BOSCH
Technik fürs Leben

DHBW
Duale Hochschule
Baden-Württemberg

Entwurf, entwicklung und validierung eines Light Distance and Ranging (LIDAR) Systems

Seminararbeit

des Studiengangs -todo-

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

-todo-

-todo-

Bearbeitungszeitraum	-todo-
Matrikelnummer, Kurs	-todo-, TEL16GR2
Ausbildungsfirma	Robert Bosch GmbH, -todo-
Betreuer	-todo-
Gutachter	

Duale Hochschule Baden Württemberg, STUTTGART
Ausbildungsbereich Technik
Fachrichtung Elektrotechnik / Informatik / Maschinenbau / Mechatronik

---

Bericht über die Ausbildung in der betrieblichen Ausbildungsstätte im ___ . Studienhalbjahr.

Name des Studierenden: _____

Studienjahrgang: _____

Einsatz in Abteilung: (sowohl Geschäftsbereich/Business-Unit/Abteilungsname ausgeschrieben als auch Abteilungs-Abk. entsprechend Outlook-Eintrag Betreuer)

Standort: _____

vom: _____ bis: _____

Thema: (Inhalt des Praktikums allgemeinverständlich abstrahiert, aussagefähig, prägnant, ohne Abkürzungen, wird als Tätigkeitsbeschreibung ins betriebliche Zeugnis übernommen, identisch zu Studentenportal)

Betreuer: _____

Stellungnahme des Betreuers:

Dieser Bericht wurde geprüft und ist sachlich und fachlich richtig.

Ort Datum _____ Abteilung, Unterschrift

Selbstständigkeitserklärung des Studenten

gemäß §5(3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 29.September 2015: Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Ort Datum _____ Unterschrift

Selbstständigkeitserklärung

Ich versichere hiermit, dass ich meine Seminararbeit mit dem Thema: *Entwurf, entwicklung und validierung eines LIDAR Systems* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Stuttgart, -todo-

_____

-todo-

Sperrvermerk

Die vorliegende Seminararbeit mit dem Titel

Entwurf, entwicklung und validierung eines LIDAR Systems

enthält unternehmensinterne bzw. vertrauliche Informationen der Robert Bosch GmbH, ist deshalb mit einem Sperrvermerk versehen und wird ausschließlich zu Prüfungszwecken am Studiengang -todo- der Dualen Hochschule Baden-Württemberg Stuttgart vorgelegt.

Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anders lautende Genehmigung der Ausbildungsstätte (Robert Bosch GmbH) vorliegt.

Stuttgart, -todo-

_____

-todo-

Abstract

TODO: deutscher Abstract....

Abstract

TODO: english abstract....

Inhaltsverzeichnis

Abkürzungsverzeichnis

BSP	Board Support Package
LIDAR	Light Distance and Ranging
ToF	Time of Flight
3D	Dreidimensional
CAD	Computer Aided Design
NEMA	National Electrical Manufacturers Association
GPIO	General Purpose Input Output

Abbildungsverzeichnis

Tabellenverzeichnis

Formelverzeichnis

Listings

1 Grundlagen Laserentferungsmessung

1.1 Lichtlaufzeitmessung

Das Grundprinzip der Lichtlaufzeitmessung oder auch Time of Flight (ToF) (Abbildung: 1.1), bezieht sich auf die Zeit, welche ein ausgesandter Lichtimpuls benötigt bis er wieder am Sender eintrifft.

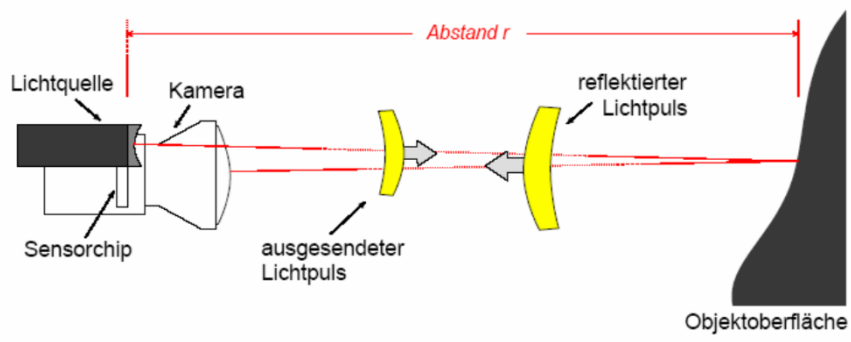

Abbildung 1.1: ToF Prinzip [2]

1.2 Phasenverschiebung

Das Phasenverschiebungsverfahren macht sich zu nutzen, dass bei einer ausgesandten Elektromagnetischen Welle die Phase immer größer wird bei steigender Entfernung. Durch Aussenden verschieden Frequentierter Wellen kann dann die Phasenverschiebung der Wellen bestimmt werden und daraus die Entfernung.

1.3 Triangulation

2 Code

Die gewählte Sprache in welcher die Steuerung realisiert ist, ist Python. Python wurde gewählt, da mittels dieser die General Purpose Input Outputs (GPIOs) des Raspberry Pi sehr einfach mittels einer Bibliothek ansteuerbar sind. Zudem ist Python eine sehr schnelle und weit verbreitete hochentwickelte Programmiersprache.
Bei der Erstellung des Codes, welcher das System steuert wurde von Anfang an eine Objektorientierte Vorgehensweiße gewählt, um eine möglichst Reibungslose und fortschrittliche Umsetzung zu realisieren.
Der gesamte Code wurde auf drei Dateien aufgeteilt, dies dient zum einen zur besseren Übersichtlichkeit, zum anderen erhielt jede Klasse eine eigene Datei.

2.1 Motor

Die erste Datei und Klasse beschäftigt sich mit der Ansteuerung der Schrittmotoren.
Sie benötigt zwei extra Bibliotheken (Listing 2.1). Die 'time' Bibliothek wird benötigt, um zwischen verschiedenen Befehlen ßchlafenßu können, sprich das Programm pausieren zu können. Die 'RPI.GPIO' Bibliothek wird benötigt um die GPIOs des Raspberry PI ansteuern zu können.

```
1 import time
2 import RPi.GPIO as GPIO
```

Listing 2.1: Bibliotheken der Motor Klasse

2.1.1 Konstruktor

Der Konstruktor der Klasse beschäftigt sich mit der Deklaration von Variablen und dem zuweisen der dem Konstruktor übergebenen Parameter.
Im Falle der Motor Klasse bekommt der Konstruktor sechs Übergabeparameter, wovon

allerdings ein Parameter ('self') eine Referenz auf das eigene Objekt ist.
Die restlichen übergebenen Parameter sind die GPIOs, welche für die Ansteuerung des Motortreibers benötigt werden.
Bei einem Blick auf den Code des Konstruktors (Listing 2.2) sieht man die übernahme der Übergabeparameter in Klasseneigene Variablen (Zeile 2-6). Anschließend wird die Kommunikationsrichtung der GPIOs festgelegt. In diesem Fall werden alle Pins als Ausgang benötigt.
Außerdem wird den GPIOs direkt ein Zustand zugewiesen, in diesem Fall ist die Konfiguration so, dass der Motor Treiber mit Achtelschritten arbeitet und den Motor gegen den Uhrzeigersinn drehen lässt.

```
1  def __init__(self, Step, Dir, MS1, MS2, MS3):
2      self.step = Step
3      self.dir = Dir
4      self.MS1 = MS1
5      self.MS2 = MS2
6      self.MS3 = MS3
7      GPIO.setup(self.step, GPIO.OUT)
8      GPIO.setup(self.dir, GPIO.OUT)
9      GPIO.setup(self.MS1, GPIO.OUT)
10     GPIO.setup(self.MS2, GPIO.OUT)
11     GPIO.setup(self.MS3, GPIO.OUT)
12     GPIO.output(self.step, GPIO.LOW)
13     GPIO.output(self.dir, GPIO.LOW)
14     GPIO.output(self.MS1, GPIO.HIGH)
15     GPIO.output(self.MS2, GPIO.HIGH)
16     GPIO.output(self.MS3, GPIO.LOW)
```

Listing 2.2: Konstruktor der Motor Klasse

2.1.2 Bewegen des Motors

Die Motor Klasse besitzt zudem noch eine Funktion, mittels welcher sich der jeweilige Motor bewegen lässt (Listing 2.3). In der Funktion wird zunächst die Drehrichtung gesetzt, und anschließend ein bzw. je nachdem wie viele Schritte gefordert werden ausführt. Um einen kompletten Schritt zu vollenden, wird der dafür vorgesehene Pin des Motortreibers Ein und wieder Aus geschaltet. Die Zeit zwischen diesen beiden Vorgängen kann über einen Übergabeparameter der Funktion eingestellt werden. Dies bestimmt direkt die Drehgeschwindigkeit des Motors.

```
1  def moveMotor(self, dir, step, speed):
2      if(dir):
3          GPIO.output(self.dir, GPIO.HIGH)
4      else:
5          GPIO.output(self.dir, GPIO.LOW)
6
7      i = 0
8      while i < step:
9          GPIO.output(self.step, GPIO.HIGH)
10         time.sleep(speed)
11         GPIO.output(self.step, GPIO.LOW)
12         time.sleep(speed)
13         i += 1
```

Listing 2.3: Funktion zum Bewegen des Motors

2.2 Lidar

Auch der Lidar Sensor hat eine eigene Datei sowie Klasse bekommen, dies soll dazu dienen, um später mehrere verschiedene Sensoren konfigurieren zu können und diese dann schell und einfach auswählen zu können.
Allerdings ist diese Klasse zum jetzigen Zeitpunkt noch nicht in der Finalen form, so dass mehrere Verschiedene Sensoren über eine Klasse Funktionieren. Bisher wurden nur die Grundsteine für den Konstruktor und die Benötigten Funktionen gelegt. Die Lidar Klasse benötigt bisher lediglich eine Bibliothek (Listing 2.4), mit welcher eine Serielle Verbindung erstellt werden kann.

```
1  import serial
```

Listing 2.4: Bibliothek der Lidar Klasse

2.2.1 Konstruktor und Variablen

Da die Klasse bisher noch nicht Vollständig ist, wurden einige Variablen vordefiniert und Werte fest zugewiesen, welche in einer Späteren Version durch den Konstruktor zugewiesen werden sollen (Listing 2.5).
Bisher initialisiert der Konstruktor lediglich die Serielle Verbindung zum LIDAR Sensor.

```
1 dist = 0
2 i2c = None
3 spi = None
4 uart = '/dev/ttyAMA0'
5 recievedData = False
6
7 def __init__(self):
8     self.ser = serial.Serial(self.uart, 115200, timeout=1)
```

Listing 2.5: Bibliothek der Lidar Klasse

2.3 Steuerung

Die dritte und letzte Datei beschäftigt sich mit der generellen Steuerung des Systems und dem Initialisieren und Aufrufen der Klassen und derer Funktionen.

3 Mechanik

3.1 Anforderungen

Damit ein 3D Abbild eines Raumes erstellt werden kann, ist es erforderlich, dass dieser möglichst leicht in mindestens zwei Achsen beweget werden kann. Deshalb muss im Rahmen dieses Projekts eine geeignete Mechanik entworfen werden, welche es ermöglicht, den Sensor auf zwei getrennt voneinander steuerbaren Achsen beliebig positionieren zu können. Damit eine solche Mechanik entworfen werden kann müssen zuerst einige Rahmenbedingungen geklärt werden. Beispielsweiße sollten die Motoren welche die Mechanik später antreiben vorher spezifiziert sein und die maximale Größe des Sensors bekannt sein. Natürlich sollte die Mechanik auch so entworfen werden, das diese dann auch in der Praxis umgesetzt werden kann.
Zur besseren Visualisierung und um genaue Zeichnungen anzufertigen wurde ein Computer Aided Design (CAD) Zeichenprogramm verwendet.

3.2 Entwurf

Der gesamte Aufbau lässt sich in drei große Teile unterteilen. Einen oberen Aufbau, welcher das Kippen des Sensors übernimmt und einen Motor halten muss. Die Basis, welche sich um 360° Drehen lassen soll. Und den Rahmen, welcher die Steuerung und den zweiten Motor enthält.

3.2.1 Oberer Aufbau

Für den oberen Aufbau der Mechanik gab es mehrere Vorraussetzungen. Zuerst soll die gesamte Mechanik so funktionieren, dass der Sensor möglichst genau im Ursprung der Dreh- und Kippachse liegt, um spätere komplizierte Umrechnungen der Punktewolke zu verhindern. Dazu soll der Aufbau möglichst leicht und klein sein, damit die Beschleunigte

Masse und die damit verbundenen Trägheitskräfte möglichst gering sind, damit unnötige Belastungen auf die Motoren vermieden werden.Außerdem müssen alle Leitungen, welche in dieser Aufbaute benötigt werden 360° Drehbar sein, weshalb ein sogenannter Schleifring unumgänglich ist.

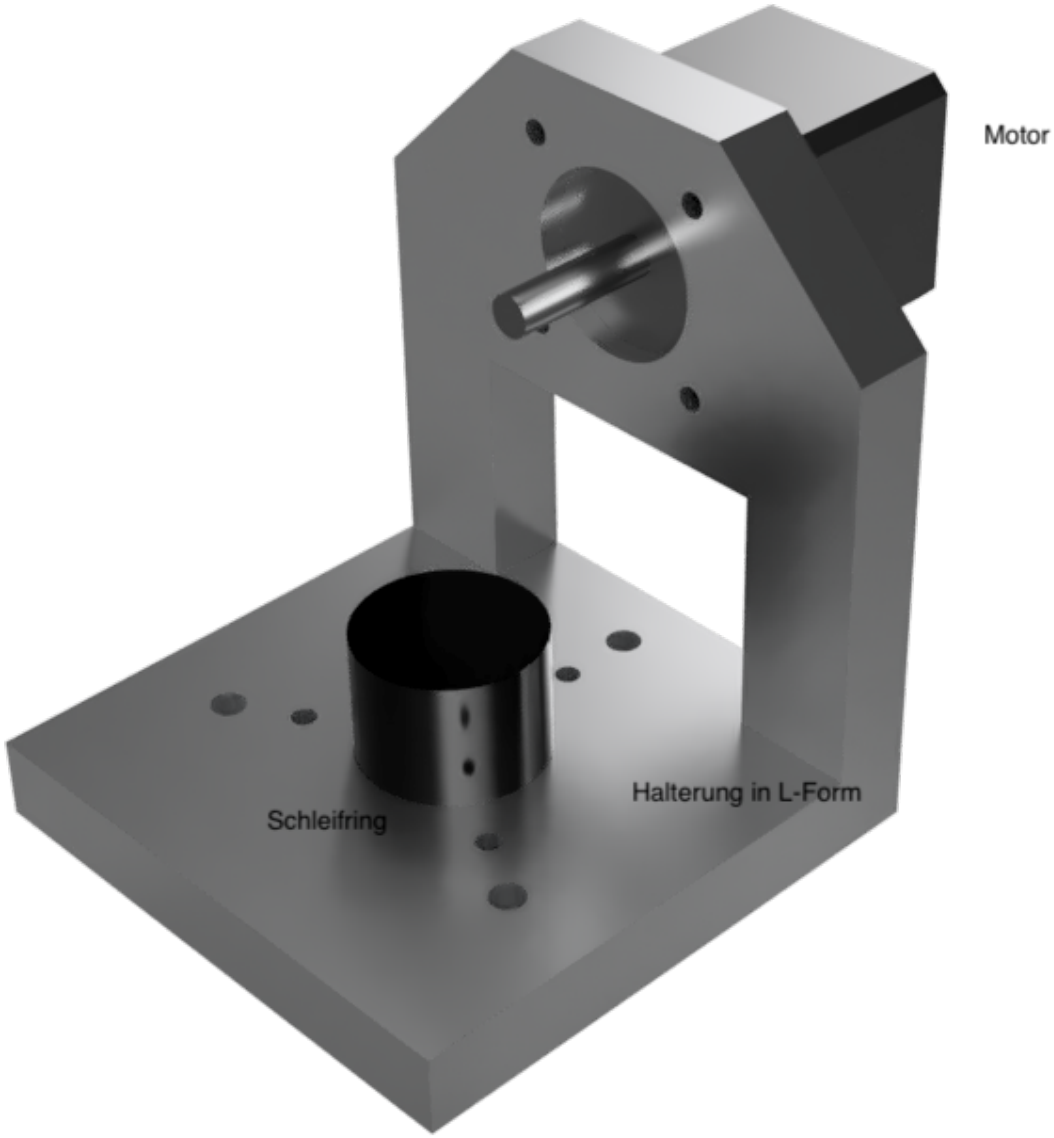

Abbildung 3.1: Oberer Aufbau der Mechanik

Der Motor welcher in Abbildung 3.1 zu sehen ist, ist von der National Electrical Manufacturers Association (NEMA) genormt und hat den Namen NEMA 11, die 11 verweist hierbei auf die Baugröße in diesem Fall 1,1" was ca. 28mm entspricht [3]. Außerdem ist in der Abbildung der Schleifring zu sehen, welcher später dazu dienen wird, dass alle Kabel des Oberen Aufbaus um 360° Drehbar sind.

Die Halterung in L-Form besteht aus zwei Teilen, welche aneinander Geschraubt werden. Ein horizontales Teil, die Grundplatte, welche den Schleifring und die Verbindung zu den weiteren Teilen sicherstellt. Und ein vertikales Teil, welches den NEMA 11 Motor in einer Vertiefung hält.
In Abbildung 3.1 fehlt allerdings ein weiteres Bauteil. Auf der Welle des Motors wird eine weitere Platte montiert, worauf später der LIDAR Sensor montiert wird. Zur besseren Übersicht wurde in der gezeigten Ansicht auf diese Platte verzichtet.

3.2.2 Basis

Die Basis stellt die Verbindung zwischen dem Oberen Aufbau und dem Rahmen dar. Die Basis ist die komplexeste Baugruppe der gesamten Mechanik, da sie den Antrieb und die Lagerung des Oberen Aufbaus übernimmt.

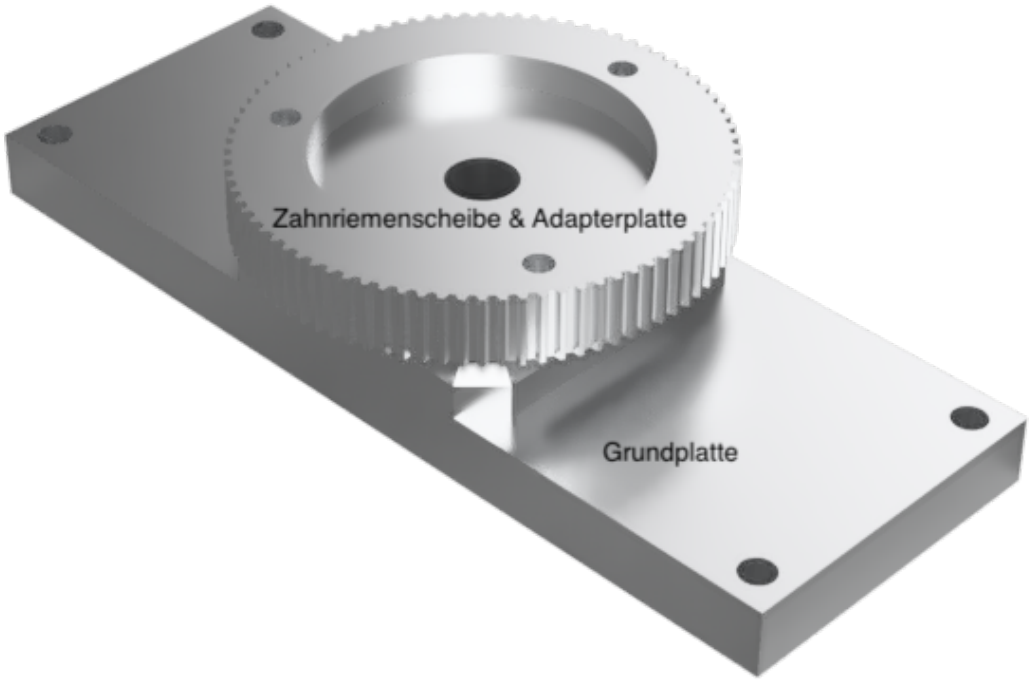

Abbildung 3.2: Basis der Mechanik

Um die Lagerung herzustellen wird ein großes Kugellager mit einem Innendurchmesser von $22mm$ in die Verbindungsplatte (Abbildung: 3.2) eingepresst. Der große Innendurchmesser des Kugellagers ist erforderlich, damit die Kabel durch dieses Hindurch geführt werden können. Der Antrieb des Oberen Aufbaus wird durch eine Zahnriemenscheibe hergestellt. Diese ist nach DIN 7721-2 T2,5 [1] entworfen da in dieser Anwendung eine große Anzahl an Zähnen gefordert ist, um eine höhere Winkelauflösung zu erhalten, wird diese Platte 3D gedruckt werden. Um die Zahnriemenscheibe mit dem Kugellager zu verbinden wird eine Adapterplatte verwendet, welche innen in das Kugellager eingepresst wird und anschießend mit Zahnriemenscheibe und Oberem Aufbau verschraubt. Diese Adapterplatte hat ein durchgängiges Loch um die Kabel heraus zu führen. Zudem sitzt die Adapterplatte vertieft in der Zahnriemenscheibe, um die Baugröße kompakt zu halten und einen Formschluss zu erzeugen.

3.2.3 Rahmen

Die dritte Baugruppe der Mechanik ist der Rahmen. Dieser dient hauptsächlich dazu eine stabile Befestigungsmöglichkeit für die Basis und den oberen Aufbau zu gewähren und die gesamte Elektronik zu ordnen. Zudem dient der Rahmen als Befestigungspunkt für den zweiten Motor. Der zweite Schrittmotor ist nach NEMA 17 genormt mit einem Außenmaß von ca $41mm$. Dieser wird über einen Zahnriementrieb den gesamten oberen Aufbau um 360° Drehen.

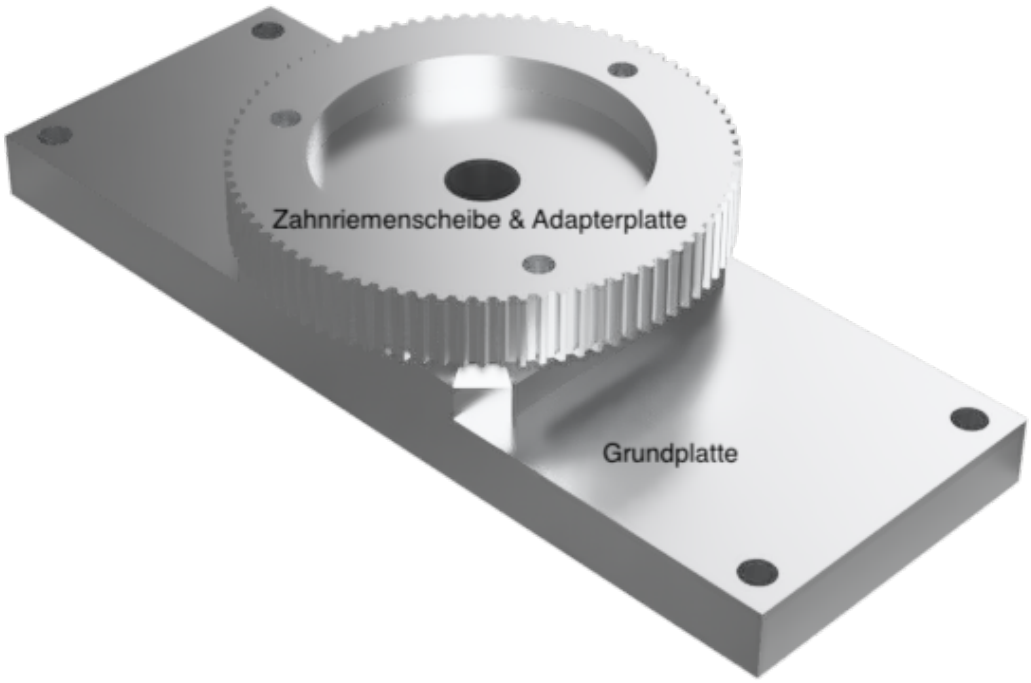

Abbildung 3.3: Motorhalterung

Um das obere Ende der Welle des zweiten Schrittmotors auf die selbe höhe wie die Oberkante der Zahnriemenscheibe zu bringen ist eine weitere Halterung erforderlich (Abbildung 3.3). Außerdem wird für den gesamten Rahmen ein Aluminiumprofil mit Nutensteinen verwendet. Dies ermöglicht unteranderem das herstellen der benötigten Spannung auf dem Riemen welcher das System dreht.

3.3 Umsetzung

Nachdem die Zeichnungen von allen Bauteilen angefertigt und überprüft wurden, konnte mit der Herstellung der einzelnen Bauteile begonnen werden. Fast alle selbst konstruierten Bauteile wurden aus Aluminium gefertigt, dabei wurde durch Fräsen, Drehen und Bohren die gewünschte Form erreicht. Lediglich eins der konstruierten Bauteile wurde mittels eines 3D-Druckers gefertigt, da ein herkömmlicher Fertigungsprozess sehr Zeitintensiv

und kompliziert gewesen wäre. Nach Fertigstellung aller Einzelteile kann die Mechanik Zusammengebaut werden und die Elektronik eingebracht werden.

Anhang

Literatur

[1] Ulrich Fischer u. a. *Tabellenbuch Metall*. Europa Lehrmittel, 2011.

[2] Nikolai Kutscher und Beate Mielke. *3D Kameras – basierend auf Lichtlaufzeitmessung*. 2005. URL: http://www.inf.fu-berlin.de/lehre/SS05/Autonome_Fahrzeuge/3dKameras.pdf.

[3] *NEMA ICS 16*. National Electrical Manufacturers Association, 2001. URL: https://www.nema.org/Standards/Pages/Motion-Position-Control-Motors-Controls-and-Feedback-Devices.aspx.