TITEL

Inhaltsverzeichnis

1. Kurzfassung 4

2. Zielsetzung 5

3. Einleitung 6

4. Konzeptfindung 7

4.1. Teilfunktionanalyse 7

4.1.1. Morphologischer Kasten (Anhang) 7

4.2. Lösungskonzepte 7

4.2.1. Auswahl 7

4.2.2. Bewertung 8

4.2.3. Entschluss 8

4.3. Fazit 9

5. Lösungskonzept 10

6. Komponentenbeschreibung 11

6.1. Hebemechanismus und Greifer 11

6.2. Fahrgestell und Lenkung 12

6.3. Entsorgung 13

6.4. Sensoren 14

6.4.1. Farbsensor 14

6.4.2. Ultraschallsensor 15

6.5. Computer-Vision-Software 16

6.6. Energieversorgung 16

6.6.1. Akku 16

6.6.2. Akkuüberwachung 17

6.6.3. Spannungswandlung 17

6.7. Hardware und Software 18

6.7.1. Produktivaufbau 18

6.7.2. Kommunikation 19

6.7.3. Entwicklung 19

6.7.4. Deployment 20

6.7.5. Produktiver Betrieb 20

6.8. Motoren 21

6.8.1. Auswahl 21

6.8.2. Motorensteuerung und Regelung 21

6.8.3. Encoder 22

7. Projektmanagement und Planung 23

7.1. Organigram 23

7.2. Funktionsbeschreib 23

7.3. Planung 23

8. Schlussdiskussion 24

8.1. Kosten 24

8.2. Lesson Learned 24

8.3. Risiken 24

8.4. Offene Punkte 24

8.5. Ausblick auf PREN 2 24

8.6. Fazit 24

9. Abbildungsverzeichnis 25

10. Literaturverzeichnis 26

11. Anhang 27

# Kurzfassung

XXX

# Zielsetzung

XXX

# Einleitung

Für das Konzept von PREN 1 wurden nach der Teilfunktionenanalyse verschiedene Lösungsmöglichkeiten entwickelt und genauer untersucht. Dabei zeigten sich sowohl Vorteile und Nachteile verschiedener Teilfunktionslösungen als auch mögliche Gesamtkonzepte.

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

verbessern

# Konzeptfindung

## Teilfunktionanalyse

### Morphologischer Kasten (Anhang)

Der Morphologische Kasten 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. In den folgenden Tabellen werden die ausgewählten Varianten präsentiert und anschliessend bewertet.

## Lösungskonzepte

### 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.

Table

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

### Bewertung

Alle Teammitglieder konnten hatten als Aufgabe eine funtionsfähige Lösung zu präsentieren. Diese wurden dann anhand von bereits vorentschiedenen Gewichtungen dementsprechend Berechnet und in der unteren Tabelle aufgelistet.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Konzepte | | | | | | | | | | | |
| Zielkriterien | **Gewichtungs-Faktor** | **Variante 1** | | **Variante 2** | | **Variante 3** | | **Variante 4** | | **Variante 5** | |
|  |  | *Bewertung* | | *Bewertung* | | *Bewertung* | | *Bewertung* | | *Bewertung* | |
| Geschwindigkeit | 0.2 | 4 | 0.8 | 4 | 0.8 | 3 | 0.6 | 1 | 0.2 | 4 | 0.8 |
| Produktionsaufwand | 0.15 | 4 | 0.6 | 2 | 0.3 | 3 | 0.45 | 1 | 0.15 | 3 | 0.45 |
| Genauigkeit | 0.3 | 4 | 1.2 | 4 | 1.2 | 5 | 1.5 | 5 | 1.5 | 5 | 1.5 |
| Gewicht | 0.05 | 3 | 0.15 | 4 | 0.2 | 3 | 0.15 | 2 | 0.1 | 3 | 0.15 |
| Programmieraufwand | 0.25 | 3 | 0.75 | 3 | 0.75 | 2 | 0.5 | 2 | 0.5 | 2 | 0.5 |
| Energiebedarf | 0.05 | 3 | 0.15 | 3 | 0.15 | 3 | 0.15 | 2 | 0.1 | 2 | 0.1 |
| Summe | 1 |  | 3.65 |  | 3.4 |  | 3.35 |  | 2.55 |  | 3.5 |

|  |  |  |
| --- | --- | --- |
| Bewertung | | |
| Zielkriterien | **1,,** | **,,5** |
| Geschwindigkeit | sehr langsam | sehr schnell |
| Produktionsaufwand | hoch | gering |
| Genauigkeit | ungenau | genau |
| Gewicht | gross | klein |
| Programmieraufwand | hoch | simpel |
| Energiebedarf | hoch | tief |
| Summe | schlechte Lösung | gute Lösung |

### Entschluss

Die Bewertung der verschiedenen Varianten erfolgt in **Error! Reference source not found.**. Nach den Bewertungen von **Error! Reference source not found.** 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 weg. Die Variante 4 ist aufgrund des hohen Produktionsaufwandes und der geringen Geschwindigkeit ausgeschieden. Der Spindelkipper lässt sich schlecht mit der Raupe kombinieren und wäre deutlich langsamer, als wenn man nur eine Klappe öffnen muss, ebenso das Förderband anstelle des Hebelarms. Daher wurde Variante 1 als Konzept ausgewählt.

## Fazit

# Lösungskonzept

Ablaufdiagramm

Übersicht (über das Lösungskonzept)

Komponentendiagramm

# Komponentenbeschreibung

Das Lösungskonzept nach der Variante 1 besteht aus einem Zweipunktgreifer mit Hebelarm, einem Fahrgestell mit vier Rädern und einer Achsschenkellenkung und eine Klappe zur Entleerung. Die Energieversorgung wird durch ein LiPo Akku sichergestellt.

Als Rechenwerk werden ein RED Brick und ein Raspberry Pi 2 eingesetzt. Die Wahl der Sensoren liegt momentan bei einer Kamera und einem RGB-Sensor. Im Folgenden werden die verschiedenen Teilfunktionen beschrieben und die Wahl dieser Funktion begründet.

## Hebemechanismus und Greifer

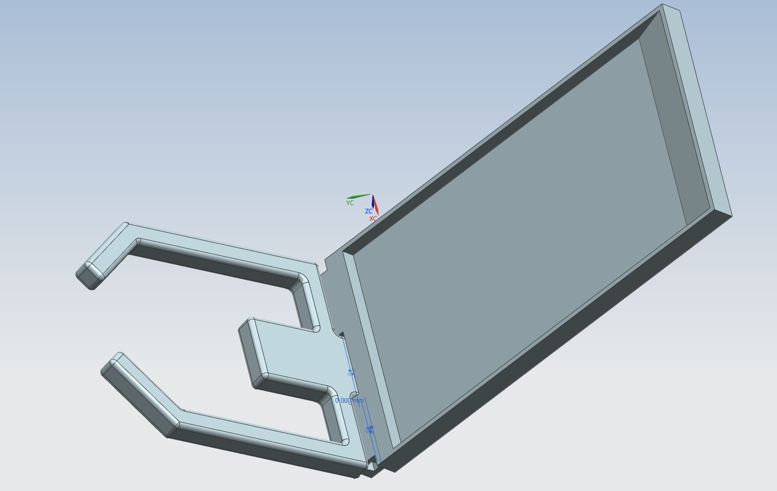
Um den Greifarm zu betätigen wurden vier Varianten in erwägung gezogen. Wir unterschieden zwischen einem zahnradgetriebenen Gelenk, einer Seilwinde, einem Schraubvorgang und einem Servomotor.

Von diesen vier Möglichkeiten wurde der Servomotor gewählt, da dieser bereits gefertigt ist und nur noch in den gewünschten Mechanismus eingebaut werden muss. Dazu kommt, dass bei industriellen Produkten die Aspekte wie Leistung, Genauigkeit und Grösse (kompakt) besser optimiert sind als bei selbsthergestellten Objekten.

Der Variante mit dem Servomotor liefert zudem das Mehrfache der benötigten Leistung, womit Risiken, wie das Verlieren des Containers beim heben, gleich ausgeschlossen werden können.

Wie in Abbildung 1 zu erkennen ist, handelt es sich um den Greiferarm. Die beiden roten Pfeile indizieren die Positionenen der beiden Servomotoren welche einerseits zum Antrieb des Arms (1) und andererseits (2) zum betätigen der Klemmen benützt werden. Zu erwähnen ist die fixe Knickung des Greifarms welche aufgrund der volumenspezifischen Produktanforderungen.

Abbildung : Provisorische Darstellung des Greiferarms



(1)

(2)

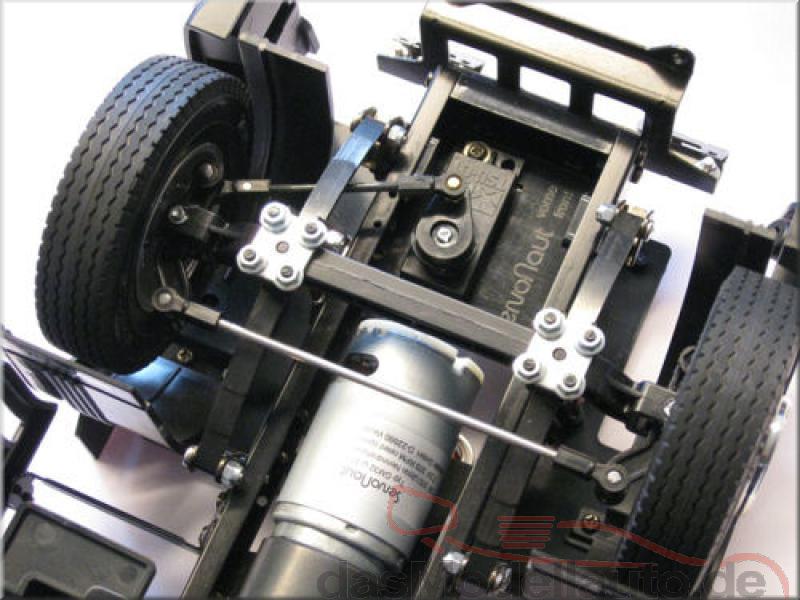
Es ist allerdings auch zu erwähnen, dass der Servomotor entsprechend für die Variante eines Greiferarms die beste ist. Dies würde gegebenenfalls anders aussehen bei einem Saugrohr, wo eine Seilwinde zu Einsatz gekommen wäre, oder einem Förderband mit einem Zahnrad und einer Seilwinde

## Fahrgestell und Lenkung

Das Lösungskonzept sieht ein Fahrzeug mit vier Rädern vor. Dieses Fahrgestell wurde gewählt, da es eine zentrierte Schwerpunktlage besitzt. Es hat eine grössere Stabilität beim Aufsammeln der Container als ein Fahrgestell mit nur drei Rädern. Vergleicht man die Stabilität beider Fahrzeuge in einer Hanglage, 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 als derjenige bei B. Aus diesen Gründen wurde für die Lenkung des autonomen Fahrzeuges eine Schenkellenkung gewählt, welche einfach anzusteuern, kostengünstig, genau und mechanisch einfach realisierbar ist.

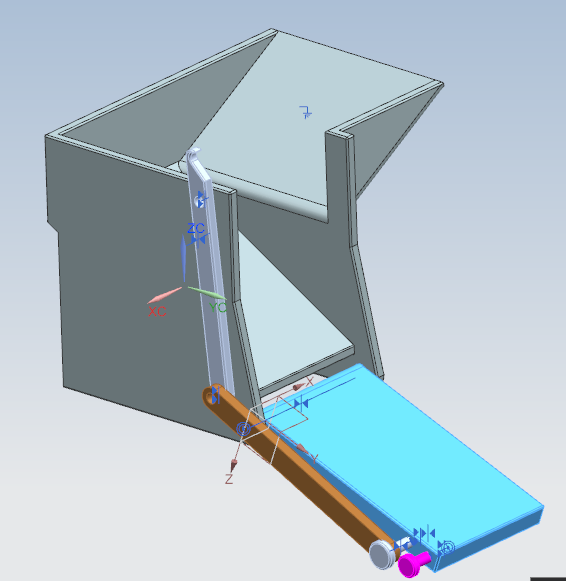


## Entsorgung

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, schliesst der Servomotor über einen Hebel und ein Scharnier die Klappe wieder.

Diese Variante erlaubt eine schnelle Entleerung. Im Gegensatz zu einer Gewindespindel ist dieses Konzept mit jedem Fahrwerk kombinierbar. Durch die Klappe wird sichergestellt, dass der Müll im Entsorgungsbecken landet und nicht zwischen Fahrzeug und Becken zu Boden fällt.

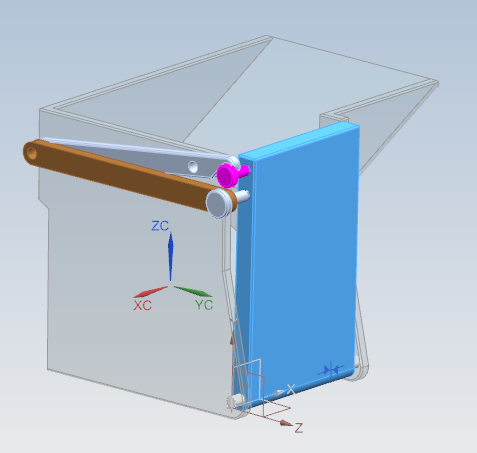
Abbildung : Mulde geöffnet



**Scharnier**

**Schräger Boden**

Abbildung : Mulde geschlossen



**Klappe**

**Müllauffang**

**Riegel**

**Servomotor**

**Hebel**

## Sensoren

Bei der Auswahl der Sensoren wurde der Fokus auf das Sortiment der Firma Tinkerforge gelegt. Das bedeutet, dass das Auslesen der Sensoren sich einfacher gestaltet, da die Firma eine gut dokumentierte API-Schnittstelle für verschiedene Programmiersprachen anbietet.

### Farbsensor

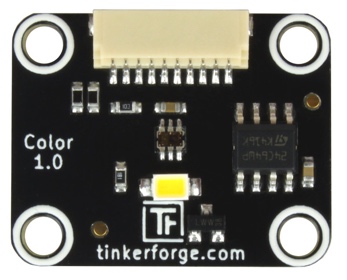


Figure : Color-Bricklet

Mit dem Farbsensor soll ausgewertet werden, ob es sich bei de

m Container, der sich auf der rechten Seite des Fahrzeuges befindet, um den „richtigen“ handelt. Der „richtige“ Container, muss von blauer oder grüner Farbe sein. Zudem wird mit dem Farbsensor ausgewertet, wann sich der Container vor dem Hebearm befindet.

|  |  |
| --- | --- |
| Eigenschaft | Wert |
| Sensor | TCS34725 |
| Stromverbrauch | 0,2mA (LED aus), 5mA (LED an) |
|  |  |
| 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) | 25 x 20 x 5mm (0,98 x 0,79 x 0,19") |
| Gewicht | 2g |

Table - Eigenschaften Color Bricklet

#### Funktionsweise

Der Farbsensor wird durch eine API gesteuert. Diese stellt zum Beispiel die Funktion „color\_get\_color“-Funktion zur Verfügung. Diese Funktion liefert die gemessene Farbe als RGB-Value zurück. Um bei verschiedenen Lichtverhältnissen noch gute Messwerte zu erhalten, können die Werte der Integration-Time und dem Gain-Value einstellen werden. Zudem enthält der Print eine SMD-LED die auch durch die API an- und abgeschaltet werden kann.

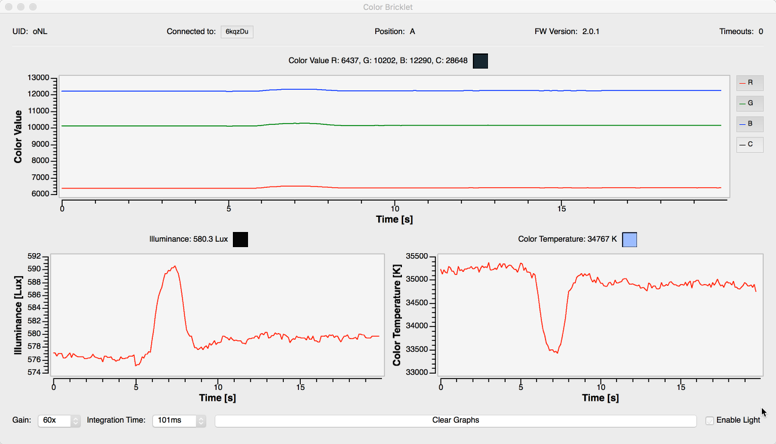


Figure : Brick Viewer mit Messwerten des Color-Bricklets

#### Tests

Beim Testen des Sensors, wurde festgestellt, dass bei Umgebungslicht der Sensor die Farbe auf einen Abstand von ca. 2cm noch genau anzeigt. Eine mögliche Lösung um die Reichweite des Sensors zu erhöhen besteht darin zusätzliche LEDs für die Beleuchtung zu verwenden.

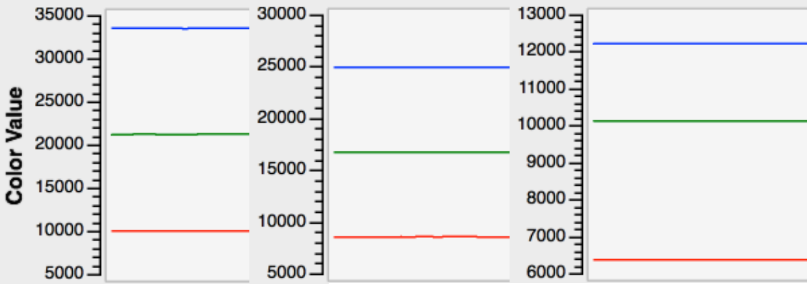


Figure : Messwerte des Color-Bricklets für verschiedene Distanzen. (2, 4, 6cm)

Bei den Tests wurde der Sensor mit den verschiedenfarbigen Containern auf immer grösser werdende Distanzen getestet. Die gemessenen Distanzen sind 2, 4 und 6cm. Für die Messung wurde der Color-Bricklet mit zwei zusätzlichen weissen LEDs ausgestattet. In der Abbildung #REF sind die verschiedenen Messwerte für die ändernde Entfernung aufgeführt.

### Ultraschallsensor



Figure : Das Distance US-Bricklet

Mit dem Ultraschallsensor soll erkannt werden

, ob sich ein Fahrzeug an der Kreuzung befindet. Mit dem Verwendeten Sensor können Entfernungen zwischen 2cm und 400cm messen. Die gemessene Entfernung wird als einheitenloser Wert ausgegeben, nicht in mm.

Dies 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.

|  |  |
| --- | --- |
| Eigenschaft | Wert |
| Sensor | HC-SR04 |
| Stromverbrauch | 8mA |
|  |  |
| Entfernungen | 2cm - 400cm, 12Bit Auflösung |
| Messwinkel | 15° |
| Aktualisierungsrate | 40Hz |
|  |  |
| Abmessungen (B x T x H) | 45 x 20 x 30mm (1,78 x 0,78 x 1,18") |
| Gewicht | 13g |

Table - Eigenschaften Ultraschalsensor

#### Funktionsweise

Der Ultraschallsensor kann wie der Farbsensor über eine API angesprochen und konfiguriert werden.

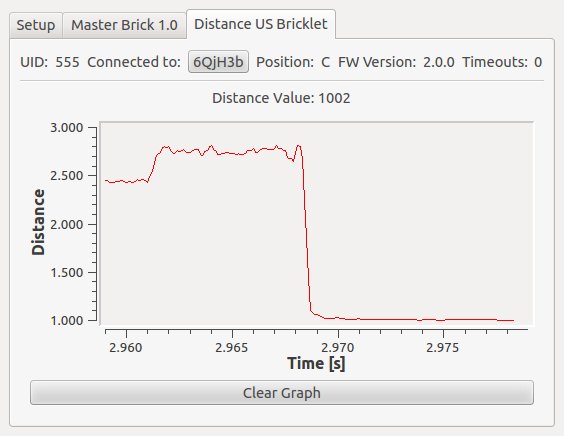


Figure : Messwerte des Ultraschallsensors

## Computer-Vision-Software

## Energieversorgung

### 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 angenommen.

|  |  |  |  |
| --- | --- | --- | --- |
| Komponente | Leistung [W] |  | Zeit t in h |
| Raspberry Pi 1 | 5 |  | 0.35 |
| Raspberry Pi 2 | 5 |  |  |
| Antriebsmotor | 60 |  |  |
| Servo Lenkung | 5 |  |  |
| Servo Greifer | 5 |  |  |
| Servo Heber | 5 |  |  |
| Servo Mulde | 5 |  |  |
| Sensoren | 5 |  |  |
| Webcam | 7 |  |  |
|  | **102** | **W** |  |
|  |  |  |  |
| Strom | 7.29 | A @14V Akkuspannung | |
|  |  |  |  |
| **Akkukapazität** | **2550.00** | **mA/h** |  |

Tabelle - Leistungen der Komponenten

Für die meisten Komponenten gab es nicht wirkliche eine Leistungsangabe. Meist musste diese selbst gerechnet 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 diesen Berechnungen und Überlegungen muss der Akku mindestens 20 Minuten halten.

### 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 OPV übernehmen. Die Schaltung ist im Bild unten ersichtlich. Mit dem Potentiometer kann die gewünschte Schaltspannung eingestellt werden. Sobald die Akkuspannung unter sinkt wird ein digitales High Signal am GPIO Eingang des Raspberry Pi angelegt. Dieser gibt den Alarm weiter an das zweite Raspberry Pi. Beide Rechenwerke führen noch Sicherungen durch und fahren herunter.



Abbildung - Schaltung Akkuüberwachung

### Spannungswandlung

Der Akku liefert bei voller Aufladung 16.8V. Bei spätestens 14.4V muss der Akku wieder aufgeladen werden. Somit ist die Eingangsspannung zwischen 14.4 bis 16.8V. Aus dieser Spannung muss die 5V Speisespannung für das Raspberry und die 7V Spannung für die Servos generiert werden. Dies wird mit Buck-Converter gemacht. Es gibt jeweils für die 5V Spannung und die 7V Spannungen eine Buck-Converter Schaltung.

LiPo Akku

14.4V – 16.8V

5V 4A

7V 5A

Raspberry1, Raspberry2, Encoder

Servo Bricklet

Sicherungen

14.4V – 16.8V

Spannug

Motor

Abbildung - Blockschaltbild Spannungen

Die Buck-Converter haben einen sehr hohen Wirkungsgrad und sind preiswert. Die Dimensionierung einer solchen Schalung 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 Schemas sind im Anhang ersichtlich.

## Hardware und Software

### Produktivaufbau

In Abbildung 4 sind alle Hardware-Module als rechteckige Kasten dargestellt. Die Software, welche auf den jeweiligen Hardware-Modulen laufen, sind als blaue Rechtecke gekennzeichnet. Die Grafik gibt einen Überblick über den Aufbau des IT-Systems des Roboters.

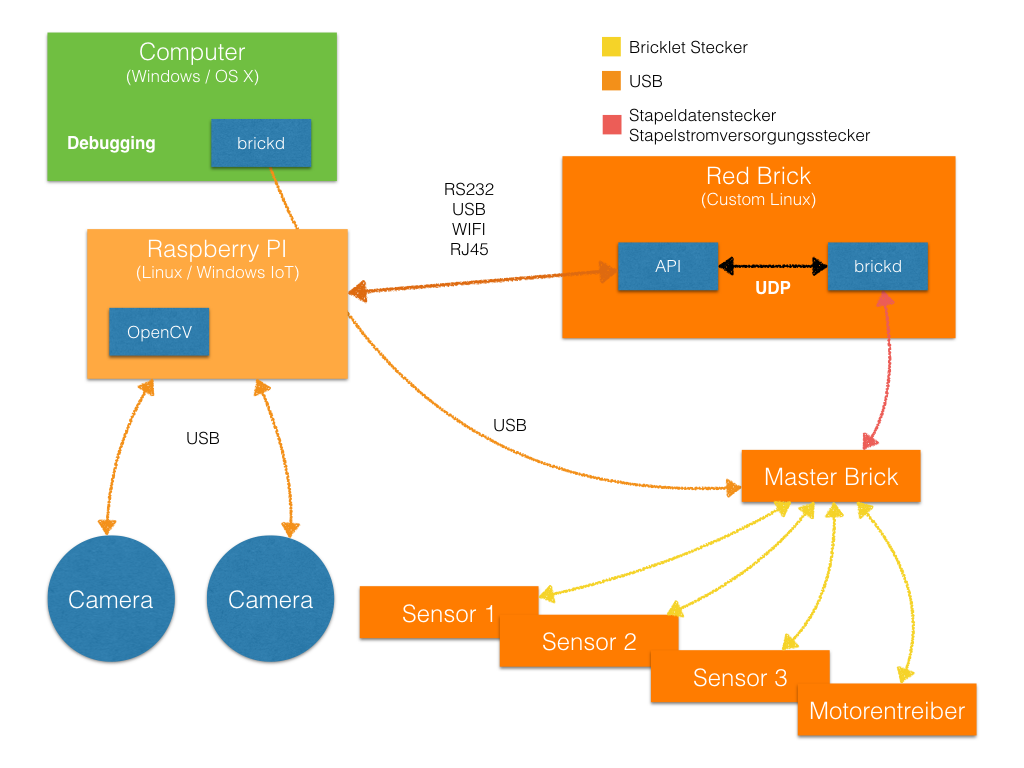


Abbildung : Übersicht der Hardware und Software für den Roboter

Für den produktiven Aufbau werden, wie in der Abbildung 4 dargestellt, zwei Computer benutzt. Das Raspberry Pi 2 wird benutzt, um die Bilder der angeschlossenen Kamera auszuwerten. Mit diesen Informationen wird das Fahrzeug gesteuert. Diese Steuerung wird durch die Software auf dem RED Brick abstrahiert. Dadurch müssen die Sensoren nicht direkt vom Raspberry Pi 2 angesteuert werden. Die beiden Software-Komponenten, welche auf dem RED Brick und dem Raspberry Pi 2 laufen, kommunizieren miteinander über eine TCP-Verbindung.

Am RED Brick sind alle Sensoren und Aktuatoren angeschlossen. Die Software (in Abbildung 4 als API bezeichnet) welche auf dem RED Brick läuft benutzt die API-Bindings welche von Tinkerforge zur Verfügung gestellt werden. Mit Hilfe dieser Bindings ist es möglich aus der API-Software heraus die Sensoren und Aktuatoren über einen Deamon (brickd) anzusprechen.

### Kommunikation

Um genügend Rechenleitung 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.

#### Protokoll

|  |  |
| --- | --- |
| Message ID | Message Type |
| Timestamp |  |
| Payload | |

**Message ID**: Inkrementierender Wert (Integer)

**Message Type**: ID für die Art der Nachricht (#REF Kapitel) (Integer)

**Time Stamp**: Wann die Nachricht abgeschickt wurde (Unix Time Stamp)

**Payload**: Je nach Art der Nachricht verschieden

#### Message Types

Der Message Type ist aus einer Zahl mit 4 Ziffern aufgebaut.

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

**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.

#### Messages

# Insert messages here

#### Payloads

# Insert Payloads here

### 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 5 aufgezeigt.

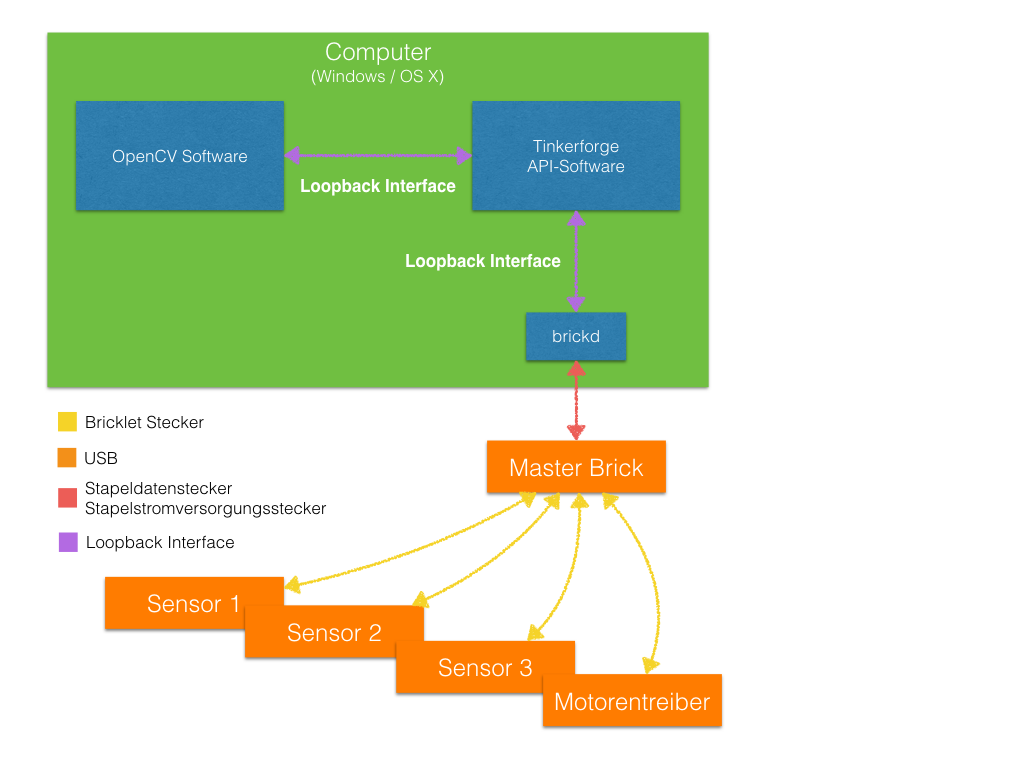


Abbildung : Aufbau der Hardware und Software während der Softwareentwicklung

Um die Software zu entwickeln, lässt man beide Software-Komponenten, welche normalerweise auf das Raspberry PI und den RED Brick aufgeteilt würden, auf dem selben Computer laufen. Um die TCP-Verbindung zwischen den beiden Komponenten zu simulieren, benutzt man das Loopback-Interface. Da die Bricks von Tinkerforge am Anfang dafür gedacht waren, vom Computer aus gesteuert zu werden, ist es kein Problem auch den brickd-Deamon auf dem Computer laufen zu lassen. Das in Abbildung 5 als “OpenCV Software” bezeichnete Rechteck, steht für die Softwarekomponente, welche die Auswertung der Kamerabilder vollzieht.

### Deployment

Für das Deployment soll eine Anleitung geschrieben werden, dass man alle Schritte, welche nötig sind, immer exakt gleich ausführt. Dazu soll es auch noch eine Software geben, welche während dem Betriebe des Roboters eine Übersicht der Sensorwerte anzeigt und ein Logfile schreiben kann, damit man allfällige Störungen im produktiven Betrieb nachvollziehen kann.

### Produktiver Betrieb

Im produktiven Betrieb läuft die Software jeweils auf dem Raspberry Pi 2 und dem RED Brick. Als Überwachung kann ein Computer via Wifi auf das Raspberry Pi 2 verbinden und in einer Software die Sensorwerte und andere Daten darstellen. Zusätzlich kann auf beiden Linuxboards eine Konsolenausgabe gemacht werden.

## Motoren

### Auswahl

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

|  |  |  |
| --- | --- | --- |
|  | Modelcraft RB350050 | Maxon RE30 |
| Bauart | DC Motor | DC Motor |
| Getriebe | 50:1 | 4.8:1 |
| Encoder | Nein | Ja, 512 Puls/U |
| Betriebsspannung | 12V | 24V |
| Leistung | 10W | 60W |
| Drehmoment | 5.39 Nm | 85.6mNm |
| Dimensionen | 106mm x 36mm | 125mm x 30mm |

Tabelle - 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 unser 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.

### Motorensteuerung und Regelung

Um mit dem Fahrzeug Positions- und Geschwindigkeitsgesteuert fahren zu können, wird eine Regelung benötigt. Dazu wird eine Sollgeschwindigkeit vorgegeben und die Geschwindigkeit mithilfe dem Encoder wieder eingelesen. Der Regelkreis ist in der untenstehenden Grafik sichtbar:

PID Regler

Störungen

UM

ωS

ωM

Encoder

Abbildung - Regelkreis

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

### Encoder

Der Encoder welcher am Maxon Motor befestigt ist liefert 512 Impulse/Umdrehung. Diese Pulse müssen alle aufgezeichnet werden, damit genau gefahren werden kann. Dies wären bei einer Motorendrehzahl von 600 Umdrehungen (bei geschätzter maximalen Geschwindigkeit) pro Sekunde 300‘000 Pulse pro Sekunde. Dies würde eine sehr grosse Abtastrate zur Folge haben. Deshalb wird ein Schnittstellen-IC verwendet. Dieses beinhaltet einen 32bit Counter, welche vom Encoder aufwärts/herunter gezählt wird. Nun kann mit dem Raspberry periodisch die Werte herausgelesen werden. Der Zeitpunkt dazu kann selber festgelegt werden, ohne dass Pulse verloren gehen.

# Projektmanagement und Planung

## Organigram

## Funktionsbeschreib

## Planung

# Schlussdiskussion

## Kosten

## Lesson Learned

## Risiken

## Offene Punkte

## Ausblick auf PREN 2

## 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 Mülleimer werden mit einem Greifer eingesammelt. Ein Hebelarm hebt den Mülleimer 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 ein Redbrick und ein Raspberry Pi 2 gesteuert und geregelt.

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

# Abbildungsverzeichnis

# Literaturverzeichnis

# Anhang

## Technischer Bericht Stereokamera

### 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.

### 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 mittgeteilt werden. Anschliessend werden beide Bilder der Kameras gleichgerichtet. Das bedeutet, dass man horizontale Linien über beide Bilder ziehen kann, die bei beiden Bildern dieselben Punkte durchläuft.

Abbildung :Bildpaar vor und nach Gleichrichtung (“slide\_21.jpg (960×720),” n.d.)

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.

Abbildung : Disparity map   
(“VWv6b.jpg (384×288),” n.d.)

### 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 dieses Bildpaar verwendet.

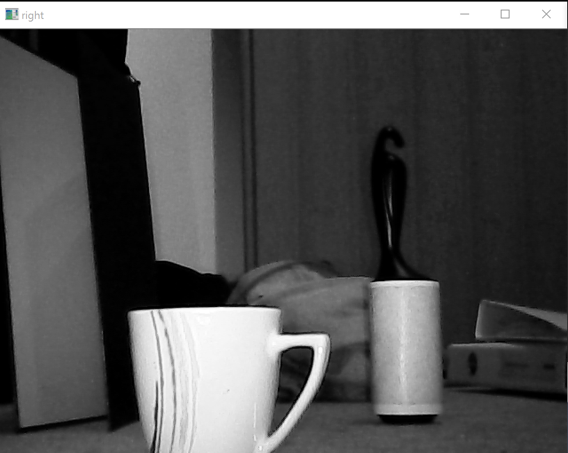
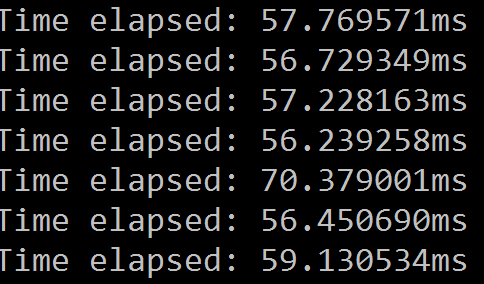


Abbildung : Versuchsaufbau Bild links

Abbildung : Versuchsaufbau Bild rechts

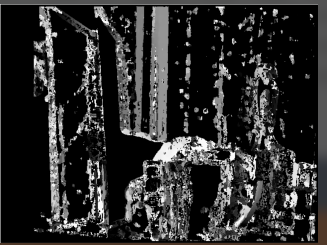
#### Versuch 1 – Block Matching-Algorithmus

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



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

Abbildung : Zeitaufwand StereoBM



Die berechnete Disparitymap (Abbildung (nr.)) erkennt zwar markante Punkte aber die Flächen dazwischen werden nicht ausgefüllt.

Abbildung : Disparity map StereoBM

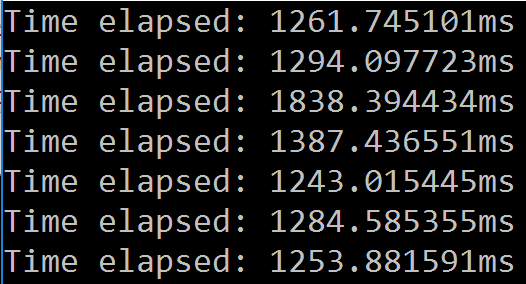
#### Versuch 2 – Semi Global Block Matching

Abbildung : Disparity map StereoSGBM



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

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



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

Abbildung : Zeitaufwand StereoSGBM

### 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.