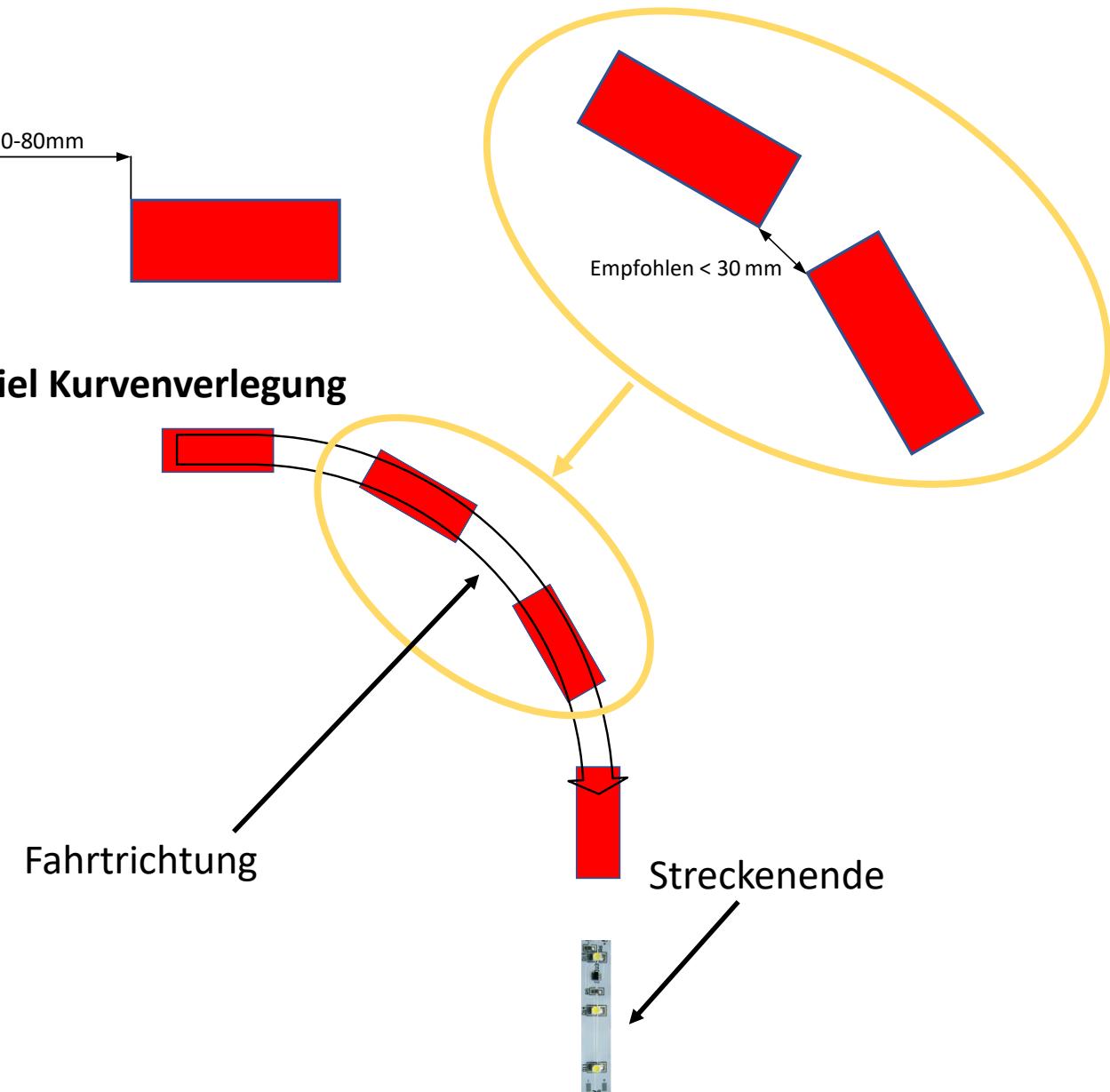


Hinweise für LED-Stripes:

Elek. Angaben beziehen sich auf 1m Länge
Conrad-Art. Nr.: 180697 - 62

	Werte
Allgemein	direkter Weg
Max. Länge	10m
Kreuzungen	keine
Kurven	nicht eckig
Hindernisse	keine
Farbe Strecke	rot
Störungen	Kleinere Reflexionen erlaubt
Kurvenradius	350mm

Beispiel Kurvenverlegung



Autonomous Rollator

Verfasser: Silvano Stecher

Datum	Gesamte Zeit pro Tag [h]	Anzahl Stunden										Arbeitschritte	Probleme	Fragestellungen	Zwischenergebnisse
		Administratives	Pläneung	Dokumentation	Einarbeitung	Risikoanalyse	Technologie Recherche/Lösungsfindung	Wegdetektierung	Schnittstelle Regler	Messungen und Test vornehmen	Präsentation vorbereiten				
21.02.2017	5	3	2									Kick off Meeting, Erstellung Projektplan und Vorlagen von Dokumenten			
22.02.2017	8	2	2	1	3							Risikobeurteilung erstellt, Projektantrag erstellt, Anforderungsspezifikationen, Doku Remote Rollator gelesen			
28.02.2017	7	4	1	2								Abschluss Projektantrag, Besprechung mit Fabian Niederberger, Git erstellt, Einarbeitung Projekt	Motor zu wenig Drehmoment, beschränkte Dokumentation		Alle Daten erhalten
01.03.2017	8	3					5					Recherche Echtzeitfähigkeit Pi, Wochensitzung inkl. Protokoll, PixyCam recherchiert			
04.03.2017	3	1		1	1							Recherche PixyCam, Erstellung Lösungsbeurteilung			
05.03.2017	3			1	2							Erste Tests PixyCam+Arduino und PixyCam+RaspberryPi3			
06.03.2017	9		3		6							Spurdetektion PixyCam test, Bewertung Lösungsvarianten erstellt, Doku ergänzt			
07.03.2017	2			1	1							Dokumentation Rollator Remote angeschaut, Lösungsvariante angepasst			
08.03.2017	9	5	3				1					Sitzungsvorbereitung und Durchführung, Protokoll schreiben, Wochensitzung, Bestellung LED Streifen, Raspberry aufgesetzt, CAN-Bus bestellt	CAN-Bus Shield kaputt		
09.03.2017	2			2								Raspberry Pi aufsetzen (Raspbian Pixel)			
12.03.2017	1			1								Raspberry Pi aufsetzen			
13.03.2017	9	1					8					Hotspot erstellt, Versucht Rollator zu flicken, PixyCam installiert, erste kleinere Implementierung	implementierung eher schwierig mit Pixy		soll auf Raspicam wechsel
14.03.2017	8	2		2	4							Sitzung vorbereitet, PixyCam und Python Werte in Txt-File schreiben, Inbetriebnahme LED-Stripes, Dokumente angepasst			
15.03.2017	7	6	1									Projektplan update, erste Bildaufnahme Wegstrecke Raspberry Pi, Roboter ausgesucht, Sitzung			als Ersatz für den Rollator wird GoPiGo Roboter verwendet
19.03.2017	6			6								Installation OpenCV und sonstiges auf Pi			
20.03.2017	9			5	1	3						Zusammenbauen Roboter und Inbetriebnahme, Implementierung			12V/1,5A benötigt
21.03.2017	10				10							Wegdetektion und Bird-eye			
22.03.2017	4	3				1						Vorbereitung Sitzung, Sitzung, Protokoll, Bestellung Akku, Implementierung Python			Bestellung Akku
26.03.2017	2	2										Anpassungen Anforderungen, Erstellung Streckenspezifikationen			
27.03.2017	10	2				8						Versucht max im Bild zu finden, SlidingWindow, Sitzung vorbereitet, Sitzung, Protokoll, Hough Transformation und Winkel ausgeben			von Martin Akkus erhalten
28.03.2017	9		2		1	6						Houghtransformation und Winkelberechnung, auf Raspi geladen, Doku, Suche nach Ersatz für Kugelrolle			
02.04.2017	4		2		2							Suche nach Ersatz für Kugelrolle und Kamerahalterung, Doku			Im Coop Bau+Hooby gefunden
09.04.2017	4		2		2							Doku, Wegdetektion implementierung Motoren			
10.04.2017	9	3				6						Mechanische Anpassungen, Video mit Houghs und Perspektive, Sitzung, Sitzungsvorbereitung			
11.04.2017	10		3		4	3						Video mit Winkelausgabe und Ansteuerung Motoren, Test gerade Strecke, Doku	Anpassungen an Code vornehmen		Video
12.04.2017	9		4		5							Test gerade aus, Bilder aufgenommen Kurve, Doku, Threshold and Center of Mass			
21.04.2017	2		2									Dokumentation gearbeitet			
22.04.2017	4		3				1					Doku und Präsentation Vorbereitet			
23.04.2017	3		2				1					Doku und Präsentation Vorbereitet			
24.04.2017	10	3	2			3	1	1				Sitzung vorbereitet, Messung Stromverbrauch hinzugefügt, Kabel für Speisung gemacht, Anpassungen Code, Implementierung Kurvebefahren			Anforderungen an Strecke wird angepasst
25.04.2017	11	2				8		1				Implementierung PID-Regler Kurve, Besprechung mit Martin, Präsentation gearbeitet			
26.04.2017	4					4						PID-Regler Kurvenbefahrung	Schwierig einzustellen passende Parameter		
29.04.2017	8				8							PID-Regler Kurvenbefahrung			
30.04.2017	6					6						Präsentation vorbereitet			
01.05.2017	11				5	4	2					Präsentation vorbereitet, Präsentation durchgeführt, PID-Regler, Lösungssuche Codeoptimierung schneller Ablauf	Totzeit beträgt 1,2s --> schwierig zu regeln		Präsentation durchgeführt
02.05.2017	9	2			4	3						Lösungssuche Codeoptimierung schneller Ablauf, Sitzung vorbereitet, Sitzung, Protokoll schreiben			
03.05.2017	5					4	1					Detektion Mittellinie, Performance auf 2 Raspi (OpenCV 3.2)			deutlich schneller!
05.05.2017	2		1	1								Backup erstellt, Doku gearbeitet			Backup auf HD
06.05.2017	4		4									Doku			
07.05.2017	3	3										Doku			
08.05.2017	9				7	2						Ende der Strecke implementiert, Anpassungen Reglers			
09.05.2017	8	2					4	2				Test Ende der Strecke detektieren, Reglerentwurf, Sitzung vorbereitet und durchgeführt	PID-Regler kommt kaum zum tragen da noch immer grosste Totzeit		
14.05.2017	8		8									Doku			
15.05.2017	12		4			6	2					Doku, Bilder erstellt, Regler hinzugefügt			
16.05.2017	11	2	3			6						Doku, Sitzung durchgeführt, Protokoll, Idee Gedächtnis einbauen, PID-Funktion überprüft			
17.05.2017	3		2			1						Doku, Regler optimiert			
20.05.2017	11	9		2								Reglerentwurf mit Matlab, Doku ergänzt und überarbeitet			
21.05.2017	4		4									Doku vorbereitet für Besprechung			
22.05.2017	6	3			3							Erstellung Reglerentwurf inkl. Regelstrecke, Doku nachgetragen			
23.05.2017	10	2	2		6							Backup erstellt, Reglerentwurf verbessern, Erstellung Broschürenbericht			
24.05.2017	9	3	3		3							Erstellung Broschürenbericht, Teststrecke erstellt, Optimierung Zentrierung			
25.05.2017	2		2									Abschluss geschrieben			
28.05.2017	6	6										Korrektur Dokumentation			
29.05.2017	7				5	2						Abschlusstest, automatischer Start Programm versucht zu implementieren			
30.05.2017	6	2	4				3					Doku überarbeitet, Start Programm automatisch beim aufstarten implementiert, Sitzung vorbereitet, Protokoll			
04.06.2017	3											Abschlusspräsentation vorbereitet			
05.06.2017	4	4										Dokumentation korrigiert, fertiggestellt			
06.06.2017	6	6										Dokumentation korrigiert, fertiggestellt, ausgedruckt			
07.06.2017	6	2				2	2					Cleancoding, Abgabe Dokumentation, Präsentation fertiggestellt			

Total

380 55 9 95 22 4 45 57 64 10 17 2

B. Anhang - Zusätzliche inhaltliche Unterlagen

Befehle Ansteuerung GoPiGo Python:

Motor control Functions:

fwd(): Move the GoPiGo forward with PID (better control)
motor_fwd(): Move the GoPiGo forward without PID
bwd(): Move the GoPiGo back with PID (better control)
motor_bwd(): Move the GoPiGo back without PID
left(): Turn GoPiGo Left slow (one motor off, better control)
left_rot(): Rotate GoPiGo left in same position (both motors moving in the opposite direction)
right(): Turn GoPiGo right slow (one motor off, better control)
right_rot(): Rotate GoPiGo right in same position both motors moving in the opposite direction
stop(): Stop the GoPiGo

Motor speed Functions:

increase_speed(): Increase the speed of the GoPiGo by 10
decrease_speed(): Decrease the speed of the GoPiGo by 10
set_left_speed(): Set speed of the left motor
set_right_speed(): Set speed of the right motor
set_speed(): Set speeds of both the motors

Encoder Functions:

enc_tgt(): Set encoder target to move the GoPiGo to a set distance
enable_encoders(): Enable the encoders
disable_encoders(): Disable the encoders

Ultrasonic ranger read:

us_dist(): Read distance from the ultrasonic sensor

LED control:

led_on(): Turn LED on
led_off(): Turn LED off

Servo control:

enable_servo(): Enables the servo
disable_servo(): Disables the servo
servo(): Set servo position

Status from the GoPiGo:

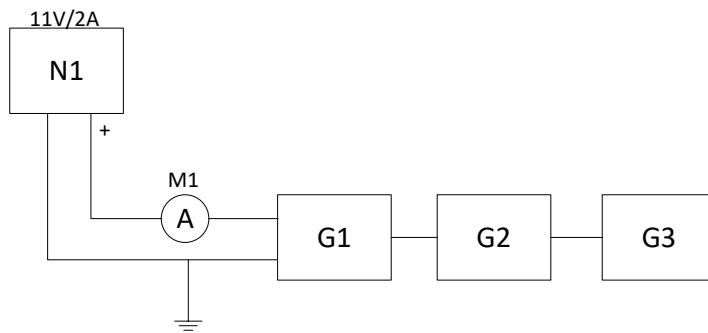
volt(): Read battery voltage in V
fw_ver(): Get the firmware version of the GoPiGo
enable_com_timeout(): Enable communication time-out(stop the motors if no command received in the specified time-out)
disable_com_timeout(): Disable communication time-out
read_status(): Read the status register on the GoPiGo
read_enc_status(): Read encoder status
read_timeout_status(): Read timeout status

Quelle:

<https://www.dexterindustries.com/GoPiGo/programming/python-programming-for-the-raspberry-pi-gopigo/> (09.04.2017)

Auslegung Akkumulator für GoPiGo

Da der Verschleiss von Batterien ziemlich gross ausfallen würde, wurde beschlossen ein LiPo-Akkumulator zu verwenden. Aus diesem Grund wurde der Stromverbrauch gemessen. Der Aufbau für die Stromverbrauchsmessung ist in der unterstehenden Abbildung ersichtlich. Das verwendete Programm ist das gleiche wie im Testbericht 1 (*video_firstTest.py*).



N1: *Netzgerät Dr. K. Wittmer, Herrliberg Zürich*
 M1: *Ampermeter Fluke 179*
 G1: *GoPiGo mit Motoren*
 G2: *Raspberry Pi 3 Model B*
 G3: *Raspberry Pi V2 Camera*

Abbildung 1: Messaufbau Strommessung Raspberry Pi und GoPiGo

Das Ergebnis der Messung ist in der folgenden Tabelle ersichtlich. Dabei wird unterschieden, ob die Motoren laufen oder nicht und ob das Raspi voll ausgelastet ist. Falls einer dieser Fälle zutrifft wird das mit einem Kreuz markiert.

Raspberry Pi Volllast	Motoren ein	Strom in mA
		300
X		550
X	X	1500
X		570
X	X	1300
X	X	1400
X		570
X		590
X	X	1000
X		560
X	X	1400
X		570
X	X	1100

Der höchste Strom der gemessen wurde, beläuft sich auf 1.5A.
 Damit der Roboter eine Stunde in Betrieb sein kann, wurde ein Akku von 1800mAh ausgewählt. Die gewählte Spannung beträgt 11.1V was 3x3.7V Zellen entspricht.

Das Laden des Akkumulator wird mit der schon vorhandenen Ladestation durchgeführt.

Modellierung und Parameterbestimmung für PID-Regler

Damit ein passender Regler mit dem Siso-Tool bestimmt werden kann, muss die Regelstrecke bekannt sein. Weil die verwendeten Motoren des GoPiGos nicht bekannt sind, werden Werte eines DC Motors der HSLU verwendet. Dabei handelt es sich um den A-max 32 Maxon Motor 236655. Die Differenzialgleichungen können somit so beschrieben werden:

$$u(t) = R \cdot i(t) + L \cdot i'(t) + K_\omega \cdot \omega(t)$$

$$i'(t) = \frac{1}{L} [u(t) - R \cdot i(t) - K_\omega \cdot \omega(t)]$$

$$J \cdot \omega'(t) = K_M \cdot i(t) - M_l - \alpha \cdot \omega(t)$$

$$\omega'(t) = \frac{1}{J} [K_M \cdot i(t) - M_l - \alpha \cdot \omega(t)]$$

Aus dem Datenblatt können folgende Werte entnommen werden:

$$R = 7.03 \Omega \quad L = 0.00104 \text{ H} \quad K_\omega = 250 (\text{minV})^{-1} \quad J = 0.000004 \text{ kg}^2\text{m}$$

$$M_l = 0 \quad \alpha = 0$$

Das Trägheitsmoment wird dabei mit $0.000004 \text{ kg}^2 \text{ m}$ angenommen. Das Ersatzschaltbild kann folgendermassen aussehen:

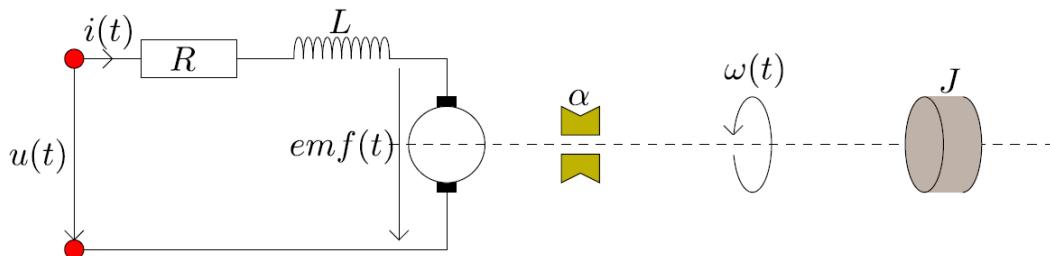


Abbildung 2: Ersatzschaltung DC-Motor

Im Simulink wurde für die Bestimmung von T_1 die Sprungantwort simuliert. Der Wirkungsplan sieht folgendermassen aus:

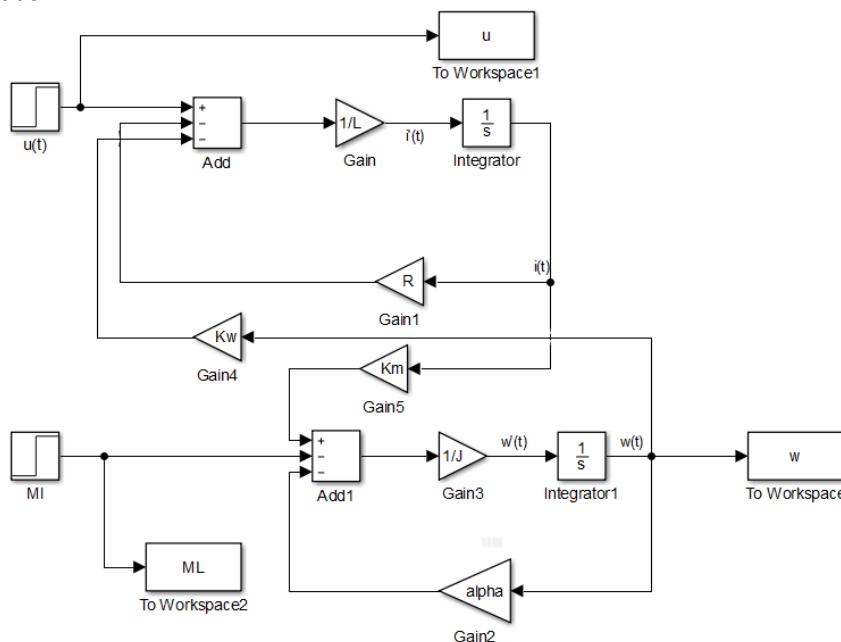


Abbildung 3: Wirkungsplan in Simulink des DC-Motors

Die erhaltene Sprungantwort wird in der nachfolgenden Abbildung dargestellt.

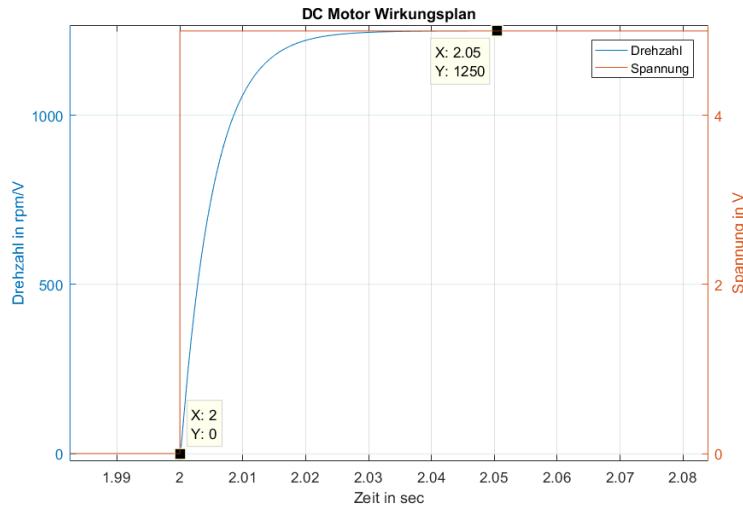


Abbildung 4: Sprungantwort des DC-Motors

Um T_1 zu bestimmen wird an der erhaltenen Sprungantwort die Zeitprozentkennwertmethode angewendet. Für diese benötigt man die Zeiten t_{10} , t_{50} und t_{90} .

Um t_{10} zu erhalten wird der Punkt bei 10% der Gesamtdrehzahl benötigt und der Zeitwert abgelesen.

$\omega_{10} = 1250 \cdot 0.1 = 125$, der abgelesene Wert für t_{10} ist somit 0.001 Sekunden.

Für t_{50} ergeben sich 0.004 Sekunden und für t_{90} ist somit 0.012 Sekunden.

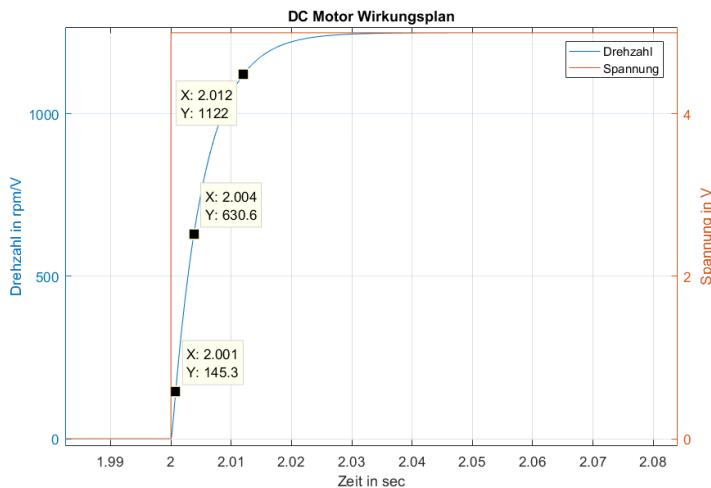


Abbildung 5: Sprungantwort des DC-Motors mit Wertangaben für die Parameterbestimmung

T_1 kann mittels dieser Formel berechnet werden.

$$T_1 = \frac{1}{3} \left[\frac{t_{10}}{\tau_{10}} + \frac{t_{50}}{\tau_{50}} + \frac{t_{90}}{\tau_{90}} \right] = 6.82ms \approx 10ms$$

Die Werte für τ können aus der Tabelle 9.3-3 auf Seite 398 des Buches *Taschenbuch der Regelungstechnik* von Lutz und Wendt entnommen werden. Da der verwendete Motor auf dem GoPiGo nicht der Qualität eines Maxon Motors entspricht, wird T_1 um einen Faktor 10 erhöht. K_s wird durch den Faktor des Sprungs bestimmt.

$$K_s = \frac{x(t \rightarrow \infty)}{x(0)} = \frac{1250 \text{ rpm}}{5V \cdot 60sec} = 4.166 (Vs)^{-1}$$

Somit kann die Regelstrecke des Motors so beschrieben werden:

$$G(s) = \frac{K_s}{1 + T_1 s} = \frac{4.166}{1 + 0.1s}$$

Um die Stabilität des Regelkreises zu überprüfen, wird das Bode-Diagramm des offenen Regelkreises gezeichnet. Als Übertragungsfunktion wird $G(s)$ verwendet, wobei $T_t = 0.5s$ ist.

$$G(s) = \frac{K_s}{1 + T_1 s} e^{-sT_t}$$

Für die Regelstrecke ergibt sich das folgende Bode-Diagramm:

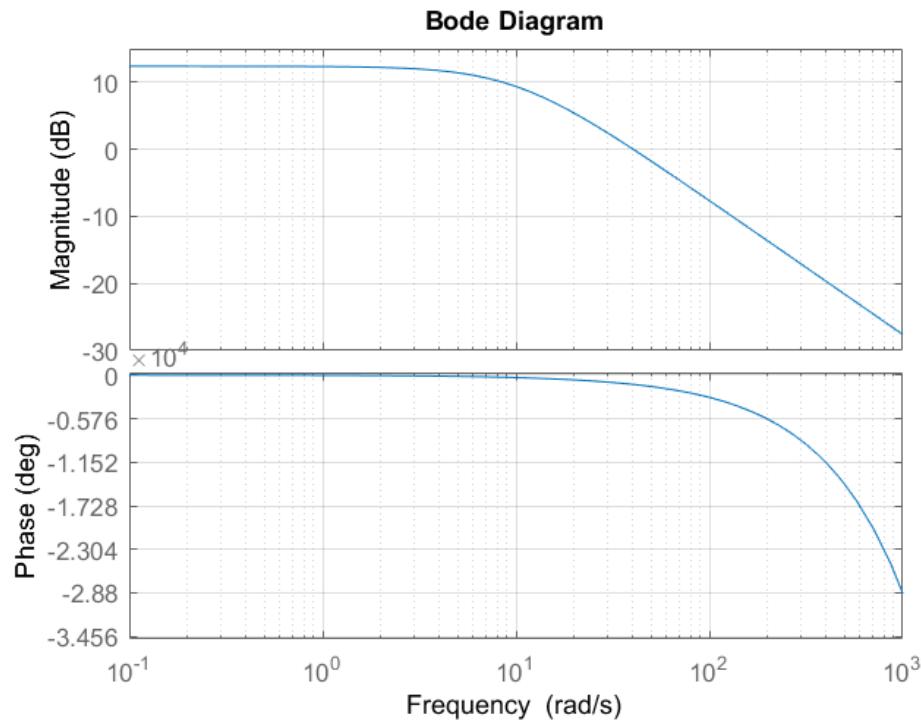


Abbildung 6: Bode-Diagramm des offenen Regelkreises

Es ist gut ersichtlich, dass der Phasengang sehr schnell in die Tiefe stürzt. Durch diesen Sturz ist die Regelstrecke hochgradig instabil. Der Grund für diesen Sturz ist die grosse Totzeit. Mittels Siso-Tool wird nun versucht einen passenden Regler zu entwerfen. Dabei geht man im Siso-Tool auf *Tuning Methodes* und danach auf *PID Tuning*.

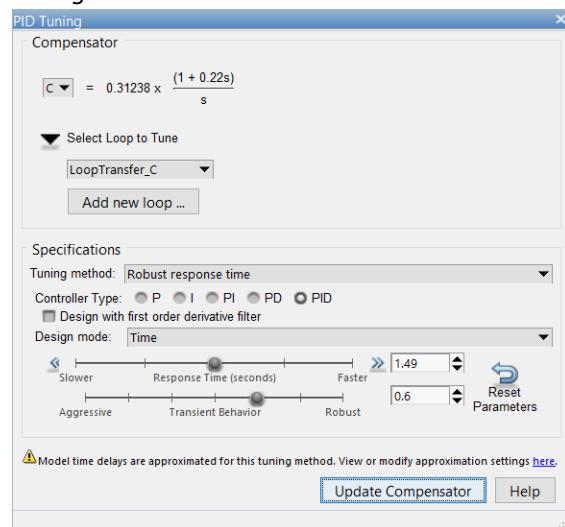


Abbildung 7: PID Tuning Fenster im Siso-Tool

Mit dem Betätigen der *Update Compensator* Taste entwirft das Siso-Tool automatisch einen passenden Regler. Das Bode-Diagramm, das Pol-Nullstellendiagramm und die Sprungantwort des Regelkreises werden zudem automatisch gezeichnet.

Es wurde nun ein Regler entworfen, der die Regelstrecke stabil bekommt und eine Phasenreserve von 60° gewährleistet. Die Anregelzeit ist trotz erreichter Stabilität sehr gross.

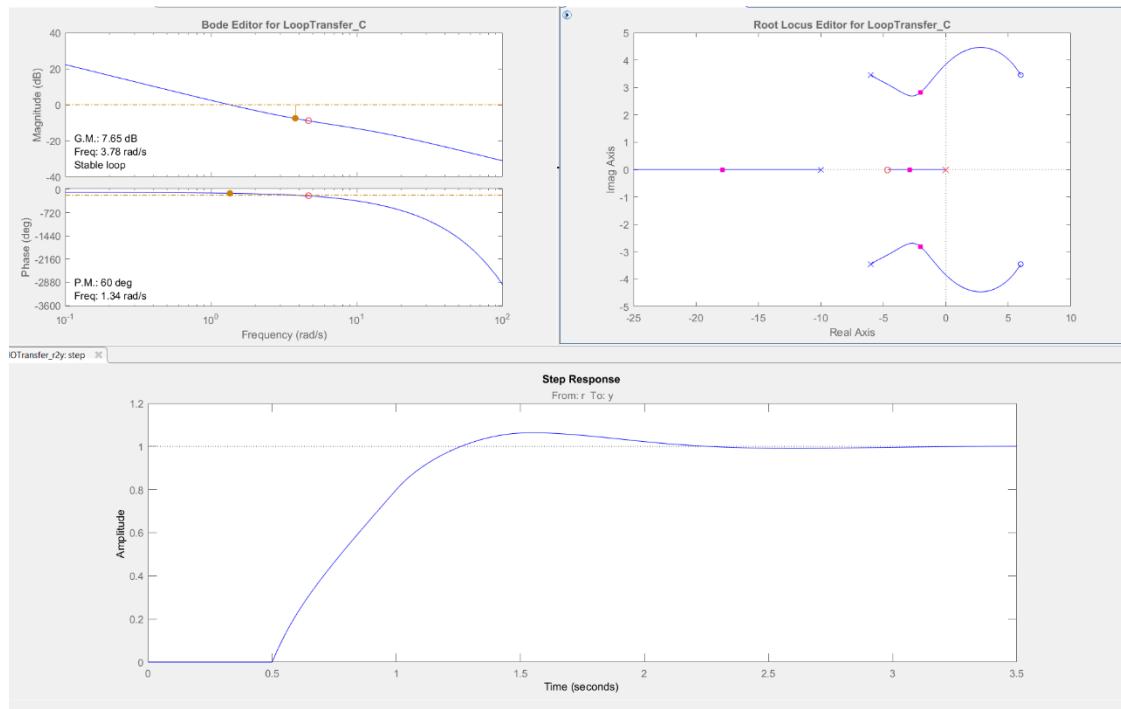


Abbildung 8: Bode-Diagramm, Pol-Nullstellendiagramm und Sprungantwort des offenen Regelkreis

Die erhaltenen Werte für den PID-Regler sehen wie folgt aus:

$$K_p * \left(1 + \frac{1}{T_i} * \frac{1}{s} \right)$$

with $K_p = 0.0672$, $T_i = 0.215$

Somit handelt es sich nicht um einen PID-Regler, sondern nur um einen PI-Regler. Der D-Anteil wird mit grosser Wahrscheinlichkeit nicht dazugenommen, da dieser erst bei schnellen Änderungen zum Tragen kommt. Dieser Regler wurde auf den GoPiGo implementiert.

Code der Funktion PID - ist für die Bestimmung des Motorwerts zuständig

```

import time
import numpy as np

Kp = 0.0672
Ki = 0.0672
Kd = 0
Ti = 0.213
Td = 1
Kp_Center = 5000

detectLeft = None
detectRight = None

time_old = int(round(time.time() * 1000))
iValue = 0
lastError= 0
time.sleep(0.5)
dValue, error = 0,0
speed = np.zeros((1,2))

def PID(angle, beta, d):
    global time_old, iValue, lastError
    time_new = int(round(time.time() * 1000))
    T = time_new - time_old

    error = 90-angle

    #P-Anteil
    pValue = Kp * error
    #I-Anteil (Trapezregel)
    iValue += Ki/Ti * (error + lastError)/2 * T
    #D-Anteil (Kd ist 0 somit wird nicht gewichtet)
    dValue = Kd * Td * (error - lastError) / T

    #Reset I-Value
    if angle > 100:
        iValue = 100

    #set new values
    lastAngle = angle
    time_old = time_new

    #calculation for output (es wird durch 180 damit value nicht zu gross)
    value = (pValue + iValue + dValue)/180

    #Mittellinie Anpassungen
    if angle > 0 and abs(angle)< 85:      #Motoren sollen rechts drehen
        detectLeft = None
        detectRight = True
    elif angle < 0 and abs(angle)< 85:      #drehe links
        detectLeft = True
        detectRight = None
    else:
        detectLeft = None
        detectRight = None

    error = np.median(beta*np.pi/180)      #abs(0-beta) = immer beta
    d = np.median(d)

    if d >= 0 and detectLeft == True:      #Mittellinie rechts und Kurve links
        value += Kp_Center * error
    elif d < 0 and detectRight == True:     #Mittellinie links und Kurve rechts
        value += Kp_Center * error
    elif d >= 0 and detectRight == True:     #Mittellinie rechts und Kurve rechts
        value -= Kp_Center * error

```

```
elif d < 0 and detectLeft == True:      #Mittellinie links und Kurve links
    value -= Kp_Center * error
elif detectLeft == None and detectRight == None and d < 0:  #Mittellinie ist rechts
    value += Kp_Center * error
elif detectLeft == None and detectRight == None and d >= 0: #Mittellinie ist links
    value += Kp_Center * error

#Saturation
if value > 40:
    value = 40
elif value < 0:
    value = 0

return value
```

Einstellung für das automatische Ausführen des Programms

Damit das Programm automatisch bei jedem Neustart ausgeführt wird, müssen gewisse Einstellungen vorgenommen werden.

Als erstens muss dafür gesorgt werden, dass der Befehl für den Aufruf der Python-Funktion erfolgt. Für diesen Zweck wird in der Konsole `sudo nano /etc/profile` eingegeben.

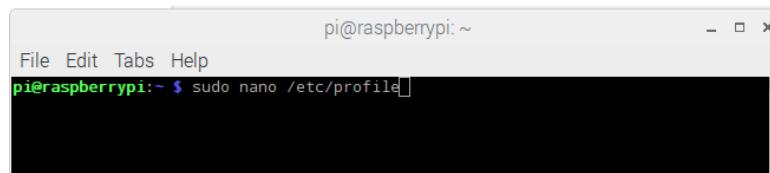


Abbildung 9: Befehl für das automatische Ausführen

Anschliessend muss im geöffnetem File ganz unten der Aufruf der Funktion eingegeben werden. Dafür muss das Verzeichnis bekannt sein, wo die Datei gespeichert ist. In diesem Fall ist die auszuführende Datei unter `/home/pi/FinalTest` zu finden.

Der Befehl für das Ausführen lautet:

`sudo python /home/pi/FinalTest/FinalTest.py`

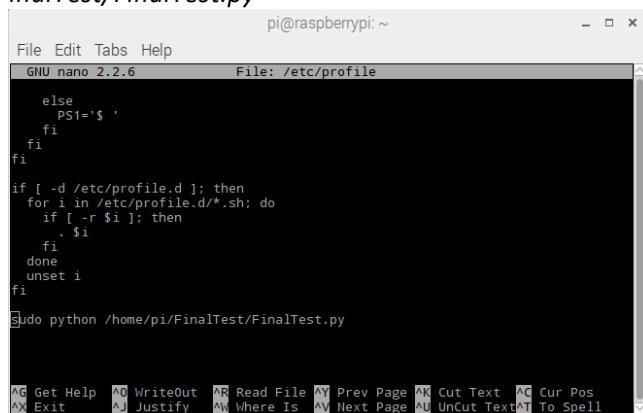


Abbildung 10: Screenshot der Eingabe

Nachher kann mit `Ctrl+X`, `y` und `Enter` diese Anpassung gespeichert werden. Das Programm wird nur ausgeführt, wenn das Hochfahren über die Konsole erfolgt. Die Einstellung dafür wird in den Konfigurationen vorgenommen. Dafür wird in der Konsole `sudo raspi-config` eingegeben. Anschliessend muss unter `3 Boot Options → B1 Desktop / CLI → B2 Console Autologin` ausgewählt werden.

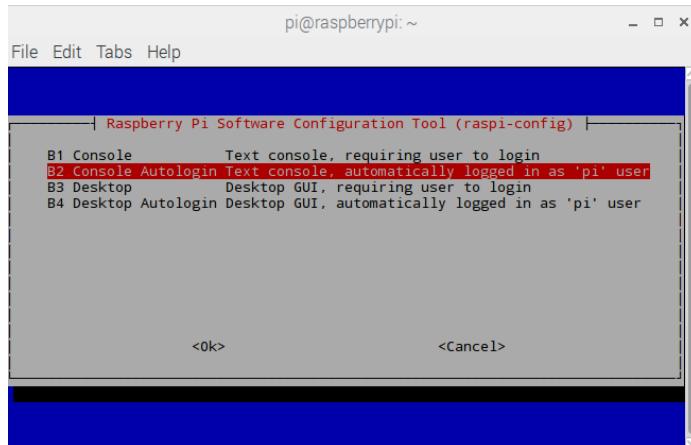


Abbildung 11: Einstellungen für das Hochfahren

Beim nächsten Neustart wird nicht mehr via Desktop hochgefahren, sondern via Konsole. Will man wieder, dass automatisch beim Hochfahren das Desktop angezeigt wird, muss unter `3 Boot Options → B1 Desktop / CLI → 4 Desktop Autologin` ausgewählt werden.

C. Anhang - Testberichte

Testbericht 1 – Gerade Linie befahren (SW8)

Für den ersten Test wurde die Datei `video_firstTest.py` verwendet. Zu Beginn muss man sich mit Putty einloggen und den Pfad zum Verzeichnis erstellen. Das passiert mit folgendem Befehl:
`cd /home/pi/Desktop/BDA/Video`

Mit folgendem Befehl wird das Programm gestartet:

`sudo python video_firstTest.py`

Mit Ctrl+C kann das Programm abgebrochen werden.

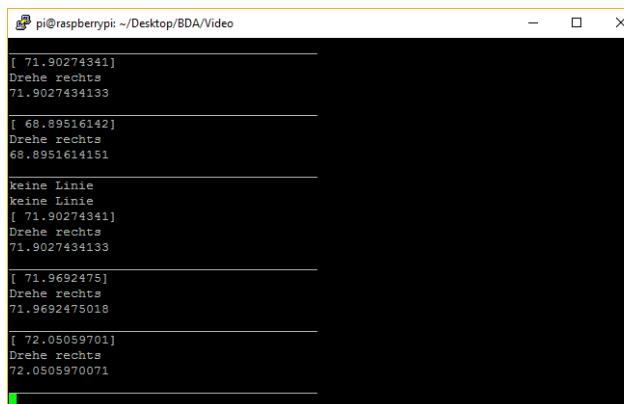
Die Teststrecke sieht wie folgt aus:



Abbildung 12: Teststrecke 1 für das gerade Fahren

Ergebnis:

In der Konsole ist ersichtlich, dass der Roboter einen nicht ausreichenden Winkel zur Linie hat. Darum wird eine Korrektur vorgenommen. Ebenfalls ist ersichtlich, dass ab und zu keine Linie detektiert wird. Dann dreht sich der Roboter ein Stück nach rechts um sie zu suchen.



```
pi@raspberrypi: ~/Desktop/BDA/Video
[ 71.90274341] Drehe rechts
71.9027434133
[ 68.89516142] Drehe rechts
68.8951614151
keine Linie
keine Linie
[ 71.90274341] Drehe rechts
71.9027434133
[ 71.9692475] Drehe rechts
71.9692475018
[ 72.05059701] Drehe rechts
72.0505970071
```

Abbildung 13: Konsolenausgabe des ersten Testes

Der Roboter reagiert somit auf eine Linie und kann dieser auch folgen. Das Fahren über den Streifen ist mit der Lenkrolle noch schwierig. Zudem ist der Untergrund auch eher rutschig. Das Fahren über Streifen erweist sich als schwierig. Aus diesem Grund wird empfohlen Klebestreifen zu verwenden.

Anpassungen bis zum nächsten Test:

- Evtl. grössere Perspektive erstellen
- Winkelgrenzwerte anpassen
- Motorenbefehle anpassen
- Erarbeiten wie Ende der Linie detektieren
- Spannungsüberwachung

Testbericht 2 – Kurve befahren (SW10)

Für den zweiten Test wurde die Datei `video_secondTest.py` verwendet. Zu Beginn muss man sich mit Putty einloggen und den Pfad zum Verzeichnis erstellen. Das passiert mit folgendem Befehl:

```
cd /home/pi/Desktop/BDA/Video/SW10
```

Mit folgendem Befehl wird das Programm gestartet:

```
sudo python video_secondTest.py
```

Mit Ctrl+C kann das Programm abgebrochen werden.

Die Teststrecke sieht wie folgt aus:

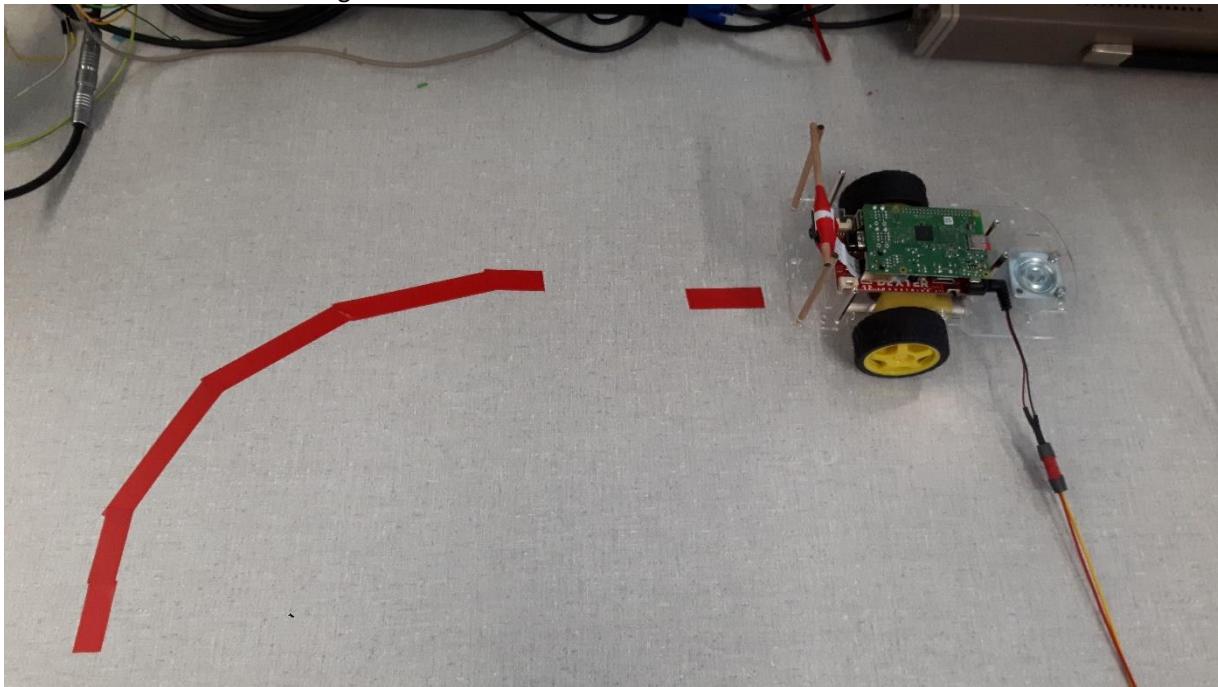


Abbildung 14: Teststrecke 2 für Kurvenbefahrung

Ergebnis:

Auf der geraden Strecke fuhr das GoPiGo gut. Auch das Anfahren der Kurve hat gut geklappt. In der Mitte der Kurve übersteuerte der Roboter dann deutlich und fand den Weg nicht mehr zurück. Auch bei einem zweiten Versuch konnte die Kurve nicht befahren werden.

Als Grund für die Übersteuerung wurden die zu hohe Totzeit (in etwa 1.3s pro Durchlauf) in Kombination mit einer zu hohen Geschwindigkeit bestimmt.

Anpassungen bis zum nächsten Test:

- Totzeit verkleinern
- Zentriertes Fahren
- Geschwindigkeit senken

Testbericht 3 – Anhalten bei LED-Stripe (SW12)

Für den dritten Test wurde die Datei `video_thirdTest.py` verwendet. Bei diesem Test wurde direkt via den Desktop, einer Maus und einer kabellosen Tastatur auf das Raspberry Pi zugegriffen:

Das ausgeführte Programm befindet sich im Verzeichnis `/home/pi/Desktop/BDA/Video/SW12`

Nachdem das Programm geöffnet wurde, kann unter *Run --> Run Module* das Programm gestartet werden. Dabei wird automatisch eine Shell geöffnet. Die Konsolenausgabe wird dort dargestellt.

Mit `Ctrl+C` kann das Programm abgebrochen werden.

Die Teststrecke sieht wie folgt aus:

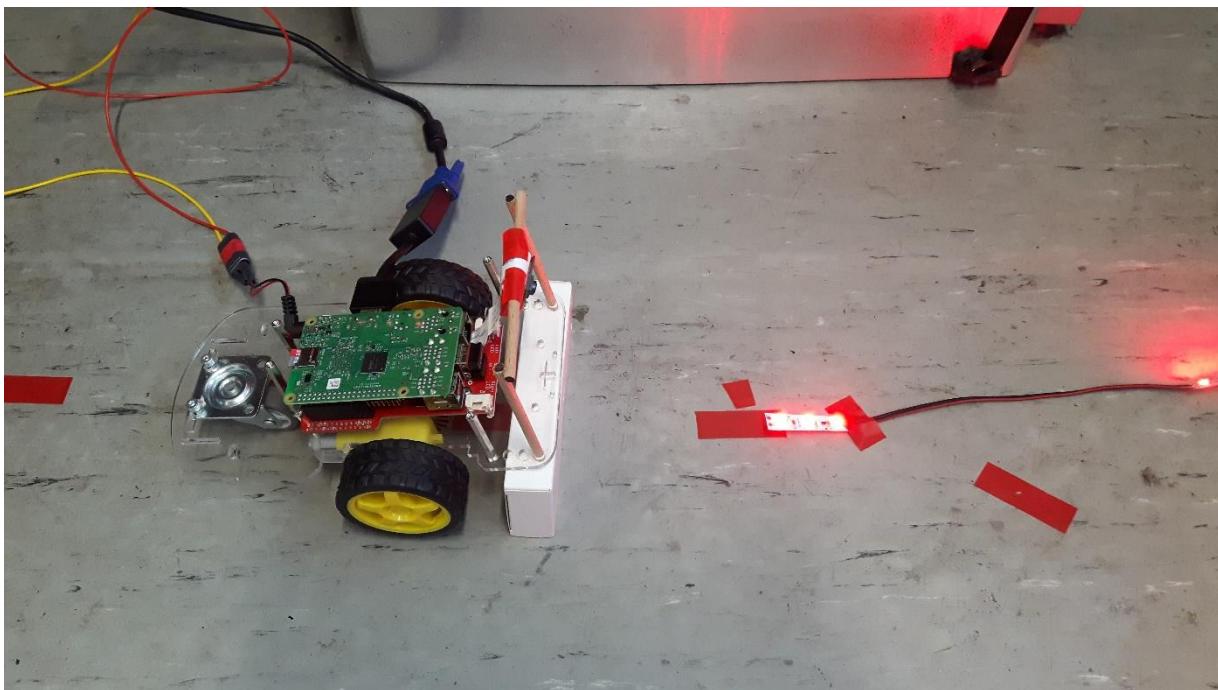


Abbildung 15: Test für das automatische Ausschalten

Ergebnis:

Zu Beginn war das LED-Stripe ausgeschaltet und die Räder bewegten sich. Als die LEDs eingeschaltet wurden detektierte dies der Roboter und schaltete die Motoren ab.

Je nach Lichteinfluss detektiert er auch das LED-Stripe, ohne dass dieser eingeschaltet ist. Durch die Lichteinstrahlung und der weißen Oberfläche erkennt die Kamera einen höheren Grauwert als der gesetzte Grenzwert.

Anpassungen bis zum nächsten Test:

- In Programm einbauen

Testbericht 4 – Fahrweite (SW13)

Für den vierten Test wurde die Datei `test_measureDistance.py` verwendet. Zu Beginn muss man sich mit Putty einloggen und den Pfad zum Verzeichnis erstellen. Das passiert mit folgendem Befehl:
`cd /home/pi/Desktop/BDA/Video/SW13`

Mit folgendem Befehl wird das Programm gestartet:
`sudo python test_measureDistance.py`

Mit Ctrl+C kann das Programm abgebrochen werden.

Die Teststrecke sieht wie folgt aus:

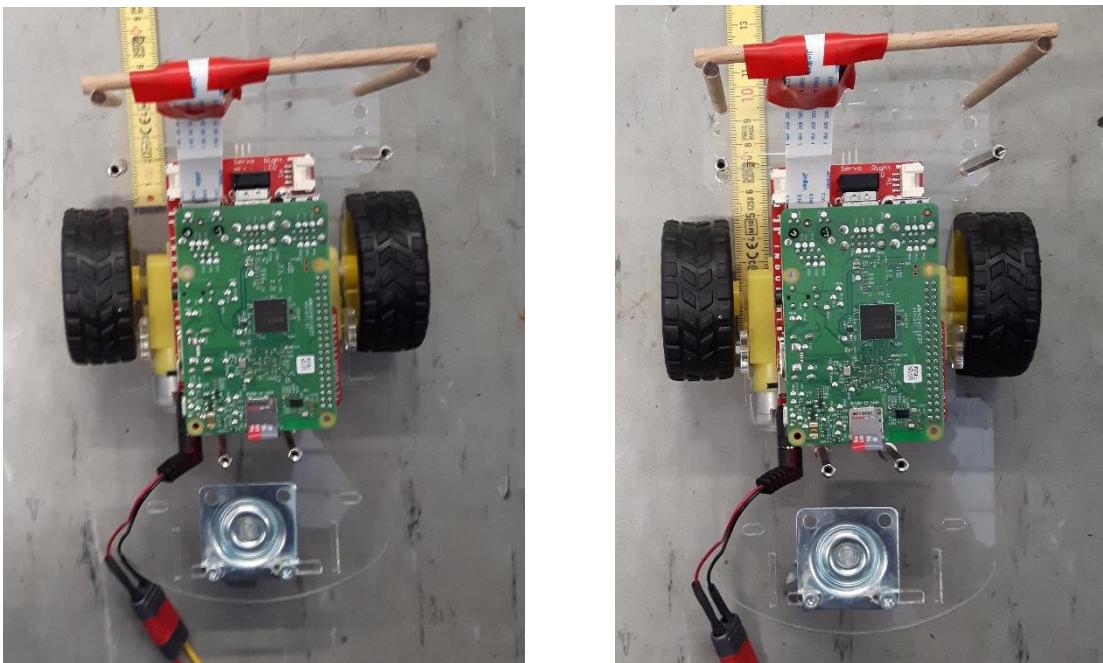


Abbildung 16: Teststrecke für die Fahrweite eines Durchgangs

Ergebnis:

Für die Messung wurde ein Meterstab verwendet. Der Start wurde bei 0cm gesetzt. Anschliessend wurde das Programm gestartet und der Roboter fuhr für 0.7s mit einem Wert von 40. Die zurückgelegte Distanz betrug nicht ganz 5cm.

Testbericht 5 – Kurvenbefahrung mit Anpassungen (SW14)

Für den fünften Test wurde die Datei `video_fifthTest.py` verwendet. Bei diesem Test wurde direkt via den Desktop, einer Maus und einer kabellosen Tastatur auf das Raspberry Pi zugegriffen:

Das ausgeführte Programm befindet sich im Verzeichnis `/home/pi/Desktop/BDA/Video/SW14/test`. Nachdem das Programm geöffnet wurde, kann unter *Run --> Run Module* das Programm gestartet werden. Dabei wird automatisch eine Shell geöffnet. Die Konsolenausgabe wird dort dargestellt.

Mit `Ctrl+C` kann das Programm abgebrochen werden.

Die Teststrecke sieht wie folgt aus:

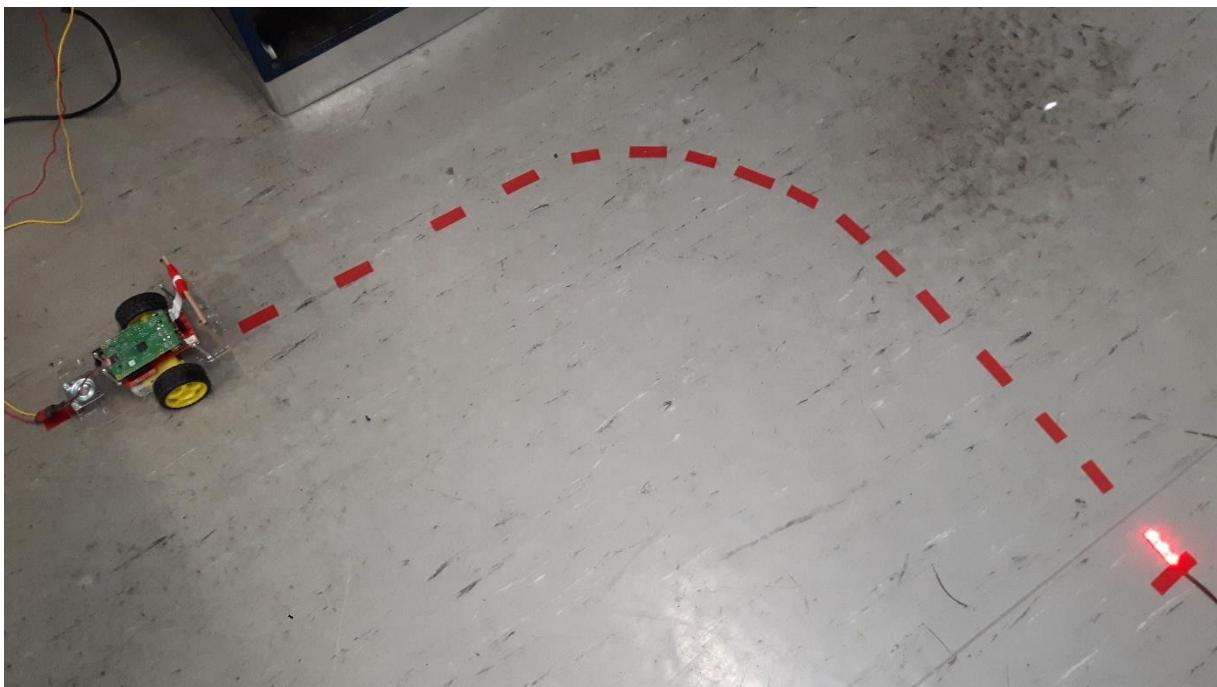


Abbildung 17: Teststrecke 3 Kurvenbefahrung

Ergebnis:

Auf dem ersten geraden Abschnitt folgt der Roboter der Linie gut. Auch die Geschwindigkeit ist passend. Sobald die Kurve erkannt und angefahren wird, nimmt die Geschwindigkeit ab. Die Kurve wird deutlich langsamer befahren. Das zentrierte Fahren der Kurve ist vor allem in der Kurve sehr schwierig und er kann das nicht gut durchführen. Die Stabilität des PI-Reglers bei einer guten Zentrierung ist aber gut ersichtlich. Die Autokorrektur kommt oft zum Tragen.

Das Ende kann der Roboter finden und schaltet an diesem auch ab. Dabei hält er schon bei der letzten Wegmarkierung an.

Ein möglicher Grund für die nicht gute Zentrierung könnte darin liegen, dass der P-Regler noch nicht passt. Dabei kann das Problem auftreten, dass die Geschwindigkeit bereits an den oberen und unteren Grenzen anstoßen und somit es nur schwer einstellbar ist. Der Wertebereich für die Geschwindigkeitseinstellung beträgt nur 25 Integer-Werte.

Anpassungen bis zum nächsten Test:

- Das zentrierte Fahren in Kurven verbessern

Abschlusstest – Befahrung Teststrecke (SW15)

Für den letzten Test wurde eine Teststrecke erstellt. Das Problem des zentrierten Fahrens aus SW14 konnte durch eine Anpassung behoben werden. Da der Roboter immer in der Saturation läuft, kann keine ausreichende Zentrierung vorgenommen werden. Darum wurde das Programm so angepasst, dass sobald ein $d > 30$ oder ein $d < -30$ vorhanden ist, das eine Rad schneller dreht als das andere was eine Korrektur zur Folge hat.

Das verwendete Programm befindet sich auf dem Pi im Verzeichnis
`/home/pi/Desktop/BDA/Video/SW15/FinalTest`

Das auszuführende Programm trägt die Bezeichnung `FinalTest.py`.
Die Teststrecke ist in der Rechten Abbildung ersichtlich.

Ergebnis:

Die vorhandenen Reflexionen konnten während des Tests nicht ignoriert werden und wurden als Lichtquelle detektiert. Aus diesem Grund hielt der Roboter an. Somit funktioniert der aktuelle Ansatz noch zu wenig zuverlässig.

Nach der Entfernung der Reflexionen konnte der Roboter die Fahrbahn gut folgen und war auch nicht allzu weit weg von der Mittellinie. Ab und zu hat der Roboter keine Wegmarkierung mehr gefunden. Als Resultat kam die automatische Rückfindung in Spiel. Dank dieser fand der Roboter wieder die Mittellinie.

Am Ende der Strecke hielt er vor den LED an.

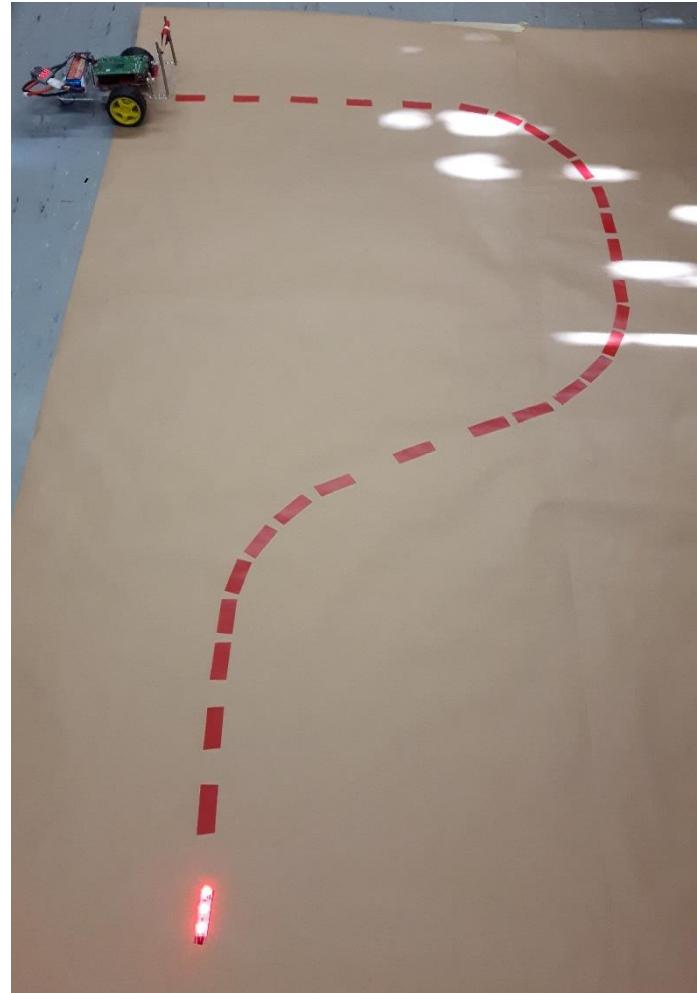


Abbildung 18: Teststrecke für Abschlusspräsentation

Fazit:

Der Roboter kann selbstständig einer Linie folgen und am markierten Ende anhalten. Durch die verschiedenen Anpassungen kann der Roboter auch selber nach der Strecke suchen falls diese nicht mehr detektiert wird.

Was nicht gut klappt ist die Resistenz gegenüber Reflexionen. Hier sollte noch mehr Zeit investiert werden um eine zuverlässige Lösung zu finden.

Verbesserungsvorschläge und nächste Schritte:

- Resistenz gegen Lichteinflüsse verbessern
- Automatisches Starten des Programms sobald der Roboter eingeschaltet wird

Inhalt der CD

- Sitzungsprotokolle
- Dokumentation und Plakat
- Präsentation
- Bilder und Videos
- Diagramme und Zeichnungen
- Code
- Datenblätter
- Matlab und Simulink Daten

CD