

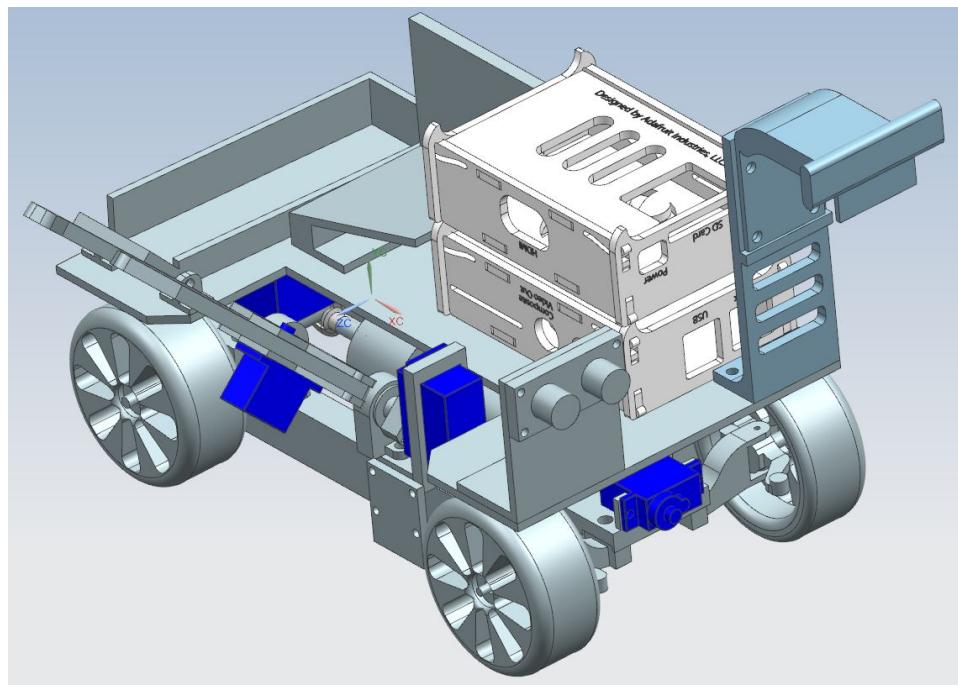
LUCERNE UNIVERSITY OF APPLIED  
SCIENCES AND ARTS

PREN 1

---

## Ökologische Müllabfuhr - OMA

---



Gruppe 34

Felber, Manuel Elektrotechnik	Minder, Pascal Elektrotechnik	Leila, Müller Maschinenbau
Marti, Roman Maschinenbau	Peltier, Valentin Maschinenbau	Hansen, Jonas Informatik
	Gabriel, Severin Informatik	

Dozent  
Vogel, Martin  
Dipl. El. Ing. ETH

6. Januar 2016

## Abstract

An der Hochschule Luzern, Technik & Architektur dürfen die Studierenden der Fachbereiche Maschinentechnik, Informatik und Elektrotechnik im Rahmen des Modules Produktentwicklung 1 (PREN1) in Zusammenarbeit ein Projekt realisieren. Die Aufgabenstellung lautet, ein autonom fahrendes Fahrzeug zu entwickeln, welches einen Kurs abfährt und zugleich Abfallcontainer einsammelt.

Im PREN 1 wird eine Konzeptlösung aufgrund von Recherchen und Versuchen entwickelt. Im PREN 2 wird die Konzeptlösung realisiert.

In dieser Arbeit wird eine mögliche Konzeptlösung beschrieben. Das in der Projektarbeit geplante autonome Fahrzeug wird durch einen Gleichstrommotor (DC-Motor) angetrieben, welcher durch ein Kegelradgetriebe zusätzlich unterstützt wird. Die Spurhaltung wird durch eine Kamera gesteuert, welche sich an den Mittellinien der Strasse orientiert. Im gleichen Prozess übernimmt die Kamera auch die Erkennung der Abfalleimer. Mittels einem Farbsensor kann die Greifvorrichtung des Fahrzeuges zum Container ausgerichtet werden. Die Aufnahme erfolgt mit einem Zwei-Punkt Greifer und einem Hebelarm, welcher den Abfalleimer über eine rotierende Achse auf dem Fahrzeug entlädt. Die Entsorgung des Abfalls in den Container wird mit einer Kippmulde realisiert.

# Inhalt

<b>1 Einleitung</b>	<b>1</b>
<b>2 Konzeptfindung</b>	<b>2</b>
2.1 Aufgabenstellung und Produktanforderung . . . . .	2
2.2 Teilfunktionenanalyse und Recherche . . . . .	2
2.3 Morphologischer Kasten . . . . .	2
2.4 Auswahl . . . . .	3
2.5 Bewertung . . . . .	5
2.6 Entschluss . . . . .	6
<b>3 Lösungskonzept</b>	<b>7</b>
<b>4 Komponentenbeschreibung</b>	<b>9</b>
4.1 Hebemechanismus und Greifer . . . . .	9
4.1.1 Funktionsweise . . . . .	9
4.1.2 Detaillierte Beschreibung . . . . .	9
4.1.3 Begründung . . . . .	10
4.2 Fahrgestell und Lenkung . . . . .	11
4.2.1 Funktionsweise . . . . .	11
4.2.2 Detaillierte Beschreibung . . . . .	11
4.2.3 Begründung . . . . .	12
4.3 Entsorgung . . . . .	13
4.3.1 Berechnung . . . . .	13
4.3.2 Funktionsweise . . . . .	14
4.3.3 Detaillierte Beschreibung . . . . .	14
4.3.4 Begründung . . . . .	15
4.4 Erkennung . . . . .	16
4.4.1 Tinkerforge Hardware-Konzept . . . . .	16
4.4.2 Farbsensor . . . . .	17
4.4.3 Ultraschallsensor . . . . .	18
4.5 Energieversorgung . . . . .	20
4.5.1 Akku . . . . .	20
4.5.2 Akkuüberwachung . . . . .	21
4.5.3 Spannungswandlung . . . . .	21
4.6 Motoren . . . . .	22
4.6.1 Auswahl . . . . .	22

4.6.2	Motorentreiber . . . . .	23
4.6.3	Encoder . . . . .	24
4.6.4	Motorensteuerung und Regelung . . . . .	25
4.7	Embedded System . . . . .	26
4.7.1	Hardware . . . . .	26
4.7.2	API . . . . .	27
4.7.3	Produktiv- und Entwicklungssystem . . . . .	29
4.8	Kommunikation . . . . .	31
4.8.1	Protokoll . . . . .	31
4.8.2	Message Types . . . . .	31
4.8.3	Payloads . . . . .	32
4.9	Software . . . . .	33
4.9.1	Spurhaltung . . . . .	33
4.9.2	Erkennung Rechtsvortritt . . . . .	35
4.9.3	Erkennung Container . . . . .	35
4.9.4	Komponentendiagramm Raspberry Pi 2 . . . . .	39
4.9.5	Ablaufsteuerung . . . . .	41
<b>5</b>	<b>Projektmanagement und Planung</b>	<b>48</b>
5.1	Organigramm . . . . .	48
5.2	Funktionsbeschrieb . . . . .	48
5.3	Planung . . . . .	49
5.3.1	Meilensteine . . . . .	49
5.3.2	Verwendete Tools . . . . .	50
5.3.3	Projektplan . . . . .	51
<b>6</b>	<b>Schlussdiskussion</b>	<b>52</b>
6.1	Kosten . . . . .	52
6.2	Lessons Learned . . . . .	53
6.2.1	Elektronik . . . . .	53
6.2.2	Maschinentechnik . . . . .	53
6.2.3	Informatik . . . . .	54
6.3	Risiken . . . . .	54
6.4	Massnahmen . . . . .	55
6.5	Ausblick PREN 2 . . . . .	56
6.6	Fazit . . . . .	56
<b>Abbildungsverzeichnis</b>		<b>58</b>

<b>Tabellenverzeichnis</b>	<b>60</b>
<b>A Glossar</b>	<b>61</b>
<b>B Quellenverzeichnis</b>	<b>62</b>
<b>C Anhang</b>	<b>63</b>
C.1 Technischer Bericht Stereokamera . . . . .	63
C.1.1 Ausgangslage . . . . .	63
C.1.2 Grundlagen . . . . .	63
C.1.3 Versuchsaufbau . . . . .	64
C.1.4 Versuch 1 - Block Matching-Algorithmus . . . . .	65
C.1.5 Versuch 2 - Semi Global Block Matching . . . . .	65
C.1.6 Fazit . . . . .	66
C.2 Kamera Spurhaltung . . . . .	67
C.2.1 Ausgangslage . . . . .	67
C.2.2 Versuchsaufbau . . . . .	67
C.2.3 Versuch - Line Detection Algorithmus . . . . .	67
C.3 Technischer Bericht Motorentreiber und DC Motor . . . . .	69
C.3.1 Ausgangslage . . . . .	69
C.3.2 Versuchsaufbau . . . . .	69
C.3.3 Fazit . . . . .	70
C.4 Technischer Bericht Sensoren . . . . .	72
C.4.1 Farbsensor . . . . .	72
C.4.2 Fazit Farbsensor . . . . .	73
C.4.3 Ultraschallsensor . . . . .	74
C.4.4 Fazit Ultraschallsensor . . . . .	74
C.5 Aufgabenstellung . . . . .	75
C.6 Produktanforderungen auf Testat 1 . . . . .	84
C.7 Morphologischer Kasten . . . . .	89
C.8 Entsorgung . . . . .	90
C.8.1 Berechnungen . . . . .	90
C.8.2 Funktionsweise . . . . .	91
C.8.3 Detaillierte Beschreibung . . . . .	91
C.9 Berechnung Klemmkraft . . . . .	93
C.10 Datenblätter . . . . .	95
C.10.1 Maxon Motor RE30 - 310007 . . . . .	95
C.10.2 Maxon Getriebe GP 32 C - 166931 . . . . .	97
C.10.3 Maxon Tacho MR - 228452 . . . . .	99

C.10.4 Motorencontroller MR001-004.2 . . . . .	101
C.10.5 Counter Interface IC HCTL 2022 . . . . .	104
C.11 Webbench-Tool zur Berechnung der Buck Converter . . . . .	126
C.11.1 Webbench Datasheet 5V Spannungsversorgung . . . . .	126
C.11.2 Webbench Datasheet 7V Spannungsversorgung . . . . .	132
C.12 Testsoftware Sensorsteuerung . . . . .	139
C.13 Testsoftware Containererkennung . . . . .	153
C.14 Testsoftware Spurhaltung . . . . .	155

## 1 Einleitung

In den Modulen PREN 1 und PREN 2 der Hochschule für Technik und Architektur Luzern werden jedes Jahr interdisziplinäre Teams zusammengestellt, die eine komplexe Aufgabenstellung lösen müssen.

Besonderen Wert wird dabei auf die Zusammenarbeit der verschiedenen Fachrichtungen gelegt. Die Aufteilung des Projekts geschieht wie folgt: Im Modul PREN 1 gilt es, ein Lösungskonzept zu erarbeiten. Es werden für die verschiedenen Teifunktionen Lösungsvarianten recherchiert, um das bestmögliche Lösungskonzept zu entwickeln. Dazu werden Berechnungen und Recherche ange stellt. Auch werden verschiedene Funktionsmuster und Tests der jeweiligen Komponenten durchgeführt. Diese Lösungsvariante wird im zweiten Teil, dem Modul PREN 2, umgesetzt. Zum Abschluss findet ein Wettbewerb statt. Bei diesem werden die Teams nach verschiedenen Kriterien bewertet. Das Team mit den meisten Punkten wird zum Sieger.

Diese Dokumentation widmet sich dem Modul PREN 1. Als erstes wird kurz auf die Lösungsfindung eingegangen. Weiter wird in Kapitel 3 eine Übersicht über das Konzept gegeben. In Kapitel 4 werden die einzelnen Teifunktionslösungen genau beschrieben. Dann wird kurz auf das Projektmanagement und die Planung eingegangen. Schliesslich wird in der Schlussdiskussion auf die Kosten, Risiken und die offengebliebenen Punkte eingegangen und ein Ausblick auf PREN 2 gemacht.

## 2 Konzeptfindung

Dieses Kapitel beschreibt den Konzeptfindungsprozess, dem das Team gefolgt ist, beginnend mit der Analyse der Aufgabenstellung bis hin zur Auswahl des Konzepts, das schliesslich in den Kapiteln 3 und 4 beschrieben wird.

### 2.1 Aufgabenstellung und Produktanforderung

Als erstes wurde die Aufgabenstellung untersucht und die Produktanforderungen genauer beschrieben. Die Aufgabenstellung ist im Anhang im Kapitel C.5 zu finden.

Die Aufgabe des Teams war es, ein Konzept für einen autonomen Müllabfuhrroboter zu entwickeln. Dieser soll in der Lage sein, selbstständig eine Strecke abzufahren, den Müll aus Containern zu sammeln und diesen in einem Becken zu entsorgen. Dabei muss sowohl auf die Containerfarbe als auch auf mögliche Passanten oder Rechtsvortritt geachtet werden. Eine Auflistung aller Produktanforderungen ist im Kapitel C.6 zu finden.

### 2.2 Teilfunktionenanalyse und Recherche

Für die Konzeptfindung wurde dann eine Teilfunktionenanalyse erstellt. Damit wird die Aufgabenstellung besser handhabbar. Diese Teilefunktionen bestehen aus: Fahrgestell und Lenkung, Heben und Greifen, Entsorgen, Energieversorgung und Erkennung der Umgebung. Für diese Teilefunktionen wurden verschiedene Lösungsmöglichkeiten recherchiert, welche im morphologischen Kasten im Anhang unter dem Kapitel C.7 zusammengetragen wurden.

### 2.3 Morphologischer Kasten

Der Morphologische Kasten (Anhang C.7) erlaubte es dem Team, eine Übersicht über die möglichen Lösungen zu erhalten. Anhand des Inputs der verschiedenen Fachrichtungen erhielten die Teammitglieder Einblick in verschiedene Methoden der Problemlösung. Es wurden mehrere mögliche Gesamtkonzepte zusammengestellt. Dabei wurde auf die Umsetzbarkeit und denkbare Probleme geachtet, um verschiedene plausible Varianten zu finden. So wurden der Hamster als Energiequelle und das Beamen zum Abfalleinsammeln auf Grund technischer Schwierigkeiten und großem Aufwand ausgeschlossen.

Für das Konzept von PREN 1 wurden nach der Teilfunktionenanalyse verschiedene Lösungsmöglichkeiten entwickelt und genauer untersucht. Dabei

zeigten sich sowohl Vorteile und Nachteile verschiedener Teilstukturierungslösungen als auch mögliche Gesamtkonzepte.

Im Folgenden werden die verschiedenen Lösungskonzepte mithilfe eines morphologischen Kastens gezeigt. Sie werden im Kapitel 2.5 nach mehreren Kriterien bewertet und analysiert. Im Kapitel 4 wird das beste Lösungskonzept beschrieben.

## 2.4 Auswahl

Für die Konzeptanalyse wurden verschiedene Zielkriterien wie Geschwindigkeit und Genauigkeit definiert und nach ihrer Wichtigkeit gewichtet. Danach wurden die verschiedenen Lösungsvarianten anhand dieser Kriterien bewertet. Eine niedrige Punktzahl bedeutete, dass das Konzept die Kriterien schlecht erfüllt, eine hohe Punktzahl, dass es sie gut erfüllt. Dabei standen für die Teilstrukturen verschiedene Lösungsmöglichkeiten zur Auswahl. Ein Beispiel ist in der Tabelle 1 zu finden.

Konzept	Kamera RGB-Sensor	Liniensensor Ultraschall RGB-Sensor	Stereokamera
Beschreibung	Kamera prüft Spurhaltung	Liniensensor prüft Spurhaltung	Kamera prüft Spurhaltung
	Kamera prüft Rechtsvortritt	Ultraschall prüft Rechtsvortritt	Stereokamera prüft Rechtsvortritt
	RGB-Sensor prüft Farbe Container	RGB-Sensor prüft Farbe Container	Kamera prüft Farbe Container
	Fahrzeug in Position sobald	Fahrzeug in Position sobald	Fahrzeugposition berechnet mit Distanz zu Container + Position
	RGB-Sensor nicht mehr blau/grün	RGB-Sensor nicht mehr blau/grün	Früherkennung Container durch Kamera
	Früherkennung Container durch Kamera	keine Früherkennung	Greifarm
Vorteile	Guter Kompromiss bezüglich Aufwand und Ertrag	Distanzerkennung für Rechtsvortritt	Distanzerkennung für grosses Sichtfeld
	Positionierung bei Container	Einfache Implementierung	Alles in einem System
		Positionierung	Einfache Ergänzung mittels Sensor
Nachteile	Keine Distanzerkennung	Keine Objekterkennung	Rechenintensiv
	Rechtsvortritt		
	Distanzerkennung Container nicht möglich		
Risiken	Rechtsvortritt erkennen	Gestrichelte Linie (Kurve)	Genauigkeit Containerposition
		Falsches Objekt greifen	

Tab. 1: Beispiele für die Konzeptfindung.

## 2.5 Bewertung

Alle Teammitglieder hatten als Aufgabe eine funktionsfähige Lösung zu präsentieren. Die Varianten wurden im Morphologischen Kasten (Anhang C.7) eingezeichnet. Diese Ideen wurden anhand der zuvor festgelegten Gewichtungen dementsprechend berechnet und in der unteren Tabelle aufgelistet.

---

### Konzept

Zielkriterien	Gewichtungsfaktor											
	Variante 1			Variante 2			Variante 3			Variante 4		
<b>Geschwindigkeit</b>	0.2	4	0.8	4	0.8	3	0.6	1	0.2	4	0.8	
<b>Produktionsaufwand</b>	0.15	4	0.6	2	0.3	3	0.45	1	0.15	3	0.45	
<b>Genauigkeit</b>	0.3	4	1.2	4	1.2	5	1.5	5	1.5	5	1.5	
<b>Gewicht</b>	0.05	3	0.15	4	0.2	3	0.15	2	0.1	3	0.15	
<b>Programmieraufwand</b>	0.25	3	0.75	3	0.75	2	0.5	2	0.5	2	0.5	
<b>Energiebedarf</b>	0.05	3	0.15	3	0.15	3	0.15	2	0.1	2	0.1	
<b>Summe</b>	<b>1</b>		<b>3.65</b>		<b>3.4</b>		<b>3.35</b>		<b>2.55</b>		<b>3.5</b>	

Tab. 2: Bewertungstabelle der einzelnen Varianten.

<b>Zielkriterien</b>	<b>Bewertung</b>	
	1	5
<b>Geschwindigkeit</b>	sehr langsam	sehr schnell
<b>Produktionsaufwand</b>	hoch	gering
<b>Genauigkeit</b>	ungenau	genau
<b>Gewicht</b>	gross	klein
<b>Programmieraufwand</b>	komplex	simpel
<b>Energiebedarf</b>	hoch	tief
<b>Summe</b>	schlechte Lösung	gute Lösung

Tab. 3: Bewertungsskala

## 2.6 Entschluss

Die Bewertung der verschiedenen Varianten erfolgt in Tabelle 2. Nach den Bewertungen von Tabelle 3 entspricht das Konzept mit der höchsten Punktzahl den Zielkriterien am Besten. Die Variante 1 erhielt die höchste Punktzahl mit 3.65. Es folgen dicht darauf Variante 5 mit 3.5 Punkten und Variante 2 mit 3.4 Punkten. Die Unterschiede der drei Varianten liegen hauptsächlich bei der Wahl der Sensoren, daher sind sie auch recht ähnlich bewertet. Variante 5 unterscheidet sich von der Variante 1 auch bei der Energiequelle. Anstelle des LiPo-Akkus soll hier Solarenergie genutzt werden. Diese liefert jedoch nicht die benötigte Spannung. Ausserdem entsprechen die Lichtverhältnisse beim Wettbewerb eventuell nicht den Bedürfnissen, daher fällt die Variante 5 aus. Die Variante 4 ist aufgrund eines zu hohen Produktionsaufwandes und einer zu geringen Geschwindigkeit ausgeschieden. Der Spindelkipper lässt sich schlecht mit der Raupe kombinieren und wäre so deutlich langsamer, würde nur eine Klappe geöffnet werden, ebenso das kommt das Förderband anstelle des Hebelarms in Anwendung. Daher wurde Variante 1 als Konzept ausgewählt.

### 3 Lösungskonzept

Der Roboter besteht nach dem ausgewählten Lösungskonzept (siehe Kapitel 2.6) aus einem Zweiachsen-Fahrgestell mit Schenkellenkung. Angetrieben wird der Roboter über einen zentralen DC Motor, welcher über ein Getriebe und Kegelrad an der Hinterachse gekoppelt ist. Die Lenkung wurde wie bei einem Modellauto mit einer Schenkellenkung und Servo realisiert. Am Fahrgestell ist ein Hebelarm mit Greifer für den Container befestigt. Der Hebelarm kann eine Schwenkung von 120° um die Horizontale ausführen. Der Greifer besteht aus zwei Schenkeln, welche den Container umschließen und so packen können.

Gestartet wird der Roboter mit einer Software auf einem Notebook, welches über eine WLAN mit dem Roboter verbunden wird (siehe Abbildung 1). Das Fahrzeug orientiert sich mithilfe der Kamera an den Seiten- und Mittellinien und beginnt den Parcours abzufahren. Zusätzlich wird während der Fahrt das Trottoir von einer Kamera nach Container abgescannt. Das heißtt der Roboter fährt mit konstanter Geschwindigkeit (geschwindigkeitsgeregelt) der Fahrbahn entlang. Sobald er einen Container gesichtet hat, wird die Distanz abgeschätzt und dem Fahrtenregler mitgeteilt. Dieser wechselt in den positionsgeregelten Betrieb, drosselt die Geschwindigkeit und fährt genau die mitgeteilte Strecke ab. Zusätzlich wird zur Kontrolle eine Farbsensor auf der Seite eingeschaltet. Dieser Sensor kann die Fahrt unterbrechen, wenn er den Container detektiert. Danach schenkt der Heber zuerst auf die horizontale Achse nach unten, greift den Container und danach wieder um seine Achse nach oben. Wenn der Heber über die vertikale Achse schwenkt, wird der Container in ein Auffangbecken entleert. Danach wird der Container wieder abgesetzt. Nun wird die Fahrt fortgesetzt. Ist der Kurs abgefahren, wird über die Kamera auf das Parkfeld manövriert. Sobald das Fahrzeug parkiert ist, beginnt sich das komplette Auffangbecken zu kippen, um den Müll zu entleeren.

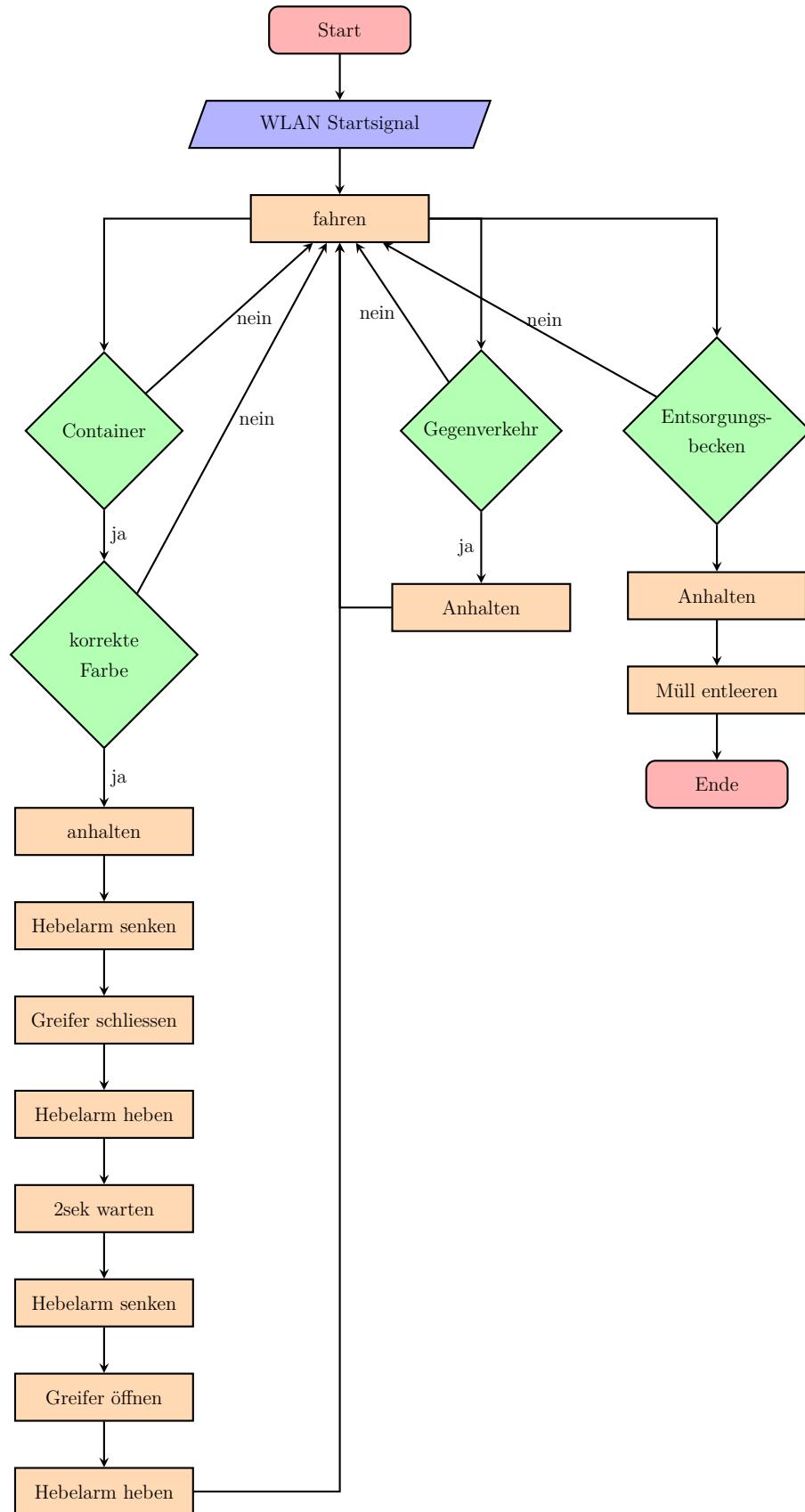


Abb. 1: Ablaufdiagramm

## 4 Komponentenbeschreibung

Das Lösungskonzept besteht aus einem Zweipunktgreifer mit Hebelarm, einem Fahrgestell mit vier Rädern, einer Achsschenkellenkung und einer Kippmulde. Die Energieversorgung wird durch ein LiPo Akku sichergestellt.

Als Rechenwerk werden zwei **Raspberry Pis** eingesetzt. Genannt werden die Rechenwerke **Raspberry Pi 1** und **Raspberry Pi 2**. Die Wahl der Sensoren liegt bei einer Kamera, einem RGB-Sensor sowie bei einem Ultraschallsensor. Im Folgenden werden die verschiedenen Teilfunktionen beschrieben und die Wahl dieser Funktion begründet.

### 4.1 Hebemechanismus und Greifer

#### 4.1.1 Funktionsweise

Ziel ist es, den Containers sicher und fest zu greifen, schnell und kontrolliert zu entleeren und schlussendlich wieder auf seine Position zurückzulegen. Mittels zwei Servomotoren werden Hebelarm und Greifer rein- und rausgeschwenkt und auf- und zugemacht.

#### 4.1.2 Detaillierte Beschreibung

Beim Hebelarm handelt es sich um einen Zweipunktgreifer, welcher mittels einem Servomotor beide Greiferplatten auf- und zumachen kann. Anhand der Abbildung 2a sind die Bemassungen der am jeweiligen Arm zu erkennen. Ausschlaggebend für die Länge des Greifers ist die maximale Gesamthöhe des Fahrzeuges, welche von Beginn weg festgelegt wurde. Die Bedeutung dieser Länge ist in Abbildung 2b besser ersichtlich, da der Hebelarm in vertikaler Ausrichtung gelagert wird und so die vorgegebene Höhe der Fahrzeuge nicht überschreiten darf.

Ebenfalls zu bemerken, ist dass die Rotationsachse im Lösungskonzept möglichst tief liegt, da so eine maximale Hebelarmlänge erreicht wird ohne die Höhenvorgaben zu verletzen.

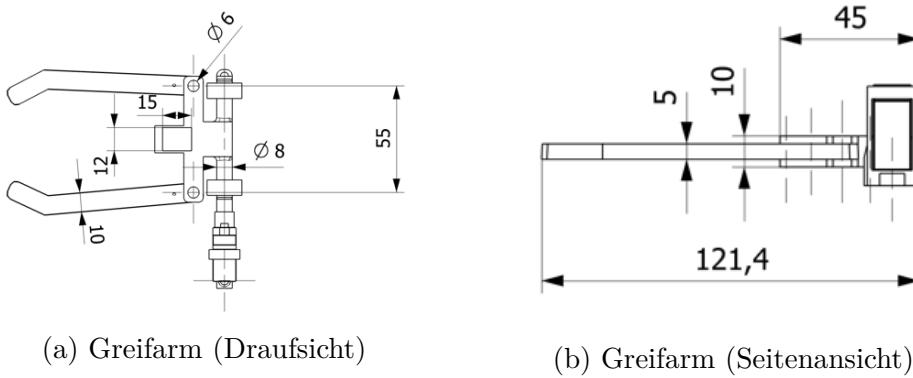


Abb. 2: Greifarm

Stehen neben dem Mülleimer andere Gegenstände oder Figuren, dürfen diese nicht berührt oder umgekippt werden. Daher ist die zulässige Dicke der einzelnen Klemmen 10mm und die Breite des Greifarms auf 55mm beschränkt. Zudem dürfen die Klemmen nicht beliebig weit geöffnet werden. Das Fahrzeug sollte so präzise als möglich positioniert werden. Wie die Positionierung umgesetzt wird, ist im Kapitel 4.9.1 genauer beschrieben.

#### 4.1.3 Begründung

Um den Greifarm zu betätigen, wurden verschiedene Varianten in Erwägung gezogen. So kamen unter anderem ein zahnradgetriebenes Gelenk, eine Seilwinde, eine Schraubung oder ein Servomotor in Frage. Aus all diesen Möglichkeiten wurde der Servomotor gewählt, da dieser das Mehrfache der benötigten Leistung liefern kann. Damit können Risiken, wie das Verlieren des Containers beim Heben, von Beginn an ausgeschlossen werden.

Die Wahl des Servomotors hat auch ihren Einfluss auf die Gestaltung des Greifers. Würde anstelle eines Servomotors eine Seilwinde gewählt, wäre ein Saugrohr die bessere Wahl und ein Förderband wäre besser mit einem Zahnrad anzutreiben.

In der untenstehenden Abbildung 3 ist ein Greifarm zu sehen. Die roten Pfeile indizieren die Positionen der beiden Servomotoren, welche einerseits zum Antrieb des Arms (1) und andererseits zum Öffnen und Schliessen der Klemmen benutzt werden (2). Die Berechnung der minimal erforderlichen Klemmkraft ist im Anhang im Kapitel C.9 zu finden.

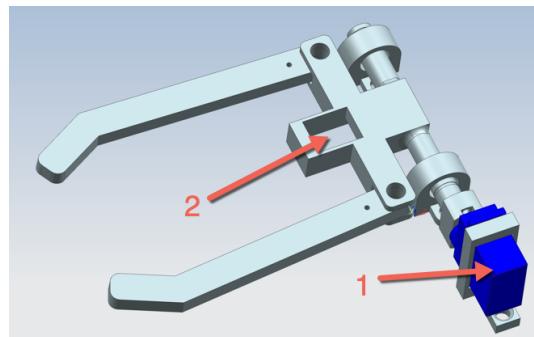


Abb. 3: Greifarm in der CAD-Ansicht

## 4.2 Fahrgestell und Lenkung

Das Lösungskonzept sieht ein Fahrzeug mit vier Rädern vor. Diese werden mittels einer Achsschenkellenkung gesteuert.

### 4.2.1 Funktionsweise

Erhält das Fahrzeug das Startsignal, wird der DC-Motor von der Steuerung eingeschaltet. Dieser treibt die Hinterachse an. Wird eine Abweichung des bisherigen Kurses zur Strasse erkannt, wird die Änderung an den Servomotor weitergegeben. Dieser lenkt die beiden Vorderräder über eine Achsschenkellenkung aus.

### 4.2.2 Detaillierte Beschreibung

Das Fahrgestell mit vier Rädern erhält einen ersten Aufbau, in dem Motor und Akku untergebracht sind. Darauf wird eine Grundplatte befestigt. Diese bietet Platz für die Mulde, die Sensoren, den Greifarm und die Rechenwerke.

Die Lenkung des Fahrzeugs erfolgt über die Vorderräder. Diese werden über eine Achsschenkellenkung gesteuert. Der Servomotor bewegt den Schenkel.

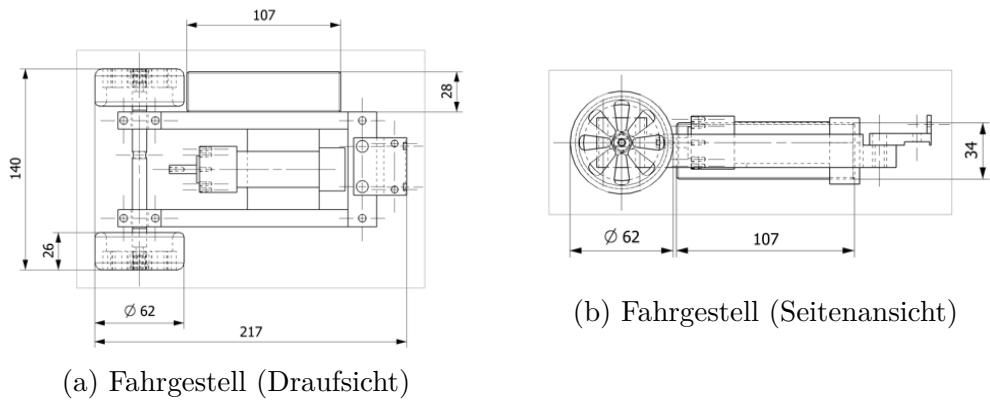


Abb. 4: Ansichten Fahrgestell

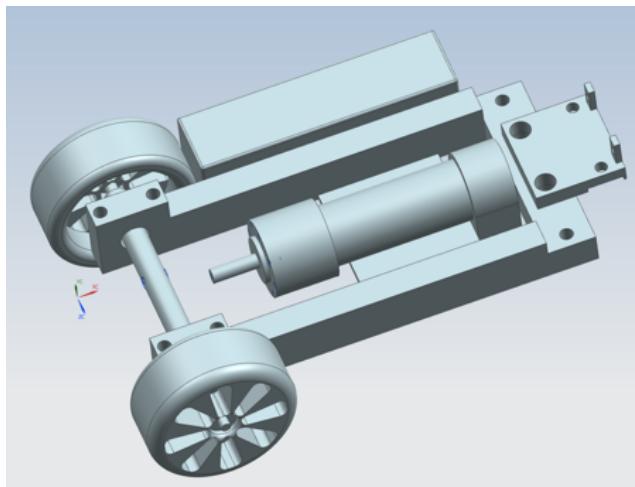


Abb. 5: Fahrgestell in der CAD-Ansicht

#### 4.2.3 Begründung

Das vierrädrige Fahrgestell wurde gewählt, da es eine zentrierte Schwerpunkt-lage besitzt. Es hat eine grössere Stabilität beim Aufsammeln der Container als ein Fahrgestell mit nur drei Rädern. Wird die Stabilität beider Fahrzeuge in einer Hanglage verglichen, welche das Aufsammeln eines Containers simuliert, so stellt sich schnell heraus, dass das Fahrzeug mit drei Rädern schneller ins Kippen käme als dessen Konkurrent mit vier Rädern.

Für die Lenkung eines autonomen Fahrzeuges mit vier Rädern stehen wiederum zwei Lösungsvorschläge zur Auswahl: Lösungsvorschlag A sieht eine Knicklenkung vor. Als Lösungsvorschlag B kommt eine Schenkellenkung in Frage. Hier wird ein Rad durch einen Hebelarm ausgelenkt.

Der mechanische Aufwand zur Realisierung von Lösungsvorschlag A ist grösser wie derjenige von B. Aus diesen Grund wurde für die Lenkung des auto-nomeren Fahrzeuges eine Schenkellenkung gewählt, welche einfach anzusteuern,

kostengünstig, genau und mechanisch einfach realisierbar ist.

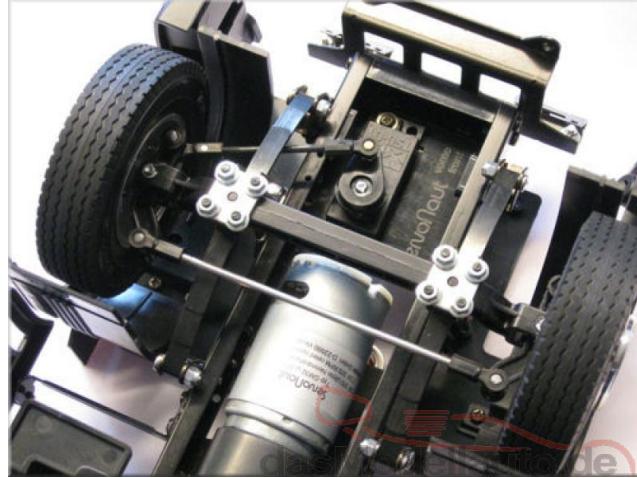


Abb. 6: Schenkellenkung des Fahrgestelles [1]

### 4.3 Entsorgung

Die Entsorgungsvorrichtung wurde in der Lösungsvariante 1 (Kapitel 2) als eine Mulde mit schrägem Boden und einer mit einem Servomotor betriebenen Klappe definiert. Es stellte sich jedoch heraus, dass die Dimensionierung zusammen mit dem Hebelarm schwierig wurde. Anstelle der ausgewählten Variante wurde entschieden, eine Kippmulde zu verwenden.

#### 4.3.1 Berechnung

##### Volumen Mulde

$$V_{Mulde} = 29 \text{ cm}^3$$

##### Dichte Plexiglas

$$\rho_{Plexi} = 1.3 \frac{\text{g}}{\text{cm}^3}$$

##### Masse Müll

$$m_{Muell} = 0.1 \text{ kg}$$

$$m_{Mulde} = \rho_{Plexi} \cdot V_{Mulde} = 1.3 \frac{\text{g}}{\text{cm}^3} \cdot 29 \text{ cm}^3 = 37.7 \text{ g}$$

$$m_{Gesamt} = m_{Muell} + m_{Mulde} = 0.1 \text{ kg} + 0.0377 \text{ kg} = 0.1377 \text{ kg}$$

$$F_{Gesamt} = m_{Gesamt} \cdot 9.81 \frac{\text{kg} \cdot \text{m}}{\text{s}^2} = 1.377 \text{ N}$$

##### Moment für Servo

$$\mathbf{M}_{\text{Servo}} = F_{Gesamt} \cdot sArm = 1.377 \text{ N} \cdot 9 \text{ cm} = \mathbf{12.393 \text{ Ncm}}$$

### 4.3.2 Funktionsweise

Der gesammelte Müll wird in einer Mulde mit schrägem Boden zum Entsorgungsbecken transportiert.

Die Mulde wird durch einen Servomotor über einen Hebel gekippt. Ist die Mulde leer, wird der Hebel wieder gesenkt. Die Mulde senkt sich durch die Schwerkraft.

### 4.3.3 Detaillierte Beschreibung

Die Mulde besitzt einen Müllauffang, in den der Greifarm die Container entleert. Dieser wird auf dem Fahrzeug festgeschraubt. Der Müllauffang besitzt eine Schräge von ca.  $30^\circ$ . Von dort rutscht der Müll hinab in das Innere der Mulde. Die Neigung des Bodens in der Mulde wird nach hinten geführt. Das verlagert den Schwerpunkt hinter die Drehachse und sorgt dafür, dass die Mulde sich wieder senkt, sobald der Servomotor sie nicht mehr hoch drückt. Der Servomotor mit Hebel ist unterhalb der Mulde angebracht. Die Mulde und der Müllauffang werden 3D-gedruckt. Für den Hebel kann auch Plexiglas verwendet werden.

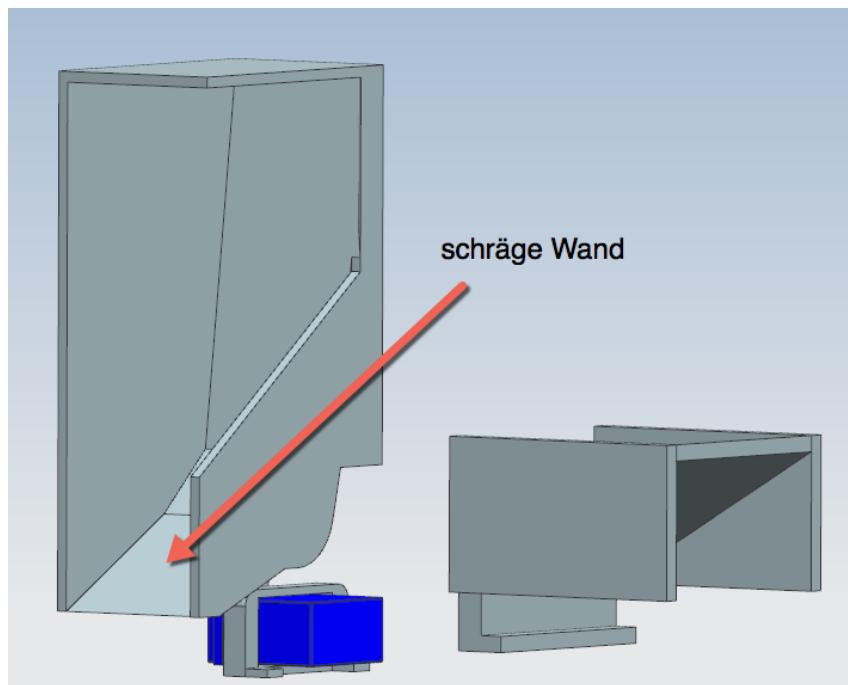


Abb. 7: Mulde gehoben

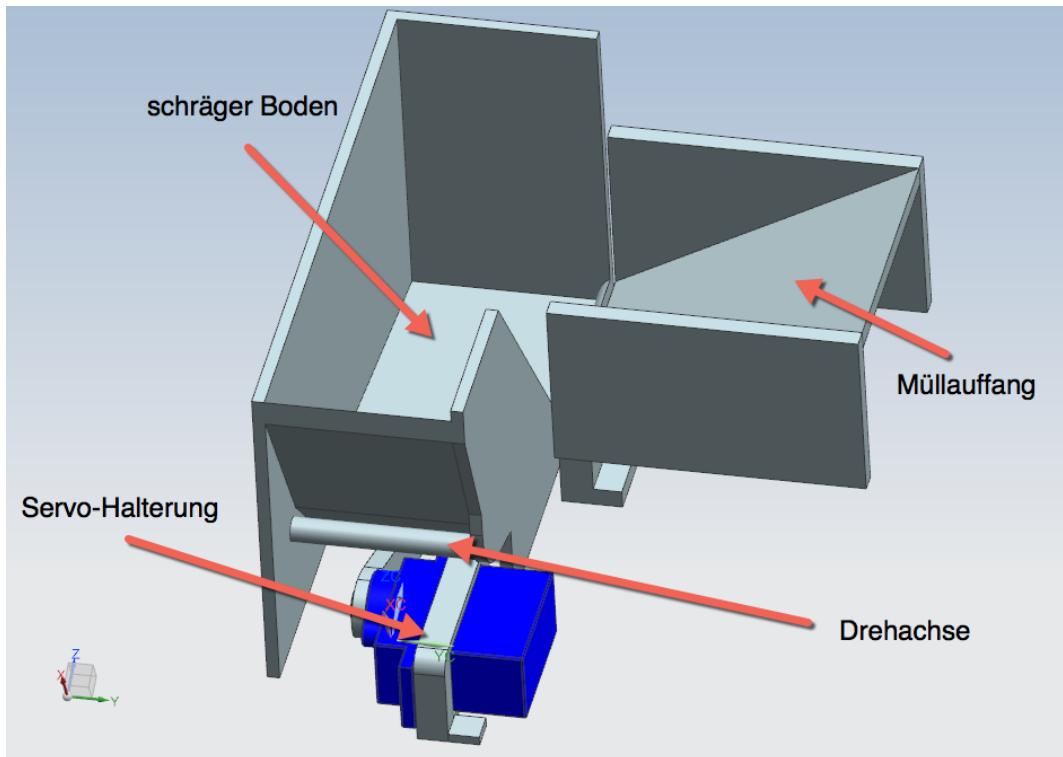


Abb. 8: Mulde gesenkt

#### 4.3.4 Begründung

Eine Kippmulde ist etwas langsamer als eine Mulde mit Klappe. Allerdings ist die Kippmulde mit dem Greifarm besser vereinbar, da die Mülleimer nicht gleich hoch gehoben werden müssen. Im Gegensatz zu einer Gewindespindel ist diese Mulde mit jedem Fahrwerk kombinierbar. Durch die schräge Wand, die im ungekippten Zustand eine Steigung von ca.  $60^{\circ}$  aufweist, wird sichergestellt, dass der Müll im Entsorgungsbecken landet und nicht zwischen Fahrzeug und Becken zu Boden fällt.

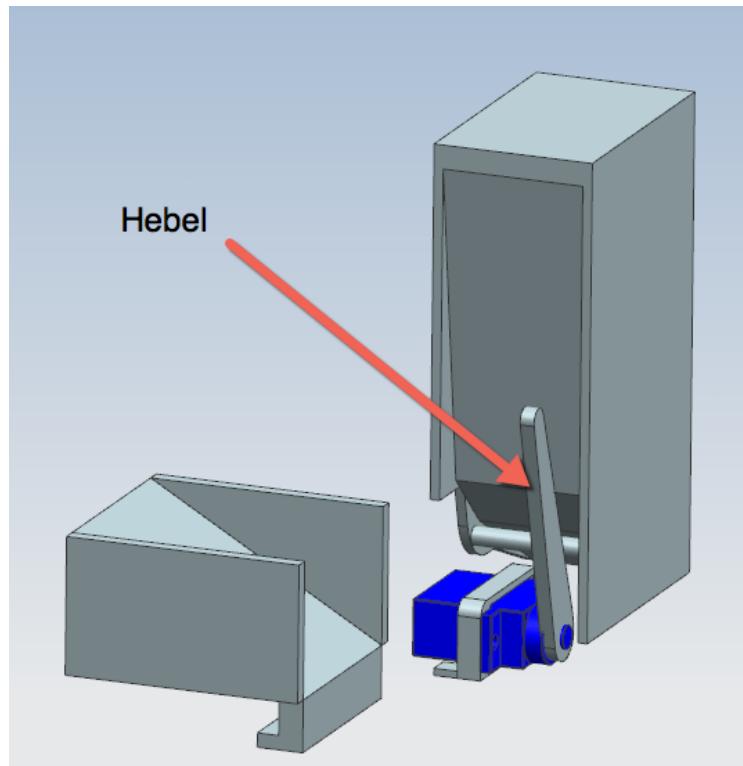


Abb. 9: Seitenansicht

## 4.4 Erkennung

Bei der Auswahl der Sensoren wurde der Fokus auf das Sortiment der Firma **Tinkerforge** gelegt.

Dieser Entscheid basiert darauf, dass die Firma Sensoren bereits auf Platinen (**Bricklet**) verbaut hat und zu dieser Hardware verschiedene API-Schnittstellen zur Verfügung stellt. So gestaltet sich das Auslesen und Konfigurieren der Sensoren einfacher. Zudem sind die API-Schnittstellen für über 15 Programmiersprachen<sup>1</sup> vorhanden.

### 4.4.1 Tinkerforge Hardware-Konzept

**Tinkerforge** ist eine deutsche Firma welche ein Teil der Hardware für dieses Projekt herstellt. Dazu gehören die **Bricks** und **Bricklets**.

Wie in Abbildung 10 gezeigt, können die einzelnen Teile zusammengesteckt werden, um den Funktionsumfang der Hardware zu erweitern. Grundsätzlich besteht das System aus drei Komponenten. In der Abbildung sind die **Bricks** auf der linken unteren Seite dargestellt. Die **Bricklets** sind auf der rechten Seite dargestellt.

---

<sup>1</sup><http://www.tinkerforge.com/de/doc/index.html#software>

**Brick:** Die **Bricks** sind das Herzstück des Systems. Sie stellen die Verbindung von Computer zur **Tinkerforge-Hardware** her. Diese Verbindung kann über USB oder falls eine passende **Master Extension** vorhanden ist, auch über WLAN und RS232 hergestellt werden.

An jedem **Brick** können vier **Bricklets** angeschlossen werden. Um noch mehr **Bricklets** anschliessen zu können, ist es möglich die **Bricks** und **Master Extensions** zu stapeln. Zwei übereinander gestapelte **Bricks** kommunizieren über zwei 30-Pin Stecker. Diese werden als Stapeldatenstecker und Stapelstromversorgungsstecker bezeichnet und sind sehr gut auf der Seite von **Tinkerforge**<sup>2</sup> beschrieben.

**Bricklet:** Die **Bricklets** stellen immer eine Funktion, einen Sensor oder einen Input, zur Verfügung. Diese werden per Zehnpinstecker an die **Bricks** angeschlossen. Die Verbindung basiert auf I<sup>2</sup>C und zusätzlichen Daten/Signal-Leitungen.

Weitere Informationen zum Tinkerforge Software-Konzept können in Kapitel 4.7.2 gefunden werden.

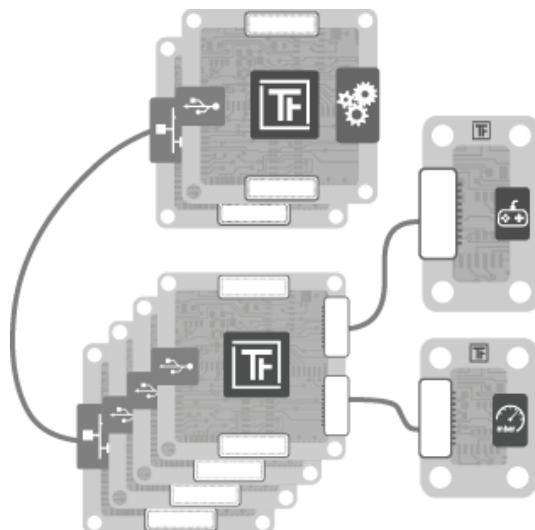


Abb. 10: Tinkerforge-Hardware Konzept [5]

#### 4.4.2 Farbsensor

Der Farbsensor, siehe Abbildung 11 wird für zwei Aufgaben benötigt. Zum Einen wird die von der Kamera erfasste Farbe nochmals verifiziert. Zum Anderen wird mit der Kamera die korrekte Position des Hebelarmes sicher gestellt. Dazu wird der Farbsensor kurz vor dem Hebelarm positioniert. So ist es möglich, das Fahrzeug anzuhalten, sobald der Sensor nicht mehr die Farbe des Containers misst.

<sup>2</sup>[http://www.tinkerforge.com/de/doc/Technical\\_Data.html#technical-data](http://www.tinkerforge.com/de/doc/Technical_Data.html#technical-data)

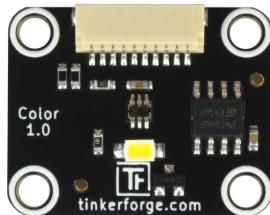


Abb. 11: Tinkerforge Color-Bricklet [3]

Der Color-Bricklet wird durch eine API gesteuert, die von [Tinkerforge](#) zur Verfügung gestellt wird. Dadurch ist das Konfigurieren und Auslesen des Sensors sehr komfortabel. So kann zum Beispiel durch den Befehl `color_get_color` die aktuellen Farbwerte des Sensors ausgelesen werden. Die Funktion liefert die gemessene Farbe als RGB-Value zurück.

Damit die Farberkennung auch bei verschiedenen Lichtverhältnissen noch gut funktioniert, können die Integration-Time und der Gain-Value eingestellt werden. Zudem enthält die Platine eine SMD-LED, die es ermöglicht den Messbereich zu erhellen. Diese kann per API an- und ausgeschaltet werden. Die Dokumentation des Tests ist im Anhang Kapitel [C.4.1](#) zu finden.

In der Tabelle 4 sind die Technischen Daten des verwendeten Sensors aufgeführt. Zusätzliche Informationen zur Steuerung der Platine sind im Kapitel [4.7.2](#) zu finden.

Eigenschaft	Wert
Sensor	TCS34725
Dynamikbereich	3800000:1
Auflösung Farbe (R,G,B,C)	jeweils 16Bit (0-65535)
Auflösung Farbtemperatur	16Bit (0-65535)
Auflösung Helligkeit	16Bit (0-65535)
Abmessungen (B x T x H)	25x20x5mm
Gewicht	2g

Tab. 4: Technische Daten des Color-Bricklets [3]

#### 4.4.3 Ultraschallsensor

Der Ultraschallsensor, siehe Abbildung 12, wird genutzt um ein Fahrzeug mit Rechtsvortritt zu erkennen: Befindet sich das Fahrzeug vor der Kreuzung, soll

der Ultraschallsensor überprüfen, ob die Fahrbahn frei und somit passierbar ist. Ein Testaufbau ist im Anhang Kapitel C.4.3 ersichtlich.



Abb. 12: Tinkerforge Distance-US-Bricklet [4]

Wie der Color-Bricklet kann auch der Distance-US-Bricklet über die API von [Tinkerforge](#) angesprochen werden.

Mit dem ausgewählten Sensor können Distanzen zwischen 2cm und 400cm gemessen werden. Zu beachten gilt es, dass die gemessenen Werte einheitenlos sind. Dies bedeutet, dass anstelle von einer Milimeter- oder Centimeter-Angabe für die Entfernung lediglich einen Wert zwischen 0 - 4096 erhält. Dieser Wert kommt durch die Auflösung des Sensors von 12bit zustande.

Der Grund, dass die Entfernung nicht direkt als mm angegeben wird, liegt daran, dass das Verhältnis von gemessenem Entfernungswert zu wirklicher Entfernung vom exakten Wert der 5V Versorgungsspannung abhängt. Abweichungen in der Versorgungsspannung führen zu Abweichungen in den gemessenen Entfernungswerten. Deshalb müsste die Versorgungsspannung genau gemessen werden können. In der Tabelle 5 sind die technischen Daten des Sensors aufgelistet.

Eigenschaft	Wert
Sensor	HC-SR04
Stromverbrauch	8mA
Entfernung	2cm-400cm, 12Bit Auflösung
Messwinkel	15°
Aktualisierungsrate	40Hz
Abmessungen (B x T x H)	45x20x30mm
Gewicht	13g

Tab. 5: Eigenschaften Ultraschall-Sensor

## 4.5 Energieversorgung

In diesem Kapitel wird auf die Stromversorgung eingegangen. Es wird aufgezeigt, wie die Akkukapazität berechnet wurde und wie die einzelnen Spannungen generiert werden.

### 4.5.1 Akku

Anhand der Konzeptfindung wurde entschieden, dass ein Lithium-Polymer Akku eingesetzt wird. Für die Dimensionierung wurden alle Verbraucher aufgelistet und ihre Leistungen berechnet. In unserer Anforderungsliste ist als Festanforderung eine Akkulaufzeit von mindestens 8 Minuten definiert. Als Wunschzeit wurden 30 Minuten definiert. Für die Berechnung der Akkukapazität wurde eine Akkulaufzeit von 20 Minuten bzw. 0.35h angenommen.

Komponente	Leistung [W]
Raspberry Pi 1	5
Raspberry Pi 2	5
Antriebsmotor	60
Servo Lenkung	6
Servo Greifer	5
Servo Heber	5
Servo Mulde	5
Sensoren	5
Webcam	7
	<b>102</b> W
<b>Strom</b>	7.29 A @ 14V Akkuspannung
<b>Akkukapazität</b>	<b>2550.00 mA/h @ 0.35h</b>

Tab. 6: Leistung der Komponenten

Für die meisten Komponenten gab es keine wirkliche Leistungsangabe. Meist musste diese selbst berechnet oder gar geschätzt werden. Auch werden nicht alle Komponenten die ganze Zeit voll belastet, was eine geringere Leistungsaufnahme zur Folge hat. Zum Beispiel werden die Servo für den Greifer, Heber und Mulde nie gleichzeitig mit dem Antriebsmotor laufen. Anhand die-

sen Berechnungen und Überlegungen muss der Akku mindestens 20 Minuten halten.

#### 4.5.2 Akkuüberwachung

Wie schon in der Konzeptfindung erwähnt, sind die Lithium-Polymer Akkus sehr empfindlich gegenüber Tiefentladung. Dabei muss der Akku überwacht werden und sobald die Zellspannung unter 3.6V sinkt, abgeschaltet werden. Diese Funktion wird eine Ladeüberwachungsschaltung mit einem Operationsverstärker übernehmen. Die Schaltung ist im Bild unten ersichtlich. Mit dem Potentiometer kann die gewünschte Schaltspannung eingestellt werden. Sobald die Akkuspannung darunter sinkt, wird ein digitales High Signal am GPIO Eingang des Raspberry Pi angelegt (Signal  $U_a$  aus der Grafik). Dieser gibt den Alarm weiter an das zweite Raspberry Pi. Beide Rechenwerke führen noch Sicherungen durch und fahren herunter. Ein Testaufbau wird aus zeitlichen Gründen erst in PREN2 durchgeführt. Die Widerstandswerte und Zenerdiode wurden erst schematisch eingesetzt und noch nicht berechnet.

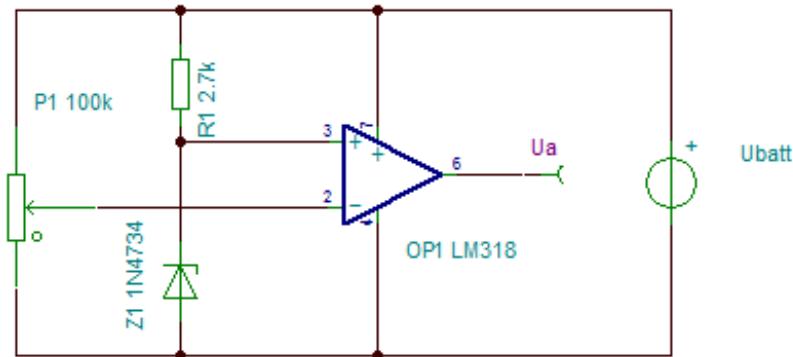


Abb. 13: Schaltung der Akkuüberwachung

#### 4.5.3 Spannungswandlung

Der Akku liefert bei voller Ladung 16.8V. Bei spätestens 14.4V muss der Akku wieder aufgeladen werden. Somit liegt die Eingangsspannung zwischen 14.4 und 16.8V. Aus dieser Spannung muss die 5V Speisespannung für das Raspberry und die 7V Spannung für die Servos generiert werden. Die Servos werden über das Servo Brick gesteuert. Dieses benötigt eine 7V Speisung, um die angeschlossenen Servos nachher mit 6V zu speisen. Die Spannungswandlung und Stabilisierung wird mit Buck-Convertern realisiert. Es gibt jeweils für die 5V Spannung und die 7V Spannung einen Buck-Converter Schaltung. Die maxi-

male Belastung der Buck-Converter wurde aus der Tabelle 6 Leistungen der Komponenten entnommen.

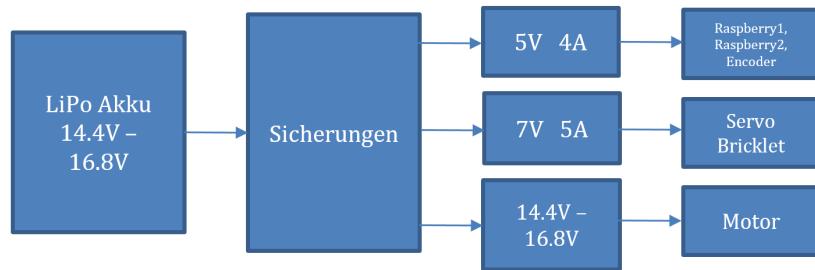


Abb. 14: Blockschaltbild Spannungen

Die Buck-Converter haben einen sehr hohen Wirkungsgrad und sind preiswert. Die Dimensionierung einer solchen Schaltung ist jedoch sehr aufwendig. Deshalb wurde das Webench-Tool von Texas Instruments eingesetzt. Mit diesem Tool kann der Eingangsspannungsbereich, die Ausgangsspannung und der Ausgangsstrom vorgegeben werden und das Tool gibt verschiedene Schaltungsvorschläge. Die daraus resultierenden Schemata sind im Anhang C.11 ersichtlich.

## 4.6 Motoren

Im folgenden Kapitel wird auf die Auswahl des Motors und Motorentreiber eingegangen. Auch wird hier der Encoder beschrieben und der Regelkreis gezeigt.

### 4.6.1 Auswahl

In der Auswahl standen lange zwei verschiedene DC Motoren. In der untenstehenden Tabelle sind die Motoren einander gegenübergestellt:

	<b>Modelcraft RB350050</b>	<b>Maxon RE30</b>
<b>Bauart</b>	DC Motor	DC Motor
<b>Getriebe</b>	50:1	4.8:1
<b>Encoder</b>	Nein	Ja, 512 Pulse/U
<b>Betriebsspannung</b>	12 V	24 V
<b>Leistung</b>	10 W	60 W
<b>Drehmoment</b>	5.39 Nm	85.6 mNm
<b>Dimensionen</b>	106x36 mm	125x30 mm

Tab. 7: Vergleich DC-Motoren

Ein sehr wichtiges Kriterium für die Regelung eines Motors ist die Rückführung. Beim Maxon Motor ist ein Encoder direkt auf die Motorenwelle geschraubt und liefert 512 Pulse/U. Der Modelcraft Motor ist nicht mit einem Encoder lieferbar. Encoder für die nachträgliche Montage sind grundsätzlich sehr teuer und passen kaum in das vorgegebene Budget. Weiter kommt dazu, dass der Encoder zusätzlich mechanisch auf die Antriebswelle angekoppelt werden muss. Dies ist aus Platzgründen aufwendig. Deshalb fiel die Wahl auf den Maxon Motor mit integriertem Encoder.

#### 4.6.2 Motorentreiber

Um einen DC Motor mit unterschiedlichen Geschwindigkeiten drehen zu lassen, werden Motorentreiber verwendet. Mit einem Motorentreiber kann die Drehrichtung mittels digitalen Signalen und die Geschwindigkeit mit einem PWM gesteuert werden. Auch werden Motorentreiber verwendet, um die benötigte Leistung zu liefern. Der Treiber soll eine Motorenspannung von 12V – 24V unterstützen und einen Bremsmodus besitzen. Von den Leistungen sollte der Treiber eigentlich mindestens für die Motorenleistung ausgelegt sein. Da der Motor 60W Bauleistung hat, würde der Treiber mit einem sehr großen Kühlkörper ausgestattet. Da der Platz auf dem Roboter begrenzt ist, wird ein Treiber mit 50W eingesetzt, welcher bautechnisch kleiner ausfällt. Es wird das Modell MR001 von M2 Microbot eingesetzt. Der Treiber kann über drei Signale gesteuert werden. In der untenstehenden Tabelle sind die Funktionen ersichtlich.

Input			Output
E1	A1	B1	
1	1	1	Bremsbetrieb
1	0	0	Bremsbetrieb
1	1	0	Vorwärts
1	0	1	Rückwärts
0	X	X	Freilauf

Tab. 8: Steuerzustände Motorentreiber



Abb. 15: Motorentreiber

#### 4.6.3 Encoder

Der Encoder, welcher am Maxon Motor befestigt ist, liefert 512 Impulse/Umdrehung. Diese Pulse müssen alle aufgezeichnet werden, damit genau gefahren werden kann. Ausgehend von der geschätzten maximalen Geschwindigkeit, ergibt sich eine Motorenendrehzahl von 600 Umdrehungen pro Sekunde. Dies ergäbe 300'000 Pulse pro Sekunde, was eine sehr grosse Abtastrate zur Folge hätte. Deshalb wird das Schnittstellen-IC HCTL 2022 verwendet. Dieses IC beinhaltet einen 32bit Counter, welche vom Encoder aufwärts/herunter gezählt wird. Nun kann mit dem Raspberry Pi periodisch die Werte herausgelesen werden. Der Zeitpunkt dazu kann selber festgelegt werden, ohne dass Pulse verloren gehen.

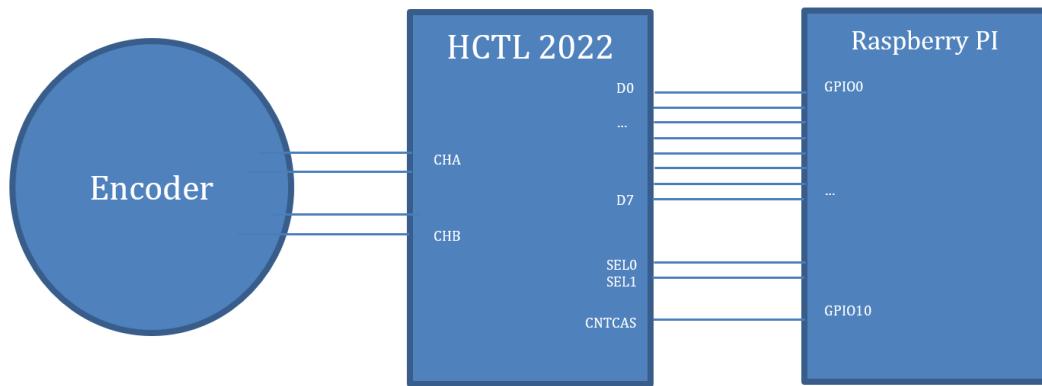


Abb. 16: Aufbau Encoder Rückführung

#### 4.6.4 Motorensteuerung und Regelung

Um mit dem Fahrzeug Positions- und Geschwindigkeitsgeregelt fahren zu können, wird ein Regler benötigt. Dazu wird eine Sollgeschwindigkeit vorgegeben und die Geschwindigkeit mithilfe des Encoders wieder eingelesen. Der Regelkreis ist in der untenstehenden Grafik sichtbar.

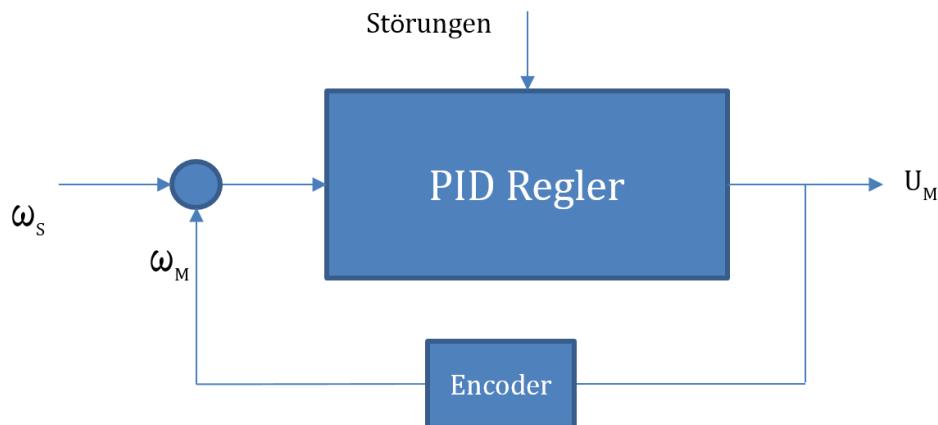


Abb. 17: Regelkreis

Meistens ist das Fahrzeug im geschwindigkeitsgeregelten Zustand unterwegs. Die Sollgeschwindigkeit wird von der Steuerung vorgegeben und danach über den PID-Regler geregelt. Wenn die Erkennungssoftware jedoch einen Container sieht, kann sie die Distanz zum Container abschätzen. Nun wird Positionsgeregelt gefahren. Das heißt das Fahrzeug fährt eine definierte Strecke ab. Dies kann gemacht werden, indem die Pulse gezählt werden. Anhand der Berechnung unten, ergeben sich 355 Pulse für eine Distanz von 1cm.

$$1 \text{ Umdrehung} \cong \frac{U_{Rad}}{\text{Untersetzung}} = \frac{2 \cdot \pi \cdot 3.3\text{cm}}{3 \cdot 4.8} = 1.44\text{cm}$$

$$1 \text{ Umdrehung} = 512 \text{ Puls} = 1.44cm$$

$$\frac{1}{1.44cm} \cdot 512 = 355.6 \text{ Puls für } 1\text{cm}$$

## 4.7 Embedded System

Zum Steuern des Roboters wurde die Software unterteilt. Der erste Teil dient der Steuerung und Auswertung der Kamera. Der zweite Teil der Software wird für das Steuern der Sensoren, der Servos und dem Motor benutzt.

In diesem Kapitel wird näher auf den Teil der Software eingegangen, der die Steuerung der Sensoren und Aktoren koordiniert. Weitere Informationen zum anderen Software-Teil sind im Kapitel 4.9 zu finden..

### 4.7.1 Hardware

In der Abbildung 18 wird gezeigt, wie das Komplettsystem ungefähr aussehen wird. Für die Konfiguration und Überwachung der Embedded Software, wird es ein Computerprogramm geben. Wie bereits erwähnt, wird die Software für die Steuerung des Roboters in zwei Teile unterteilt. Auf der Abbildung ist zu sehen, dass diese auch auf getrennten Computern läuft. Die Steuerung und Auswertung der Kamera wird auf einem Raspberry Pi 2 gemacht. Die Embedded Software läuft auf einem zweiten Raspberry Pi B+. Da es sich um zwei getrennte Systeme handelt, wird das Kommunikationsprotokoll in Kapitel 4.8 beschrieben.

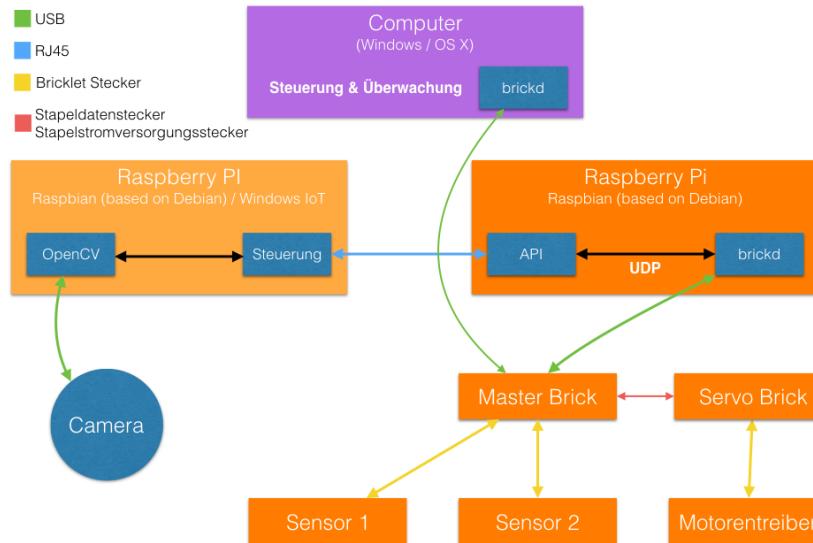


Abb. 18: Grobübersicht des Produktivsystems

**Raspberry Pi 2** In der Abbildung 18 ist das Raspberry Pi 2 als gelbes Rechteck dargestellt. Auf diesem läuft die Software, welche die Kamerabilder auswertet und dementsprechende Befehle an die anderen Komponenten der Software schickt. Detaillierte Informationen dazu sind im Kapitel 4.9 zu finden.

**Raspberry Pi 1 B+** In der Abbildung 18 ist das Raspberry Pi B+ als oranges Rechteck dargestellt. Auf diesem läuft zum Einen der **brickd**. Dabei handelt es sich um einen Deamon, der ein Loopback-Interface zur Verfügung stellt an welches Befehle für die Tinkerforge-Hardware senden kann. Diese werden danach über USB an den **Master-Brick** gesendet.

Zum Anderen läuft eine selbst entwickelte Software welche die Tinkerforge-Hardware verwaltet. Diese wird näher im Kapitel 4.7.2 beschrieben.

**Computer** Damit das Debuggen und Überwachen des Roboters einfacher ist, wird zusätzlich eine Software geschrieben, welche es ermöglicht die verschiedenen Werte der Hard- und Softwareteile anzeigen zu lassen.

#### 4.7.2 API

Für die zentrale Verwaltung aller Tinkerforge-Hardware und den zusätzlichen Controllern, wird eine Software geschrieben, welche über eine Netzwerkinterface gesteuert werden kann. Im Anhang C.12 ist der aktuellen Source-Code zu finden.

Die Software ist so aufgebaut, dass es eine Komponente gibt, welche den empfangenen Netzwerktraffic auswertet und die erkannten Befehle an die ver-

schiedenen Steuerungsthreads weiterleitet. Bei den Steuerungssthreads gibt es zum Beispiel ein Thread, welcher den Motor steuert oder ein anderer Thread, welches die Sensoren steuert. Damit die Thread untereinander kommunizieren können, bekommt jeder Thread eine Message Queue. So wird gewährleistet, dass keine Befehle oder Daten verloren gehen. Die Abbildung 19 zeigt die schematische Darstellung der Software.

Die gesammte Software, ausgenommen die **Tinkerforge**-Bindings, wird in C++ geschrieben. Als Buildsystem wird cmake verwendet.

Der aktuelle Stand der Software erlaubt es den Servo Brick, den Color Bricklet und den Distance US Bricklet zu steuern. Die Software wird während dem PREN 2 weiterentwickelt.

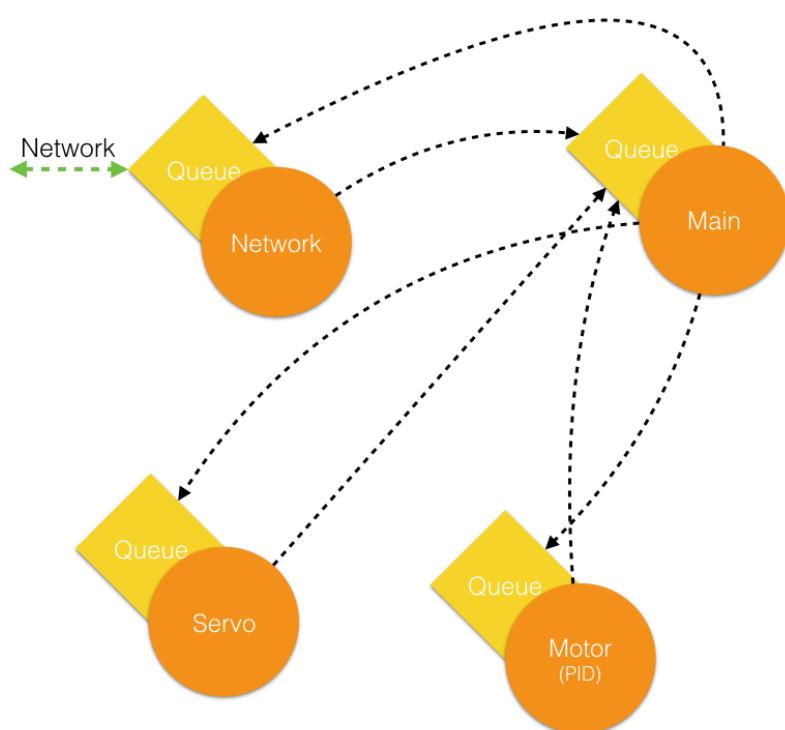


Abb. 19: konzeptioneller Aufbau der API

**Tinkerforge** Für die Kommunikation mit den Bricks und Bricklets, stellt **Tinkerforge** drei Programme und viele Bindings für die Softwareentwicklung bereit. Zur benötigten Software gehört der **brickd**. Diese Software stellt einen Daemon zur Verfügung, welche eine lokale IP Verbindung akzeptiert und Befehle für die **Bricks** und **Bricklets**.

Sobald die Befehle für die Bricks und Bricklets beim **brickd** angekommen sind, werden diese an den **Master-Brick** per USB, Wifi oder RS232 weitergeleitet. Dieser verteilt die Befehle auf den dazugehörigen **Brick** oder **Bricklet**.

Zum Testen und Updaten der angeschlossenen Bricks und Bricklets, stellt **Tinkerforge** den **brickv** zur Verfügung. Diese Software kann alle Bricks und Bricklets auslesen und steuern.

Die API-Bindings, welche von **Tinkerforge** zur Verfügung gestellt werden, gibt es für über 15 Programmiersprachen. Diese spezifizieren, wie die einzelnen Sensoren und Aktoren angesprochen werden können. Auch stellt **Tinkerforge** einfache Beispiele bereit, um den Anfang mit der Hardware zu vereinfachen.

#### 4.7.3 Produktiv- und Entwicklungssystem

Der Systemüberblick während der Entwicklung und später im Produktiveinsatz unterscheidet sich in mehreren Punkten. In der Abbildung 20 und 21 werden die beiden Systeme genau gezeigt.

**Entwicklung** Während der Entwicklung der Software ist es wichtig eine einfache und gute Möglichkeit zu haben, die Software zu debuggen. Für diesen Zweck unterscheidet sich der Entwicklungsaufbau grundlegend vom Produktivaufbau. Der Aufbau ist in Abbildung 20 aufgezeigt.

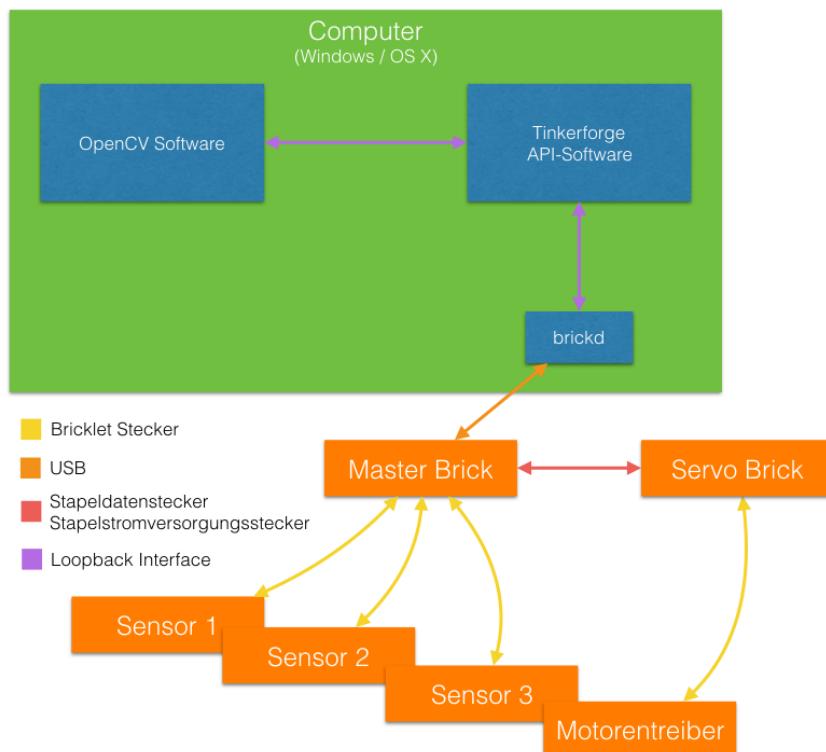


Abb. 20: Entwicklungssystems

Um die Software zu entwickeln, werden beide Software-Komponenten, welche normalerweise auf den **Raspberry Pis** aufgeteilt würden, auf dem selben

Computer laufen gelassen. Um die TCP-Verbindung zwischen den beiden Komponenten zu simulieren, wird das Loopback-Interface benutzt. Da die Bricks von **Tinkerforge** am Anfang dafür gedacht waren, vom Computer aus gesteuert zu werden, ist es kein Problem auch den `brickd` auf dem Computer laufen zu lassen. Das in Abbildung als "OpenCV Software" bezeichnete Rechteck, steht für die Softwarekomponente, welche die Auswertung der Kamerabilder vollzieht.

**Deployment** Für das Deployment werden alle nötigen Schritte in einer Anleitung beschrieben. So wird das Deployment immer exakt gleich ausgeführt. Eine weitere Software zeigt während des Betriebs des Roboters eine Übersicht der Sensorwerte an und schreibt ein Logfile. Damit können allfällige Störungen nachvollzogen werden.

**Produktiver Betrieb** In Abbildung 21 ist der entgültige Aufbau des Systems gezeigt.

Im produktiven Betrieb läuft die Software jeweils auf den beiden **Raspberry Pis**. Als Überwachung kann ein Computer via Wifi auf das **Raspberry Pi** verbinden und in einer Software die Sensorwerte und andere Daten darstellen.

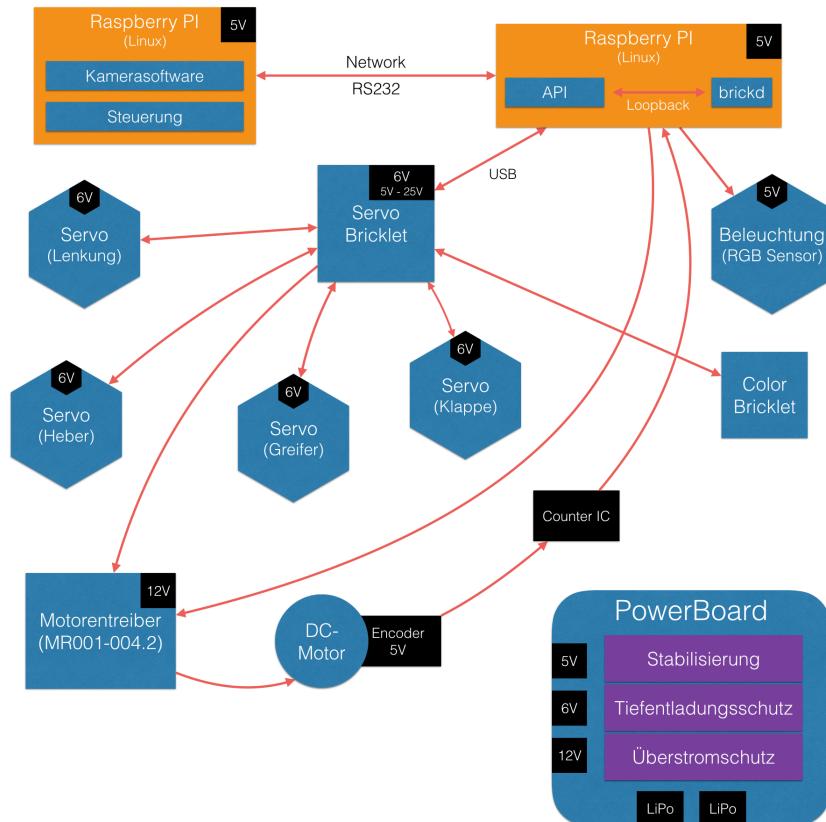


Abb. 21: Produktivsystems

## 4.8 Kommunikation

Um genügend Rechenleistung für das Auswerten der Kamera und das Steuern der Sensoren und Aktoren zu haben, werden zwei **Raspberry Pis** eingesetzt. Somit werden auch zwei eigenständige Programme laufen. Ein Programm wird für die Auswertung der Kamera und der Steuerung des gesamten Ablaufs zuständig sein. Das zweite Programm wird für die Steuerung der Sensoren und Aktoren zuständig sein.

Die beiden Programme werden über eine Netzwerkverbindung untereinander Daten austauschen. Damit dies möglich ist, wurde ein Kommunikationsprotokoll definiert.

### 4.8.1 Protokoll

Als Grundlage für die Kommunikation zwischen den beiden **Raspberry Pis** wird TCP/IP genutzt. Die Abbildung 22 zeigt die grundlegenden Packet Struktur für die Kommunikation zwischen den beiden **Raspberry Pis**.

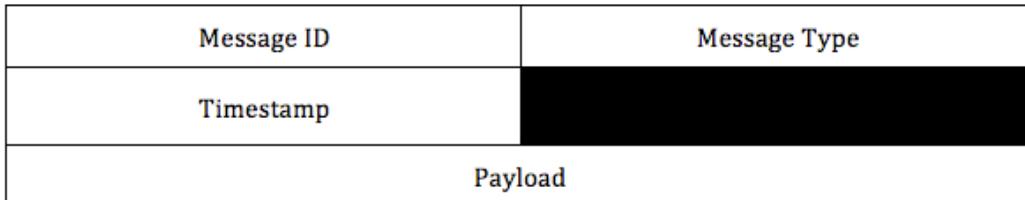


Abb. 22: Paket Struktur

Field	Description	Data Type
Message ID	Inkrementierender Wert	uint16
Message Type	ID für die Art der Nachricht (Kapitel 4.8.2)	
Timestamp	Unix timestamp	uint64
Payload	Enthält einen der definierten Payloads (Kapitel 4.8.3)	

Tab. 9: Beschreibung der Felder im Paket.

### 4.8.2 Message Types

Um die Art der Nachricht zu erkennen, wird ein Message Type mitgeschickt. Aus dieser ID kann auf den Payload-Typ geschlossen werden. Der Message Ty-

pe besteht aus einer vierstelligen Nummer. Der Aufbau wird in der Abbildung 23 gezeigt.

Board Identifier	Command Group Identifier	Command ID	
0-9	0-9	0-9	0-9

Abb. 23: Message Type ID Aufbau

---

Field	Description
Board Identifier	Identifiziert das Board (Raspberry Pi) welches die Nachricht gesendet hat.
Command Group Identifier	Jede Nachricht gehört zu einer Gruppe. (Drive, Garbage Collection, Garbage Disposal, System State Messages)
Command ID	Identifiziert das Kommando.

---

Tab. 10: Beschreibung der Felder im Paket.

Die verschiedenen Nachrichten, welche verschickt werden, sind in verschiedene Command Groups aufgeteilt. Alle mögliche Command Groups sind in Tabelle 11 aufgelistet.

---

Name	Wert
Drive	1
Garbage Collection	2
Garbage Disposal	3
Street Obstacle Detection	4
System State Message	5

---

Tab. 11: Auflistung aller Command Groups.

In der Tabelle 12 sind alle mögliche Nachrichten welche von einem der beiden Programme versendet werden können aufgelistet.

#### 4.8.3 Payloads

Bei den verschiedenen Message Types müssen auch verschiedene Daten zwischen den beiden Raspberry Pis übertragen werden. Um anzugeben, welche

Art von Daten übertragen werden, wurden verschiedene Typen von Payloads definiert. Diese geben an, welche Art von Daten übertragen werden.

**Empty Payload** Dieser Payload ist leer. Es werden keine zusätzlichen Daten übertragen.

**Speed Payload** Bei diesem Payload wird die neue Geschwindigkeit für den Antrieb übertragen. Eine Rückwärtsbewegung wird mit einem negativen Wert und eine Vorwärtsbewegung mit einem positiven Wert definiert.

**Angle Payload** Mit diesem Payload wird die Ausrichtung der Räder übertragen. Dabei wird ein Winkel zwischen -90 und +90 Grad übertragen.

**Container Payload** Dieser Payload enthält die Containerfarbe als RGB-Wert.

## 4.9 Software

### 4.9.1 Spurhaltung

Die Spurhaltung wird mit einer Kamera realisiert. Zu Beginn wurde eine Stereokamera in Betracht gezogen, welche sich nach einem ausgiebigen Test (Anhang Kapitel C.1) allerdings nicht als optimale Lösung herausstellte. Danach wurden weitere Tests mit einer einzelnen Kamera durchgeführt. Bei dem ersten Test mit dem LineDetection-Algorithmus (Anhang Kapitel C.2) stellte sich heraus, dass die Kurven ein Problem darstellen. Die Spurhaltung wird deshalb mittels Blob Detection Algorithmus realisiert.

**Blob Detection Algorithmus** Zu Beginn wird über das Kamerabild ein Farbfilter gelegt (Abbildung 24). Danach werden zusammenhängende Bereiche als Blob erkannt. Für diese Blobs wird automatisch der Mittelpunkt berechnet. Zudem können mit Hilfe der Abmessungen zu breite/lange Objekte grundsätzlich ausgeschlossen werden. Im folgenden Codeausschnitt ist ein Beispiel einer solchen Blob-Filterung aufgezeigt.

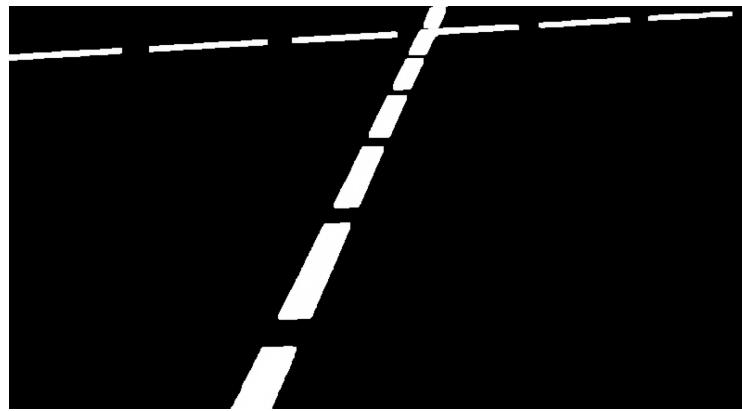


Abb. 24: Farbfilter auf Fahrbahn

**Filtern von Bereichen (SimpleCV)**

```

colorDist = imgsmall.colorDistance(SimpleCV.Color.BLACK).dilate(2)
segmented = colorDist.stretch(240, 255)
blobs = segmented.findBlobs(-1, 600, 6500)

```

Für die Spurhaltung werden nun die Mittelpunkte verbunden. Der Lenkwinkel für die Schnittstelle zur Motorensteuerung wird über die Verbindung der Mittelpunkte berechnet. Da das Fahrzeug aber nicht auf der Mittellinie fährt, soll die Fahrlinie entsprechend der Fahrposition nach rechts verschoben werden. Dazu werden Geraden im 90° Winkel an die Mittelpunkte gelegt. Mit diesen Geraden werden nun die Mittelpunkte der Linien auf die Mitte der Fahrspur projiziert. Somit kann der Fahrtwinkel auf der Fahrspur berechnen werden.

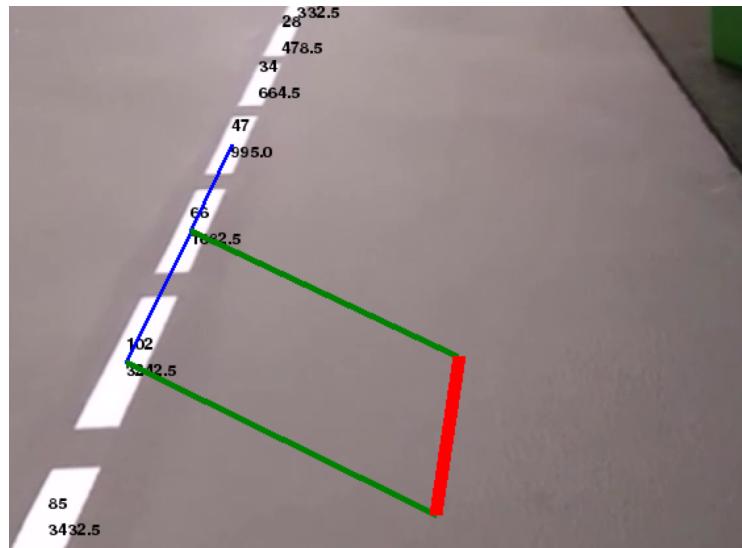


Abb. 25: Fahrlinie auf Fahrbahn abgebildet

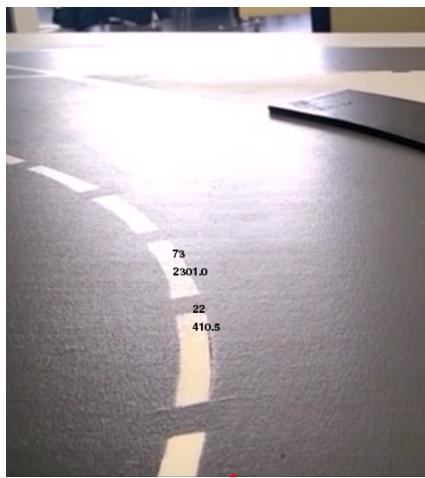
Damit die Blobs in der richtigen Reihenfolge verbunden werden wird ein

Referenzpunkt am unteren Rand des Bildes erstellt. Die Blobs werden dann sortiert nach Abstand zu diesem Punkt. Diese Sortierung wird mit folgendem Code erreicht.

#### Blobs nach Abstand ordnen (SimpleCV)

```
1 blobs_sort = blobs.sortDistance(point=(260, imgsmall.height))
```

Ein noch nicht gelöstes Risiko dieser Methode ist direkter Lichteinfall auf die Strasse. Dann werden die Farbwerte der Linien verändert und die Strasse reflektiert teilweise so stark, dass die Beleuchtung als weiss erkannt wird (Abbildung 26). Massnahmen dazu sind in der Risikoliste eingetragen.



(a) Orginalbild bei Lichteinfall



(b) Farbfiltersicht bei Lichteinfall

Abb. 26: Aufnahmen Fahrbahn bei Sonnenlicht

#### 4.9.2 Erkennung Rechtsvortritt

An der Kreuzung der Fahrbahn muss einem von rechts nahenden Fahrzeug Vortritt gewährt werden. Ein von rechts kommendes Fahrzeug wird via Ultraschall, welcher vorne rechts am eigenen Fahrzeug befestigt wird, ermittelt. Der Ultraschall-Sensor wird von der Spurhaltungs-Komponente gestartet, sobald sich das eigene Fahrzeug der Kreuzung nähert. Erkennt dieser ein von rechts kommendes Objekt, wird der Motor blockiert. Sobald der Sensor nichts mehr erkennt, wird wieder zur ursprünglichen Geschwindigkeit gewechselt. Setzt das andere Fahrzeug den Rechtsvortritt nicht um, wird nach einer bestimmten Zeit (Timeout) weitergefahren. Somit können Deadlocks vermieden werden.

#### 4.9.3 Erkennung Container

Die Container-Erkennung kann in zwei Phasen eingeteilt werden.

- Phase 1 macht eine grobe Vorerkennung durch Bildverarbeitung
- Phase 2 ermittelt die genaue Containerposition durch einen RGB-Sensor

Sobald ein Container auf einer Distanz von circa 30cm erkannt wird, wird der RGB-Sensor auf der rechten Seite des Fahrzeuges gestartet. Die Geschwindigkeit des Fahrzeuges wird gedrosselt und der RGB-Sensor analysiert die erhaltenen Farbwerte.

Sobald die Farbwerte während einer Zeitdauer von  $t = \frac{s}{v}$  (wobei s die Containerbreite und v die momentane Geschwindigkeit des Fahrzeuges ist) die beiden gültigen Farbcodes (blau oder grün) misst, kann die Position des Containers genau ermittelt und der Container somit aufgeladen werden. In diesem Kapitel wird die Erkennung der Container mittels Kamera beschrieben. Der komplette Sourcecode ist im Anhang C.13 aufgelistet.

### Erkennung in 4 Schritten

**Schritt 1: Farbdistanz zur Referenzfarbe ermitteln** Als erstes wird in auf dem Ursprungsbild die Farbdistanz zur Referenzfarbe (hier blau) errechnet, welche in Abbildung 27 dargestellt ist.

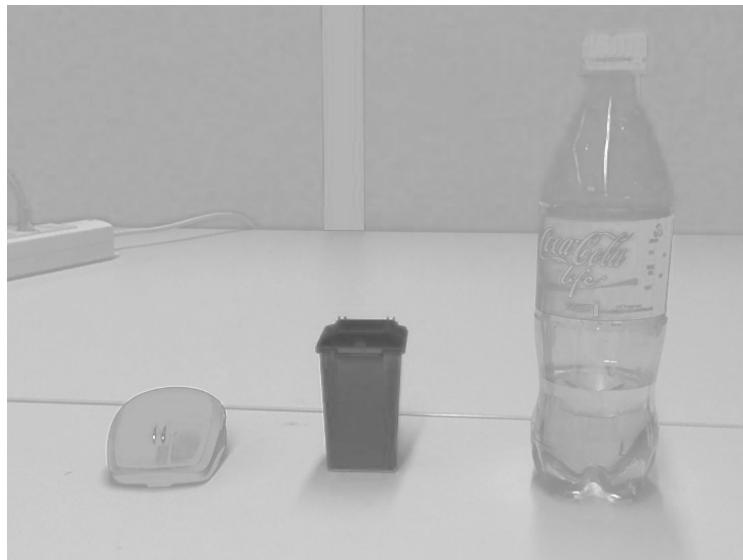


Abb. 27: Farbdistanz zur Referenzfarbe

### Codesnippet (SimpleCV)

```
1 blue_distance = img.colorDistance(Color.BLUE)
```

**Schritt 2: Binärbild erstellen** Anhand der ermittelten Farbdistanzen wird ein Binärbild erstellt (siehe Abbildung 28). Da die Farbdistanz beim Container am kleinsten ist, bleibt dieser als einziges Objekt übrig.



Abb. 28: Binärbild des Containers

#### Codesnippet (SimpleCV)

```
1 blue_distance_binarized = blue_distance.binarize(125)
```

**Schritt 3: Blob erkennen** Auf diesem Binärbild wird ein Algorithmus zur Blob-Erkennung gestartet. Ein Blob ist ein zusammenhängender Bereich in einem Bild, welcher dieselben Farbwerte enthält.

Da das zuvor erzeugte Binärbild nur schwarze und weiße Flächen enthält, liefert die Blob-Detection genaue Werte. Die Fläche des Blobs (und somit Containers) wurde in Abbildung 29 grün eingefärbt.

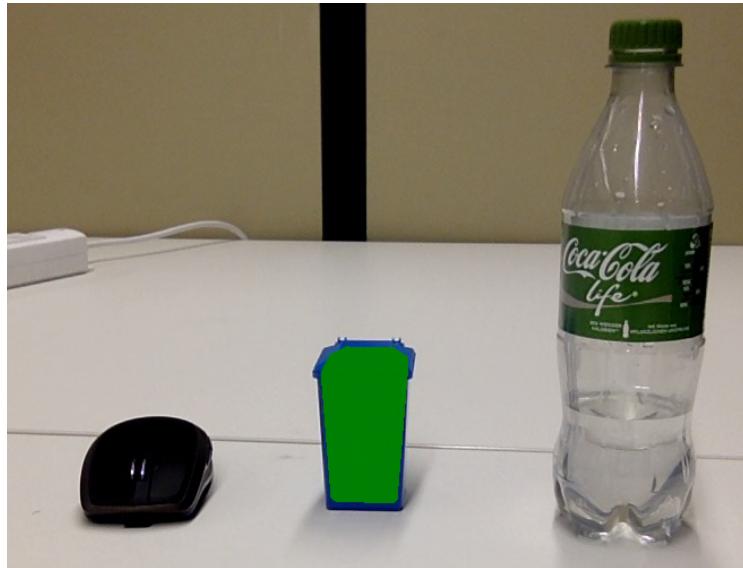


Abb. 29: Bloberkennung im Binärbild

**Codesnippet (SimpleCV)**

```
1 blobs = blue_distance_binarized.findBlobs(minsize=500)
```

Blobs können in SimpleCV auf Minimal- und Maximalgrösse selektiert werden. Zusätzlich kann via SimpleCV die Form des Blobs analysiert werden. Mit folgendem Codesnippet wird nach allen gefundenen Blobs gefiltert, welche eine rechteckähnliche Form haben (mit einem *Threshold* von 0.2)

**Codesnippet (SimpleCV)**

```
1 rects = blobs.filter([b.isRectangle(0.2) for b in blobs])
```

**Distanzermittlung** Die Distanz des Containers kann mittels zwei Referenzwerten [2] errechnet werden. Die beiden Referenzmessungen wurden beim Testen mit einer eingebauten iSight-Kamera (Macbook) ermittelt.

$$R_W \cdot R_D = M_W \cdot M_D \cong \frac{R_W}{M_W} = \frac{M_D}{R_D}$$

$R_W$  = Referenzbreite des Blobs (Containers) mit Referenzmessung 150 Pixel

$R_D$  = Referenzdistanz zum Container mit Referenzmessung 297mm

$M_W$  = Breite des Blobs während der Bildverarbeitung

$M_D$  = Gesuchte Distanz des Containers

Die gesuchte Distanz ergibt sich somit aus:

$$M_D = \frac{R_W \cdot R_D}{M_W}$$



Abb. 30: Containererkennung nach Farbe und Distanzabschätzung

Schwachpunkt dieser Berechnung ist ein liegender Container. Da bei diesem die Breite viel grösser ist, wird die errechnete Distanz kleiner. Da dies in der Aufgabenstellung ausgeschlossen ist, ist dies nicht relevant.

#### 4.9.4 Komponentendiagramm Raspberry Pi 2

Das Komponentendiagramm in Abbildung 31 zeigt die Abhängigkeiten auf dem Raspberry Pi 2, welches den Ablauf der Fahrt steuert sowie die Bildbearbeitung der Kamera durchführt.

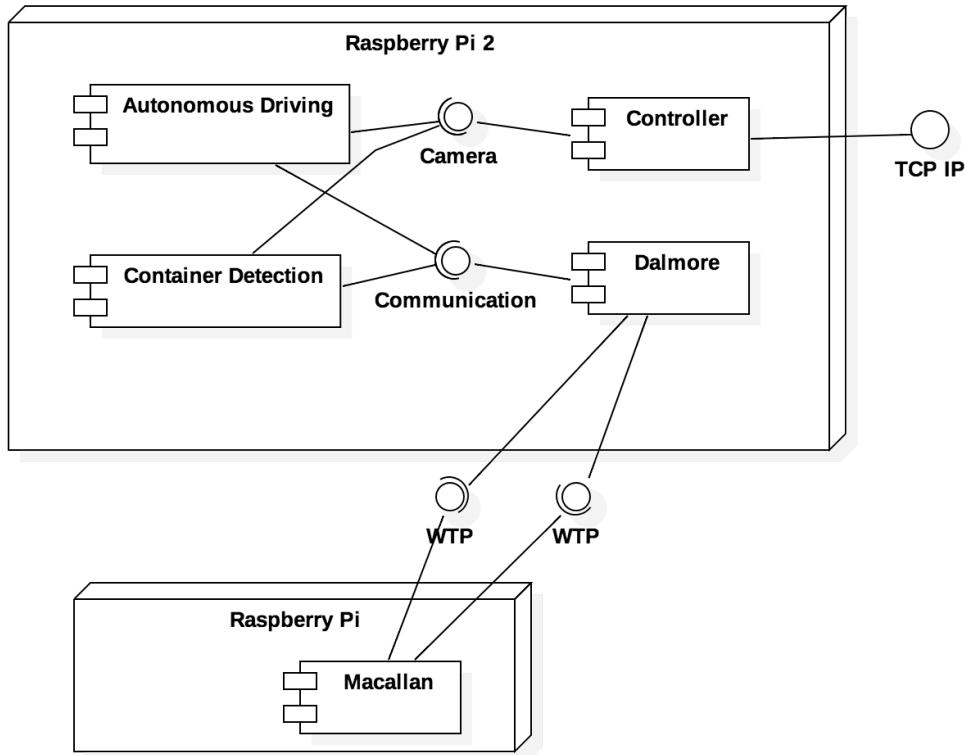


Abb. 31: Komponentendiagramm Raspberry Pi 2

**Controller** Der Controller dient als Einstiegskomponente auf dem Raspberry Pi 2. Zudem ist die Komponente für die Ablaufsteuerung zuständig. Diese Komponente definiert somit, in welchem Zustand aus Kapitel 4.9.5 sich das System aktuell befindet.

Zum Starten des Fahrzeuges wird über die TCP/IP Schnittstelle der Startbefehl mit der gewünschten Containerfarbe übermittelt. Danach startet der Controller die Kamera und beginnt damit, Bilder an die Komponenten Autonomous Driving und Container Detection zu schicken. Zur Verwaltung des Ablaufes meldet die Komponente Autonomous Driving, ob sie eine Kreuzung erkannt hat. Die Komponente Container Detection meldet, ob ein Container der gewünschten Farbe erkannt wurde.

**Autonomous Driving** Die Komponente Autonomous Driving ist für die Bildbearbeitung der Spurhaltung zuständig. Aus der Spurhaltung werden Lenkwinkel und Geschwindigkeit an die Kommunikationskomponente Dalmore weitergegeben. Zudem wird erkannt, ob der Roboter sich an einer Kreuzung befindet, damit der Rechtsvortritt geprüft werden kann. Am Ende der Fahrt wird ermittelt, ob das Parkfeld erreicht wurde und der Entleerungsvorgang wird

gestartet.

**Container Detection** Die Komponente Container Detection prüft das Bild auf vorhandene Container der Referenzfarbe. Falls ein Container innerhalb einer definierten Distanz gefunden wird, meldet die Komponente der Kommunikationskomponente, dass ein Container gefunden wurde inkl. Distanz zum Container. Danach wird der Container über die Steuerung des Raspberry Pi aufgeladen.

**Dalmore** Komponente für die Kommunikation mit dem Raspberry Pi. Für die Kommunikation wird das Whiskey Transport Protokoll verwendet.

#### 4.9.5 Ablaufsteuerung

Im nachfolgenden Kapitel werden der Ablauf des Systemes aufgezeigt und beschrieben.

**Zustände** Das System kennt die folgenden Zustände, welche in Abbildung 32 in einem Zustandsdiagramm dargestellt werden.

---

Zustand	Beschreibung
<b>INITIALISATION</b>	Alle Systeme werden hochgefahren, das Fahrzeug wird fahrbereit gemacht.
<b>DRIVE_AND_LOAD</b>	Das Fahrzeug fährt und lädt Container auf.
<b>WAIT_CROSSROAD</b>	Das autonome Fahrsystem hat einen gültigen Rechtsvortritt erkannt und wartet an der Kreuzung, bis es weiterfahren kann.
<b>CONTAINER_DETECTION</b>	Ein Container wurde von der Kamera entdeckt. Der RGB-Sensor wird eingeschalten, welcher den Container genauer erkennt.
<b>CONTAINER_LOADING</b>	Der Container wurde vom RGB-Sensor erkannt und wird gerade vom Greifarm geladen.
<b>CONTAINERS_LOADED</b>	Alle Container wurde geladen. Das Fahrzeug fährt auf schnellstem Wege zum Parkfeld.
<b>DISPOSING</b>	Der Müll wird vom Fahrzeug abgeladen.

---

Tab. 13: Grobbeschreibung der Systemzustände

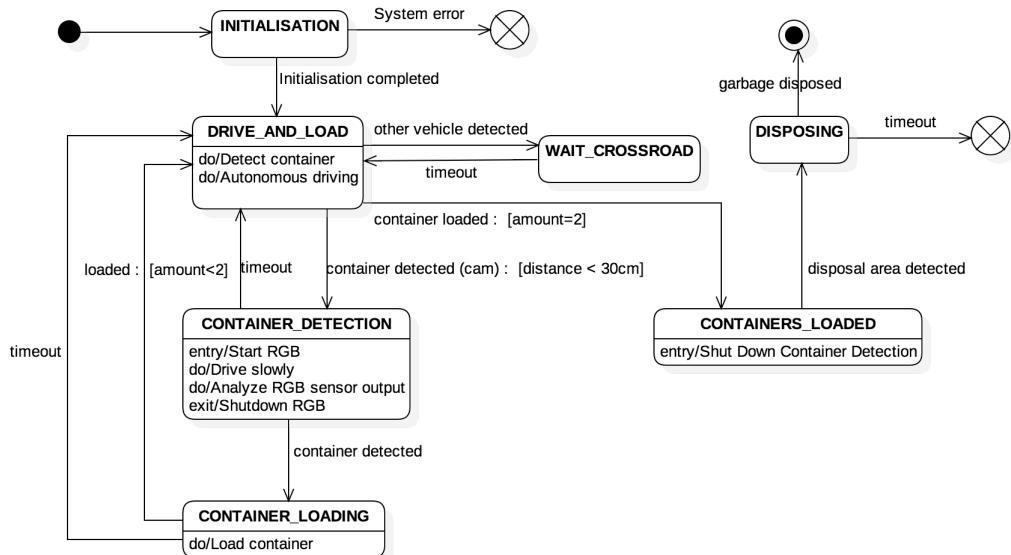


Abb. 32: Zustanddiagramm des Fahrzeuges

**Zustand INITIALISATION** Bei diesem Zustand handelt sich um den Initialzustand. Das Fahrzeug befindet sich nach dem Einschalten in diesem Zustand. Die beteiligten Module und Komponenten werden gestartet und miteinander synchronisiert.

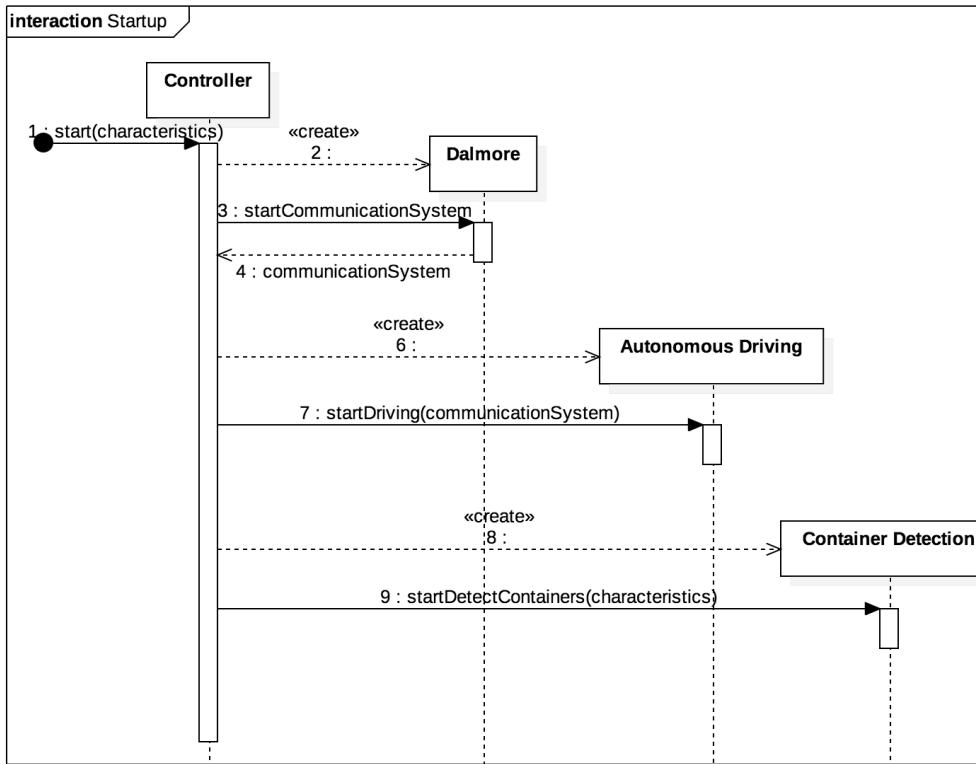


Abb. 33: Sequenzdiagramm des Startvorganges

Nach diesem Prozess ist das Fahrzeug betriebsbereit und wechselt in den Zustand DRIVE\_AND\_LOAD.

**Zustand DRIVE\_AND\_LOAD** In diesem Zustand sind softwareseitig folgende Komponenten aktiv:

- Erkennung der Fahrspur via Kamera
- Erkennung von Container via Kamera
- Erkennung Rechtsvortritt

Die Fahrspurerkennung analysiert laufend die Bilder der Kamera und versucht dabei die Fahrspur zu halten. Zeigt die Analyse, dass das Fahrzeug sich der Kreuzung nähert, werden die Ultraschallsensoren eingeschaltet, um den Rechtsvortritt zu erkennen.

Sobald ein Container auf circa 30cm von der Kamera erkannt wurde wechselt das System in den Zustand CONTAINER\_DETECTION. Wurde ein gültiger Rechtsvortritt erkannt, wechselt das System in den Status WAIT\_CROSSROAD.

Sobald das Fahrzeug bereits zwei Container aufgeladen hat, wechselt das System automatisch in den Zustand CONTAINERS\_LOADED.

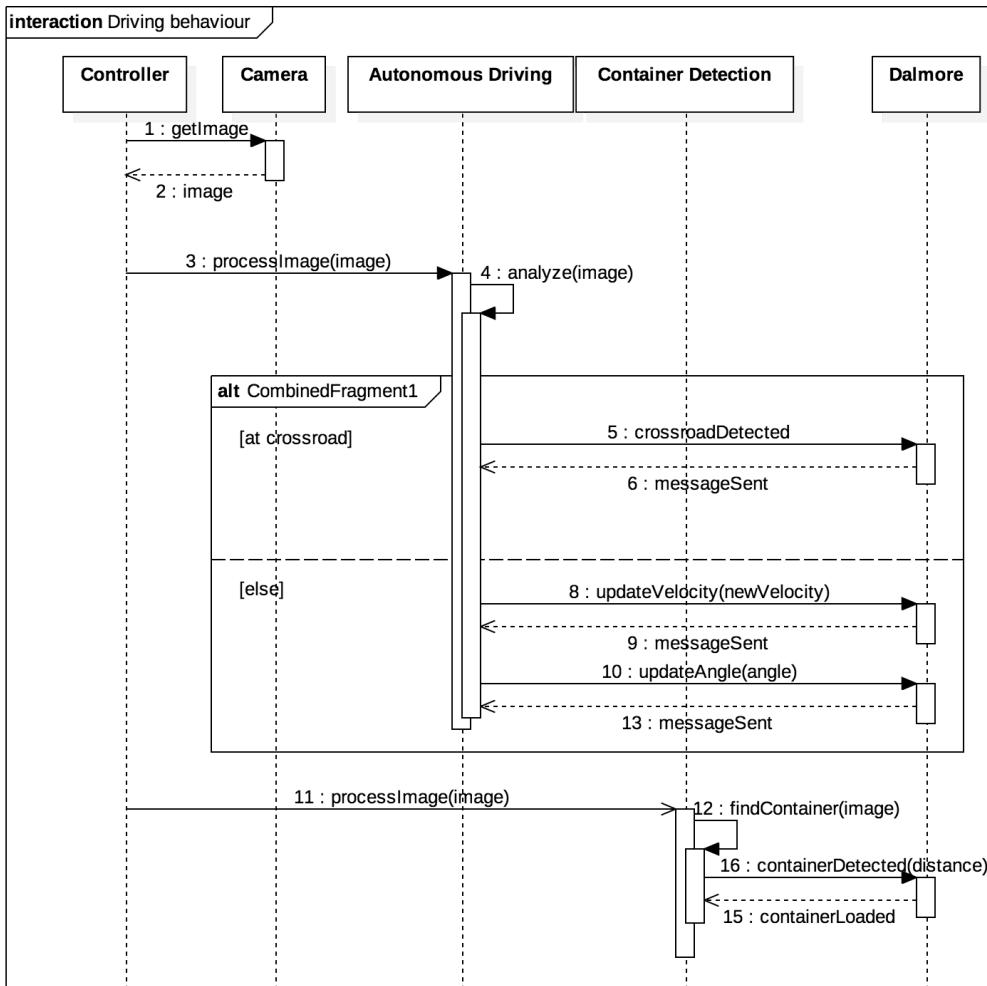


Abb. 34: Sequenzdiagramm des Zustandes DRIVE\_AND\_LOAD

**Zustand WAIT\_CROSSROAD** Ein Rechtsvortritt wurde erkannt, dem anderen Fahrzeug wird der Vortritt gewährt. Das eigene Fahrzeug hält an und wartet, bis kein Fahrzeug mehr im Rechtsvortritt erkannt wird.

Wird der Rechtsvortritt des anderen Fahrzeuges nicht umgesetzt, fährt das eigene Fahrzeug nach einem gewissen Timeout weiter.

Das System wechselt dann wieder in den Zustand DRIVE\_AND\_LOAD.

**Zustand CONTAINER\_DETECTION** Ein Container wurde von der Kamera grob auf die Distanz 30cm ermittelt. Das System wechselt daraufhin in diesen Zustand, um eine genauere Container-Erkennung mittels einem RGB-Sensor, welcher an der Seite des Fahrzeuges auf Höhe des Containers

angebracht ist, durchzuführen.

Der RGB-Sensor wird deshalb eingeschaltet und analysiert die erhaltenen Farbwerte. Um eine genauere Messung zu erreichen, wird zudem die Geschwindigkeit des Fahrzeugs gedrosselt.

Sobald der Sensor für eine gewisse Zeitdauer einen gesuchten Farbwert (blau oder grün) registriert hat, kann davon ausgegangen werden, dass ein Container aufgeladen werden muss.

Das System wechselt bei einer erfolgreichen Erkennung in den Zustand CONTAINER\_LOADING.

Wurde nichts erkannt, wechselt das System nach einem Timeout automatisch wieder in den Zustand DRIVE\_AND\_LOAD.

**Zustand CONTAINER\_LOADING** In diesem Zustand befindet sich das System, wenn der Greifarm direkt vor dem Container steht. Dieser wird ausfahren und der Container wird aufgeladen, geleert und wieder abgestellt.

Sobald der Greifarm den Container wieder abgestellt hat, wechselt der Zustand wieder zu DRIVE\_AND\_LOAD.

Zusätzlich wird bei einem Timeout auch wieder in den Zustand DRIVE\_AND\_LOAD gewechselt.

**Zustand CONTAINERS\_LOADED** Vom Zustand DRIVE\_AND\_LOAD wechselt das System automatisch in diesen Zustand, wenn es zwei Container aufgeladen hat. Die Containererkennung per Kamera wird deaktiviert, nur noch die Spurhaltung läuft. Das System versucht in diesem Zustand möglichst schnell, das Fahrzeug ins Zielfeld zu bringen.

Sobald das Fahrzeug neben der Entsorgungsstelle steht, wechselt das System in den letzten Zustand DISPOSING.

**Zustand DISPOSING** Das Fahrzeug steht direkt neben der Entsorgungsstelle und leitet die Abfallentsorgung ein. Wenn der Müll entsorgt wurde (oder ein Timeout aufgetreten ist) wechselt das System in den Endzustand.

Der Prozess ist beendet.

<b>ID</b>				<b>Name</b>	<b>Beschreibung</b>
1	1	1	1	Geschwindigkeit	Setzt die Geschwindigkeit für den Antrieb.
1	1	1	2	Lenkwinkel	Setzt den Winkel der Räder.
2	1	1	1	Distanz	Liefert die gefahrene Distanz seit der letzten Nachricht.
1	2	1	1	Container gefunden	Es wurde ein Container auf dem Kamerabild erkannt.
2	2	1	1	Container erkannt	Das Ende des Containers wurde vom Farbsensor erkannt.
1	2	1	2	Abbruch	Das erkennen des Containers soll abgebrochen werden.
2	2	1	2	Container aufgeladen	Der Inhalt des Containers wurde aufgeladen und der Container wurde wieder abgesetzt.
1	3	1	1	Inhalt abladen	Der Inhalt der Mulde auf dem Fahrzeug soll in die Entsorgungsmulde gekippt werden.
2	3	1	1	Inhalt abgeladen	Der Inhalt der Mulde auf dem Fahrzeug wurde erfolgreich abgeladen.
1	4	1	1	Kreuzung erkannt	Die Kamera hat eine Kreuzung erkannt.
2	4	1	1	Fahrzeug erkannt	Der Ultraschallsensor hat ein Fahrzeug an der Kreuzung erkannt.
2	4	1	2	Fahrzeug weg	Der Ultraschallsensor erkennt kein Fahrzeug mehr an der Kreuzung.
1	4	1	2	Abbruch	Das Erkennen eines Fahrzeuges an der Kreuzung soll abgebrochen werden.
1	5	1	1	Start	Das Fahrzeug soll initialisiert werden.
1	5	1	2	Stop	Das Fahrzeug und alle Boardcomputer sollen heruntergefahren werden.
2	5	1	1	Akku Warnung	Der Akku hat eine niedrige Ladung. Alle Boardcomputer müssen heruntergefahren werden.

Tab. 12: Auflistung aller Messages.

## 5 Projektmanagement und Planung

In den nachfolgenden Kapitel ist die Team Organisation und die ganze Planung ersichtlich.

### 5.1 Organigramm

In Abbildung 35 ist das Organigramm von Team 34 abgebildet. Wie üblich in einem Projekt, wurde eine Projektleiter festgelegt. Im Projektleiter wurde je ein Leiter der drei Fachbereiche unterstellt. Die Verantwortung für Dokumentation wurde dem Amt Redaktor/in unterstellt. Schlussendlich wurden die Stellvertretenden für den Projektleiter und Leiter Maschinenbau definiert, damit jedes Teammitglied auf dem Organigramm zu finden ist.

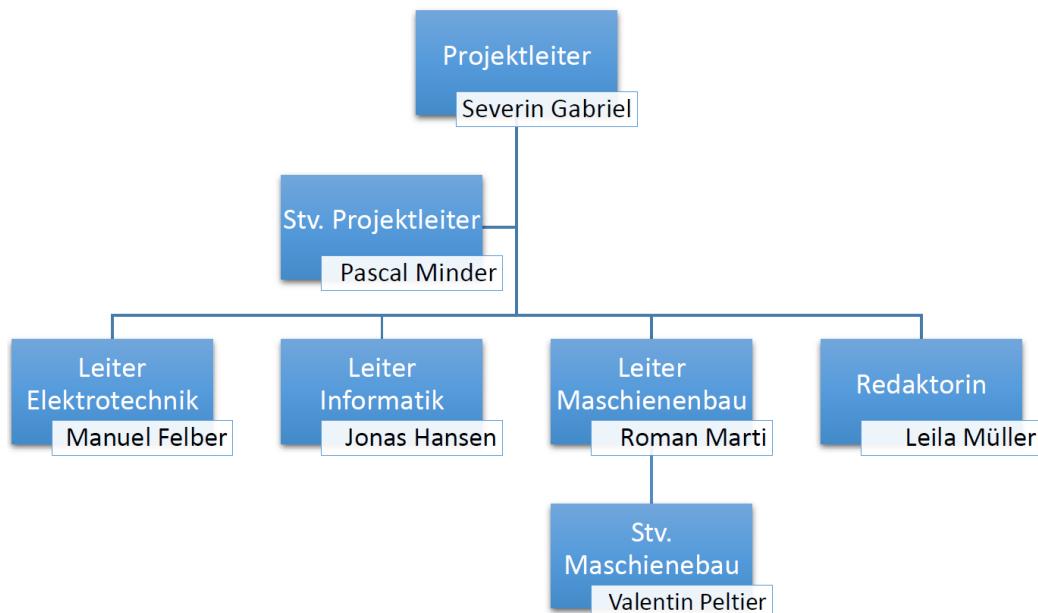


Abb. 35: Organnigramm der Gruppe 34

### 5.2 Funktionsbeschrieb

In diesem Kapitel werden zu den verschiedenen Funktionen die zugehörigen Aufgaben und Pflichten aufgelistet.

- **Projektleiter** Der Projektleiter ist verantwortlich für das Erstellen und aktualisieren des Projektplanes. Er koordiniert auch die verschiedenen Aufgaben.
- **Stv. Projektleiter** Der stellvertretende Projektleiter kommt zum Einsatz, wenn der Projektleiter verhindert ist.

- **Leiter Elektrotechnik** Der Leiter im Bereich der Elektrotechnik hat immer den Überblick über die noch zu erledigenden Aufgaben in seinem Bereich. Er ist Verantwortlich für alle elektrische Probleme und Aufgaben.
- **Leiter Informatik** Der Leiter im Bereich der Informatik hat immer den Überblick über die noch zu erledigenden Aufgaben in seinem Bereich. Der Bereichsleiter hat die Verantwortung in der Kozeptionierung und Implementierung der Software.
- **Leiter Maschienenbau** Der Leiter im Bereich der Maschinenbau hat immer den Überblick über die noch zu erledigenden Aufgaben in seinem Bereich. Er ist verantwortlich für die CAD Zeichnungen und den Fahrzeugaufbau.
- **Stv. Maschinenbau** Der stellvertretende Leiter Maschinenbau hilft dem Bereichsleiter bei seinen Aufgaben.
- **Redaktorin** Die Aufgabe der Redaktorin besteht darin, die einzelnen Texte zusammenzutragen und in ein einheitliches Dokument zu übernehmen. Sie ist auch für die pünktliche Abgabe der Meilensteine verantwortlich.

## 5.3 Planung

### 5.3.1 Meilensteine

In Tabelle 14 wird eine Übersicht über die Meilensteine des Projektes gegeben. Der Projektplan (Kapitel 5.3.3) enthält zusätzliche Informationen und Daten des Projektes.

#	Bezeichnung	Datum
1	Technologierecherche & Produktanforderungen	09.10.2015
2	Evaluation der Lösungsprinzipien & Auswahl der Kombinationen	06.11.2015
3	Freigabe Gesamtkonzept & Projektdokumentation (80%)	11.12.2015
4	Abgabe Dokumentation	08.01.2016

Tab. 14: Übersicht Meilensteine

### 5.3.2 Verwendete Tools

Als Tool zur Planung wurde Trello<sup>3</sup> verwendet. Mit Trello können Aufgaben einfach aufgeteilt und unter den Team-Mitgliedern verteilt werden. Trello organisiert die Arbeiten in verschiedenen Boards. Diese Boards wiederum besitzen Listen, in denen die Aufgaben als Kärtchen abgelegt werden können. In Abbildung 36 ist ein Trello-Board zu sehen, welches wie ein Scrum-Board aufgebaut ist. Alle Aufgaben wanderten jeweils von der Liste *Backlog* zu *In Progress* zu *To Verify* und schliesslich in die *Done*-Liste.

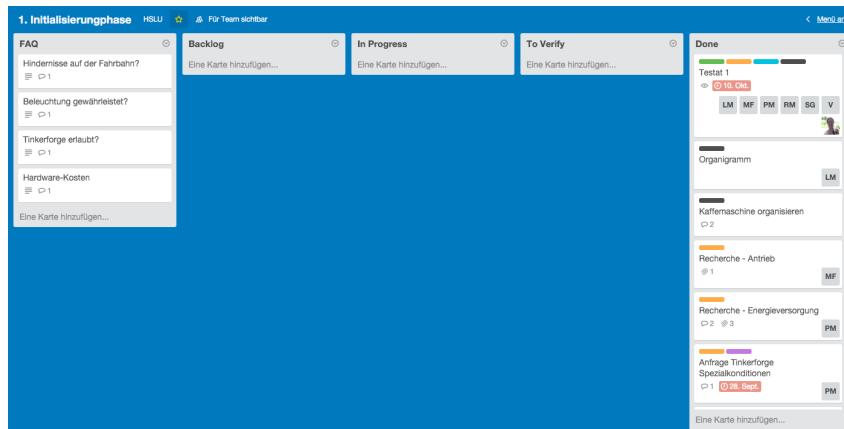


Abb. 36: Trello Board-Oberfläche

Ein iterativ-inkrementelles Vorgehensmodell ist somit problemlos zu realisieren.

<sup>3</sup><https://trello.com>

### 5.3.3 Projektplan

Der Projektplan wurde in einem Excel-Sheet festgehalten und gepflegt. Der komplette Projektplan ist in Abbildung 37 ersichtlich.

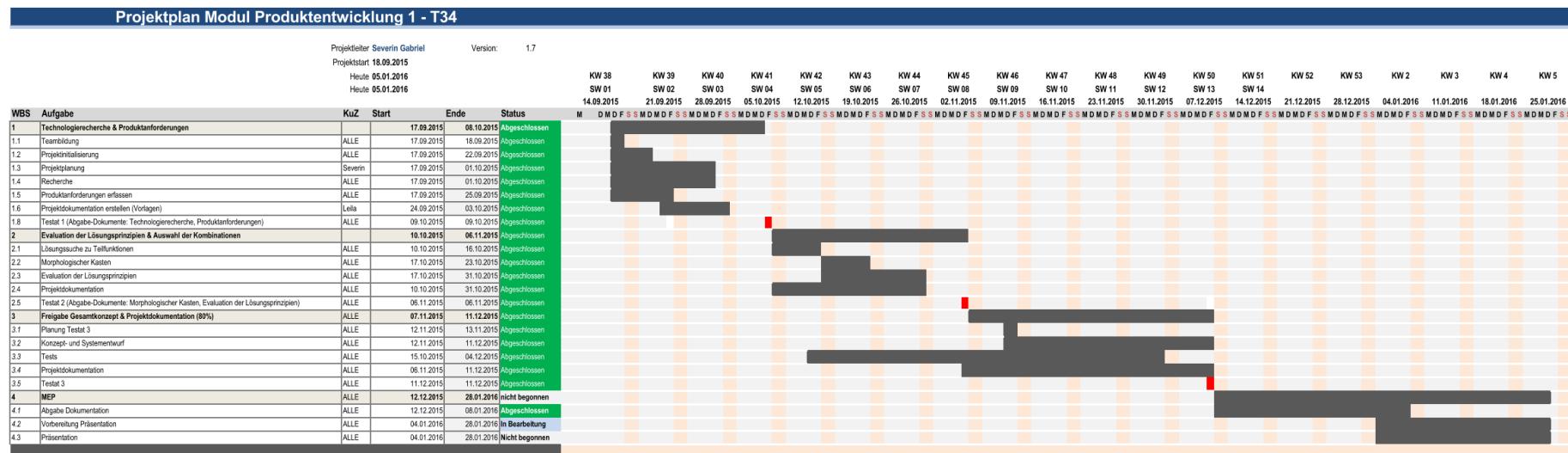


Abb. 37: Projektplan

## 6 Schlussdiskussion

Im Kapitel Schlussdiskussion wird das ganze Projekt nochmals Revue passiert. Die Ausgaben sind aufgelistet und wichtige Erkenntnisse wurden festgehalten. Auch wird ein Ausblick auf das kommende PREN2 gemacht und die Risiken dokumentiert.

### 6.1 Kosten

Für das gesamte Projekt stehen 500 CHF zur Verfügung. Davon dürfen maximal 200 CHF ausgegeben werden. In diesem Projekt wurde von Anfang an das Budget für das gesamte Projekt aufgestellt. Das heisst es wurden auch schon Kosten für das PREN2 abgeschätzt und eingerechnet. In der untenstehenden Tabelle sind alle Kostenpunkte aufgelistet.

Beschreibung	Preis pro Stück	Anzahl	Gesamt
Raspberry Pi 2 (Model B)	CHF 46.60	1	CHF 46.60
Raspberry Pi 1 (Model B+)	CHF 30.00	1	CHF 30.00
Master Bricklet	CHF 35.61	1	CHF 35.61
Servo Brick	CHF 60.00	1	CHF 60.00
Color Bricklet	CHF 13.67	1	CHF 13.67
IO-16 Bricklet	CHF 13.41	1	CHF 13.41
US-Distance Bricklet	CHF 13.67	1	CHF 13.67
DC-Motor	CHF 20.00	1	CHF 20.00
Servo	CHF 15.00	1	CHF 15.00
Motorentreiber	CHF 13.00	1	CHF 13.00
Kamera	CHF 30.00	1	CHF 30.00
Akku	CHF 50.00	1	CHF 50.00
Mechanik (Lager, Zahnräder, ..)	CHF 50.00	1	CHF 50.00
Rad	CHF 15.00	1	CHF 15.00
			<b>CHF 450.96</b>

Tab. 15: Budgettabelle

Die zwei **Raspberry Pis** sind schon vorhanden und mussten nicht eingekauft werden. Trotzdem wurden sie zu einem tieferen Preis eingerechnet. Die **Tinkerforge Module Master Brick**, **Color Bricklet**, **IO-16 Bricklet**, **Distance US Bricklet** und **Servo Brick** wurden beim Hersteller bestellt. Diese wurden zum Originalpreis eingerechnet. Der DC-Motor wurde aus einem Ausschuss wiederverwertet und zu einem Preis von 20 CHF eingerechnet. Der Motorentreiber wurde über einen Onlineshop bestellt. Die ganzen Servos und Akku wurden ebenfalls aus einem Modellbaushop bestellt. Die Kamera und die Räder sind eingekauft bei Conrad. Die Mechanikteile werden in einem späteren Zeitpunkt bei Mädler bestellt.

## 6.2 Lessons Learned

Die Lessons learned sind in die unterschiedlichen Fachbereiche unterteilt. Jeder Bereich erfährt seine eigenen Erfahrungen, welche das nächste Mal besser gemacht werden können.

### 6.2.1 Elektronik

Es wurde realisiert, dass die Farbsensoren und Ultraschallsensoren mit den äusseren Störeinflüssen schlechter funktionieren als erwartet. Die Datenblatt Werte beinhalten manchmal schon Störungen, doch in Wirklichkeit sind die Werte oft noch schlechter. Diese Einflüsse müssen zwingend beachtet werden. Ebenfalls nicht zu verachten sind die Erfahrungen bei der Zusammenarbeit mit der Mechanik. Es müssen meist Kompromisse in Baugrösse und Bauteil Anzahl gemacht werden. Für die Mechanik ist ein Bauteil einfacher, kann jedoch elektronisch nicht immer so gelöst werden.

### 6.2.2 Maschinentechnik

Es wurde für die konstruktive Auslegung der autonomen Müllabfuhr mögliche Teillösungen gefunden. Dabei wurde aber zu fest auf die Teilstrukturen an sich und zu wenig auf die Interaktionen zwischen ihnen geachtet. Die Teilstrukturen waren für sich genommen gut gewählt. Bei der genaueren Dimensionierung der Mulde mit Klappe und des Greifarms stellte sich jedoch heraus, dass die benötigte Höhe für die Mulde kaum durch den gewählten Greifarm sichergestellt werden konnte. Daher wurde am Ende noch eine weitere Lösungsvariante in Betracht gezogen. Für die weiteren Schritte ist es wichtig, den ganzen Roboter im Blickfeld zu haben.

### 6.2.3 Informatik

Im Informatikbereich wurden zwei Erkenntnisse gewonnen:

- 1. Für Recherche früher auf Programmiersprache / Entwicklungs-umgebung einigen**

Es wurde zu lange mit unterschiedlichen Programmiersprachen (unter anderem Java, C++, C#, Matlab-Code) und Frameworks experimentiert, bis die Wahl auf Python gefallen ist. Das Evaluieren von unterschiedlichen Plattformen ist wichtig, jedoch soll schon früh ein Framework und eine Programmiersprache ausgewählt werden.

- 2. Früher Softwarearchitektur planen**

Über Softwarearchitektur wurde zu Beginn des Projektes gesprochen, es wurde jedoch zu wenig dokumentiert und detailliert spezifiziert. Ein Gesamtüberblick über die Komponenten, ihre Interaktion und den groben Ablauf hätte während den Recherchen nicht geschadet.

## 6.3 Risiken

Zu Beginn der Produktentwicklung wurden diverse Risiken identifiziert, welche sich im Anhang befinden. Einige Risiken konnten bereits stark reduziert werden. Die folgende Risikoliste ist auf PREN 2 bezogen.

---

#	Risiko	Auswirkungsgrad (1-10)	Wahrscheinlichkeit (1-10)
1	Budget ist zu klein	9	2
2	Rechenleistung zu klein	9	3
3	Starker Lichteinfall verhindert Spurerkennung	6	8
4	Container wird über Farbe nicht erkannt	4	3
5	Ultraschallsensor deckt nicht genügend grossen Bereich ab	5	4
6	Lenkung zu ungenau für Zielgeschwindigkeit	8	3
7	Fahrzeug zu lang für Fahren auf eigener Spur	7	3

Tab. 16: Risiken

## 6.4 Massnahmen

In der unterstehenden Tabelle werden die Massnahmen für obenstehenden Risiken aufgelistet.

---

#	Massnahme
1	Budgetplanung aktuell halten damit Probleme frühzeitig erkannt werden
2	Algorithmen auf effektiver Umgebung testen und allfällige Alternativen finden
3	Hardwareseitig Webcaminstellungen prüfen. Softwareseitig lichtunabhängige Ergänzungen finden. Z.B Eckenerkennung
4	Alternative Objekterkennung finden
5	Zusätzlich über Kamerabild Rechtsvortritt prüfen
6	Geschwindigkeit verringern
7	Früh mittels Fahrzeulgänge Kurvenfahrt testen

---

Tab. 17: Massnahmen

## 6.5 Ausblick PREN 2

Für den zweiten Teil von PREN, sind noch verschiedene Aufgaben zu erledigen. Dazu gehört die Weiterentwicklung der Software auf dem Raspberry Pi 2, insbesondere die Spurerkennung. Im Bereich der Sensor- und Aktorensteuerung dient die aktuelle Software zurzeit nur als Test für das Ansteuern der einzelnen Sensoren und Aktoren. Sie muss erweitert werden, damit die einzelnen Aufgaben nebenläufig erledigt werden können. Zudem muss eine PID-Regelung für den Motor geschrieben werden. Im Bereich Elektronik soll ein Testaufbau möglichst früh erstellt werden, um allfällige Fehler schnell zu erkennen. Die Maschinenbauer können alle erstellen 3D-Teile als Prototypen im 3D-Drucker des FabLabs drucken. Somit können auch Überlegungsfehler in der Konstruktion erkannt werden.

## 6.6 Fazit

Es zeigten sich grosse Ähnlichkeiten bei den Lösungswegen. Unterschiede lagen hauptsächlich bei der Wahl der Sensoren. Daher ist die Bewertung der meisten Gesamtkonzepte recht ähnlich. Variante 1 entsprach den Kriterien am besten. Die anderen hochbewerteten Varianten wählten hauptsächlich andere Sensoren. Die Wahl der Sensoren ist in einem nächsten Schritt durch Tests und

Analysen zu entscheiden.

Das Konzept nach der Variante 1 sieht ein Fahrgestell mit vier Rädern und einer Achsschenkellenkung vor. Die Müllheimer werden mit einem Greifer eingesammelt. Ein Hebelarm hebt den Müllheimer an und leert ihn in die Mulde aus. Die Mulde hat einen schrägen Boden und ist mit einer Klappe geschlossen. Diese wird durch einen Servo geöffnet und geschlossen. Als Energiequelle dient ein LiPo-Akku. Das gesamte System wird über zwei Raspberry Pis gesteuert und geregelt.

In den folgenden Schritten wird nun das ausgewählte und angepasste Konzept ausgearbeitet und getestet.

## Abbildungsverzeichnis

1	Ablaufdiagramm . . . . .	8
2	Greifarm . . . . .	10
3	Greifarm in der CAD-Ansicht . . . . .	11
4	Ansichten Fahrgestell . . . . .	12
5	Fahrgestell in der CAD-Ansicht . . . . .	12
6	Schenkellenkung des Fahrgestelles [1] . . . . .	13
7	Mulde gehoben . . . . .	14
8	Mulde gesenkt . . . . .	15
9	Seitenansicht . . . . .	16
10	Tinkerforge-Hardware Konzept [5] . . . . .	17
11	Tinkerforge Color-Bricklet [3] . . . . .	18
12	Tinkerforge Dinstance-US-Bricklet [4] . . . . .	19
13	Schaltung der Akkuüberwachung . . . . .	21
14	Blockschaltbild Spannungen . . . . .	22
15	Motorentreiber . . . . .	24
16	Aufbau Encoder Rückführung . . . . .	25
17	Regelkreis . . . . .	25
18	Grobübersicht des Produktivsystems . . . . .	27
19	konzeptioneller Aufbau der API . . . . .	28
20	Entwicklungssystems . . . . .	29
21	Produktivsystems . . . . .	30
22	Paket Struktur . . . . .	31
23	Message Type ID Aufbau . . . . .	32
24	Farbfilter auf Fahrbahn . . . . .	34
25	Fahrlinie auf Fahrbahn abgebildet . . . . .	34
26	Aufnahmen Fahrbahn bei Sonnenlicht . . . . .	35
27	Farbdistanz zur Referenzfarbe . . . . .	36
28	Binärbild des Containers . . . . .	37
29	Bloberkennung im Binärbild . . . . .	38
30	Containererkennung nach Farbe und Distanzabschätzung . . . . .	39
31	Komponentendiagramm Raspberry Pi 2 . . . . .	40
32	Zustanddiagramm des Fahrzeuges . . . . .	43
33	Sequenzdiagramm des Startvorganges . . . . .	44
34	Sequenzdiagramm des Zustandes DRIVE_AND_LOAD . . . . .	45
35	Organnigramm der Gruppe 34 . . . . .	48
36	Trello Board-Oberfläche . . . . .	50

37	Projektplan . . . . .	51
38	Bildpaar vor und nach Gleichrichtung . . . . .	64
39	Disparity map . . . . .	64
40	Orginalbilder Versuchsaufbau . . . . .	65
41	Auswertung Versuch 1 - StereoBM . . . . .	65
42	Auswertung Versuch 2 - StereoSGBM . . . . .	66
43	HoughLineP Gerade . . . . .	68
44	HoughLineP Kurve . . . . .	68
45	Thinkerforge Module Motorensteuerung . . . . .	69
46	Blockschaltbild Aufbau . . . . .	69
47	Testaufbau . . . . .	70
48	Testaufbau Color-Sensor . . . . .	72
49	Messwerte des Color-Bricklet auf verschiedene Distanzen . . . . .	73
50	Testaufbau und Messwerte des Distance US Bricklets . . . . .	74
51	Morphologischer Kasten . . . . .	89
52	Mulde offen/geschlossen . . . . .	91
53	Vereinfachung des Greifsystems . . . . .	93

## Tabellenverzeichnis

1	Beispiele für die Konzeptfindung.	4
2	Bewertungstabelle der einzelnen Varianten.	5
3	Bewertungsskala	6
4	Technische Daten des Color-Bricklets [3]	18
5	Eigenschaften Ultraschall-Sensor	19
6	Leistung der Komponenten	20
7	Vergleich DC-Motoren	23
8	Steuerzustände Motorentreiber	24
9	Beschreibung der Felder im Paket.	31
10	Beschreibung der Felder im Paket.	32
11	Auflistung aller Command Groups.	32
13	Grobbeschreibung der Systemzustände	42
12	Auflistung aller Messages.	47
14	Übersicht Meilensteine	50
15	Budgettabelle	52
16	Risiken	55
17	Massnahmen	56

## A Glossar

## B Quellenverzeichnis

### Literatur

- [1] dasmodellauto.de. *Servolenkung*. 23. Dez. 2015. URL: [http://www.tinkerforge.com/de/doc/Hardware/Bricklets/Distance\\_US.html](http://www.tinkerforge.com/de/doc/Hardware/Bricklets/Distance_US.html).
- [2] Adrian Rosebrock. *Find distance from camera to object/marker using Python and OpenCV*. 19. Jan. 2015. URL: <http://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/>.
- [3] Tinkerforge. *Tinkerforge Color Bricklet*. 21. Dez. 2015. URL: <http://www.tinkerforge.com/de/doc/Hardware/Bricklets/Color.html>.
- [4] Tinkerforge. *Tinkerforge Distance US Bricklet*. 21. Dez. 2015. URL: [http://www.tinkerforge.com/de/doc/Hardware/Bricklets/Distance\\_US.html](http://www.tinkerforge.com/de/doc/Hardware/Bricklets/Distance_US.html).
- [5] Tinkerforge. *Tinkerforge Hardware*. 21. Dez. 2015. URL: [http://www.tinkerforge.com/de/home/what\\_is\\_tinkerforge/](http://www.tinkerforge.com/de/home/what_is_tinkerforge/).
- [6] Tinkerforge. *Tinkerforge IO 16 Bricklet*. 2. Jan. 2016. URL: <https://www.tinkerforge.com/de/shop/servo-brick.html>.
- [7] Tinkerforge. *Tinkerforge Servo Brick*. 2. Jan. 2016. URL: <https://www.tinkerforge.com/de/shop/bricklets/io16-bricklet.html>.

## C Anhang

### C.1 Technischer Bericht Stereokamera

In den nächsten Kapitel sind die Versuche mit der Kamera dokumentiert.

#### C.1.1 Ausgangslage

Für das Projekt soll eine Spurhaltung und Objekterkennung realisiert werden. Für die Spurhaltung wird eine Möglichkeit zur Spurerkennung benötigt. Zudem ist für die Erkennung des Rechtsvortritts eine Distanzermittlung vorteilhaft. In diesem Versuch wird evaluiert, ob eine Stereokamera beides abdecken kann.

#### C.1.2 Grundlagen

Eine Stereokamera kann zusätzlich zum Kamerabild Distanzen ermitteln. Dies funktioniert wie beim Auge dadurch, dass die Kameras eine fixe Distanz zueinander haben. Mittels Bildbearbeitungsalgorithmen wird räumliches Sehen ermöglicht.

Damit dies möglich wird, müssen die Kameras zu Beginn konfiguriert werden. Dies geschieht mit mehreren Bildpaaren die ein Muster enthalten, welches gut verarbeitet werden kann. Zum Beispiel ein Schachbrett ist geeignet, da die Ecken gut erkannt werden können. Durch diese Konfiguration kann dem Stereoalgorithmus die Position der Kameras zueinander mitgeteilt werden. Anschliessend werden beide Bilder der Kameras gleichgerichtet. Dies bedeutet, dass horizontale Linien über zwei Bilder gezogen werden, welche bei beiden Bildern dieselben Punkte durchlaufen. In Abbildung 38 wird eine Gleichrichtung zwischen zwei Bildpaaren dargestellt.

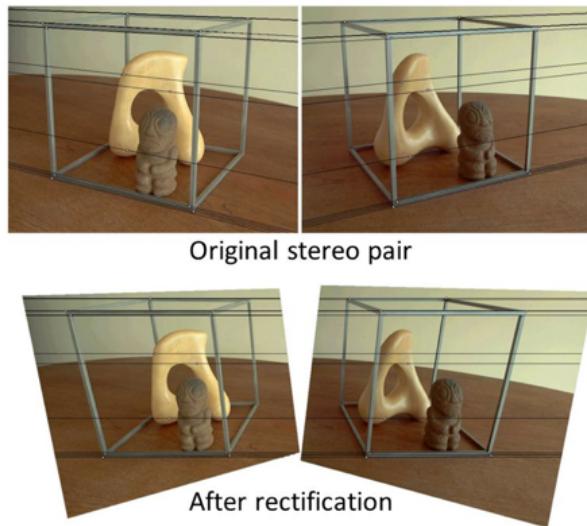


Abb. 38: Bildpaar vor und nach Gleichrichtung

Auf diese Bilder kann nun der Stereomatch-Algorithmus angewendet werden um Distanzen zu berechnen. Zur Visualisierung werden Disparity maps verwendet. Weit entfernte Flächen werden dunkel und nahe Flächen hell dargestellt. (siehe optimale Disparity Map 39)



Abb. 39: Disparity map

### C.1.3 Versuchsaufbau

Für diesen Versuch wurden 2 Baugleiche Kameras vom Typ LogiLink UA0072A verwendet. Die Software wurde auf einem MacBook Pro ausgeführt. Die Kalibrierung erfolgte mit den OpenCV-Methoden in C++. Dafür wurden 50 Bildpaare verwendet. Für die folgende Versuchsbeschreibung wird das Bildpaar aus

Abbildung 40 verwendet.

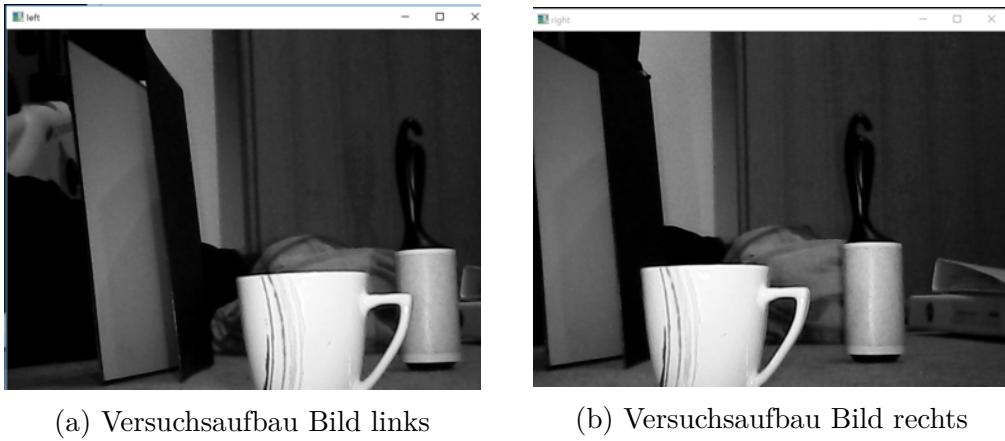


Abb. 40: Orginalbilder Versuchsaufbau

#### C.1.4 Versuch 1 - Block Matching-Algorithmus

Der erste Versuch wurde mit dem StereoBM-Algorithmus von OpenCV durchgeführt.

Aus der Abbildung 41a wird ersichtlich, dass dieser Algorithmus eine sehr gute Performance bietet. Bei einer durchschnittlichen Bearbeitungszeit von 60ms pro Bildpaar wären 16.66 FPS möglich.

```
Time elapsed: 57.769571ms
Time elapsed: 56.729349ms
Time elapsed: 57.228163ms
Time elapsed: 56.239258ms
Time elapsed: 70.379001ms
Time elapsed: 56.450690ms
Time elapsed: 59.130534ms
```

(a) Zeitaufwand StereoBM

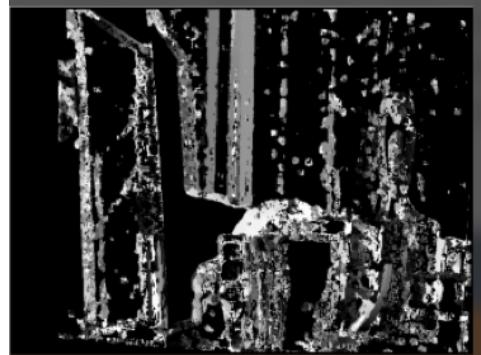


Abb. 41: Auswertung Versuch 1 - StereoBM

Die berechnete Disparitymap 41b erkennt zwar markante Punkte aber die Flächen dazwischen werden nicht ausgefüllt.

#### C.1.5 Versuch 2 - Semi Global Block Matching

Der zweite Versuch wurde mit dem StereoSGBM-Algorithmus von OpenCV durchgeführt.

Die Disparitymap (Abbildung 42b) sieht bereits wesentlich besser aus. Die Flächen werden gut ausgefüllt. Allerdings entsprechen die Grauwerte nicht den effektiven Entfernungswerten.

Der Algorithmus ist zudem sehr rechenintensiv mit durchschnittlich 1.3 Sekunden pro Bild (siehe Abbildung 42a). Damit wären nur 0.77 FPS möglich.

```
Time elapsed: 1261.745101ms
Time elapsed: 1294.097723ms
Time elapsed: 1838.394434ms
Time elapsed: 1387.436551ms
Time elapsed: 1243.015445ms
Time elapsed: 1284.585355ms
Time elapsed: 1253.881591ms
```

(a) Zeitaufwand StereoSGBM



(b) Disparity map StereoSGBM

Abb. 42: Auswertung Versuch 2 - StereoSGBM

### C.1.6 Fazit

Mit den im Versuchsaufbau getesteten Komponenten ist die Disparitymap zu ungenau für Distanzermittlungen. Zudem scheidet der SGBM-Algorithmus aufgrund der Performance aus. Der einzige Vorteil dieser Lösung gegenüber einer Lösung mit einer Kamera und Ultraschall ist die flächendeckende Distanzerkennung. Der Rechen- und Implementierungsaufwand ist allerdings zu gross für die zur Verfügung stehenden Mittel. Deshalb wird für dieses Projekt keine Stereokamera verwendet.

## C.2 Kamera Spurhaltung

In diesem Kapitel wird der erste Versuch der Kamera-Spurhaltung beschrieben. Im Lösungskonzept befindet sich der 2. Versuch, da dieser sich für das Projekt besser eignet.

### C.2.1 Ausgangslage

Für das Projekt soll eine Spurhaltung realisiert werden. In diesem Versuch wird geprüft, ob eine Spurhaltung mittels Kamera realisierbar ist.

### C.2.2 Versuchsaufbau

Für diesen Versuch wurde SimpleCV/OpenCV verwendet. Die Software wurde auf einem MacBook Pro ausgeführt. Die Line Detection wurde mit Java realisiert. Die Blobdetection wurde mit Python realisiert.

### C.2.3 Versuch - Line Detection Algorithmus

Für diesen Versuch wurde direkt auf die OpenCV-Bibliothek zugegriffen. Mit dem Hough Line Transform-Modul können Linien erkannt werden. Dabei gibt es die Methode HoughLines, welche Linien über das gesamte Bild zieht und die Methode HoughLinesP, welche Liniensegmente sucht. HoughLinesP ist für die Anforderungen besser geeignet, da die Mittellinie gestrichelt ist.

Es wurden 2 Testbilder aufgenommen, welche eine Gerade sowie eine Kurve zeigen. Bei der Geraden werden die Segmente gut erkannt (siehe Abbildung 43). Es gibt noch einige Fehler, die mittels Farbfilter und anderen Vorbearbeitungen des Bildes behoben werden können.

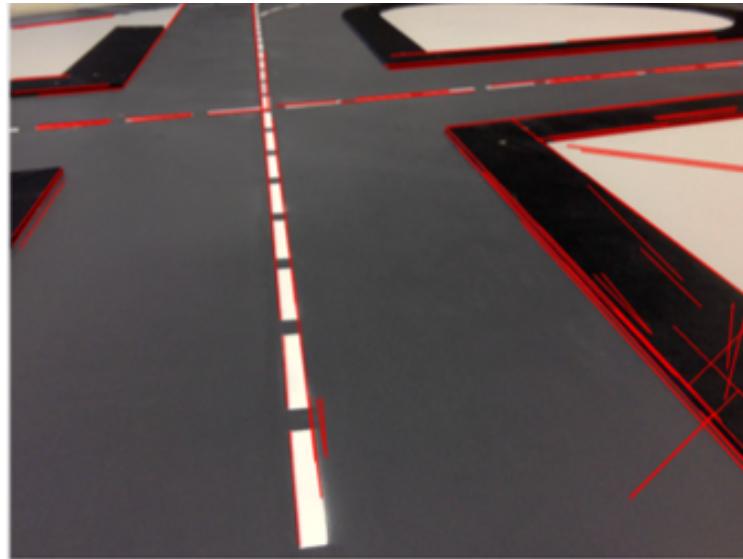


Abb. 43: HoughLineP Gerade

Bei der Kurve hat sich gezeigt, dass die Mittellinien teilweise zu rundlich gezeichnet sind für die HoughLines-Methode (siehe Abbildung 44). Dies ist ein grosses Risiko und deshalb wird im Lösungskonzept der Blob-Detection-Algorithmus verwendet.



Abb. 44: HoughLineP Kurve

### C.3 Technischer Bericht Motorentreiber und DC Motor

In diesem Kapitel wird der Testaufbau mit dem Motorentreiber, DC Motor und den Thinkerforge Bricks dokumentiert.

#### C.3.1 Ausgangslage

Um die Hardware und Software zu testen, wurde ein Testaufbau realisiert. Dazu wurde der Motor und Motorentreiber aus Kapitel Komponentenbeschreibung - Motoren 4.6 zusammen mit dem Thinkerforge Servo Brick und IO Brick aufgebaut.

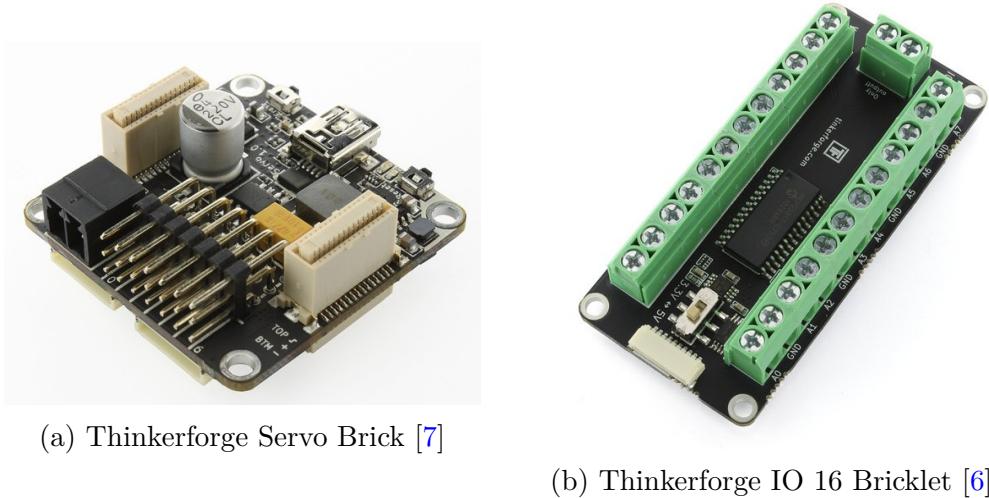


Abb. 45: Thinkerforge Module Motorensteuerung

#### C.3.2 Versuchsaufbau

Bei diesem Testaufbau wurde der ausgewählte Motor Maxon RE30 mit dem Motorentreiber M2 Microbot MR001 und dem Servo- und IO-Brick betrieben. Der genauere Testaufbau ist in der untenstehenden Grafik ersichtlich.

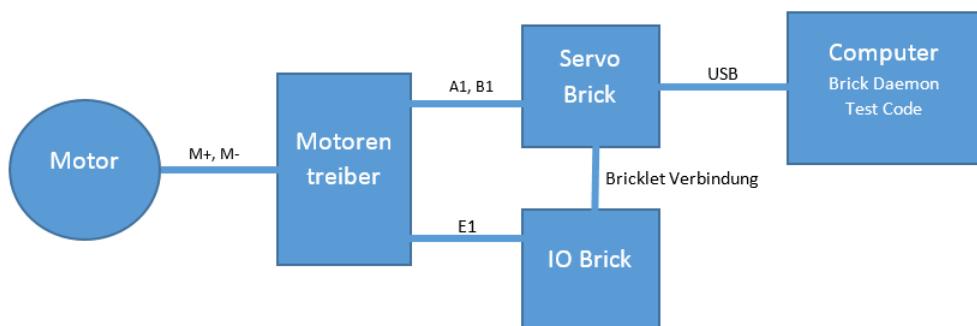


Abb. 46: Blockschaltbild Aufbau

Dazu wurden die Motorentreiber Eingänge A1 und B1 mit dem Servo Brick verbunden. Über diese Eingänge kann die Drehrichtung und Drehgeschwindigkeit gesteuert werden. Der Motorentreiber Eingang E1 wurde mit dem IO Brick verbunden. Mit diesem Eingang kann der Motorentreiber ausgeschaltet werden. Als Spannungsversorgung für den Motorentreiber diente ein Labornetzgerät. Die beiden Thinkerforge Module wurden über ein USB Kabel am Computer angehängt. Der Software Code ist im folgenden Kapitel ersichtlich [C.12](#). Über die Konsole kann nun die Drehrichtung und Geschwindigkeit des Motors verändert werden. Im nachfolgenden Bild ist der Testaufbau ersichtlich.

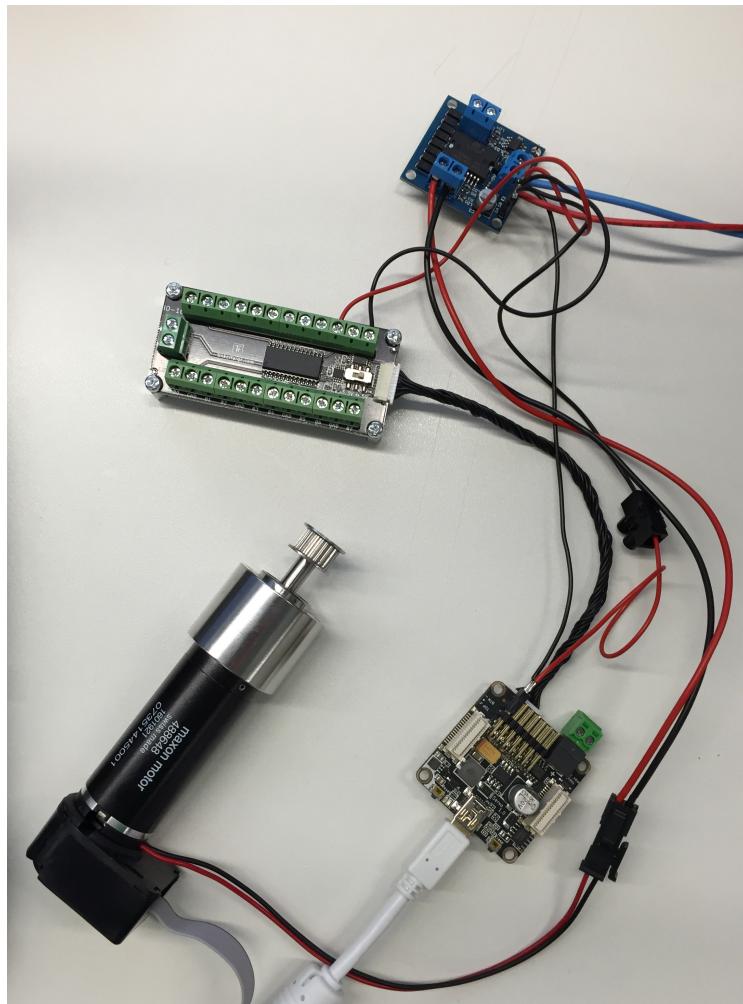


Abb. 47: Testaufbau

### C.3.3 Fazit

Der Aufbau funktionierte wie erwartet. Die Drehrichtung kann bei jeder Geschwindigkeit verändert werden. Wichtig zu testen war die geringste Drehgeschwindigkeit des Motors. Je kleiner diese Geschwindigkeit, desto genauer kann der Roboter bewegt werden. Diese geringste Geschwindigkeit wurde auf

ca. 0.3 Umdrehungen pro Sekunde geschätzt. Mit dieser Geschwindigkeit kann sehr genau gefahren werden. Dazu hilf der Motorenregler diese Geschwindigkeit beizubehalten, auf wenn das Drehmoment des Motors nicht ausreicht.

## C.4 Technischer Bericht Sensoren

Im folgenden Kapitel sind die Tests mit dem Farbsensor und Ultraschall dokumentiert.

### C.4.1 Farbsensor

Farbsensoren arbeiten normalerweise nur auf geringe Distanzen zuverlässig. Deshalb musste getestet werden, ob der Sensor ohne eine zusätzliche Linse auch auf grössere Distanzen von ca. 5-10cm funktionieren würde.

Bei den ersten Tests wurde festgestellt, dass der Sensor ohne zusätzliche Unterstützung und bei Umgebungslicht die Farbe des Containers noch auf 2cm Entfernung zuverlässig erkannte. Da der Container jedoch eher weiter entfernt vom Sensor sein wird, wurde für die nächsten Tests eine zusätzliche Platine mit zwei LEDs gebaut. So wurde die gemessene Oberfläche des Containers genügend hell ausgeleuchtet, sodass genügend Licht zurück auf den Farbsensor geworfen wurde. Dieser Versuchsaufbau wird in Abbildung 48 gezeigt.

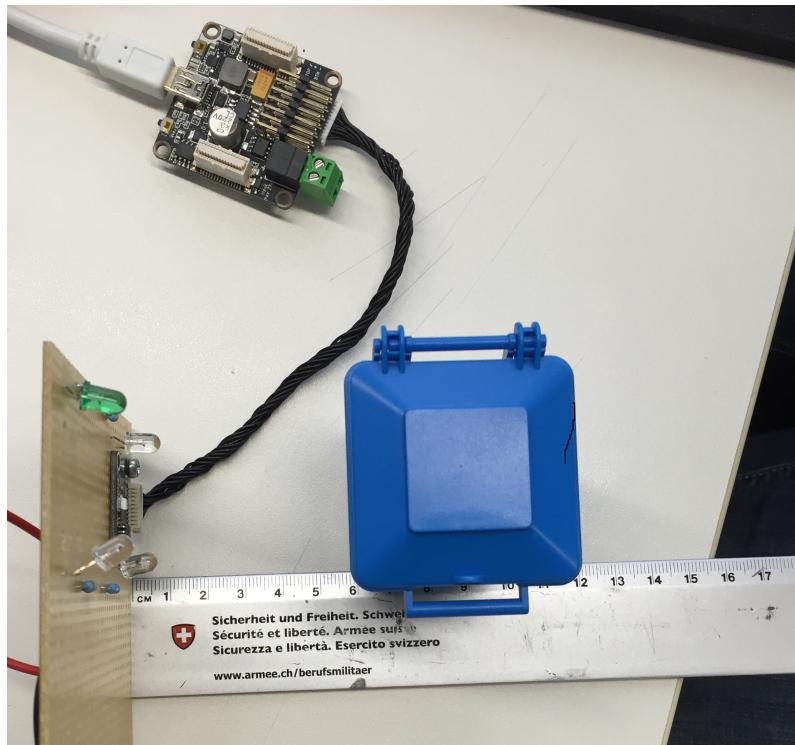
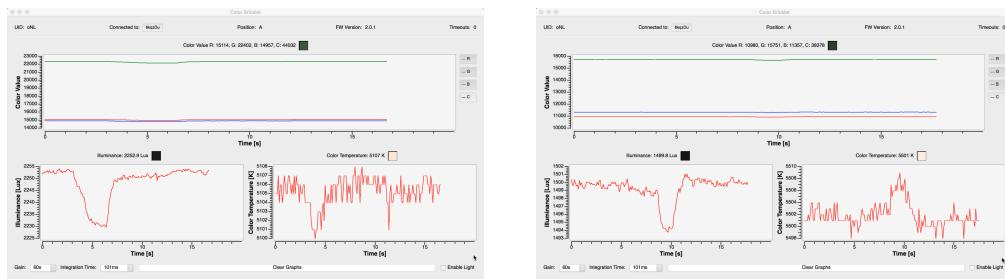
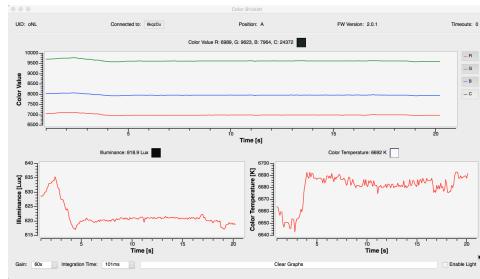


Abb. 48: Testaufbau Color-Sensor

Während den Tests, wurde die Farbe des Container auf drei verschiedene Entferungen gemessen. Dazu gehören 4, 6 und 8cm. In der Abbildung 49 sind die verschiedenen Messwerte für die ändernden Entfernungen aufgeführt.



(a) Messwerte auf eine Distanz von 4cm (b) Messwerte auf eine Distanz von 6cm



(c) Messwerte auf eine Distanz von 8cm

Abb. 49: Messwerte des Color-Bricklet auf verschiedene Distanzen

In den Abbildungen 49 ist zu sehen, dass der gemessene Container grün gewesen sein muss, da die Messline für die Grün-Komponente des RGB-Values immer am höchsten ist.

#### C.4.2 Fazit Farbsensor

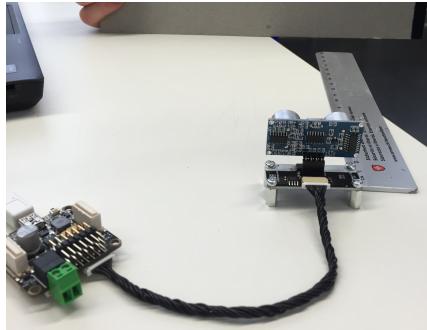
Die verschiedenen Versuchsaufbauten haben gezeigt, dass der Sensor ohne zusätzliche Hilfe die Farbe eines Objektes auf ca. 2cm Entfernung noch genau bestimmen kann. Durch zusätzliche Beleuchtung, kann die Entfernung auf ca. 8cm erhöht werden. Diese Messwerte stimmen jedoch nur, falls es keine direkte Lichteinstrahlung in den Sensor von einer dritten Quelle gibt. Sollte die Lichteinstrahlung zu gross sein, kann diese am Illuminance-Wert, der auch direkt vom Sensor zurückgegeben wird, erkannt werden. Dies kann durch ändern des Gain Wertes und der Integration-Time zu einem gewissen Punkt abgefangen werden.

Die Abbildung 49 zeigt, dass bei einem grünen Container die Messlinie des Grün-Anteils des RGB-Values immer am höchsten ist. Jedoch wird die Distanz zwischen dem Grün-Anteil und den anderen Anteilen kleiner, falls sich der Sensor weiter vom Container entfernt. Das bedeutet, dass in der Software nicht auf einen festen Threshold-Wert geachtet werden darf.

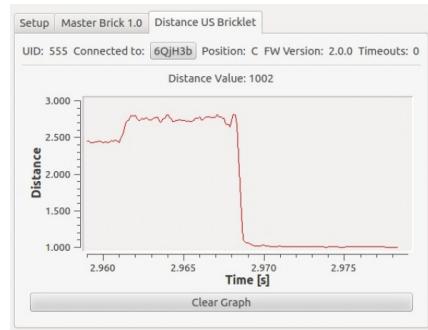
### C.4.3 Ultraschallsensor

Um den Sensor zu testen, wurde dieser über den Servo-Brick an den Computer angeschlossen. Danach wurde mit einem Block ein Objekt in verschiedenen Distanzen simuliert.

Die gemessenen Daten sind in Abbildung 50b zu finden.



(a) Testaufbau



(b) Messwerte

Abb. 50: Testaufbau und Messwerte des Distance US Bricklets

### C.4.4 Fazit Ultraschallsensor

Beim Testen des Distanz US Bricklets konnte festgestellt werden, dass die Messdistanz wie auf dem Datenblatt angegeben ca. vier Metern entspricht. Jedoch hat sich auch gezeigt, dass der Sensor eher ungenau misst. Zudem ist es nicht möglich eine genaue Angabe über die Distanz in mm oder einer anderen Einheit zu erhalten.

Diese Einschränkung sollte jedoch keine Auswirkungen haben. Der einzige Zweck des Sensors ist, das Vorhandensein eines Fahrzeugs auf der Kreuzung zu überprüfen. Die exakte Distanz zum Fahrzeug ist irrelevant.

## C.5 Aufgabenstellung

## Aufgabenstellung PREN1 Herbstsemester 2015

17. September 2015, Version 3  
Adrian Omlin

# Autonomes Entsorgungsfahrzeug

1	Einleitung .....	2
2	Aufgabe .....	2
2.1	Ausblick auf PREN 2 .....	2
3	Randbedingungen.....	3
3.1	Parcours .....	3
3.2	Fahrbahn.....	4
3.3	Abfallcontainer.....	4
3.4	Entsorgungsbecken.....	4
3.5	Zu realisierendes Fahrzeug.....	5
3.6	Wettbewerbskriterien .....	5
3.7	Material und Beschaffung .....	6
3.8	Kosten.....	7
4	Ausführung und Bewertung PREN 1 .....	7

Modulverantwortlicher: Ernst Lüthi

Fachliche Begleitung: De Angelis Marco  
Habegger Jürg  
Joss Marcel  
Klaper Martin  
Koller Thomas  
Lang Udo  
Lustenberger Stefan  
Lüthi Ernst  
Mettler Rolf  
Omlin Adrian  
Thalmann Markus  
Vogel Martin

Version	Datum	Änderung	Verantwortlich
1	16.9.15	Ersterstellung	Adrian Omlin
2	17.9.15	Seite 2 etwa in der Mitte: Korrektur: gelbe blaue	Adrian Omlin
3	29.10.15	Seite 8: Testat 2 in SW8, nicht in SW9	Adrian Omlin

## 1 Einleitung

Die aktuellen Herausforderungen in der Produktentwicklung lassen sich meist nicht mehr von einer einzelnen Disziplin lösen. Deshalb erarbeiten an der Hochschule Luzern - Technik & Architektur Teams aus Studierenden der Studiengänge Elektrotechnik, Informatik und Maschinentechnik Lösungen zu einer interdisziplinären, exemplarischen Aufgabenstellung.

In PREN 1 im Herbstsemester erarbeitet jedes Team ein Lösungskonzept. In PREN 2 im folgenden Frühlingssemester bauen die Teams basierend auf ihrem Lösungskonzept ein Funktionsmuster, um die Tauglichkeit des Konzepts zu beweisen.

Zentral in PREN ist die strukturierte, professionelle Projektabwicklung unter Anwendung des in Kontext 1 und 2 sowie in den fachspezifischen Modulen Gelernten. Die Arbeit soll in späteren Projektaufgaben als Beispiel für die Vorgehensweise und die Projektdokumentation dienen.

## 2 Aufgabe

Das Fahrzeug, das Sie im HS15 und FS16 realisieren, muss in der Lage sein, autonom einer Strasse zu folgen, die richtigen Abfallcontainer zu finden, diese zu leeren und das Entsorgungsgut in einer Deponie zu entsorgen. Diese Aufgabe soll in möglichst kurzer Zeit erledigt werden. Vor dem Startsignal befindet sich Ihr Fahrzeug auf einem Parkplatz. Die Abfallcontainer sind auf der rechten Strassenseite auf dem Trottoir aufgestellt. Es gibt gelbe, grüne und blaue Abfallcontainer. Die gelben gehören einem andern Unternehmen und müssen stehen bleiben. Nebst gelben stehen zwei grüne oder zwei blaue Container am rechten Strassenrand. Diese müssen geleert und wieder hingestellt werden. Der Inhalt ist in einem Entsorgungsbecken zu entleeren.

Das Fahrzeug eines zweiten Teams ist gleichzeitig in Gegenrichtung unterwegs und leert dort die Abfallcontainer. Es ist also mit Gegenverkehr zu rechnen. Es darf nur auf der Strasse gefahren werden. Auf dem Trottoir können sich Fussgänger aufhalten.

Die Hauptaufgabe in PREN 1 ist das Erarbeiten eines Konzeptes. Aus diesem Gesamtkonzept soll auch im Detail ersichtlich sein, wie das Gesamtfunktionsmuster, das Sie in PREN 2 realisieren werden, aufgebaut sein wird.

Der Lösungsansatz für einzelne kritische Teilprobleme muss in PREN 1 durch den Aufbau von Teilstukturmustern verifiziert werden.

### 2.1 Ausblick auf PREN 2

In PREN 2 wird das System basierend auf dem in PREN 1 erarbeiteten Lösungskonzept aufgebaut und ausgetestet.

Als Höhepunkt findet im Rahmen des Kompetenznachweises im Sommer 2016 ein Wettbewerb statt, an dem Sie Ihr Fahrzeug mit denen der anderen Teams messen. Ein Teil der Bewertungspunkte (10 bis 20% der Gesamtpunktzahl von PREN 2) wird entsprechend dem Wettbewerbserfolg vergeben. Bewertet wird in erster Linie die korrekte Ausführung der Aufgabe. Die Zeit wird auch mitberücksichtigt.

### 3 Randbedingungen

#### 3.1 Parcours

Der Parcours ist in Abbildung 1 dargestellt. Diese Abbildung ist nicht massstäblich. Die Position der Abfallcontainer auf dem Trottoir ist beliebig.

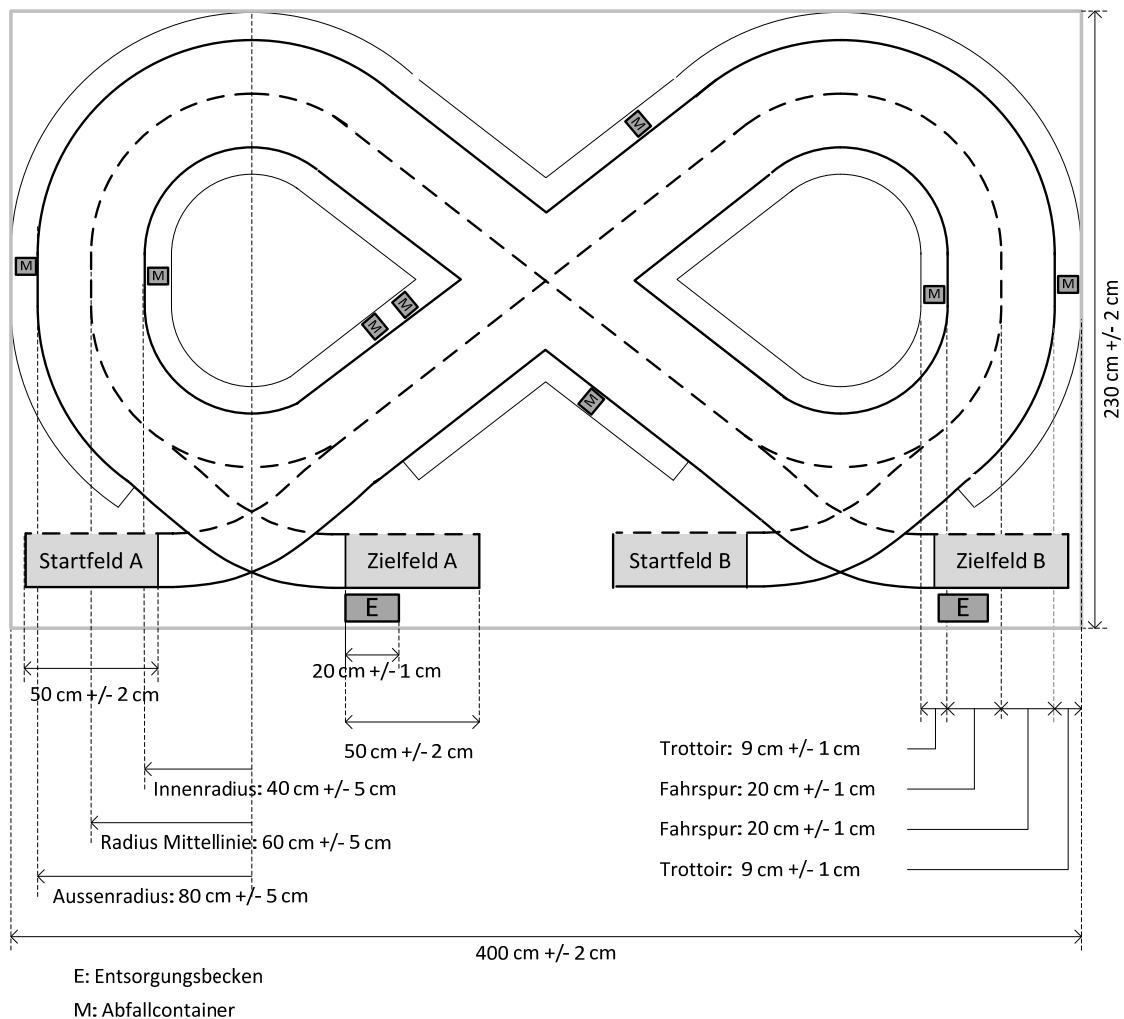


Abbildung 1: Parcours, Draufsicht, nicht massstäblich

Der Parcours wird mit Spanplatten realisiert. Falls er aus mehreren Platten aufgebaut werden muss, werden die Spanplatten mit Nägeln oder Senkkopfschrauben auf einem Grundrahmen befestigt. Auf der Fahrbahn ist also mit kleinen Fugen zu rechnen (Fugenbreite < 2 mm, Überstand < 2 mm).

Das Spielfeld liegt auf dem Boden oder auf einem Tisch. Die Hindernisfreiheit über dem Feld beträgt mindestens 1 m. Der hindernisfreie Raum um das Feld beträgt mindestens 0.5 m.

Das Spielfeld darf nicht verändert werden. Es dürfen beispielsweise keine Führungsschienen oder Navigationsmittel angebracht werden.

### 3.2 Fahrbahn

Die Strassenoberfläche wird grau gestrichen. Die beiden Fahrbahnhälften sind durch eine weisse, gestrichelte Linie getrennt.

Die Fahrspurbreite beträgt 20 cm +/- 1 cm. Das ist der Abstand zwischen der Mitte der Strassenmittellinie und dem rechten Rand der Fahrbahn.

Die Mittellinie ist weiss, gestrichelt und 1 cm +/- 0.2 cm breit.

Der rechte Rand der Fahrspur ist:

- durch die Trottoirkante gegeben (dort, wo ein Trottoir ist),
- mit einer weissen Linie markiert (z.B. bei der Einfahrt ins Zielfeld)
- oder gar nicht gekennzeichnet (z.B. auf der Kreuzung).

Die weisse Linie ist, wo vorhanden, 1 cm +/- 0.2 cm breit. Ihre Mitte verläuft entlang dem Strassenrand, hat also zur Mitte der Mittellinie einen Abstand von 20 cm +/- 1 cm.

Der minimale Kurvenradius der Strassenmittellinie beträgt 60 cm +/- 5 cm.

Das Trottoir ist 9 cm +/- 1 cm breit. Die Trottoirkante hat eine Höhe von 0.5 cm +/- 0.1 cm.

Auf der Kreuzung gilt Rechtsvortritt.

Auf dem Trottoir können sich Fussgänger aufhalten. Auch ist mit Bauarbeitern, Abschrankungen etc. zu rechnen. Personen und Material sind mindestens 10 cm von den Mülltonnen entfernt.

Das Start- und Zielparkfeld ist 50 cm +/- 2 cm lang und 20 cm +/- 1 cm breit. Auf drei Seiten ist der Rand mit einer weissen, 1 cm +/- 0.2 cm breiten Linie markiert. Die vom Fahrzeug aus gesehene linke Begrenzung ist wie die Mittellinie gekennzeichnet.

### 3.3 Abfallcontainer

Die Abfallcontainer sind ca. 8 cm hoch, 6 cm tief und 5 cm breit. Hergestellt werden Sie von der Firma Bruder im Massstab 1:16. Sie sind blau, grün oder gelb.

Die Abfallcontainer stehen auf dem Trottoir und überragen dieses nicht. Sie werden von Hand positioniert. Der Haltegriff der Container ist der Strasse zugewandt. Die Seitenlinie des Containers weicht maximal 20° von der Trottoirkante ab. Stehen zwei Container nebeneinander, beträgt der Abstand dazwischen mindestens 5 cm.

Die Abfallcontainer stehen nur auf den geraden Streckenabschnitten.

Der leere Container hat ein Gewicht von ca. 24 g. Er wird mit maximal 50 g Entsorgungsgut beladen. Das Entsorgungsgut ist schüttbar (z.B. Schrauben, Muttern oder Kügelchen, etc.).

### 3.4 Entsorgungsbecken

Das Entsorgungsbecken hat eine rechteckige Grundfläche mit einer Länge von 20 cm +/- 1 cm einer Breite von 10 cm +/- 1 cm. Die Wandhöhe beträgt 2 cm +/- 0.5 cm. Die Wand ist maximal 1 cm stark.

Auf dem Parcours sind zwei Entsorgungsbecken platziert. Sie stehen am rechten Strassenrand. Der Abstand von der Innenkante der Wand zum inneren, d.h. dem Fahrzeug zugewandten Rand der Fahrbahnbegrenzungslinie beträgt 2 cm +/- 1 cm (horizontal gemessen).

### 3.5 Zu realisierendes Fahrzeug

Das Fahrzeug muss eine Eigenkonstruktion sein. Einzelne Systemkomponenten wie z.B. Servos, das Lenkgetriebe eines Modellautos, ein Mikrocontrollerboard oder eine Kamera dürfen zugekauft und eingesetzt werden.

Das Fahrzeug muss die Aufgabe autonom bewältigen. Nach dem Startbefehl dürfen keine Eingriffe mehr vorgenommen werden. Insbesondere muss das Fahrzeug die Position der Abfallcontainer selbstständig finden. Zum Starten darf das Fahrzeug berührt werden (z.B. Startknopf auf dem Fahrzeug). Ein automatischer Start oder ein Start über Funk ist natürlich eleganter. Es ist nicht erlaubt, das Fahrzeug von einem stationären Rechner aus zu steuern. Auch müssen sich eine allfällige Kamera und andere Sensoren auf dem Fahrzeug befinden.

Die maximalen Abmessungen des Fahrzeugs während der Fahrt und auf dem Parkplatz betragen:

Breite: 15 cm (das entspricht im Massstab 1:16 ca. 2.4 m)

Länge: 35 cm (das entspricht im Massstab 1:16 ca. 5.6 m)

Höhe: 20 cm (das entspricht im Massstab 1:16 ca. 3.2 m)

Während dem Aufnehmen und Abladen des Entsorgungsgutes dürfen diese Abmessungen in einem vernünftigen Maße überschritten werden. Ein Kreuzen mit einem Fahrzeug eines anderen Teams auf einer zweispurigen Strecke muss natürlich möglich sein.

### 3.6 Wettbewerbskriterien

Am Wettbewerb anlässlich des Kompetenznachweises in PREN 2 haben Sie vor dem Start maximal 2 Minuten Zeit, um das Fahrzeug startklar zu machen. Die Position der Abfallcontainer ist noch nicht bekannt.

Vor dem Startsignal steht das Fahrzeug auf dem zugewiesenen Startfeld.

Das Startsignal erfolgt akustisch durch Zählen („Drei, Zwei, Eins, Start!“).

Die Abfallcontainer und allfällige Hindernisse werden nach dem Startsignal nicht mehr verschoben. Wie gesagt ist aber ein weiteres Fahrzeug auf dem Parcours unterwegs.

Die Endzeit wird mit einer Stoppuhr gemessen. Die Zeit wird genommen, wenn Ihr Fahrzeug das Entsorgungsgut abgeladen hat und auf dem Endparkfeld steht.

Die maximal zulässige Zeit beträgt 4 Minuten. Nach dieser Zeit wird der Vorgang abgebrochen.

Folgende Kriterien werden gewertet:

- Kollisionsfreie Fahrt ohne Verlassen der Fahrbahn.
- Beide Abfallcontainer gefunden, geleert und wieder korrekt auf dem Trottoir abgestellt.
- Das Entsorgungsgut im Becken entsorgt.
- Das Fahrzeug korrekt parkiert.

Wird der erste Abfallcontainer gefunden, geleert und nachher wieder richtig platziert, gibt es 3 Punkte.

Wird der zweite Abfallcontainer gefunden, geleert und nachher wieder richtig platziert, gibt es 3 Punkte.

Wird das Entsorgungsgut im vollständig Becken entsorgt, gibt es 3 Punkte.

Für die Zeit gibt es maximal 6 Punkte. Das schnellste Team erhält 6 Punkte, das langsamste 0 Punkte. Dazwischen werden die Punkte entsprechend der gestoppten Zeit linear verteilt.

Abzüge von den 3 möglichen Punkten je Container:

Der Container wird nicht vollständig geleert oder Entsorgungsgut wird verschüttet: - 1 Punkt

Der Container wird offensichtlich richtig erkannt aber nicht geleert: - 2 Punkte

Der Container steht nach dem Leeren nicht korrekt auf dem Trottoir: - 1 Punkt

Abzüge von den 3 möglichen Punkten fürs Entsorgen:

Ein kleiner Teil des Entsorgungsgutes wird verschüttet:

- 1 Punkt

Ein grosser Teil des Entsorgungsgutes wird verschüttet:

- 2 Punkte

Maximal sind also 15 Punkte möglich (Auftrag vollständig erfüllt und Bestzeit). Von der erreichten Gesamtpunktzahl werden weiter folgende Punkte abgezogen:

Verlässt das Fahrzeug die Fahrbahn um maximal 4 cm, pro Vorfall:

- 1 Punkte

Verlässt das Fahrzeug die Fahrbahn um maximal 10 cm, pro Vorfall:

- 2 Punkte

Verlässt das Fahrzeug die Fahrbahn um mehr als 10 cm, kollidiert es mit einem Hindernis oder dem entgegenkommenden Fahrzeug oder fährt es einen Fussgänger an, ist der Auftrag nicht erfüllt und der Durchgang wird mit 0 Punkten bewertet.

Überragt das Fahrzeug am Ende der Spielzeit den Parkplatz um maximal 5 cm:

- 1 Punkt

Überragt das Fahrzeug am Ende der Spielzeit den Parkplatz um maximal 15 cm,

- 2 Punkte

Wird das Parkfeld um mehr als 15 cm überragt, wird der Durchgang mit 0 Punkten bewertet.

Wird ein gelber Container nicht stehen gelassen (berührt, verschoben, umgeworfen) - 3 Punkte

Beispiel:

- Fahrt ohne Kollision
- Beide Container gefunden und geleert. Bei einem wurde etwas Entsorgungsgut verschüttet, der andere Container liegt nach dem Leeren auf dem Trottoir:  
(3 Punkte – 1 Punkte) plus (3 Punkte – 1 Punkte) = 4 Punkte
- Das Entsorgungsgut wird vollständig im Becken entsorgt: 3 Punkte
- Das Fahrzeug fährt einmal kurz aufs Trottoir: - 1 Punkt
- Das Fahrzeug ist nicht ganz korrekt parkiert. Es überragt das Parkfeld um 2 cm: - 1 Punkt
- Die Zeit zum Ausführen des Auftrags wurde mit 2 Minuten 25 Sekunden gemessen. Das Beste Team hatte genau 1 Minute, das schlechteste 3 Minuten: Das gibt 1.75 Punkte

Der Lauf wird mit 6.75 Bewertungspunkten bewertet.

Gemäss diesen Bewertungspunkten wird die Rangliste erstellt. Die Punkte, die in die Notengebung für den Kompetenznachweis PREN 2 einfließen, werden entsprechend der Rangierung bestimmt. Sie sind also nicht identisch mit den Bewertungspunkten. Die Umrechnungstabelle, die zeigt, welcher Rang wie viele Punkte für die Notengebung ergibt, wird in PREN 2 bekanntgegeben.

### 3.7 Material und Beschaffung

Wird bereits in PREN 1 für Tests oder für den Aufbau von Funktionsmustern Material benötigt, so kann der Kauf beim betreuenden Dozierenden beantragt werden. Der Entscheid zur Beschaffung obliegt dem betreuenden Dozenten oder dem Dozententeam.

Damit Sammelbestellungen getätigten werden können, soll das beschaffte Material vorzugsweise von folgenden Lieferanten kommen:

- Conrad Electronic
- Distrelec
- Mädler
- Farnell

Wenn nötig, kann Material auch bei andern Lieferanten bestellt werden.

Wird Material vom Team selber eingekauft, können die Kosten zurückgefördert werden. Das ist nur bei Abgabe des Originals des Kaufbeleges möglich. Eine selbst getätigte Materialbestellung muss auf die Privatadresse erfolgen. Es darf kein Material auf den Namen der Hochschule Luzern Technik & Architektur beschafft werden.

Es wird abgeraten, Material im Ausland zu bestellen, da die Lieferkosten und die Zollgebühren sehr hoch sind und oft beträchtliche Lieferzeiten bestehen.

Die Hochschule hat aus ehemaligen PREN-Durchführungen einiges an Material an Lager wie Servoantriebe, DC- und Schrittmotoren (detaillierte Liste siehe ILIAS). Dieses Material kann ausgeliehen werden.

### 3.8 Kosten

Für den Bau der Teilstücksmuster in PREN 1 und für die Realisierung des Systems in PREN 2 stehen Ihnen als Team insgesamt CHF 500.- zur Verfügung. Davon dürfen maximal CHF 200.- in PREN 1 ausgegeben werden.

Aus diesem Betrag müssen sämtliche Kaufteile sowie allfällige Software bezahlt werden. Die Kosten für Normteile wie Schrauben, Lager, Rohmaterial, Widerstände, Kondensatoren usw. werden nicht verrechnet, sofern die Teile gemäss Lagerliste in den Werkstätten der HSLU - T&A am Lager sind. (Detaillierte Liste siehe ILIAS).

Die Verwendung von „gesponserten“ Komponenten ist möglich. Um kein Team zu benachteiligen, werden diese Komponenten, auch wenn der HSLU keine Auslagen entstehen, mit einem realistischen Preis in die Kostenrechnung einbezogen.

Private Laptops, Computer, Smartphones und Tablets fallen nicht in die Kostenrechnung. Verwendete Netz- und Ladegeräte fallen ebenfalls nicht in die Kostenrechnung, ausser wenn Sie extra für diese Anwendung beschafft und von der Hochschule Luzern bezahlt werden.

Das von der HSLU zum Bau der Teilstücksmuster ausgeliehene Material wird ebenfalls verrechnet, und zwar zum halben Listenpreis. Sobald Sie das Material in einwandfreiem Zustand zurückgeben, wird Ihnen der entsprechende Betrag wieder gutgeschrieben. Wenn Sie das Material in PREN 2 verwenden möchten, wird es Ihnen ebenfalls zum halben Kaufpreis verrechnet.

Die Nutzung von freien Softwarekomponenten oder -services ist zulässig und belastet die Kostenrechnung nicht.

Falls gewünscht, kann von der HSLU ein HCS08 µP-Starterkit ausgeliehen werden.

Es können Bauteile im Rapid Prototyping Verfahren mit dem 3D-Drucker (FDM Verfahren, Werkstoff ABS) der HSLU - T&A hergestellt werden.

Im Fablab lässt sich mit einem Lasergerät Plexiglas und Holz zuschneiden.

Die Kosten für die Arbeitszeit von Mitarbeitenden der HSLU - T&A zur Herstellung von Teilen sind in den oben erwähnten CHF 500.- nicht mit eingerechnet.

Jedem Team stehen für PREN 1 und PREN 2 zusammen folgende Hilfen zur Verfügung:

- maximal 25 h Maschinenlaufzeit des 3D-Druckers
- maximal 1 h Maschinenlaufzeit des Lasergeräts
- maximal 10 Arbeitsstunden des Werkstattpersonals Elektrotechnik
- maximal 10 Arbeitsstunden des Werkstattpersonals Maschinentechnik

## 4 Ausführung und Bewertung PREN 1

Neben der technischen Richtigkeit legen wir unser Augenmerk auch auf die professionelle Abwicklung des Projekts. Dazu gehören unter anderem:

- Kontinuierliche Projektplanung mit Vergleich von Planung und Realität
- Definition der Produktanforderungen in einer Anforderungsliste
- Dokumentation der Technologierecherche
- Risikomanagement
- Erarbeiten von Lösungsvarianten und systematische Lösungsfindung
- Vollständige, verständliche und nachvollziehbare Dokumentation des Gesamtkonzepts inkl. Designüberlegungen

Die Arbeit muss in einem Projektbericht dokumentiert werden. Der Aufbau der Dokumentation basiert auf den Inputs aus dem Kontextmodul 1.

Für die Zulassung zum Kompetenznachweis müssen die folgenden Punkte erfüllt sein:

- Technologierecherche und Anforderungsliste (Testat 1 in SW4)
- Evaluation der Lösungsprinzipien und Auswahl der optimalen Lösungskombination(en) (Testat 2 in ~~SW9~~ SW8)
- Freigabe des Gesamtkonzepts.  
Dokumentation zu 80% fertig gestellt (Testat 3 in SW13)

Für den Kompetenznachweis werden die folgenden Kriterien mit der entsprechenden Gewichtung bewertet (PREN 1):

Kriterien	Gewichtung
<b>Teamarbeit und Arbeitsweise</b> Zusammenarbeit / Arbeitsplanung / Problemerfassung / Konfliktbewältigung / Systematik / Informationsbeschaffung / Interdisziplinarität / Projektmanagement /persönlicher Einsatz / Initiative / Effizienz / Arbeitsmenge	<b>20 %</b>
<b>Resultate und Ergebnisse</b> Innovationsgehalt / technische Machbarkeit / technische Richtigkeit / Einfachheit / Herstellbarkeit / sinnvoller Einsatz von Technologien / Vollständigkeit / Schnittstellen / Wirtschaftlichkeit / Nachvollziehbarkeit / Layout / Softwarearchitektur / Zuverlässigkeit / Ästhetik / Bedienbarkeit  Technologierecherche / Produktanforderung (Teil-)Funktionsmuster	<b>50 %</b>
<b>Dokumentation</b> Formales / Aufbau / Integration der Disziplinen / Sprache / Vollständigkeit / Verständlichkeit / Glaubwürdigkeit / Kohärenz / Abbildungen / Tabellen / Quellenangaben	<b>20 %</b>
<b>Präsentation</b> Beginn / Schluss / Sprache / Inhalt / Verständlichkeit / Glaubwürdigkeit / Vorgehen / nonverbale Aspekte / Einsatz visueller Hilfsmittel	<b>10 %</b>

Wir erwarten eine Zusammenarbeit über die Grenzen der Disziplinen hinweg. Jede Disziplin muss einen nachweisbaren Beitrag zum Erfolg leisten.

Alle Mitglieder des Teams erhalten die gleiche Bewertung. In Ausnahmefällen können einzelne Teammitglieder separat bewertet werden.

Wird ein Team am Kompetenznachweis mit „FX“ bewertet, erhält es die Gelegenheit zur Nachbesserung. Das kann eine Teamaufgabe sein. Alle Teammitglieder erhalten in diesem Fall nach der Nachprüfung ein „F“ oder ein „E“. Es ist auch möglich, dass jedes Teammitglied zur Nachbesserung eine individuelle Aufgabe lösen muss. Nach der Nachprüfung wird für jedes Teammitglied einzeln entschieden, ob es ein „F“ oder ein „E“ erhält.

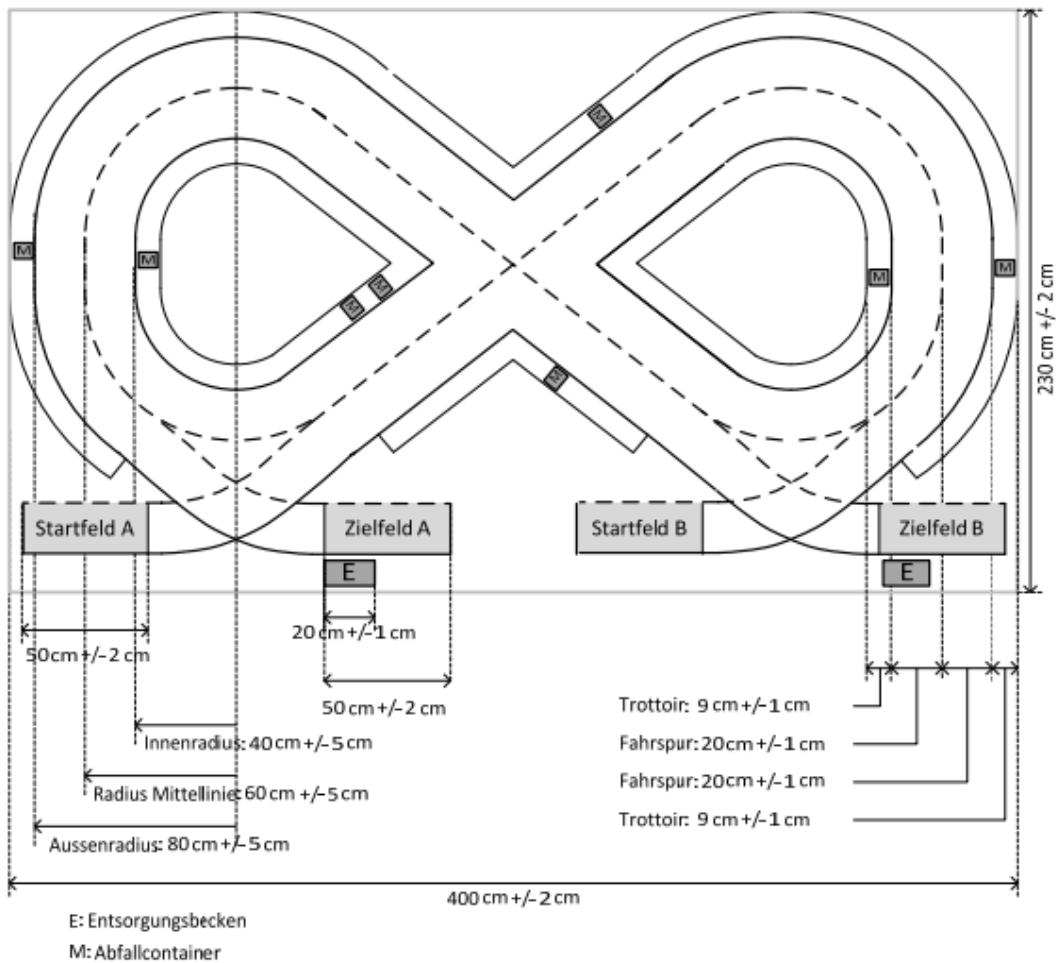
## C.6 Produktanforderungen auf Testat 1

# Produktanforderungen

Nr.	F / M / W	Bezeichnung	Werte Daten Erläuterungen Änderungen	Verantwortlich
<b>1</b>		<b>Fahrzeug</b>		
1.1	F	Lenken	minimaler Radius 35cm	Team
1.2	F	autonomes Fahren	darf nicht von aussen gesteuert werden	Team
1.3	F	Masse	Länge x Breite x Höhe (35cm * max. 15cm * max. 20cm)	Team
1.4	F	Fahrbahn nicht verlassen	Überhang	Team
1.5	W	Gewicht Fahrzeug	<2.5 kg (max. 5 kg)	Team
1.6	M	Beständigkeit gegen Immissionen/ gegen Elemente	Nicht wasserfest, nicht beständig gegenüber Radioaktivität	Team
1.7	M	Design	Funktionstüchtiges Design (form follows function)	Team
<b>2</b>		<b>Hebemechanismus</b>		
2.1	F	Mülltonnengrösse	5cm x 6cm x 8cm	doz
2.2	F	Mülltonnengewicht	50g (Materialgewicht) + 24g (Leergewicht)	doz
2.3	F	Material	Muttern, Schrauben	doz
2.4	F	Entleerung	muss vollständig entleeren	Team
2.5	F	Verschiebungen	alle Objekte werden nach Startsignal nicht mehr verschoben	doz
2.6	F	Zurückstellen	Container muss wieder an den selben Ort zurück gestellt werden	Team
<b>3</b>		<b>Energieversorgung</b>		
3.1	W	Energiespeicher	Wasserstoff, Batterien, Akku	Team
3.2	W	Energieüberwachung	Akku nicht zerstören durch Tiefentladung	Team
3.3	F	Energieverteilung	abgesicherte Spannungsverteilung	Team
3.4	M	Akkulaufzeit	Min. 8 Minuten	Team
3.5	W	Akkulaufzeit	Min. 30 Minuten	Team

## OMA - die Oekologische Müllabfuhr

<b>4</b>					<b>Objekterkennung</b>	
4.1	F	Farberkennung	Kann zwischen drei Farben (grün, blau, gelb) unterscheiden.		Team	
4.2	F	Sammeln 1	Sammelt entweder grüne oder blaue Abfallcontainer (nicht beide gleichzeitig). Beim Start der Fahrt wird ermittelt, welche Farbe eingesammelt werden soll. Die gelben Container werden immer ignoriert.		Team	
4.3	F	Sammeln 2	Erkennt Personen auf dem Trottoir. Diese werden nicht eingesammelt		Team	
4.4	F	Sammeln 3	Erkennt Baustellen (Absperrungen, Schubkarren, Warnschilder). Diese werden nicht eingesammelt.		Team	
4.5	F	Kreuzung (Gewährung Rechtsvortritt)	Erkennt korrekt ein nahendes Fahrzeug an der Kreuzung. Rechtsvortritt wird gewährt.		Team	
4.6	F	Kreuzung (Objekt von links)	Wenn sich ein von links kommendes Fahrzeug bereits in der Kreuzung befindet, wird dieses erkannt und gewartet.		Team	
4.7	F	Kreuzung (Rechtsvortritt)	Bei einem von links kommenden Fahrzeug wird der eigene Rechtsvortritt umgesetzt.		Team	
4.8	F	Sammelstelle	Erkennt die Abfallsammelstelle.		Team	
4.9	F	Parkplatz	Erkennt Parkfeld.		Team	
<b>5</b>					<b>Autonomes fahren</b>	
5.1	F	Fahrbahn	Fährt immer korrekt auf der eigenen Fahrbahn. Wechselt die Fahrbahn nicht. Überholmanöver werden nicht unterstützt.		Team	
5.2	F	Randstein	Erkennt den Randstein / Trottoir. Dieser wird nicht befahren.		Team	
5.3	F	Parkieren	Parkiert korrekt im definierten Bereich (siehe Anforderung 6.5)		Team	
5.4	W	Start	Die Fahrt kann kabellos über einen Knopf (Software/Hardware) gestartet werden.		Team	



6.1	F	Lichtverhältnisse	Min. 200 Lux	doz
6.2	F	Temperatur	5°C - 40°C	doz
6.3	F	Parkfeld	Länge: 50cm +/- 2cm Breite: 20cm +/- 1cm	doz
6.4	F	Strassenoberfläche	grau gestrichen	doz
6.5	F	Fahrbahn	Breite 20cm+/- 1cm	doz
6.6	F	Mittellinie	weiss gestrichelt und 1cm+/-0.2cm	doz
6.7	F	minimaler Kurvenradius	60cm+/-5cm	doz
6.8	F	Trottoirbreite	9cm+/-cm	doz
6.9	F	Trottoirkantenhöhe	0.5cm+/-0.1cm	doz
6.10	F	Personen und Material	mind. 10cm von Mülltonne entfernt	doz
6.11	F	Entsorgungsbecken: Dimensionen	Länge 20 cm +/- 1 cm, Breite 10 cm +/- 1 cm, Höhe 2 cm +/- 0.5 cm	doz
6.12	F	Entsorgungsbecken: Wandstärke	max. 1 cm	doz
6.13	F	Platzierung Entsorgungs- container zu Strassenrand	Max 3 cm vom Strassenrand entfernt mit Griff in Richtung der Strasse	doz
6.14	F	Abstand zwischen Mülltonnen	Mind. 5 cm	doz

## OMA - die Oekologische Müllabfuhr

6.15	F	Platzierung Mülltonnen	nur an geraden Streckenabschnitten, max 20° von Trottoirkante abweichend aufgestellt	doz
6.16	F	Gesamtkosten	die Kosten müssen unter 500 CHF gehalten werden	doz
6.17	F	Mülltonnenfarbe	Gelb, grün oder blau	doz
6.18	F	Luftfeuchtigkeit	≤ 70%	doz
<b>7</b>		<b>Entsorgung</b>		
7.1	W	Ausleeren des Mülls	in Entsorgungsbecken oder auf Rand / Nur Müll im Container	Team
<b>8</b>		<b>Diverses</b>		
8.1	W	Überwachung des Roboters	In einem Gui können all die Sensoren, Motoren und ev. Kamera überwacht werden (Debuggen)	Team
8.2	W	Zeit für einen Durchlauf	Max. 2 Minuten	Team
8.3	F	Zeit für einen Durchlauf	Max. 4 Minuten	Team

F = Festanforderung

M = Mindestanforderung

W = Wunschanforderung

doz = Dozenten

## C.7 Morphologischer Kasten

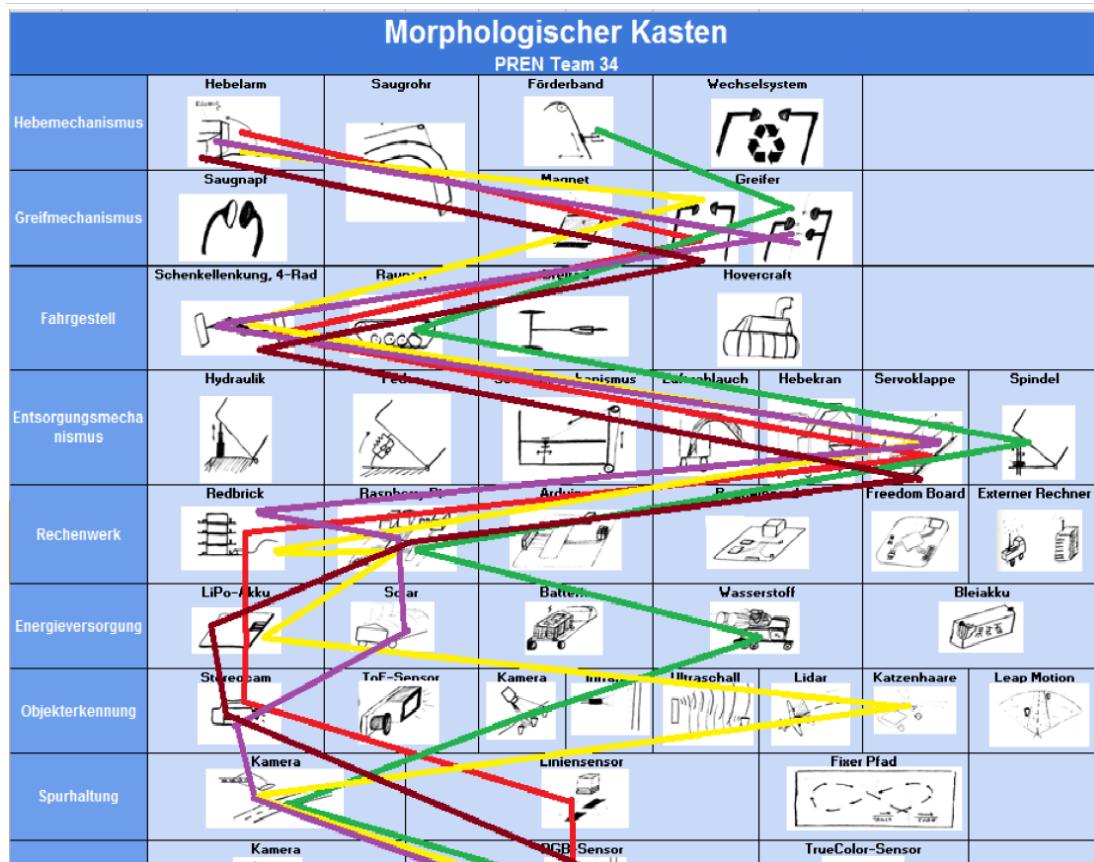


Abb. 51: Morphologischer Kasten

## C.8 Entsorgung

Für die Teilfunktion Entsorgung wurde in der Variante 1, die mithilfe des morphologischen Kastens ausgewählt wurde, eine Mulde mit Klappe gewählt. Bei der Umsetzung ergaben sich jedoch Probleme: Die berechnete benötigte Steigung, damit der Müll herausfällt, belief sich auf ca.  $30^\circ$ . Damit wäre die Mulde allerdings sehr viel höher geworden, als der Greifarm den Mülleimer hätte heben können. Daher wurde die Mulde angepasst und anstelle der Lösungsvariante mit Klappe eine Kippmulde eingesetzt.

### C.8.1 Berechnungen

$$\mu_{Mulde} = 0.5$$

$$m_{Muell} = 0.1kg$$

$$F_g = m_{Muell} \cdot 9.81 \frac{kg \cdot m}{s^2} = 0.981N$$

**benötigte Steigung**

$$Hangabtriebskraft \Rightarrow F_a = F_g \cdot \sin(\alpha)$$

$$Normalkraft \Rightarrow F_n = F_g \cdot \cos(\alpha)$$

$$Haftreibungskraft \Rightarrow F_h = \mu_{Mulde} \cdot F_n$$

$$F_a \geq F_h \rightarrow \alpha \geq 26.565$$

**benötigtes Moment**

$$V_{Klappe} = 5cm \cdot 6.1cm \cdot 0.56cm = 17.2cm^3$$

$$\rho_{Plexiglas} = 1.18 \frac{g}{cm^3}$$

$$m_{Klappe} = \rho_{Plexi} \cdot V_{Mulde} = 1.18 \frac{g}{cm^3} \cdot 17.2cm^3 = 20.2g$$

$$F_{glk} = m_{Klappe} \cdot 9.81 \frac{kg \cdot m}{s^2} = 0.202N$$

$$F_{Servo} = F_{Gesamt} \cdot \sin(\beta) = 0.202N \cdot \sin(60^\circ) = 1N$$

$$F_{ab} = m_{Muell} \cdot g \cdot \sin(\beta) = 0.1kg \cdot 9.81 \frac{kg \cdot m}{s^2} \cdot \sin(60^\circ) = 0.87N$$

$$M_{Servo} = F_{ab} \cdot sArm = 0.87N \cdot 5cm = 4\text{Ncm}$$

### C.8.2 Funktionsweise

Der gesammelte Müll wird in einer Mulde mit schrägem Boden zum Entsorgungsbecken transportiert. Die Mulde ist mit einer Klappe versehen. Ein Riegel hält die Klappe geschlossen. Beim Entsorgungsbecken hebt ein Servomotor den Riegel an und senkt die Klappe. Durch die Schräge der Mulde wird die Schwerkraft zur Entleerung genutzt. Sobald die Mulde leer ist, senkt der Servomotor den Riegel. Über ein Scharnier wird die Klappe wieder geschlossen.

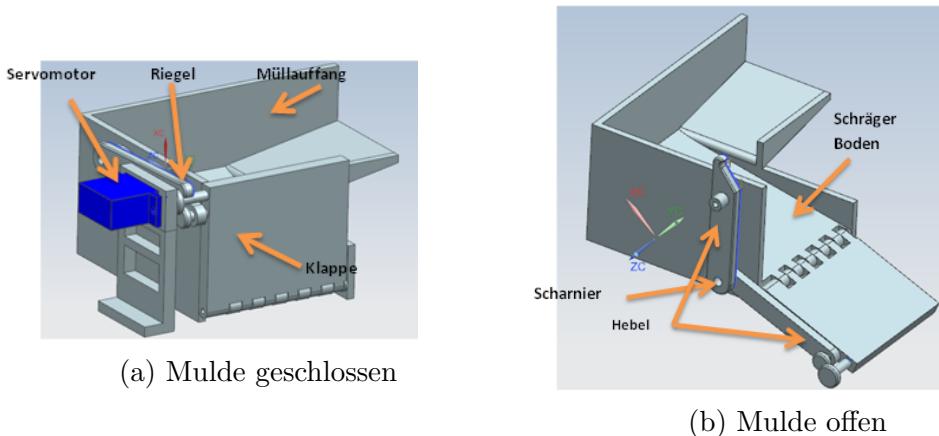


Abb. 52: Mulde offen/geschlossen

### C.8.3 Detaillierte Beschreibung

Die Mulde besitzt einen Müllauffang, in den der Greifarm die Container entleert. Von dort rutscht der Müll hinab in das Innere der Mulde. Diese besitzt einen Boden mit einer Schräge von ca. 30°. Dadurch liegt das Gewicht des Mülls auf der Klappe. Diese wird durch einen Riegel verschlossen gehalten. Die Belastung für den Riegel beläuft sich auf ein Moment von 4 Ncm. Um den Riegel zu öffnen, wird ein Servomotor angesteuert. Die Klappe öffnet sich auf ca. 105°. Der Boden der Mulde ist über eine Achse mit der Klappe verbunden. Um zu verhindern, dass der Müll an der Verbindungsstelle hängen bleibt, wird die Konstruktion mit einer Folie versehen. Das Schliessen der Klappe erfolgt über denselben Servomotor. Der Riegel ist über ein Scharnier mit der

Klappe verbunden. Dieses ist ähnlich konstruiert wie eine Klappenschere. Dadurch wird die zum Schliessen benötigte Kraft auf ca. 0.175 N eingeschätzt. Die Mulde selbst wird 3D-gedruckt. Für die Klappe kann auch Plexiglas verwendet werden. benötigte Kraft auf ca. 0.175 N eingeschätzt. Die Mulde selbst wird 3D-gedruckt. Für die Klappe kann auch Plexiglas verwendet werden.

## C.9 Berechnung Klemmkraft

In der unteren Abbildung ist eine Vereinfachung des Greifarms dargestellt, welches die Berechnung der verschiedenen Kräfte und des erforderlichen Moments ermöglicht.

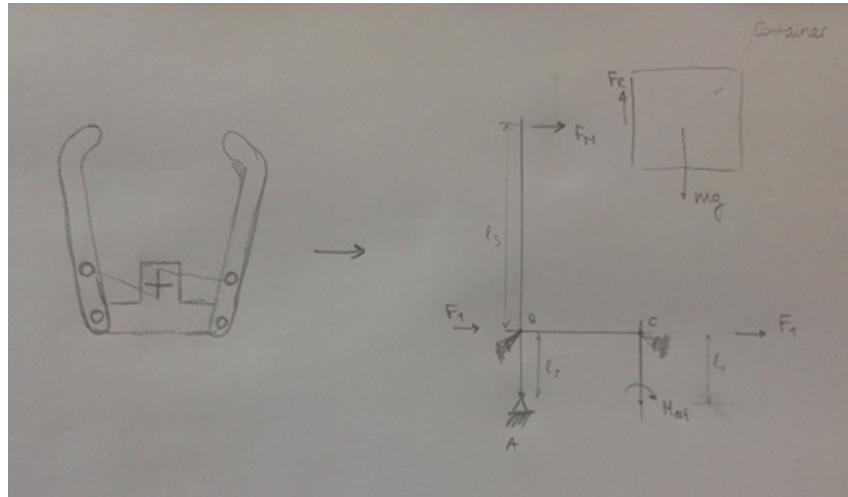


Abb. 53: Vereinfachung des Greifsystems

Um auf das erforderliche Drehmoment zu kommen, werden die benötigten Kräfte am Container wie folgt berechnet. In erster Linie muss die Reibungskraft  $F_R$  größer sein wie die entgegenwirkende Schwerkraft  $F_G$ . Dabei wird noch ein Sicherheitsfaktor  $S_K$  mitberechnet.

$$F_G = m \cdot g$$

$$F_R = \mu \cdot F_N$$

$$S_k \cdot F_G \leq F_R$$

$$S_k \leq \frac{F_R}{F_G} = \frac{\mu \cdot F_N}{m \cdot g}$$

Mit den weiteren Unterteilungen der Gewichts- und Normalkraft kann die Kraft  $F_1$  und das erforderliche Moment  $M_{erf}$  hergeleitet werden:

$$F_N = F_1 \cdot \frac{l_2}{l_3}$$

$$F_1 = \frac{M_{erf}}{l_1}$$

Mit diesen Gleichungen kann so das erforderliche Drehmoment berechnet wer-

den:

$$S_k \leq \frac{\mu}{m \cdot g} \cdot \frac{l_2}{l_3 \cdot l_1} \cdot M_{erf} \rightarrow M_{erf} \geq S_k \cdot \frac{m \cdot g}{\mu} \cdot \frac{l_1 \cdot l_3}{l_2}$$

Mit folgenden Parametern kann so das Drehmoment berechnet werden:

$S_k = 1.5$  Sicherheitsfaktor

$\mu = 0.7$  Reibungswert

$m_{Container} = 100g$  Container Gewicht

$l_1 = 10mm$  Hebelarm Länge 1

$l_2 = 15mm$  Hebelarm Länge 2

$l_3 = 70mm$  Hebelarm Länge 3

$$M_{erf} = S_k \cdot \frac{m \cdot g}{\mu} \cdot \frac{l_1 \cdot l_3}{l_2} = 1.5 \cdot \frac{0.1kg \cdot 9.81 \frac{kg \cdot m}{s^2}}{0.7} \cdot \frac{0.01m \cdot 0.07m}{0.015m} = 0.098Nm = 9.807Ncm$$

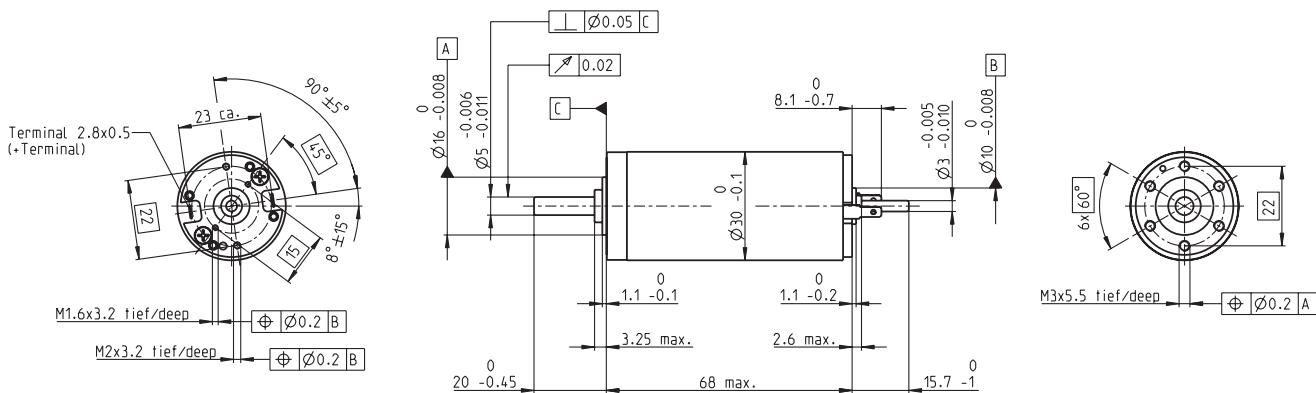
Der Servomotor weist eine Kraft in  $Kg \cdot cm$  auf. Das erforderliche Drehmoment in  $Kg \cdot cm$  umgerechnet ist:

$$M_{erf}[Kg \cdot cm] = \frac{M_{erf}}{g} = \frac{9.807Ncm}{9.81 \frac{kg \cdot m}{s^2}} = 1Kg \cdot cm$$

## C.10 Datenblätter

### C.10.1 Maxon Motor RE30 - 310007

# RE 30 Ø30 mm, Graphitbürsten, 60 Watt



M 1:2

- Lagerprogramm
- Standardprogramm
- Sonderprogramm (auf Anfrage)

## Artikelnummern

gemäss Massbild  
Wellenlänge 15.7 gekürzt auf 8.7 mm

	310005	310006	310007	310008	310009
	268193	268213	268214	268215	268216

## Motordaten

### Werte bei Nennspannung

1 Nennspannung	V	12	18	24	36	48		
2 Leerlaufdrehzahl	min <sup>-1</sup>	8170	8590	8810	8590	8490		
3 Leerlaufstrom	mA	301	213	165	106	78.6		
4 Nenndrehzahl	min <sup>-1</sup>	7630	7910	8050	7840	7760		
5 Nennmoment (max. Dauerdrehmoment)	mNm	51.6	75.5	85.6	86.6	89.7		
6 Nennstrom (max. Dauerbelastungsstrom)	A	4	4	3.47	2.28	1.74		
7 Anhaltemoment	mNm	853	1000	1020	1000	1050		
8 Anlaufstrom	A	61.1	50.3	39.3	25.2	19.6		
9 Max. Wirkungsgrad	%	85	87	87	87	88		
<b>Kenndaten</b>								
10 Anschlusswiderstand	Ω	0.196	0.358	0.611	1.43	2.45		
11 Anschlussinduktivität	mH	0.034	0.07	0.119	0.281	0.513		
12 Drehmomentkonstante	mNm/A	13.9	19.9	25.9	39.8	53.8		
13 Drehzahlkonstante	min <sup>-1</sup> /V	685	479	369	240	178		
14 Kennliniensteigung	min <sup>-1</sup> /mNm	9.64	8.61	8.7	8.61	8.09		
15 Mechanische Anlaufzeitkonstante	ms	3.4	3.24	3.05	2.98	2.94		
16 Rotorträgheitsmoment	gcm <sup>2</sup>	33.7	35.9	33.5	33.1	34.7		

## Spezifikationen

### Thermische Daten

17 Therm. Widerstand Gehäuse-Luft	6.0 K/W
18 Therm. Widerstand Wicklung-Gehäuse	1.7 K/W
19 Therm. Zeitkonstante der Wicklung	16.3 s
20 Therm. Zeitkonstante des Motors	593 s
21 Umgebungstemperatur	-30...+100°C
22 Max. Wicklungstemperatur	+125°C

### Mechanische Daten (Kugellager)

23 Grenzdrehzahl	12000 min <sup>-1</sup>
24 Axialspiel	0.05 - 0.15 mm
25 Radialspiel	0.025 mm
26 Max. axiale Belastung (dynamisch)	5.6 N
27 Max. axiale Aufpresskraft (statisch) (statisch, Welle abgestützt)	110 N
28 Max. radiale Belastung, 5 mm ab Flansch	1200 N
	28 N

### Weitere Spezifikationen

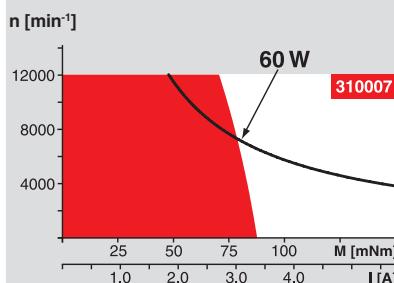
29 Polpaarzahl	1
30 Anzahl Kollektorsegmente	13
31 Motorgewicht	260 g

Motordaten gemäss Tabelle sind Nenndaten.  
Erläuterungen zu den Ziffern Seite 107.

### Option

Vorgespannte Kugellager

## Betriebsbereiche



## Legende

### Dauerbetriebsbereich

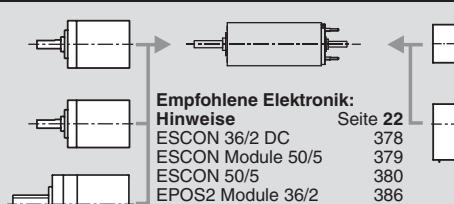
Unter Berücksichtigung der angegebenen thermischen Widerstände (Ziffer 17 und 18) und einer Umgebungstemperatur von 25°C wird bei dauernder Belastung die maximal zulässige Rotortemperatur erreicht = thermische Grenze.

### Kurzzeitbetrieb

Der Motor darf kurzzeitig und wiederkehrend überlastet werden.

### Typenleistung

## maxon Baukastensystem



## Übersicht Seite 20–25

### Planetengetriebe

Ø32 mm  
0.75 - 6.0 Nm

Seite 303–309

### Koaxdrive

Ø32 mm  
1.0 - 4.5 Nm  
Seite 312

### Spindelgetriebe

Ø32 mm  
Seite 334–336

### Empfohlene Elektronik:

Hinweise	Seite 22	
ESCON 36/2 DC	378	
ESCON Module 50/5	379	
ESCON 50/5	380	
EPOS2 Module 36/2	386	
EPOS2 24/5, EPOS2 50/5	387	
EPOS2 P 24/5	390	
EPOS3 70/10 EtherCAT	393	
MAXPOS 50/5	396	

### Encoder MR

256 - 1024 Imp.,  
3 Kanal  
Seite 356

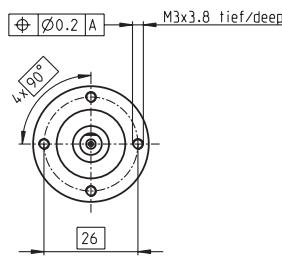
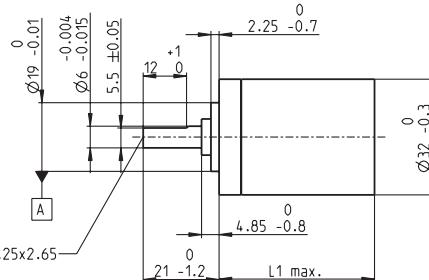
### Encoder HED\_5540

500 Imp.,  
3 Kanal  
Seite 362/364

**C.10.2 Maxon Getriebe GP 32 C - 166931**

# Planetengetriebe GP 32 C Ø32 mm, 1.0–6.0 Nm

Keramikversion



## Technische Daten

Planetengetriebe	geradeverzahnt
Abtriebswelle	rostfreier Stahl
Wellendurchmesser als Option	8 mm
Abtriebswellenlagerung	Kugellager
Radialspiel, 5 mm ab Flansch	max. 0.14 mm
Axialspiel	max. 0.4 mm
Max. axiale Belastung (dynamisch)	120 N
Max. axiale Aufpresskraft	120 N
Drehinn., Antrieb zu Abtrieb	=
Max. Eingangsrehzahl dauernd	8000 min <sup>-1</sup>
Empfohlener Temperaturbereich	-40...+100°C
Stufenzahl	1 2 3 4 5
Max. radiale Belastung, 10 mm ab Flansch	90 N 140 N 200 N 220 N 220 N

M 1:2

Option: Geräuschreduzierte Ausführung

- Lagerprogramm
- Standardprogramm
- Sonderprogramm (auf Anfrage)

## Artikelnummern

	166930	166933	166938	166939	166944	166949	166954	166959	166962	166967	166972	166977
<b>Getriebedaten</b>												
1 Untersetzung	3.7:1	14:1	33:1	51:1	111:1	246:1	492:1	762:1	1181:1	1972:1	2829:1	4380:1
2 Untersetzung absolut	26/7	676/49	529/46	17576/343	13824/125	421824/1715	86112/175	19044/25	10123776/8575	8626176/4375	495144/175	109503/25
3 Max. Motorwellendurchmesser	mm	6	6	3	6	4	4	3	3	4	4	3
<b>Artikelnummern</b>	<b>166931</b>	<b>166934</b>		<b>166940</b>	<b>166945</b>	<b>166950</b>	<b>166955</b>	<b>166960</b>	<b>166963</b>	<b>166968</b>	<b>166973</b>	<b>166978</b>
1 Untersetzung	4.8:1	18:1		66:1	123:1	295:1	531:1	913:1	1414:1	2189:1	3052:1	5247:1
2 Untersetzung absolut	24/5	624/35		16224/245	6877/56	101062/343	331776/625	36501/40	2425488/1715	536406/245	190712/625	839523/160
3 Max. Motorwellendurchmesser	mm	4	4		4	3	3	4	3	3	3	3
<b>Artikelnummern</b>	<b>166932</b>	<b>166935</b>		<b>166941</b>	<b>166946</b>	<b>166951</b>	<b>166956</b>	<b>166961</b>	<b>166964</b>	<b>166969</b>	<b>166974</b>	<b>166979</b>
1 Untersetzung	5.8:1	21:1		79:1	132:1	318:1	589:1	1093:1	1526:1	2362:1	3389:1	6285:1
2 Untersetzung absolut	23/4	299/14		3887/49	3312/25	389376/1225	20631/35	278941/256	9345024/6125	2066688/675	474513/140	6436343/1024
3 Max. Motorwellendurchmesser	mm	3	3		3	3	4	3	3	3	3	3
<b>Artikelnummern</b>	<b>166936</b>		<b>166942</b>	<b>166947</b>	<b>166952</b>	<b>166957</b>		<b>166965</b>	<b>166970</b>	<b>166975</b>		
1 Untersetzung	23:1		86:1	159:1	411:1	636:1		1694:1	2548:1	3656:1		
2 Untersetzung absolut		<sup>576/25</sup>		14976/175	1587/10	359424/875	79488/125		1162213/686	7982624/3125	457056/125	
3 Max. Motorwellendurchmesser	mm	4		4	3	4	3		3	4	3	
<b>Artikelnummern</b>	<b>166937</b>		<b>166943</b>	<b>166948</b>	<b>166953</b>	<b>166958</b>		<b>166966</b>	<b>166971</b>	<b>166976</b>		
1 Untersetzung	28:1		103:1	190:1	456:1	706:1		1828:1	2623:1	4060:1		
2 Untersetzung absolut		<sup>138/5</sup>		3588/35	12167/64	89401/196	158171/224		2238912/1225	2056223/784	3637933/896	
3 Max. Motorwellendurchmesser	mm	3		3	3	3	3		3	3	3	
<b>Artikelnummern</b>	<b>166938</b>		<b>166944</b>	<b>166949</b>	<b>166954</b>	<b>166959</b>		<b>166966</b>	<b>166971</b>	<b>166976</b>		
1 Untersetzung	28:1		103:1	190:1	456:1	706:1		1828:1	2623:1	4060:1		
2 Untersetzung absolut		<sup>138/5</sup>		3588/35	12167/64	89401/196	158171/224		2238912/1225	2056223/784	3637933/896	
3 Max. Motorwellendurchmesser	mm	3		3	3	3	3		3	3	3	
<b>Stufenzahl</b>	1	2	2	3	3	4	4	4	5	5	5	5
5 Max. Dauerdrehmoment	Nm	1	3	3	6	6	6	6	6	6	6	6
6 Kurzzeitig zulässiges Drehmoment	Nm	1.25	3.75	3.75	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5
7 Max. Wirkungsgrad	%	80	75	75	70	70	60	60	50	50	50	50
8 Gewicht	g	118	162	162	194	194	226	226	226	258	258	258
9 Mittleres Getriebespiel unbelastet	°	0.7	0.8	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
10 Massenträgheitsmoment	gcm <sup>2</sup>	1.5	0.8	0.8	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
11 Getriebelänge L1	mm	26.5	36.4	36.4	43.1	43.1	49.8	49.8	56.5	56.5	56.5	56.5

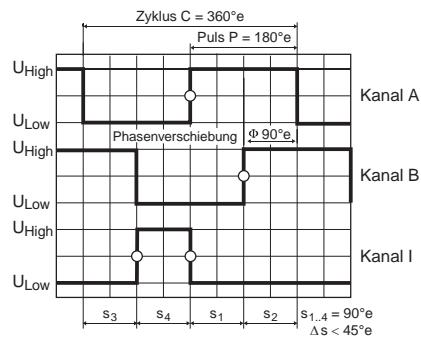


## maxon Baukastensystem

+ Motor	Seite	+ Sensor/Bremse	Seite	Gesamtlänge [mm] = Motorlänge + Getriebelänge + (Sensor/Bremse) + Montageteile
RE 25, 10 W	135/137			81.1 91.0 91.0 97.7 97.7 104.4 104.4 104.4 111.1 111.1 111.1 111.1
RE 25, 10 W	135/137 MR			92.1 102.0 102.0 108.7 108.7 115.4 115.4 115.4 122.1 122.1 122.1 122.1
RE 25, 10 W	135/137 Enc 22			95.2 105.1 105.1 111.8 111.8 118.5 118.5 118.5 125.2 125.2 125.2 125.2
RE 25, 10 W	135/137 HED_ 5540			101.9 111.8 111.8 118.5 118.5 125.2 125.2 125.2 131.9 131.9 131.9 131.9
RE 25, 10 W	135/137 DCT 22			103.4 113.3 113.3 120.0 120.0 126.7 126.7 126.7 133.4 133.4 133.4 133.4
RE 25, 20 W	136			69.6 79.5 79.5 86.2 86.2 92.9 92.9 92.9 99.6 99.6 99.6 99.6
RE 25, 20 W	136 MR			80.6 90.5 90.5 97.2 97.2 103.9 103.9 103.9 110.6 110.6 110.6 110.6
RE 25, 20 W	136 HED_ 5540			90.4 100.3 100.3 107.0 107.0 113.7 113.7 113.7 120.4 120.4 120.4 120.4
RE 25, 20 W	136 DCT22			91.9 101.8 101.8 108.5 108.5 115.2 115.2 115.2 121.9 121.9 121.9 121.9
RE 25, 20 W	136 AB 28			103.7 113.6 113.6 120.3 120.3 127.0 127.0 127.0 133.7 133.7 133.7 133.7
RE 25, 20 W	136 HED_ 5540/AB 28			120.9 130.8 130.8 137.5 137.5 144.2 144.2 144.2 150.9 150.9 150.9 150.9
RE 25, 20 W	137 AB 28			115.2 125.1 125.1 131.8 131.8 138.5 138.5 138.5 145.2 145.2 145.2 145.2
RE 25, 20 W	137 HED_ 5540/AB 28			132.4 142.3 142.3 149.0 149.0 155.7 155.7 155.7 162.4 162.4 162.4 162.4
RE 30, 60 W	139			94.6 104.5 104.5 111.2 111.2 117.9 117.9 117.9 124.6 124.6 124.6 124.6
RE 30, 60 W	139 MR			106.0 115.9 115.9 122.6 122.6 129.3 129.3 129.3 136.0 136.0 136.0 136.0
RE 30, 60 W	139 HED_ 5540			115.4 125.3 125.3 132.0 132.0 138.7 138.7 138.7 145.4 145.4 145.4 145.4
RE 35, 90 W	140			97.6 107.5 107.5 114.2 114.2 120.9 120.9 120.9 127.6 127.6 127.6 127.6
RE 35, 90 W	140 MR			109.0 118.9 118.9 125.6 125.6 132.3 132.3 132.3 139.0 139.0 139.0 139.0
RE 35, 90 W	140 HED_ 5540			118.3 128.2 128.2 134.9 134.9 141.6 141.6 141.6 148.3 148.3 148.3 148.3
RE 35, 90 W	140 DCT 22			115.7 125.6 125.6 132.3 132.3 139.0 139.0 139.0 145.7 145.7 145.7 145.7
RE 35, 90 W	140 AB 28			133.7 143.6 143.6 150.3 150.3 157.0 157.0 157.0 163.7 163.7 163.7 163.7
RE 35, 90 W	140 HEDS 5540/AB 28			150.9 160.8 160.8 167.5 167.5 174.2 174.2 174.2 180.9 180.9 180.9 180.9
A-max 26	161-168			71.3 81.2 81.2 87.9 87.9 94.6 94.6 94.6 101.3 101.3 101.3 101.3
A-max 26	162-168 MEnc 13			78.4 88.3 88.3 95.0 95.0 101.7 101.7 101.7 108.4 108.4 108.4 108.4
A-max 26	162-168 MR			80.1 90.0 90.0 96.7 96.7 103.4 103.4 103.4 110.1 110.1 110.1 110.1
A-max 26	162-168 Enc 22			85.7 95.6 95.6 102.3 102.3 109.0 109.0 109.0 115.7 115.7 115.7 115.7
A-max 26	162-168 HED_ 5540			89.7 99.6 99.6 106.3 106.3 113.0 113.0 113.0 119.7 119.7 119.7 119.7
A-max 32	169/171			89.5 99.4 99.4 106.1 106.1 112.8 112.8 112.8 119.5 119.5 119.5 119.5
A-max 32	170/172			88.1 98.0 98.0 104.7 104.7 111.4 111.4 111.4 118.1 118.1 118.1 118.1
A-max 32	170/172 MR			99.3 109.2 109.2 115.9 115.9 122.6 122.6 122.6 129.3 129.3 129.3 129.3
A-max 32	170/172 HED_ 5540			108.9 118.8 118.8 125.5 125.5 132.2 132.2 132.2 138.9 138.9 138.9 138.9

**C.10.3 Maxon Tacho MR - 228452**

# Encoder MR Typ L, 256–1024 Impulse, 3 Kanal, mit Line Driver



■ Lagerprogramm  
■ Standardprogramm  
■ Sonderprogramm (auf Anfrage)

## Typ

Impulszahl pro Umdrehung	256	500	512	1000	1024
Anzahl Kanäle	3	3	3	3	3
Max. Impulsfrequenz (kHz)	80	200	160	200	320
Max. Drehzahl (min <sup>-1</sup> )	18750	24000	18750	12000	18750



## Artikelnummern

225783	228452	225785	228456	225787
--------	--------	--------	--------	--------

## maxon Baukastensystem

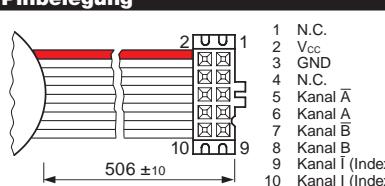
+ Motor	Seite	+ Getriebe	Seite	+ Bremse	Seite	Gesamtlänge [mm] / ● siehe Getriebe				
RE 30, 15 W	138					79.4	79.4	79.4	79.4	79.4
RE 30, 15 W	138	GP 32, 0.75 - 4.5 Nm	305			●	●	●	●	●
RE 30, 60 W	139					79.4	79.4	79.4	79.4	79.4
RE 30, 60 W	139	GP 32, 0.75 - 4.5 Nm	303			●	●	●	●	●
RE 30, 60 W	139	GP 32, 0.75 - 6.0 Nm	305-309			●	●	●	●	●
RE 30, 60 W	139	GP 32 S	334-336			●	●	●	●	●
RE 35, 90 W	140					82.4	82.4	82.4	82.4	82.4
RE 35, 90 W	140	GP 32, 0.75 - 4.5 Nm	303			●	●	●	●	●
RE 35, 90 W	140	GP 32, 0.75 - 6.0 Nm	305-309			●	●	●	●	●
RE 35, 90 W	140	GP 32, 4.0 - 8.0 Nm	310			●	●	●	●	●
RE 35, 90 W	140	GP 42, 3 - 15 Nm	314			●	●	●	●	●
RE 35, 90 W	140	GP 32 S	334-336			●	●	●	●	●
RE 40, 25 W	141					82.4	82.4	82.4	82.4	82.4
RE 40, 150 W	142					82.4	82.4	82.4	82.4	82.4
RE 40, 150 W	142	GP 42, 3 - 15 Nm	314			●	●	●	●	●
RE 40, 150 W	142	GP 52, 4 - 30 Nm	318			●	●	●	●	●
A-max 32	170/172					72.7	72.7	72.7	72.7	72.7
A-max 32	170/172	GP 32, 0.75 - 6.0 Nm	305-308			●	●	●	●	●
A-max 32	170/172	GS 38, 0.1 - 0.6 Nm	313			●	●	●	●	●
A-max 32	170/172	GP 32 S	334-336			●	●	●	●	●
EC-max 40, 70 W	228					73.9	73.9	73.9	73.9	73.9
EC-max 40, 70 W	228	GP 42, 3 - 15 Nm	315			●	●	●	●	●
EC-max 40, 120 W	229					103.9	103.9	103.9	103.9	103.9
EC-max 40, 120 W	229	GP 52, 4 - 30 Nm	319			●	●	●	●	●

## Technische Daten

Versorgungsspannung V <sub>cc</sub>	5 V ± 5%
Ausgangssignal	TTL kompatibel
Phasenverschiebung $\Phi$	90°e ± 45°e
Indexpulsbreite	90°e ± 45°e
Betriebstemperaturbereich	-25...+85°C
Trägheitsmoment der Impulsscheibe	≤ 1.7 gcm <sup>2</sup>
Strom pro Kanal	max. 5 mA

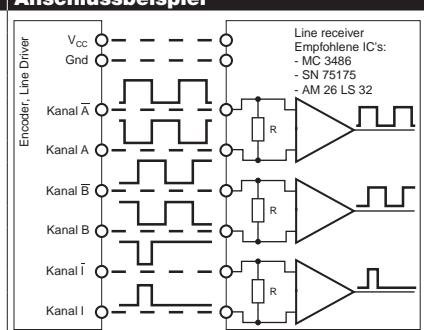
Das Indexsignal I ist synchronisiert mit Kanal A und B.

## Pinbelegung



Stecker nach DIN 41651/  
EN 60603-13  
Flachbandkabel AWG 28

## Anschlussbeispiel



Opt. Abschlusswiderstand R > 1 kΩ

#### C.10.4 Motorencontroller MR001-004.2



Name: **DC Dual Motor Driver 30V 4A V2**  
Code: **MR001-004.2**



The *DC Dual Motor Driver 30V 4A V2* allows to independently drive two DC motors, controlling both velocity and direction.

It is based on the famous integrated circuit L298, produced by STMicroelectronics; the L298 is an integrated monolithic circuit in a PowerSO20 package. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors.

The minimum supply voltage allowed is 7V, so you can use also two-cell LiPo batteries (7.4V) that grant small dimensions and low weight characteristics. The maximum supply voltage supported by this board is 30V.

To enable or disable each channel independently of the input signals there are provided two enable inputs (*E1*, *E2*) positioned on the front of the board; motor velocity regulation is obtained applying to these pins a PWM signal with a 20KHz max. frequency.

This board also provides direction LED indicators for both channels; this is very useful during setup stage to verify the firmware behaviour (also without applying a real motor to the output).

## **INSTRUCTIONS**

Two via terminal blocks are the two outputs for motors (M1 and M2). The 3 pins strip connectors next to the power terminal block are used to control the 2 channels of this *DC Dual Motor Driver 30V 4A V2*. Each of them has signals as reported on table 1.

Channel 1			Channel 2	
Name	Function		Name	Function
1A	Input A of ch.1 (TTL input)		2A	Input A of ch.2 (TTL input)
1B	Input B of ch.1 (TTL input)		2B	Input B of ch.2 (TTL input)
E1	Enable ch.1 (TTL input)		E2	Enable ch.2 (TTL input)

**Tab.1 - Connections**

To understand the meaning of these signals and their use you can read the following table (Tab.2), where all conditions are reported. Note that there are reported only the conditions for channel 1 because conditions for channel 2 are just alike them.

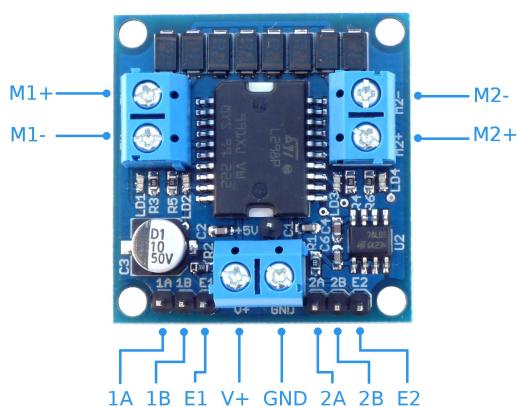
Inputs			M1+ and M1- output
E1	1A	1B	
1	1	1	HIGH state for both output (motor stopped)
1	0	0	LOW state for both output (motor stopped)
1	1	0	Current flows from M1+ to M1- (direction 1)
1	0	1	Current flows from M1- to M1+ (direction 2)
0	X	X	High impedance (motor is in free running)

**Tab.2 - Conditions**

About the *E1* and *E2* signals, they have pull-up resistors so in some applications you don't need to drive them and you need just 2 signals to control each motor.

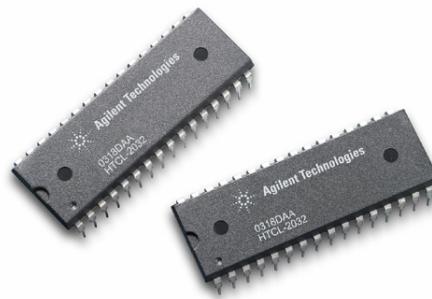
## SPECIFICATIONS

Supply voltage	7 - 30V
Supply current (logic)	24mA typ. (36mA max.)
Output current	4A (2A for each channel)
Data I/O voltage	TTL standard
Dimensions	37x36x13mm (connectors included)
Weight	12.5g / 0.44oz
Operating temperature	-25 - 130°C



**C.10.5 Counter Interface IC HCTL 2022**

## Data Sheet



### Description

The HCTL-20XX-XX is CMOS ICs that perform the quadrature decoder, counter, and bus interface function. The HCTL-20XX-XX is designed to improve system performance in digital closed loop motion control systems and digital data input systems. It does this by shifting time intensive quadrature decoder functions to a cost effective hardware solution. The HCTL-20XX-XX consists of a quadrature decoder logic, a binary up/down state counter, and an 8-bit bus interface. The use of Schmitt-triggered CMOS inputs and input noise filters allows reliable operation in noisy environments. The HCTL-20XX-XX contains 32-bit counter and provides LSTTL compatible tri-state output buffers. Operation is specified for a temperature range from -40 to +100°C at clock frequencies up to 33MHz.

The HCTL-2032 and HCTL-2032-SC have dual-axis capability and index channel support. Both devices can be programmed as 4x/2x/1x count mode. The HCTL-2032 and HCTL2032-SC also provides quadrature decoder output signals and cascade signals for use with many standard computer ICs.

The HCTL-2022 has most of the HCTL-2032 features, but it can only supports single axis and fixed at 4x count mode. The HCTL-2022 doesn't provide decoder output and cascade signals.

### Features

- Interfaces Encoder to Microprocessor
- 33 MHz Clock Operation
- Programmable Count Modes (1x, 2x or 4x)
- Single or Dual Axis Support
- Index Channel Support
- High Noise Immunity:
- Schmitt Trigger Inputs and Digital Noise Filter
- 32-Bit Binary Up/Down Counter
- Latched Outputs
- 8-Bit Tristate Interface
- 8, 16, 24, or 32-Bit Operating Modes
- Quadrature Decoder Output Signals, Up/Down and Count
- Cascade Output Signals, Up/Down and Count
- Substantially Reduced System Software
- 5V Operation ( $V_{DD} - V_{SS}$ )
- TTL/CMOS Compatible I/O
- Operating Temperature: -40°C to 100°C
- 32-Pin PDIP, 32-Pin SOIC, 20-Pin PDIP

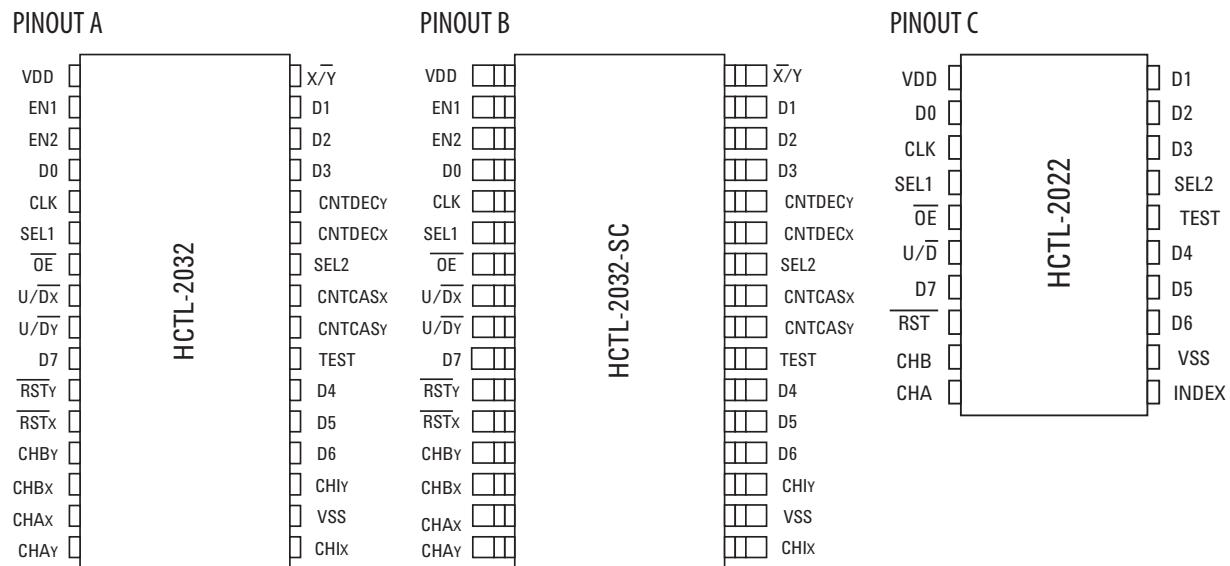
### Applications

- Interface Quadrature Incremental Encoders to Microprocessors
- Interface Digital Potentiometers to Digital Data Input Buses

*ESD WARNING: Standard CMOS handling precautions should be observed with the HCTL-2032 family ICs.*

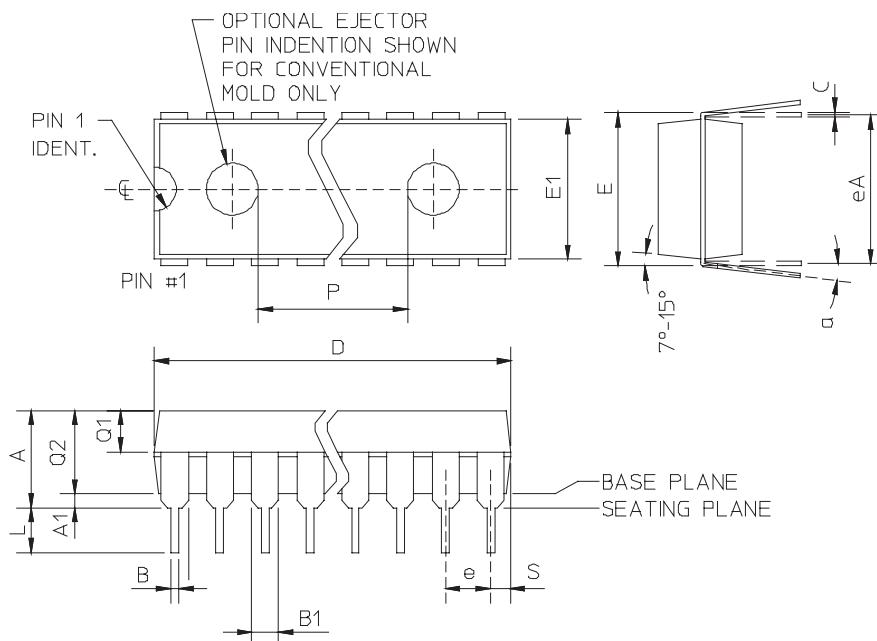
## Devices

Part Number	Description	Package Drawing
HCTL-2032	32-bit counter, dual axis, decoder and cascade outputs, index channel support, programmable count modes, and 33 Mhz clock operation.	A
HCTL-2032-SC	All features of HCTL-2032.	B
HCTL-2022	Most of the HCTL-2032 features. The device supports single axis, and no decoder output and cascade signals. The programmable count mode is set to 4x internally.	C



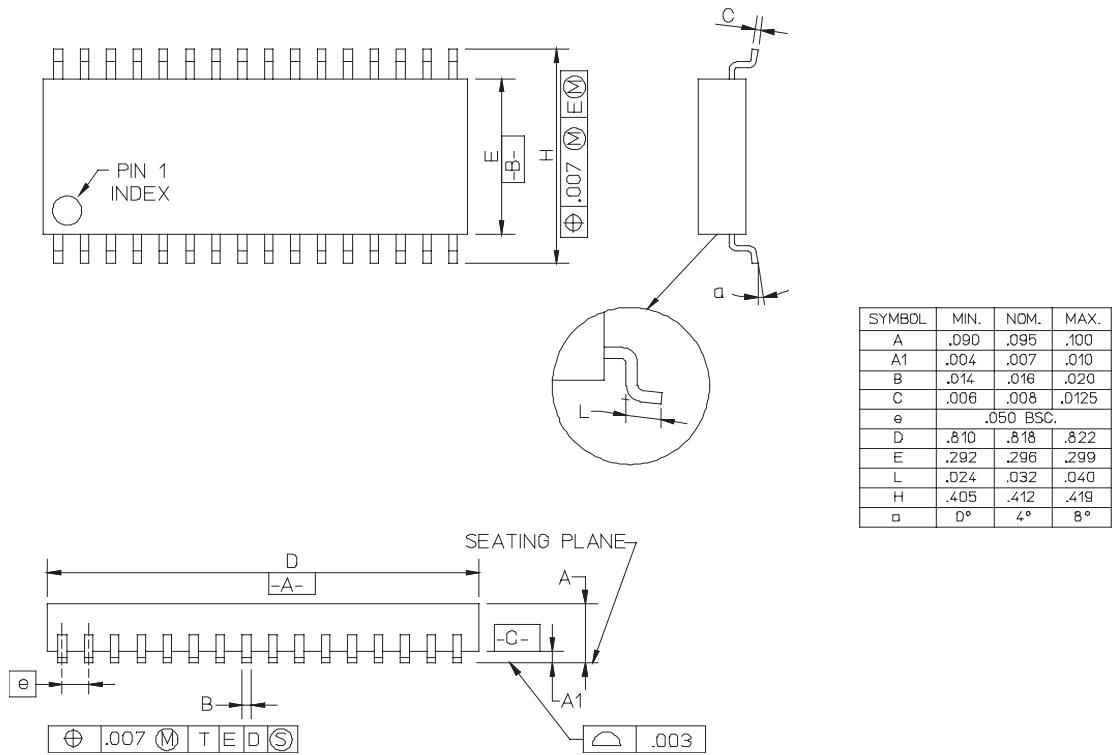
## Package Dimensions (dimensions in inches)

### 1) HCTL - 2032

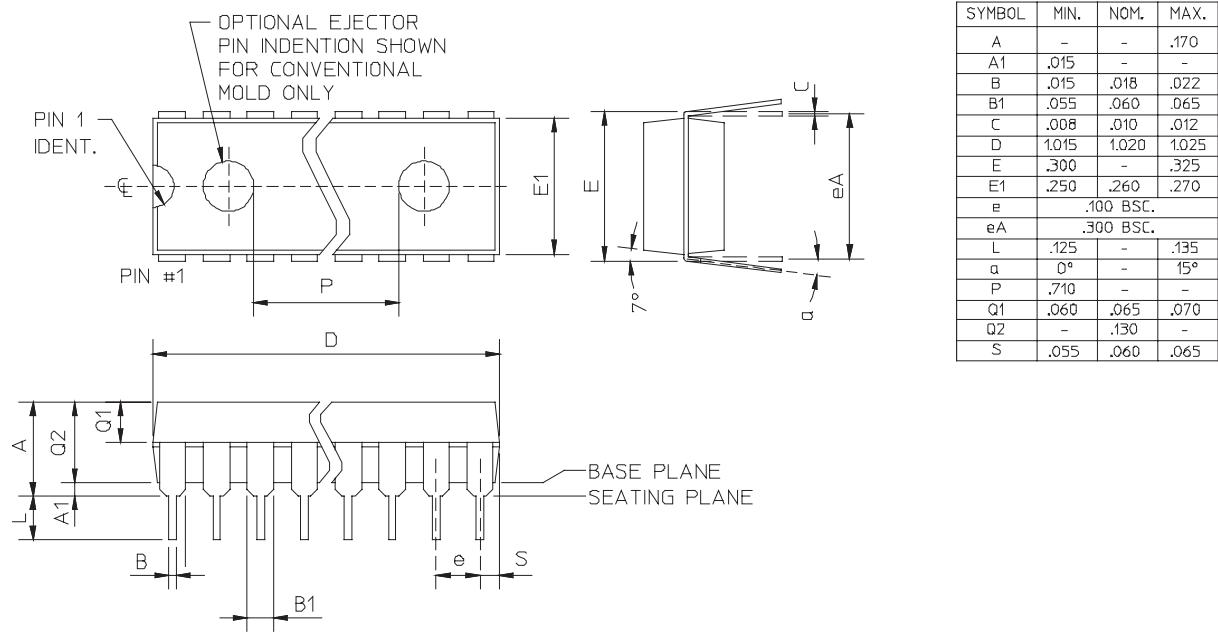


SYMBOL	MIN.	NOM.	MAX.
A	-	-	-
A1	-	-	-
B	.016	.018	.020
B1	.045	.050	.055
C	-	.010	-
D	1.640	1.650	1.660
E	.590	.610	.630
E1	.546	.550	.554
e	.100 TYP		
eA	-		
L	.100	-	-
a	-	-	-
Q1	.066	.070	.074
Q2	-	-	-
S	-	-	-

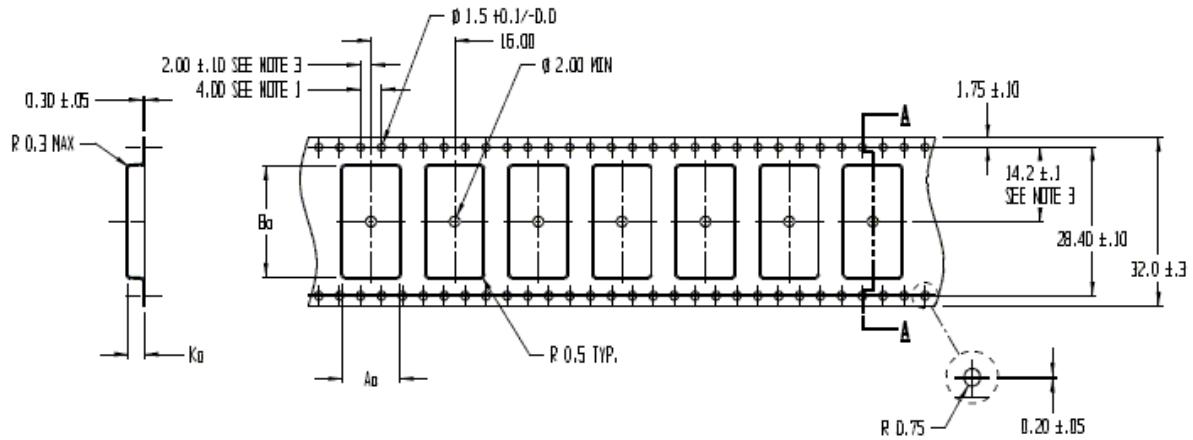
## 2) HCTL - 2032 - SC



## 3) HCTL - 2022



#### **4) HCTL-2032 –SCT (Tape and Reel Version of HCTL-2032-SC)**



**SECTION A - A**

**A<sub>D</sub> = 10.90**

Bu = 21,20

$$K_0 = 3.10$$

## Notes:

1. 10 Sprocket hole pitch cumulative tolerance 0.2
  2. Camber in compliance with EIA 481
  3. Pocket position relative to sprocket hole measured as true position of pocket, not pocket hole
  4. All dimensions in mm

## Operating Characteristics

**Table 1. Absolute Maximum Ratings**

(All voltages below are referenced to V<sub>SS</sub>)

Parameter	Symbol	Limits	Units
DC Supply Voltage	V <sub>DD</sub>	-0.3 to +6.0	V
Input Voltage	V <sub>IN</sub>	-0.3 to (V <sub>DD</sub> + 0.3)	V
Storage Temperature	T <sub>S</sub>	-55 to +150	°C
Operating Temperature [1]	T <sub>A</sub>	-40 to +100	°C

**Table 2. Recommended Operating Conditions**

Parameter	Symbol	Limits	Units
DC Supply Voltage	V <sub>DD</sub>	4.5 to 5.5	V
Ambient Temperature [1]	T <sub>A</sub>	-40 to +100	°C

**Table 3. DC Characteristics V<sub>DD</sub> = 5V ± 5%; T<sub>A</sub> = -40 to 100°C**

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V <sub>IL</sub> [2]	Low-Level Input Voltage			1.5		V
V <sub>IH</sub> [2]	High-Level Input Voltage		3.5			V
V <sub>T+</sub>	Schmitt-Trigger Positive-Going Threshold			3.5	4.0	V
V <sub>T-</sub>	Schmitt-Trigger Negative-Going Threshold		1.0	1.5		V
V <sub>H</sub>	Schmitt-Trigger Hysteresis		1.0	2.0		V
I <sub>IN</sub>	Input Current	V <sub>IN</sub> =V <sub>SS</sub> or V <sub>DD</sub>	-10	1	+10	µA
V <sub>OH</sub> [2]	High-Level Output Voltage	I <sub>OH</sub> = -3.75 mA	2.4	4.5		V
V <sub>OL</sub> [2]	Low-Level Output Voltage	I <sub>OL</sub> = +3.75mA	0.2	0.4		V
I <sub>OZ</sub>	High-Z Output Leakage Current	V <sub>O</sub> =V <sub>SS</sub> or V <sub>DD</sub>	-10	1	+10	µA
I <sub>DD</sub>	Quiescent Supply Current	V <sub>IN</sub> =V <sub>SS</sub> or V <sub>DD</sub>		1	10	µA
C <sub>IN</sub> [3]	Input Capacitance	Any Input		5		pF
C <sub>OUT</sub> [3]	Output Capacitance	Any Output		5		pF

### Notes

- Free Air
- In general, for any V<sub>DD</sub> between the allowable limits (+4.5V to +5.5V), V<sub>IL</sub> = 0.3V<sub>DD</sub> and V<sub>IH</sub> = 0.7V<sub>DD</sub>; typical values are V<sub>OH</sub> = V<sub>DD</sub> - 0.5V and V<sub>OL</sub> = V<sub>SS</sub> + 0.2V
- Including package capacitance

## Functional Pin Description

**Table 4. Functional Pin Descriptions**

Symbol	Pin			Description																																				
	HCTL 2032/ 2032-SC	HCTL 2022																																						
V <sub>DD</sub>	1	1		Power Supply																																				
V <sub>SS</sub>	18	12		Ground																																				
CLK	5	3		CLK is a Schmitt-trigger input for the external clock signal.																																				
CHA <sub>X</sub>	15	10		CHA <sub>X</sub> , CHA <sub>Y</sub> , CHB <sub>X</sub> , and CHB <sub>Y</sub> are Schmitt-trigger inputs that accept the outputs from a quadrature-encoded source, such as incremental optical shaft encoder. Two channels, A and B, nominally 90 degrees out of phase, are required. CHA <sub>X</sub> and CHB <sub>X</sub> are the 1st axis and CHA <sub>Y</sub> and CHB <sub>Y</sub> are the 2nd axis.																																				
CHA <sub>Y</sub>	16	NC																																						
CHB <sub>X</sub>	14	9																																						
CHB <sub>Y</sub>	13	NC																																						
CHI <sub>X</sub>	17	11		CHI <sub>X</sub> and CHI <sub>Y</sub> are Schmitt-trigger inputs that accept the outputs of Index channel from an incremental optical shaft encoder.																																				
CHI <sub>Y</sub>	19	NC																																						
RSTN <sub>X</sub>	12	8		This active low Schmitt-trigger input clears the internal position counter and the position latch. It also resets the inhibit logic. RST <sub>X</sub> / and RST <sub>Y</sub> / are asynchronous with respect to any other input signals. RST <sub>X</sub> / is to reset the 1st axis counter and RST <sub>Y</sub> / is to reset the 2nd axis counter.																																				
RSTN <sub>Y</sub>	11	NC																																						
OEN	7	5		This CMOS active low input enables the tri-state output buffers. The OE/, SEL1, and SEL2 inputs are sampled by the internal inhibit logic on the falling edge of the clock to control the loading of the internal position data latch.																																				
SEL <sub>1</sub>	6	4																																						
SEL <sub>2</sub>	26	17		These CMOS inputs directly controls which data byte from the position latch is enabled into the 8-bit tri-state output buffer. As in OE/ above, SEL <sub>1</sub> and SEL <sub>2</sub> also control the internal inhibit logic.																																				
				<table border="1"> <thead> <tr> <th colspan="6">BYTE SELECTED</th> </tr> <tr> <th>SEL1</th> <th>SEL2</th> <th>MSB</th> <th>2ND</th> <th>3RD</th> <th>LSB</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> <td>D4</td> <td></td> <td></td> <td></td> </tr> <tr> <td>1</td> <td>1</td> <td></td> <td>D3</td> <td></td> <td></td> </tr> <tr> <td>0</td> <td>0</td> <td></td> <td></td> <td>D2</td> <td></td> </tr> <tr> <td>1</td> <td>0</td> <td></td> <td></td> <td></td> <td>D1</td> </tr> </tbody> </table>	BYTE SELECTED						SEL1	SEL2	MSB	2ND	3RD	LSB	0	1	D4				1	1		D3			0	0			D2		1	0				D1
BYTE SELECTED																																								
SEL1	SEL2	MSB	2ND	3RD	LSB																																			
0	1	D4																																						
1	1		D3																																					
0	0			D2																																				
1	0				D1																																			
EN <sub>1</sub>	2	NC		These CMOS control pins are set to high or low to activate the selected count mode before the decoding begins.																																				
EN <sub>2</sub>	3	NC																																						
				<table border="1"> <thead> <tr> <th colspan="5">Count Modes</th> </tr> <tr> <th>EN1</th> <th>EN2</th> <th>4x</th> <th>2x</th> <th>1x</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td></td> <td>Illegal Mode</td> <td></td> </tr> <tr> <td>1</td> <td>0</td> <td>On</td> <td></td> <td></td> </tr> <tr> <td>0</td> <td>1</td> <td></td> <td>On</td> <td></td> </tr> <tr> <td>1</td> <td>1</td> <td></td> <td></td> <td>On</td> </tr> </tbody> </table>	Count Modes					EN1	EN2	4x	2x	1x	0	0		Illegal Mode		1	0	On			0	1		On		1	1			On						
Count Modes																																								
EN1	EN2	4x	2x	1x																																				
0	0		Illegal Mode																																					
1	0	On																																						
0	1		On																																					
1	1			On																																				
X/Y	32	NC		Select the 1 <sup>st</sup> or 2 <sup>nd</sup> axis data to be read. Low bit enables the 1 <sup>st</sup> axis data, while high bit enables the 2 <sup>nd</sup> axis data.																																				
CNTDEC <sub>X</sub>	27	NC		A pulse is presented on this LSTTL-compatible output when the quadrature decoder (4x/2x/1x) has detected a state transition. CNTDECX is for 1 <sup>st</sup> axis and CNTDECY is for 2 <sup>nd</sup> axis.																																				
CNTDEC <sub>Y</sub>	28	NC																																						
U/Dx	8	6		This LSTTL-compatible output allows the user to determine whether the IC is counting up or down and is intended to be used with the CNTDEC and CNTCAS outputs. The proper signal U (high level) or D/ (low level) will be present before the rising edge of the CNTDEC and CNTCAS outputs.																																				
U/Dy	9	NC																																						

CNTCAS <sub>X</sub>	25	NC	A pulse is presented on this LSTTL-compatible output when the HCTL-2032 / 2032-SC internal counter overflows or underflows. The rising edge on this waveform may be used to trigger an external counter.
TEST	23	16	This pin is used for internal testing. Tied it to ground or leave it floating for normal operation.
D0	4	2	These LSTTL-compatible tri-state outputs form an 8-bit output ports through which
D1	31	20	the contents of the 32-bit position latch may be read in 4 sequential bytes. The MSB is read first followed by the rest of the bytes with the LSB is read last.
D2	30	19	
D3	29	18	
D4	22	15	
D5	21	14	
D6	20	13	
D7	10	7	

## Switching Characteristics

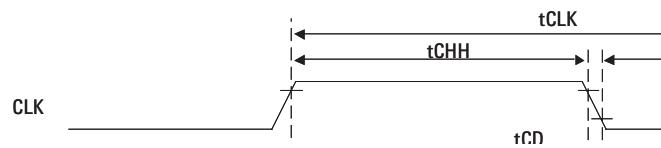
**Table 5. Switching Characteristics**

Max/Min specifications at  $V_{DD} = 5V \pm 5\%$ ;  $T_A = -40$  to  $100^\circ C$ ,  $C_L = 40 \text{ pF}$

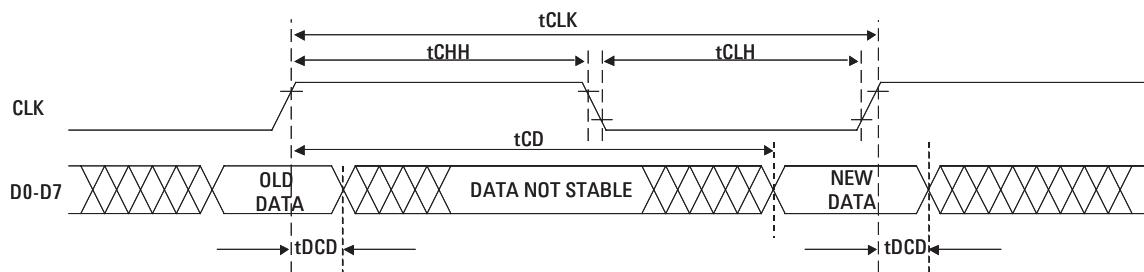
Symbol	Description	Min.	Max.	Units
1 t <sub>CLK</sub>	Clock Period		33	MHz
2 t <sub>CHH</sub>	Pulse width, clock high		1/f	ns
3 t <sub>CD</sub>	Delay time, rising edge of clock to valid, updated count information on D0-7		31	ns
4 t <sub>ODE</sub>	Delay time, OEN fall to valid data		29	ns
5 t <sub>OZ</sub>	Delay time, OEN rise to Hi-Z state on D0-7		29	ns
6 t <sub>SDV</sub>	Delay time, SEL0~SEL1 valid to stable, selected data byte (delay to High Byte = delay to Low Byte)		29	ns
7 t <sub>XNYDV</sub>	Delay time, XNY valid to stable, selected data byte.		29	ns
8 t <sub>CLH</sub>	Pulse width, clock low		15	ns
9 t <sub>SS</sub>	Setup time, SEL1~SEL2 before clock fall		12	ns
10 t <sub>S</sub>	Setup time, OEN before clock fall		12	ns
11 t <sub>XNYS</sub>	Setup time, XNY before clock fall		12	ns
12 t <sub>SH</sub>	Hold time, SEL1~SEL2 after clock fall		0	ns
13 t <sub>OH</sub>	Hold time, OEN after clock fall		0	ns
14 t <sub>XNYH</sub>	Hold time, XNY after clock fall		0	ns
15 t <sub>RST</sub>	Pulse width, RSTNX~RSTNY low		10	ns
16 t <sub>DCD</sub>	Hold time, last position count stable on D0-7 after clock rise		2	ns
17 t <sub>DSD</sub>	Hold time, last data byte stable after next SEL state change		2	ns
18 t <sub>DOD</sub>	Hold time, data byte stable after OEN rise		2	ns
19 t <sub>DXNYD</sub>	Hold time, data byte stable after XNY change		2	ns
20 t <sub>UDDX</sub>	Delay time, U/DNX valid after clock rise	4	29	ns
21 t <sub>UDDY</sub>	Delay time, U/DNY valid after clock rise	4	29	ns
22 t <sub>CHXD</sub>	Delay time, CNTDECX or CNTCASX high after clock rise	4	31	ns
23 t <sub>CHYD</sub>	Delay time, CNTDECY or CNTCASY high after clock rise	4	31	ns
24 t <sub>CLXD</sub>	Delay time, CNTDECX or CNTCASX low after clock fall	4	31	ns
25 t <sub>CLYD</sub>	Delay time, CNTDECY or CNTCASY low after clock fall	4	31	ns
26 t <sub>UDXH</sub>	Hold time, U/DNX stable after clock rise	2		ns
27 t <sub>UDYH</sub>	Hold time, U/DNY stable after clock rise	2		ns
28 t <sub>UDCXS</sub>	Setup time, U/DNX valid before CNTDECX or CNTCASX rise		Note 1	ns
29 t <sub>UDCYS</sub>	Setup time, U/DNY valid before CNTDECY or CNTCASY rise		Note 1	ns
30 t <sub>UDCXH</sub>	Hold time, U/DNX stable after CNTDECX or CNTCASX rise		Note 2	ns
31 t <sub>UDCYH</sub>	Hold time, U/DNY stable after CNTDECY or CNTCASY rise		Note 2	ns

### Notes

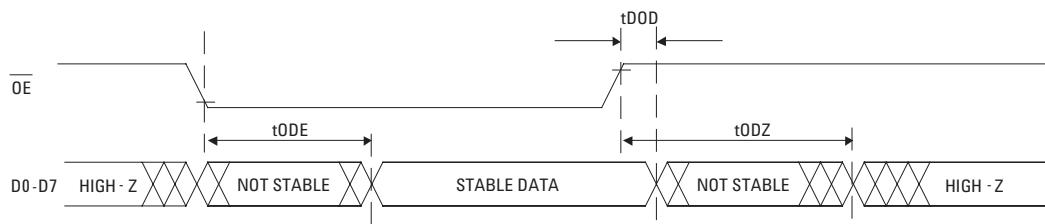
1. tclk - max delay (item 20/21) + min delay (item 22/23)
2. tclk - max delay (item 22/23) + min delay (item 20/21)



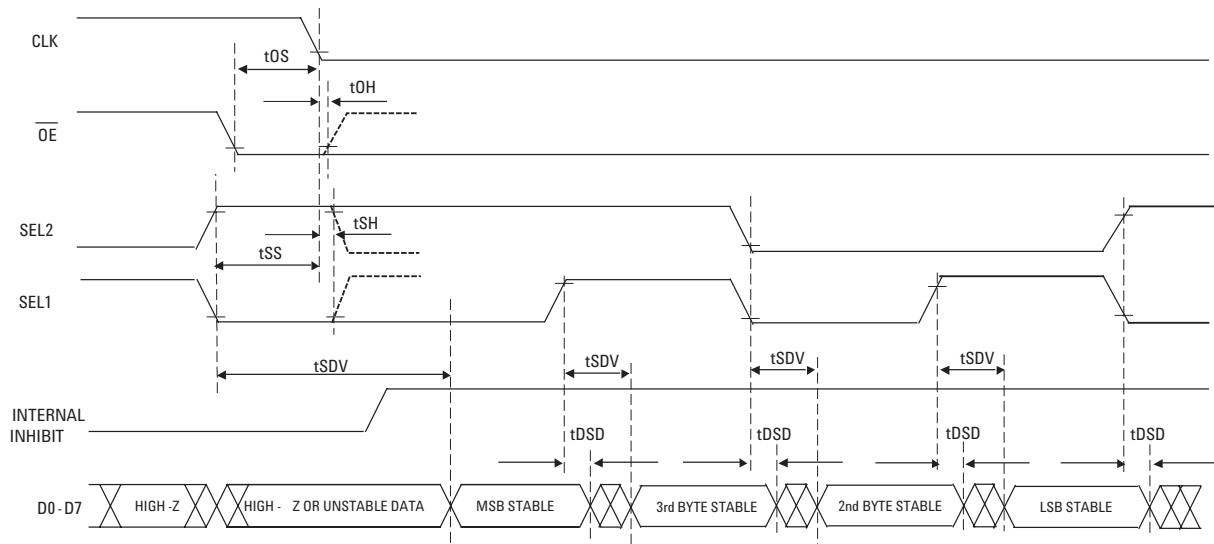
**Figure 1. Reset Waveform**



**Figure 2. Waveforms for Positive Clock Edge Related Delays**



**Figure 3. Tri-State Output Timing**



**Figure 4. Bus Control Timing**

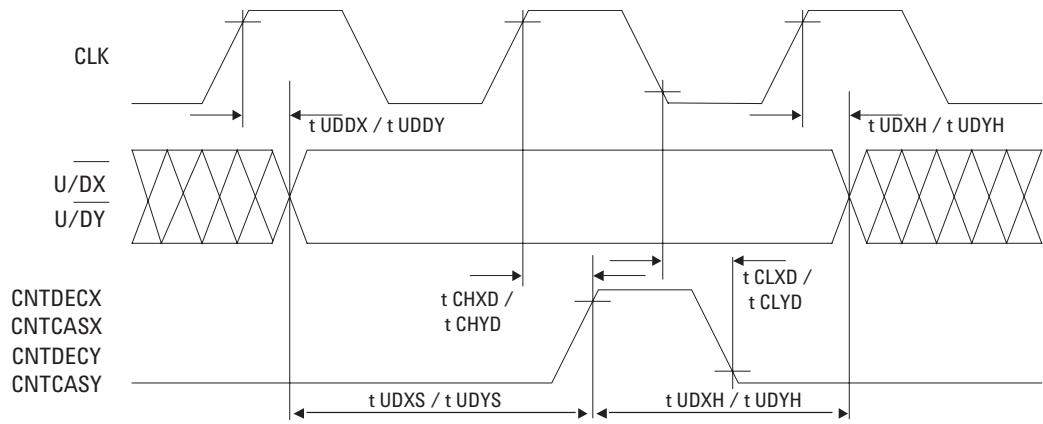


Figure 5. Decoder, Cascade Output Timing

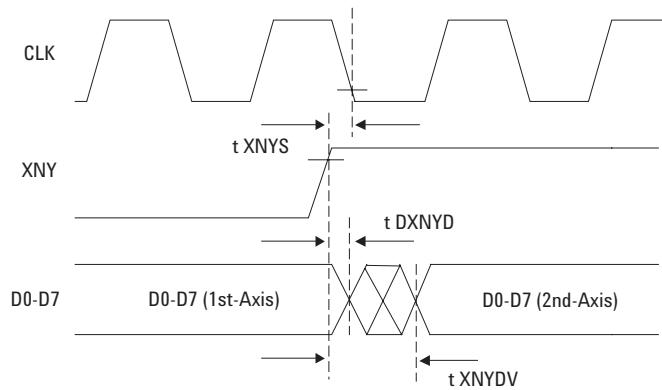


Figure 6. Output Data from 1<sup>st</sup>-axis and 2<sup>nd</sup>-axis

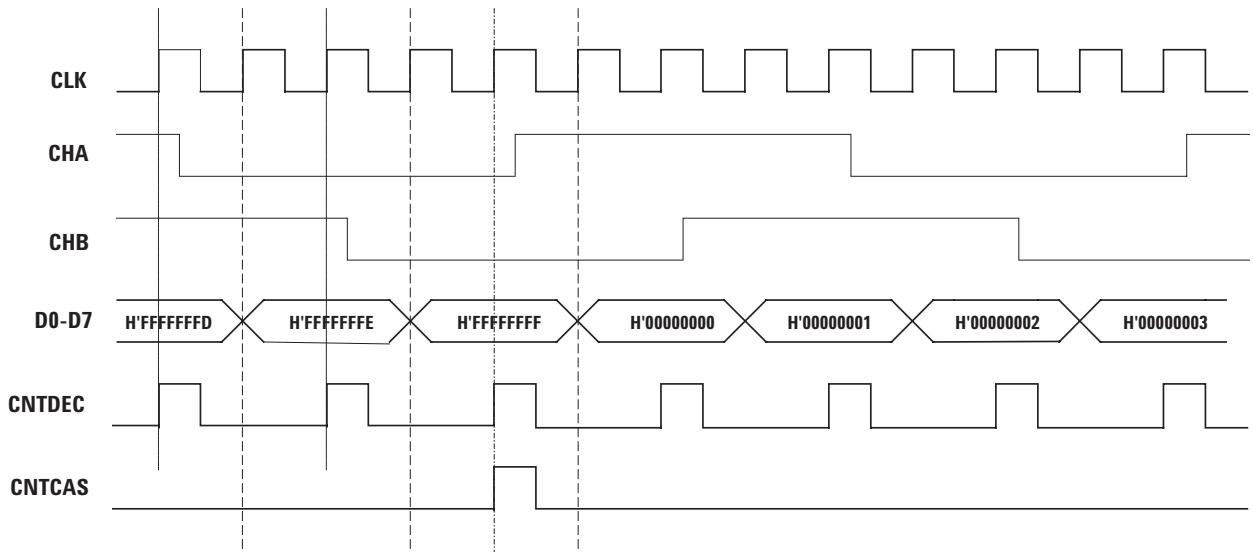


Figure 7. Quadrature decoder for 1<sup>st</sup>-axis/2<sup>nd</sup>-axis (4x count mode)

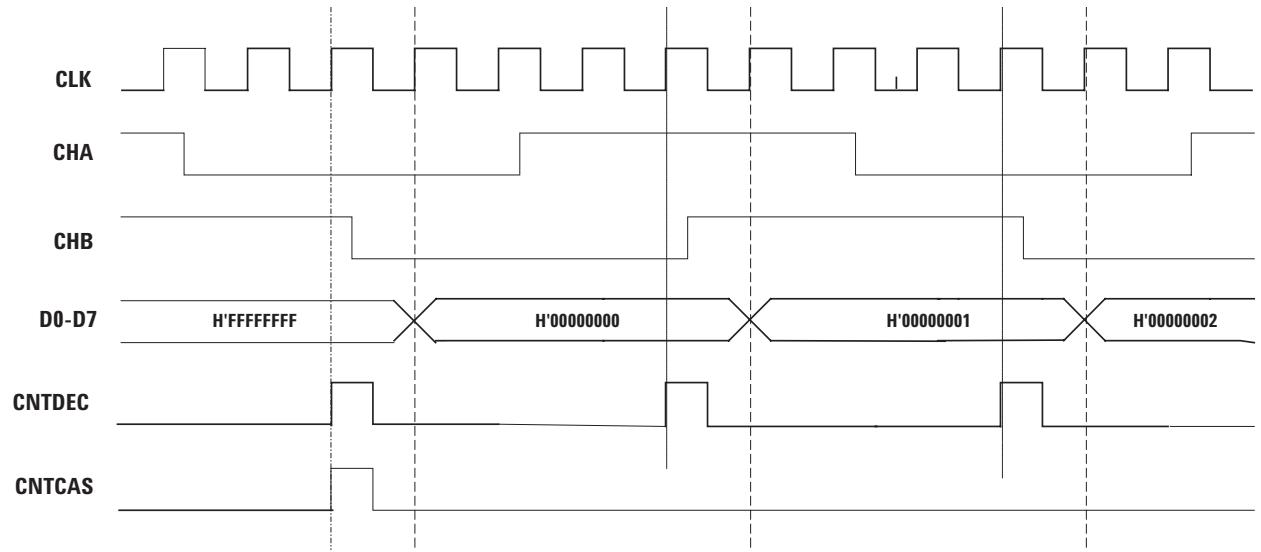


Figure 8. Quadrature decoder for 1<sup>st</sup>-axis/ 2<sup>nd</sup>-axis (2x count mode)

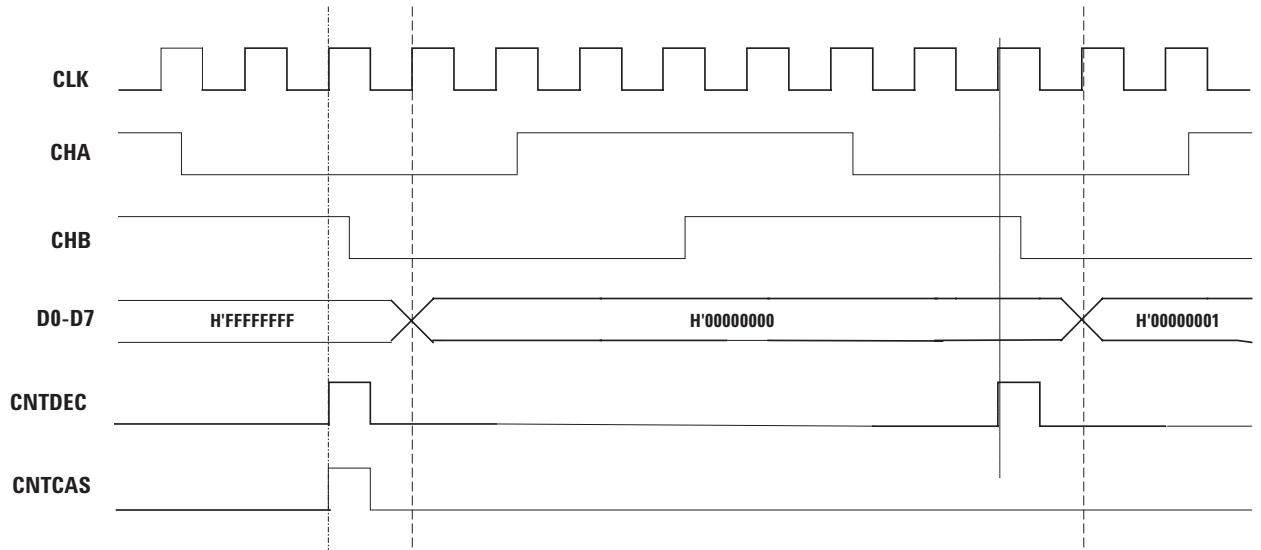
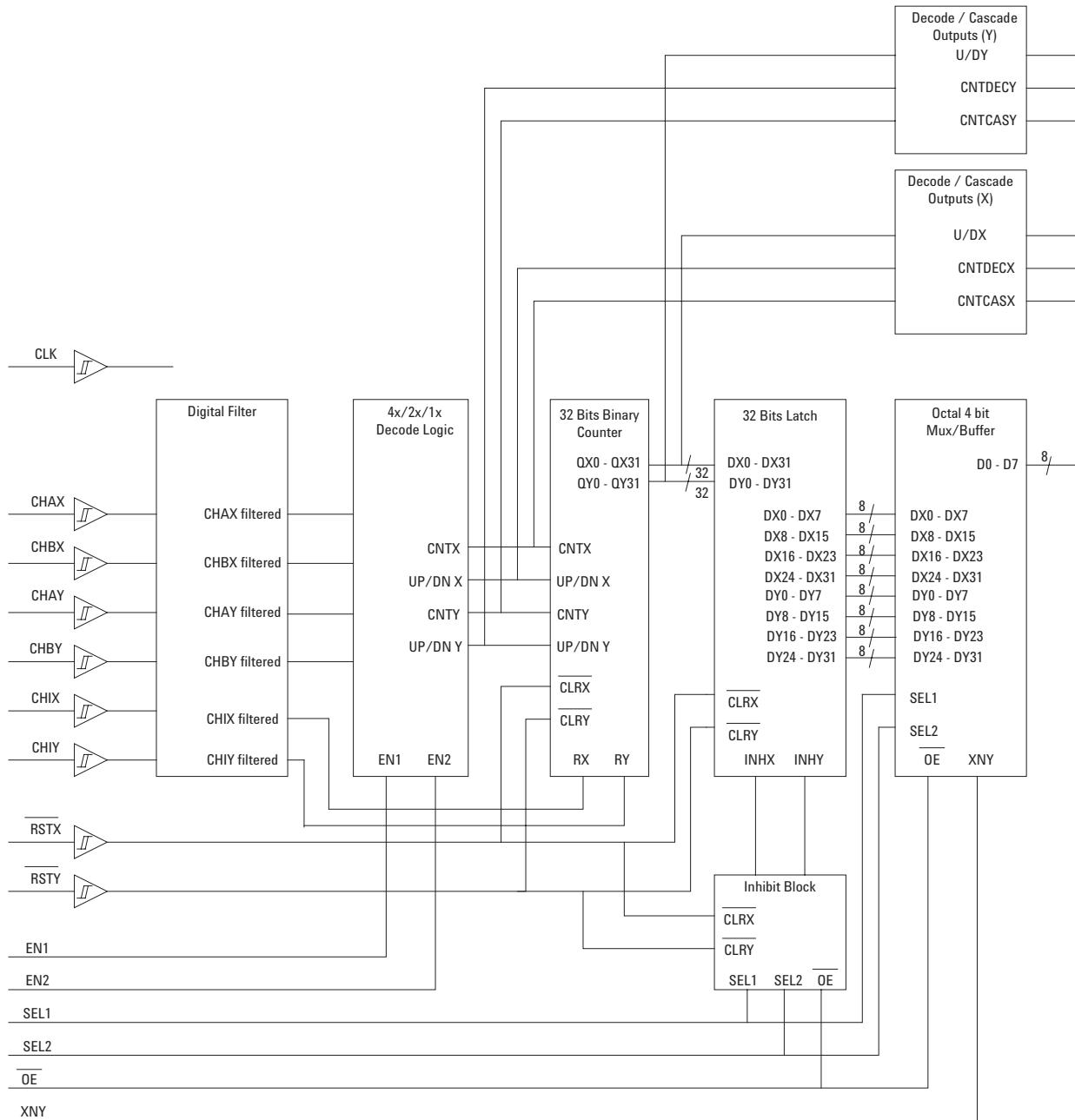


Figure 9. Quadrature decoder for 1<sup>st</sup>-axis/ 2<sup>nd</sup>-axis (1x count mode)

## Operation

A block diagram of the HCTL-20XX-XX family is shown in Figure 10. The operation of each major function is described in the following sections.



**Figure 10. Simplified Logic Diagram**

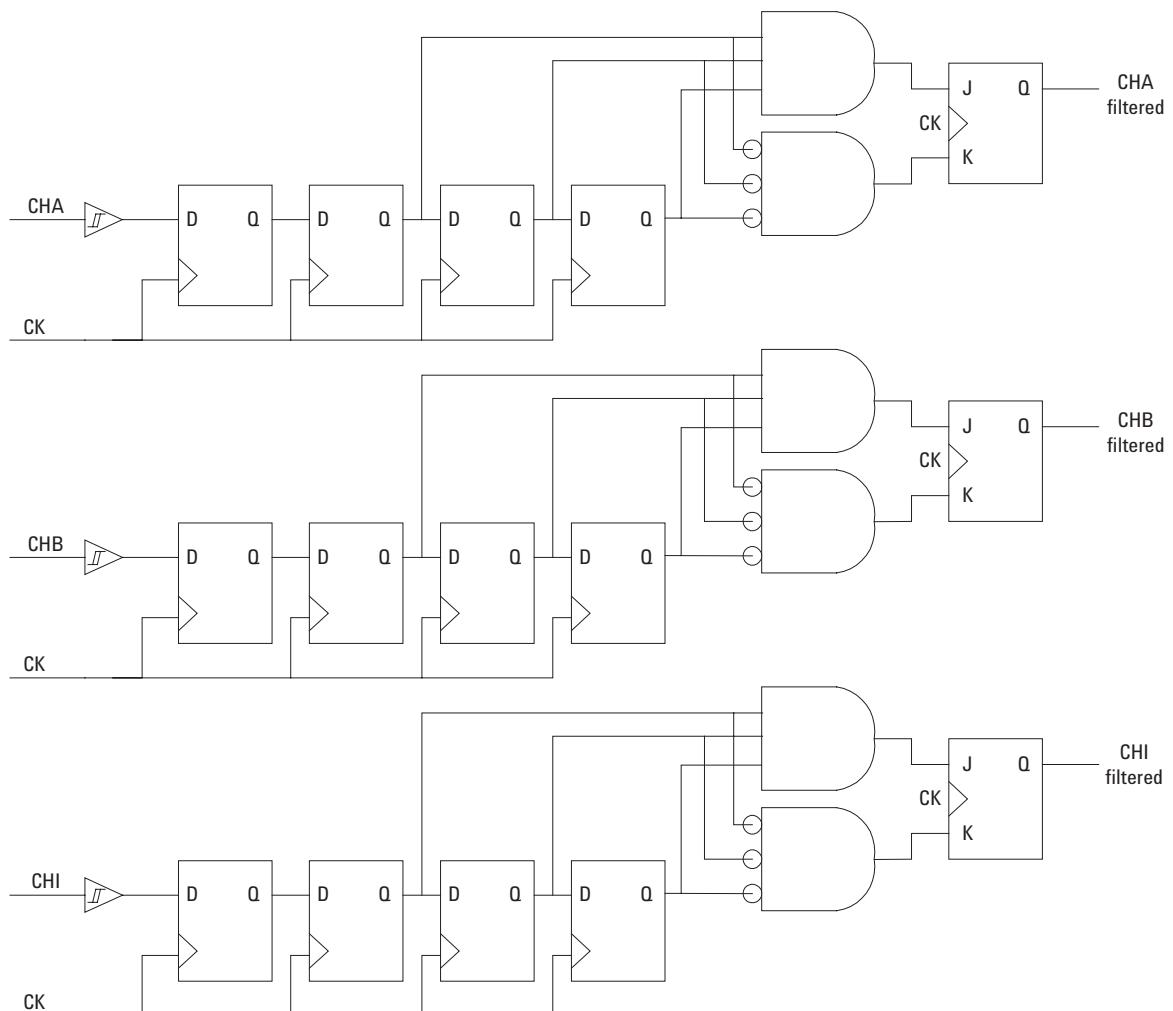
## Digital Noise Filter

The digital noise filter section is responsible for rejecting noise on the incoming quadrature signals. The input section uses two techniques to implement improved noise rejection. Schmitt-trigger inputs and a three-clock-cycle delay filter combine to reject low level noise and large, short duration noise spikes that typically occur in motor system applications. Both common mode and differential mode noise are rejected. The user benefits from these techniques by improved integrity of the data in the counter. False counts triggered by noise are avoided.

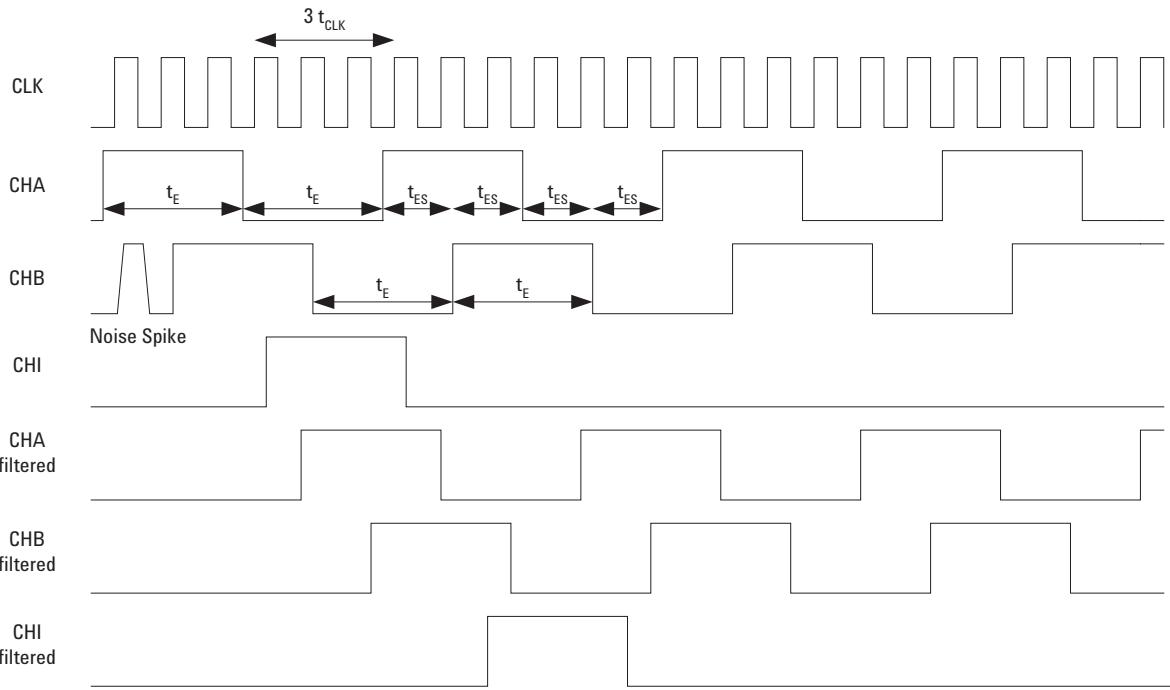
Figure 11 shows the simplified schematic of the input section. The signals are first passed through a Schmitt-trigger buffer to address the problem of input signals with

slow rise times and low-level noise (approximately  $< 1V$ ). The cleaned up signals are then passed to a four-bit delay filter. The signals on each channel are sampled on rising clock edges. A time history of the signals is stored in the four-bit shift register. Any change on the input is tested for a stable level being present for three consecutive rising clock edges. Therefore, the filtered output waveforms can change only after an input level has the same value for three consecutive rising clock edges.

Refer to Figure 12, which shows the timing diagram. The result of this circuitry is that short noise spikes between rising clock edges are ignored and pulses shorter than two clock periods are rejected.



**Figure 11. Simplified Digital Noise Filter Logic**



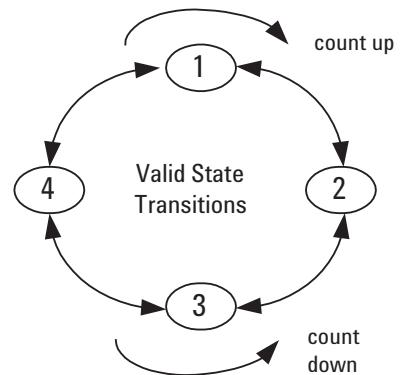
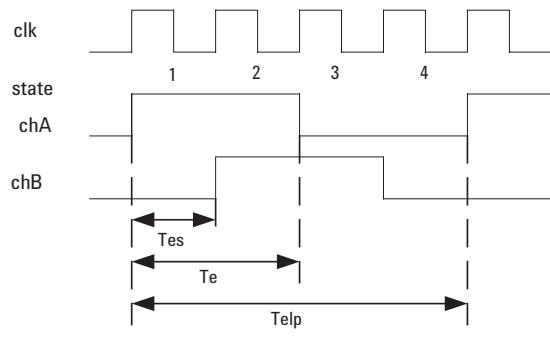
**Figure 12. Signal Propagation through Digital Noise Filter**

### Quadrature Decoder

The quadrature decoder decodes the incoming filtered signals into count information. This circuitry multiplies the resolution of the input signals by a factor of one, two or four (1X, 2X, 4X decoding) depending on the resolution mode. When using an encoder for motion sensing, the user benefits from the selectable resolution by being able to provide better system control.

The quadrature decoder samples the outputs of the CHA and CHB filters. Based on the past binary state of the two signals and the present state, it outputs a count signal and a direction signal to the integral position counter.

Figure 13 shows the quadrature states of Channel A and Channel B signals. The?4x decoder mode will output a count signal for every state transition (count up and count down). Figure 14 shows the valid state transitions for 2x and 1x decoder modes. The 2x/1x decoder will output a count signal at respective state transition, depending on the counting direction. Channel A leading channel B results in counting up. Channel B leading channel A results in counting down. Illegal state transitions, caused by faulty encoders or noise severe enough to pass through the filter, will produce an erroneous count.



			<b>4X Decoder (Count Up &amp; Count Down)</b>
<b>CHA</b>	<b>CHB</b>	<b>STATE</b>	
1	0	1	Pulse
1	1	2	Pulse
0	1	3	Pulse
0	0	4	Pulse

**Figure 13. 4x Decoder Mode**

<b>CHA</b>	<b>CHB</b>	<b>STATE</b>	<b>2x Count Up</b>	<b>2x Count Down</b>	<b>1x Count Up</b>	<b>1x Count Down</b>
1	0	1	Pulse	-	Pulse	-
1	1	2	-	Pulse	-	Pulse
0	1	3	Pulse	-	-	-
0	0	4	-	Pulse	-	-

**Figure 14. 2x and 1x Decoder Modes**

## Design Considerations

The designer should be aware that the operation of the digital filter places a timing constraint on the relationship between incoming quadrature signals and the external clock. Figure 12 shows the timing waveform with an incremental encoder input. Since an input has to be stable for three rising clock edges, the encoder pulse width ( $t_E$  - low or high) has to be greater than three clock periods ( $3t_{CLK}$ ). This guarantees that the asynchronous input will be stable during three consecutive rising clock edges. A realistic design also has to take into account finite rise time of the waveforms, asymmetry of the waveforms, and noise. In the presence of large amounts of noise,  $t_E$  should be much greater than  $3t_{CLK}$  to allow for the interruption of the consecutive level sampling by the three-bit delay filter. It should be noted that a change on the inputs that is qualified by the filter will internally propagate in a maximum of seven clock periods.

The quadrature decoder circuitry imposes a second timing constraint between the external clock and the input signals. There must be at least one clock period between consecutive quadrature states. As shown in Figure 13, a quadrature state is defined by consecutive edges on both channels. Therefore,  $t_{ES}$  (encoder state period)  $> t_{CLK}$ . The designer must account for deviations from the nominal 90 degree phasing of input signals to guarantee that  $t_{ES} > t_{CLK}$ .

## Position Counter

This section consists of a 32-bit (HCTL-20XX-XX) binary up/down counter which counts on rising clock edges as explained in the Quadrature Decoder Section. All 32 bits of data are passed to the position data latch. The system can use this count data in several ways:

- A. System total range is 32 bits, so the count represents "absolute" position.
- B. The system is cyclic with 32 bits of count per cycle. RST/ is used to reset the counter every cycle and the system uses the data to interpolate within the cycle.
- C. System count is  $>8, 16, 24$ , or 32 bits, so the count data is used as a relative or incremental position input for a system software computation of absolute position. In this case counter rollover occurs. In order to prevent loss of position information, the processor must read the outputs of the IC before the count increments one-half of the maximum count capability. Two's-complement arithmetic is normally used to compute position from these periodic position updates.
- D. The system count is  $>32$  bits so the HCTL-2032 / 2032-SC can be cascaded with other standard counter ICs to give absolute position.

## Position Data Latch

The position data latch is a 32-bit latch which captures the position counter output data on each rising clock edge, except when its inputs are disabled by the inhibit logic section during four-byte read operations. The output data is passed to the bus interface section. When active, a signal from the inhibit logic section prevents new data from being captured by the latch, keeping the data stable while successive reads are made through the bus section. The latch is automatically re-enabled at the end of these reads. The latch is cleared to 0 asynchronously by the RST signal.

## Inhibit Logic

The Inhibit Logic Section samples the OE, SEL1 and SEL2 signals on the falling edge of the clock and, in response to certain conditions (see Figure 15), inhibits the position data latch. The RST signal asynchronously clears the inhibit logic, enabling the latch.

## Bus Interface

The bus interface section consists of a 32 to 8 line multiplexer and an 8-bit, three-state output buffer. The multiplexer allows independent access to the low and high bytes of the position data latch. The SEL1, SEL2 and OE signals determine which byte is output and whether or not the output bus is in the high-Z state. In the HCTL-20XX-XX, the data latch is 32 bit wide.

## Quadrature Decoder Output (HCTL-2032 / 2032-SC only)

The quadrature decoder output section consists of count and up/down outputs derived from the 4x/2x/1x decoder mode of the HCTL-2032 / 2032-SC. When the decoder has detected a count, a pulse, one-half clock cycle long, will be output on the CNT<sub>DCDR</sub> pin. This output will occur during the clock cycle in which the internal counter is updated. The U/D pin will be set to the proper voltage level one clock cycle before the rising edge of the CNT<sub>DCDR</sub> pulse, and held one clock cycle after the rising edge of the CNT<sub>DCDR</sub> pulse. These outputs are not affected by the inhibit logic.

## Cascade Output (HCTL-2032 / 2032-SC only)

The cascade output also consists of count and up/down outputs. When the HCTL-2032 / 2032-SC internal counter overflows or underflows, a pulse, one-half clock cycle long, will be output on the CNT<sub>CAS</sub> pin. This output will occur during the clock cycle in which the internal counter is updated. The U/D pin will be set to the proper voltage level one clock cycle before the rising edge of the CNT<sub>CAS</sub> pulse, and held one clock cycle after the rising edge of the CNT<sub>CAS</sub> pulse. These outputs are not affected by the inhibit logic.

Step	SEL1	SEL2	OE	CLK	Inhibit Signal	Action
1	L	H	L	—	1	Set inhibit; Read MSB
2	H	H	L	—	1	Read 2 <sup>nd</sup> Byte
3	L	L	L	—	1	Read 3 <sup>rd</sup> Byte
4	H	L	L	—	1	Read LSB
5	X	X	H	—	0	Completes inhibit logic reset

Figure 15. Four Bytes Read Sequence

## Cascade Considerations (HCTL-2032 / 2032-SC only)

The HCTL-2032 / 2032-SC's cascading system allows for position reads of more than four bytes. These reads can be accomplished by latching all the bytes and then reading the bytes sequentially over the 8-bit bus. It is assumed here that, externally, a counter followed by a latch is used to count any count that exceeds 32 bits. This configuration is compatible with the HCTL-2032 / 2032-SC internal counter/latch combination.

Consider the sequence of events for a read cycle that starts as the HCTL-2032 / 2032-SC's internal counter rolls over. On the rising clock edge, count data is updated in the internal counter, rolling it over. A count-cascade pulse (CNT<sub>CAS</sub>) will be generated with some delay after the rising clock edge ( $t_{CHD}$ ). There will be additional propagation delays through the external counters and registers. Meanwhile, with SEL and OE low to start the read, the internal latches are inhibited at the falling edge and do not update again till the inhibit is reset.

If the CNT<sub>CAS</sub> pulse now toggles the external counter and this count gets latched a major count error will occur. The count error is because the external latches get updated when the internal latch is inhibited.

Valid data can be ensured by latching the external counter data when the high byte read is started (SEL and OE low). This latched external byte corresponds to the count in the inhibited internal latch. The cascade pulse that occurs during the clock cycle when the read begins gets counted by the external counter and is not lost.

For example, suppose the HCTL-2032 / 2032-SC count is at FFFFFFFFh and an external counter is at F0h, with the count going up. A count occurring in the HCTL-2032 / 2032-SC will cause the counter to roll over and a cascade pulse will be generated. A read starting on this clock cycle will show FFFFFFFFh from the HCTL-2032 / 2032-SC. The external latch should read F0h, but if the host latches the count after the cascade signal propagates through, the external latch will read F1h.

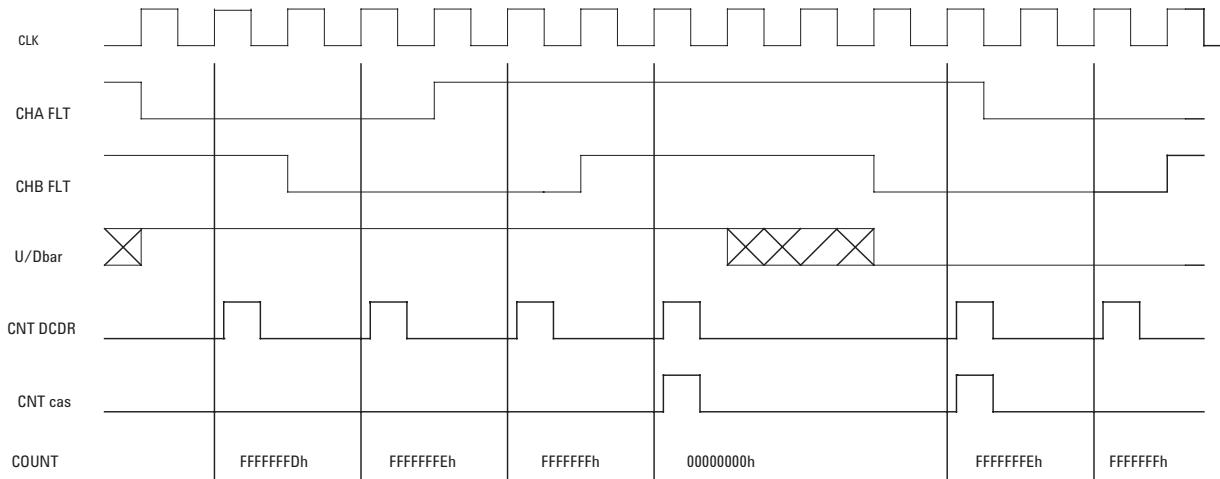


Figure 16. Decode and Cascade Output Diagram (4x)

## Interfacing the HCTL-2032 to an Atmel AVR 90S8535

The circuit shown in Figure 17 shows the connections between an HCTL-2032 and an Atmel AVR controller. Data lines D0-D7 are connected to the Atmel AVR bus port. The 8 MHz oscillators clock the Atmel AVR, whereas the external 33 MHz oscillators clock the HCTL-2032. Figure 18 illustrates the program that interfaces with an Atmel AVR 90S8535.

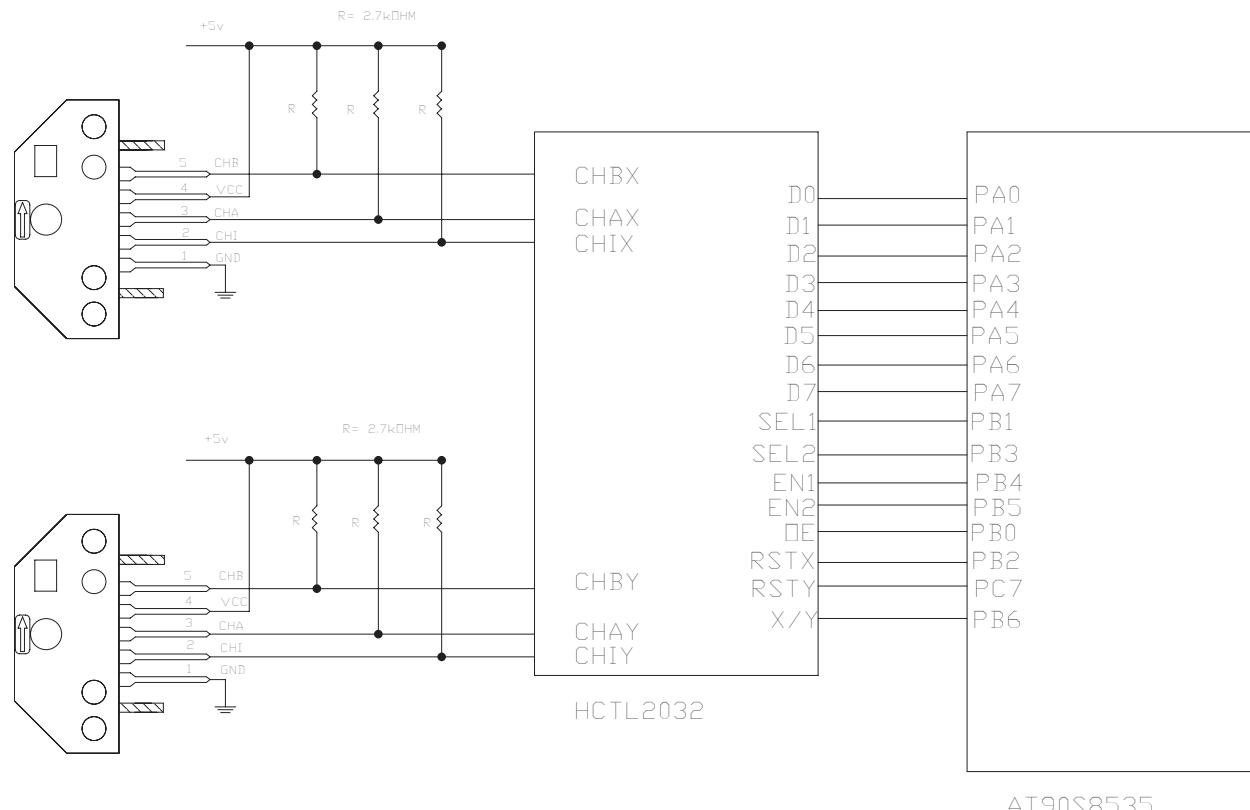


Figure 17. An HCTL-2032-to-Atmel AVR Interface

```

Set Portb.4          'EN1=1
Reset Portb.5        'EN2=0
Reset Portb.6        'Select X-axis

Result_new = 0
Result_old_x = 0
Result_old_y = 0

Do

    Set Portb.0          'Disable OE
    Waitms 25

    Reset Portb.1        'SEL1=0 (MSB)
    Set Portb.3          'SEL2=1 (MSB)
    Reset Portb.0        'Enable OE

    Gosub Get_hi         'Get MSB

    Set Portb.1          'SEL1=1 (2nd Byte)
    Set Portb.3          'SEL2=1 (2nd Byte)

    Gosub Get_2nd         'Get 2nd Byte
    Reset Portb.1        'SEL1=0 (3rd Byte)
    Reset Portb.3        'SEL2=0 (3rd Byte)

    Gosub Get_3rd         'Get 3rd Byte

    Set Portb.1          'SEL1=1 (LSB)
    Reset Portb.3        'SEL2=0 (LSB)
    Gosub Get_lo          'Get LSB
    Set Portb.0          'Disable OE
    Waitms 25
    Mult = 1
    Temp = Result_lo * Mult      'Assign LSB
    Result = Temp
    Mult = Mult * 256
    Temp = Result_3rd * Mult     'Assign 3rd Byte
    Result = Result + Temp
    Mult = Mult * 256
    Temp = Result_2nd * Mult     'Assign 2nd Byte
    Result = Result + Temp
    Mult = Mult * 256
    Temp = Result_hi * Mult      'Assign MSB
    Result = Result + Temp
    '
    Result = 32-bits Count Data
    '

Loop

```

**Figure 18. Typical Program for Reading HCTL-2032 with Atmel AVR**

```

Get_hi:
Hi_old = Pina          'Get current data
Hi_new = Pina          'Get 2nd Data
If Hi_new = Hi_old Then
    Result_hi = Hi_new      'Get stable data
    Return
Else
    Goto Get_2nd
End If

Get_2nd:
2nd_old = Pina          'Get current data
2nd_new = Pina          'Get 2nd Data
If 2nd_new = 2nd_old Then
    Result_2nd = 2nd_new      'Get stable data
    Return
Else
    Goto Get_2nd
End If

Get_3rd:
3rd_old = Pina          'Get current data
3rd_new = Pina          'Get 2nd Data
If 3rd_new = 3rd_old Then
    Result_3rd = 3rd_new      'Get stable data
    Return
Else
    Goto Get_3rd
End If

Get_lo:
Lo_old = Pina          'Get current data
Lo_new = Pina          'Get 2nd Data
If Lo_new = Lo_old Then
    Result_lo = Lo_new      'Get stable data
    Return
Else
    Goto Get_lo
End If

```

**Figure 18 Cont. Typical Program for Reading HCTL-2032 with Atmel AVR**

## Actions

1. At first, Port B4, B5, and B6 are setup for 4X encoding and X/Y axis selection.
2. The HCTL-2032 detects that OE/ are low on the next falling edge of the CLK and asserts the internal inhibit signal. Data can be read without regard for the phase of the CLK.
3. SEL1 and SEL2 are setup to select the appropriate bytes. The "Get\_hi" subroutine is called and the data is read into the AVR.
4. Step 3 is repeated by changing the SEL1 and SEL2 combinations and specific subroutine is called to read in the appropriate data.
5. The HCTL-2032 detects OE/ high on the next falling edge of the CLK. The program set OE/ high by writing the correct value to the respective Port. This causes the data lines to be tristated. On the next rising CLK edge new data is transferred from the counter to the position data latch.
6. For displaying purposes, the data is arranged in 32-bit data by shifting the MSB to the left through multiplication.

## Ordering Information

HCTL - 20  - 		
32	Blank	32-PDIP Package
32	SC	32-SOIC Package
32	SCT	32-SOIC Package in Tape and Reel (1000 Pcs / Reel)
22	Blank	20-PDIP Package

For product information and a complete list of distributors, please go to our web site: [www.avagotech.com](http://www.avagotech.com)

Avago, Avago Technologies, and the A logo are trademarks of Avago Technologies, Limited in the United States and other countries.  
Data subject to change. Copyright © 2007 Avago Technologies Limited. All rights reserved. Obsoletes 5989-0060EN  
AV02-0096EN - January 16, 2007



## C.11 Webbench-Tool zur Berechnung der Buck Converter

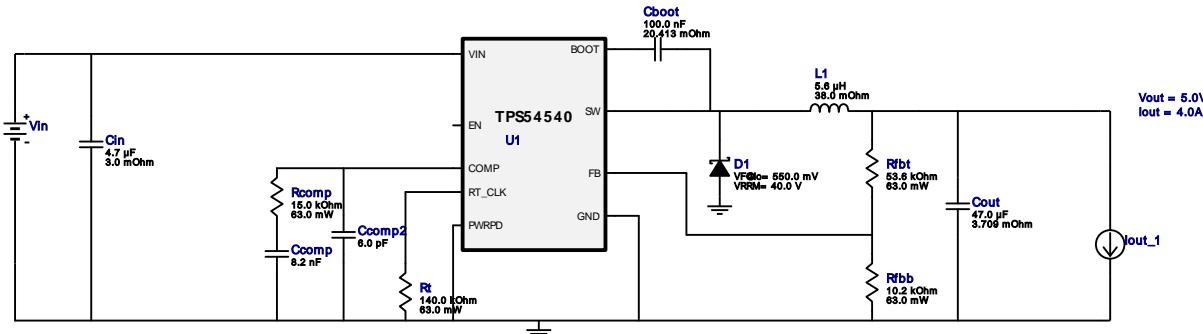
### C.11.1 Webbench Datasheet 5V Spannungsversorgung

# WEBENCH® Design Report

Design : 4550507/16 TPS54540DDAR  
 TPS54540DDAR 12.0V-24.0V to 5.00V @ 4.0A

VinMin = 12.0V  
 VinMax = 24.0V  
 Vout = 5.0V  
 Iout = 4.0A

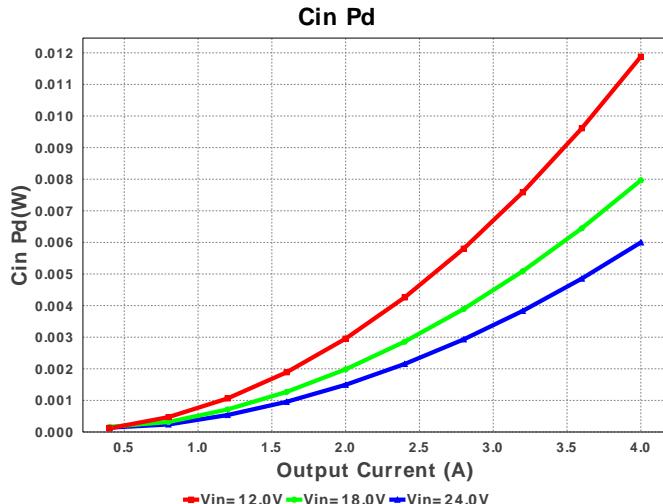
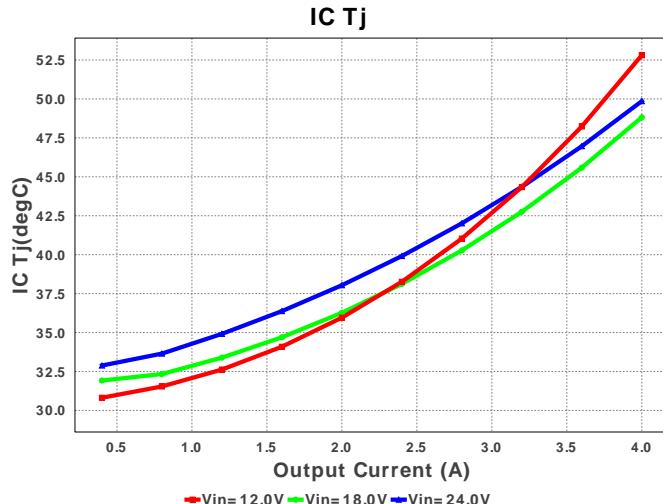
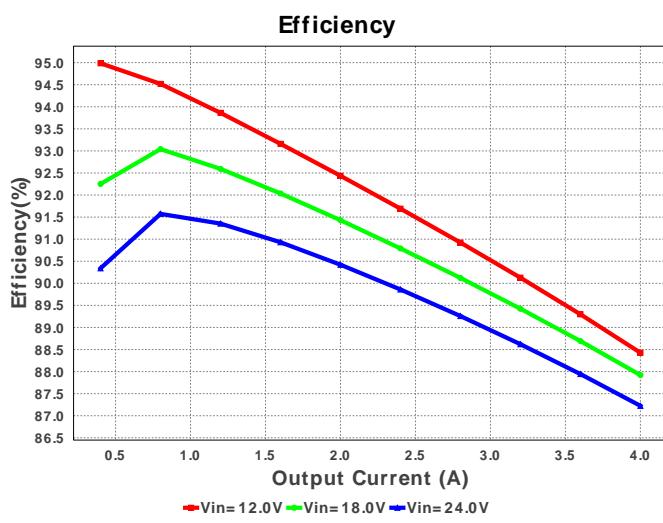
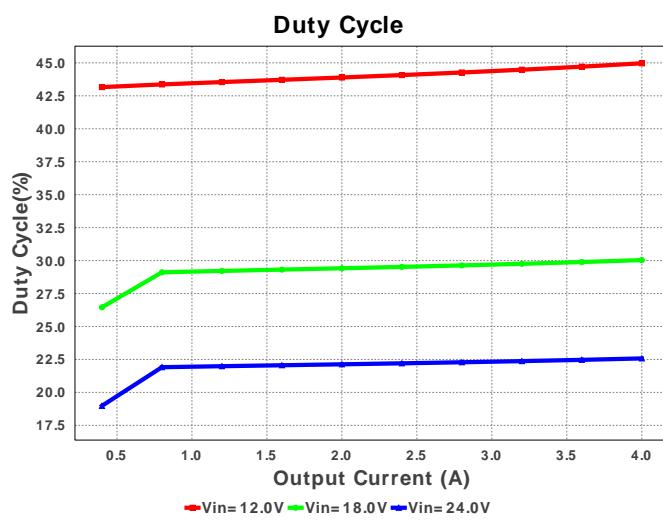
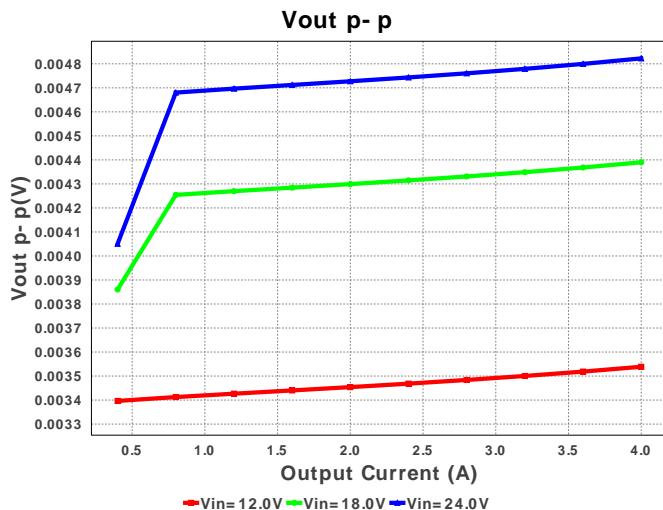
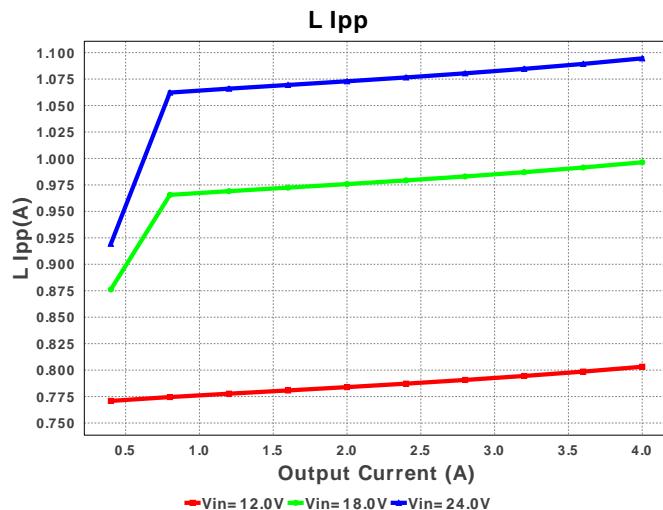
Device = TPS54540DDAR  
 Topology = Buck  
 Created = 11/28/15 5:57:26 AM  
 BOM Cost = \$2.98  
 BOM Count = 12  
 Total Pd = 3.2W

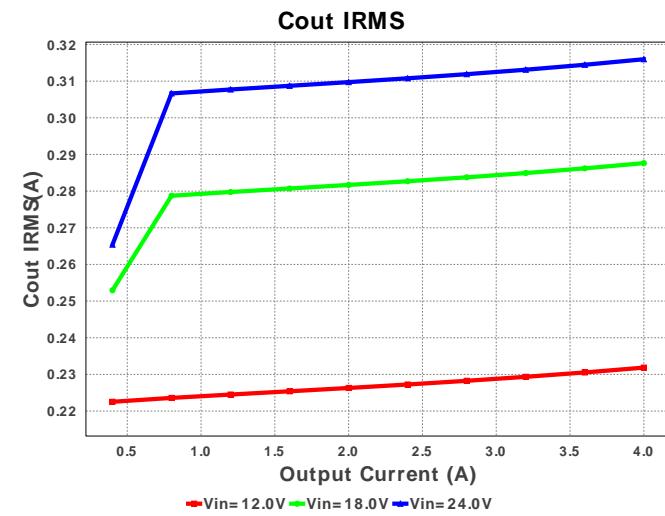
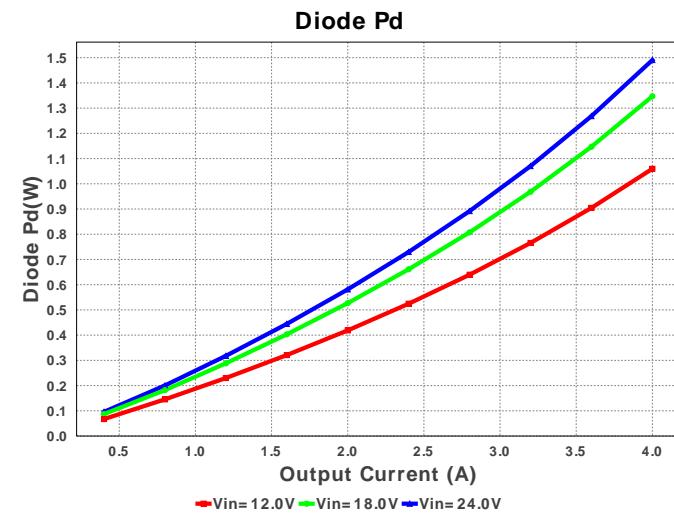
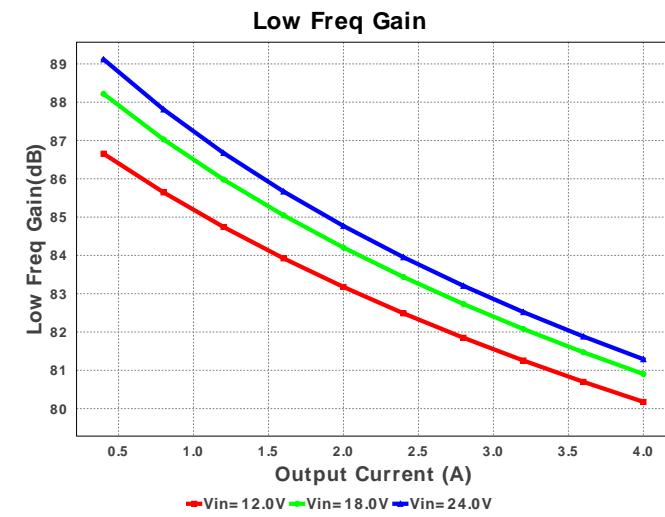
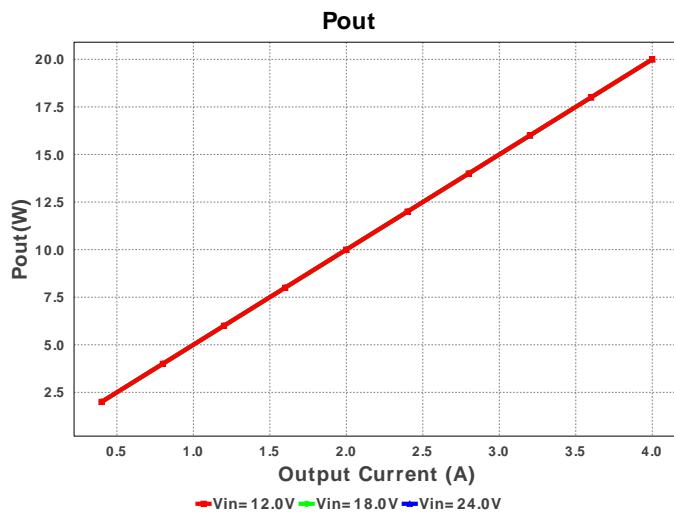
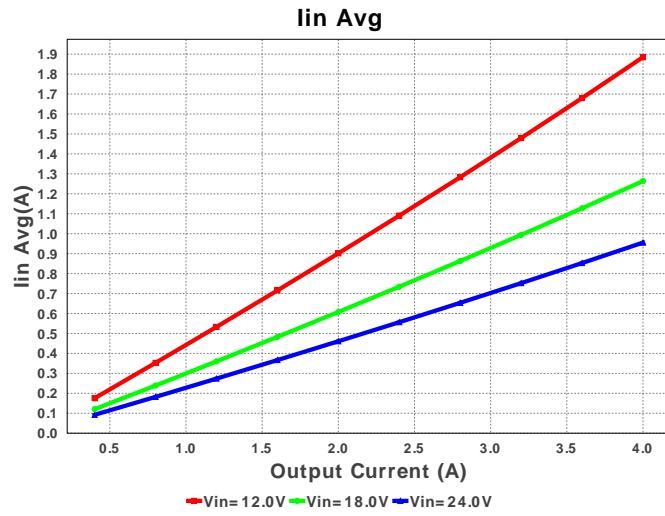
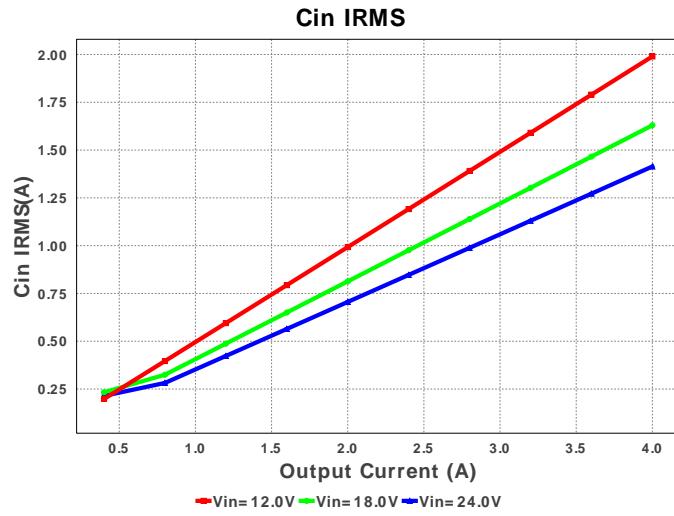


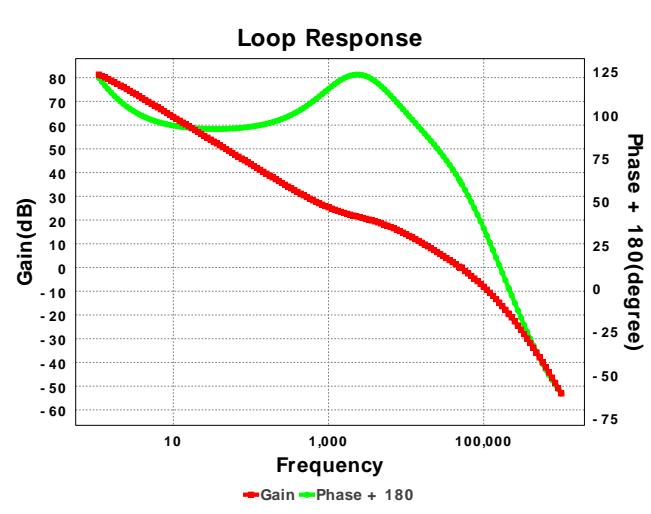
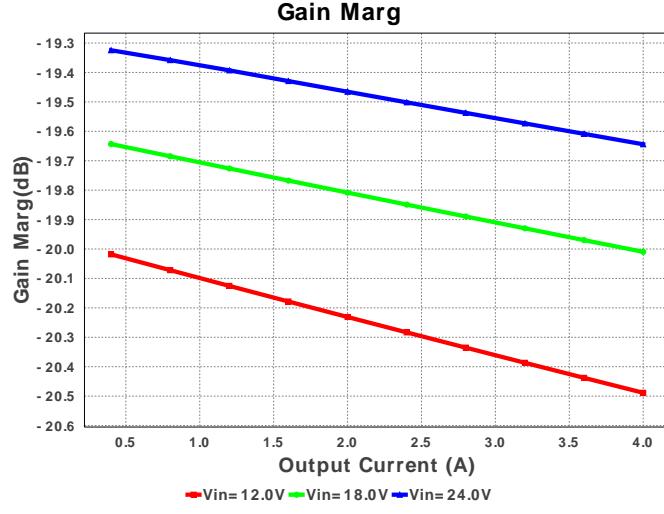
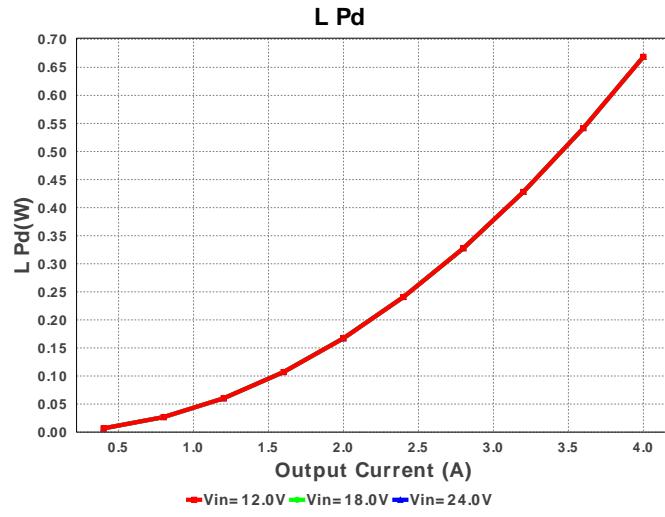
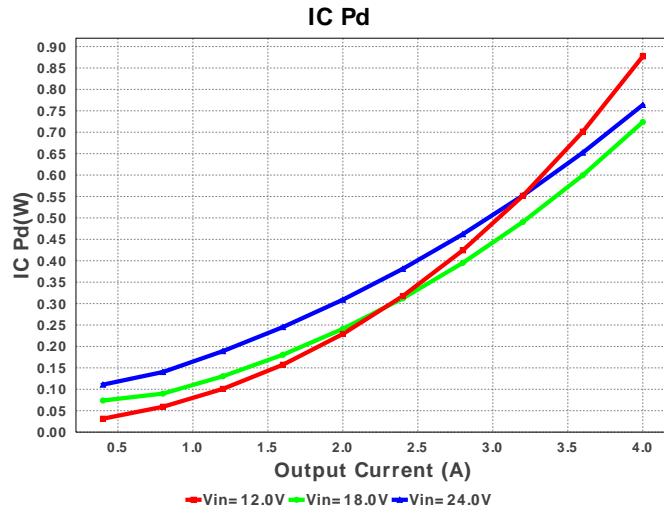
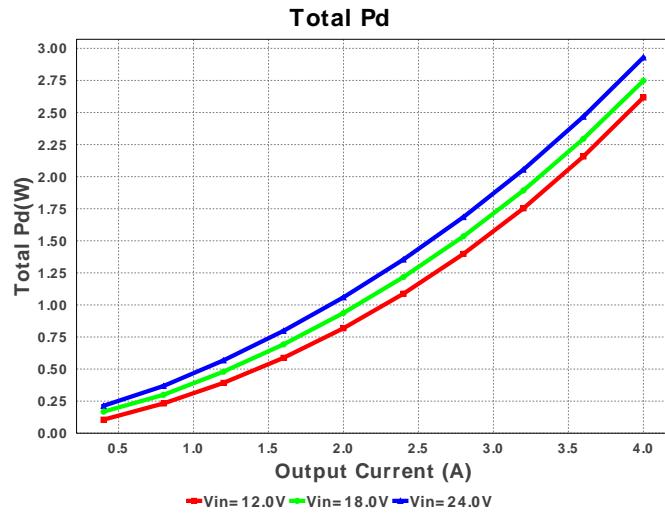
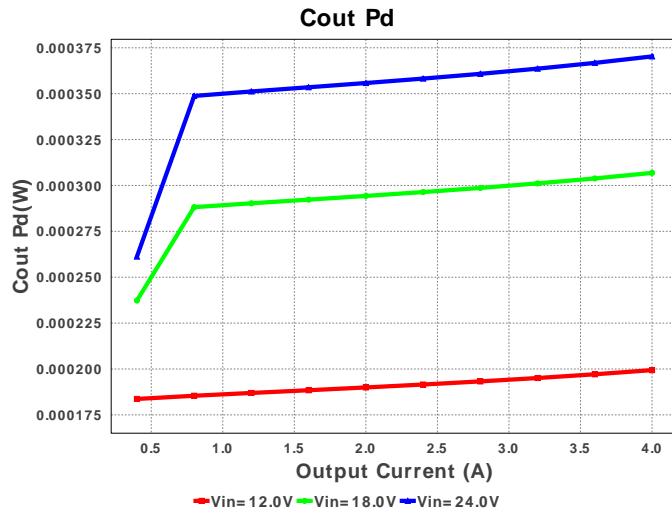
## Electrical BOM

#	Name	Manufacturer	Part Number	Properties	Qty	Price	Footprint
1.	Cboot	TDK	C1005X5R1A104K Series= X5R	Cap= 100.0 nF ESR= 20.413 mΩ VDC= 10.0 V IRMS= 0.0 A	1	\$0.01	■ 1005_3 mm²
2.	Ccomp	MuRata	GRM033R61A822KA01D Series= X5R	Cap= 8.2 nF VDC= 10.0 V IRMS= 0.0 A	1	\$0.01	■ 0201_2 mm²
3.	Ccomp2	Yageo America	CC0805DRNP09BN6R0 Series= C0G/NP0	Cap= 6.0 pF VDC= 50.0 V IRMS= 0.0 A	1	\$0.03	■ 0805_7 mm²
4.	Cin	MuRata	GRM31CR71H475KA12L Series= X7R	Cap= 4.7 uF ESR= 3.0 mΩ VDC= 50.0 V IRMS= 4.98 A	1	\$0.07	■■ 1206_11 mm²
5.	Cout	MuRata	GRM31CR61A476KE15L Series= X5R	Cap= 47.0 uF ESR= 3.709 mΩ VDC= 10.0 V IRMS= 4.2862 A	1	\$0.21	■■ 1206_190_11 mm²
6.	D1	Diodes Inc.	B540C-13-F	VF@Io= 550.0 mV VRMM= 40.0 V	1	\$0.17	■■■■■ SMC 83 mm²
7.	L1	Bourns	SRP6540-5R6M	L= 5.6 μH DCR= 38.0 mΩ	1	\$0.49	■■■■■ SRP6540 83 mm²
8.	Rcomp	Vishay-Dale	CRCW040215K0FKED Series= CRCW..e3	Res= 15.0 kΩ Power= 63.0 mW Tolerance= 1.0%	1	\$0.01	■ 0402_3 mm²
9.	Rfbb	Vishay-Dale	CRCW040210K2FKED Series= CRCW..e3	Res= 10.2 kΩ Power= 63.0 mW Tolerance= 1.0%	1	\$0.01	■ 0402_3 mm²
10.	Rfbt	Vishay-Dale	CRCW040253K6FKED Series= CRCW..e3	Res= 53.6 kΩ Power= 63.0 mW Tolerance= 1.0%	1	\$0.01	■ 0402_3 mm²
11.	Rt	Vishay-Dale	CRCW0402140KFKED Series= CRCW..e3	Res= 140.0 kΩ Power= 63.0 mW Tolerance= 1.0%	1	\$0.01	■ 0402_3 mm²

#	Name	Manufacturer	Part Number	Properties	Qty	Price	Footprint
12. U1	Texas Instruments		TPS54540DDAR	Switcher	1	\$1.95	

R-PDSO-G8 57 mm<sup>2</sup>





## Operating Values

#	Name	Value	Category	Description
1.	Cin IRMS	1.418 A	Current	Input capacitor RMS ripple current
2.	Cout IRMS	319.899 mA	Current	Output capacitor RMS ripple current
3.	Iin Avg	966.68 mA	Current	Average input current
4.	L Ipp	1.108 A	Current	Peak-to-peak inductor ripple current
5.	BOM Count	12	General	Total Design BOM count
6.	FootPrint	269.0 mm <sup>2</sup>	General	Total Foot Print Area of BOM components
7.	Frequency	699.837 kHz	General	Switching frequency
8.	Pout	20.0 W	General	Total output power
9.	Total BOM	\$2.98	General	Total BOM Cost
10.	ICThetaJA Effective	26.0 degC/W	Op_Point	Effective IC Junction-to-Ambient Thermal Resistance
11.	Low Freq Gain	81.29 dB	Op_Point	Gain at 10Hz

#	Name	Value	Category	Description
12.	Vout OP	5.0 V	Op_Point	Operational Output Voltage
13.	Cross Freq	49.876 kHz	Op_point	Bode plot crossover frequency
14.	Duty Cycle	22.858 %	Op_point	Duty cycle
15.	Efficiency	86.205 %	Op_point	Steady state efficiency
16.	Gain Marg	-19.644 dB	Op_point	Bode Plot Gain Margin
17.	IC T <sub>j</sub>	49.989 degC	Op_point	IC junction temperature
18.	IOUT_OP	4.0 A	Op_point	Iout operating point
19.	Phase Marg	61.918 deg	Op_point	Bode Plot Phase Margin
20.	VIN_OP	24.0 V	Op_point	Vin operating point
21.	Vout p-p	4.883 mV	Op_point	Peak-to-peak output ripple voltage
22.	Cin Pd	6.03 mW	Power	Input capacitor power dissipation
23.	Cout Pd	379.562 μW	Power	Output capacitor power dissipation
24.	Diode Pd	1.756 W	Power	Diode power dissipation
25.	IC Pd	768.798 mW	Power	IC power dissipation
26.	L Pd	668.8 mW	Power	Inductor power dissipation
27.	Total Pd	3.2 W	Power	Total Power Dissipation

## Design Inputs

#	Name	Value	Description
1.	Iout1	4.0	Output Current #1
2.	VinMax	24.0	Maximum input voltage
3.	VinMin	12.0	Minimum input voltage
4.	Vout1	5.0	Output Voltage #1
5.	Vout2	7.0	Output Voltage #2
6.	base_pn	TPS54540	Base Product Number
7.	source	DC	Input Source Type
8.	Ta	30.0	Ambient temperature

## Design Assistance

1. **TPS54540 Product Folder :** <http://www.ti.com/product/TPS54540> : contains the data sheet and other resources.

Texas Instruments' WEBENCH simulation tools attempt to recreate the performance of a substantially equivalent physical implementation of the design. Simulations are created using Texas Instruments' published specifications as well as the published specifications of other device manufacturers. While Texas Instruments does update this information periodically, this information may not be current at the time the simulation is built. Texas Instruments does not warrant the accuracy or completeness of the specifications or any information contained therein. Texas Instruments does not warrant that any designs or recommended parts will meet the specifications you entered, will be suitable for your application or fit for any particular purpose, or will operate as shown in the simulation in a physical implementation. Texas Instruments does not warrant that the designs are production worthy.

**You should completely validate and test your design implementation to confirm the system functionality for your application prior to production.**

Use of Texas Instruments' WEBENCH simulation tools is subject to [Texas Instruments' Site Terms and Conditions of Use](#). Prototype boards based on WEBENCH created designs are provided AS IS without warranty of any kind for evaluation and testing purposes and are subject to the terms of the [Evaluation License Agreement](#).

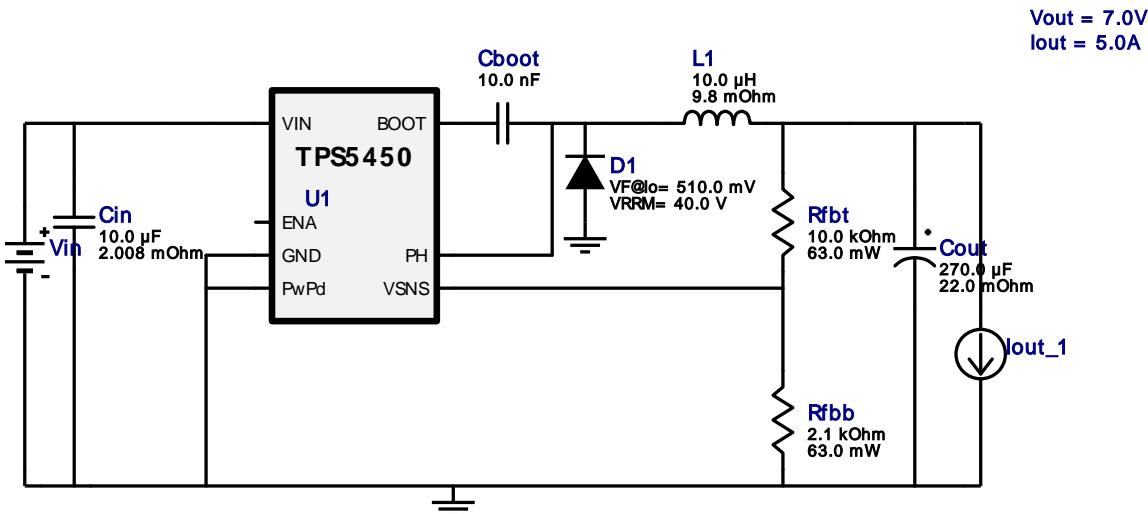
### C.11.2 Webbench Datasheet 7V Spannungsversorgung

**WEBENCH® Design Report**

Design : 4550507/22 TPS5450DDAR  
 TPS5450DDAR 12.0V-24.0V to 7.00V @ 5.0A

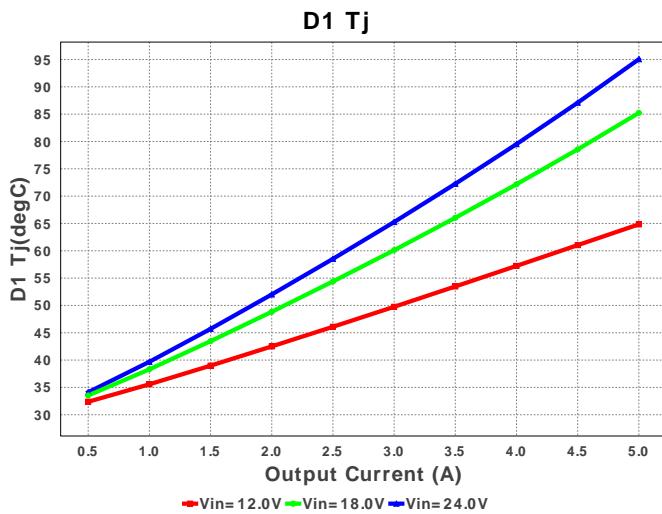
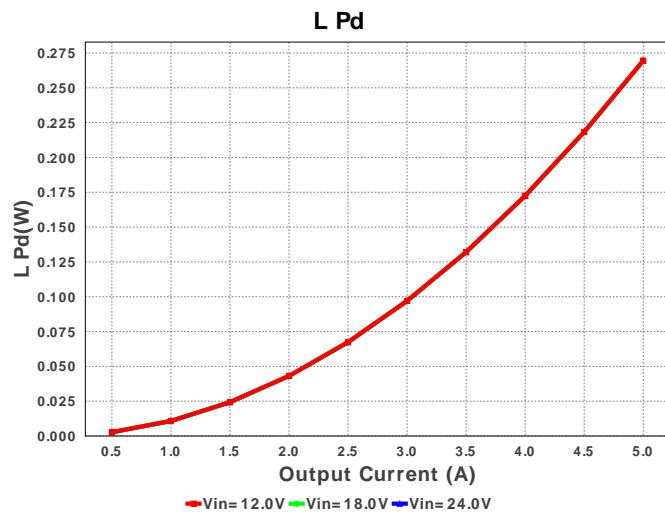
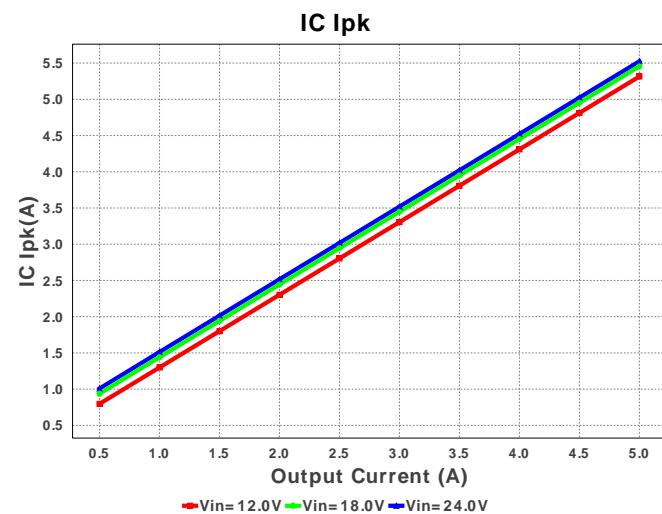
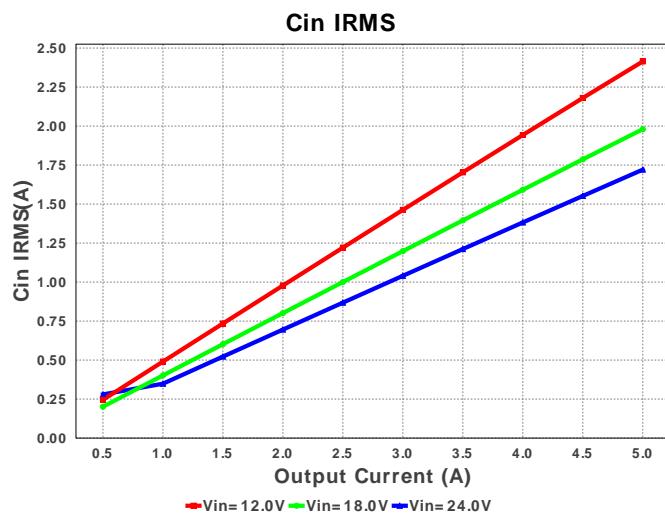
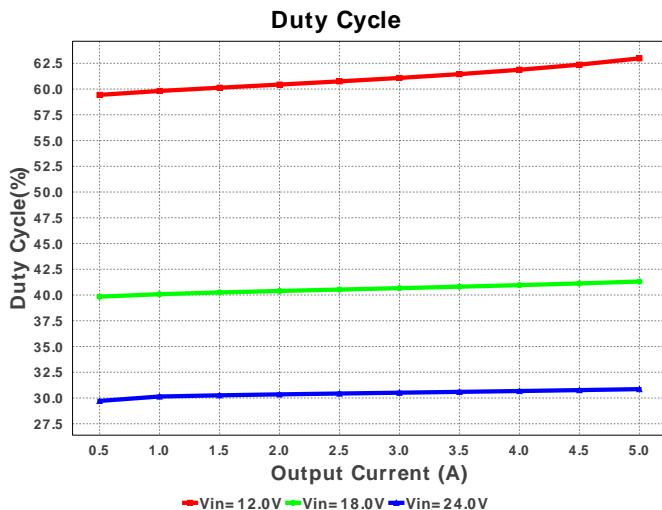
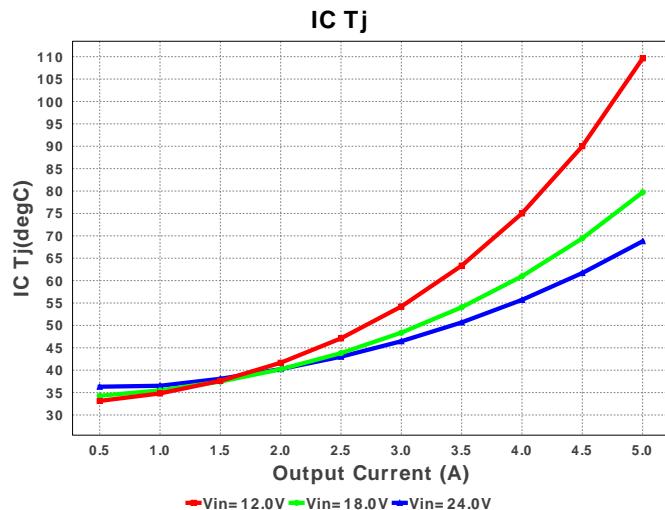
VinMin = 12.0V  
 VinMax = 24.0V  
 Vout = 7.0V  
 Iout = 5.0A

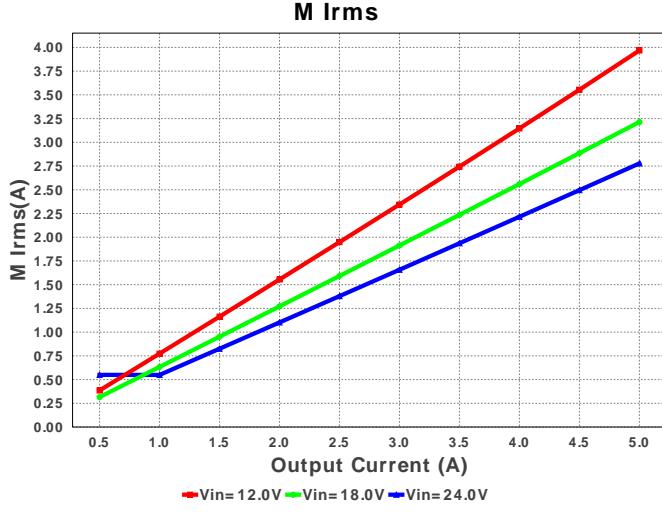
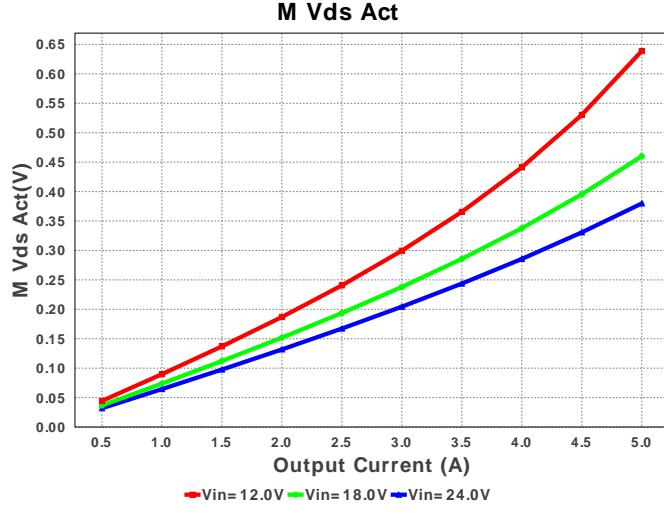
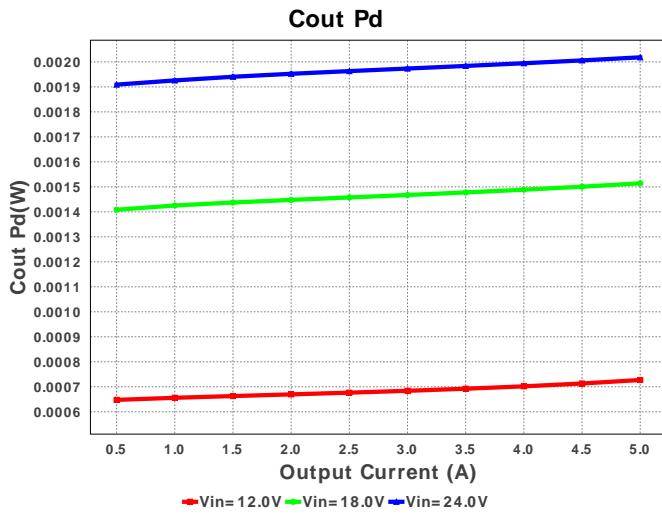
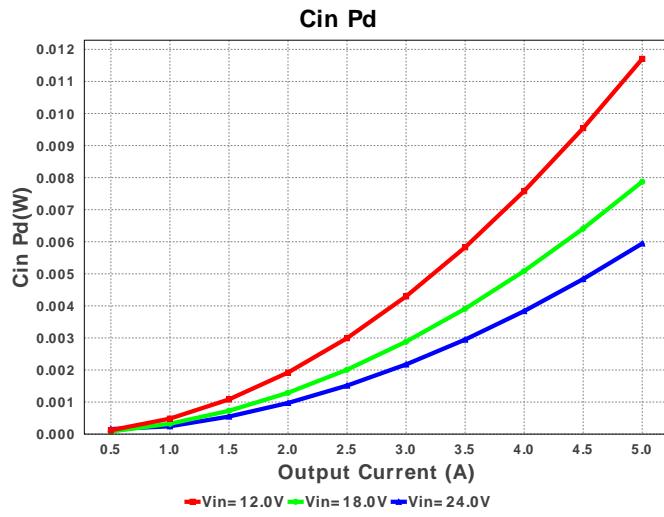
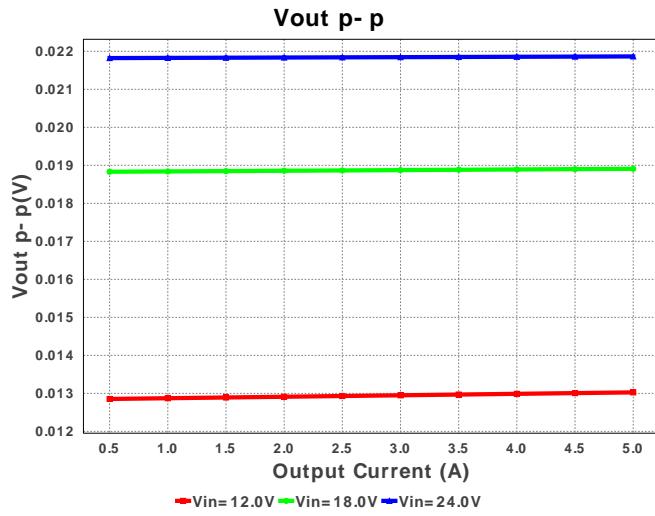
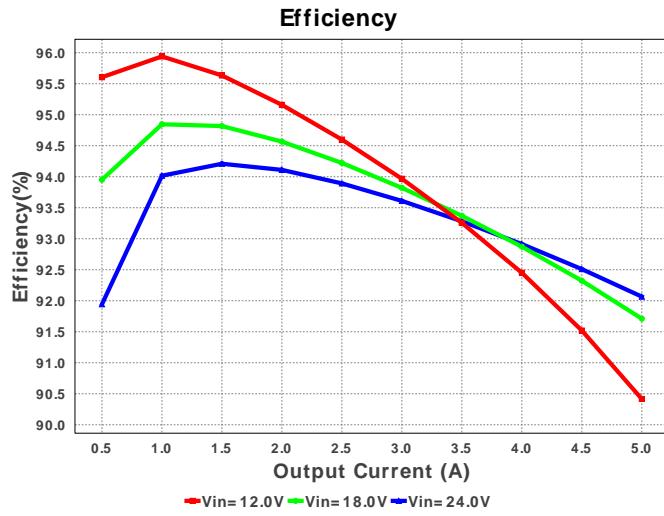
Device = TPS5450DDAR  
 Topology = Buck  
 Created = 11/28/15 7:04:54 AM  
 BOM Cost = \$4.15  
 BOM Count = 8  
 Total Pd = 3.02W

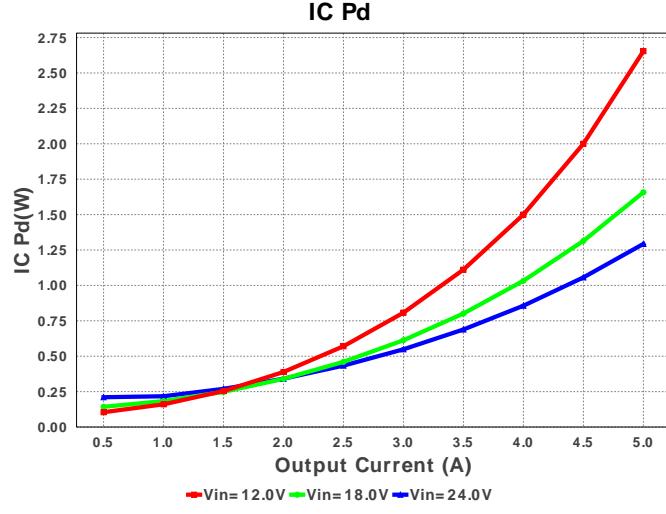
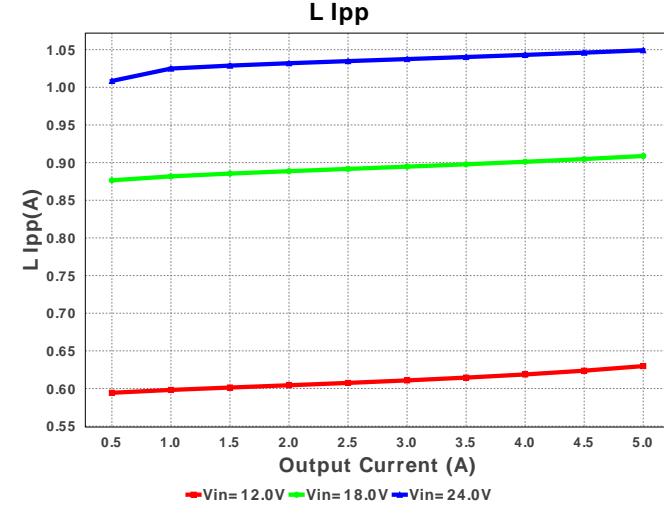
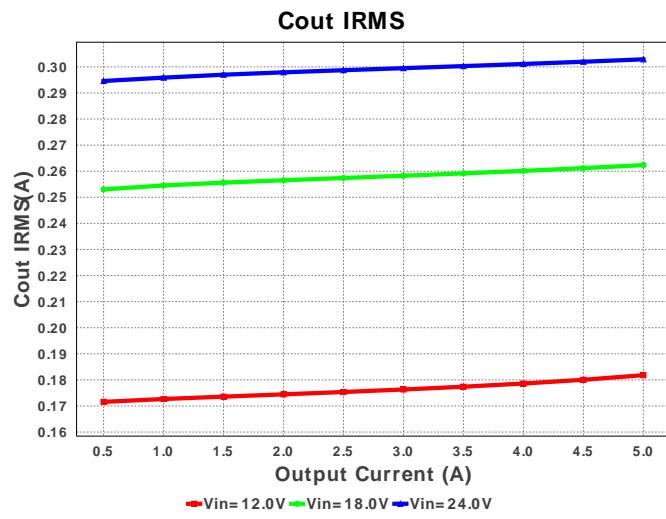
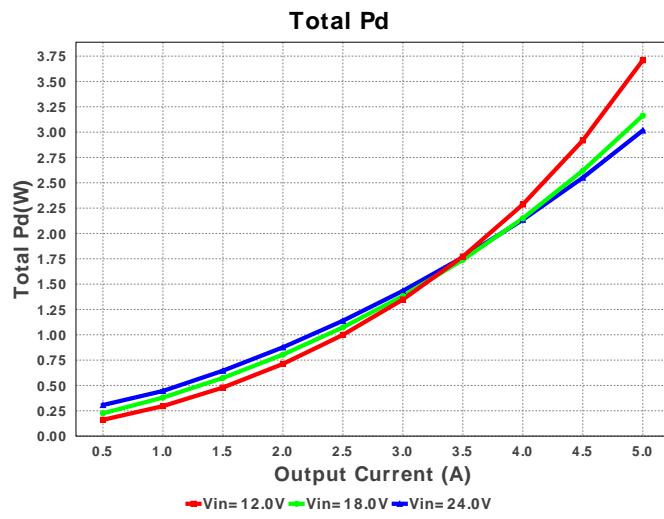
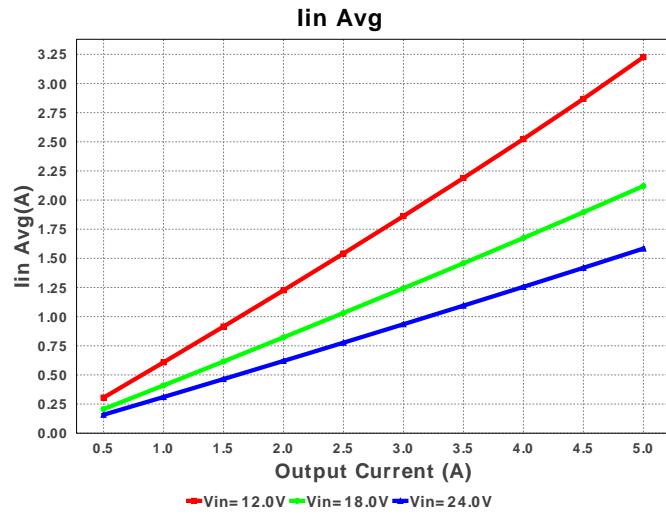
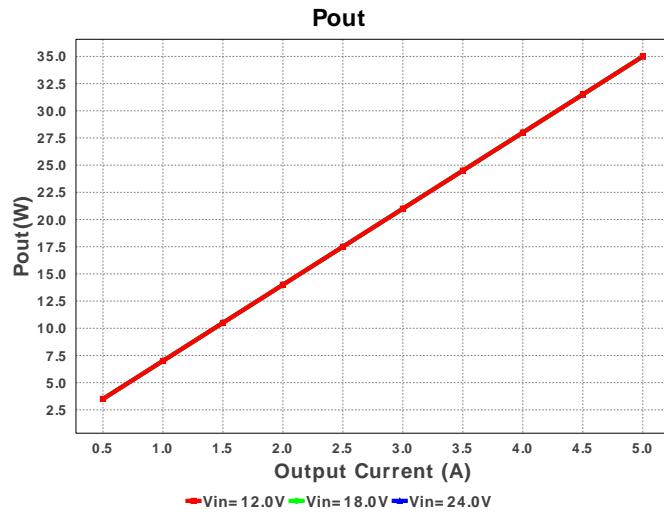

**Electrical BOM**

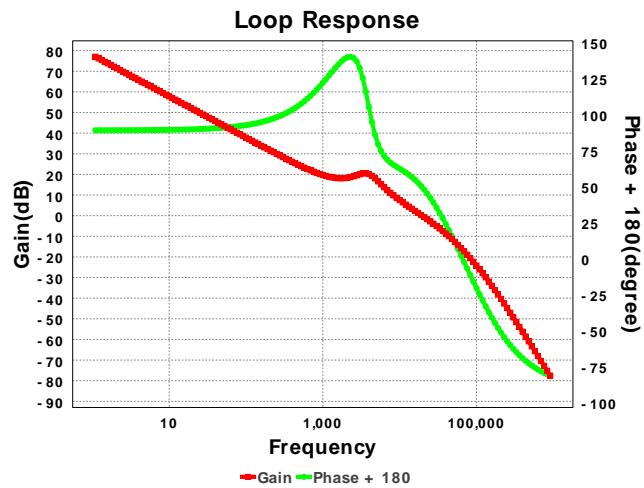
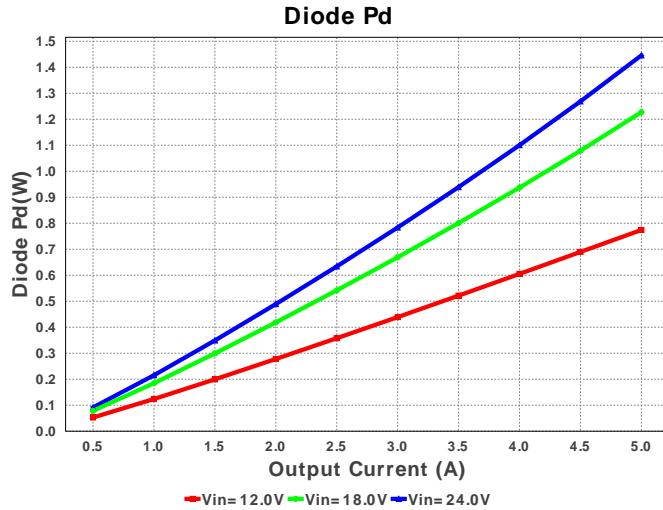
#	Name	Manufacturer	Part Number	Properties	Qty	Price	Footprint
1.	Cboot	MuRata	GRM155R71H103KA88D Series= X7R	Cap= 10.0 nF VDC= 50.0 V IRMS= 0.0 A	1	\$0.01	■ 0402 3 mm <sup>2</sup>
2.	Cin	MuRata	GRM32ER7YA106KA12L Series= X7R	Cap= 10.0 uF ESR= 2.008 mOhm VDC= 35.0 V IRMS= 4.6772 A	1	\$0.25	□□ 1210_280 15 mm <sup>2</sup>
3.	Cout	Panasonic	16SVPF270M Series= SVPF	Cap= 270.0 uF ESR= 22.0 mOhm VDC= 16.0 V IRMS= 3.3 A	1	\$0.50	○ CAPSMT_62_E7 106 mm <sup>2</sup>
4.	D1	Vishay-Semiconductor	50WQ04FNPBF	VF@Io= 510.0 mV VRMM= 40.0 V	1	\$0.40	□□□ DPAK 102 mm <sup>2</sup>
5.	L1	Coilcraft	SER1360-103KLB	L= 10.0 μH DCR= 9.8 mOhm	1	\$0.72	□□□ SER1360 225 mm <sup>2</sup>
6.	Rfb	Vishay-Dale	CRCW04022K10FKED Series= CRCW..e3	Res= 2.1 kOhm Power= 63.0 mW Tolerance= 1.0%	1	\$0.01	■ 0402 3 mm <sup>2</sup>
7.	Rfbt	Vishay-Dale	CRCW040210K0FKED Series= CRCW..e3	Res= 10.0 kOhm Power= 63.0 mW Tolerance= 1.0%	1	\$0.01	■ 0402 3 mm <sup>2</sup>

#	Name	Manufacturer	Part Number	Properties	Qty	Price	Footprint
8.	U1	Texas Instruments	TPS5450DDAR	Switcher	1	\$2.25	

R-PDSO-G8 57 mm<sup>2</sup>







## Operating Values

#	Name	Value	Category	Description
1.	Cin IRMS	1.721 A	Current	Input capacitor RMS ripple current
2.	Cout IRMS	302.888 mA	Current	Output capacitor RMS ripple current
3.	IC Ipk	5.525 A	Current	Peak switch current in IC
4.	Iin Avg	1.584 A	Current	Average input current
5.	L Ipp	1.049 A	Current	Peak-to-peak inductor ripple current
6.	M1 Irms	2.778 A	Current	Q lavg
7.	BOM Count	8	General	Total Design BOM count
8.	FootPrint	514.0 mm <sup>2</sup>	General	Total Foot Print Area of BOM components
9.	Frequency	500.0 kHz	General	Switching frequency
10.	IC Tolerance	18.315 mV	General	IC Feedback Tolerance
11.	M Vds Act	379.832 mV	General	Voltage drop across the MosFET
12.	Pout	35.0 W	General	Total output power
13.	Total BOM	\$4.15	General	Total BOM Cost
14.	D1 Tj	95.056 degC	Op_Point	D1 junction temperature
15.	Vout OP	7.0 V	Op_Point	Operational Output Voltage
16.	Cross Freq	19.611 kHz	Op_point	Bode plot crossover frequency
17.	Duty Cycle	30.86 %	Op_point	Duty cycle
18.	Efficiency	92.066 %	Op_point	Steady state efficiency
19.	IC Tj	68.79 degC	Op_point	IC junction temperature
20.	ICThetaJA	30.0 degC/W	Op_point	IC junction-to-ambient thermal resistance
21.	IOUT_OP	5.0 A	Op_point	Iout operating point
22.	Phase Marg	50.598 deg	Op_point	Bode Plot Phase Margin
23.	VIN_OP	24.0 V	Op_point	Vin operating point
24.	Vout p-p	23.104 mV	Op_point	Peak-to-peak output ripple voltage
25.	Cin Pd	5.945 mW	Power	Input capacitor power dissipation
26.	Cout Pd	2.018 mW	Power	Output capacitor power dissipation
27.	Diode Pd	1.446 W	Power	Diode power dissipation
28.	IC Pd	1.293 W	Power	IC power dissipation
29.	L Pd	269.5 mW	Power	Inductor power dissipation
30.	Total Pd	3.016 W	Power	Total Power Dissipation

## Design Inputs

#	Name	Value	Description
1.	Iout	5.0	Maximum Output Current
2.	VinMax	24.0	Maximum input voltage
3.	VinMin	12.0	Minimum input voltage
4.	Vout	7.0	Output Voltage
5.	base_pn	TPS5450	Base Product Number
6.	source	DC	Input Source Type
7.	Ta	30.0	Ambient temperature

## Design Assistance

1. Feature Highlights: 5A, 500kHz Fixed Switching Frequency, Internal Compensation
2. **TPS5450 Product Folder** : <http://www.ti.com/product/TPS5450> : contains the data sheet and other resources.

Texas Instruments' WEBENCH simulation tools attempt to recreate the performance of a substantially equivalent physical implementation of the design. Simulations are created using Texas Instruments' published specifications as well as the published specifications of other device manufacturers. While Texas Instruments does update this information periodically, this information may not be current at the time the simulation is built. Texas Instruments does not warrant the accuracy or completeness of the specifications or any information contained therein. Texas Instruments does not warrant that any designs or recommended parts will meet the specifications you entered, will be suitable for your application or fit for any particular purpose, or will operate as shown in the simulation in a physical implementation. Texas Instruments does not warrant that the designs are production worthy.

**You should completely validate and test your design implementation to confirm the system functionality for your application prior to production.**

Use of Texas Instruments' WEBENCH simulation tools is subject to [Texas Instruments' Site Terms and Conditions of Use](#). Prototype boards based on WEBENCH created designs are provided AS IS without warranty of any kind for evaluation and testing purposes and are subject to the terms of the [Evaluation License Agreement](#).

## C.12 Testsoftware Sensorsteuerung

In diesem Teil des Anhangs, ist der Source Code aufgeführt, welcher für das Testen der eingesetzten Sensoren verwendet wurde. Den vollständigen Source Code inklusive den CMake ist auf der CD beigelegt.

```

1 #ifndef BRICKLET_H
2 #define BRICKLET_H
3
4 #include <inttypes.h>
5
6 typedef struct {
7     char id[8];
8     char idConnected[8];
9     char position;
10    uint8_t hardwareVersion[3];
11    uint8_t firmwareVersion[3];
12    uint16_t deviceIdentifier;
13} BrickletIdentification;
14
15#endif // BRICKLET_H

```

..../SensorControlCode/Sensors/inc/BrickletIdentification.h

```

1 #ifndef SERVO_CONFIGURATION_H
2 #define SERVO_CONFIGURATION_H
3
4 #include <inttypes.h>
5
6 typedef struct {
7     uint16_t velocity;
8     uint16_t acceleration;
9     uint16_t minPulseWidth;
10    uint16_t maxPulseWidth;
11    int16_t minDegree;
12    int16_t maxDegree;
13    uint16_t periodLength;
14} ServoConfiguration;
15
16#endif // SERVO_CONFIGURATION_H

```

..../SensorControlCode/Sensors/inc/ServoConfiguration.h

```

1 #ifndef COLOR_SENSOR_H
2 #define COLOR_SENSOR_H
3
4 #include "ip_connection.h"

```

```

6 #include "bricklet_color.h"
8
10 // color values from 0 to 65535
12 typedef struct {
14     uint16_t r;
16     uint16_t g;
18     uint16_t b;
20 } RGBColor;
22 }

24 // color composition in percentage
26 typedef struct {
28     float r;
30     float g;
32     float b;
34 } RGBChroma;
36
38 class ColorSensor
39 {
40     public:
41         ColorSensor(char const *brickletId);
42         ~ColorSensor();
43
44         int initialize(IPConnection *connection);
45
46         void enableLight();
47         void disableLight();
48         bool isLightOn();
49
50         RGBColor getCurrentColor();
51         RGBChroma getCurrentColorComposition();
52         uint32_t getIlluminance();
53         uint64_t getIlluminanceInLux();
54         uint16_t getColorTemperature();
55
56     private:
57         Color bricklet;
58         BrickletIdentification identification;
59         bool initialized;
60     };
61
62 #endif // COLOR SENSOR H

```

..../SensorControlCode/Sensors/inc/ColorSensor.h

```
#include <iostream>
2
#include "ColorSensor.h"
4
ColorSensor::ColorSensor(char const *brickletId)
6{
    strncpy(this->identification.id, brickletId, 8 >= strlen(
        brickletId) ? strlen(brickletId) : 8);
8
    this->initialized = false;
10}
12
ColorSensor::~ColorSensor()
{
14    if (this->initialized) {
        color_destroy(&this->bricklet);
16    }
18}
19
int ColorSensor::initialize(IPConnection *connection)
20{
    int deviceSetupErrorCode = 0;
22
    color_create(&this->bricklet, this->identification.id,
        connection);
24
    std::memset(&this->identification, 0x00, sizeof(this->
        identification));
26
    deviceSetupErrorCode = color_get_identity(
        &this->bricklet,
        this->identification.id,
        this->identification.idConnected,
        &this->identification.position,
        this->identification.hardwareVersion,
        this->identification.firmwareVersion,
        &this->identification.deviceIdentifier
    );
28
30
    if (deviceSetupErrorCode != 0 || this->identification.id[0] ==
        0x00) {
32        return ERROR_NO_BRICKLET_CONNECTION;
34    }
36
38    deviceSetupErrorCode = color_set_config(
40
```

```
42         &this->bricklet ,
43         COLOR_GAIN_60X,
44         COLOR_INTEGRATION_TIME_101MS
45     );
46
47     if (deviceSetupErrorCode != 0) {
48         return ERROR_DEVICE_SETUP_FAILED;
49     }
50
51     this->initialized = true;
52
53     return 0;
54 }
55
56 void ColorSensor::enableLight()
57 {
58     color_light_on(&this->bricklet);
59 }
60
61 void ColorSensor::disableLight()
62 {
63     color_light_off(&this->bricklet);
64 }
65
66 bool ColorSensor::isLightOn()
67 {
68     uint8_t lightReturn = 0;
69
70     color_is_light_on(&this->bricklet, &lightReturn);
71
72     if (lightReturn == COLOR_LIGHT_ON) {
73         return true;
74     }
75
76     return false;
77 }
78
79 RGBColor ColorSensor::getCurrentColor()
80 {
81     RGBColor currentColor;
82     uint16_t clear; // TODO: unused
83
84     color_get_color(
85         &this->bricklet ,
86         &currentColor.r ,
87         &currentColor.g ,
```

```
88         &currentColor.b,
90         &clear);
92     }
94 RGBChroma ColorSensor::getCurrentColorComposition()
95 {
96     RGBChroma currentColorComposition;
97     RGBColor currentColor = this->getCurrentColor();
98
99     currentColorComposition.r = ((float)currentColor.r / (float)
100        UINT16_MAX);
101     currentColorComposition.g = ((float)currentColor.g / (float)
102        UINT16_MAX);
103     currentColorComposition.b = ((float)currentColor.b / (float)
104        UINT16_MAX);
105
106     return currentColorComposition;
107 }
108
109 uint32_t ColorSensor::getIlluminance()
110 {
111     int deviceCode = E_OK;
112     uint32_t illuminance = 0;
113
114     deviceCode = color_get_illuminance(
115         &this->bricklet,
116         &illuminance
117     );
118
119     if (deviceCode != E_OK) {
120         return 0;
121     }
122
123     return illuminance;
124 }
125
126 uint64_t ColorSensor::getIlluminanceInLux()
127 {
128     uint32_t illuminance = this->getIlluminance();
129     //TODO: check if rgb values are not over 65535
130
131     //illuminance * 700 / gain / integration_time
132     return illuminance * 700 / 60 / 101;
133 }
```

```

132     uint16_t ColorSensor :: getColorTemperature()
133     {
134         int deviceCode = E_OK;
135         uint16_t colorTemperature;
136
137         //TODO: check if rgb values are not over 65535
138         deviceCode = color_get_color_temperature(
139             &this->bricklet,
140             &colorTemperature
141         );
142
143         if (deviceCode != E_OK) {
144             return 0;
145         }
146
147         return colorTemperature;
148     }

```

..../SensorControlCode/Sensors/src/ColorSensor.cxx

```

2 #ifndef USDISTANCE_CONTROLLER_H
3 #define USDISTANCE_CONTROLLER_H
4
5 #include "ip_connection.h"
6 #include "bricklet_distance_us.h"
7
8 #include "Defines.h"
9 #include "BrickletIdentification.h"
10
11 class USDistanceController
12 {
13     public:
14         USDistanceController(char const *brickletId);
15         ~USDistanceController();
16
17         int initialize(IPConnection *connection);
18
19         uint16_t getDistanceValue();
20
21     private:
22         Servo bricklet;
23         BrickletIdentification identification;
24         bool initialized;
25     };
26 #endif // USDISTANCE_CONTROLLER_H

```

..../SensorControlCode/Sensors/inc/USDistanceController.h

```
#include <iostream>
2
#include "ColorSensor.h"
4
ColorSensor::ColorSensor(char const *brickletId)
6{
    strncpy(this->identification.id, brickletId, 8 >= strlen(
        brickletId) ? strlen(brickletId) : 8);
8
    this->initialized = false;
10}
12
ColorSensor::~ColorSensor()
{
14    if (this->initialized) {
        color_destroy(&this->bricklet);
16    }
18}
19
int ColorSensor::initialize(IPConnection *connection)
20{
    int deviceSetupErrorCode = 0;
22
    color_create(&this->bricklet, this->identification.id,
        connection);
24
    std::memset(&this->identification, 0x00, sizeof(this->
        identification));
26
    deviceSetupErrorCode = color_get_identity(
28        &this->bricklet,
        this->identification.id,
30        this->identification.idConnected,
        &this->identification.position,
32        this->identification.hardwareVersion,
        this->identification.firmwareVersion,
34        &this->identification.deviceIdentifier
    );
36
    if (deviceSetupErrorCode != 0 || this->identification.id[0] ==
        0x00) {
38        return ERROR_NO_BRICKLET_CONNECTION;
    }
}
```

```
40     deviceSetupErrorCode = color_set_config(
41         &this->bricklet,
42         COLOR_GAIN_60X,
43         COLOR_INTEGRATION_TIME_101MS
44     );
45
46     if (deviceSetupErrorCode != 0) {
47         return ERROR_DEVICE_SETUP_FAILED;
48     }
49
50     this->initialized = true;
51
52     return 0;
53 }
54
55 void ColorSensor::enableLight()
56 {
57     color_light_on(&this->bricklet);
58 }
59
60 void ColorSensor::disableLight()
61 {
62     color_light_off(&this->bricklet);
63 }
64
65 bool ColorSensor::isLightOn()
66 {
67     uint8_t lightReturn = 0;
68
69     color_is_light_on(&this->bricklet, &lightReturn);
70
71     if (lightReturn == COLOR_LIGHT_ON) {
72         return true;
73     }
74
75     return false;
76 }
77
78 RGBColor ColorSensor::getCurrentColor()
79 {
80     RGBColor currentColor;
81     uint16_t clear; // TODO: unused
82
83     color_get_color(
84         &this->bricklet,
```

```

86         &currentColor.r,
87         &currentColor.g,
88         &currentColor.b,
89         &clear);
90
91     return currentColor;
92 }
93
94 RGBChroma ColorSensor ::getCurrentColorComposition()
95 {
96     RGBChroma currentColorComposition;
97     RGBColor currentColor = this->getCurrentColor();
98
99     currentColorComposition.r = ((float)currentColor.r / (float)
100        UINT16_MAX);
101     currentColorComposition.g = ((float)currentColor.g / (float)
102        UINT16_MAX);
103     currentColorComposition.b = ((float)currentColor.b / (float)
104        UINT16_MAX);
105
106     return currentColorComposition;
107 }
108
109
110 uint32_t ColorSensor ::getIlluminance()
111 {
112     int deviceCode = E_OK;
113     uint32_t illuminance = 0;
114
115     deviceCode = color_get_illuminance(
116         &this->bricklet,
117         &illuminance
118     );
119
120     if (deviceCode != E_OK) {
121         return 0;
122     }
123
124     return illuminance;
125 }
126
127 uint64_t ColorSensor ::getIlluminanceInLux()
128 {
129     uint32_t illuminance = this->getIlluminance();
130     //TODO: check if rgb values are not over 65535
131
132     //illuminance * 700 / gain / integration_time
133 }
```

```

130     return illuminance * 700 / 60 / 101;
131 }

132 uint16_t ColorSensor :: getColorTemperature()
133 {
134     int deviceCode = E_OK;
135     uint16_t colorTemperature;
136
137     //TODO: check if rgb values are not over 65535
138     deviceCode = color_get_color_temperature(
139         &this->bricklet,
140         &colorTemperature
141     );
142
143     if (deviceCode != E_OK) {
144         return 0;
145     }
146
147     return colorTemperature;
148 }
```

..../SensorControlCode/Sensors/src/ColorSensor.cxx

```

2 #ifndef SERVO_CONTROLLER_H
3 #define SERVO_CONTROLLER_H

4 #include "ip_connection.h"
5 #include "brick_servo.h"

6 #include "Defines.h"
7 #include "BrickletIdentification.h"
8 #include "ServoConfiguration.h"

10 class ServoController
11 {
12     public:
13         ServoController(char const *brickletId);
14         ~ServoController();

16         int initialize(IPConnection *connection);

18         int setOutputVoltage(uint16_t voltage);
19         int initializeServo(const uint8_t servoPosition,
20             ServoConfiguration const * setup);
21         int initializePWM(const uint8_t servoPosition);

22         void enableServo(const uint8_t servoPosition);
```

```

24         void disableServo( const uint8_t servoPosition);

26         void setServoPosition( const uint8_t servoPosition, const
27             uint16_t newPosition);
28         uint16_t getServoPosition( const uint8_t servoPosition);

29     void setDutyCycle( uint8_t servoPosition, uint8_t
30             percentage);

31     private:
32         Servo bricklet;
33         BrickletIdentification identification;
34         bool initialized;

35         uint16_t outputVoltage;
36         ServoConfiguration* configurations[7];

37         const uint16_t defaultPWMFrequency = 20000;
38     };

39 #endif // SERVO_CONTROLLER_H

```

..../SensorControlCode/Sensors/inc/ServoController.h

```

1 #include <iostream>

2 #include "ColorSensor.h"

4 ColorSensor::ColorSensor( char const *brickletId )
5 {
6     strncpy( this->identification.id, brickletId, 8 >= strlen(
7         brickletId) ? strlen(brickletId) : 8);

8     this->initialized = false;
9 }

11 ColorSensor::~ColorSensor()
12 {
13     if (this->initialized) {
14         color_destroy(&this->bricklet);
15     }
16 }

17 int ColorSensor::initialize( IPConnection *connection )
18 {
19     int deviceSetupErrorCode = 0;
20 }

```

```
    color_create(&this->bricklet, this->identification.id,
24      connection);

26      std::memset(&this->identification, 0x00, sizeof(this->
28        identification));

30      deviceSetupErrorCode = color_get_identity(
32        &this->bricklet,
34        this->identification.id,
36        this->identification.idConnected,
38        &this->identification.position,
40        this->identification.hardwareVersion,
42        this->identification.firmwareVersion,
44        &this->identification.deviceIdentifier
46      );

48      if (deviceSetupErrorCode != 0 || this->identification.id[0] ==
50        0x00) {
52        return ERROR_NO_BRICKLET_CONNECTION;
54      }

56      deviceSetupErrorCode = color_set_config(
58        &this->bricklet,
60        COLOR_GAIN_60X,
62        COLOR_INTEGRATION_TIME_101MS
64      );

66      if (deviceSetupErrorCode != 0) {
68        return ERROR_DEVICE_SETUP_FAILED;
70      }

72      this->initialized = true;

74      return 0;
76    }

78    void ColorSensor::enableLight()
79    {
80      color_light_on(&this->bricklet);
81    }

83    void ColorSensor::disableLight()
84    {
85      color_light_off(&this->bricklet);
86    }
88  }
```

```
66 bool ColorSensor :: isLightOn()
67 {
68     uint8_t lightReturn = 0;
69
70     color_is_light_on(&this->bricklet, &lightReturn);
71
72     if (lightReturn == COLOR_LIGHT_ON) {
73         return true;
74     }
75
76     return false;
77 }
78
79 RGBColor ColorSensor :: getCurrentColor()
80 {
81     RGBColor currentColor;
82     uint16_t clear; // TODO: unused
83
84     color_get_color(
85         &this->bricklet,
86         &currentColor.r,
87         &currentColor.g,
88         &currentColor.b,
89         &clear);
90
91     return currentColor;
92 }
93
94 RGBChroma ColorSensor :: getCurrentColorComposition()
95 {
96     RGBChroma currentColorComposition;
97     RGBColor currentColor = this->getCurrentColor();
98
99     currentColorComposition.r = ((float)currentColor.r / (float)
100        UINT16_MAX);
101     currentColorComposition.g = ((float)currentColor.g / (float)
102        UINT16_MAX);
103     currentColorComposition.b = ((float)currentColor.b / (float)
104        UINT16_MAX);
105
106     return currentColorComposition;
107 }
108
109 uint32_t ColorSensor :: getIlluminance()
110 {
111     int deviceCode = E_OK;
```

```
110     uint32_t illuminance = 0;
111
112     deviceCode = color_get_illuminance(
113         &this->bricklet,
114         &illuminance
115     );
116
117     if (deviceCode != E_OK) {
118         return 0;
119     }
120
121     return illuminance;
122 }
123
124 uint64_t ColorSensor::getIlluminanceInLux()
125 {
126     uint32_t illuminance = this->getIlluminance();
127     //TODO: check if rgb values are not over 65535
128
129     //illuminance * 700 / gain / integration_time
130     return illuminance * 700 / 60 / 101;
131 }
132
133 uint16_t ColorSensor::getColorTemperature()
134 {
135     int deviceCode = E_OK;
136     uint16_t colorTemperature;
137
138     //TODO: check if rgb values are not over 65535
139     deviceCode = color_get_color_temperature(
140         &this->bricklet,
141         &colorTemperature
142     );
143
144     if (deviceCode != E_OK) {
145         return 0;
146     }
147
148     return colorTemperature;
149 }
```

..../SensorControlCode/Sensors/src/ColorSensor.cxx

### C.13 Testsoftware Containererkennung

In diesem Teil des Anhangs, ist der Source Code aufgeführt, welcher für das Testen der Container-Erkennung verwendet wurde.

```
from SimpleCV import *

class Box:
    CONTAINER_WIDTH_MM = 41.17

    def __init__(self, center_x, center_y, width, height):
        self.center_x = center_x
        self.center_y = center_y
        self.width = width
        self.height = height

        self.top_x = center_x - width / 2
        self.top_y = center_y - height / 2

    def calculate_distance(self, focal_length):
        self.distance = (focal_length) / self.width

# http://www.pyimagesearch.com/2015/01/19/find-distance-camera-
# objectmarker-using-python-opencv/
reference_distance_mm = 297
reference_width_of_container_px = 150
focal_length_insight_camera = (reference_width_of_container_px *
                                reference_distance_mm)

def detect_box(img):
    blue_distance = img.colorDistance(Color.BLUE)
    blue_distance_binarized = blue_distance.binarize(125)
    blobs = blue_distance_binarized.findBlobs(minsize=500)

    if blobs:
        rectangles = blobs.filter([b.isRectangle(0.2) for b in
                                   blobs])

        if len(rectangles) is 1:
            rectangle = rectangles[0]

            box = Box(rectangle.x, rectangle.y, rectangle._mWidth,
                      rectangle._mHeight)
```

```
38         box.calculate_distance(focal_length_insight_camera)
39         box.rectangle = rectangle
40
41         return box
42     return None
43
44
45     display = Display()
46     cam = SimpleCV.Camera()
47     while display.isNotDone():
48         img = cam.getImage()
49         box = detect_box(img)
50
51         if box is not None:
52             box.rectangle.drawHull(layer=img.dl())
53
54             distance_text = "Distance: {} cm".format(round(box.
55                 distance / 10, 2))
56             img.drawText(distance_text, box.top_x, box.top_y - 30,
57                         color=Color.YELLOW, fontsize=25)
58
59             img.show()
```

..../ContainerDetectionTest/container\_detection.py

## C.14 Testsoftware Spurhaltung

In diesem Teil des Anhangs, ist der Source Code aufgeführt, welcher für das Testen der Spurhaltung verwendet wurde.

```

from SimpleCV import *
import os
import cv2

display = SimpleCV.Display()
cam = SimpleCV.Camera(0)
normaldisplay = True
#Video drive around track mounted on vehicle
vidcam = VirtualCamera("fahrbahn3.mp4", "video")

while display.isNotDone():

    if display.mousePosition:
        #cv2.waitKey(1)
        normaldisplay = not (normaldisplay)
        print "Display Mode:", "Normal" if normaldisplay else "Segmented"

    img = vidcam.getImage()
    width = 800
    height = 450
    yOffset = img.height - height
    xOffset = 0
    imgsmall = img.crop(xOffset, yOffset, width, height)
    #img = cam.getImage()
    linelayer = DrawingLayer((img.width, img.height))

    linelayer.circle((260, img.height), 5, Color.RED, 1, True)

    #Filter white parts
    white = imgsmall.colorDistance(SimpleCV.Color.WHITE)
    segmented = white.binarize(-1)
    colorDist = imgsmall.colorDistance(SimpleCV.Color.BLACK).
        dilate(2)
    segmented = colorDist.stretch(240, 255)

    #Filter size of blobs. Range is quite high because of camera
    #perspective.
    blobs = segmented.findBlobs(-1, 600, 6500)
    if blobs:

```

```

    for blob in blobs:
40        pointblob = (blob.x + xOffset, blob.y + yOffset)
        linelayer.text(str(blob.area()), pointblob, Color.
                      BLACK)
42        linelayer.text(str(blob.height()), (blob.x + xOffset,
                                             blob.y - 20 + yOffset), Color.BLACK)
        sizedblobs = segmented.findBlobs(-1, 600, 6500)

44
45    if sizedblobs:
46        blobs = sizedblobs.filter(sizedblobs.height() >= 20)
47        if blobs:
48            heightRestricted = blobs.filter(blobs.height() <= 150)

50
51    if heightRestricted.__len__() >= 4:
52        blobs_sort = blobs.sortDistance(point=(260,
                                                 imgsmall.height))
53        start = 0
54        end = 1
55        if blobs_sort[0].below((0, imgsmall.height - 150)):
56            :
57            blobs_sort.remove(blobs_sort[0])

58        points = [start, end]
59        for i in range(0, 2):
60            x1 = blobs_sort[i].x
61            y1 = blobs_sort[i].y
62            x2 = blobs_sort[i+1].x
63            y2 = blobs_sort[i+1].y
64            point = (x1 + xOffset, y1 + yOffset)
65            point2 = (x2 + xOffset, y2 + yOffset)
66            linelayer.line(point, point2, Color.BLUE, 3)
67            print blobs_sort[i].height()
68            print blobs_sort[i+1].height()

69
70    # Calc driving line
71    X = x1 + (y1 - y2)
72    Y = y1 + (x2 - x1)
73    dX = X-x1
74    dY = Y-y1
75    expdX = pow(dX, 2)
76    expdY = pow(dY, 2)
77    distance = math.sqrt(expdX + expdY)
78    distance = math.floor(distance)
79    X = X + (150/distance) * dX
80    Y = Y + (150/distance) * dY
    #print distance

```

```
82         point3 = (X + xOffset , Y + yOffset)
83         points[ i ] = point3
84         linelayer.line( point , point3 , Color.GREEN, 5)
85
86         linelayer.line( points[ start ] ,points[ end ] ,Color.RED
87                         , 10)
88
89     img.addDrawingLayer(linelayer)
90
91     if normaldisplay:
92         img.show()
93     else:
94         segmented.show()
```

..../SpurhaltungTest/Spurhaltung.py