FH Technikum Wien

PROJEKTBERICHT

im Studiengang BMR4-7

MULE – Weiterentwicklung der mobilen Plattform zu einem autonomen Transporter

Michael Kraupp (Projektleiter)
1050 Wien, Arbeitergasse 9/5
Lukas Beier
1110 Wien, Geiselbergstraße 60-64/2
Stefan Kautz
2103 Langenzersdorf, Dr. Ludwigstraße 17/7

Begutachter: DI Dr. Wilfried Kubinger
Wien, 16.02.2011

FACHHOCHSCHULE
TECHNIKUM WIEN

Eidesstattliche Erklärung

„Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.“

Ort, Datum	Michael Kraupp
	Lukas Beier
	Stefan Kautz

Kurzfassung

Die Vorgängergruppe dieses Projektes realisierte eine mobile Plattform. Diese war bereits mit Sicherheitseinrichtungen wie Ultraschallsensoren, Bumpern und einem Not-Aus ausgestattet. Sie konnte programmgesteuert fahren, Hindernisse erkennen und ihnen ausweichen. Eine gezielte Navigation war jedoch nicht möglich, da keine Sensoren hierfür implementiert waren. Außerdem waren die für den autonomen Fahrbetrieb erforderlichen Sicherheitsmaßnahmen noch nicht evaluiert.

Im Zuge dieses Projektes wurde die mobile Plattform zu einem autonomen Transporter erweitert. Hierzu wurden zwei Navigationsroutinen implementiert. Der autonome Transporter ist nach Abschluss des Projekts in der Lage sowohl einer Linie zu folgen, als auch vorgegebene Drehungen und Fahrmanöver mit Hilfe von odometrischen Messverfahren durchzuführen. Des Weiteren kann der Transporter zwei festgelegte Farben unterscheiden und so bestimmte Funktionen im laufenden Steuerprogramm aufrufen. Für die Realisierung dieser neuen Funktionen wurden Sensoren für die Linienerkennung, die Odometrie sowie für das Erkennen von Farben integriert. Zum Schutz von Personen und Gegenständen während des autonomen Fahrbetriebes, wurde eine Sicherheitsanalyse erstellt und aufgrund der Ergebnisse entsprechende Maßnahmen entwickelt und realisiert.

Der autonome Transporter ist nun in der Lage, vorgegebene Strecken zu fahren. Hierbei ist ein minimaler Kurvenradius einzuhalten. Mit der Farberkennung kann eine kontrollierte Drehung um 180° durchgeführt werden, sowie eine Halten von 18 Sekunden. Akustische und visuelle Warneinrichtungen, sowie ein Verbau der angetriebenen Räder wurden als zusätzliche Sicherheitseinrichtungen implementiert. Die Transportaufgabe wurde für den Transport von Getränken und Lebensmitteln ausgelegt.

Schlagwörter: MULE, Odometrie, Linienverfolgung, fahrerlose Transportsysteme

Inhaltsverzeichnis

1 Einleitung

In der Industrie werden mobile Roboter immer öfter für Transportaufgaben eingesetzt. Dies geschieht nicht nur aus Rationalisierungsgründen, sondern oft werden autonom fahrende Roboter auch zum Transport von schweren oder gesundheitsschädlichen Materialien eingesetzt. Aber auch im Dienstleistungsbereich gewannen autonom fahrende Serviceroboter in den letzten Jahren zunehmend an Bedeutung. Aus diesem Grund beschäftigte sich dieses Projekt mit der Entwicklung eines autonomen Transporters.
Dieser Bericht gliedert sich in einen theoretischen, einen praktischen und einen Ergebnis Teil. Im theoretischen Teil werden die Grundlagen der behandelten Themen erläutert. Der praktische Teil ist in mechanischen Aufbau, elektrischen Aufbau sowie die Einbindung in das Gesamtsystem untergliedert. Abschließend folgt die Dokumentation der Ergebnisse, die Zusammenfassung der Arbeit sowie ein Ausblick auf zukünftige Projekte.

1.1 Motivation

Das Projekt “MULE – Weiterentwicklung der mobilen Plattform zu einem autonomen Transporter” wurde an der Fachhochschule Technikum-Wien abgewickelt und fand im vierten Semester des Bachelor-Studiengangs Mechatronik/Robotik statt. Das Ziel war es, den praktischen Umgang mit Sensoren und Aktoren zu erlernen, um einen mobilen Roboter mittels Mikrocontroller navigieren lassen zu können. Dabei stand neben einem potentiellen industriellen Nutzen vor allem der Vorführeffekt im Vordergrund.
In vorangegangenen Projekten wurde bereits eine mobile Roboterplattform auf Basis eines elektrischen Rollstuhls erstellt. Diese wird von zwei Elektromotoren betrieben, welche von einem Mikrocontroller gesteuert werden. Des Weiteren wurden Ultraschallsensoren installiert, welche ein kollisionsfreies Fahren ermöglichen. Sollte es dennoch zu einer Kollision kommen, unterbrechen die ebenfalls schon installierten Bumper den Stromkreis des Roboters und er kommt zum Stillstand.

1.2 Aufgabenstellung

Die Gesamtaufgabe des Projektes war die Entwicklung eines autonomen Transporters, welcher in der Lage ist, die Besucher des Vitero-Labors der FH Technikum Wien als Kellner zu bedienen.
Um dieses Ziel zu erreichen, wurden zwei Navigationsroutinen implementiert und anschließend bewertet. Ebenfalls war es notwendig, einen mechanischen Aufbau zu konstruieren und zu realisieren, welcher die problemlose Entnahme von gefüllten Getränkebechern und Lebensmitteln ermöglicht.
Für die Betriebssicherheit des autonomen Fahrbetriebes wurde auch eine Sicherheitsanalyse durchgeführt, um Maßnahmen zu tätigen, welche die Gefährdung der

Besucher des Vitero-Labors auf ein Minimum reduziert. Weiters wurde der Betrieb durch optische und akustische Signale angezeigt.

1.3 Lösungsansatz

Eine der beiden realisierten Navigationsroutinen besteht in der Verfolgung einer Linie. Diese Linie wird entlang des gewünschten Pfades des Roboters auf den Boden geklebt. Der Roboter selbst wurde mit Infrarotsensoren ausgestattet. Mithilfe der unterschiedlichen Reflexionseigenschaften des Untergrundes bezüglich der schwarzen Linie, ist ein Erkennen der Linie möglich. Damit ist der Roboter durch geregeltes Aktivieren der jeweiligen Motoren für das linke bzw. rechte Antriebsrad fähig, der Linie zu folgen. Zusätzlich wurde ein Farbsensor implementiert, welcher zwei verschiedene Farben erkennen kann. Dadurch können bei laufender Linienverfolgung zwei zusätzliche Kommandos übergeben werden, auf die der Roboter reagieren kann.
Die zweite Navigationsroutine, wurde durch das Messen der Umdrehungen der Antriebsräder (Odometrie) realisiert. Hierbei wurde bei jedem der beiden differenziell gesteuerten Antriebsräder je ein Drehgeber angebracht, wodurch der zurückgelegte Weg, sowie die Geschwindigkeit des jeweiligen Motors vom Mikrocontroller der Robotersteuerung errechnet werden kann. Dies ermöglicht dem Roboter festgelegte Kurven und Strecken zu fahren.
Der mechanische Aufbau wurde für eine einfache Entnahme von Getränkebechern in einer Höhe, ausgehend von der Durchschnittsgröße eines Menschen, realisiert. Getränke können direkt von oben aus einer Halterung entnommen werden, wobei sich darunter ein frei zugängliches Fach für Lebensmittel befindet. Für die Sicherheit der Besucher wurden ebenfalls alle drehenden- und stromführenden Teile des Roboters entsprechend verbaut, bzw. abgesichert. Die optische Signalisierung des Betriebes wurde mit einer orange blinkenden Leuchte realisiert, welche sich am höchsten Punkt des Roboters befindet. Eine akustische Signalisierung des autonomen Fahrbetriebes wurde mit einem Piezo-Schallgeber realisiert.

2 Grundlagen der Indoor-Navigation

Die Navigation in geschlossenen Räumen ist eine der grundlegenden Technologien für fahrerlose Transportsysteme. Diese finden immer häufiger Verwendung in Kliniken, öffentlichen Einrichtungen und auch in der industriellen Fertigung. Je nach Einsatzort kommen unterschiedliche – bereits erprobte – Navigationsverfahren zum Einsatz.
Das Ziel all dieser Verfahren ist es die Position des Fahrzeugs festzustellen. Die einfachste und gleichzeitig fehleranfälligste Methode ist die Odometrie. Dabei werden die Umdrehungen der Räder messtechnisch ermittelt (Inkrementalgeber) und über mathematische Zusammenhänge der zurückgelegte Weg berechnet. Allerdings steigt mit dem zurückgelegten Weg auch die Messabweichung. Gründe dafür sind beispielsweise der

Schlupf der Räder, sowie die durch wechselnde Last hervorgerufene Änderung des Raddurchmessers. Ein weiterer Nachteil dieser Methode ist, dass es sich um eine relative Messung handelt. Das Fahrzeug muss also vor jeder neuen Messung genau an einem vorher definierten Startpunkt platziert werden. Trotz dieser Nachteile wird die Odometrie gerne verwendet, da die Kosten für die Sensoren sowie die Auswerteelektronik im Vergleich zu anderen Verfahren gering sind. Um den auftretenden Fehler so gering wie möglich zu halten, wird oft eine Kombination aus Odometrie und anderen Messverfahren (z.B. Magnetkompass, Zeitzähler) verwendet. (Ullrich, 2011)

Eine andere Methode um ein fahrerloses Transportfahrzeug navigieren zu lassen, ist die Verfolgung einer kontinuierlichen Leitlinie. Diese kann induktiv oder optisch ausgeführt sein und wird von entsprechenden Sensoren erfasst. Bei der induktiven Variante unterscheidet man außerdem auch unter einer aktiven und einer passiven Linie. Ein Nachteil dieser Naviationsroutine ist, dass für eine Änderung des Pfades u.U. bautechnische Maßnahmen erforderlich sind. Es ist darauf zu achten, dass die Leitlinie sich deutlich vom Untergrund abhebt und keine Unterbrechungen aufweist. In der industriellen Anwendung wird für das Erfassen der Linie zumeist ein Kamerasystem verwendet. Die Auswertung erfolgt mithilfe eines Kantenerkennungs-Algorithmus und kann durch weitere Bildbearbeitung auch stark verschmutzte oder beschädigte Linien korrekt erkennen. (Ullrich, 2011)

Da der im Projekt verwendete Roboter durch einen Mikrocontroller gesteuert wird, ist die Verwendung eines Kamerasystems aufgrund der erforderlichen hohen Rechenleistung nicht möglich. Aus diesem Grund wurde die Linienerkennung mit – in einer Reihe angeordneten - Infrarotdioden und Phototransistoren verwirklicht. Diese Methode wird in Kapitel 2.2 näher beschrieben. Ein weiteres Verfahren, welches im Vorfeld des Projekts diskutiert wurde, ist die Lasernavigation.

Hierbei werden im Raum angebrachte Reflektoren durch einen rotierenden Laserscanner abgetastet. Für eine korrekte Positionsbestimmung müssen je nach gewähltem Verfahren mindestens zwei oder drei Reflektoren für den Laserscanner sichtbar sein. Der Vorteil dieses Systems liegt darin, dass das Ändern der Route keine bautechnischen Veränderungen erforderlich macht, sondern der neue Pfad einfach softwaretechnisch geändert werden kann. (Ullrich, 2011)

Aufgrund des Bedarfes einer eigenen Steuerung, welche die erforderliche Rechenleistung aufweist, wurde dieses Verfahren nicht gewählt.

Eine andere Methode um in Bereichen zu navigieren, in welchen kein Kontakt zu GNSS Satelliten aufgebaut werden kann, ist iGPS. Dieses System wird von der Nikon Metrology GmbH entwickelt und vertrieben. Dieses kann zur Positionsbestimmung, aber auch für Vermessungsanwendungen im dreidimensionalen Raum verwendet werden. Es arbeitet mit mindestens 2 Infrarot-Lasersendern, welche ähnlich dem GPS, ihre Position sowie die aktuelle Uhrzeit versenden. Dieses Signal wird von Sensoren empfangen und an einen Empfänger weitergeleitet, welcher daraus die aktuellen Winkeldaten errechnen kann. Diese wiederum werden von einer Software in die Position, sowie die Orientierung umgerechnet.

Die Genauigkeit ist abhängig von der Anzahl der Sender und kann bis zu 0,4mm bei einem Abstand von 50m betragen. (Nikon Metrology GmbH, 2011)
Aufgrund der hohen Kosten für die Anschaffung wurde auch diese Methode der Navigation nicht realisiert.

2.1 Odometrie

Eine der im Projekt zu realisierenden Navigationsroutinen war mit Hilfe der Odometrie zu realisieren. Wie bereits beschrieben, ist es notwendig die Umdrehungen der Räder zu erfassen. Dies wurde im Projekt mit zwei Drehgebern durchgeführt, welche an den beiden angetriebenen Rädern des MULE-Roboters montiert wurden. Diese liefern ein Rechtecksignal mit einer Frequenz, welche von der Geschwindigkeit abhängig ist. Dieses Signal wird an den Mikrocontroller gesendet, welcher die Impulse zählt und somit die zurückgelegte Strecke berechnen kann.

2.1.1 Odometrie – Berechnung des zurückgelegten Weges

Die gezählten Impulse werden im Programm mit den Formeln 1, 2 und 3 in den zurückgelegten Weg umgerechnet. Die daraus resultierenden Größen sind in Abbildung 1 dargestellt. Das in Formel 5 verwendete Übersetzungsverhältnis, zwischen Motor und Getriebe, hat in diesem Projekt den Wert eins, da die Drehgeber direkt mit dem Antriebsrad verbunden sind.

$$C = \frac{\pi \cdot D}{n \cdot I} \qquad (1)$$

$$\Delta s_{L/R} = C \cdot I_{L/R} \qquad (2)$$

$$\Delta\theta = \frac{\Delta s_L - \Delta s_R}{b} \qquad (3)$$

C	Umrechnungsfaktor
D	Raddurchmesser
n	Übersetzungsverhältnis
I	Impulse pro Umdrehung
Δs	Zurückgelegter Weg
$I_{L/R}$	Gezählte Impulse
$\Delta\theta$	Winkel
b	Radabstand

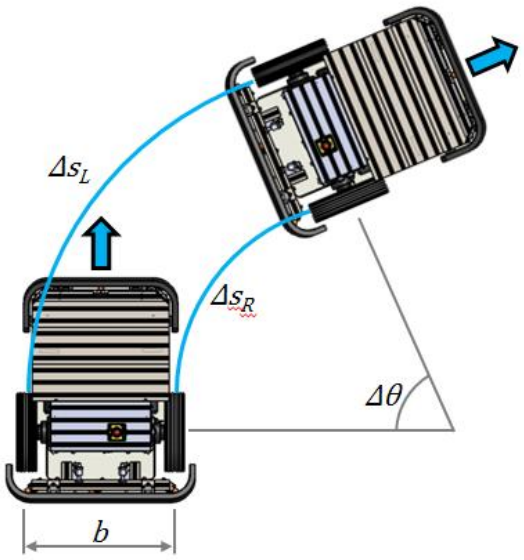


Abbildung 1: Odometrie– Berechnung

2.2 Linienverfolgung

Die Navigation mittels Linienverfolgung in der mobilen Robotik wird bereits in vielen Bereichen erfolgreich eingesetzt. Dies liegt vor allem an den relativ kostengünstigen Sensoren, welche hierfür benötigt werden, sowie an der universellen Implementierungsfähigkeit der Linienverfolgung in eine große Anzahl von Systemen.
Bei dieser Art der Navigation folgt der Roboter einer am Boden befindlichen Linie. Diese Linie muss sich durch ihre Eigenschaften von denen des Untergrundes unterscheiden. Viele Systeme nutzen die unterschiedliche Lichtreflexion zwischen Linie und Untergrund aus, um diese zu erkennen.

Die Navigation mittels optischer Linienverfolgung bietet zahlreiche Vorteile:

- Gezielte Steuerung des Roboters auf einer vorgegebenen Bahn
- Einfache Routenänderung durch Verlegen der Linie
- Unempfindlich gegen Störungen
- kostengünstig Sensorik

Nachteile der Linienverfolgung:

- Wird die Linie unterbrochen, ist eine Weiterfahrt ohne zusätzliche Navigationsroutinen nicht möglich
- Der Radius der Kurve ist durch die Art der Implementierung begrenzt

Es existieren zahlreiche Möglichkeiten um eine Linie mittels Sensoren zu erkennen. Besitzt die Linie spezielle Reflexionseigenschaften, so kann beispielsweise mit Zeilenkameras, Phototransistoren oder Farbsensoren gearbeitet werden.

2.2.1 Linienerkennung mittels Infrarot

Bei optischen Verfahren zur Linienerkennung bereiten Störlichteinflüsse oft Probleme. Um dies zu vermeiden, arbeitet die Linienerkennung in diesem Projekt mit Infrarot.
Der Spektralbereich von Infrarotwellen liegt knapp über dem für den Menschen sichtbaren Bereich und besitzen eine Wellenlänge von $\lambda \cong 1 - 300\mu m$. (Smith, 2008)
Der Vorteil bei der Verwendung von Infrarot ist, dass die entsprechenden Phototransistoren einen Tageslichtfilter besitzen und das Einsatzgebiet nur mit Leuchtstoffröhren beleuchtet wird. Da der Tageslichtfilter die Störeinflüsse von Sonnenlicht minimiert, ist die Aufnahme von Fremdlicht durch die Phototransistoren gering. Bei der Linienerkennung wurde eine schwarze Linie eingesetzt, da diese ein geringeres Reflexionsvermögen besitzt als der vergleichsweise helle Untergrund. Anders gesagt wird von der schwarzen Linie mehr Licht absorbiert als von dem umgebenden Boden.
Für die Linienerkennung des MULE Roboters wurden Infrarot LEDs und Phototransistoren eingesetzt. Zur Erkennung einer am Boden befestigten schwarzen Linie wurden zwei gleichartige Sensorreihen, wie in Abbildung 2 zu sehen, realisiert. Eine Sensorreihe besteht jeweils aus 5 Phototransistoren und 4 Infrarot LEDs, welche nebeneinander auf einer Lochrasterplatte angeordnet sind. Bei jeder der beiden Sensorreihen besteht die Möglichkeit, die mittleren Infrarot LEDs und die äußeren LEDs getrennt voneinander ein- bzw. auszuschalten. Jeder Phototransistor der Sensorreihe besitzt eine eigene Verbindung zum Mikrocontroller des MULE Roboters, womit ein punktuelles Auslesen des reflektierten Infrarotlichts ermöglicht wird.

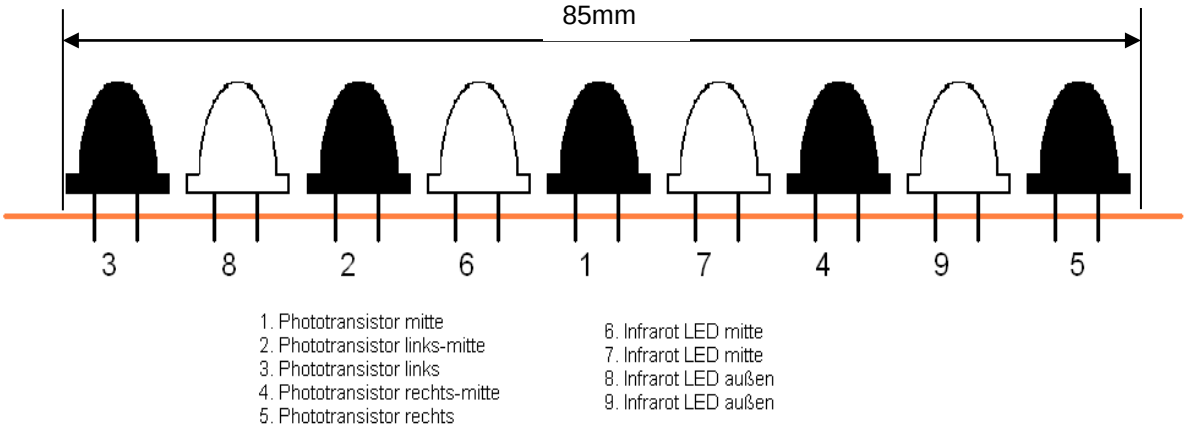


Abbildung 2: Anordnung der Infrarot LEDs und Phototransistoren

2.2.2 Regelung der Motoren mit einem PID Regler

Um einer Linie flüssig folgen zu können ist eine individuelle Regelung der Geschwindigkeit der beiden angetriebenen Räder erforderlich, welche besonders in Kurven entscheidend ist. Ein einfacher schaltender Regler, welcher mit gleichbleibender Geschwindigkeit, bei nach links abweichender Linie nach links und bei nach rechts abweichender Linie nach rechts lenken, würde, ist ungeeignet. Dieser würde in Kurven zu einem Ruckelverhalten führen, da

er kontinuirlich mit hoher Frequenz nach links und anschließend wieder nach rechts lenken würde. Daher wurde ein PID Regler zur Geschwindigkeitsregulation eingesetzt. Dieser ermöglicht bei richtiger Einstellung eine angepasste Beschleunigung bzw. Verzögerung, wodurch eine ruckelfreie Kurvenfahrt ermöglicht wird.
Bei einem PID Regler werden der P, der I und der D Anteil kombiniert, was einer Addition der einzelnen Anteile entspricht. Der P Anteil bewirkt eine Proportionalität der Stellgröße zur Eingangsgröße, welche durch den Proportionalbeiwert K_P eingestellt werden kann. Der I Anteil führt dazu, dass die Stellgröße proportional dem Zeitintegral der Regeldifferenz e_b ist. Diese ergibt sich aus der Differenz zwischen Eingangssignal und Ausgangssignal. Der D Anteil bewirkt eine Proportionalität der Ausgangsgröße zum zeitlichen Differential der Eingangsgröße. Der Proportionalitätsfaktor wird mit K_D bezeichnet. (Reuter und Zacher, 2004)
Der am autonomen Transporter eingesetzte Regler ist ein digitaler Regler, der folgende Stellgrößenberechnung aufweist.

$$Stellgröße = K_P \cdot e_b + K_I \cdot e_b + e_{balt} \cdot T_A + K_D \cdot e_b - e_{balt} \cdot \frac{1}{T_A} \quad (4)$$

Die Konstanten der Regelung wurden empirisch festgelegt:

$$K_P = 0{,}1 \;\; 1 \qquad K_I = 333\ s^{-1} \qquad K_D = 0{,}06[s]$$

Die Abtastzeit T_A ist bei dem verwendeten Algorithmus konstant, da die auftretenden Interrupts vernachlässigt werden können und so immer gleichlange Intervalle zwischen den Abtastungen entstehen. Bei aktivierter Linienverfolgung beträgt die Abtastzeit $\mathrm{T_A}$ 30ms. Diese wurde in weiterer Folge bereits in den Regelparametern berücksichtigt wodurch sich folgende Konstanten ergeben (die Einheiten wurden durch das Einsetzen von $\mathrm{T_A[s]}$ bereits gekürzt):

$$K_P = 0{,}1 \;\; 1 \qquad K_I = 10 \qquad K_D = 2$$

Wie bereits erwähnt, ermöglicht dieser Regler bei einer bestehenden Regelabweichung, welche bei der Linienverfolgung zum Beispiel durch das Abweichen der Linie von der Mitte der Sensorreihe entstehen würde, eine Geschwindigkeitsanpassung der Antriebsräder. Die errechnete Stellgröße wird von der festgelegten Maximalgeschwindigkeit, wie in Formel 5 ersichtlich, abgezogen.

$$Geschwindigkeit\ des\ Rades = maximale\ Geschw. - Stellgröße \quad (5)$$

Hierbei sorgt der I-Anteil des Reglers, bei einer Regeldifferenz, für eine Stellgrößenzunahme und der D-Anteil, bei abnehmender Regelabweichung, für eine Stellgrößenabnahme. In

Abbildung 3 ist die Reaktion des Reglers auf eine Regelabweichung (zwischen Abtastung 4 - 8) nochmals grafisch verdeutlicht.

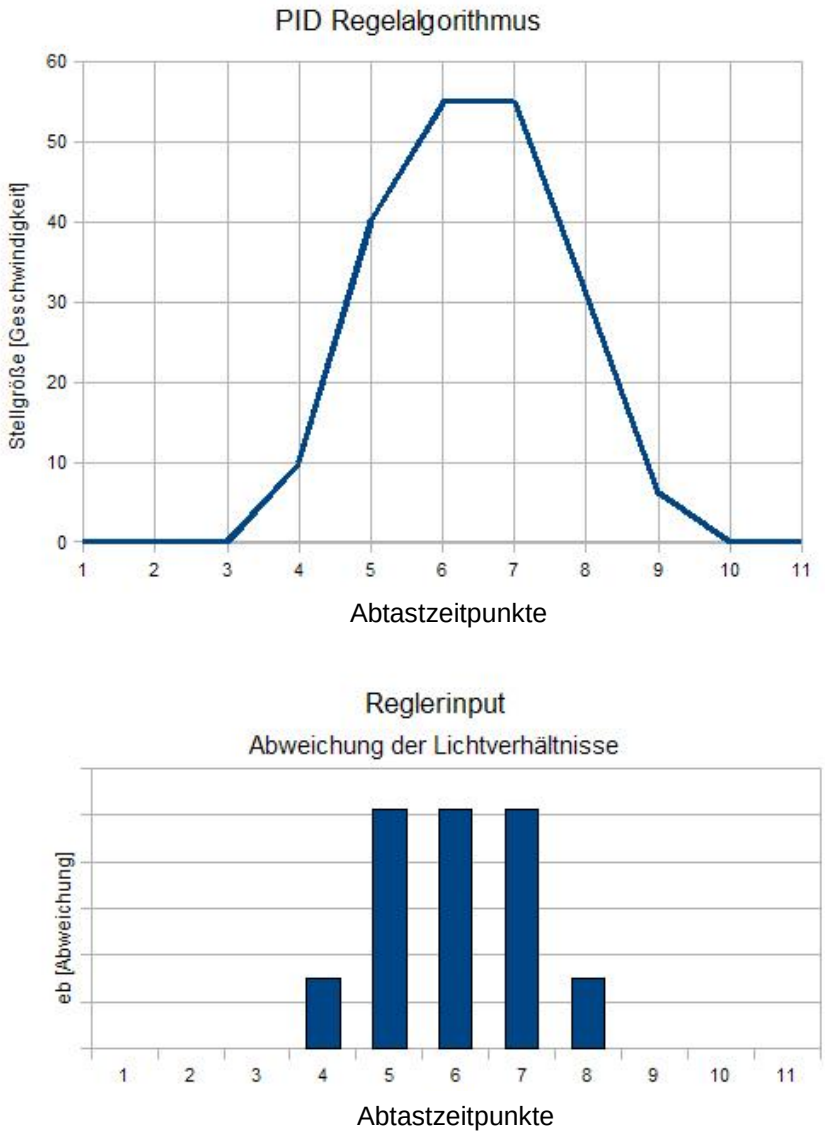


Abbildung 3: PID Regler

Beispielsweise muss bei zunehmender Regelabweichung vom Normalwert auf der linken Seite der Sensorreihe, das linke Rad gebremst werden, damit die Linie wieder in die Mitte der Sensorreihe zurückkehrt. Dies gilt analog auch für die rechte Seite, bzw. das rechte Rad. Der implementierte digitale Regler versucht also durch das Anpassen der Geschwindigkeit, die Linie in der Mitte der Sensorreihe zu halten.

2.2.3 Farbsensor

Der in dem Projekt eingesetzte CSM Farbsensor der Firma Sick kann bis zu zwei verschiedene Farben erlernen, die manuell mittels Teach-In-Modus eingelesen werden können. Direkt am Sensor ist ein Toleranzbereich einzustellen, in welchem die Farben erkannt werden können. Die Toleranzen entscheiden, wie genau der Farbton mit der zuvor gespeicherten Farbe übereinstimmen muss, um erkannt zu werden. Entscheidend für die Einstellung sind hier die Testergebnisse im jeweiligen Einsatzbereich. Bei starken Licht- und Abstandsschwankungen muss eine größere Toleranz gewählt werden, was das Risiko mit sich bringt, dass auch eine Farbe erkannt wird, obwohl keine Übereinstimmung mit den definierten Werten vorliegt. Es wurde eine mittlere Einstellung gewählt, da die Lichtverhältnisse im Raum des Vitero-Labors nicht immer gleich sind.
Beide Farbkanäle wurden für das Projekt eingesetzt, zum einen wurde ein Blauton und zum anderen ein Gelbton eingelesen. Wird nun eine der beiden Farben erkannt, so wird der jeweilige Kanal auf Masse gezogen.
Es wurde festgelegt, dass die mobile Plattform bei blauer Farbe eine Drehung um 180° vollzieht und anschließend die Linie, zur Fortsetzung der Linienverfolgung, wieder sucht. Bei der Erkennung einer gelben Farbmarkierung hält die Plattform zur Becherentnahme für 18 Sekunden an und fährt anschließend mit der unterbrochenen Routine fort.

3 Mechanischer Aufbau

In diesem Kapitel werden die Konstruktion und die Montage der entworfenen Applikationen erläutert. Es wird die mechanische Installation der Sicherheitsvorkehrungen, der Drehgeber, der Linienverfolgung und die des Serviceaufbaus beschrieben. Zu Beginn dieses Abschnitts ist in Abbildung 4 das Gesamtsystem abgebildet, um einen Überblick zu geben.

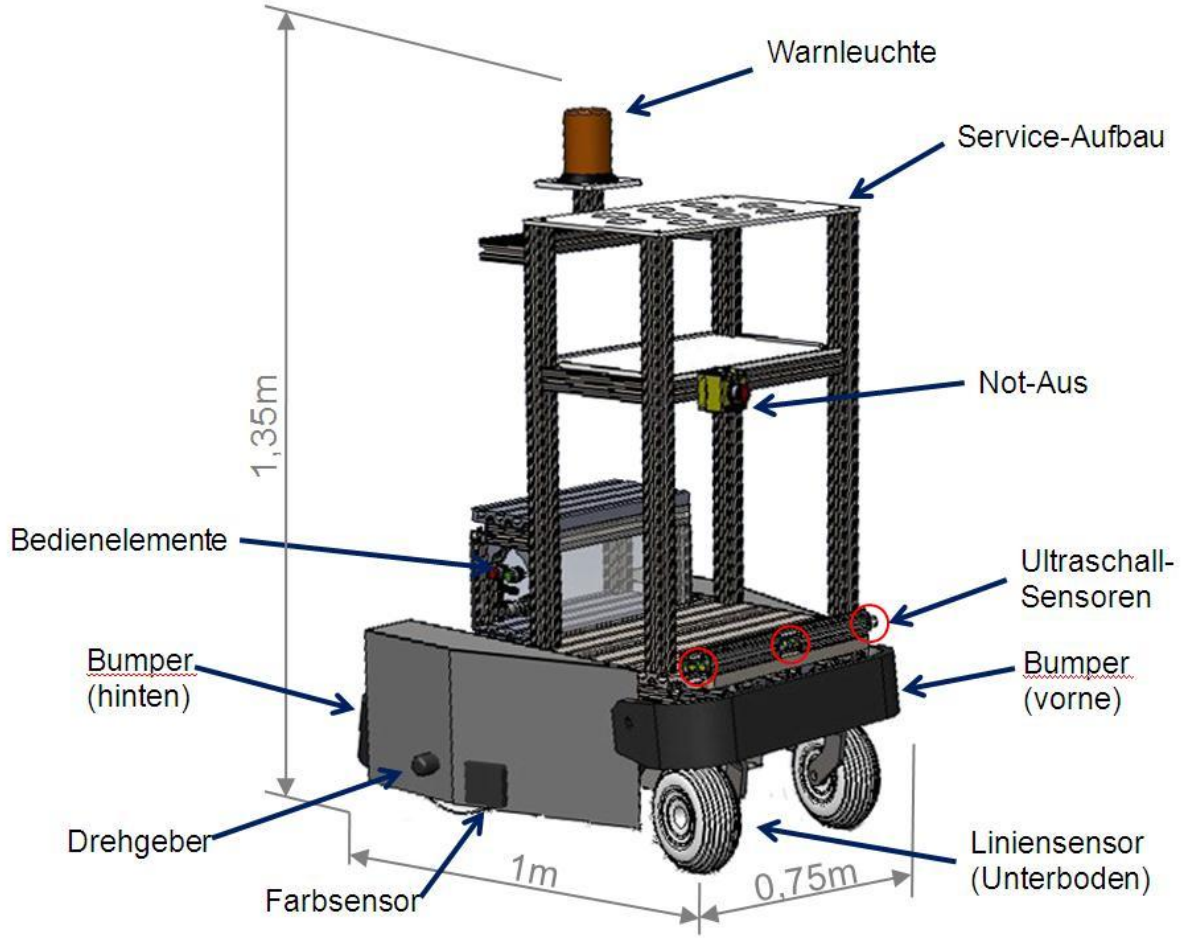


Abbildung 4: Gesamtsystem - Überblick

3.1 Sicherheitsverbau

Auf Grundlage der Ergebnisse der Sicherheitsanalyse wurde die übernommene Plattform mit Abdeckungen versehen, um den Menschen vor potentiellen Gefahren bei der Interaktion mit der Maschine zu schützen. Die seitlichen Abdeckungen schützen davor, dass Körperteile zu den Antriebsrädern gelangen können. Die Abdeckungen wurden aus 2mm-Stahlblech gefertigt, da sich dieses dünnwandige Material zum einen leicht bearbeiten lässt und zum anderen genügend Stabilität mit sich bringt. Das Blech wurde, wie in
Abbildung 5 ersichtlich gebogen, um es der Form des Roboters anzupassen. Die einzelnen Bauteile wurden mit lösbaren Schraubverbindungen miteinander verknüpft. Am hinteren Ende der Plattform wurden die sich gegenüber befindlichen Seitenabdeckungen aus Stabilitätsgründen nochmals miteinander verbunden. Die Bleche wurden lackiert. Die Vorteile einer Lackierung ist das Verhindern von Rost bei Eisenwerkstoffen sowie die optische Aufwertung des Roboters. Die Metallwinkel, zur Befestigung mit der Plattform, wurden eigens gefertigt, entgratet und die Kanten mit Textilklebeband überklebt. In
Abbildung 5 kann man erkennen, wie die Abdeckungen an der Seite der Plattform angeordnet wurden.

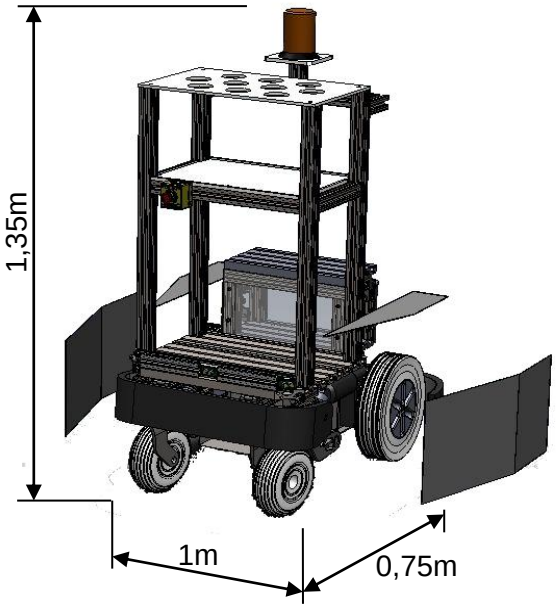


Abbildung 5: Sicherheitsverbau

3.2 Service-Aufbau

Wie schon in der Aufgabenstellung beschrieben war es ein Ziel, mit der autonomen Plattform Verpflegung für die Besucher bereitzustellen. Um dies zu ermöglichen, war eine geeignete Konstruktion eines Aufbaus zu entwerfen, zu fertigen und zu montieren, um Lebensmittel sicher zu transportieren und komfortabel an die Besucher zu bringen.

Für die Verstrebungen des Aufbaus wurden Fabrikate der Firma Robotunits gewählt, da diese Teile zu der zu Projektbeginn übernommenen Plattform kompatibel sind. Ein weiterer Vorteil dieser Konstruktion ist, dass die Applikation mit Schraubverbindungen zu montieren ist. Somit ist sichergestellt, dass Studenten zukünftiger Semester die Verbindung leicht und ohne Schaden lösen können, um den Roboter für andere Zwecke umzurüsten.

Eine in Abbildung 6 ersichtliche Lochrasterplatte stellt sicher, dass die Becher mit den Getränken ohne Spiel, und somit sicher im Aufbau transportiert werden können. Der Durchmesser der Bohrungen, wurde an die verwendeten Becher angepasst, sodass 20 mm des Bechers zur Entnahme überstehen. Als Material wurde hier eine durchsichtige Kunststoffplatte verwendet, da so die sich darunter auf einer Platte befindlichen Lebensmittel für Personen leicht ersichtlich sind. Um sicherstellen zu können, dass die Transportgüter beim Navigieren nicht von der Transportplatte gleiten, wurde ein Rahmen gefertigt und aufgeklebt.

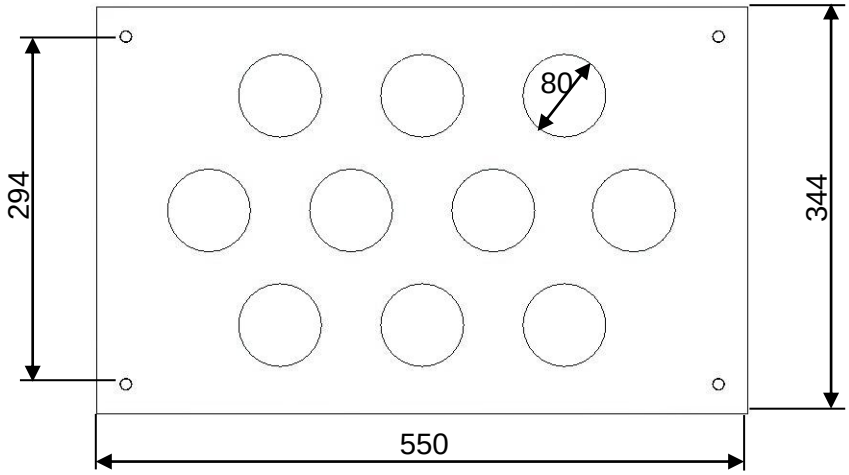


Abbildung 6: Serviceaufbau - Lochraster

3.3 Drehgeber-Montage

Die verwendeten Drehgeber wurden auf dem im Kapitel 3.1 beschriebenen Sicherheitsverbau mit jeweils drei M3 Senkkopfschrauben montiert. Die Verbindung der Drehgeber-Welle mit den Antriebsrädern erfolgte mit einer Federkupplung. Diese wurde verwendet, um einerseits die Fertigungstoleranzen zu berücksichtigen und andererseits, um die Drehgeber von einer möglichen Überlastung zu schützen. Um die Kupplung mit den Antriebsrädern zu verbinden war es notwendig, Adapterschrauben zu konstruieren und zu fertigen, da die original verbauten Radschrauben nicht über die geeignete Montagemöglichkeit verfügten. In der folgenden Abbildung 7 ist dieser Aufbau dargestellt.

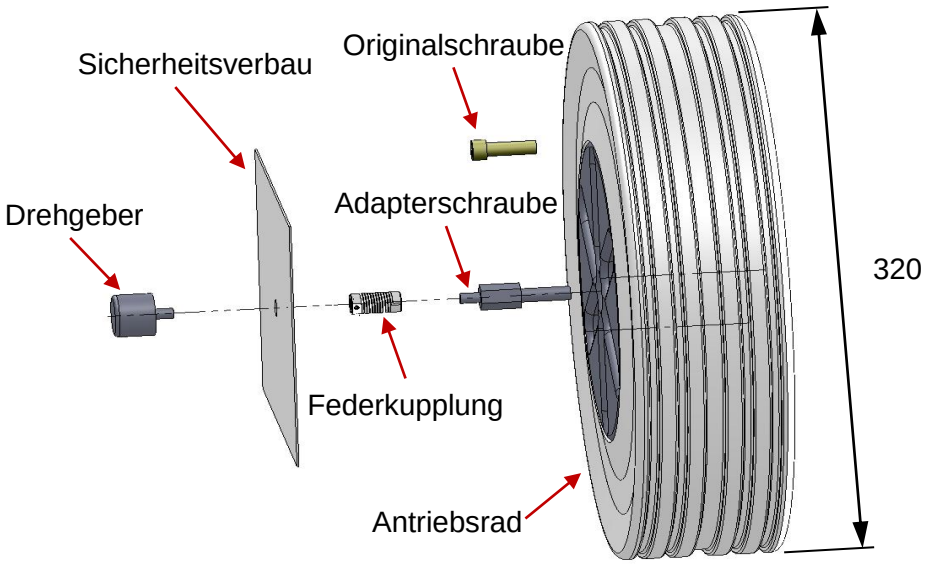


Abbildung 7: Drehgeber-Montage

3.4 Montage der Linienverfolgungselektronik

Es wurden zwei Sensorreihen getrennt voneinander in einem Abstand von $a = 172mm$ auf der Unterseite des Roboters montiert. Dies gibt zusätzliche Information über die Verdrehung des Roboters gegenüber der am Boden befindlichen Linie. In Abbildung 8 ist der Winkel zu sehen, auf welchem die Sensorreihen mit den Infrarot LEDs und den Phototransistoren, nach unten durch die Schlitze auf den Boden gerichtet, montiert sind.

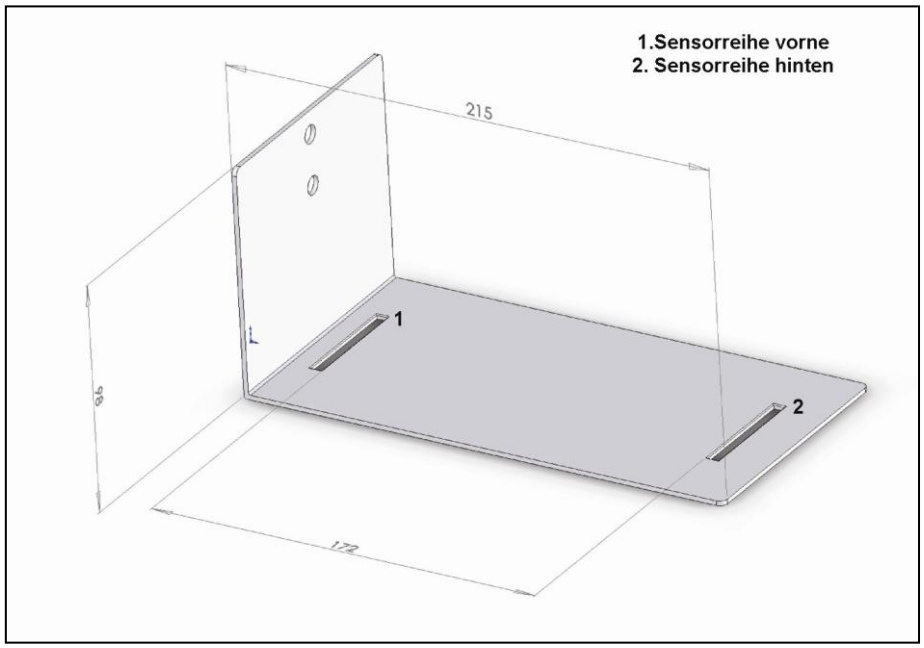


Abbildung 8: Anordnung der Sensorreihen

Der Winkel wurde so montiert, dass eine einfache Justierung des Abstandes zum Boden möglich ist. Nach jeder Änderung des Abstandes zum Boden muss jedoch eine erneute Kalibrierung der Ausgleichswerte im Steuerungsprogramm des Roboters durchgeführt werden, da sich dadurch, die von den Phototransistoren aufgenommene Lichtmenge ändert.

3.5 Optische und akustische Warnelemente

Die optischen und akustischen Warnelemente wurden so ausgelegt, dass Personen in der Nähe des Roboters, auch bei geringer Aufmerksamkeit vor der potenziellen Gefahr, ausgehend durch den MULE-Robtoter, gewarnt werden. Das Risiko eines Personenschadens wird somit minimiert. Der akustische Warnton wurde so eingestellt, dass er selbst bei hoher Umgebungslautstärke zu hören ist. Ein praktisches Beispiel, wo dieses Warnelement eingesetzt wird, ist der Rückfahrwarner beim LKW. Beim MULE-Roboter wurde der Piezo-Schallgeber in die Steuerung im Schaltschrank montiert.

Für die Montage der Warnleuchte muss ein gut ersichtlicher Montageplatz gewählt werden. Hierzu ist sicherzustellen, dass das Signal von allen Seiten gut wahrzunehmen ist. Es lag somit auf der Hand, die Leuchte am höchsten Punkt des Aufbaus für die autonome Plattform zu montieren. Die Verbindung wurde mit Teilen der Firma Robotunits hergestellt. Als Schnittstelle zwischen dem Gerüst aus Aluprofilen und der Signallampe wurde eine quadratische Verbindungsplatte entworfen, deren Konstruktionszeichnung im Anhang ersichtlich ist.

4 Elektrischer Aufbau

In den folgenden Kapiteln werden sämtliche, im Projekt verwendeten, elektrischen Schaltungen erklärt. Dabei wird auf die Dimensionierung der Bauteile, sowie die Funktion der Schaltungen eingegangen.

4.1 Odometrie – Elektronik

Die verwendeten Drehgeber wurden von einem Sponsor zur Verfügung gestellt. Da diese nicht mehr vertrieben werden konnte kein Datenblatt gefunden werden. Die grundlegenden Eigenschaften sind in Tabelle 1 angeführt. Die Kanäle A und B liefern ein zueinander phasenverschobenes Signal, womit es möglich ist die Drehrichtung zu bestimmen. In diesem Projekt wurde allerdings nur der Kanal A verwendet, da für die Implementierung der Navigationsroutine keine Erfassung der Drehrichtung notwendig ist und diese durch die Steuerung der Motoren ohnehin bekannt ist.

Hersteller:	PEWATRON
Typ:	REX-32-360
Versorgungsspannung:	$U_{DC} = 12V \pm 10\%$
Ausgänge:	Kanal A und B
Ausgangssignal:	Rechteck
Impulse/Umdrehung:	360
Art:	Inkrementell

Tabelle 1: Drehgeber-Eigenschaften

Das Kabel der Drehgeber verfügte nicht über die geeignete Länge um es mit der Versorgung und dem Mikrocontroller verbinden zu können. Daher wurde es mit einem Kabel aus dem Bestand der FH Technikum-Wien verlängert, welches allerdings nicht über die gleiche Farbkombination verfügt. Die entsprechende Belegung der einzelnen Adern ist in der Tabelle 2 dargestellt.

Funktion	Farbe Drehgeber	Farbe Verlängerung
$U_{DC} = 12V$	rot	braun
Kanal A	grün	grün
Kanal B	weiß	weiß
GND	schwarz	grau
Schirm	Schirm	Schirm

Tabelle 2: Drehgeber - Kabelbelegung

4.1.1 Odometrie - Pegelanpassung

Die bereits beschriebenen Drehgeber liefern am Ausgang ein rechteckförmiges Signal mit einer Amplitude von $U_{DC} = 12V$. Da die Eingänge des Mikrocontroller maximal mit einer Spannung von $\mathrm{U_{DC}} = 5\mathrm{V}$ betrieben werden dürfen, war es erforderlich das Signal anzupassen. Daher wurde die Spannungsreduktion mit Hilfe einer Optokoppler-Schaltung realisiert, welche in der Abbildung 9 dargestellt ist. Der Anschluss der Zu- und Ableitungen erfolgt über Schraubklemmen, welche ebenfalls in der Abbildung 9 gekennzeichnet sind. Die Taster 1 und 2 dienen dazu, die im Kapitel 5.2 beschriebenen Interrupts, zu Testzwecken manuell auszulösen. Die Widerstände R1–R4 dienen als Vorwiderstände für die Optokoppler. Dabei bilden R1 und R2 einen Vorwiderstand für eine Drehgeberspannung von $U_{DC} = 5V$ und R3 und R4 für $U_{DC} = 12V$. Die Widerstände R5 und R6 sind Pullupwiderstände und dienen dazu den Ausgang des Optokopplers auf ein definiertes Potential zu ziehen.

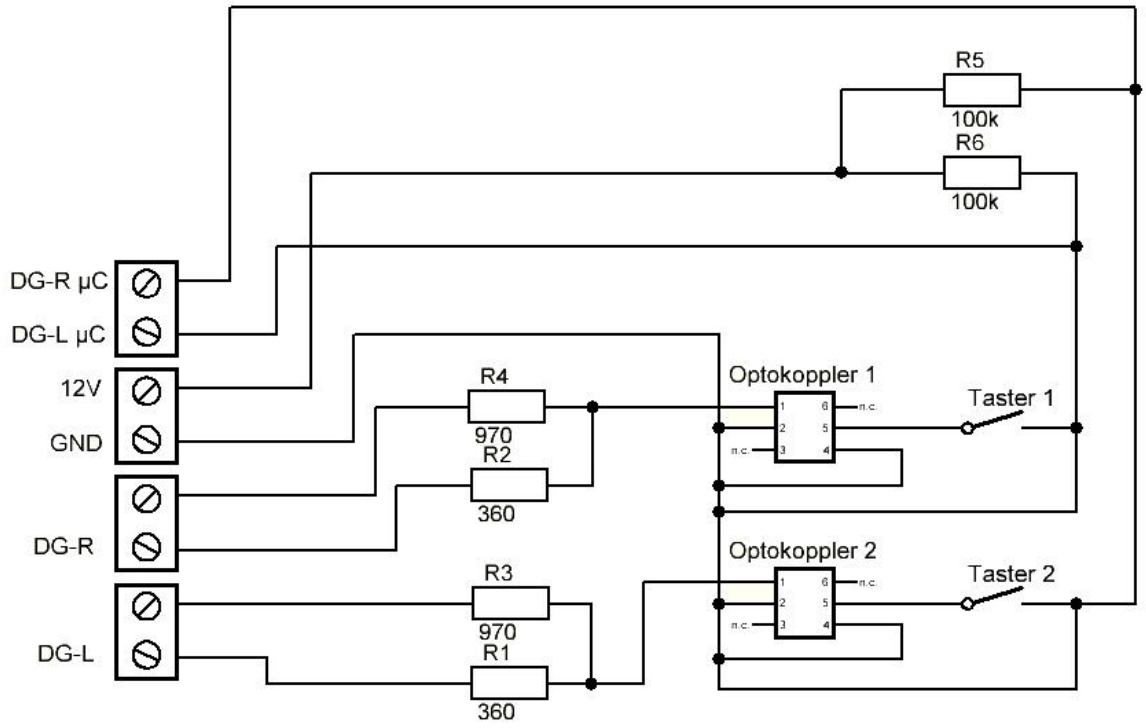


Abbildung 9: Optokoppler-Interface

4.2 Linienverfolgung-Elektronik

Um das reflektierte Licht, welches den Leitwert der Phototransitoren verändert, in brauchbare Werte für den Mikrocontroller des Roboters umzuwandeln, empfiehlt sich eine AD Konversion. Dies wird durch die Beschaltung der Phototransistoren als Spannungsteiler ermöglicht, wie in Abbildung 10 ersichtlich.

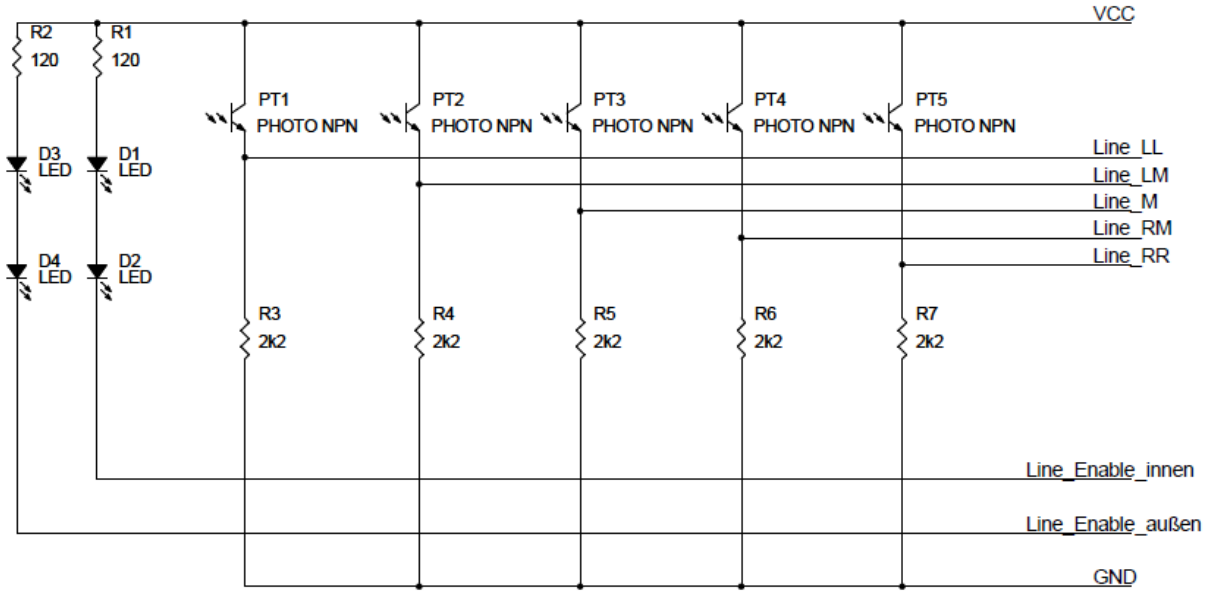


Abbildung 10: Linienverfolgungselektronik

Umso mehr Infrarotlicht von einem Phototransitor (PT) aufgenommen wird, desto geringer wird sein Widerstand und umso höher wird die Spannung an der jeweiligen Leitung zum Mikrocontroller. Vom Spannungsteiler kommende Leitungen (Line_LL, Line_LM, Line_M, Line_RM, Line_RR) wurden an Port A des ATmega32 angeschlossen, da dieser für eine AD Konversion reserviert ist. Somit ist der Mikrocontroller imstande, die am jeweiligen Pin des Port A anliegende Spannung durch AD Konversion in eine Zahl zwischen 0 und 1023 umzuwandeln. Als Referenzspannung wurde AVCC gewählt, da diese der Versorgungsspannung des Mikrocontrollers, also $U_{DC} = 5V$ entspricht. Dies begründet sich durch die Auslegung der Sensorreihen auf eine Versorgungsspannung von ebenfalls $U_{DC} = 5V$. In Tabelle 3 sind einige typische Quantisierungswerte aufgelistet.

10 Bit ADC	
Quantisierungsbereich: 0 - 5V q = 0,004882813 V	
Spannung(V)	**unsigned Integerwert nach Konversion**
0	0
0,5	102
1	204
1,5	307
2	409
2,5	511
3	614
3,5	716
4	819
4,5	921
5	1023

Tabelle 3: typische Quantisierungswerte bei der AD Konversion

Die Infrarot-LEDs sind in Serie mit einem passenden Vorwiderstand beschalten. Die IR-LEDs können vom Mikrocontroller ein- bzw. ausgeschaltet werden, indem die Leitung Line_Enable durch einen Ausgang des Mikrocontrollers auf GND, bzw. auf V_{cc} gezogen wird. Es ist möglich, die inneren Infrarot-LEDs getrennt von den äußeren anzusteuern. Dies ermöglicht die Anwendung von Modulationsverfahren und die Messung des Störlichtes bei ausgeschalteten Infrarot-LEDs.

4.3 Piezo-Schallgeber-Elektronik

Die Eigenschaften des in Kapitel 3.5 erwähnten Piezo-Schallgebers sind in Tabelle 4 angeführt. Dieser wird in diesem Projekt mit einer Frequenz von $f = 4000Hz$ und einer Spannung von $\mathrm{U_{DC}} = 5\mathrm{V}$ betrieben, da sich in einem Testaufbau herausstellte, dass der dadurch erzeugte Ton am geeignetsten für die Anwendung ist. Die Erzeugung dieser Frequenz konnte nicht mit dem Mikrocontroller realisiert werden da die dazu notwendigen Timer/Counter Einheiten bereits durch andere Funktionen belegt waren. Daher wurde die Frequenz mittels einer astabilen Kippstufe realisiert, welche in Abbildung 11 (Grüner Abschnitt) dargestellt ist. Die Dimensionierung der Bauteile erfolgte mit der Formel 6. Um den Piezo-Schallgeber ähnlich eines Rückfahrwarners von LKW ertönen zu lassen, war eine zweite astabile Kippstufe notwendig. Diese ist ebenfalls in Abbildung 11 (Roter Abschnitt) dargestellt und wurde mit einem NAND-Baustein realisiert. Das Potentiometer P1 dient der Einstellung des Ton-Intervalls. Die gesamte Schaltung kann mit einem Ausgang des Mikrocontrollers (Blauer Kreis) ein- bzw. ausgeschaltet werden. Die Dimensionierung der verwendeten Bauteile erfolgte mit der Formel 7. Es wurden zwei unterschiedliche Arten von Kippstufen realisiert da die Schaltung mit im Labor vorhandenen Komponenten aufgebaut

wurde und für zwei identische Kippstufen nicht die erforderliche Anzahl an Bauteilen vorhanden war.

Hersteller:	Sonitron
Typ:	SMAT 21
Versorgungsspannung:	$U_{DC} = 0 - 30V$
Frequenzbereich:	$f = 600 - 5000Hz$
Schalldruck bei $\mathrm{f} = 4000\mathrm{Hz}$ **:**	$A \cong 83dB$

Tabelle 4: Piezo-Schallgeber – Eigenschaften

$$f_1 = \frac{1}{T} = \frac{1}{\ln(2) \cdot (R_2 \cdot C_1 + R_3 \cdot C_2)} = \frac{1}{\ln(2) \cdot (8k2\Omega \cdot 22nF + 8k2\Omega \cdot 22nF)} \cong 4\ kHz \quad (6)$$

$$f_2 = \frac{1}{T} = \frac{1}{\ln(3) \cdot (2 \cdot P_1 \cdot C_3)} = \frac{1}{\ln(3) \cdot (2 \cdot [1 \ldots 1k\Omega] \cdot 1{,}5mF)} \cong 303 \ldots 0{,}303\ Hz \quad (7)$$

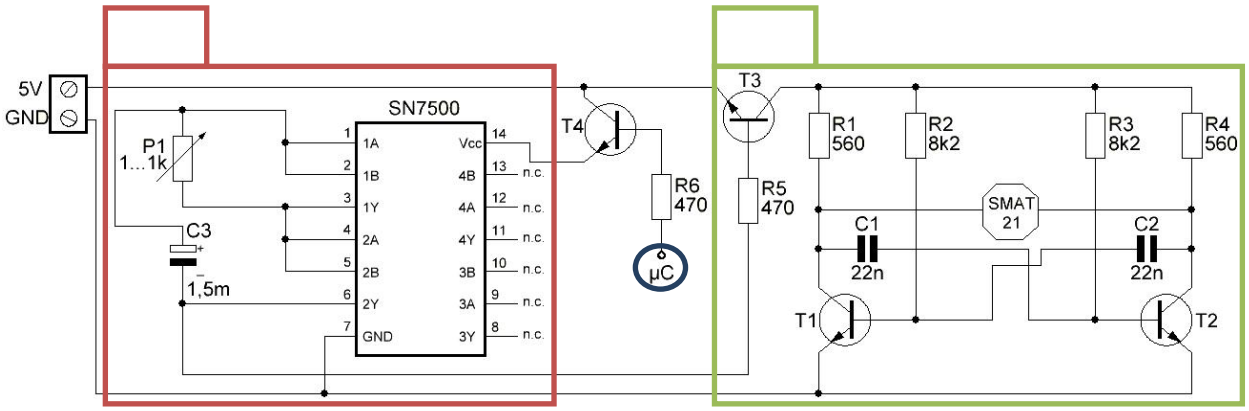
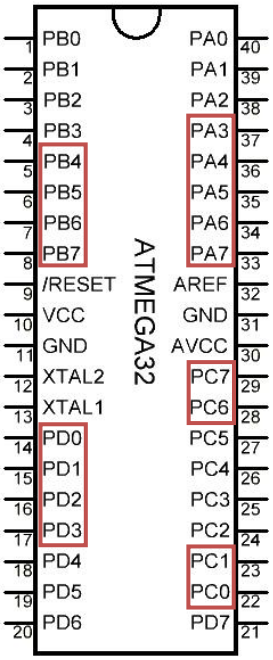


Abbildung 11: Piezo-Schallgeber-Elektronik

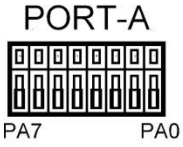
4.4 ATmega32 Pinbelegung

Die Pinbelegung des Mikrocontrollers wurde im Vergleich zum Vorgängerprojekt geringfügig geändert. Dies war erforderlich da einige, in diesem Projekt verwendete, Funktionen des Mikrocontrollers nur an bestimmten Pins verfügbar sind. Die geänderte Pinbelegung ist in den Abbildungen Abbildung 12 und Abbildung 13 dargestellt.



Pin	Funktion	
PB4	v. l., h. r.	Ultraschallsignal
PB5	v. m.	
PB6	v. r., h. l.	
PB7	h. l.	Ultraschallecho
PC0	v. r.	
PC1	v. m.	
PD0	h. r.	
PD1	v. l.	
PD2	INT0	Drehgeber
PD3	INT1	
PC6	Blau	Farbsensor
PC7	Gelb	

Abbildung 12: ATmega32 Pinbelegung



Pin	Funktion
PA0	n. c.
PA1	n. c.
PA2	n. c.
PA3	PT5
PA4	PT1
PA5	PT4
PA6	PT3
PA7	PT2

Abbildung 13: ATmega32 - Port-A Pinbelegung

5 Steuerung

In den folgenden Kapiteln wird nun die Einbindung der bereits beschriebenen Sensoren und des PID Reglers in das Steuerprogramm erläutert. Es wird sowohl der Ablauf, als auch der in einzelne Funktionen getrennte Programmcode beschrieben. Zuerst erfolgt die Beschreibung der Linienverfolgung und des Farbsensors. Anschließend wird die Einbindung der Drehgeber erläutert und abschließend ist der Programmablauf dargestellt.

5.1 Einbindung der Linienverfolgung und des Farbsensors

Für die Implementierung der Linienverfolgung musste zunächst eine Kalibrierung der Sensoren durchgeführt werden. Diese wird im nächsten Kapitel erörtert. In den nachfolgenden Abschnitten werden zunächst die Softwarefunktionen der AD Konversion und anschließend der PID Regler beschrieben. Den dritten Teil bildet die Implementierung des Farbsensors.

5.1.1 Kalibrierung

Um gewährleisten zu können, dass die Linienverfolgung in dem vorher definierten Bereich einwandfrei funktioniert, muss das von der Umgebung reflektierte Licht eingelesen und die Werte gespeichert werden. Die Linierverfolgung in diesem Projekt ist für den Einsatz im Vitero-Labor ausgelegt. Der einzulesende Untergrund des Einsatzortes weist unterschiedliche Helligkeitswerte auf, die sich durch Verschmutzungen, unterschiedliche Lichtverhältnisse und Abstandsdifferenzen des teilweise unebenen Bodens ergeben. Auch die gelegte Linie, realisiert mit schwarzem Isolierband, weist leichte Helligkeitsunterschiede auf. Diese werden beispielsweise durch Höhenunterschiede verursacht, da beim Aufkleben des Bandes Wellen entstehen können. Auch entstehen am Band mit der Zeit Verschmutzungen oder Beschädigungen, wie Risse oder dergleichen.
Für das Einlesen der Kalibrierungswerte wird der autonome Transporter auf möglichst unterschiedlichste Orte, bei unterschiedlichsten Lichtverhältnissen, im Labor platziert. Die vier sendenden IR-LED‘s strahlen nun auf den Boden und die fünf lesenden Fototransistoren nehmen die jeweils aktuellen Helligkeitswerte wahr und es entstehen der Helligkeit entsprechende Spannungswerte. Jeder Transistor hat leicht unterschiedliche Spannungswerte, jedoch ist die Reaktion auf die Lichtveränderungen immer klar feststellbar. Die Spannungen werden nun von dem ADC im Mikrocontroller in binäre Werte gewandelt. Über die USB-Verbindung zum Computer mittels JTAG Programmiergerät können nun über den Debugging-Modus im AVR-Studio diese Werte ausgelesen werden. Dieser Vorgang wird auf den unterschiedlichsten Orten zum einen ohne Band und zum anderen mit Band wiederholt.
Man kann aufgrund der digitalen Werte, entsprechend der Helligkeit, klare Tendenzen erkennen, die nun signalisieren, ob sich am Untergrund ein Band befindet, oder nicht. Diese

eingelesenen Helligkeitswerte werden auch die in den nachfolgend beschriebenen Berechnungen verwendet und so kann die Lage der Linie ermittelt werden.

5.1.2 Implementierung der Linienverfolgung

Die Linienverfolgung wurde in das bestehende Roboterprogramm integriert. Dazu wurden fünf Funktionen erzeugt: `Linienverfolgung_init()`, `Linie_lesen()`, `ADC_Read()`, `ADC_Read_Avg()`, `vsoll_pid()`.
Bevor Werte vom ADC gelesen werden können, muss dieser initialisiert werden. Dies geschieht in der Funktion `Linienverfolgung_init()`, welche in Abbildung 14 zu sehen ist. Zunächst wird die Quelle der Referenzspannung angegeben, wobei hier beim ATmega32 drei Möglichkeiten zur Verfügung stehen: `AVCC`, `AREF` und die interne Referenzspannung. Wird `AVCC` oder `AREF` gewählt, muss am entsprechenden Pin des Mikrocontrollers eine Referenzspannung zur Verfügung stehen. Wird die interne Referenzspannung gewählt, können diese Pins frei bleiben, da der Mikrocontroller eine eigene Referenzspannung von $U_{ref} = 2{,}2V$ nützt. Anschließend wird die Abtastfrequenz eingestellt. Hierbei ist auf die Einhaltung des Abtasttheorems zu achten:

Die Frequenz, mit welcher ein zu konvertierendes Signal abgetastet wird, muss mindestens doppelt so groß sein als die höchste im Signal vorkommende Frequenz. (Flegel et al., 2009)

Die erste AD-Konversion wird durchgeführt und verworfen, da die erste Konversion nach dem Initialisieren oft Fehler aufweisen kann.

```
voidLinienverfolgung_init(void){
/* ADC initialisieren */
  uint16_t result;
  ADMUX =(0<<REFS1)|(1<<REFS0);// AVcc als Referenz benutzen
//  ADMUX = (1<<REFS1) | (1<<REFS0);      // interne Referenzspannung
nutzen
  ADCSRA =(1<<ADPS1)|(1<<ADPS0);// Frequenzvorteiler
  ADCSRA |=(1<<ADEN);// ADC aktivieren
/* nach Aktivieren des ADC wird ein "Dummy-Readout" empfohlen, man
liest
     also einen Wert und verwirft diesen, um den ADC "warmlaufen zu
lassen" */
  ADCSRA |=(1<<ADSC);// eine ADC-Wandlung
while(ADCSRA &(1<<ADSC)){}// auf Abschluss der Konvertierung warten
/* ADCW muss einmal gelesen werden, sonst wird Ergebnis der nächsten
     Wandlung nicht übernommen. */
result= ADCW;

}
```

Abbildung 14: Die Funktion Linienverfolgung_init()

Um Einzelmessungen mit dem ADC durchzuführen, wurde die Funktion ADC_Read() programmiert. Wie in Abbildung 15 zu sehen ist, wird beim Aufruf der Funktion zunächst der entsprechende Pin an den ADC „angeschlossen“. Dies ist notwendig, da der ATmega32 nur einen ADC für alle 8 Pins des Port A besitzt. Danach folgt eine AD Konversion. Die while-Schleife wartet so lange, bis das ADSC Bit im ADCSRA Register vom ADC gesetzt wurde, da dies den Abschluss der Quantisierung signalisiert. Anschließend wird der gemessene 10 Bit Wert von der Funktion zurückgegeben.

```
/* ADC Einzelmessung */
uint16_t ADC_Read( uint8_t channel)
{
// Kanal waehlen, ohne andere Bits zu beeinflußen
ADMUX =(ADMUX &~(0x1F))|(channel &0x1F);
ADCSRA |=(1<<ADSC);// eine Wandlung "singleconversion"
while(ADCSRA &(1<<ADSC)){}// auf Abschluss der Konvertierung warten
return ADCW;// ADC auslesen und zurückgeben
}
```

Abbildung 15: Die Funktion ADC_Read()

In Abbildung 16 ist die Funktion `ADC_Read_Avg` dargestellt, welche dazu dient, mehrere Einzelmessungen durchzuführen und einen Durchschnittswert zu errechnen. Dies hat den Vorteil, dass eventuelle Spannungsspitzen, bzw. Störungen durch mehrere, zeitlich leicht verschobene, Messungen ausgeglichen werden können.

```
/*ADC Mehrfachmessung mit Durchschnittsbildung*/
uint16_t ADC_Read_Avg( uint8_t channel, uint8_t average)
{
uint32_t result =0;

for(uint8_t i =0; i < average;++i )
result+=ADC_Read( channel );

return(uint16_t)( result / average );
}
```

Abbildung 16: Die Funktion ADC_Read_Avg();

Die vierte für eine Linienverfolgung nötige Funktion trägt den Namen `Linie_lesen()`. Diese führt die Schritte 2-7 des in Abbildung 17 dargestellten Flussdiagrammes aus.

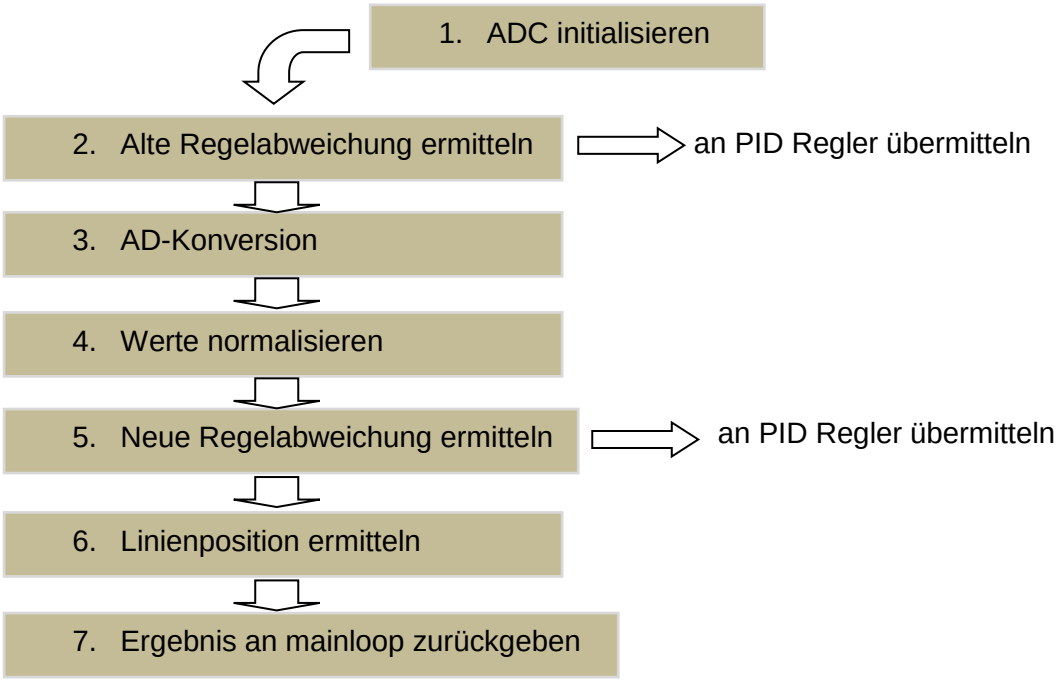


Abbildung 17: Flussdiagramm Funktion Linie_lesen()

Bevor eine AD-Konversion stattfindet, werden zunächst die bestehenden Regelabweichungen der einzelnen Phototransistoren für die Übertragung an den PID Regler, welcher für die Geschwindigkeitsregelung verantwortlich ist, gesichert. Dieser wird im Zuge der Funktion `vsoll_pid()` näher beschrieben.
Dann werden mit Hilfe der Funktion `ADC_Read_Avg()` die einzelnen, von den Phototransistor-Spannungsteilern zurückgelieferten Spannungen in je eine Integer-Zahl gewandelt. Diese werden dann mit den Kalibrierungswerten verglichen und in einen Bereich von 0-1023 umgewandelt, wobei 0 einer erkannten Linie entspricht und 1023 den Helligkeitswert des Untergrundes wiedergibt. Dies geschieht anhand der Formel 8:

$$Abweichung_{neu} = (Abweichung_{gem.} - Helligkeit_{Linie}) \cdot \frac{1023}{Helligkeit_{Boden} - Helligkeit_{Linie}} \quad (8)$$

Dadurch kann sichergestellt werden, dass alle gemessenen Werte auf den gleichen Bereich skaliert sind. Die so errechnete Abweichung wird ebenfalls an den PID Regler übergeben
Im nächsten Schritt werden nun die normalisierten Abweichungen mit einer festgelegten Toleranz verglichen. Wird diese überschritten, lässt dies auf die Präsenz der Linie unter dem entsprechenden Phototransistor, welcher die Abweichung verursachte, schließen. Anschließend gibt die Funktion `Linie_lesen()` ein Ergebnis zurück, welches die in Tabelle 5 angegebenen Werte annehmen kann.

Ergebnis	Bedeutung
10	Phototransistor links sieht Linie
20	Phototransistor links-mitte sieht Linie
30	Phototransistor mitte sieht Linie
40	Phototransistor rechts-mitte sieht Linie
50	Phototransistor rechts sieht Linie
0	keine Linie erkannt oder mehrere Phototransistoren sehen Linie

Tabelle 5: Ergebnisse der Funktion Linie_lesen()

Die Einerstelle des Ergebnisses ist für die zweite Sensorreihe reserviert, welche nicht verwendet wurde, da sich im Laufe der Entwicklung herausstellte, dass diese für eine fehlerfreie Linienverfolgung nicht benötigt wird.
Die fünfte Funktion hat die Regelung der Geschwindigkeit zur Aufgabe. Dies ist vor allem beim Fahren einer Kurve von großer Bedeutung, da hier ein sanftes An- und Abregeln der Geschwindigkeit Voraussetzung für eine störungsfreie Fahrt ist. Dies geschieht in der Funktion `vsoll_pid()`, welche in Abbildung 18 zu sehen ist.

```
intvsoll_pid(int direction)
{
 //direction gibt das Rad an welches geregelt werden soll
 //direction = 1 --> linkes Rad direction = 2 --> rechtes Rad
 //Konstanten der Regelung definieren
 #defineKp 1
 #defineKi 10
 #defineKd 2
 #define Skalierung 1000
 //aktuelle Abweichung errechnen
 if(direction==1)eb=1023-frontleft;
 if(direction==2)eb=1023-frontright;
 if(direction==11) eb=1023-frontlefta;
 if(direction==22)eb=1023-frontrighta;
```

Abbildung 18: Funktion vsoll_pid() Teil1

Die Funktion `vsoll_pid()` gibt als Ergebnis einen Sollwert für die zu fahrende Geschwindigkeit zurück. Da der Roboter zwei Motoren besitzt und ebenfalls zwischen den inneren und den äußeren Phototransistoren unterschieden wird, kann die Funktion in vier verschiedenen Modi aufgerufen werden. Diese werden durch die Übergabe eines Parameters namens `direction` ausgewählt. Im ersten Schritt wird nun die Regelabweichung e_b erstellt, welche umgekehrt proportional zum reflektierten Licht ist. Je nach gewähltem Modus wird die Abweichung des linken bzw. rechten inneren bzw. äußeren Phototransistors zur Berechnung herangezogen. Die Werte hierfür wurden bereits in der Funktion `Linie_lesen()` durch die AD-Wandlung bestimmt. Im nächsten Schritt, welcher in Abbildung 19 zu sehen ist, wird der Sollwert der Geschwindigkeit unter Berücksichtigung

der aktuellen Regelabweichung e_b, sowie der vorangegangenen Regelabweichung e_{balt} berechnet.

```
//Regelabweichung berechnen
if(direction==1)vsollwert=Kp*eb+Ki*(eb+eblalt)+Kd*(eb-eblalt);
if(direction==2)vsollwert=Kp*eb+Ki*(eb+ebralt)+Kd*(eb-ebralt);
if(direction==11)vsollwert=Kp*eb+Ki*(eb+eblaalt)+Kd*(eb-eblaalt);
if(direction==22)vsollwert=Kp*eb+Ki*(eb+ebraalt)+Kd*(eb-ebraalt);
//Regelabweichung skalieren
//Der berechnete Wert wird zwischen max. und min. Geschwindigkeit
//eingeordnet
normieren =(((double)GESCHWMAX -
(double)GESCHW)/(((double)Kp*1024+(double)Ki*(1024+1024))/(double)Skal
ierung));
vsollwert=((vsollwert/Skalierung)*normieren);
returnvsollwert;
}
```

Abbildung 19: Funktion vsoll_pid() Teil2

Hierzu wird der bereits erläuterte PID Regelalghoritmus verwendet, dessen Berechnung mit Formel 9 erfolgt.

$$v_{Sollwert} = K_P \cdot e_b + K_I \cdot \ e_b + e_{balt} \ + K_D \cdot \ e_b - e_{balt} \qquad (9)$$

Nach der Berechnung des Geschwindigkeitssollwerts wird dieser wieder auf einen Bereich zwischen der festgelegten Minimalgeschwindigkeit und Maximalgeschwindigkeit normiert. Der errechnete Sollwert wird von der festgelegten Maximalgeschwindigkeit subtrahiert und das Ergebnis bildet die neu einzustellende Geschwindigkeit. Eine Einstellung der Geschwindigkeit geschieht durch Manipulation des an die Brückenschaltung der Motoren weitergegebenen PWM Signales. Dies kann im bestehenden Roboterprogramm komfortabel durch Verändern des Compare-Wertes des Timerregisters, welches das PWM Signal erzeugt, erfolgen. Dies erfolgt in 5 zeitversetzten Stufen, damit durch große Sollwertänderungen keine zu großen Differenzen entstehen, welche zu Ruckelverhalten des Roboters führen würden.

5.1.3 Implementierung des Farbsensors

Die zwei Ausgänge des Farbsensors, wobei je einer für eine erlernte Farbe des Sensors reserviert ist, sind über ein Opptokoppler Interface mit zwei Eingängen auf dem ATMega32 verbunden. Diese Eingänge sind PC6 und PC7 und werden im Mainloop des Steuerprogrammes zyklisch abgefragt. Wird PC6 aktiv, so wird mittels Odometrie eine Drehung um 180° durchgeführt und anschließend bei gerader Fahrt versucht, die Linie wiederzufinden. Gelingt dies nicht innerhalb von 10 Sekunden, so stoppt der Roboter und führt keine weiteren Aktionen mehr aus bis die Funktion `Linie_lesen()` wieder ein gültiges Ergebnis liefert. Für das Wiederauffinden der Linie sollte vor der Farbmarkierung, welche PC6 aktiviert, die Linie in einer geraden von 2 Metern Länge ausgeführt sein. PC7

aktiviert einen Halt des Roboters, wobei dieser die Motoren stoppt und 18 Sekunden wartet, bevor mit der Linienverfolgung fortgefahren wird.

5.2 Einbindung der Drehgeber

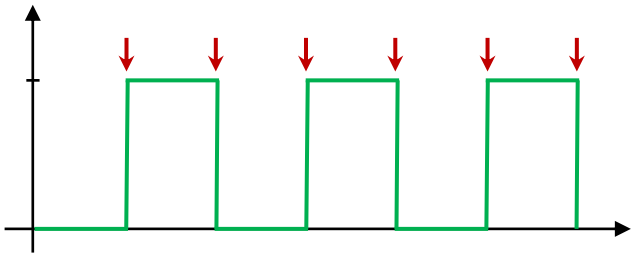
Wie auch die Linienverfolgung, wurde auch die Odometrie-Routine in das bestehende Roboterprogramm integriert. Wie schon beschrieben, ist es notwendig, die vom Drehgeber gelieferten Impulse zu zählen. Dies wird in den meisten Fällen mit Hilfe einer Timer/Counter Einheit realisiert, welche in Hardware auf dem Mikrocontroller vorhanden ist. Da aber alle drei vorhanden Timer/Counter bereits durch das vorhandene Programm belegt waren, wurden externe Interrupts für das Zählen der Impulse verwendet. Bei dieser Art der Implementierung wird bei einer definierten Änderung des Eingangssignals eine Interrupt Service Routine aufgerufen. In dieser wird dann eine Zählervariable inkrementiert oder dekrementiert und so die zurückgelegte Strecke berechnet. Die Abbildung 20 zeigt die Initialisierung der verwendeten externen Interrupts INT0 und INT1 in der main-Funktion. Die Bits `ISC11`. `ISC10`, `ISC01` und `ISC00` sind Teil des MCU Controll Registers und legen fest, auf welche Ereignisse die Interrupts reagieren sollen. In diesem Projekt wurde festgelegt, dass bei jeder High-/Low-Änderung des Drehgebersignals ein Interrupt ausgelöst wird (Abbildung 22). Ist dies der Fall, wird die in Abbildung 21 gezeigte ISR für das jeweilige Rad aufrufen. Abhängig vom Wert der globalen Variablen `vzl` bzw.`vzr` werden die globalen Variablen `odo_links` bzw. `odo_rechts` inkrementiert oder dekrementiert. Die Variablen `vzl` bzw. `vzr` werden in den Funktionen, für das Ansteuern der Motoren, entsprechend gesetzt. Die Funktion `sei()` aktiviert die globalen Interrupts des Mikrocontrollers und ist zwingend erforderlich, um einen korrekten Programmablauf zu gewährleisten.

```
//Externe Interrupts initialisieren:
MCUCR =(0<<ISC11)|(1<<ISC10)|(0<<ISC01)|(1<<ISC00);
//INT0 und INT1 reagieren auf jede High-/Low Änderung
GICR =(1<<INT1 |1<<INT0);      //INT0 und INT1 aktivieren
sei();                         //globale interrupts aktivieren
```

Abbildung 20: Externe Interrupts initialisieren

```
//Interrupt-Service-Routine für int0(linkes Rad)
ISR(INT0_vect)
{
 if(vzl==0)odo_links++;
 elseodo_links--;
}
```

Abbildung 21: Odometrie – ISR



➔ 720 Interrupt-Requests/Umdrehung

Abbildung 22: Odometrie – Interruptrequest

5.3 Einbindung der Warnelemente

Da nach der Implementierung der in den vorigen Kapiteln beschriebenen Funktionen alle Ein- bzw. Ausgänge des Mikrocontrollers belegt waren, wurden die Warnelemente ohne diesen angesteuert. Die Elektronik des Piezoschallgebers wurde an die Versorgungsspannung des Mikrocontrollerboards angeschlossen. Somit ist das akustische Warnelement aktiv, wenn der Modi-Schalter des Roboters auf „Programm“ oder „Drive“ steht. Die Warnleuchte wurde auf die - durch ein Relais gesteuerte - Stromversorgung des Motortreibers geschaltet. Dadurch ist die Warnleuchte aktiv, wenn der Modi-Schalter auf „Drive“ steht und der Start-Knopf gedrückt wurde.

5.4 Ablauf des Programmes

An dieser Stelle wird nun der Programmablauf näher beschrieben. Wie in Abbildung 23 gezeigt, beginnt das Steuerprogramm des autonomen Transporters mit der Initialisierung. Hier werden die PWM, der ADC, die Ports und die Interrupts initialisiert. Anschließend wird eine Hindernisdetektion mit Hilfe der Ultraschallsensoren durchgeführt. Sollten diese ein Hindernis erkennen, hält die autonome Plattform für die Zeit der Präsenz des Hindernisses. Im nächsten Schritt wird die Linie gesucht. Wird bei laufender Detektion keine Linie gefunden, fährt der Roboter für maximal 1,5 Sekunden gerade weiter und hält dann an bis er erneut eine Linie erkennt. Wird eine Linie erkannt, wird der PID Regler aufgerufen. Dann wird die Geschwindigkeit der Antriebsräder in 5 Stufen auf das Ergebnis der Regelung eingestellt. Abschließend findet eine Abfrage des Farbsensors statt.

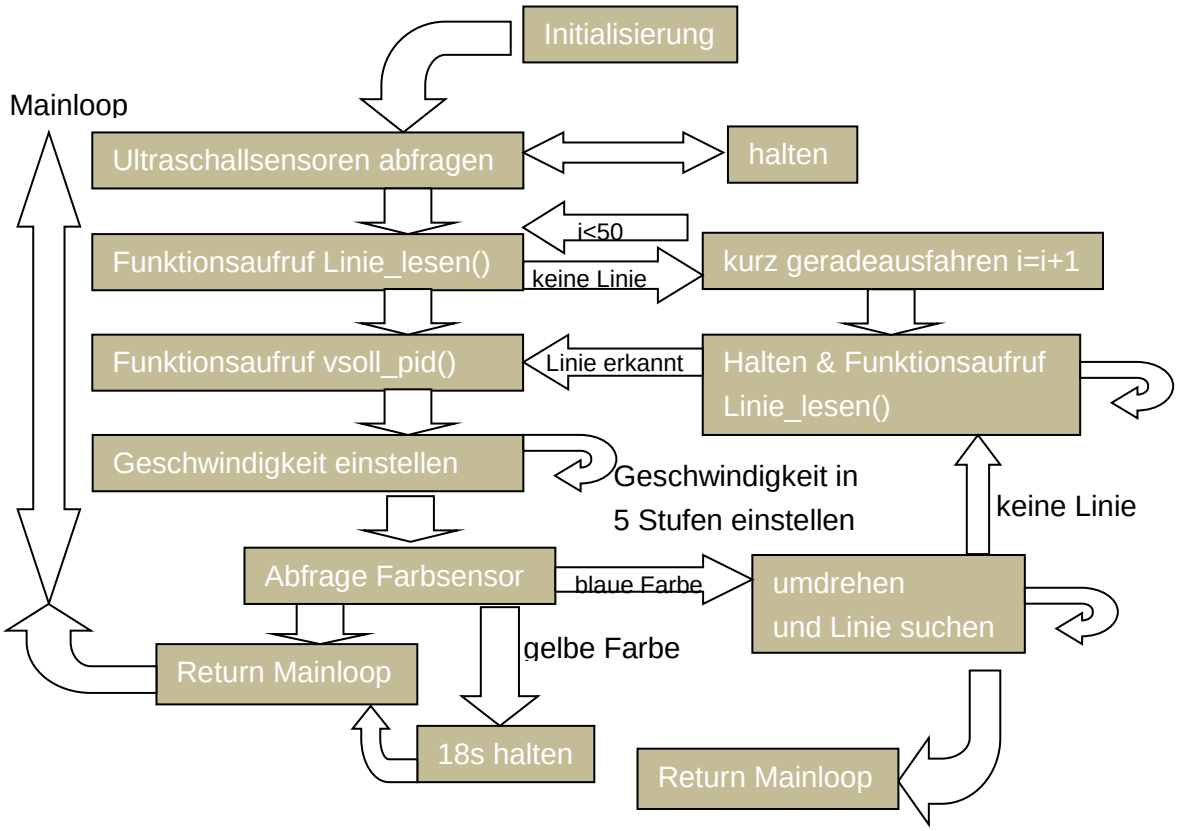


Abbildung 23: Programmablauf

6 Ergebnisse

Die implementierten Funktionen wurden sowohl im Laufe der Projektzeit, als auch in den abschließenden Tests überprüft. Die daraus resultierenden Ergebnisse sind in den nächsten Kapiteln dokumentiert.

6.1 Test der Odometrie

Für die Tests der odometrischen Navigationsroutine wurden zwei Testverfahren verwendet. Um die Genauigkeit bei der Geradeaus-Fahrt zu ermitteln, wurde ein Programm geschrieben, welches den Roboter so lange geradeaus fahren lässt, bist jeweils einer der beiden Drehgeber 1440 Impulse (zwei Umdrehungen) gesendet hat. Die Geschwindigkeit des Roboters wurde im Programm auf einen konstanten Wert eingestellt. Nachdem die Strecke zurückgelegt war, wurde sie mit einem Maßband vermessen und die Werte dokumentiert. Dieses Messverfahren wurde 40-mal durchgeführt und anschließend ein Mittelwert gebildet. Es ergaben sich folgende Werte für die Wiederholgenauigkeit (Tabelle 6).

Drehgeber links:	2 % (± 20mm)
Drehgeber rechts:	2,5 % (± 25mm)

Tabelle 6: Odometrie - Wiederholgenauigkeit Geradeausfahrt

Für die Ermittlung der Wiederholgenauigkeit bei Drehungen wurde ein ähnliches Prinzip verwendet. Das Testprogramm wurde so adaptiert, dass der Roboter sich um 180° (links oder rechts) dreht. Wie auch schon im vorigen Testverfahren, wurde die Geschwindigkeit des Roboters im Programm mit einem konstanten Wert eingestellt. Die Drehung erfolgt dabei aus dem Stillstand mit definierter Stellung der Castor-Räder. Auch hier wurde die Messung 40-mal wiederholt und ein Mittelwert gebildet. Die Ergebnisse sind in Tabelle 7dargestellt.

Drehung links:	0,56 % (≈ ± 1°)
Drehung rechts:	0,6 % (≈ ± 1°)

Tabelle 7: Odometrie - Wiederholgenauigkeit 180° Drehung

Bei den Tests der Odometrie konnte festgestellt werden, dass die Drehung des Roboters mit höherer Wiederholgenauigkeit möglich ist. Als Gründe dafür sind der geringere Einfluss der Castor-Räder, sowie die geringere Wegstrecke zu nennen.

6.2 Test der Linienverfolgung

Bei den Tests der Linienverfolgung ging es darum, den minimalen Kurvenradius zu bestimmen, welchen der Roboter fahren kann, ohne dass die Linie aus dem Erfassungsbereich verschwindet. Hierzu wurden Linien mit verschiedenen Kurvenradien vorgegeben, welche alle einen Mittelpunktswinkel von 90° aufwiesen. Für die Zu- und Abfahrt zu/von dieser Kurve wurden zwei gerade Linien verwendet. Der Test wurde des Weiteren bei unterschiedlichen Lichtverhältnissen (Raumbeleuchtung an/aus) durchgeführt. Dabei stellte sich heraus, dass dies keinen Einfluss auf den minimalen Kurvenradius hat. Die Geschwindigkeit wurde im Programm vom bereits beschriebenen Regelalgorithmus angepasst. Außerdem wurde der Test sowohl für eine Links- als auch für eine Rechtskurve ausgeführt. Dabei ergab sich ein minimaler Kurvenradius von $r_{min} = 600mm$.

6.3 Test der Ultraschallsensoren

Für das Testen der Ultraschallsensoren wurde kein spezielles Testverfahren angewendet. Es wurden Situationen, welche im Gebrauch des Roboters (Robotics Day, etc.), auftreten könnten, nachgestellt und die Reichweite der Sensoren angepasst. Dabei konnte festgestellt werden, dass aufgrund der geometrischen Anordnung und des Abstrahlwinkels der Ultraschallsensoren ein toter Winkel besteht. Dieser könnte durch das Anbringen weiterer Sensoren verringert, oder sogar komplett eliminiert werden. In diesem Projekt konnte dies allerdings nicht realisiert werden, da am Mirkocontroller keine Ein- bzw. Ausgänge mehr verfügbar sind und dass Integrieren eines zweiten Mikrocontrollers zeitlich nicht möglich war.

6.4 Sicherheitsanalyse

Die durchzuführende Sicherheitsanalyse soll helfen, die Fehler im System zu minimieren und vor Unfällen und fehlerhaften Zwischenfällen schützen. Es gibt in der Praxis verschiedene Ansätze, wie Sicherheitsanalysen durchzuführen sind.
Der Roboter wird, u.U. inmitten von vielen Besuchern, selbstständig mittels Linienverfolgung und Odometrie navigieren. Er nimmt mittels Ultraschallsensoren seine unmittelbare Umgebung wahr, um Kollisionen zu vermeiden. Als zusätzliche Sicherheitsmaßnahme sind vorne und hinten am Roboter Bumper angebracht, welche bei Berührung das ganze System stromlos schalten. Akustische und optische Warnelemente werden Besucher auf das fahrerlose Transportfahrzeug aufmerksam machen. Im Fehlerfall kann jederzeit ein an der Front montierter Not-Aus-Schalter betätigt werden. Es wurde sich bei den installierten Sicherheitsmaßnahmen an die Broschüre „Einsatz von Flurförderfahrzeugen“ von Trabolt (2009) angelehnt.

6.4.1 Analyse des Raumes

Der autonome Transporter ist nur für den Einsatz innerhalb des Vitero-Labors des Studienganges Mechatronik/Robotik an der FH Technikum-Wien vorgesehen. Mögliche raumbedingte Fehlerquellen für die Linienverfolgung sind zum einen die schwankenden Lichtverhältnisse und der unregelmäßige Boden. Außerdem ist zu beachten, dass mehrere Personen vor Ort sein können. Es muss darauf geachtet werden, dass sich keine groben Verunreinigungen auf der Wegstrecke befinden, um die Linienverfolgung nicht zu beschädigen und keine Lesefehler der Phototransistoren auftreten können.

6.4.2 FMEA

Bei der FMEA werden die einzelnen potentiellen Fehler herangezogen und nach den Kriterien Eintrittswahrscheinlichkeit, Schwere des Vorfalls und Möglichkeit der Aufdeckung bewertet. Es werden die einzelnen Punkte mit Zahlen in aufsteigender Form zwischen 1 und 10 bewertet. Anschließend werden die drei Werte der Einstufungen miteinander multipliziert und man erhält die Risikoprioritätszahl (RPZ). Diese Zahl trifft Aussage darüber, welche potenzielle Gefahr von dem jeweiligen Risiko ausgeht. (Biegert, 2003)
In der Tabelle 8 ist ein Ausschnitt der durchgeführten FMEA dargestellt.

Beschreibung des Fehlers	RPZ	Ergriffene Maßnahmen
Kollisionen aller Art während des Fahrbetriebes Es kann zu Personen -und Sachschaden kommen	100	Ultraschallsensoren vorne, Bumper vorne und hinten, Sicherheitsverbau, optische und akustische Warnelemente, robuste Bauart, Kantenschutz, reduzierte Geschwindigkeit, Not-Aus
Personen und Gegenstände von vorne	60	Ultraschallsensoren vorne, Bumper vorne, optische und akustische Warnelemente, robuste Bauart, reduzierte Geschwindigkeit, Not-Aus
Personen und Gegenstände von der Seite	35	Sicherheitsverbau, reduzierte Geschwindigkeit, Kantenschutz
Gefährliche Berührungsspannung	100	Schutzkleinspannung, Isolierungen, Schaltschrank, ordnungsgemäße Kabelverlegung
Personen haben Angst vor der Plattform wegen ruckartigem Bewegens	40	PID-Regler für sanfte Ausregelung, reduzierte Geschwindigkeit
Kurzschließen der Kontakte an der Autobatterie im Reparaturmodus	70	Abdeckung der Pole, Laienverbot durch die Richtlinien der FH
Überlast und Kurzschluss und somit Schäden an der Elektronik, bzw. Brandgefahr	64	Ordnungsgemäße Verkabelung, verschlossener Schaltschrank, Schmelzsicherung
Fehler durch leichtsinnigen Umgang von Laien mit der Plattform	40	Akustische und optische Warnelemente die Achtung signalisieren sollen

Tabelle 8: FMEA

Bei dieser Art der Sicherheitsanalyse kommt es sehr stark auf die Intensität der Testphase am Ende des Projektes, beziehungsweise auf die Erfahrung der Ingenieure an, die durch ihr Fachwissen bereits übliche Sicherheitsvorkehrungen implementieren und bewerten können. (Biegert, 2003)

Wie man in der Tabelle erkennen kann, steht in der Sicherheitsanalyse der Personenschutz ganz klar vor der Geräte- und Betriebssicherheit. Es muss, unter Betrachtnahme der Kosten für die implementierten Sicherheitsmaßnahmen, gewährleistet werden, dass möglichst keine Personen zu Schaden kommen. Je höher die RPZ des jeweiligen Fehlerfalles ist, desto höher prior sind die Maßnahmen, die gesetzt werden müssen. Die höchsten Risikoeinstufungen haben in der Analyse der Schutz vor Kollisionen und der Schutz vor gefährlicher Spannung. Durch die angeführten Sicherheitsmaßnahmen werden diese Fehlerpotenziale auf ein Minimum reduziert. Die vollständige FMEA befindet sich im Anhang A.

6.4.3 FTA -Baumdiagramm

Die FTA, also die Fehlerbaumanalyse, geht im Gegensatz zur FMEA nicht von den Fehlern aus die passieren können, sondern von den Ereignissen, die unter Kombination eine Ursache bilden. Sie ist in die umgekehrte Richtung zu lesen, wie das vorher erklärte FMEA-Verfahren. Der Vorteil dieser Analyse ist, dass hier logische Operatoren den Zusammenhang von verschiedenen Ereignissen, die zu einem Fehler führen können, graphisch aufbereitet werden. Der Baum ist leicht zu lesen und bietet daher eine übersichtliche Darstellung kritischer Faktoren des Gesamtsystems. (Biegert, 2003)

Für die Symbolik wurde sich bei der Zeichnung an die Dissertation von Biegert (2003) angehalten. In den folgenden Abbildungen Abbildung 24 und Abbildung 25 wird ein Ausschnitt der FTA dargestellt. Für die ausführliche Fehlerbaumanalyse darf auf den Anhang verwiesen werden.

In der ersten Grafik kann man die Ursachen erkennen, die dazu führen, dass Personen von den Warnsystemen nicht in Aufmerksamkeit versetzt werden. Versagen der Schallgeber und die Warnleuchte aufgrund von Schäden am Bauteil oder Kontaktfehlern, so werden Personen nicht auf die herannahende Gefahr aufmerksam. Es ist auch zu bedenken, dass Personen, die mit dem Rücken zum Roboter stehen, diesen dann nicht wahrnehmen können.

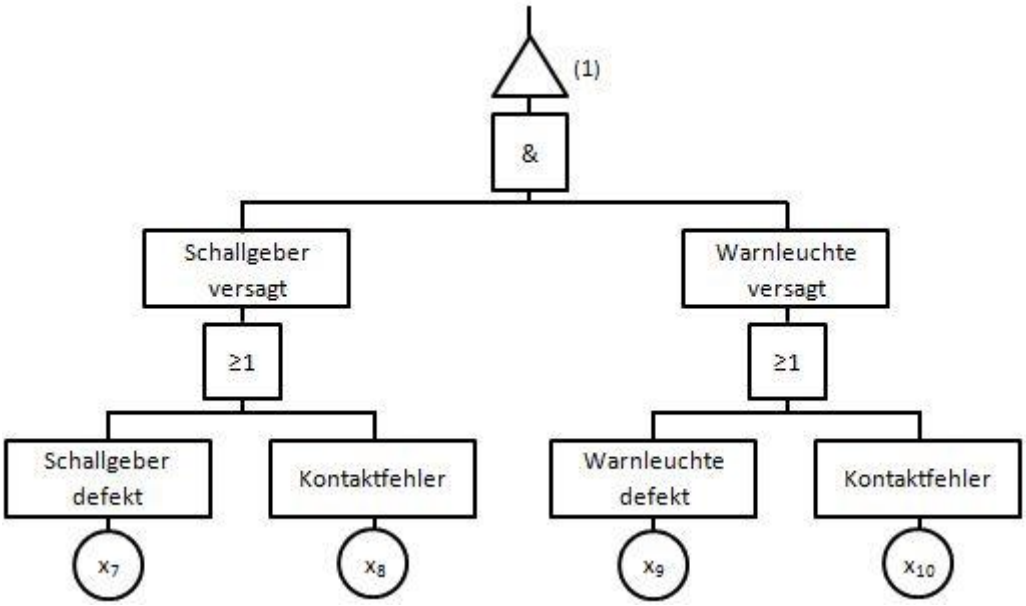


Abbildung 24: Fehlerbaum Ausschnitt 1

Sind die Personen nun nicht ausreichend gewarnt, kann es dazu kommen, dass sie von der Seite, bei nicht installiertem Sicherheitsverbau, die sich bewegenden Räder berühren und sich somit verletzen. Ist der Not-Aus Schalter nun aus diversen Gründen auch nicht aktiv, so steigt die Gefahr um ein Vielfaches, da das Gerät nicht mehr kontrolliert zu stoppen ist. Das Ereignis an der Spitze des Baumabschnittes wäre hier erreicht.

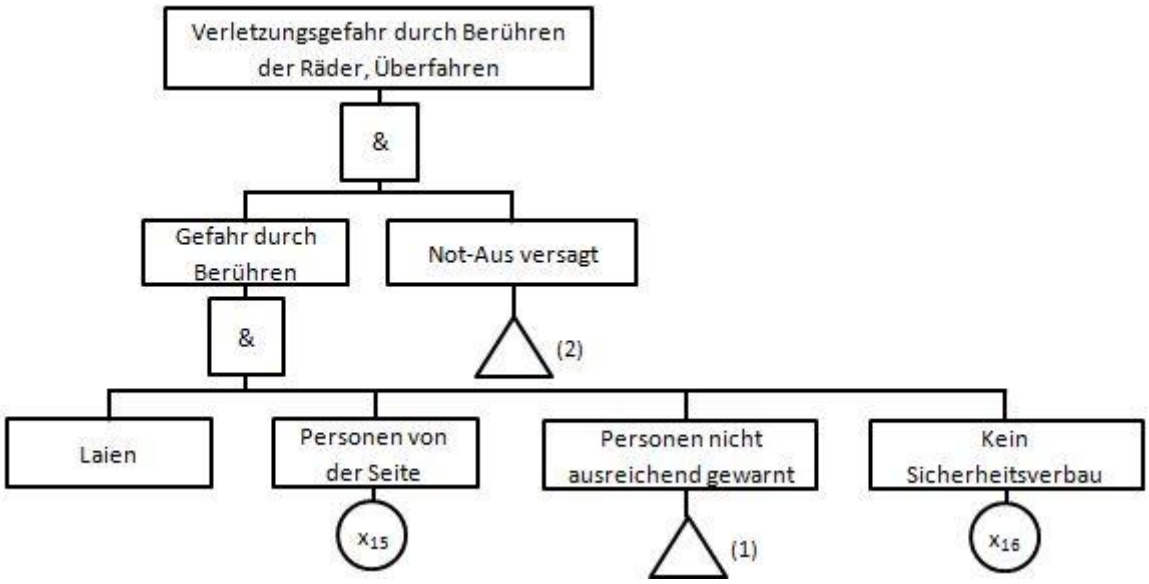


Abbildung 25: Fehlerbaum Ausschnitt 2

Conclusio der Ereigniskette ist nun, dass der Sicherheitsverbau installiert werden muss, um die von der Seite der Plattform ausgehende Gefahr zu bannen.

6.5 Evaluierung der realisierten Navigationsroutinen

Nach Abschluss der beschriebenen Tests und dem Einsatz am Robotics Day 2011 können die implementierten Navigationsroutinen als voll funktionsfähig betrachtet werden. Die Linienverfolgung ist wider Erwarten sehr unempfindlich was verschiedenen Lichtverhältnisse anbelangt. Und auch kleine Unterbrechungen (<10mm) der Linie haben keinen Einfluss auf die korrekte Funktionsweise. Die Odometrie ist für die verwendeten Fahrmanöver (Drehung um 180°) ausreichend genau.

7 Zusammenfassung und Ausblick

Durch die Erweiterung der mobilen Plattform zu einem autonomen Transporter können nun Becher und Lebensmittel entlang einer vorher gelegten Route mittels Linienverfolgung transportiert werden. Die Plattform kann ebenfalls mittels Odometrie mit einer Wiederholgenauigkeit von einem Grad pro 180° Drehung navigieren. Studenten zukünftiger Semester haben nun die Möglichkeit, das Zusammenspiel von Linienverfolgung und Odometrie auszuweiten.

Um zusätzliche Navigationsroutinen implementieren zu können, müsste ein größerer Microkontroller oder ein geeigneter Computer angekauft werden, da sowohl alle Ein –und Ausgänge, als auch alle Timer und Inerrupts des RN-Control-Boards bereits belegt sind.

Die Plattform ist mittlerweile zu einer sowohl betriebs-, als auch personensicheren Anlage weiterentwickelt worden und könnte jederzeit eingesetzt werden. Es wäre eine mögliche Aufgabe für spätere Semester, ein anderes Konzept für den Einsatz des MULE-Roboters zu entwickeln, bei der ein neuer Aufbau für andere Zwecke entworfen wird. Durch die maximale Traglast von 80kg bleibt hier viel Spielraum, wie die Plattform modifiziert werden könnte.

Das geschriebene Programm ist mit Kommentaren zum allgemeinen Verständnis versehen und kann somit leicht erweitert werden. Hier wäre es möglich, eventuell zusätzlich eingebaute Sensoren zur erweiterten Umgebungswahrnehmung in das Programm einzubinden.

Der sich auf der Plattform befindliche Schaltschrank bietet nur noch wenig Platz für weitere Schaltplatinen, jedoch kann er durch den Austausch der vertikalen Profile von Robotunits leicht adaptiert und somit vergrößert werden.

Bei voll aufgeladenen Batterien kann, aufgrund der Erfahrung am Robotics Day 2011, kann der autonome Transporter einen ganzen Tag eingesetzt werden, ohne ersichtliche Leistungseinbußen zu zeigen.

Eine mögliche Erweiterung wäre ebenfalls, die mobile Roboterplattform für den Einsatz außerhalb des Labors aufzubereiten und mit Solarzellen auszustatten, um Energieautonom fahren zu können.

Abschließend ist zu sagen dass das durchgeführte Projekt eine Vielzahl von interessanten Aufgaben bot und so viele neue Erfahrungen ermöglichte.

Literaturverzeichnis

Biegert, U., 2003. *Ganzheitliche modellbasierte Sicherheitsanalyse von Prozessautomatisierungssystemen*. Dissertation, (Dr. Ing.). Universität Stuttgart

Flegel, G., Birnstiel, K., Nerreter, W., 2009. *Elektrotechnik für Maschinenbau und Mechatronik*. 9 Auflage. München: Carl Hanser Verlag.

Nikon Metrology GmbH, 2011. *iGPS*. Verfügbar von: http://de.nikonmetrology.com/large_volume_tracking__positioning/igps/ [Aufruf: 28 April 2011].

Reuter, M. und Zacher, S., 2004. *Regelungstechnik für Ingenieure Analyse, Simulation und Entwurf von Regelkreisen.* 11 Auflage. Wiesbaden: Friedr. Vieweg & Sohn Verlag.

Smith, W., 2008.*Modern Optical Engineering.*4 Auflage. New York: McGraw-Hill Inc.

Trabolt, R.J., 2009. *Einsatz von Flurförderfahrzeugen.* 2.Auflage. Bonn, Verfügbar von: http://www.bghw.de/medienangebot/sparte-grosshandel-und-lagerei/spezial/spezial-pdf-dateien/SP_04.pdf [Abgerufen am 9. Mai 2011].

Ullrich, G., 2011. *Fahrerlose Transportsysteme*. Wiesbaden: Vieweg+Teubner Verlag.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

µC	Mikrocontroller
AD	Analog-Digital
ADC	Analog-Digital-Converter
DC	Direct Current
DG	Drehgeber
FH	Fachhochschule
FMEA	Failure Mode and Effect Analysis
FTA	Failure tree analysis
GND	Ground
GNSS	Global Navigation Satellite System
h. l.	Hinten links
h. r.	Hinten rechts
IR	Infrarot
ISR	Interrupt-Service-Routine
JTAG	Joint Test Action Group
LED	Light emitting diode
LKW	Lastkraftwagen
n. c.	Not connected
PT	Phototransistor
PWM	Pulsweitenmodulation
RPZ	Risikoprioritätszahl
v. l.	Vorne links
v. m.	Vorne mitte
v. r.	Vorne rechts
V_{CC}	Common-Collector-Voltage

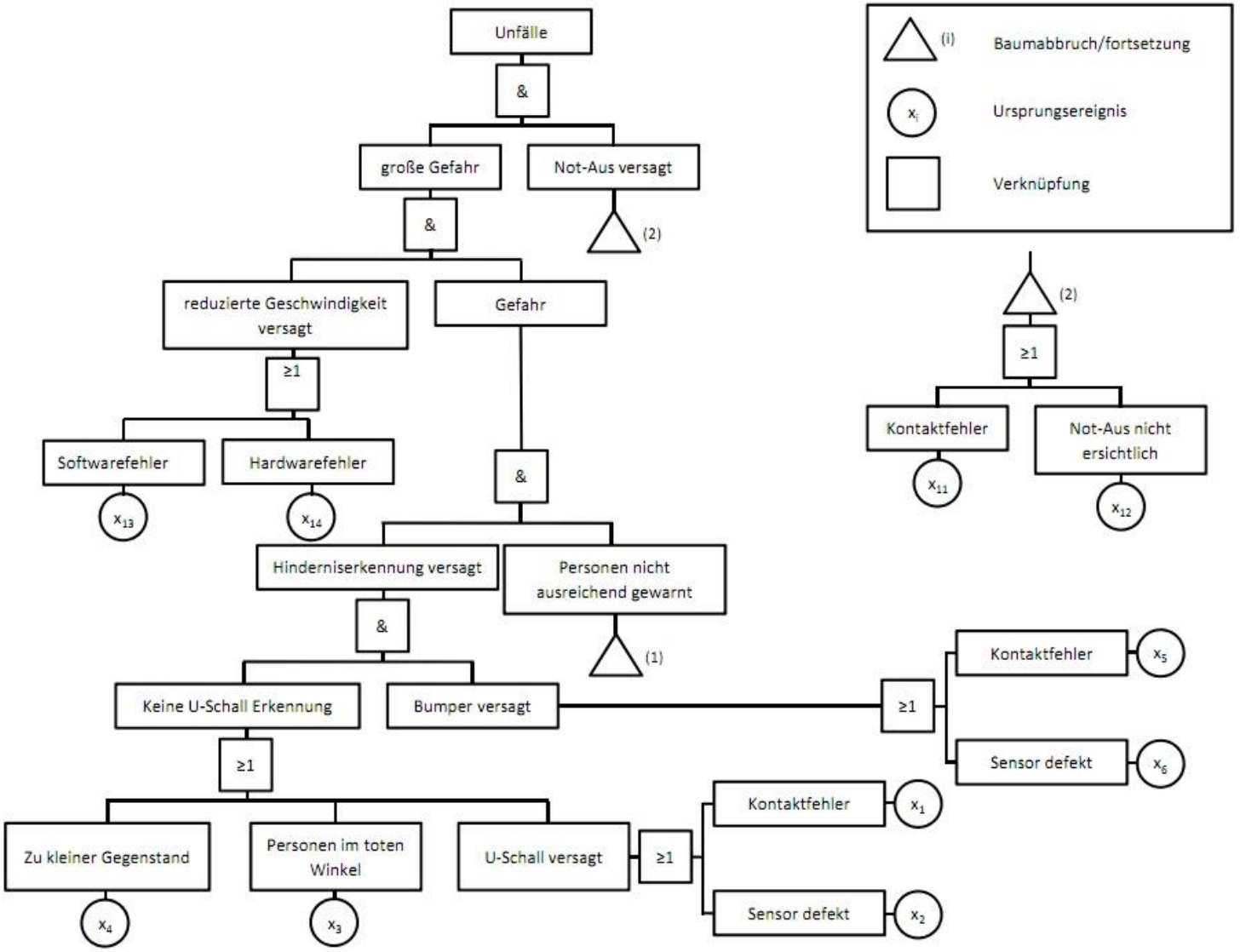
Autorenverzeichnis

Anhang A: FMEA

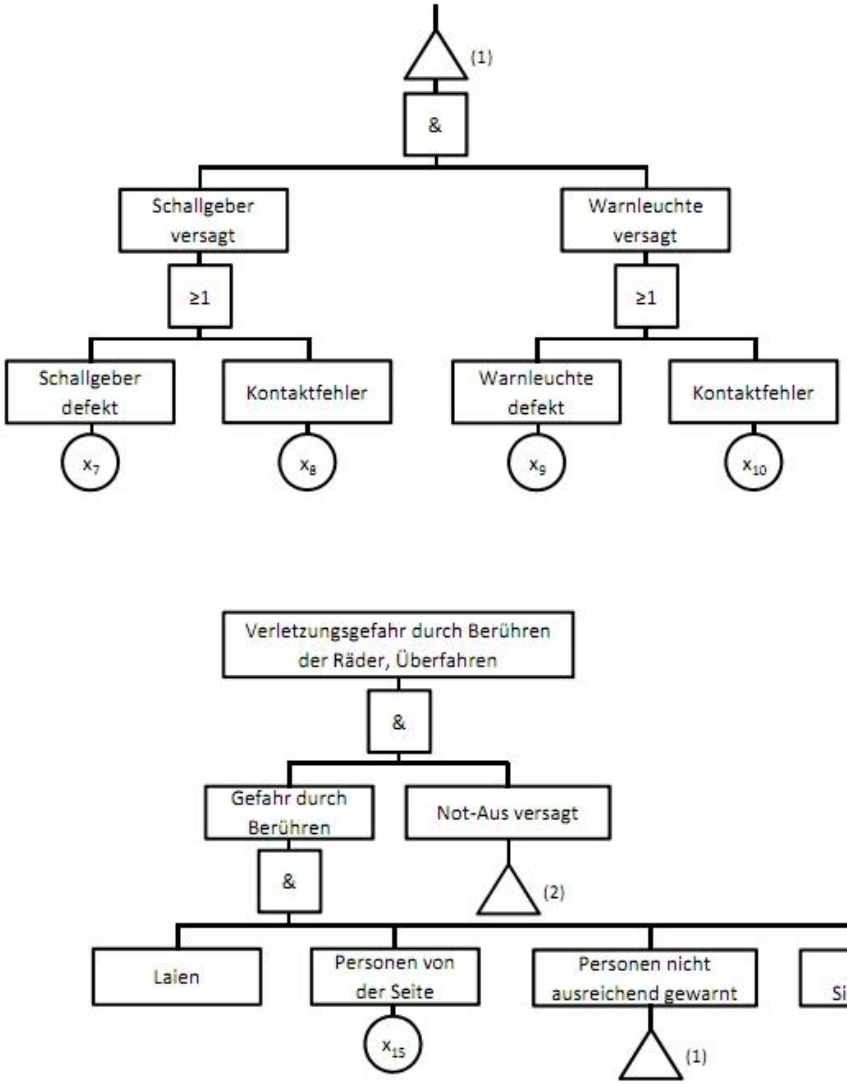
Betrieb bzw. Einatz	Fehlerbeschreibung	Auswirkungen	Grund für den Fehler	EW	S/G	AW	RPZ	Maßnahme dagegen
Fahrbetrieb	Sicherheismaßnahmen daektiviert	Personen und Geräteschaden	Die mobile Plattform erkennt keine Hindernisse und die Sicherheitsvorrichtungen sind nicht verfügbar	10	10	1	100	Ultraschallsensoren, Bumper, Not-Aus-Schalter, Warnleuchte, Bieper, Leistungsbegrenzung, keine scharfen Kanten, reduzierte Geschwindigkeit, robuste Bauart
Fahrbetrieb nach vorne	Personen oder Gegenstände im toten Winkel der Ultraschallsensoren	Potenzieller Schaden an Personen, Gegenständen oder am Gerät	Die drei Sensoren auf der Vorderseite decken nicht den gesamten Raum ab	3	10	2	60	Bumper, Not-Aus-Schalter, Warnleuchte, Bieper, Leistungsbegrenzung, keine scharfen Kanten, reduzierte Geschwindigkeit
Fahrbetrieb	Personen oder Gegenstände von der Seite bei nicht vorhandenem Schutz	Personen können unter die Antriebsräder kommen oder diese berühren	Auf der Seite sind weder Ultraschallsensoren noch Bumper installiert	5	7	1	35	mechanischer Sicherheitsverbau und somit Abdeckung der gesamten Seite, reduzierte Geschwindigkeit, Kantenschutz
Fahr- und Standbetrieb	Notsituationen wo die Sicherheitsmaßnahmen versagen, bzw. noch nicht aktiviert wurden	Personen und Geräte können zu Schaden kommen	Alle Notsituationen	1	10	5	50	Not-Aus-Schalter
Fahrbetrieb	gefählicher Fahrbetrieb	Personen haben Angst vor der Plattform	Zu schneller und ruckartiger Fahrbetrieb	10	4	1	40	reduzierte Geschwindigkeit, PID Regler für sanfte Ausregelung
Fahr- und Standbetrieb	gefährliche, spannungsführende Teile berührbar	Personen könnten spannungsführende Teile berühren und zu Schaden kommen	Spannungsführende Teile werden berührt	10	10	1	100	Isolierungen, Schaltschrank, Schutzkleinspannung, Autobatterien verstaut und die Kontakte abgedeckt
Fahrbetrieb	Tiere im Arbeitsbereich	Geräte -und Sachschaden	Die autonome Plattform ist nur auf den Umgang mit Menschen ausgelegt	2	5	3	30	Indoorbereich als Einsatzort definiert
Fahrbetrieb	Personen sind nicht vorsichtig genug	Es kann zu Unfällen durch leichtsinn kommen	Laien	4	10	1	40	Akustische und optische Warnelemente, Aufsichtspflicht während der Vorführung
Fahrbetrieb	Die Linie wird verloren und der definierte Fahrbereich verlassen	Kollisionen	minimaler Kurvenradius unterschritten, Fehler an der Linie oder an den Transistoren, Softwarefehler	9	10	1	90	Softwareimplementierter Halt bei Verlieren der Linie
Reperatur	Kurzschließen der Autobatterie	Personenschaden und Sachschaden	Unachtsamkeit bzw. Fehler im Umgang mit der Batterie	7	10	1	70	Abdeckung der Pole, Laienverbot durch die Richtlinien der FH
Fahrbetrieb	Fehler der Sensoren	Personen -und Sachschaden	akustische Anomalien, Schaden an den Sensoren, Helligkeitsunterschiede	2	7	5	70	Not-Aus, in der Software implamentierter Halt
Fahr -und Standbetrieb	Überlast oder Kurzschluss	Geräteschaden	Kontaktfehler	2	8	4	64	ordnungsgemäße Verkabelung, Schmelzsicherung, verschlossener Schaltschrank

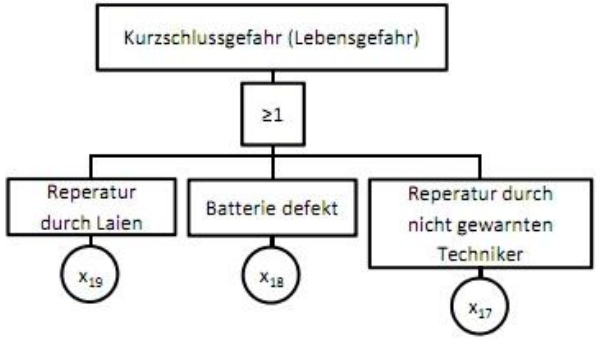
EW	**Eintrittswahrscheinlichkeit**
S/G	**Schwere/Gewichtung**
AW	**Aufdeckungswahrscheinlichkeit**

Anhang B: FTA 1

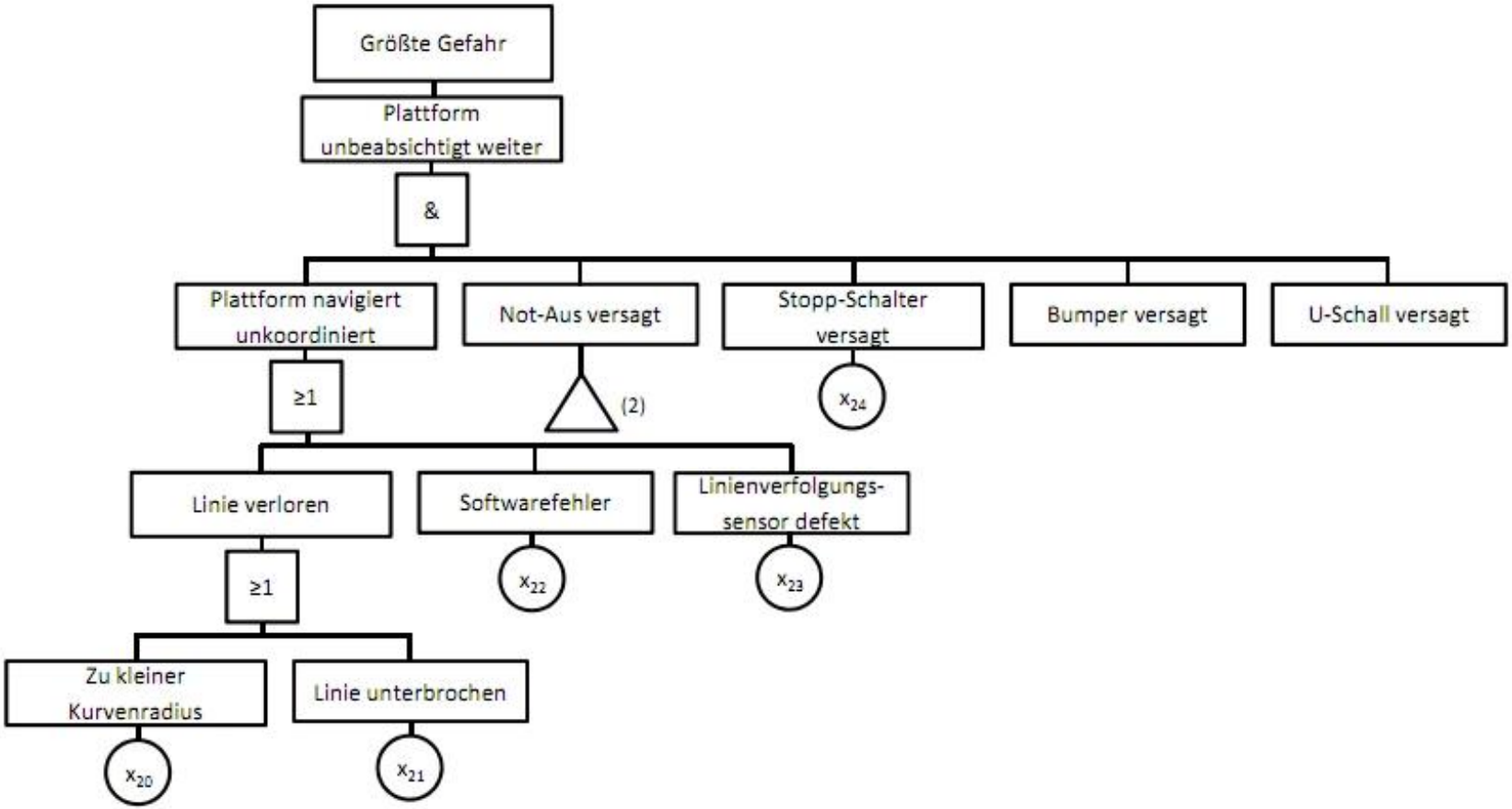


Anhang C: FTA 2

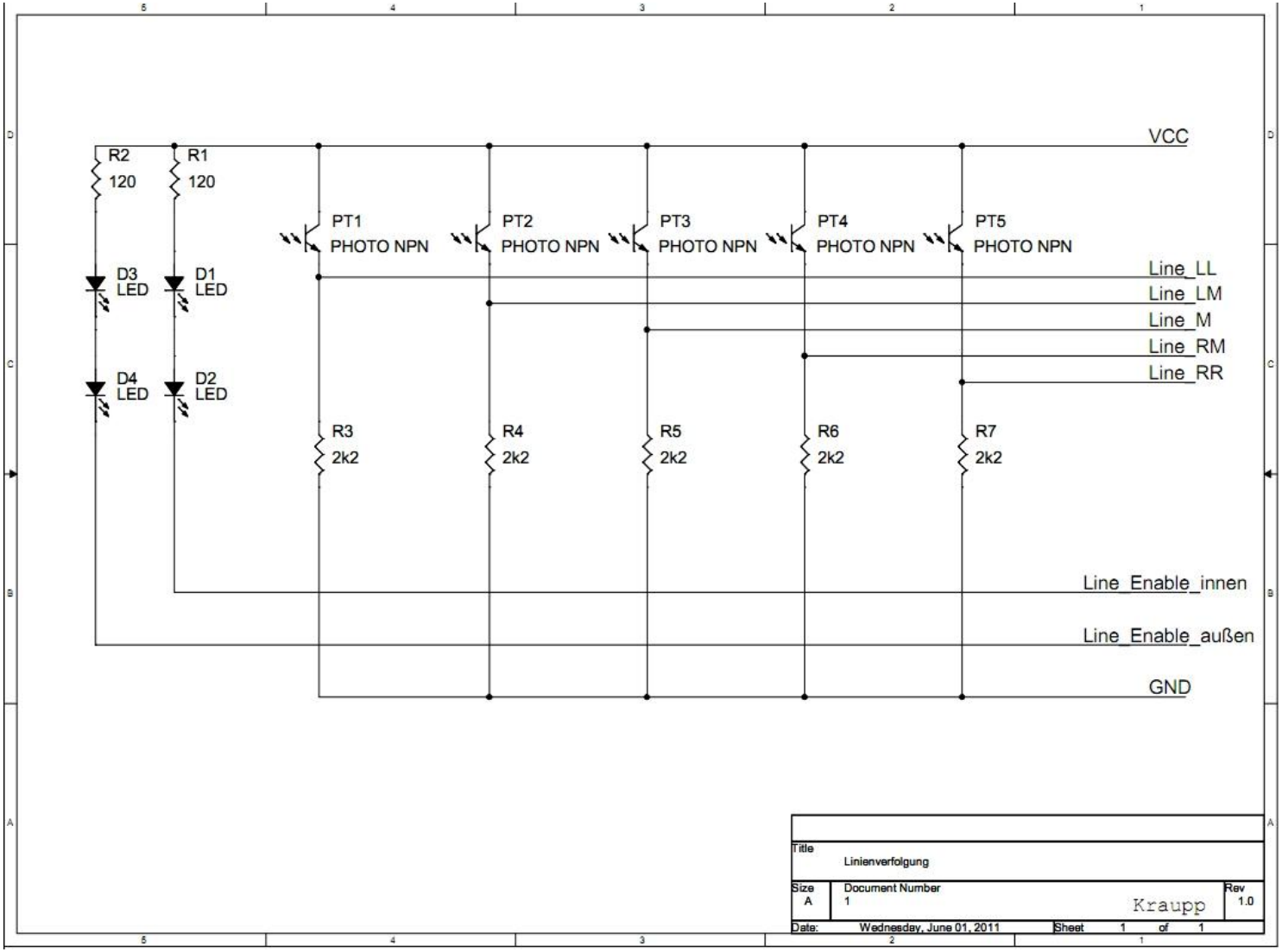


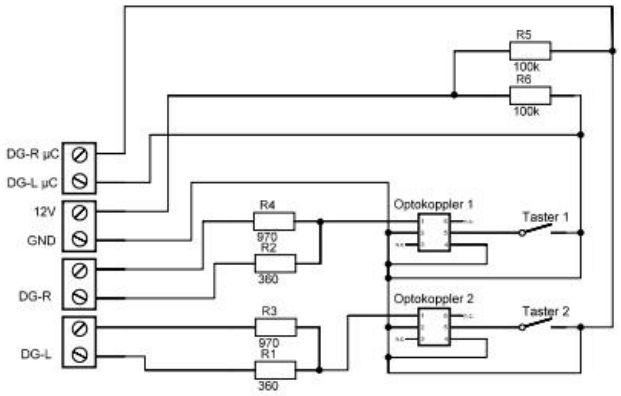


Anhang D: FTA 3

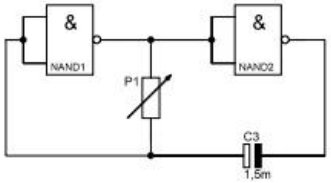


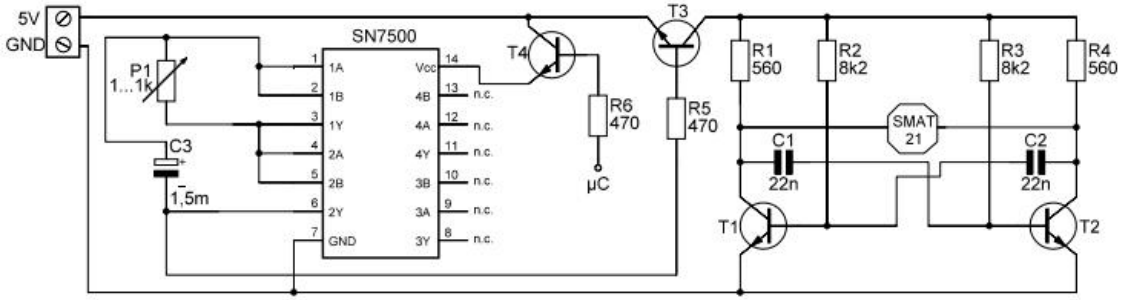
Anhang E: Elektrische Schaltungen





Titel:	Odometrie Interface
Ersteller:	Michael Kraupp
Datum:	22.04.2011





Titel:	Piezo Schallgeber Elektronik
Ersteller:	Stefan Kautz
Datum:	26.04.2011

Anhang F: Source code

```
#define F_CPU 16000000L
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include <avr/eeprom.h>

#include <util/delay.h>
#include <stdio.h>
#include <inttypes.h>

int count;
int entfernung;
int tastzahl = 0;

// Variablen für das Auslesen der Phototransistoren
volatile int frontleft;
volatile int frontmiddle;
volatile int frontright;
volatile int frontlefta;
volatile int frontrighta;

//Minimum Geschwindigkeit
#define GESCHW 5
//Maximum Geschwindigkeit
#define GESCHWMAX 40

//Ultraschall-Parameter
#define SENSOK 7000
#define SENSKURZ 5000
#define SENSDREH 3800
#define SENSTASTH 3800
int uschalltol;

//Odometrie-Parameter
#define RADDURCHMESSER 34.5
#define TIKSPROUMDREHUNG 720
#define ABSTANDRADZUMITTE 36
#define KORREKTURODOR 1.241
#define KORREKTURODOL 1.286

int halt();
int fahrevor();
int fahrerueck();
int rdrehen();
int ldrehen();
void fehler();
void uschall();

//Linienverfolgung - Parameter
volatile int odo_links=0;
volatile int odo_links_ovf=0;
volatile int odo_rechts=0;
volatile int odo_rechts_ovf=0;
//Odometrie - Drehrichtung (0=positiv 1=negativ)
volatile int vzl;
volatile int vzr;
```

```
double phi;

volatile uint16_t average_frontleft_B=187;
volatile uint16_t average_frontmiddle_B=255;
volatile uint16_t average_frontright_B=169;
volatile uint16_t average_frontlefta_B=88;
volatile uint16_t average_frontrighta_B=76;
volatile uint16_t average_frontleft_S=67;
volatile uint16_t average_frontmiddle_S=199;
volatile uint16_t average_frontright_S=63;
volatile uint16_t average_frontlefta_S=32;
volatile uint16_t average_frontrighta_S=28;

//PID Regler
int eblalt=0;
int ebralt=0;
int eblaalt;
int ebraalt;
int abweichung=0;
int vsollwert=0;
int eb=0;
int aktvl;
int aktvr;
int zl;
int zr;
int zal;
int zar;
double normieren=0;

int init(){
      DDRB =      0b00101110;
      PORTB = 0b00001111;
      DDRD =  0b11100000;
      PORTD = 0b11111100;
      DDRC =  0b00111100;
      PORTC = 0b00000011;
      return 0;
}

void Linienverfolgung_init(void){

/* ADC initialisieren */
  uint16_t result;

  ADMUX = (0<<REFS1) | (1<<REFS0);       // AVcc als Referenz benutzen
  ADCSRA = (1<<ADPS1) | (1<<ADPS0);      // Frequenzvorteiler
  ADCSRA |= (1<<ADEN);                   // ADC aktivieren

  /* nach Aktivieren des ADC wird ein "Dummy-Readout" empfohlen, man liest
     also einen Wert und verwirft diesen, um den ADC "warmlaufen zu lassen"
      */

  ADCSRA |= (1<<ADSC);                  // eine ADC-Wandlung
  while (ADCSRA & (1<<ADSC) ) {}   //auf Abschluss der Konvertierung warten
  /* ADCW muss einmal gelesen werden, sonst wird Ergebnis der nächsten
     Wandlung nicht übernommen. */
  result = ADCW;

}
```

```
/* ADC Einzelmessung */
uint16_t ADC_Read( uint8_t channel )
{
  // Kanal wählen, ohne andere Bits zu beeinflussen
  ADMUX = (ADMUX & ~(0x1F)) | (channel & 0x1F);
  ADCSRA |= (1<<ADSC);            // eine Wandlung "single conversion"
  while (ADCSRA & (1<<ADSC) ) {}  // auf Abschluss der Konvertierung warten
  return ADCW;                    // ADC auslesen und zurückgeben
}

/*ADC Mehrfachmessung mit Durchschnittsbildung*/
uint16_t ADC_Read_Avg( uint8_t channel, uint8_t average )
{
  uint32_t result = 0;

  for (uint8_t i = 0; i < average; ++i )
    result += ADC_Read( channel );

  return (uint16_t)( result / average );
}



int Linie_lesen(){

      //Toleranzvariable gibt den Unterschied an bei dem zwischen dunkel
      und hell unterschieden wird
      #define TOLERANCE 450

      int ergebnis=0;

      //alte Regelabweichung speichern
      eblalt = 1023 - frontleft;
      ebralt = 1023 - frontright;
      eblaalt = 1023 - frontlefta;
      ebraalt = 1023 - frontrighta;

      // Messung der Linie erneut durchführen
      //Anschließend abziehen des Umgebungslichtes
      frontleft=ADC_Read_Avg(7, 4); // Kanal 1, Mittelwert aus 4 Messungen
      frontmiddle=ADC_Read_Avg(6, 4);// Kanal 2, Mittelwert aus 4 Messungen
      frontright=ADC_Read_Avg(5, 4);// Kanal 3, Mittelwert aus 4 Messungen
      frontlefta=ADC_Read_Avg(4, 4);// Kanal 1, Mittelwert aus 4 Messungen
      frontrighta=ADC_Read_Avg(3, 4);// Kanal 2, Mittelwert aus 4 Messungen

      //Jetzt Werte normalisieren (Spreizung auf 0-1024)
            if ((frontleft-average_frontleft_S)<0){
                  frontleft=0;
            }else{
                  frontleft=frontleft-average_frontleft_S;
            }
            frontleft=frontleft*(1024/(average_frontleft_B -
            average_frontleft_S));


            if ((frontmiddle-average_frontmiddle_S)<0){
                  frontmiddle=0;
            }else{
```

```
            frontmiddle=frontmiddle-average_frontmiddle_S;
      }
      frontmiddle=frontmiddle*(1024/(average_frontmiddle_B -
      average_frontmiddle_S));


      if ((frontright-average_frontright_S)<0){
            frontright=0;
      }else{
            frontright=frontright-average_frontright_S;
      }
      frontright=frontright*(1024/(average_frontright_B -
      average_frontright_S));

      if ((frontlefta-average_frontlefta_S)<0){
            frontlefta=0;
      }else{
            frontlefta=frontlefta-average_frontlefta_S;
      }
      frontlefta=frontlefta*(1024/(average_frontlefta_B -
      average_frontlefta_S));

      if ((frontrighta-average_frontrighta_S)<0){
            frontrighta=0;
      }else{
            frontrighta=frontrighta-average_frontrighta_S;
      }
      frontrighta=frontrighta*(1024/(average_frontrighta_B -
      average_frontrighta_S));


// Toleranz berücksichtigen
if (frontleft < TOLERANCE){
      frontleft=0;
}
if (frontmiddle < TOLERANCE){
      frontmiddle = 0;
}
if (frontright < TOLERANCE){
      frontright = 0;
}
if (frontlefta < TOLERANCE){
      frontlefta = 0;
}
if (frontrighta < TOLERANCE){
      frontrighta = 0;
}


//Ergebnis der Linienlesung zurückgeben
// nur mitte links
if (frontleft==0 && frontright != 0 && frontmiddle != 0 && frontlefta
!=0 &&      frontrighta !=0) ergebnis=20;
// nur mitte rechts
if (frontright==0 && frontmiddle !=0 && frontleft !=0 && frontrighta
!=0 &&      frontlefta !=0) ergebnis=40;
// nur mitte
if (frontmiddle==0 && frontleft !=0 && frontright !=0 && frontlefta
!=0 &&      frontrighta !=0) ergebnis=30;
```

```
	// alle zusammen
	if (frontright==0 && frontleft ==0 && frontmiddle==0 && frontlefta==0
	&&    frontrighta==0) ergebnis=0;
	// nur aussen links
	if (frontlefta==0 && frontmiddle != 0 && frontleft != 0 && frontright
	!= 0 &&     frontrighta != 0) ergebnis=10;
	// nur aussen rechts
	if (frontrighta==0 && frontlefta != 0 && frontmiddle != 0 &&
	frontright !=0    && frontleft != 0) ergebnis=50;
	// rechts aussen und mitte rechts
	if (frontrighta==0 && frontright==0 && frontmiddle !=0 && frontleft
	!= 0 &&     frontlefta != 0) ergebnis=50;
	// links aussen und mittel links
	if (frontlefta==0 && frontleft==0 && frontmiddle !=0 && frontright !=
	0 &&  frontrighta !=0) ergebnis=10;
	// mitte und mitte links
	if (frontmiddle==0 && frontleft==0 && frontright !=0 && frontrighta
	!= 0 &&     frontlefta != 0) ergebnis=20;
	// mitte und mitte rechts
	if (frontmiddle==0 && frontright==0 && frontrighta !=0 && frontleft
	!= 0 &&     frontlefta != 0) ergebnis=40;

	// links aussen und mittel links und mitte
	if (frontlefta==0 && frontleft==0 && frontmiddle ==0 && frontright !=
	0 &&  frontrighta !=0) ergebnis=10;
	      // rechts aussen und mitte rechts und mitte
	if (frontrighta==0 && frontright==0 && frontmiddle ==0 && frontleft
	!= 0 &&     frontlefta != 0) ergebnis=50;

	return ergebnis;
}

int dvrs1(){
	DDRB |= (1<<PB0);                    //Den Pin B0 als Ausgang setzen
	double i = 32/10000;
	_delay_ms(i);                        //3,2 us warten
	DDRB &= ~(1<<PB0);                   //Den Pin B0 als Eingang setzten
	return 0;
}

int dvrs2(){
	DDRD |= (1<<PD4);
	double i = 32/10000;
	_delay_ms(i);
	DDRD &= ~(1<<PD4);
	return 0;
}

int pwminit1(){

	TCCR0 |= (1<<CS02);           //Setzen des Prescaler auf den Wert 256
	TCCR0 |= (1<<WGM00);          //PWM auf Fast-PWM-Mode
	TCCR0 |= (1<<WGM01);
	TCCR0 |= (1<<COM01)|(1<<COM00);     //Setzen auf invertierten Modus
	OCR0 = GESCHWMAX;                    //Setzen des Compare-Wertes auf 40
	DDRB |= (1<<PB3);                    //Aktivierung des Output-Pin
	return 0;
}
```

```
int pwminit2(){
	TCCR2 |= (1<<CS22);			//Setzen des Prescaler auf den Wert 256
	TCCR2 |= (1<<WGM20);			//PWM auf Fast-PWM-Mode
	TCCR2 |= (1<<WGM21);
	TCCR2 |= (1<<COM21)|(1<<COM20);		//Setzen auf invertierten Modus
	OCR2 = GESCHWMAX;			//Setzen des Compare-Wertes auf 40
	DDRD |= (1<<PD7);			//Aktivierung des Output-Pin
	return 0;
}

int sigsens1a4(){ // Signal Sensor vorne links und hinten rechts
	PORTB |= (1<<PINB4);			//Ausgang Pin B4 auf 1 setzen
	_delay_us(12);				//12us warten
	PORTB &= ~(1<<PINB4);			//Ausgang Pin B4 auf 0 setzen
	return 0;
}

int sigsens2(){   // Signal Sensor vorne mitte
	PORTB |= (1<<PINB5);
	_delay_us(12);
	PORTB &= ~(1<<PINB5);
	return 0;
}

int sigsens3a5(){ // Signal Sensor vorne rechts und hinten links
	PORTB |= (1<<PINB6);
	_delay_us(12);
	PORTB &= ~(1<<PINB6);
	return 0;
}

int echosens1(){ // Echo von Sensor vorne links
	TCCR1A = 0;			//Resetten der Timer-Bits
	TCCR1B = 0;
	TCNT1 = 0;
	count = 0;			//Resetten der Referenzvariable
	int exitbed=0;
	while(!(PIND & (1<<PD1))){
		exitbed++;
		_delay_us(1);
		if (exitbed>5000) break;
	}
	//Warten auf Signal am Echo-Pin des Sensors

	TCCR1B = (1<<CS11);			//Starten des 16-Bit-Timers
	while((PIND & (1<<PD1))){
		if (TCNT1 > (SENSOK*2)) break;
	}
	//Warten bis Ende des Signals des Sensors

	TCCR1B = 0;		//Stoppen des Timers
	count = TCNT1; //Überschreiben des Timerstandes auf Referenzvariable
	return 0;
}

int echosens2(){ // Echo von Sensor vorne mitte
	TCCR1A = 0;
	TCCR1B = 0;
	TCNT1 = 0;
```

```
	count = 0;
	int exitbed=0;
	while(!(PINC & (1<<PC1))){
		exitbed++;
		_delay_us(1);
		if (exitbed>5000) break;
	}
	TCCR1B = (1<<CS11);
	while((PINC & (1<<PC1))){
		if (TCNT1 > (SENSOK*2)) break;
	}
	TCCR1B = 0;
	count = TCNT1;
	return 0;
}

int echosens3(){ // Echo von Sensor vorne rechts
	TCCR1A = 0;
	TCCR1B = 0;
	TCNT1 = 0;
	count = 0;
	int exitbed=0;
	while(!(PINC & (1<<PC0))){
		exitbed++;
		_delay_us(1);
		if (exitbed>5000) break;
	}
	TCCR1B = (1<<CS11);
	while((PINC & (1<<PC0))){
		if (TCNT1 > (SENSOK*2)) break;
	}
	TCCR1B = 0;
	count = TCNT1;
	return  0;
}

int echosens4(){ // Echo von Sensor hinten rechts
	TCCR1A = 0;
	TCCR1B = 0;
	TCNT1 = 0;
	count = 0;
	int exitbed=0;
	while(!(PIND & (1<<PD0))){
		exitbed++;
		_delay_us(1);
		if (exitbed>5000) break;
	}
	TCCR1B = (1<<CS11);
	while((PIND & (1<<PD0))){
		if (TCNT1 > (SENSOK*2)) break;
	}
	TCCR1B = 0;
	count = TCNT1;
	return 0;
}

int echosens5(){ // Echo von Sensor hinten links
	TCCR1A = 0;
	TCCR1B = 0;
```

```
	TCNT1 = 0;
	count = 0;
	int exitbed=0;
	while(!(PINB & (1<<PB7))){
		exitbed++;
		_delay_us(1);
		if (exitbed>5000) break;
	}
	TCCR1B = (1<<CS11);
	while((PINB & (1<<PB7))){
		if (TCNT1 > (SENSOK*2)) break;
	}
	TCCR1B = 0;
	count = TCNT1;
	return 0;
}

int auswsig(unsigned int a){
	int entfernung = (172*(a*(1/16000000))*100);
	return entfernung;
}

int fahrenvor(){
	vzl=0;
	vzr=0;
	PORTB |= (1<<PB1);			//Setzen des Pin B1 auf logisch 1
	PORTD |= (1<<PD6);			//Setzen des Pin D6 auf logisch 1
	PORTB &= ~(1<<PB2);			//Setzen des Pin B2 auf logisch 0
	PORTD &= ~(1<<PD5);			//Setzen des Pin D5 auf logisch 0
	return 0;
}

int fahrenrueck(){
	vzl=1;
	vzr=1;
	PORTB |= (1<<PB2);			//Setzen des Pin B2 auf logisch 1
	PORTD |= (1<<PD5);			//Setzen des Pin D5 auf logisch 1
	PORTB &= ~(1<<PB1);			//Setzen des Pin B1 auf logisch 0
	PORTD &= ~(1<<PD6);			//Setzen des Pin D6 auf logisch 0
	return 0;
}

int rdrehen(){
	vzl = 0;
	vzr = 1;
	PORTB |= (1<<PB2);			//Setzen des Pin B2 auf logisch 1
	PORTD |= (1<<PD6);			//Setzen des Pin D6 auf logisch 1
	PORTD &= ~(1<<PD5);			//Setzen des Pin D5 auf logisch 0
	PORTB &= ~(1<<PB1);			//Setzen des Pin B1 auf logisch 0
	return 0;
}

int rdrehenodo(int grad){
	double uebersetzung;
	double gradinrad;
	odo_rechts=0;
	odo_links=0;
	//Korrekturfaktor odometrie
	grad = grad / KORREKTURODOR;
```

```
	phi=0;
	//Uebersetzung berechnen
	uebersetzung = (((double)RADDURCHMESSER *
	3.14)/(double)TIKSPROUMDREHUNG);
	//grad in rad umrechnen
	gradinrad=((double)grad*6.28)/360;
	//Drehrichtung festlegen
	vzl = 0;
	vzr = 1;
	while (phi<=gradinrad)
	{
		uschall();
		PORTB |= (1<<PB2);		//Setzen des Pin B2 auf logisch 1
		PORTD |= (1<<PD6);		//Setzen des Pin D6 auf logisch 1
		PORTD &= ~(1<<PD5);		//Setzen des Pin D5 auf logisch 0
		PORTB &= ~(1<<PB1);		//Setzen des Pin B1 auf logisch 0
		//Winkel berechnen
		phi=((((double)odo_rechts*uebersetzung) -

	(double)odo_links*uebersetzung))/((double)ABSTANDRADZUMITTE*2);
		//Absolutbetrag
		phi=sqrt(phi*phi);
	}
	return 0;
}

int ldrehenodo(int grad){
	double uebersetzung;
	double gradinrad;
	odo_rechts=0;
	odo_links=0;
	//Korrekturfaktor odometrie
	grad = grad / KORREKTURODOL;
	phi=0;
	//Uebersetzung berechnen
	uebersetzung = (((double)RADDURCHMESSER *
	3.14)/(double)TIKSPROUMDREHUNG);
	//grad in rad umrechnen
	gradinrad=((double)grad*6.28)/360;
	//Drehrichtung festlegen
	vzl = 1;
	vzr = 0;
	while (phi<=gradinrad)
	{
		uschall();
		PORTB |= (1<<PB1);		//Setzen des Pin B1 auf logisch 1
		PORTD |= (1<<PD5);		//Setzen des Pin D5 auf logisch 1
		PORTB &= ~(1<<PB2);		//Setzen des Pin B2 auf logisch 0
		PORTD &= ~(1<<PD6);		//Setzen des Pin D6 auf logisch 0
		//Winkel berechnen
		phi=((((double)odo_rechts*uebersetzung) -
		(double)odo_links*uebersetzung))/((double)ABSTANDRADZUMITTE*2);
		//Absolutbetrag
		phi=sqrt(phi*phi);
	}
	return 0;
}

int ldrehen(){
```

```
	vzl = 1;
	vzr = 0;
	PORTB |= (1<<PB1);		//Setzen des Pin B1 auf logisch 1
	PORTD |= (1<<PD5);		//Setzen des Pin D5 auf logisch 1
	PORTB &= ~(1<<PB2);		//Setzen des Pin B2 auf logisch 0
	PORTD &= ~(1<<PD6);		//Setzen des Pin D6 auf logisch 0
	return 0;
}

int halt(){
	PORTB |= (1<<PB2);		//Setzen des Pin B2 auf logisch 1
	PORTD |= (1<<PD5);		//Setzen des Pin D5 auf logisch 1
	PORTB |= (1<<PB1);		//Setzen des Pin B1 auf logisch 1
	PORTD |= (1<<PD6);		//Setzen des Pin D6 auf logisch 1
	return 0;
}

void fehler(){
	//Fehler aufgetreten - halten & Piepscode ausgeben
	halt();
	while (1){
		PORTC |= (1<<PC2);
		_delay_ms(3000);
		PORTC &= ~(1<<PC2);
		_delay_ms(3000);
	}
}

//interruptroutine für int0(linkes Rad)
ISR(INT0_vect)
{
	if (vzl==0)
	{
		odo_links++;
	}else{
		odo_links--;
	}
}

//interruptroutine für int1 (rechtes Rad)
ISR(INT1_vect)
{
	if (vzr==0)
	{
		odo_rechts++;
	}else{
		odo_rechts--;
	}
}
int vsoll_pid(int direction)
{
	//direction gibt das Rad an welches geregelt werden soll
	//direction = 1 --> linkes Rad direction = 2 --> rechtes Rad
	//Konstanten der Regelung definieren
	#define Kp 1
	#define Ki 10 //10
	#define Kd 2
	#define Skalierung 1000
	//aktuelle Abweichung errechnen
```

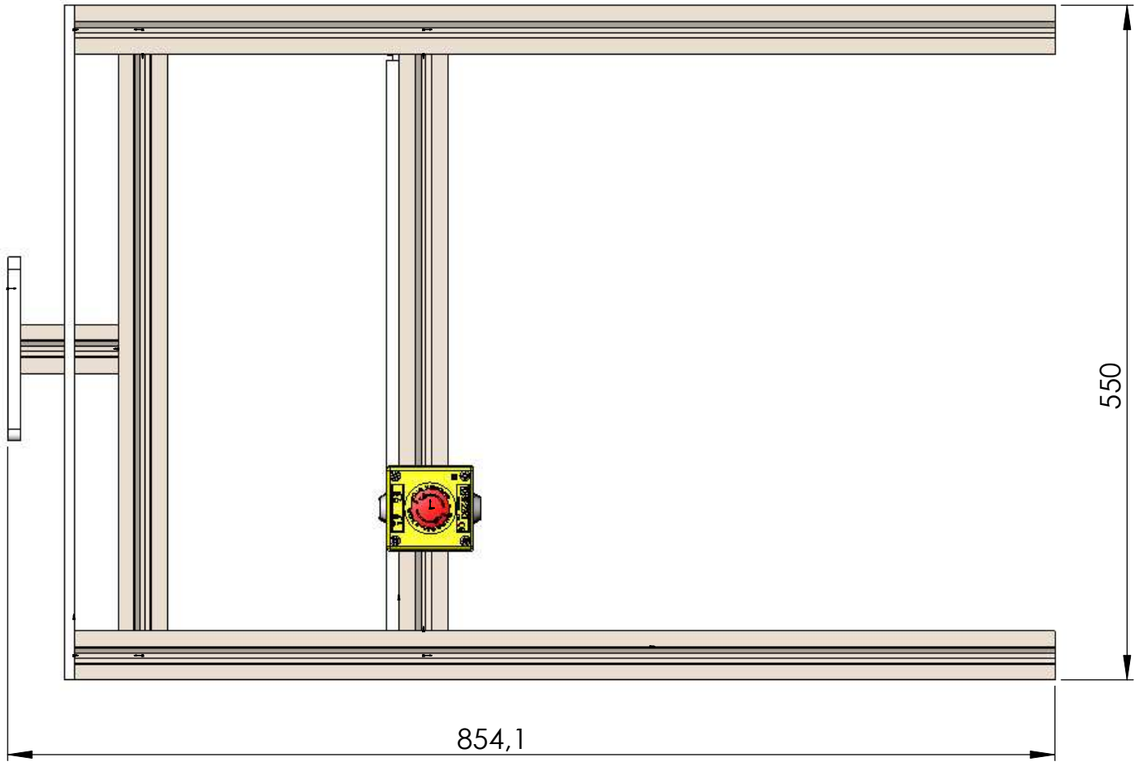
```
	if (direction==1)
	{
		eb=1023 - frontleft;
	}
	if (direction==2)
	{
		eb=1023 - frontright;
	}
	if (direction==11)
	{
		eb=1023 - frontlefta;
	}
	if (direction==22)
	{
		eb=1023 - frontrighta;
	}
	//Regelabweichung berechnen
	if (direction==1) vsollwert = Kp * eb + Ki * (eb + eblalt) + Kd * (eb
	- eblalt);
	if (direction==2) vsollwert = Kp * eb + Ki * (eb + ebralt) + Kd * (eb
	- ebralt);
	if (direction==11) vsollwert = Kp * eb + Ki * (eb + eblaalt) + Kd *
	(eb - eblaalt);
	if (direction==22) vsollwert = Kp * eb + Ki * (eb + ebraalt) + Kd *
	(eb - ebraalt);
	//Regelabweichung skalieren und normieren
	//Skalieren: Da hohe Werte bei der Berechnung entstehen werden diese
	//um die Skalierung verkleinert(Division)
	//Normieren: Der berechnete Wert wird zwischen max. und min.
	//Geschwindigkeit eingeordnet
	normieren = (((double)GESCHWMAX -
	(double)GESCHW)/(((double)Kp*1024+(double)Ki*(1024+1024))/(double)Ska
	lieru ng));
	vsollwert = ((vsollwert/Skalierung)*normieren);
	return vsollwert;
}

void uschall(int wartezeitnachmessung, int reichweite){
	int uschallmitte=0;
	int uschalllinks=0;
	int uschallrechts=0;
	//Ultraschallsensoren abfragen (Hindernisdetektion)
	sigsens2();		// Sensor Vorne Mitte abfragen
	echosens2();
	uschallmitte=count;
	_delay_ms(1);
	sigsens1a4();	// Sensor Vorne Links abfragen
	echosens1();
	uschalllinks=count;
	_delay_ms(1);
	sigsens3a5();	// Sensor Vorne Rechts abfragen
	echosens3();
	uschallrechts=count;
	if (reichweite==0){
		if (uschallmitte<=SENSOK || uschalllinks<=SENSKURZ ||
					uschallrechts<=SENSKURZ) {
			uschalltol++;
		}else{
			uschalltol=0;
```

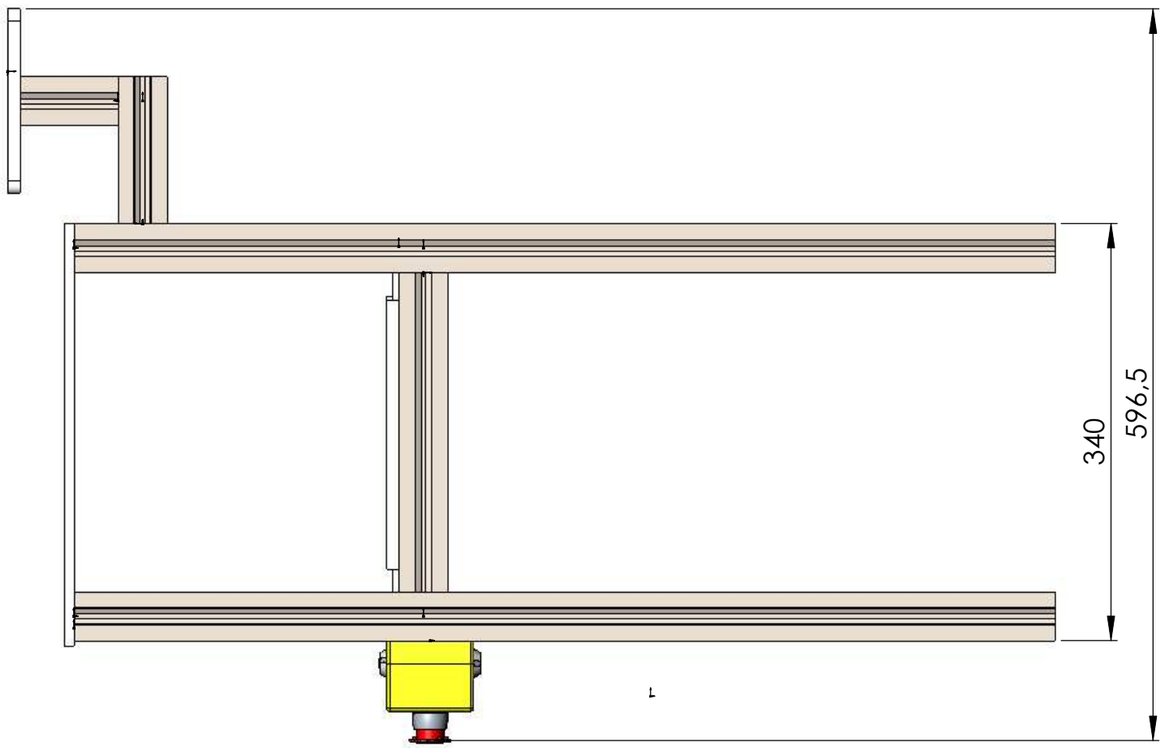
```
		}
	}else{
		if (uschallmitte<=SENSDREH || uschalllinks<=SENSDREH ||
				uschallrechts<=SENSDREH) {
			uschalltol++;
		}else{
			uschalltol=0;
		}
	}

	//Wenn Hindernis detektiert und Toleranz überschritten
	if (uschalltol>5){
		uschalltol=0;
		while (uschallmitte<=SENSOK || uschalllinks<=SENSKURZ ||
				uschallrechts<=SENSKURZ)
			{
				halt();
				_delay_ms(20);
				sigsens2();  // Sensor Vorne Mitte abfragen
				echosens2();
				uschallmitte=count;
				_delay_ms(1);
				sigsens1a4();     // Sensor Vorne Links abfragen
				echosens1();
				uschalllinks=count;
				_delay_ms(1);
				sigsens3a5(); // Sensor Vorne Rechts abfragen
				echosens3();
				uschallrechts=count;
			}
	}
	_delay_ms (wartezeitnachmessung);
}

int main(void){
	//Zeit zwischen einzelnen Linienmessung (Zeit des Fahrens) 12
	#define stepgroesse 30 //30
	int richtung;
	int fehlerlinie=0;
	int i=1;
	init();
	dvrs1();
	dvrs2();
	pwminit1();
	pwminit2();
	halt();
	Linienverfolgung_init();
	//Initialisieren der externen Interrupts
	MCUCR = (0<<ISC11) | (1<<ISC10) | (0<<ISC01) | (1<<ISC00);
	//Any logical     //Change detected
	GICR = (1<<INT1 | 1<<INT0);         //Aktivieren der Pins
	sei();              //globale interrupts aktivieren
	_delay_ms(1000);
	aktvl = GESCHWMAX;
	aktvr = GESCHWMAX;
	while(1){
	uschall(1,0);
	richtung = Linie_lesen();
	if (richtung==0){
```

```
		//Fehler: keine Linie gefunden
		//nach 50mal anhalten (Um kleine Unterbrechungen in der Linie
		//zu übergehen)
		OCR0 = 30;
		OCR2 = 30;
		aktvl = 30;
		aktvr = 30;
		fahrenvor();
		fehlerlinie++;
		if (fehlerlinie > 200) fehlerlinie=60;
		if (fehlerlinie > 50)
		{
			halt();
		}
	}else{
		fehlerlinie=0;
		fahrenvor();
		//Geschwindigkeitsziel abfragen
		zl = GESCHWMAX - vsoll_pid(1);
		zr = GESCHWMAX - vsoll_pid(2);
		zal = GESCHWMAX - vsoll_pid(11);
		zar = GESCHWMAX - vsoll_pid(22);
		if (richtung==30){
			zl=GESCHWMAX/2;
			zr=GESCHWMAX/2;
		}
		//Priorität der Phototransistoren
		if (richtung==20 || richtung==10 || richtung==40 ||
		richtung==50){
			if (richtung==20) zl = (zl*4 + zal)/5;
			if (richtung==10) zl = (zl + zal*4)/5;
			if (richtung==40) zr = (zr*4 + zar)/5;
			if (richtung==50) zr = (zr + zar*4)/5;
		}else{
			zl=(zl+zal)/2;
			zr=(zr+zar)/2;
		}
		//Geschwindigkeit in 5 Stufen erhöhen
		for(i=1; i<6; i++)
		{
			//linkes Rad
			if (aktvl > zl)
			{
				aktvl = aktvl - (int)(((aktvl-zl)*i)/5);
				OCR2 = aktvl;
			}else{
				aktvl = aktvl + (int)(((zl-aktvl)*i)/5);
				OCR2 = aktvl;
			}
			//rechtes Rad
			if (aktvr > zr)
			{
				aktvr = aktvr - (int)(((aktvr-zr)*i)/5);
				OCR0 = aktvr;
			}else{
				aktvr = aktvr + (int)(((zr-aktvr)*i)/5);
				OCR0 = aktvr;
			}
			_delay_ms(stepgroesse/5);
```

```
		}
	}
	//Linie lesen und entsprechend fahren
	//R=rechts, L=links, M=Mitte
	// 10=L, 20=LM, 30=M, 40=RM, 50=R
	//Abfrage des Farbsensors ######################
	if ( !(PINC & (1<<PINC6)) ) {
		//GELB
		OCR0 = GESCHWMAX;
		OCR2 = GESCHWMAX;
		aktvr = GESCHWMAX;
		aktvl = GESCHWMAX;
		rdrehenodo(170);
		_delay_ms(200);
		OCR0 = 18;
		OCR2 = 18;
		aktvr = 18;
		aktvl = 18;
		i=0;
		while (Linie_lesen()==0 && i<500){
			uschall(1, 1);
			fahrenvor();
			_delay_ms(10);
			i++;
		}
		_delay_ms(200);
		OCR0 = GESCHWMAX;
		OCR2 = GESCHWMAX;
		aktvr = GESCHWMAX;
		aktvl = GESCHWMAX;
		ldrehenodo(1);
		_delay_ms(200);
		OCR0 = 24;
		OCR2 = 24;
		aktvr = 24;
		aktvl = 24;
		richtung=0;
		i=0;
		while (richtung!=20 && richtung !=30 && richtung != 40 &&
		i<500){
			uschall(1, 1);
			fahrenvor();
			richtung = Linie_lesen();
			_delay_ms(10);
			i++;
		}


	}
	if ( !(PINC & (1<<PINC7)) ) {
		//GELB
		halt();
		_delay_ms(6000);
		_delay_ms(6000);
		_delay_ms(6000);
	}
	}
}
```

Anhang G: CAD-Zeichnungen





POS-NR.	BENENNUNG	BESCHREIBUNG	MENGE
1	P_011_002	Profil 800	4
2	P_011_001	Profil 470	2
3	P_011_000	Profil 260	2
4	P_011_005	Profil 120	2
5	P_011_006	Profil 470	1
6	P_011_004	Lochrasterplatte	1
7	P_011_003	Ablageplatte	1
8	P_011_007	Profil 80	1
9	P_011_008	Adapterplatte Lampe	1
10	P_000_031	Not-Aus	1

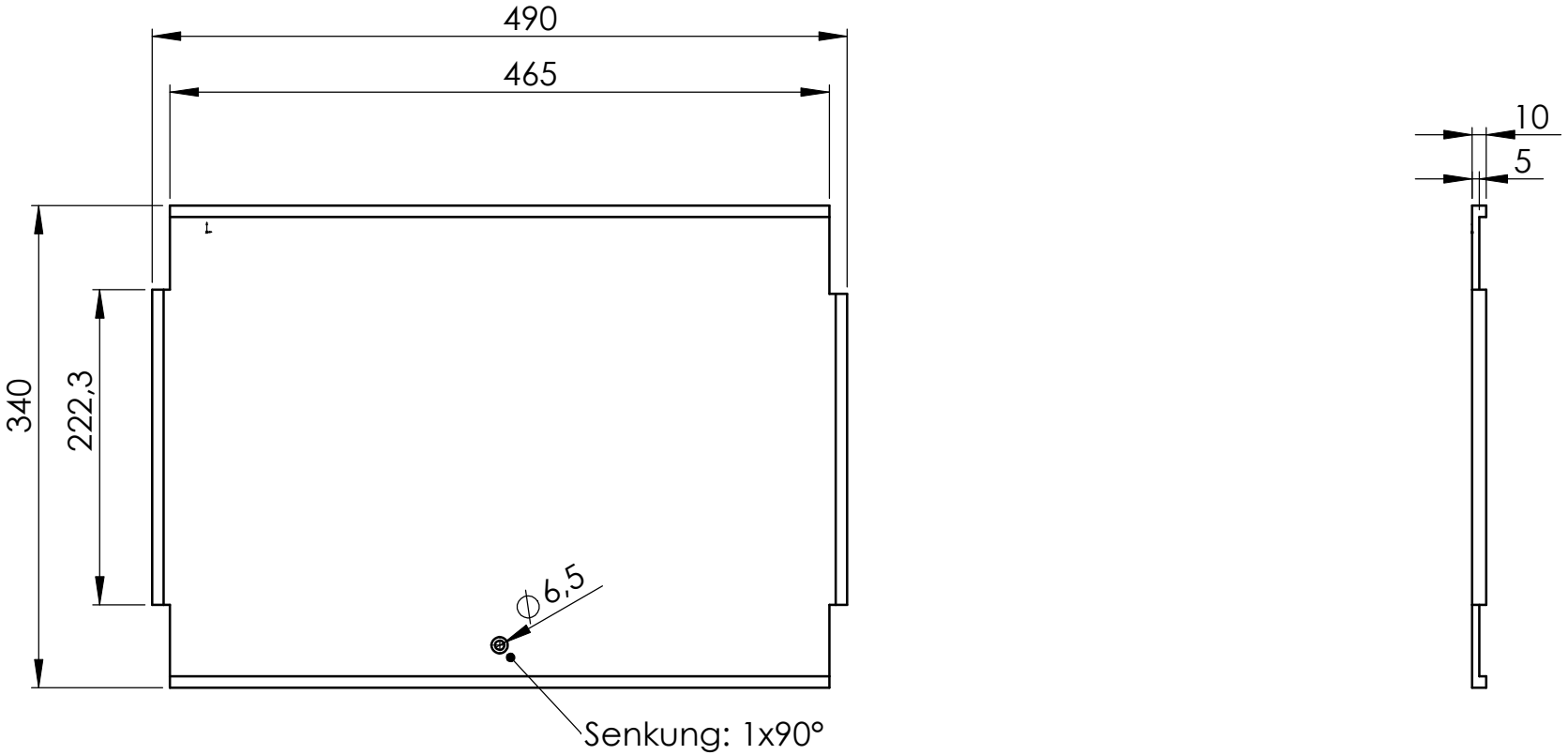
DESIGNATION / BENENNUNG: Gesamtaufbau

DRAWING NR. / ZEICHNUNGS NR.: A_011_000

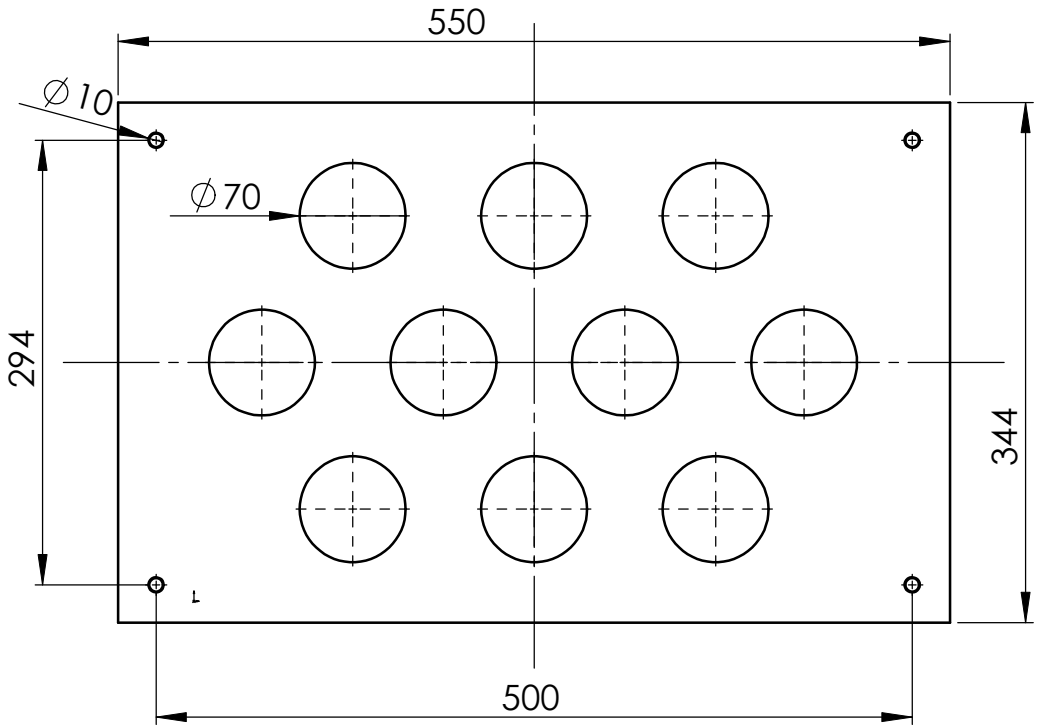
SUPERS. / ERSETZT: –

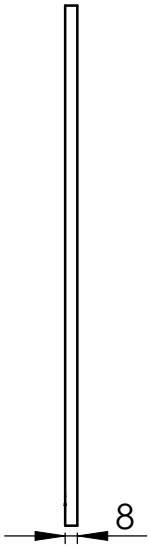
SCALE / MASZSTAB	1:5	MATERIAL	
WEIGHT / GEWICHT		SURFACE / OBERFLÄCHE	–
SHEETSIZE / BLATTFORMAT	DIN - A3	Name	Beier Lukas
SHEET / BLATT	1 / 1	ID	03102040XX
DATE / DATUM	15.04.2011	SEM	SEM4

PROPERTY OF / EIGENTUM VON: Fachhochschule Technikum Wien - Studiengang: Mechatronik / Robotik

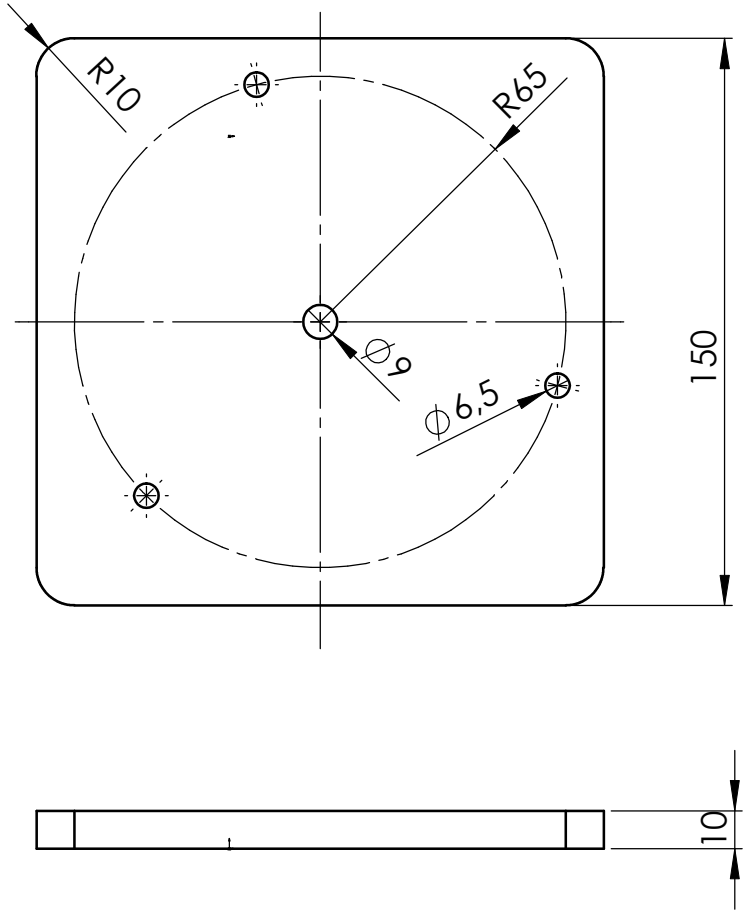


DESIGNATION / BENENNUNG	Ablageplatte		DRAWINGNR. / ZEICHNUNGSNR.	P_011_003	
			SUPERS. / ERSETZT	–	
SCALE / MASZSTAB	1:5	MATERIAL	SHEETSIZE / BLATTFORMAT	DIN - A4	Name: Beier Lukas
			SHEET / BLATT	1 / 1	ID: 03102040XX
WEIGHT / GEWICHT	kg	SURFACE / OBERFLÄCHE –	DATE / DATUM	27.04.2011	SEM: SEM4
PROPERTY OF / EIGENTUM VON	Fachhochschule Technikum Wien - Studiengang: Mechatronik / Robotik				



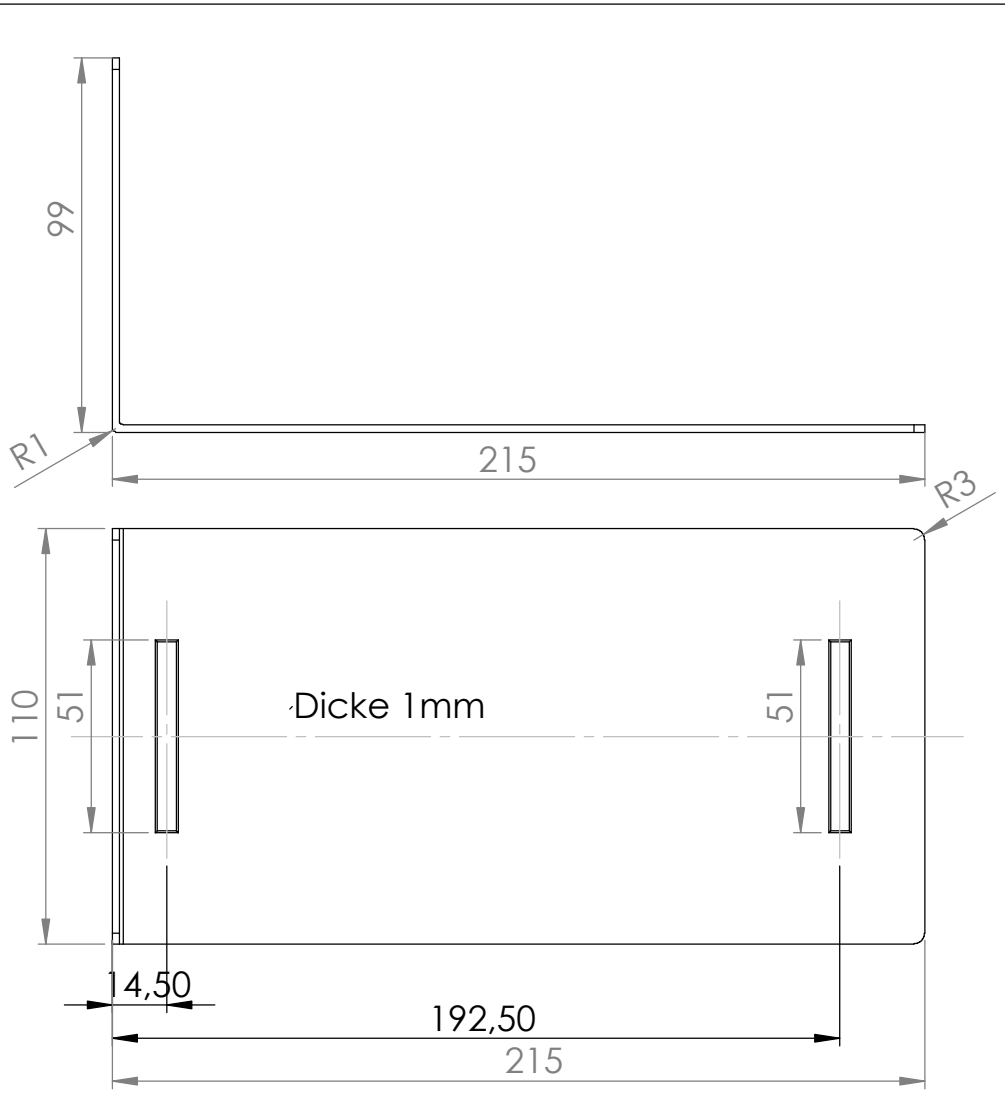


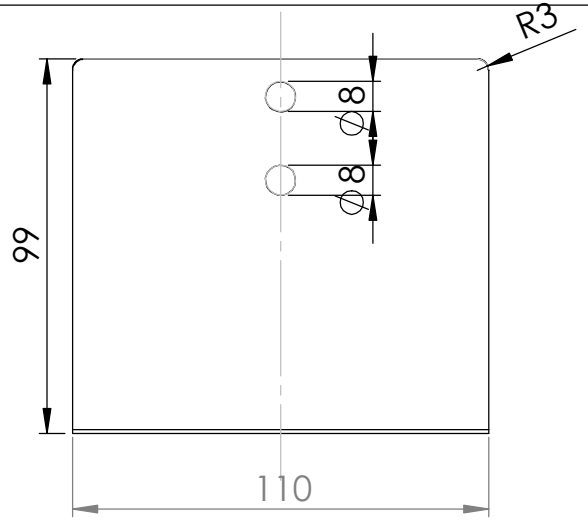
DESIGNATION / BENENNUNG: Lochrasterplatte			DRAWINGNR. / ZEICHNUNGSNR.: P_011_004		
			SUPERS. / ERSETZT: –		
SCALE / MASZSTAB	1:5	MATERIAL	SHEETSIZE / BLATTFORMAT	DIN - A4	Name: Beier Lukas
			SHEET / BLATT	1 / 1	ID: 03102040XX
WEIGHT / GEWICHT	kg	SURFACE / OBERFLÄCHE: –	DATE / DATUM	27.04.2011	SEM: SEM4
PROPERTY OF / EIGENTUM VON	Fachhochschule Technikum Wien - Studiengang: Mechatronik / Robotik				

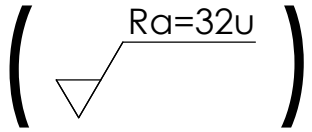


DESIGNATION / BENENNUNG	Adapterplatte Lampe		DRAWINGNR. / ZEICHNUNGSNR.	P_011_008	
			SUPERS. / ERSETZT	–	
SCALE / MASZSTAB	1:2	MATERIAL	SHEETSIZE / BLATTFORMAT	DIN - A4	Name: Beier Lukas
			SHEET / BLATT	1 / 1	ID: 03102040XX
WEIGHT / GEWICHT	kg	SURFACE / OBERFLÄCHE –	DATE / DATUM	27.04.2011	SEM: SEM4
PROPERTY OF / EIGENTUM VON	Fachhochschule Technikum Wien - Studiengang: Mechatronik / Robotik				

Schutzvermerk nach DIN 34



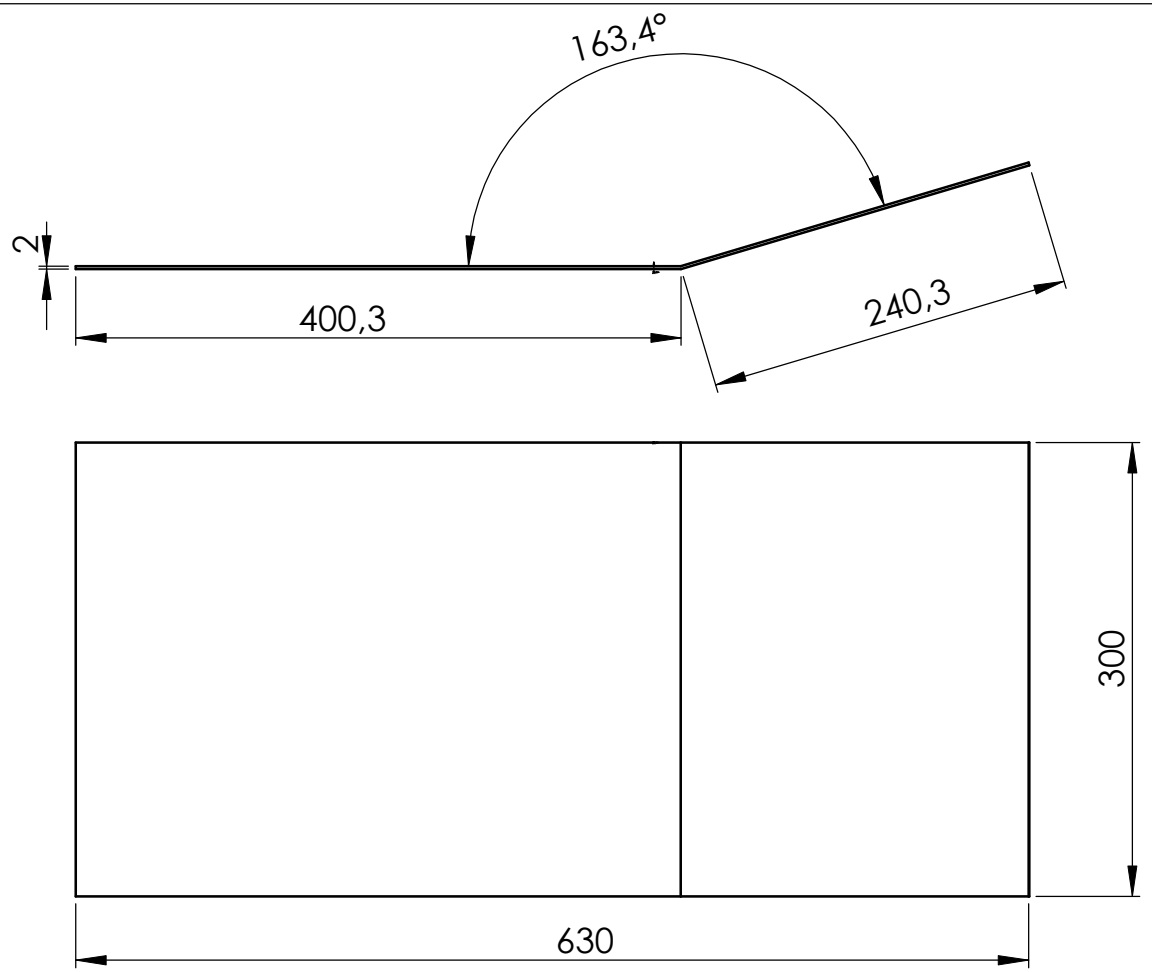




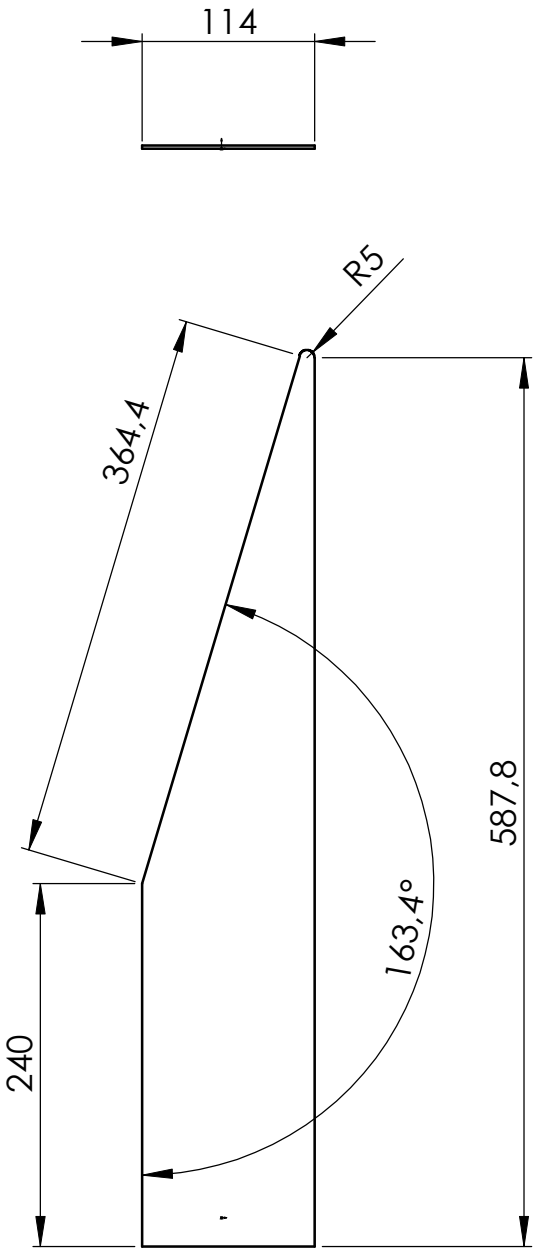
Alle unbemaßten Radien R=0.2 mm

DESIGNATION / BENENNUNG	Linienverfolgung gebogen			DRAWINGNR. / ZEICHNUNGSNR.	P_012_000
				SUPERS. / ERSETZT	–
SCALE / MASZSTAB	1:2	MATERIAL	Blech	SHEETSIZE / BLATTFORMAT	DIN - A4
WEIGHT / GEWICHT	~300g	SURFACE / OBERFLÄCHE	32u	SHEET / BLATT	1 / 1
				DATE / DATUM	27.08.2007
PROPERTY OF / EIGENTUM VON	Fachhochschule Technikum Wien - Studiengang: Mechatronik / Robotik				

Name	Kraupp
ID	mr09b033
SEM	SEM 4



DESIGNATION / BENENNUNG	Seitenabdeckung		DRAWINGNR. / ZEICHNUNGSNR.	P_012_001	
			SUPERS. / ERSETZT	–	
SCALE / MASZSTAB	1:5	MATERIAL	SHEETSIZE / BLATTFORMAT	DIN - A4	Name: Beier Lukas
			SHEET / BLATT	1 / 1	ID: 03102040XX
WEIGHT / GEWICHT	kg	SURFACE / OBERFLÄCHE –	DATE / DATUM	27.04.2011	SEM: SEM4
PROPERTY OF / EIGENTUM VON	Fachhochschule Technikum Wien - Studiengang: Mechatronik / Robotik				



Obere Abdeckung

DESIGNATION / BENENNUNG	Obere Abdeckung		DRAWINGNR. / ZEICHNUNGSNR.	P_012_002	
			SUPERS. / ERSETZT	–	
SCALE / MASZSTAB	1:5	MATERIAL	SHEETSIZE / BLATTFORMAT	DIN - A4	Name: Beier Lukas
			SHEET / BLATT	1 / 1	ID: 03102040XX
WEIGHT / GEWICHT	kg	SURFACE / OBERFLÄCHE –	DATE / DATUM	27.04.2011	SEM: SEM4
PROPERTY OF / EIGENTUM VON	Fachhochschule Technikum Wien - Studiengang: Mechatronik / Robotik				