Autonomes Fahren mit Lego Mindstorms auf Untergrund durch Erkennen von Kanten

Benjamin Mydla, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Autonome System treten in Industrie und Alltag in vielen verschiedenen Formen in Erscheinung. Oft greifen sie dabei mit einer sehr geringen Reaktionszeit in Abläufe ein, die ein Mensch nicht erreichen kann. Diese Arbeit präsentiert die Entwicklung einer linearen Steuerung eines selbstständig fahrenden Roboters. Dieser vermeidet es, eine befahrbare Oberfläche zu verlassen. Die Umsetzung erfolgt mit MATLAB auf einer Lego-Mindstorms-NXT-Hardware-Plattform unter zu Hilfenahme der RWTH - Mindstorms NXT Toolbox. Grundsätzliche Überlegungen was Geometrie des Aufbaus und Software-Umsetzung betrifft, werden erörtert und eine Lösung wird präsentiert.**

***Schlagwörter*—autonomes Fahren, Kantenerkennung ,Lego Mindstorms, lineare Steuerung**

I. Einleitung

Im Zuge steigender Nachfrage elektrischer Mobilität, dem Fortschreiten der Digitalisierung, sowie der zunehmenden Miniaturisierung - und der damit verbundenen Verbesserung der Rechenleistung - der verwendeten Hardware werden immer häufiger autonom arbeitende Systeme möglich und gewünscht. So bremsen Assistenzsysteme eigenständig, sollte ein Hindernis den Weg versperren, halten die Spur auf Fahrbahnen oder informieren über geeignete Parklücken [1]. Straßenbahnen und Züge navigieren vollautomatisch und fahrerlos [2] in fest geplanten Routen bzw. Schienensystemen in einer veränderlichen Umwelt. Selbst Verkehrsflugzeuge [3] sollen autonom ihren Weg durch den Luftraum finden. Für die Umsetzung dieser Systeme existiert eine große Bandbreite an Sensoren, sowie eine Fülle an programmiertechnischen Lösungen. So ist die Arbeit, die im Rahmen des Projektseminars Elektrotechnik/Informationstechnik (LEGO Mindstorms) im Wintersemester 2018/2019 in Zusammenarbeit mit Jan Adamczyk entstand, als ein erster Versuch zu verstehen, sich diesem umfangreichen, wie weitreichenden Thema zu widmen. Als Zielstellung gilt es ein autonom fahrendes Vehikel zu konstruieren, welches sich auf einer Unterlage bewegt und es vermeidet über die Unterlage hinauszufahren. Explizit soll der Automat auf einem Tisch fahren und nicht herunter fallen. Als Hardware dient ein LEGO-Mindstorms-Education-Paket [4], welches softwareseitig mit Mathworks MATLAB [5] in Kombination mit der Mindstorms NXT Toolbox for MATLAB der RWTH Aachen [6] programmiert wird.

II. Vorbetrachtungen

Ein Mindstorms-Projekt besteht aus zwei essentiellen Teilen, der Hardware, die als Plattform dient, und der Software, die auf dieser Plattform läuft und sie steuert.

DOI: 10.24352/UB.OVGU-2019-049

A. Die Hardware-Grundlagen

Als Grundlage dient das Lego-Mindstorms-Education-Paket, bestehend aus Lego-Technik-Bausteinen, drei Servomotoren mit eingebauten Rotations-, Ultraschall-, Licht- und Tastsensoren, Anschlusskabeln und dem NXT-Baustein. Der Baustein trägt den 32-Bit-Mikroprozessor, einen Lautsprecher, sowie die Anschlüsse für die Sensoren, Motoren und den PC. [7]

Zur Realisierung des Gefährts wird neben den Lego-Technik-Bauteilen, die das Fahrgestell und Fahrwerk bilden, dem NXT-Baustein der als Steuereinheit fungiert, auf Sensoren zurückgegriffen, die die äußeren Bedingungen detektieren.

B. Die Software-Grundlagen

Ursprünglich wird mit der von Lego bereitgestellten Programmierumgebung NXT-G grafikbasiert mit fertigen Modulen gearbeitet und die Programme werden auf die Steuerungseinheit des NXT-Bausteins geladen. Mittlerweile existieren alternative Möglichkeiten in anderen Sprachen wie, beispielsweise Java, C, LUA oder Assembler zu arbeiten [8].

In der Projektarbeit wird eine weitere Methode, den NXT anzusteuern, genutzt. Hierbei fungiert MATLAB als Fernbedienung des NXT über USB. So können Daten der an den NXT angeschlossenen Sensoren und Motoren von MATLAB empfangen und verarbeitet, sowie auszuführende Befehle an den NXT gesendet werden [6].

Für die Umsetzung des Projekts - Fahren auf einer Unterlage und Vermeidung von Kanten (Absturz des Roboters über die Kante) - sind also zwei Dinge nötig:

1) Eine Plattform, die den NXT-Baustein (, die Motoren und Sensoren trägt. Die Sensoren erfassen die Umgebung die die Bewegung in zwei Raumrichtungen gewährleistet und die Umgebung erfassen kann.
2) Ein MATLAB-Programm, welches die Sensor-Daten vom NXT entgegennimmt, und entsprechend Befehle an den NXT sendet, der die Motoren steuert.

Die Möglichkeiten der nutzbaren Sensoren sind dank MATLAB vielfältig, und lassen auch ein einfaches Einbinden von USB-Webcams zu [9]. So ist es ein Ansatz, die Umgebung mittels Visueller Erfassung aufzunehmen und auszuwerten.

C. Visuelle Erfassung der Umgebung

Dafür werden eine oder mehrere Kameras ortsfest auf dem Fahrzeug (Roboter) installiert, deren Sichtfeld jeweils so groß

sein muss, den Raum vor dem Fahrzeug in einer sinnvollen Größe zu erfassen. Damit die Fahrt inklusive Lenkmanöver, die Sicherheit beim Wenden und das Vermeiden des Absturzes von der Unterlage möglich ist. Die Dimension des Fahrzeugs, die Anzahl der Räder, die gewählte Lenkgeometrie und Antriebsart, ob direkter Antrieb je Rad oder ein Getriebe, spielen beim Spur- bzw. Wendekreis [10] eine Rolle. Diese Eigenschaften formulieren die Anforderung an die optische Erfassung, und wo diese platziert wird. So kann es von Nöten sein, zwei statt einer Kamera zu verwenden. Die Software muss aus den aufgenommenen Bilddaten Informationen gewinnen, um abschätzen zu können, wo sich eine Kante befindet und diese durch Steuerung der Motoren vermeiden. Die Tiefeninformation der Umwelt lassen sich dabei durch eine Kopplung zweier Kameras erreichen. ”Die Bestimmung einer absoluten Tiefeninformation erfordert die genaue Kenntnis der Parameter aller Komponenten des Aufbaus und der Kameras. Das die Bestimmung dieser Parameter nicht trivial ist, kann für einfache Anwendungen der Wert der Disparität selbst als einziger Indikator für die Tiefeninformation verwendet werden. Hieraus ist eine relative Entfernung/Tiefeninformation ableitbar, welche oft für die Steuerung einer Anwendung ausreicht, ... - [11]. Die Bereitstellung der Daten zur Steuerung ist demnach mit erheblichem Aufwand verbunden. Die tiefenwertbasierte Detektierung von Höhenunterschieden führt zu der Überlegung, statt die gesamten Bilddaten zu nutzen, die Information direkt aus der Umgebung unter Zuhilfenahme von Lichtsensoren abzulesen. Damit ist die aufgenommene Datenmenge und der Rechenaufwand wesentlich geringer und nebst Lego-Mindstorms keine gesonderte Hardware nötig. Ein Linienfolgen [12] nutzt dieses Konzept, das adaptierbar scheint.

D. Der Linienfolger

Ist ein Roboter, der sich in zwei Raumrichtungen bewegen kann und eine Sensorik besitzt, die zwischen Untergrund und einer aufgebrachten Linie unterscheidet und dieser folgt.

Um charakteristischere Daten erfassen zu können wird hier ein heller, meist weißer Untergrund benutzt, auf den eine schwarze Linie aufgebracht ist. Diese zwei Bereiche haben unterschiedliche Reflexionsverhalten, wobei Weiß viel und Schwarz wenig Licht reflektiert. Der Lichtsensor [13], der über eine rote LED Licht auf die Unterlage emittiert und über eine Photodiode den reflektierten Anteil (Grauwert) absorbiert, kann dafür genutzt werden diese unterschiedlichen Bereiche zu erkennen. Der Grauwert wird vom Sensor in einem Bereich von 0 bis 1023 dargestellt [14], wobei 0 keiner und 1023 einer sehr großen absorbierten Lichtintensität entspricht. Mit diesen Daten lassen sich zwei Antriebsmotoren betreiben, damit der Roboter beim Fahren auf oder in der Nähe der Linie bleibt und somit dieser folgt. Die Menge der absorbierten Photonen ist von der Position des Sensors in Bezug zur Linie abhängig. Mit diesen Informationen lässt sich eine Steuerung der Motoren zum Fahren realisieren.

Die Oberflächen einer Unterlage und deren Kante besitzen jeweils auch unterschiedliche Reflexionseigenschaften. Die Oberfläche selbst reflektiert viel, die Kante wenig Licht. Dadurch lässt sich das Prinzip des Linienfolgers an eine Kantenvermeidung anpassen.

III. HAUPTTEIL

Das grundsätzliche Konzept ist ein Roboter, der auf einem Untergrund, Beispielsweise auf einem Tisch, fährt und Kanten vermeidet, um die Oberfläche nicht zu verlassen. Er bleibt nicht stehen, sobald eine Kante erreicht wird, sondern korrigiert eigenständig die Fahrtrichtung um weiterzufahren.

A. Die Hardware

Als Antrieb dienen dabei zwei Lego-Mindstorms-NXT-Motoren, die jeweils direkt mit einem Rad verbunden sind. Die Stabilität gewährleistet ein drittes, drehbares Rad am Heck. Die schematische Darstellung des Roboters in Abbildung 1 zeigt die Räder als graue Kästen am blauen Chassis des Roboters. Um der Probleme des Traktionsverlustes eines der Räder in einer Kurvenfahrt Rechnung zu tragen, sind die zwei Lichtsensoren dabei wie in Abbildung 1 zu sehen, angebracht. Die Überlegung sieht wie folgt aus: Da sich der Drehmittelpunkt beim Differentialantrieb auf halber Strecke auf der Achse zwischen beiden Motoren befindet (Punkt M in Abbildung 1), gibt bei der gewählten Bauform das Heckrad den Radius des Spurkreises an. Der Wenderadius ist größer als der Spurradius [10] und wird als Mindestmaß verwandt. Die Sensoren müssen diesen Mindestabstand einhalten, so dass bei einer Wendung das Hinterrad nicht von der Unterlage abgleitet. Beide Sensoren sind mit einem Versatz nach außen, bezogen auf das jeweilige Antriebsrad, angebracht. Durch diesen Versatz wird eine Kante, auch wenn sich ihr unter einem spitzen Winkel genähert wird, erkannt. Was wiederum bei einem anschließenden Lenkmanöver den möglichen Verlust der Traktion des der Kante zugewandten Antriebsrad verhindert. Die Sensoren sind auf den Untergrund gerichtet und empfangen so das von der Unterlage reflektierte Licht. Der Abstand von Sensor zum Untergrund beträgt circa 10 Millimeter.

B. Die Software

Die Steuerung der Motoren beruht auf dem Helligkeitswert des reflektierten Lichts. Dabei ergeben sich die in Abbildung 2 dargestellten drei Abschnitte mit ihren zugeordneten Sensorwerten. Diese sind ein grobes Resultat verschiedener Testläufe eines Sensors. Das gewünschte Verhalten kann wie folgt beschrieben werden: Der Roboter soll ungestört vorwärts fahren solange beide Sensoren die Oberfläche detektieren. Er soll wenden, wenn ein Sensor eine Kante oder einen Abgrund misst und stehen bleiben, wenn beide gleichzeitig eine Kante oder einen Abgrund erfassen.

Der weite Bereich der Sensorwerte der einzelnen Abschnitte in Abbildung 2 resultiert aus unterschiedlichen Lichtverhältnissen während der Testphase und aus dem verschiedenen Reflexionsvermögen je nach Untergrund. Dies ist bei jedem Lauf zu berücksichtigen und erfordert notwendige Anpassungen, um für die Steuerung nutzbar zu sein.

Es ist deshalb, vor jedem Start die Fahrroutine auf die vorhandene Unterlage zu kalibrieren. Es werden dafür 50 bis 500 Einzelmessungen pro Sensor durchgeführt und die Werte gemittelt. So entsteht je ein Kalibrierungswert für den linken und rechten Sensor, der das Reflexionsverhalten der Unterlage

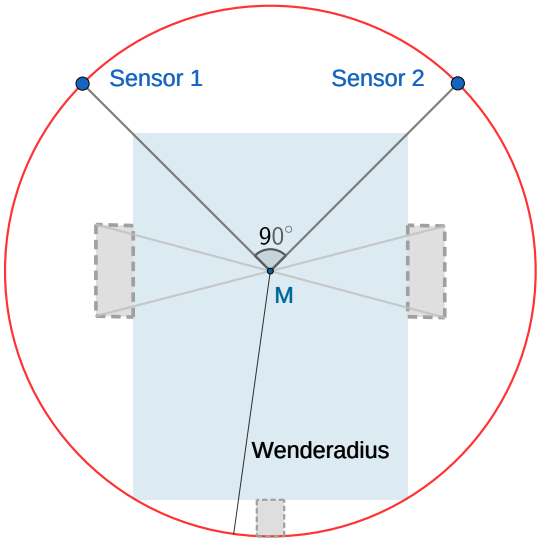

Abbildung 1. Positionierung der Sensoren aufgrund des Wenderadius

widerspiegelt. Die Unterlage muss homogen sein muss, damit das Reflexionsvermögen nicht zu stark variiert.

Für das Projekt ist dieser Umstand gegeben, so dass zur Ansteuerung der Motoren nicht die absolut gemessene Helligkeit, sondern eine berechnete Helligkeitsänderung, bezogen auf den Kalibrierungswert, genutzt wird. Siehe Gleichungen (1) und (2). Die Helligkeitsänderung ist ein Maß für die Entfernung des Sensors zum Untergrund (in Bezug auf die Ausgangslage). Mit diesen Daten ist es möglich, auftretende Höhendifferenzen durch die größeren Lichtunterschied, zu erkennen und somit Steuerungs-Signale für die Motoren zu generieren.

Die Geschwindigkeit mit der sich die Motoren drehen folgt dem Graphen aus Abbildung 3, der für den linken Motor aus der Gleichung (3) und für den rechten Motor aus Gleichung (4) folgt. Das Steuerungs-Prinzip ist dergestalt, dass der linke Sensor Einfluss auf das Drehverhalten des rechten Motors hat und der rechte Sensor auf das des Linken. Siehe dazu Abbildung 4, in der das Verhalten eines Motor-Sensor-Paares dargestellt ist.

Die Programm-Routine, wie in Abbildung 5 gezeigt, wiederholt sich nach dem Senden der Signale an die Motoren mit der Aufnahme der Sensorwerte, und läuft dann in einer Schleife, solange die berechnete Helligkeitsabweichung unter einem Maximum liegt. Das Abbruchkriterium wird erreicht, wenn einer der beiden Sensoren plötzlich einen zu starken Abfall der Helligkeit misst. Ursache dafür kann ein Hardware-Fehler, ein Störsignal (unvorhergesehene Lichtquellen) oder das plötzliche Geraten über eine Kante sein.

C. Gleichungen

$$\triangle Light_1 = |Cal_1 - Sensorwert_1| \quad (1)$$

$$\triangle Light_2 = |Cal_2 - Sensorwert_2| \quad (2)$$

$$MotorL = (\triangle Max - \triangle Light_2)/Cal_1) \times 100 + const \quad (3)$$

$$MotorR = (\triangle Max - \triangle Light_1)/Cal_2) \times 100 + const \quad (4)$$

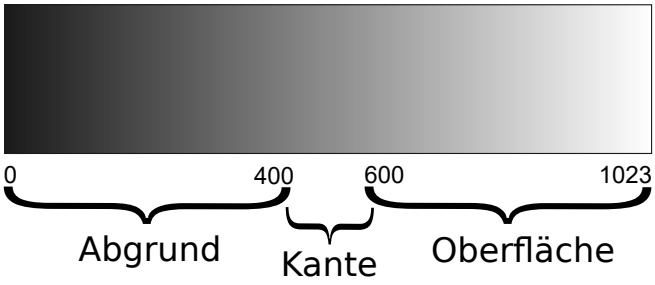

Abbildung 2. Darstellung der Sensorwerte zu den Messbereichen und der Grauwertskala

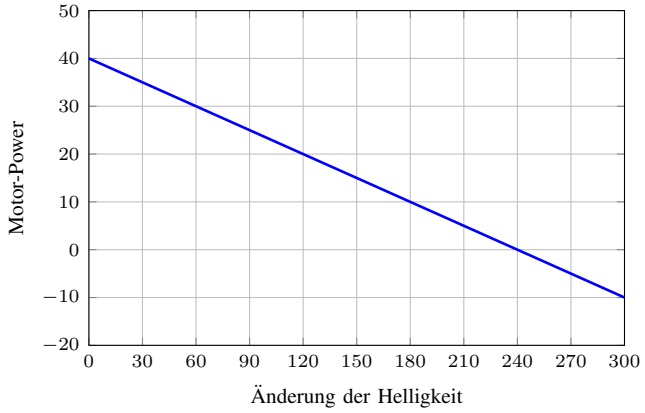

Abbildung 3. Helligkeitsänderung zu Motor-Power für jeweils Sensor-Motor-Paar: linker Sensor für rechten Motor und rechter Sensor für linken Motor. Vergleiche (3) und (4), wobei $\triangle Max = 60$, $Cal = 600$.

mit

$\triangle Max$ ... *Schwellwert*
$\triangle Light_1$ ... *Helligkeitsaenderung linker Sensor*
$\triangle Light_2$ ... *Helligkeitsaenderung rechter Sensor*
Cal_1 ... *Kalibrierungswert linker Sensor*
Cal_2 ... *Kalibrierungswert rechter Sensor*
const ... *Geschwindigkeitsregelung*

Der Schwellwert $\triangle Max$ ist ein Maß für die maximale Abweichung der Helligkeit vom Kalibrierungswert. Ist $\triangle Light$ größer als $\triangle Max$, so ist das Ergebnis negativ und der entsprechende Motor dreht Rückwärts. Ansonsten dreht der Motor vorwärts. Die Größe *const* ist für eine nachträgliche Geschwindigkeitsregelung vorhanden.

IV. Ergebnisdiskussion

Die Erkennung der Kanten einer farblich homogenen Oberfläche und die notwendige Steuerung der Motoren, um die Oberfläche nicht zu verlassen und weiterzufahren, wurde durch die implementierte Steuerung - bis auf einen Spezialfall - erreicht. Die genutzten Gleichungen (3) und (4), die die Motoren-Geschwindigkeit regeln, enthalten die nur empirisch bestimmten Größen $\triangle Max$ und *const*, die redundant sind. Bei der Erprobung der Fahrdynamik ließ sich mit Hilfe des konstanten Zusatzterms *const* die Fahrgeschwindigkeit

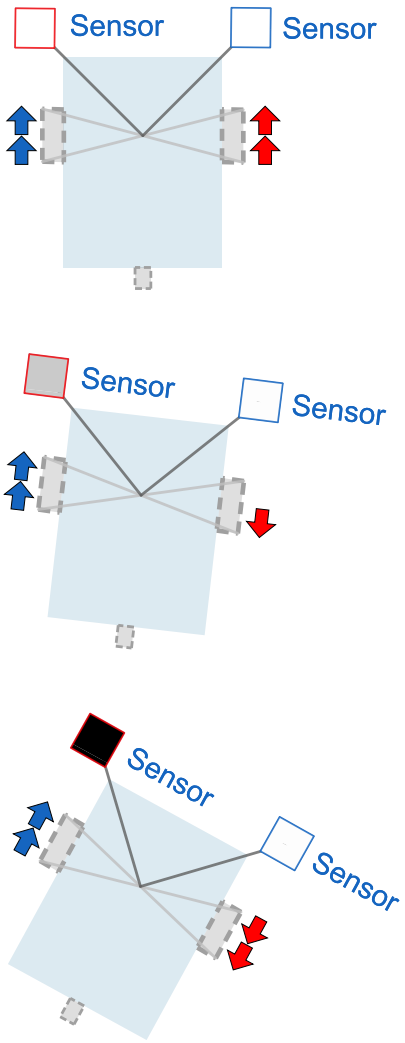

Abbildung 4. Beispiel für Motorenverhalten bezogen auf Sensorwerte. Weißer Sensor entspricht kleiner oder keiner Helligkeitsänderung, grauer Sensor ist Anfang einer Kante und schwarzer Sensor über einer Kante hinaus.

einfacher und direkter justieren, als mit $\triangle Max$. Im besonderen Fall war bei gleichzeitigem Auftreffen beider Sensoren auf eine Kante der Roboter in eine Schleife geraten. Beide Motoren bekamen dabei identische Anweisung und stoppten, fuhren rückwärts bis sie wieder auf der Unterlage waren und fuhren dann beide wieder vorwärts. Dieser Vorgang endete erst, wenn durch eine Störung beide Motoren stark unterschiedliche Werte bekamen. Im Normalfall sorgte die Steuerung dafür, dass der Roboter orthogonal zur Kante verblieb.

V. Zusammenfassung und Fazit

Die vorgestellte Steuerungs-Lösung bietet im Rahmen des Anforderungsprofils, auf einer farblich homogenen Oberfläche autonom zu fahren und diese nicht zu verlassen, eine mögliche Umsetzung. Sie ist übersichtlich in der Programmierung und enthält bei der Geschwindigkeitsregelung keine Bedingungen und/oder Verzweigungen, wodurch die Anfälligkeit für logische Fehler und die Laufzeit sinkt. Es sind jedoch die Implementierungsmöglichkeiten von Spezialfällen (Rückwärtsfahren, Fahren auf sich farblich ändernden Unterlagen) zu eruieren.

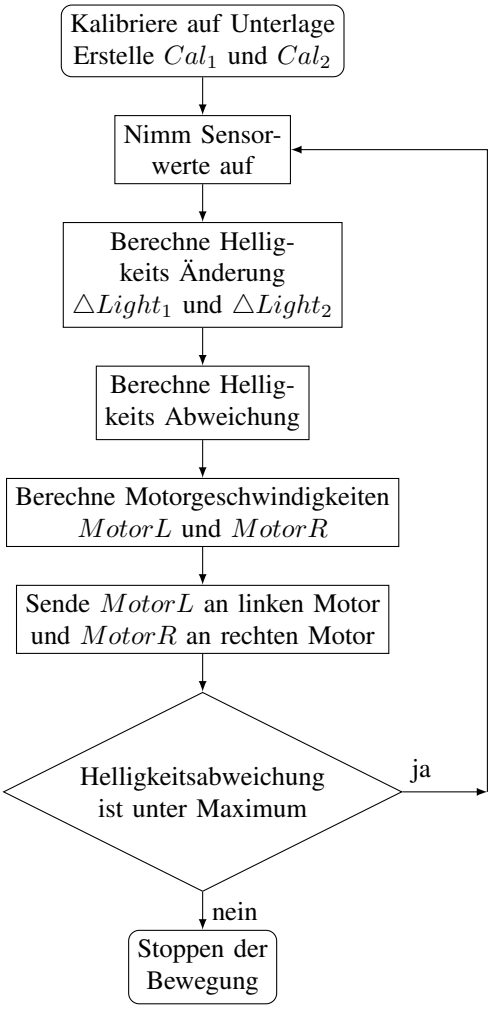

Abbildung 5. Programmablaufplan des Vorwärtsfahr-Algorithmus

Im Allgemeinen kann auf Oberflächen ein sich änderndes Reflexionsverhalten durch Lücken im Material oder durch aufgebrachte Farbe entstehen. Intensives, einfallendes Licht kann die Sensorik beeinflussen. Diese Fälle gilt es zu berücksichtigen.

Literaturverzeichnis

[1] ADAC: *ADAC Info - Fahrerassistenzsysteme*. https://www.adac.de/infotestrat/technik-und-zubehoer/fahrerassistenzsysteme/uebersicht/fahrerassistenzsysteme_uebersicht.aspx. Version: 2019. – [Online; Stand 15. März 2019]

[2] Wikipedia: *Automatic Train Operation — Wikipedia, Die freie Enzyklopädie*. https://de.wikipedia.org/w/index.php?title=Automatic_Train_Operation&oldid=183846734. Version: 2018. – [Online; Stand 15. März 2019]

[3] Finn Mayer-Kuckuk, Frankfurter R.: *Ohne Pilot über den Wolken*. https://www.fr.de/wirtschaft/ohne-pilot-ueber-wolken-10968097.html. Version: 2019. – [Online; Stand 15. März 2019]

[4] Lego: *LEGO® MINDSTORMS® Education*. https://education.lego.com/de-de/product/mindstorms-ev3. Version: 2019. – [Online; Stand 15. März 2019]

[5] Mathworks: *MATLAB*. https://de.mathworks.com/products/matlab.html?s_tid=hp_products_matlab. Version: 2019. – [Online; Stand 15. März 2019]

[6] Mathworks: *RWTH - Mindstorms NXT Toolbox*. https://de.mathworks.com/matlabcentral/fileexchange/18646-rwth-mindstorms-nxt-toolbox. Version: 2019. – [Online; Stand 15. März 2019]

[7] WIKIPEDIA: *Lego Mindstorms NXT — Wikipedia, Die freie Enzyklopädie*. https://de.wikipedia.org/w/index.php?title=Lego_Mindstorms_NXT&oldid=185430460. Version: 2019. – [Online; Stand 15. März 2019]

[8] WIKIPEDIA: *Lego Mindstorms NXT Programmierung— Wikipedia, Die freie Enzyklopädie*. https://de.wikipedia.org/w/index.php?title=Lego_Mindstorms_NXT&oldid=185430460#Programmierung. Version: 2019. – [Online; Stand 15. März 2019]

[9] MATHWORKS: *Webcam Support from MATLAB*. https://de.mathworks.com/hardware-support/matlab-webcam.html. Version: 2019. – [Online; Stand 15. März 2019]

[10] WIKIPEDIA: *Wendekreis (Fahrzeug) — Wikipedia, Die freie Enzyklopädie*. https://de.wikipedia.org/w/index.php?title=Wendekreis_(Fahrzeug)&oldid=185193522. Version: 2019. – [Online; Stand 17. März 2019]

[11] BRUENIG, Technischen Hochschule N.: *Bearbeitung und Gewinnung von Tiefeninformation durch die Kopplung zweier Kameras*. http://zuse1.efi.fh-nuernberg.de:8050/interaktion/index.php5?title=Bearbeitung_und_Gewinnung_von_Tiefeninformation_durch_die_Kopplung_zweier_Kameras&oldid=124. Version: 2019. – [Online; Stand 20. März 2019]

[12] WANG, Z.: *A MODEL OF LINE FOLLOWING ROBOT USING PID CONTROLLER: An Educational Platform Based on LEGO Mindstorms NXT Kit*. 2015

[13] WESTFALL, By S. ; MALOVICH, Dennis: *LEGO NXT: Features & Limitations*. Faculty of Science, University of Windsor, 2011. http://nxt.cs.uwindsor.ca/499football/features_limitations.pdf

[14] AZRAEL: *Unterlagen zum Praktikum Lego Mindstorms in der FEIT*. Fakultät für Elektrotechnik, Otto-vov-Guericke-Universität Magdeburg, 2012. https://elearning.ovgu.de/mod/resource/view.php?id=44972