

Vorwort

Wir als Gruppe, bestehend aus zwei Informatikern Fachrichtung Applikationsentwicklung und einem Elektroniker, sind sehr technikbegeistert.

Wir wollen zusammen ein Produkt erschaffen, indem wir unsere beruflichen Fähigkeiten kombinieren können. Als Produkt haben wir uns für einen Druckroboter entschieden, da er unser Interesse an der Kunst und Kreativität miteinbezieht.

Wir bedanken uns herzlich bei allen, die uns während unserer Arbeit unterstützt haben.

Die Entwicklung eines Kunstroboters

Der Artomat

Interdisziplinäre Projektarbeit

Digitalisierung / Angewandte Mathematik / Informatik

Fachlehrperson: Andreas Schneider

Berufsfachschule BBB

Wiesenstrasse 32

CH-5400 Baden

Eingereicht von

David Wild, Jhg. 1999, Lernender bei libs

Leonie Born, Jhg. 2001, Lernende bei GIA Informatik AG

Nils Egger, Jhg. 2000, Lernender bei Avectris AG

2019

Abstract

Durch die Kunst erhalten Städte ihre Farben. Inspiration wird in Menschen entfacht und sowie Passanten nur gelegentlich einen Blick darauf werfen, so regt die Kunst doch vielmals zum Philosophieren an.

Für unsere interdisziplinäre Projektarbeit setzten wir uns mit der Frage auseinander, wie wir ein Produkt gestalten können, welches unsere Fähigkeiten im Bereich Hardware und Software kombiniert und sich gleichzeitig als fortschrittliches Produkt im Bereich der modernen Kunst zeigt.

Wir entwickelten gemeinsam einen Roboter, der mit Hilfe einer Halterung glatte Wände bedrucken kann. Dabei waren folgende Komponenten von Bedeutung: Ein Raspberry Pi mit Kamera, der den Pfad des Roboters überwacht und die Befehle für die Steuerung gibt, ein mobiler Roboter, der für das Drucken zuständig ist und an dem eine Spraydose befestigt ist, auf dessen Kommando dieser mit Farbe sprühen kann. Für den Bewegungsmechanismus sorgen zwei Rollen, die ebenfalls am Roboter montiert sind. Diese ziehen den Roboter mit Hilfe eines Seils quer über die Ebene.

Unser Projekt war ein Erfolg, wir haben den Druckprozess gemeistert und können mit unserem Roboter effektiv Bilder auf eine Wand bringen. Der Roboter kann im Freien eingesetzt werden und auch die Genauigkeit hat unsere Erwartungen übertroffen.

Inhaltsverzeichnis

1 Einleitung	5
2 Produktübersicht	6
3 Mechanik	10
4 Ansteuerung der Elektronik	19
5 Simulation	28
6 Steuerung	42
7 Schlusswort	47
8 Glossar	49
9 Abbildungen	55

1 Einleitung

1 Einleitung

Man kann sie überall sehen, bei Unterführungen, an Zügen, Brücken und an Hauswänden - Das bunte Graffiti. Doch diese moderne Kunst hat viele Gegner und das verwundert nicht. Denn die meisten Sprayereien sind illegal und laufen unter Vandalismus.

Nichtsdestotrotz gibt es Graffiti-Kunstwerke die auf Anordnung erstellt werden. Es handelt sich meistens um Gebäude, die aus der Masse herausstechen wollen, oder einfach einen Teil der Kunst wider-spiegeln wollen. Bei unserem Projekt wollen wir uns von illegalen Machenschaften distanzieren.

Es geht uns lediglich darum, das Drucken an einer Wand mit einem Roboter zu meistern und die benötigten Kenntnisse dazu zu erlernen.

Der Roboter soll sich auf einer senkrechten Wand bewegen und über ein Programm soll man ein Bild für den Druck auswählen können.

Die erste Schwierigkeit die wir antreffen, ist das Herumfahren auf einer senkrechten Ebene. Es ist ein Weg zu finden, der es erlaubt, mit ständig gleichbleibendem Abstand zur Wand und in gleichmässigen Tempo zu navigieren. Im Rahmen unseres Projekts entscheiden wir uns gegen das Drucken mit Magenta, Yellow, Blue, Schwarz und Weiss und wählen anstelle dessen, die Variante eines Monobildes. Der Roboter wurde jedoch so konzipiert, dass der Ausbau für mehrere Farbdosen gleichzeitig möglich ist. Dabei werden wir die Kontrastwerte eines Eingabebildes untersuchen und anpassen können.

Für den Pfad des Roboters, werden wir eine Simulation programmieren, an welcher wir verschiedene Variablen testen können. Das Programmieren des Roboters erfolgt mit Python. In der Theorie soll der Pfad über eine Kamera verfolgt und korrigiert werden. Dies erreichen wir durch die OpenSource Library OpenCV. Die Library ermöglicht es, gewisse Bildpunkte aus dem Sichtfeld der Kamera auszulesen und das Soll-Bild somit zu kopieren. Als Bildpunkte dienen UV-LEDs, die am Roboter selbst befestigt werden und ebenfalls an der Wand, um den Rahmen des Bildes festzulegen.

Unser Ziel ist es, den Roboter funktionsfähig zu machen - bedeutet, dass Eingabebild soll nach dem besprühen der Wand erkennbar sein -, die Library OpenCV kennenzulernen und das Arbeiten in einer Gruppe von verschiedenen Berufen zu koordinieren.

Roboter die Bilder an eine Wand drucken können und unseren Plänen ähnlich sind, wurden schon gebaut. Es gibt allerdings keine exakten Bauanleitungen, da die meisten Instanzen durch Hobbyinge-

2 Produktübersicht

nieure kreiert wurden. Im Gegensatz zu bereits existierenden Produkten, soll bei unserem Roboter die Grösse des Bildes einstellbar sein. Die Möglichkeiten reichen dabei von einer kleinen Tapete zu einer grösseren Hauswand.

Unser Lösungsansatz, den Roboter aktiv mit einer Kamera zu überwachen und zu steuern, wurde nach unserer Recherche noch nicht angegangen und wir bauen die Hardware selber zusammen.

Zu Drucktechnologien hat noch keiner von uns Erfahrung, wir haben jedoch Zugang zu Experten. Die Bilderkennung durch Software ist ebenfalls noch keiner von uns zuvor angegangen. Wir haben jedoch Erfahrung im Bereich Programmieren mit Java, C#, C++, JavaScript, Scripting und Python und sind somit bestens darauf vorbereitet, neue Technologien und Bibliotheken kennenzulernen.

Unser Druckroboter soll nicht punktgenau sein, er hat für die Zeit des Drucks keine Einschränkungen und wir erwarten lediglich den Druck mit einer Spraydose zu meistern, nicht mit mehreren gleichzeitig. Der Roboter wird jedoch ausbaufähig gebaut. Als Musskriterien soll der Druckkopf des Roboters sprayen und geortet werden können.

Mit unserem Produkt wollen wir alle, ob technik- oder kunstbegeistert ansprechen.

Als Ressourcen stehen uns 3D-Drucker, Computer, ein Labor in der Firma Hapa, Schul- und Kellerzimmer zur Verfügung. Für den Bau des Roboters benötigen wir neben Material für das Gehäuse, Motoren, Rollen, Seile, Schrauben und Muttern, elektronische Bauteile, Kabel und Raspberry Pi's.

2 Produktübersicht

Um unsere Arbeit übersichtlicher zu gestalten, haben wir unser Produkt, den Artomat, in Module eingeteilt. Jedes Modul hat eine bestimmte Aufgabe. Die Module sind mit einer Schnittstelle verbunden. In der Abbildung 1, sind die Module mit Buchstaben und die Schnittstellen mit Zahlen referenziert.

2.1 Module

2.1.1 Modul A: Simulation

Das Modul A in der Abbildung 1 ist eine Simulation von der Hardware (Modul C, D und E). Mit ihr kann das Modul B getestet werden. Dies ist notwendig, weil der Zugriff auf die Hardware (Modul C, D und E), nicht für alle Projektmitglieder möglich ist.

2 Produktübersicht

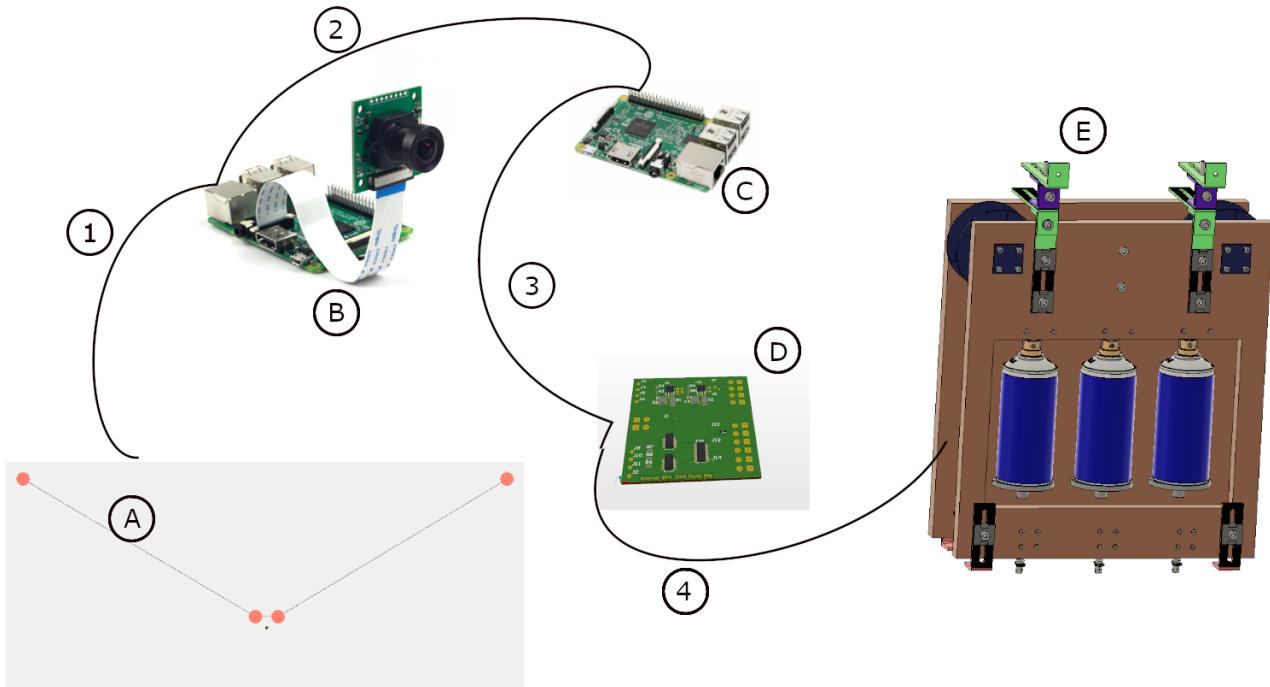


Abbildung 1: Übersicht Artomat

2.1.2 Modul B: Steuerung

Das Modul B in der Abbildung 1 ist die Steuerungseinheit. Modul B sagt dem Modul E, wohin es fahren soll. Für die Positionsbestimmung verwenden wir eine Kamera. Die Kamera erkennt das Modul E anhand von Markern, die hell im Ultraviolettbereich leuchten. Als Computer auf dem die Software läuft, verwenden wir einen Raspberry Pi. Als Kamera verwenden wir die Raspberry Pi camera.

2.1.3 Modul C: Controller Steuerung

Das Modul C in der Abbildung 1 verwandelt die Signale des Modul B in Signale um, mit denen man das Modul D ansteuern kann. Dafür wird ein Raspberry Pi verwendet. Ein Arduino / Mikroprozessor wäre für diese Anwendung geeigneter gewesen. Aber wir verwenden den Raspberry Pi weil wir den schon vor Projektbeginn hatten und der Remote-Zugriff sich einfach gestaltet.

2 Produktübersicht

2.1.4 Modul D: Artomat controller

Das Modul D in der Abbildung 1 soll gemäss Schnittstelle 3 ansteuerbar sein. Die Aufgabe dieses Modul ist es, die Aktoren von Modul E anzusteuern. Die Ansteuerung soll gemäss Schnittstelle 4 funktionieren.

2.1.5 Modul E: Artomat

Das Modul E in der Abbildung 1 soll sich mit zwei Elektromotoren an Seilen hochziehen können. Weiter soll es über einen Druckmechanismus verfügen, mit dem es drei verschiedenen Farben drucken kann. Die UV-Markierungen sind ebenfalls in diesem Modul enthalten.

2.2 Schnittstellen

Die Schnittstellen regeln, wie die Module miteinander agieren sollten.

2.2.1 Schnittstelle 1

Diese Schnittstelle überträgt den momentanen Stand vom Modul A zum Modul B. Beide Module laufen auf unabhängigen Threads, können jedoch auf einander zugreifen. Modul B greift so oft wie möglich auf Modul A zu, um die Kamera zu imitieren.

2.2.2 Schnittstelle 2

Diese Schnittstelle regelt den Datenfluss vom Modul B zum Modul C. Übertragen werden müssen die Steuersignale für die Motoren (genannt M1 und M2) und die drei Solenoiden (genannt S1, S2 und S3). Zusätzlich soll ein Notfallsignal und ein Zeitstempel übertragen werden.

2 Produktübersicht

Reihenfolge	Objekt	Datentyp und Wertebereich	Beschreibung
1	M1	int 10 bis -10	Geschwindigkeit
2	M2	int 10 bis -10	Geschwindigkeit
3	S1	int siehe sprüh tmax	Sprühzeit
4	S2	int siehe sprüh tmax	Sprühzeit
5	S3	int siehe sprüh tmax	Sprühzeit
6	SOS	bool	Notfall
7	Timestamp	time as float number	Absende Zeit

2.2.3 Schnittstelle 3

Die Schnittstelle 3 regelt den Datenfluss der Modulen C und D. Alle Signale sind active-high bei einer 3.3V Logik.

GPIO	Artomat Controller Pin	Beschreibung
GPIO 18	J3	M1 Richtung 1
GPIO 23	J4	M1 Richtung 2
GPIO 17	J5	M2 Richtung 1
GPIO 27	J6	M2 Richtung 2
GPIO 24	J3	S1
GPIO 25	J3	S2
GPIO 12	J3	S3

3 Mechanik

2.2.4 Schnittstelle 4

Die Schnittstelle 4 regelt die Ansteuerung / Verdrahtung der Aktoren.

Aktor	Artomat Controller Pin	Kabel Farbe
M1	J7	dunkel blau, grau
M2	J8	hell blau, violett
S1	J12	violett, weis
S2	J13	weis-blau, grau
S3	J14	hell blau, orange

3 Mechanik

Das Modul E ist unterteilt in Baugruppen. Bei einer Baugruppe handelt es sich um eine Anzahl verschiedene Teile, die gemeinsam eine Aufgabe erfüllen. Das Modul E enthält folgende Baugruppen: Seilzug und Motor, Druckmechanismus, Gehäuse, Seil und Seilzugbefestigung, sowie UV-Marker. Der Seilzug ist zuständig für den Antrieb. Er soll das ganze Gewicht des Artomaten nach oben ziehen können. Der Druckmechanismus ist zuständig für das Auftragen von Farbe. Er soll Farbpunkte sprühen können und einfach auszuwechseln sein, aber dennoch fest sitzen. Um alles befestigen zu können, muss ein Gehäuse konstruiert werden. Dieses soll robust sein und Einstellungen zur Balance aufweisen. Weiter soll die Distanz zur Wand einstellbar sein, um die Grösse des Druckpunktes zu verändern. An der Baugruppe Seil und Seilbefestigung hängt der Artomat. Für die Fertigung der Baugruppen stehen uns die Mittel 3D-Druck (SLA und FDM), Drehen, Fräsen und Bohren zur Verfügung. Für die Konstruktion wird die Software Autodesk Fusion 360 gewählt. Zum 3D-Drucker muss noch erwähnt werden, dass unser Bottompad, auf dem gedruckt wird, sich nicht zuverlässig erwärmt. Dies erzeugt Spannungen im Druck oder sogar erheblichen Verformungen, welche die Funktionsweise des Druckprodukts beeinträchtigen können.

3 Mechanik

3.1 Seilzug und Motor

Die Baugruppe Seilzug und Motor übersetzt die Kraft des Motors auf das Gehäuse und auf das Seil. Es besteht aus den Teilen Motor, Achse, Rolle und Seilführung. Die Kraftübertragung auf das Seil geschieht über eine Seilrolle. Die Seilrolle wird im Innenraum des Gehäuses auf einer Achse befestigt. Das Gewicht des Artomaten wird über zwei Kugellager auf die Achse übertragen. Der Motor wird auf der Aussenseite des Gehäuses befestigt und überträgt die Drehbewegung direkt auf die Achse. Entscheidend für die Zugkraft des Seilzuges sind der Durchmesser der Seilrolle, das Drehmoment des Motors und das Zuggewicht. Der Durchmesser des Seiles spielt auch eine Rolle, diesen haben wir aber vernachlässigt.

3.1.1 Motor

Um das Drehmoment des Motors zu berechnen, haben wir einen Seilrollendurchmesser von 8 cm angenommen. Um das Gewicht zu definieren, wogen wir eine volle 400 ml Buntsprühlackdose. Das Gewicht dieser Dose liegt bei 350 g. Bei den Berechnungen haben wir das Gewicht der Dose auf 400g aufgerundet. Für den Rest des Gehäuses haben wir mit einem Schätzwert von 1.5 kg gerechnet. Das ergibt:

$$m_{\text{tot}} = n_{\text{Dose}} * m_{\text{Dose}} + m_{\text{Rest}} = 3 * 0.4\text{kg} + 1.5\text{kg} = 2.7\text{kg} \quad (1)$$

Um auf der sicheren Seite zu bleiben, rechneten wir mit 5 kg. Das Drehmoment lässt sich wie folgt berechnen.

$$M = \frac{d}{2} * m * a = \frac{0.08\text{m}}{2} * 5\text{kg} * 9,81\text{ m/s}^2 = 1,962\text{ N m} \quad (2)$$

Entschieden haben wir uns für den Motor 33GN2738-132-GV-5 312:1 von IGARASHI MOTOREN GMBH.

Aus der Kennlinie der Abbildung 2 entnehmen wir eine Umdrehungszahl von 12.5 pro Minute bei 2 N m. In der Wirklichkeit sind wir von diesen Werten abgekommen, wie im Unterkapitel Rolle darauf eingegangen wird 3.1.3.

3 Mechanik

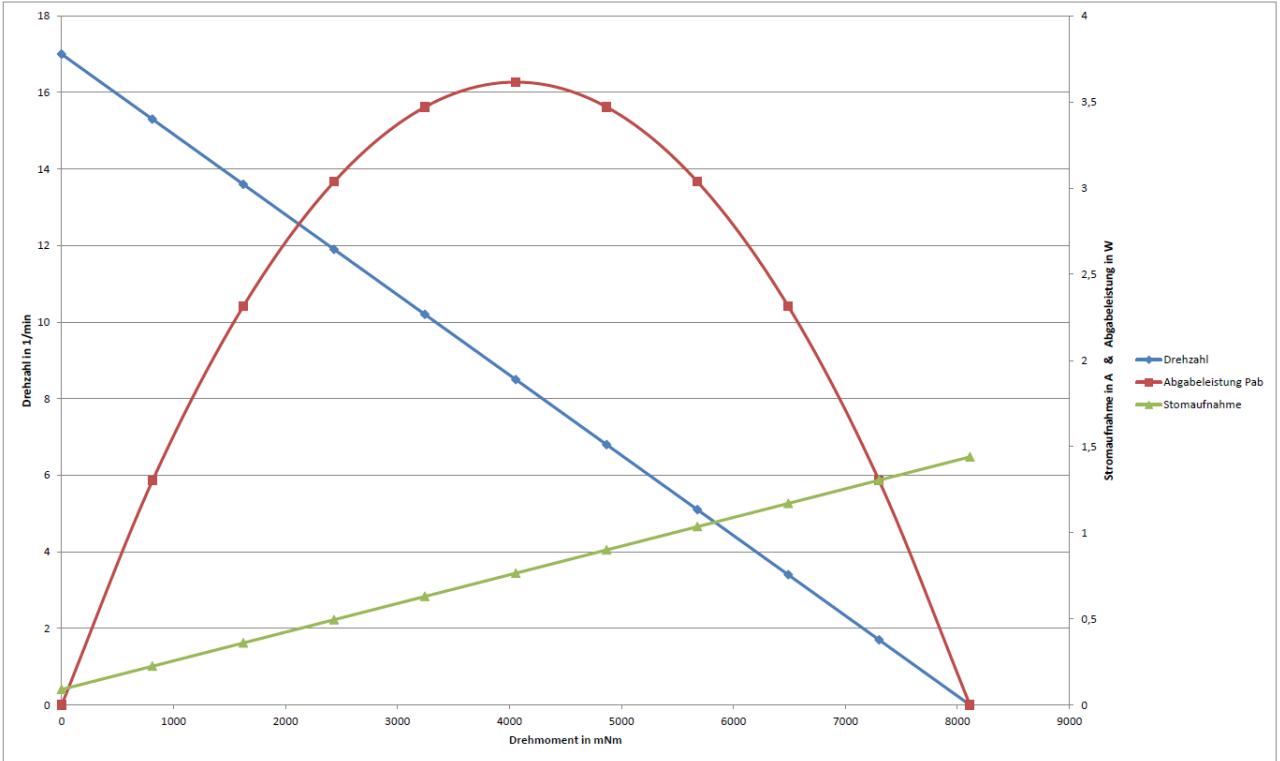


Abbildung 2: Kennlinie 33GN2738-132-GV-5 312:1o.V. o.D.(a)

3.1.2 Achse

Die Achse überträgt die Drehbewegung des Motors auf die Seilrolle. Das Gewicht wird mittels Kugellager auf das Gehäuse übertragen. Diese Teil-Baugruppe beinhaltet die Befestigung des Motoren, die Kupplung des Motoren zur Achse und die Befestigung der Achse.

Motorbefestigung Die Motorbefestigung (Abbildung 3) bietet eine Halterung für den Motor. Dabei handelt es sich um einen, auf dem 3D-Drucker hergestellten Bauteil, der an die Wand geschraubt wird. In diesem Bauteil befindet sich ein Hohlraum, in dem später die Verbindung zur Achse stattfinden wird. Oberhalb dieses Hohlraumes befindet sich eine Einbuchtung für die Befestigung der Achse. Ein Spalt auf der Frontseite des Bauteils (Abbildung 3) fungiert als Öffnung für zwei Stellschrauben, mit denen die Achse fixiert wird. Der Motor wird dabei in die vorgesehene Öffnung geschraubt. In dieser Öffnung kann der Motor sich nicht mehr bewegen, weil der Dreipunkt an der Motorwelle exzentrisch im

3 Mechanik

Motor sitzt. Die Anschlusskontakte des Motors werden durch Kappen geschützt. Um den elektrischen Kontakt herzustellen, werden die Stecker der Stromversorgung in die Kappe eingeführt(Abbildung 4).

Kupplung Die Kupplung(Abbildung 5) vermittelt zwischen Motorwelle mit Durchmesser 5mm, auf die Achse mit Durchmesser 8mm. Zwei Schrauben fixieren die beiden Achsen. Da der 3D-Druck zu wenig stabil wäre, besteht die Kupplung aus Aluminium, die David Wild gedreht hat.

Befestigung Achse Die Achse läuft auf zwei Kugellagern im Inneren je einer Seitenwand. Um das Kugellager in der Wand zu halten, werden 3D gedruckte Plättchen an der Aussenseite der Wand befestigt. Damit die Achse sich seitlich nicht verschiebt, wird sie mit diesen Plättchen fixiert(vgl. Abbildung 6).

3.1.3 Rolle

Im Kapitel Rolle ist die Seilrolle und die Kupplung von Seilrolle zur Achse beschrieben. Wie dem Kapitel 3.1.1 zu entnehmen ist, gingen wir zuerst von einer Rolle mit 80 mm Durchmesser ausgegangen. Eine Rolle dieser Grösse auszudrucken, würde viel Zeit und hohe Kosten generieren. Da die Filamentrolle des 3D-Druckers in der Firma von David Wild fast die ideale Form hat, wollten wir nun diese verwenden. Für diese Filamentrolle musste noch eine Kopplung auf unsere Achse stattfinden, da der Innendurchmesser der Rolle 57 mm beträgt. Da der Seilrollendurchmesser mit 100 mm grösser, als ursprünglich angenommen wurde, musste überprüft werden, ob die Kraft des Motors ausreichend ist.

$$M = \frac{d}{2} * m * a = \frac{0.1\text{m}}{2} * 5\text{kg} * 9,81\text{ m/s}^2 = 2,4525\text{ N m} \quad (3)$$

Der Motor sollte gemäss Kennlinie der Abbildung 2 die Kraft von 2,5 N m liefern können. In der Realität konnte diese Annahme nicht bestätigt werden. Der Motor ist nicht stark genug. Um die Kraft zu verringern, musste der Durchmesser der Seilrollendurchmesser verringert werden. Die Übersetzungsrolle für die Filamentrolle hat einen Durchmesser von nur 57 mm und eignet sich somit. Es mussten lediglich die Zähne für die Fixierung der Filamentrolle entfernt werden. Damit das Seil trotzdem auf der Rolle bleibt, wurden Abdeckungen auf der Seite der Rolle angebracht (vlg. Abbildung 8).

3 Mechanik

$$M = \frac{d}{2} * m * a = \frac{0.057\text{m}}{2} * 5\text{kg} * 9,81\text{ m/s}^2 = 1,398\text{ N m} \quad (4)$$

Gemäss Berechnung 4 sollte nun das erforderliche Drehmoment 1,398 N m betragen. Das Gewicht konnte so gut gehoben werden, einfach mit hörbarer Mühe. Die Motoren scheinen ihrer Spezifikation nicht zu entsprechen. Bei solchen Kleinspannung DC Motoren ist es üblich, dass sie über ein Getriebe aus Plastik verfügen. Dieses ist bruchanfällig.

Um die Kraft von der Achse auf die Seilrolle zu koppeln, ist eine Achsenklemme als Bindeglied nötig. Diese Klemme wird mittels Schraube auf der Achse fixiert. Mit Nabe und Kerbe wird das Drehmoment der Achse auf die Seilrolle übertragen. In der Abbildung 9 ist die Nabe auf der Achsenklemme und auf Abbildung 13 die Kerbe auf der Seilrolle zu sehen. Da normaler ABS Kunststoff sich beim Festziehen der Schrauben schnell verformen würde, wird die Achsenklemme mit einem SLA 3D-Drucker gedruckt. Dieser drückt mit einem Kunstharsz, das härter als ABS ist. Trotzdem zerbrach die Klemme auch mit diesem Material (vgl. Abbildung 10). Bei der Bruchstelle war klar zu sehen, dass die Schwachstelle bei der Ecke der Mutterhalterung liegt. Dies lag daran, dass dort am wenigsten Material vorhanden ist und die Kraft sich in den Ecken an einem Punkt konzentriert. Um einen Bruch zu verhindern, wurde eine stärkere Version der Klemme entworfen. Diese ist grösser und zusätzlich, um die Kraft besser zu verteilen, wurden die Ecken der Mutterhalterung abgerundet (vgl. Abbildung 11). Das Brechen konnte so nicht verhindert werden, wie in der Abbildung 12 zu erkennen ist. Nach erneutem Bruch der Klemme fiel auch der SDA 3D-Drucker aus. Deshalb wurden die Achsenklemmen aus Aluminium gefertigt.

Auf den Rat eines Ingenieur der Lehrfirma von David Wild, verzichteten wir schlussendlich ganz auf eine Seilrolle, um das Risiko eines Getriebebruchs zu vermindern. Das Seil verläuft somit direkt auf der Achse. Der klar ersichtliche Nachteil von dem ist, dass die Geschwindigkeit mit dem Rollendurchmesser sinkt. Weiterhin nötig ist die Achsenklemme, mit der das Seil an der Achse befestigt wird.

3.1.4 Seilführung

Die Seilführung ist essenziell für die Balance des Artomaten. Mit der Seilführung wird die Schräglage kontrolliert. Die Seilführung muss verstellbar sein, weil sich der Schwerpunkt des Artomaten verändern kann. Zudem muss die Reibung auf ein Minimum reduziert werden. Um die Reibung zu minimieren,

3 Mechanik

wird das Seil von einem Flaschenzug geführt. Für die Positionierung wurde dieser auf einen Winkel geschraubt, wie in der Abbildung 14 zu erkennen ist.

3.2 Druckmechanismus

Für den Druck verwenden wir Buntsprühlackdosen. Dies hat den Vorteil, kostengünstig zu sein. Gedruckt wird, indem ein Solenoid eine Kraft auf den Druckkopf ausübt. Wichtig zu beachten ist, dass die Dose bei steigender Temperatur einen höheren Druck aufweist. Dieser ist zudem von der Füllmenge. Zudem muss die Spraydose in aufrechter Position stehen, da ansonsten kein Farbstoff durch das Steigrohr gelangen kann, wie auf der Abbildung 15 ersichtlich ist. Je nach Sprühdose kann es sein, dass sich das Treibgas nicht nur durch den Innendruck mit der Farbe mischt. Solche Sprühdosen haben eine Mischkugel; durch Schütteln der Dose vermischt sich das Gas mit dem Farbstoff. In unserem Fall wäre ein Schüttelmechanismus zu komplex, daher muss die Dose einfach zu entfernen sein, um sie schütteln zu können. Für den Druck muss der Sprühkopf eingedrückt werden. Da der Druckkopf sich auf dem Sprühkopf befindet, verändert sich der Druckpunkt in der Höhe. Wir vermuten, dass die Eindrücktiefe sich von Hersteller zu Hersteller unterscheiden, jedenfalls haben wir keine Norm gefunden. Wir gehen daher von einer Eindrücktiefe von 1.5 mm bis 8 mm aus. Im Falle von 8mm führt die Eindrücktiefe zu einer Ungenauigkeit. Um diese Ungenauigkeit zu umgehen, drückt der Solenoid nicht auf den Sprühkopf, sondern auf den Boden der Dose, wie in Abbildung 16 zu erkennen ist. Die Kraft des Solenoids wird dabei durch eine Feder verstärkt und über einen Teller auf die Dose übertragen. Der Sprühkopf wird vom Sprühkopfhalter fixiert. Der Teller und der Sprühkopfhalter wurde mit einem 3D-Drucker ausgedruckt.

3.2.1 Dimensionierung Solenoid

Die Druckkraft, die der Solenoid aufbringen muss, ergibt sich durch die Gewichtskraft der Dose und die Auslösekraft der Dose. Die Auslösekraft ist keine Konstante, da der Druck innerhalb der Dose ebenfalls variieren kann. Leider haben wir nicht die erforderlichen Werkzeuge um diese Kraft zu messen. Die Abbildung 17 zeigt unsere Schätzung der Kraft, in Abhängigkeit der Eindrücktiefe. Kraft und Eindrücktiefe sind Dimensionslos und gelten nur zur anschauungs- Zwecken. Ab der Eindrücktiefe von 2, öffnet sich das Ventil. Um Kosten zu sparen und Ungleichheiten, wie sie von Temperatur verursacht

3 Mechanik

werden, auszugleichen, wird die Kraft des Solenoiden von einer Feder verstrkt, die zwischen dem Teller und Solenoiden eingeklemmt wird. Unser Solenoid hat eine Kraft von 6 N bei einem Duty-cycle von 50% und 12 V. Zusammen mit den Federn, sollte dieser Solenoid genigend Kraft aufbringen.

3.2.2 Sprhkopfhalter

Das Wichtigste beim Sprhkopfhalter ist, das er senkrecht nach unten drckt. Der Sprhkopf kann einknicken, was zu einer falschen Positionierung des Druckpunktes fhren kann oder viel mehr Kraft braucht. Wie in der Abbildung 18 zu erkennen ist, ist der Sprhkopf optimiert fr eine menschliche Fingerkappe.

Der Sprhkopfhalter von der Abbildung 19 drckt auf den hchsten Punkt des Sprhkopfs und auf die blau markierte Flche aus der Abbildung 18.

Der Sprhkopfhalter wird mithilfe von Schrauben an der Gehusewand befestigt.

3.2.3 Teller

Der Teller muss das Gewicht und die Kraft, die fr das Sprhventil zum Sprhen geraucht wird aushalten. Da der Teller direkt auf dem Solenoid befestigt wird, braucht er eine Halterung, damit er sich nicht drehen kann. Zudem soll er nicht einfach herausfallen knnen.

Der Teller hat einen Durchmesser von 73 mm und einen inneren Halbkreis, der als Anschlag fungiert, mit einem Durchmesser von 70 mm hat wie auf der Abbildung 20 zu sehen ist. Unten hat es eine Kupplung fr den Solenoiden. Auf der Unterseite des Tellers werden zustzlich zwei Magnete angebracht. Die Polung der Magnete ist gleich. Auf der Unterseite befinden sich drei weitere Magnete, jedoch in entgegengesetzter Polung. Dies ergibt eine Einrastungszone. Wenn der Teller ber diesen Magneten schwebt, rastet er ein und kann sich nicht frei bewegen (vgl. Abbildung 21).

3.2.4 Drucktest

Der Druck funktioniert wie erwartet, ist aber von der Temperatur abhngig. Steigt die Temperatur so braucht es mehr Kraft um den Druckmechanismus auszulsen. In einem Versuch bei einer Temperatur von 30 °C brauchte man zwei Federn, bei einer Temperatur von 20 °C nur eine, die gegen den Teller drcken.

3.3 Gehäuse

Das Gehäuse fungiert als Grundgerüst für alle weiteren Baugruppen, mit Ausnahme der Baugruppe Seil und Seilbefestigung. Es soll genug gross sein, um Flexibilität für Anpassungen zuzulassen. Zusätzlich soll es leicht zu bearbeiten sein aber genügend Stabilität aufweisen.

Als Baustoff wird MDF gewählt. Dieser ist billig und stabil. Um Flexibilität und Anpassungen zuzulassen wird das Gehäuse leicht zugänglich konstruiert. Es werden zwei MDF Platten mithilfe von Distanzbolzen parallel befestigt.

Das Gehäuse soll Platz für den Seilzug sowie für den Druckmechanismus bieten. Die Seitenwand der Seilrolle hat dabei einen Durchmesser von 10 mm, der Druckmechanismus eine Höhe von 26 mm. Um genügend Platz zu haben wurden die Dimensionen 36 x 40 cm genommen. Um das schaukeln zu minimieren, wird der Druckmechanismus unten im Gehäuse platziert um ein tiefen Massenmittelpunkt zu erzielen. Der Druckmechanismus muss in der Aufrechtposition gehalten werden, damit bei geringem Füllstand der Dose die Farbe ins Steigrohr aus der Abbildung 15 gelangen kann.

Um zu verhindern, dass das Gehäuse in die Wand kracht und der Druckmechanismus ungleich grosse Punkte drückt, muss ein konstanter Abstand zur Wand eingehalten werden. Dafür dient der Wandabstandhalter. Wird das Seil direkt an die Wand geführt, wirkt das Gewicht des Artomaten als Pendel, wie in der Abbildung 22 zu sehen ist. Für den Artomaten bedeutet dies, dass er gegen die Wand gedrückt wird. Die Kraft, die gegen die Wand drückt berechnet man wie folgt:

$$F = F_G * \sin \alpha \quad (5)$$

F_G steht für die Gewichtskraft. Für unseren Artomaten heisst das, dass wenn wir hoch, also Richtung Seilbefestigung fahren, die Kraft, die gegen die Wand drückt, grösser wird. Wenn die Wandabstandhalter so angeordnet wird, wie auf der Abbildung 23 ersichtlich ist, so dreht aufgrund der Pendelkraft der Artomat ab. Diese Drehung ist unerwünscht, da sie den Druckpunkt verfälscht. Den Wandabstandhalter weiter aussen zu positionieren ist ebenfalls keine Option. Das Problem entsteht durch den bereits gedruckte Teil des Bildes. Die Abbildung 24 zeigt die Drucklinie. Alles unter dieser Drucklinie kann ein triefend nasses Bild sein. Durch das entsteht eine Einschränkung der Positionierung des Wandabstand-

3 Mechanik

halters. Aufgrund dieser Einschränkung muss der Wandabstandhalter so positioniert werden, wie auf der Abbildung 23 gezeigt. Um eine Drehbewegung zu vermeiden muss der Winkel α (vgl. Abbildung 22), reduziert werden. Dies wird bei der Seilbefestigung realisiert und somit auch in diesem Kapitel dokumentiert. Aus diesen Gründen haben wir den Wandabstandhalter oberhalb der Seilbefestigung montiert, wie auf der Abbildung 25 Alle Masse, Befestigungen und Positionierung sind gemäss 3D Modell des Artomaten dokumentiert.

3.4 Seil und Seilbefestigung

Das Seil und dessen Befestigung sind ein essenzieller Teil des Projektes. Am Seil wird sich der Artomat hochziehen, weshalb es genügend stark sein muss. Der Durchmesser des Seiles bestimmt zugleich den maximalen Druckrahmen. Dies, weil das Seil an der Seilrolle aufgewickelt wird; bei grossen Distanzen muss mehr Seil aufgewickelt werden, was den Durchmesser der Seilrolle vergrössert. Es wird aber eine Methode beschrieben, wie man dies umgehen könnte. Da der Artomat nicht schnell fahren wird, muss das Seil auch nicht elastisch sein (muss die Kraft nicht abfedern). Weil das Seil nicht fest installiert ist, hat D.Wild ein Seil genommen, das zuhause verfügbar war. Zur Befestigung des Seiles an der Wand wird ein Loch gebohrt und danach einen Hacken befestigt. Falls wir nicht bohren dürften, müssen wir Gewichtsklötzte oberhalb der Wand installieren, an dem wir den Hacken befestigen können. Dies geht aber nur, wenn sich oberhalb der Wand ein Dach ohne Hindernisse befindet. Falls beide Methoden nicht funktionierten, können wir einen Rahmen platzieren, der an die Wand gelehnt wird. Falls das Seil fast auf der Höhe der Seilbefestigung ist, kann dies beim einseitigen Hochfahren einen Drall auslösen. Dies ist nicht erwünscht, da der Druckpunkt aufgrund dieser Dralls verschoben wird. Um dies zu beheben, muss das Seil weiter weg von der Wand. Einfach geht dies mit einem Winkel und einer Stellplatte, wie in der Abbildung 26 zu sehen ist. Benutzt man statt einem Haken einen Ring, um das Seil zu befestigen, kann man das Seil an den Boden spannen. Während des Druckprozesses kann man so mehr Seil geben oder nehmen. Und so kann der Druckrahmen vergrössert werden.

3.5 UV-Marker

Die UV Marker dienen zur Ortung des Artomaten. Die UV Marker dienen als Referenzpunkte, mit denen der Bildrahmen und der Artomat von der Software geortet werden kann. Um bei den Markern

4 Ansteuerung der Elektronik

nicht durch die Grösse limitiert zu sein, müssen die Marker an der Wand mittels Akku betrieben sein. Als Akku fungieren zwei AAA Batterien, die in Serie geschaltet werden, um so das für LED notwendige U_F von 3 V zu erbringen. Das Ganze wird zusammengehalten über ein Batteriergehäuse, das an die Wand geklebt wird, wie in der Abbildung 27 gezeigt wird. Die Marker, die sich auf dem Artomaten befinden, werden in Löchern befestigt. Befestigt werden die Marker gemäss 3D-Modell. Beim Testen bemerkten wir, dass reflektierende Objekte in OpenCV die Erkennung stören. Um Reflexionen zu vermeiden, haben wir Metallgegenstände im Sichtfeld der Kamera mit schwarzer Farbe bedeckt.

4 Ansteuerung der Elektronik

Dieses Modul wird zeitgleich mit dem Modul E entwickelt. Damit das Modul E getestet werden kann, wird zuerst ein Prototyp (Version 1) der Elektronik erstellt. Sobald das Modul E funktioniert und der Schaltungsumfang bekannt ist, wird eine finale Version (Version 2) entwickelt, welches als PCB bestellt wird.

4.1 Version 1

Die Version 1 soll drei Aktoren ansteuern können. Die zwei Motoren und der Solenoid. Angesteuert werden soll es vom Raspberry Pi. Eine Geschwindigkeitsregulierung der Motoren soll möglich sein.

4.2 Ansteuerung der Motoren

Mit den GPIO's des Raspberry Pi, soll ein Motor links und rechts betrieben werden. Zusätzlich soll die Geschwindigkeit reguliert werden können. Die GPIO's des Raspberry Pi verfügen nur über 3.3V und können nur 16mA abgeben. Der Motor benötigt aber 12VDC und 1,44A Spitzenstrom. Zudem wird eine Freilaufdiode als Schutz vor dem Motorstopp für den Raspberry Pi benötigt. Die links-rechts Ansteuerung erfolgt mittels H-Brücke (siehe Abbildung 28). Die Geschwindigkeit wird mittels PWM (Puls-Weiten-Modulation) von der Software gesteuert.

4.2.1 Funktionsweise einer H-Brücke

Eine H-Brücke besteht aus zwei Halbbrücken. Eine Halbbrücke besteht aus zwei Schaltungselementen. Hier in der Abbildung 28 sind es zwei Schalter S1 und S2. Durch die Parallelschaltung von zwei Schaltern kann die H-Brücke in Abbildung 28 das Spannungspotenzial zwischen S1 und S2 von high zu low schalten, indem man die Schalter S1 und S2 schliesst/öffnet. Durch Schalten des entgegengesetzten Potenzials an der anderen Halbbrücke erzeugt man einen Stromfluss. Nun bewegt sich der Motor. Invertiert man die Schalterstellung auf beiden Seiten, so dreht sich der Motor in die entgegengesetzte Richtung. Setzt man die Schalter auf beiden Seiten auf die gleiche Stellung, so fliesst kein Strom, beziehungsweise der Motor steht. Werden beide Schalter geschlossen (S1 und S2) so entsteht ein Kurzschluss.

4.2.2 H-Brücke im Artomat Controllerboard V1

Was beim Schema der Abbildung 29 kompliziert aussieht, ist im wesentlichen die selbe Schaltung von Abbildung 28, einfach nur als Halbbrücke abgebildet. Die Potenzialschaltung übernimmt nun statt S1 der Transistor T5 und statt S2, die Transistoren T3 und T7. Wo bei Abbildung 28 die Verbindung zum Motor war, ist beim Schema der Abbildung 29 die Verbindung bei M1_L. Die Transistoren T3 und T4 bilden eine Darlingtonsschaltung. Mit einer Darlingtonsschaltung kann man einen grösseren Strom schaltenMotayed und Mohammad 2001. Beim Transistor T5 handelt sich es um ein Transistor des Types PNP, der PNP Transistor hat die invertierte Funktionsweise, wie die des NPN Types Schaeerer 2019a. Das heisst, ist das Spannungspotenzial an der Base des Transistors High, so schaltet der NPN Transistor aber der PNP Transistor nicht. Ist aber das Spannungspotenzial Low so schaltet der PNP Transistor und der NPN Transistor nicht. Die Transistoren T1, T3 und T5 sind vom Typ NPN. Mit dem Transistor T1 schaltet man den Transistor T3, T5 und T7. Weil der Transistor T5 entgegengesetzt schaltet, kann es nicht zu einem Kurzschluss kommen, unter der Bedingung, dass die Transistoren richtig dimensioniert worden sind. Bei D1 und D2 handelt es sich um Freilaufdioden. Die sind wichtig um die Schaltung vor dem Überstrom, bei Motorstopp zu schützenEngelhardt 2016. U1A ist ein Optokoppler, der zur Galvanischetrennung dient. Die Galvanischetrennung⁸ ist notwendig, um den SBC zu schützen. Besonders, weil die Schaltung als Laboraufbau realisiert wird.

4.2.3 Dimensionierungen des Artomat Controllerboard v1

Für die Spezifikationen des Raspberry Pi's und der Motoren siehe Abschnitt 4.2. Transistor T5, T6, T13 und T14 wird der PNP Typ BDW54B eingesetzt. Dieser hat ein h_{fe} von 750 und ein U_{EB} von 2.5 Vo.V. 2002.

Für Transistor T1, T2, T3, T4, T11 und T12 wird der NPN Typ BC237 verwendet. Dieser hat ein h_{fe} von 400, ein U_{BE} von 0.8 V und ein U_{CE} von 0.2 V. Bei einem I_c von 10 mA, soll im gesättigten Zustand I_b 0.5 mA seino.V. 2001.

Für Transistor T7, T8, T15 und T16 wird der NPN Typ BD243B verwendet. Dieser hat ein h_{fe} von 30¹ und ein U_{BE} von 2 Vo.V. 2000.

Für den Optokoppler U1 wird der Typ ILD207 verwendet. Dieser hat ein U_F von 1.3 V, ein U_{CE} von 0.4 V und ein I_F von 10 mA. 2015.

Für alle Transistor Berechnungen wird ein Übersteuerungsfaktor (\ddot{U}) von 5 angenommen.

$$R1 = \frac{U_{Raspi} - U_F}{I_F} = \frac{3.3V - 1.3V}{10mA} = 200k\Omega \quad (6)$$

$$R9 = \frac{U1 - U_{EB}@T5 - U_{CE}@T1}{\frac{I_c}{h_{FE}*\ddot{U}}} = \frac{12V - 2.5 - 0.2V}{\frac{1.6A}{750*5}} = 61.25k\Omega \quad (7)$$

Nach E12 würde dies dann 56 kΩ geben. Wie aus dem Laborbericht Wild 2019c, der auf der Abbildung 30 zu erkennen ist, befindet sich der Transistor T5 mit einem Widerstand von 56 kΩ nicht in der Sättigung, daher wird ein Widerstand von 8,2 kΩ eingesetzt, um eine geringere V_{EC} Sättigungsspannung zu erzielen.

$$R7 = \frac{U1 - U_{BE}@T3 - U_{BE}@T7}{\frac{I_c}{h_{FE}@T3*h_{FE}@T7*\ddot{U}}} \quad (8)$$

Für R7 wird 150k gewählt, für R5 56k. Der Widerstand R1 wird wie folgt berechnet.

$$R1 = \frac{U_{raspi} - U_{arg}@ILD207}{I} = \frac{3,3V - 1,3V}{10mA} = 200\Omega \quad (9)$$

¹Irreführende Angaben von Hersteller, es wird das Minimum angenommen

4 Ansteuerung der Elektronik

$$R3 = \frac{U1 - U_{CE}@ILD207 - U_{BE}@BC273}{I_{sat}@BC237} = \frac{12V - 0.8V - 0.4V}{0.5mA} = 21,6\text{ k}\Omega \quad (10)$$

Der Widerstand R3 wird nach E12 auf 22kΩ dimensioniert.

Die Abbildung 31 zeigt eine Messung der Schaltung von der Abbildung 29. CH1 misst V_{cc} und CH4 misst den Gesamtstrom. Die Spannung wurde dabei manuell am Labornetzgerät verstellt. Dabei ist ersichtlich, dass bei der Erhöhung der Spannung zu einem signifikanten Stromanstieg führt. Die Halbbrücke kann sich mit diesen Dimensionen selbst auslösen. Dies kommt von einem ungewollten Spannungsteiler. Dieser wird gebildet von T5 zu R9 + R7 zu T3 + T7. Sobald die 10 V an V_{cc} überschritten werden, reicht die Spannung an der Basis des Transistors T3 aus, um diesen zu schalten. In diesem Fall schalten beide Transistoren und es führt zu einem Kurzschluss. Um diesen Kurzschluss zu vermeiden, könnte der Spannungsteiler von R5 zu R7 anders dimensioniert werden oder T5 könnte ebenfalls als Darlingtontransistor betrieben werden.

Überschreitet man die 10 V an V_{cc} nicht, so funktioniert die Schaltung ordnungsgemäß, wie die Abbildung 32 zeigt. CH1 wurde an M1_L angeschlossen, CH2 auf J3, CH3 auf J4 und CH4 misst den Strom, der durch den Motor fließt. Der Motor befindet sich im Leerlauf.

4.3 Ansteuerung der Solenoiden

Mit den GPIO's (general purpose input output) des Raspberry Pi, soll ein Solenoid, auch Hubmagnet genannt, gesteuert werden. Über die Limitationen des Raspberry Pi haben wir bereits im Abschnitt 4.2 erfahren. Von dem Datenblatt des Herstellerso.V. 2017a entnehme ich, dass wir ungefähr 250mA benötigen, um den Solenoid zu steuern. Jedoch entnehmen wir auch aus dem Datenblatt, dass die Spule ungefähr einen Widerstand von 12Ω haben soll. Bei einer Betriebsspannung von 12V können die 250mA nicht stimmen. Für die Berechnungen werden wir daher von 1.2A ausgehen.

4.3.1 Ansteuerung der Solenoiden im Artomat Controllerboard V1

Die Abbildung 33 zeigt einen Ausschnitt aus dem Schema Artomat controller v1. Der Solenoid wird mit dem NPN Transistor T17 geschaltet. Um mehr Strom schalten zu können, bildet T17 zusammen mit dem Fototransistor im Optokoppler U5A eine Darlingtonsschaltung. Als Überstromschutz vor dem Solenoid dient die Diode D5. Der Raspberry Pi wird durch den Optokoppler U5A geschützt.

4 Ansteuerung der Elektronik

Die Abbildung 34 misst die Schaltung der Abbildung 33 unter der Betriebsspannung 10V, CH1 misst an J1 Pin 2 und CH4 misst den Strom, der vom Stecker J1 vom Pin 1 zu Pin 2 fliest. Anhand der Messresultate entnehme ich, das die Schaltung wie erwartet funktioniert.

4.4 Stromversorgung

Für die Versorgung des Artomatcontrollers müssen zwei Betriebsspannungen zur Verfügung stehen, 12V für die Aktoren und 5V für den Raspberry Pi.

Da eine selbstgebaute H-Brücke eingesetzt wird, muss für die 12V Spannung eine Strombegrenzungsschaltung eingesetzt werden, um einen Kurzschluss zu vermeiden.

4.4.1 12V Versorgung v1

In der Abbildung 35 ist eine Kollektorschaltung zu sehen. Eine Kollektorschaltung kann die Spannung regulieren und den Strom begrenzen. Die eingesetzte Kollektorschaltung reguliert die Spannung mit Hilfe einer Zenerdiode auf 12V (in der Abbildung 35 die D1). Sobald die Spannung 12V übersteigt, verringert sich der Widerstand über Zenerdiode und die Spannung regelt sich wieder bei 12V. Für die Strombegrenzung betrachten wir die Schaltung im nicht begrenzten Zustand. Der Strom fliest von V_{CC} (technische Stromrichtung) zum Kollektor und zur Base des Transistor T1. Sperrt der Transistor T2, so schaltet der Strom den Transistor T1 und am Ausgang haben wir 12V abzüglich U_{BE} von T1 und U_{RM} . Steigt der Strom, so steigt die Spannung an RM, bis sich T2 beginnt zu öffnen. Wird T2 geöffnet, fliest der Strom der Base von T1 über den Kollektor zum Emitter von T2. Der Transistor T1 beginnt zu sperren, der Strom wird begrenzt *Mathematik für Elektroniker/in für Automatisierungstechnik 2018*. Mit dem Widerstand RV kontrolliert man den Kollektorstrom von T2. Wenn der Kollektorstrom sinkt, dann sinkt auch die Stromverstärkung. ($h_{fe} = I_c/I_b$)

4.4.2 5V Versorgung v1

Wie Geerling in seinem Blog erwähnt benötigt der Raspberry pi bei maximaler Auslastung 980mA Geerling 2019.

Für die 5V Spannungsquelle wird ein LM341-T eingesetzt. Dies ist ein linearer Spannungsregler, der nur zwei Stützkondensatoren benötigt. V. 2016. Da ein einzelner LM341-T nur 0.5A liefern kann, werden

4 Ansteuerung der Elektronik

zwei im Parallelbetrieb betrieben (siehe Abbildung 36). Die Abbildung 4, im Datenblatt des LM341-T zeigt, dass die Differenz zwischen dem In- und Output bei einer Betriebstemperatur von 100°C 1.5V nicht überschritten werden darf. V. 2016. Aus diesem Grund verbraucht RV1 und 2 die restliche Leistung. Wie beim booten des Raspberry Pi's herausgefunden wurde, gilt die 980mA ausschliesslich für das absolute Minimum. Dies reicht nicht für unsere Zwecke, deshalb muss die Version 2 mehr Strom liefern können.

4.5 Version 2

Mit den Erkenntnissen von der Version 1 wird die ganze Elektronik überarbeitet. Das Blockschaltbild wird beibehalten. Anschliessend wird die Version 2 als PCB(Printed circuit board) bestellt. Bei Möglichkeit werden Schaltungselemente durch integrierte Lösungen ersetzt um die Wahrscheinlichkeit zu reduzieren, nochmals Re-Design machen zu müssen.

4.5.1 Ansteuerung der Motoren im Artomat controllerboard V2

Bei der Motorsteuerung im Artomat controllerboard V1 war der Verlust über die Transistoren in der Halbbrücke gross. Das Ziel der Motorensteuerung im Artomat controllerboard V2 ist, diese Verlustleistung zu reduzieren. Dies heisst, statt auf bipolare Transistoren wird nun auf unipolare gesetzt. Kriterien für integrierte H-Brücke

$$R_{DS(on)} = \frac{U_{Verlust\ max.}}{I_{Motor}} = \frac{1V}{1.44A} \quad (11)$$

Nach den Werten von der Tabelle 37, wurde das Bauteil DRV8870 von Texas Instruments gewählt. Es verfügt über die notwendigen Spezifikationen. Die Steuerung für die Motoren mit dem DRV8870 ist im Schema der Abbildung 38 zu sehen. Der DRV8870 verfügt ausserdem über eine integrierte Strombegrenzung, die sich wie folgt berechnen lässt. Der DRV8870 verfügt über eine integrierte Strombegrenzung, die sich wie folgt berechnen lässt.

$$V_{REF} = \frac{U1}{R3 + R4} * R4 = \frac{12V}{8,2\text{ k}\Omega + 3,3\text{ k}\Omega} = 3.4V \quad (12)$$

4 Ansteuerung der Elektronik

$$I_{\max} = \frac{V_{\text{REF}}}{A * R_{\text{ISEN}}} = \frac{3.4V}{10 * 0.2\Omega} = 1.7A \quad (13)$$

Für die Pufferkondensatoren werden die vom Hersteller empfohlenen verwendet. Leider hat der Hersteller die falschen Widerstände zugeschickt, daher mussten wir die Widerstände brückeln. In der Abbildung 39, kann man erkennen, dass die Schaltung ordnungsgemäss funktioniert.

4.5.2 Ansteuerung der Solenoiden im Artomat Controllerboard V2

Das Design in der Version 1 funktionierte bereits gut, daher wird das selbe Design verwendet und kostenorientiert umgesetzt. Daher wird statt einzelnen Transistoren ein Transistorarray verwendet. Ein Optokopplerarray ist nicht Preis effektiv, daher wird ein Optokoppler verwendet, der zwei in einem Gehäuse hat. Kriterien für Transistorarray Nach den Werten von der Tabelle ?? wurde das Transistorarray ULN2004 von STMicroelectronics gewählt. Um den maximalen Strom liefern zu können, werden zwei Transistorstufen parallel betrieben. Als Endnutzen für den ULN2004 wird im Datenblatt explizit Solenoiden erwähnt. Der ULN2004 verfügt sogar über eine interne Freilaufdiode. Der Schaltstrom an der Base ist gemäss Datenblatt ungefähr 1 mA. 2018. Dieser ist zugleich der I_c , des Optokopplers. Dem entsprechen eine Vielzahl von Optokopplern. Gewählt wurde der ILD207 von Vishay, da er preisgünstig und gut dokumentiert ist. Als Vorwiderstand des Optokopplers wurde derselbe eingesetzt, wie im Schema der Abbildung 29.

4.5.3 Stromversorgung V2

Bei der Version 2 der Stromversorgung ist das Ziel, den Leistungsverlust zu reduzieren. Zusätzlich soll dem Raspberry Pi mehr Strom zur Verfügung stehen. Um den Leistungsverlust zu reduzieren, werden statt Linearenspannungswandler oder Transistoren, Schaltregler verwendet. Über ein Netzgerät, das im Elektroschrott lag, wird die Netzspannung in 20VDC mit maximal 4.75A konvertiert. Das Stromversorgungsboard konvertiert diese 20V in 12V und eine 5V Spannungsquelle. Für die Konvertierung von 20V zu 12V und 5V werden Stepdown Konverter vom Typ LM2596 von Texas Instruments verwendet. Der LM2596 ist in 3 Versionen erhältlich, einer 12V, 5V und eine Adj, bei der man die Spannung einstellen kann. Wir verwenden die 12V und die 5V Version. Das offizielle Netzteil des Raspberry Pi's liefert 2.5A, wir brauchen aber nicht unbedingt so viel, da wir keine USB-Geräte anschliessen werden.

4 Ansteuerung der Elektronik

Aus diesem Grund werden wir nicht mehr als 2A benötigen o.V. o.J.(c). Für die Stromversorgung des Artomat controllers gehen wir davon aus, dass nicht alle Aktoren gleichzeitig eingeschaltet werden. Dies weil der maximal zugelassene Strom des Netzgerätes überschritten werden könnte und dies dem Raspberry Pi den Strom abschnüren würde. Aus diesem Grund spezifizieren wir die Stromaufnahme für die 12V Quelle auf 2.5A um die Leistung des Netzgeräts nicht zu überschreiten. Der LM2596 erfordert wenig zusätzliche Bauteile. Das verwendete Schema ist eine Modifikation des Beispiel Schemas aus dem Datenblatto.V. 1999. Das verwendete Datenblatt ist in der Abbildung 40 zu sehen. Die Grösse der Spule bestimmt den maximalen Stromfluss, wie die Abbildungen 41 und 42 zeigen. In der Abbildung 43 ist die Schaltung der Stromversorgung v2 unter voller Last zu sehen. Der Kanal CH1 misst den Spannungsausgang und der CH2 dessen Stroms über der Last von einem $3,7\Omega$ Widerstand. Zu sehen ist ein starker Ripple, der ist vernachlässigbar, da nur Aktoren damit betrieben werden.

4.5.4 PCB Entwicklung

Hersteller für die im Projekt entwickelten Leiterplatten ist JLCPCB. Der Hersteller bietet ein indstrie-führendes Preisleistungsverhältnis. Da die Leiterplatten aus China kommen, muss mit einer Liefer- und Herstellungszeit von zusammen einer Woche gerechnet werden. Um die Entwicklung der Leiterplatten, den Herstellungsmöglichkeiten von JLCPCB anzupassen, kann man RUL-Files von der Herstellerseite herunterladen. Die RUL-Files benützt das Layout Programm um zu warnen, wenn man die Herstellungsmöglichkeiten des Herstellers nicht einhaltet o.V. o.J.(b). Das PCB für die Stromversorgung wurde mit Eagle entwickelt. Der Artomat Controller wurde mit Altium 2017 entwickelt.

4.6 Steuerung Artomat Controller

Die Aufgabe des Artomat Controllers ist, ein Verbindungsauftbau zum Modul B zu erstellen, die Signale des Artomat cam zu verarbeiten und für das Modul D aufzubereiten. Kommuniziert wird gemäss Schnittstelle 2 und 3.

4.6.1 Verwendete Bibliotheken

socket, pickle, time, RPI.GPIO, _thread Für den Verbindungsauftbau zum Modul B wurden die Bibliotheken socket und pickle verwendet. Dabei ist die Bibliothek soket für die Verbindung zuständig.

4 Ansteuerung der Elektronik

Pickle ist für das decodieren der Übertragungsdaten zuständig. Die Bibliothek RPI.GPIO wird verwendet um auf die Ausgaberegister des Raspberry Pi's, genannt GPIO's zuzugreifen. Die Bibliothek time, wird verwendet um die Zeit zu messen und um Verzögerungen einzubauen. _thread erstellt verschiedene Threads innerhalb eines Programm, diese wird verwendet um sequenzielle Programmierung zu realisieren. Alle diese Bibliotheken sind auf dem Raspberry Pi bereits vorinstalliert.

4.6.2 Erhalten von Daten auf der Schnittstelle 2

Für die Kommunikation wurde eine Klasse namens ArtomatSocket erstellt. Diese kann eine sich mit einem im gleichen Netzwerk befindenden Socket-Server verbinden, Daten empfangen und sich von der Verbindung trennen. Die Verbindung wird erstellt durch die Funktion connect in der ArtmatSocket Klasse. Um eine Verbindung zu erstellen muss der Funktion die IPV4 Adresse des Servers und der Port des Sockets übermittelt werden. Falls die Verbindung erstellt werden konnte, gibt die Funktion den Wert True zurück, sonst wird False zurückgegeben. Mit der Funktion receiveData kann auf einkommende Daten reagiert werden. Wenn sie einmal aufgerufen wird, wartet sie, bis sie etwas empfängt und returniert erst dann. Die Daten werden als bits empfangen. Es ist ein Buffer von 16384 Bits eingestellt. Werden mehr gesendet werden diese nicht berücksichtigt. Die Nachricht wird mit der pickle Bibliothek zu Bits konvertiert, anschliessend wird ein Zeitstempel in die Nachricht verpackt. Da der Sender dies ebenfalls gemacht hat, kann so die Sendezeit überwacht werden. Scheitert das Lesen dieser Funktion, z.B. durch einen Verbindungsabbruch, so wird das ganze Übertragungsarray auf null gesetzt bis auf SOS, dies wird auf True gesetzt. Mit dem Aufrufen der Funktion disconnect wird die Verbindung gekappt.

4.6.3 Ausgabe auf Schnittstelle 3

Die Ausgabe auf die Schnittstelle 3 erfolgt mit der Klasse Outputcontroll. Das Initialisieren der GPIO's erfolgt beim erwecken einer Instance der Klasse. Dabei werden alle Ausgänge auf 0 gesetzt. Für die Kontrolle des Motors ist die Funktion motor_M1 und motor_M2 zuständig. Der Funktion wird eine Variable, genannt velocity übergeben. Die Variabel velocity kann negativ, positiv oder null sein. Null bedeutet Stillstand, negativ und positiv unterscheiden sich durch die Motordrehrichtung. Die Grösse des Werts definiert die Geschwindigkeit. Die Solenoiden wird durch die Funktionen solenoid_s1, so-

5 Simulation

lenoid_s2 und solenoid_s3 gesteuert. Übergeben wird der Funktion, die Variable ontime, wie in der Abbildung 44 zu sehen ist. Sobald ein Wert grösser als null der Funktion zugesendet wird, wird der Sprühvorgang aktiviert. Die Variabel ontime bestimmt wie lange sie in Sekunden aktiv sein soll. Da ein theoretisch Endloser Sprühvorgang möglich wäre, muss dies begrenzt werden. Dies ist nötig als Überhitzungsschutz für den Solenoiden und um tropfende Farbe an der Wand zu verhindern.

4.6.4 Koordination

Die Koordination der beiden Klassen findet im main.py File statt. Wurde das Script gestartet, wird alles initialisiert und wartet, bis eine Verbindung zum Socket-Server hergestellt werden kann. Danach wird in einer Schleife die receiveData Funktion in einem separaten Thread aufgerufen. In einer zweiten Schleife werden die vom Modul B erhaltenen Werte an die Outputcontroll Klasse zugeschickt. Zusätzlich wird die Verbindung überprüft. Bei einem Time-out wird alles auf null gesetzt und die Verbindung wird abgebrochen. Bei einem Notfall wird Modul E ausgeschaltet.

5 Simulation

5.0.1 Grundidee

Die Simulation entstand, als wir uns die Frage stellten, wie wir am besten Parallel arbeiten könnten. Ohne der Simulation hätten wir den Navigierungsalgorithmus erst nach Bau des Roboters anfangen können. Mit dieser Lösung jedoch, konnten wir das Skript so gut wie möglich vorbereiten und schlussendlich nur noch für den Roboter anpassen.

5.1 Installation

5.1.1 Voraussetzungen

1. Python Version ≥ 3.6
<https://www.python.org/downloads/>
2. Python Bibliotheken

Nach dem Download von Python müssen noch folgende Bibliotheken installiert werden.

5 Simulation

- tkinter
- numpy
- Pillow

Der Konsolen Command funktioniert folgenderweise:

```
pip install <bibliothek>
```

Der Code befindet sich im IDPA Ordner, oder kann über das folgende Git Repository lokal auf den Computer gecloned werden.

```
git clone https://github.com/nilsegger/idpa.git
```

5.1.2 Skript Starten

Starten Sie eine Konsole und wechseln Sie den Pfad zum Skript Ordner. Stellen Sie sicher, dass Sie das Skript simulation.py im Ordner sehen. Starten Sie dann dieses mit:

```
python simulation.py
```

5.1.3 Visuelle Erklärung

In der Abbildung 45 sehen Sie die Visuelle Erklärung der Simulation.

Die Roten Punkte imitieren die LEDs, welche an der Wand und am Roboter befestigt sind. Diese müssen dann von unserem Navigationsskript korrekt erkannt werden, damit ein Bild überhaupt gesprayed werden kann.

5.1.4 Steuerungen

Die Simulation lässt sich über folgende Tastaturtasten steuern.

- W

Mit W dreht sich der linke Motor auf, somit wird das linke Seil kleiner.

- S

Mit S gibt der linke Motor nach, somit wird das linke Seil grösser.

5 Simulation

- ↑

Mit ↑ dreht sich der rechte Motor auf, somit wird das rechte Seil kleiner.

- ↓

Mit ↓ gibt der rechte Motor nach, somit wird das rechte Seil grösser.

- Q

Mit Q wird die Wand (in diesem Fall Canvas) mit einem Punkt bemalt.

5.2 Bibliotheken

Für die Simulation gebrauche ich Tkinter für die Darstellung. Im Nachhinein war dies vermutlich eine Fehlentscheidung, dies nicht weil die Bibliothek die Simulation schlecht darstellt, sondern weil sie die Darstellung nicht einfach in eine Pixel Matrix kopieren kann. Müsste ich nochmals eine auswahl treffen, so würde diese bei OpenCV liegen, denn mit OpenCV hätte ich nur kleine Performance Verluste beim Transport des Bildes zum Navigationsalgorithmus gehabt. Dies wusste ich im voraus nicht da ich weder mit OpenCV oder Tkinter bekannt war, in Zukunft wird dies aber ein No-Brainer. Damit ich eine Kamera auf die Simulation simulieren konnte, musste ich den Tkinter Canvas in das einzige Unterstützte Format exportieren, nähmlich Postscript ein Vektor Format. Postscript wird jedoch natürlich nicht von OpenCV unterstützt und anstatt eine Kamera einfach auf den Bildschirm zu richten, musste ich eine weitere Bibliothek benutzen. PIL (Python Imaging Library) kam hierbei zur Lösung, diese Bibliothek kann Postscript in ein normales Bild verwandeln welches von Numpy (Matrix Bibliothek, welche von OpenCV für die Pixel Matrix von Bildern benutzt wird.) dann als Multidimensionaler Array eingelesen wird und von OpenCV bereit ist. Dies entspricht folgendem Code:

```
numpy.array(Image.open(io.BytesIO(self.ps_frame.encode('utf-8'))))
```

Diese eine Linie Code entnimmt dem Navigationsskript jegliche Möglichkeit bei 60 Bildern pro Sekunde zu arbeiten.

5.3 Aufbau

Die Simulation besteht aus folgenden Klassen:

5 Simulation

1. Window

Die Window-Klasse ist unter anderem dafür zuständig, einen Canvas für die Simulation bereit zu stellen und diesen gleichzeitig per Interface Funktion an OpenCV weiterzugeben. Die Hauptfunktion ist jedoch die Endlosschleife zu steuern, diese wird von Tkinter durch folgende Funktion übernommen:

```
Tk().after(zeit_in_ms, callback_funktion)
```

Bei jeder Wiederholung wird die vergangene Zeit gemessen und anhand von abgefangen Tastenschlägen die Geschwindigkeit der Motoren in der Simulation festgelegt.

2. Simulation

Die Simulation-Klasse ist das Herzstück dieses Skripts. Sie berechnet die Positionen des Robotors und zeichnet diese auch gleich auf dem bereitgestellten Canvas der Window Klasse.

3. Object

Ursprünglich hatte ich die Idee ganz viele kleinere Klassen zu schreiben. Im Sinne, dass der Motor sowie der Spraykopf durch eine eigene Klasse repräsentiert werden. Diese Objekte würden dann alle vom Window gezeichnet werden. Dieser Anlauf habe ich dann aber auch schnell wieder geändert. Nun ist es so, dass die Simulation-Klasse eine Kinderklasse der Object Klasse ist und somit alle Funktionen erbt. Die Object Klasse hat Funktionen wie zum Beispiel die Distanz zwischen zwei Vektoren zu kalkulieren, oder eine Linie auf einen Canvas zu zeichnen.

4. Vec2

Die Vec2-Klasse beinhaltet Hilfsfunktionen für Berechnungen mit einem zweidimensionalen Vektor. Diese Funktionen sind unter anderem die Länge des Vektor zu berechnen, den Vektor zu drehen, oder zwei Vektoren zusammen zu rechnen.

5. ObjectDimension

Der Name dieser Klasse beschreibt vielleicht nicht so gut was sie genau macht. Diese Klasse speichert nur zwei Vektoren, die Position und Grösse eines Objekts der Simulation (zum Beispiel der Motoren), zudem hat es eine Hilfsfunktion um die Mitte eines Objekts dieser Klasse zu berechnen.

5 Simulation

5.4 Simulation Physik

Bei der Physik der Simulation war für mich wichtig, dass diese so echt wie möglich ist. Dies ist mir meiner Meinung nach im zweiten Versuch gut gelungen. Ich wollte, dass sich die Seile anfühlen wie Seile, damit meine ich zum Beispiel, dass wenn man mit dem linken Motor das linke Seil anzieht, dass dieses auch ein bisschen oberhalb des rechten ist, weil die Gravitation das rechte Seil mit dem Gewicht des Roboters noch runter zieht.

5.4.1 Demonstration

In der Ausgangslage 46 sieht man die Ausgangslage der Simulation.

Linker Motor angezogen:

Wie Sie in der Abbildung 47, welche ich mit Gelb angestrichen habe nicht mehr gerade. Dies sieht vielleicht nicht nach viel aus, es war für mich aber ein grosser Erfolg. Der linke Motor wurde angezogen, jedoch nur so viel, dass der rechte Motor noch am gleichen Ort bleibt. Da die Verbindung zwischen den zwei Motoren immer noch gleich lang ist, wäre der linke Motor noch weiter hoch gezogen worden, so wäre auch der rechte Motor verschoben.

In der Abbildung 48, 49, 50 sehen Sie verschiedene Extremfälle. Bei der Ersten dreht sich der rechte Motor aus und der linke Motor dreht sich ein. Das rechte Seil ist lang und das linke ist kurz.

Wie Sie vermutlich schon selbst bemerkt haben, funktioniert die Simulation in diesen Extremfällen nicht mehr flüssig, darüber werde ich weiter unten noch mehr erzählen.

5.4.2 Code Erklärung

Die Simulation wird über zwei wichtige Funktionen gesteuert,

```
def spin_left_motor(self, speed):  
    pass
```

5 Simulation

```
def spin_right_motor(self, speed):  
    pass
```

Diese zwei Funktionen sind verkehrte Kopien von einander, weswegen nur die Variante der linken Seite erklärt wird. Wichtig zu wissen ist, dass diese Funktionen bei den richtigen Tastenanschlägen in der Hauptschleife von Window immer wieder aufgerufen werden. Die Hauptfunktion basiert somit auf mehreren Aufrufen der letzteren Funktionen.

5 Simulation

```
def spin_left_motor(self, speed):  
  
    """  
    Bei negativer Geschwindigkeit verkleinert sich das Seil,  
    somit wird bei positiver Geschwindigkeit das Seil natuerlich groesser.  
    has_rope_tension() testet ob die Laengen der zwei Seile  
    plus der Anfangsabstand der zwei Motoren gleichgross oder kleiner als  
    der Abstand zwischen den zwei Wand Markierungen ist.  
    Wenn Wahr,  
    kann man die Motoren nicht weiter anspannen weil sonst das Seil reissen  
    wuerde.  
    Somit wird die Funktion fruehzeitig verlassen.  
    """  
  
    if speed < 0 and self.has_rope_tension():  
        return  
  
    """  
    Position und zwischengespeicherte Seil Laenge werden  
    auf gewuenschte Position/ Laenge veraendert.  
    """  
  
    self.left_rope_distance += speed  
    self.motor_left.pos.add(self.motor_left_to_left_corner_vec, speed)  
  
    # noch unwichtig.  
    self.slow_forward = 0.05  
    self.medium_forward = 0.1  
    self.fast_forward = 1  
    self.faster_than_fast_forward = 1.25
```

5 Simulation

```
if speed < 0:  
    pass  
else:  
    pass
```

Die Funktion wird nun in zwei Richtungen aufgeteilt, wenn der Motor aufgerollt wird, oder eben ausrollt. Der Algorithmus für das Aufrollen des Seiles ($speed < 0$):

Beim Aufrollen des Seiles werden zwischen zwei verschiedenen Situationen unterschieden.

1. Die Seile sind noch locker. Der Roboter ist die Seile noch am herunterziehen, erkennbar dadurch, dass sich die beiden Seile noch überkreuzen könnten.
2. Die Seile sind richtig gespannt. Diese Situation wird erreicht, wenn sich die beiden Seile nicht mehr überkreuzen könnten.

...

```
if speed < 0:  
    if self.rope_intercept:  
        # Situation Seile koennen sich noch ueberkreuzen. (1)  
        pass  
    else:  
        # Situation Seile koennten sich nicht mehr ueberkreuzen. (2)  
        pass
```

Stellen Sie sich nun vor, der linke Motor wird angespannt und die Seile sind immer noch locker, somit sind wir in der ersten Situation. Nun ist die Frage, ob der rechte Motor vom linken Motor mitgezogen wird oder nicht. Um dies herauszufinden misst man den neuen Abstand zwischen den zwei Motoren und vergleicht ihn mit dem Startabstand dieser zwei. Ist der neue Abstand grösser, so muss der rechte Motor nachgeschoben werden. Unser Zwischenstand:

...

```
if speed < 0:  
    if self.rope_intercept:
```

5 Simulation

```
while self.current_motor_to_motor_distance \
> self.motor_to_motor_starting_distance:
    pass
```

Nun muss nur noch der rechte Motor nachgeschoben werden. Dieser darf sich jedoch nur um die rechte Wandmarkierung im Radius der Länge des rechten Seiles bewegen. Visuell mit Gelb gekennzeichnet in der Abbildung 51 Für diese Bewegung ist für beide Motoren eine gegenseitig kopierte Funktion vorhanden. Erklärung für die Bewegung des linken Motors:

```
def move_left_motor(self, degree: float, forward: float):
    distance_to_corner = self.current_left_motor_to_corner_distance
    """

```

Hier wird der Richtungsvektor von der Linkenmarkierung zum Motor berechnet,

dieser wird dann um die gewuenschte Grad (degree) anzahl gedreht.

```
"""

```

```
rotation_vec = self.calculate_vec(self.corner_left.center, self.
    motor_left.center)
rotation_vec.rotate(degree)
"""

```

Neuer Punkt wird erstellt und ein neuer Richtungsvektor vom Motor zu diesem Punkt wird berechnet,

danach wird dieser neuer Richtungsvektor dem Motor hinzugefuegt. Der Motor ist jetzt am neuen gewuenschten Punkt.

```
"""

```

```
rotated_point = Vec2(copy=self.corner_left.center)
rotated_point.add(rotation_vec, distance_to_corner)
forward_vec = self.calculate_vec(self.motor_left.center, rotated_point)
self.motor_left.pos.add(forward_vec, forward)
"""

```

5 Simulation

```
Hier wird die Position noch korrigiert ,  
da die neue Position einbisschen innerhalb des Kreises liegt ,  
muss der neue Punkt noch einbisschen nach aussen korrigiert werden.  
Sonst wuerde der Motor irgendwann bei der Wandmarkierung sein .
```

```
"""
```

```
    self.motor_left.pos.add(rotation_vec , distance_to_corner - self.  
        current_left_motor_to_corner_distance)
```

Beim Nachschieben des rechten Motors sieht die Situation folgendermassen aus - Der Motor wird nun in die richtige Richtung gedreht. Wir beobachten, dass dieser nicht über das gewollte Mass hinausgeht, in dem Sinne, dass der rechte Motor immer tiefer oder auf der gleichen Höhe bleibt wie der linke Motor.

```
...
```

```
if speed < 0:  
    if self.ropes_intercept:  
        while self.current_motor_to_motor_distance \  
            > self.motor_to_motor_starting_distance and self.motor_right.center.y  
                \  
                >= self.motor_left.center.y:  
                    self.move_right_motor(1 , self.slow_forward)  
                    # Falls der Motor hoeher als der Linke wurde ,  
                    # wird der Motor wieder zurueck positioniert .  
                    if self.motor_right.center.y < self.motor_left.center.y:  
                        self.move_right_motor(-1 , self.slow_forward)  
                    break
```

Es kann hier passieren, dass der rechte Motor bis auf die gleiche Höhe des linken nachgeschoben wird. In diesem Fall sind die Motoren jedoch immer noch weiter voneinander entfernt, als sie es in echt sein könnten. Dies wird wieder mit dem original Abstand versus dem momentanem Abstand der Motoren bestimmt und um den Abstand zu korrigieren, wird der linke Motor der Linie nach, nach oben verschoben, bis der kleinstmögliche Abstand erreicht wird.

5 Simulation

```
...
if speed < 0:
    if self.ropes_intercept:
        while self.current_motor_to_motor_distance > \
            self.motor_to_motor_starting_distance and self.motor_right.center.y \
                \ 
            >= self.motor_left.center.y:
            self.move_right_motor(1, self.slow_forward)
            # Falls der Motor hoher als der Linke wurde,
            # wird der Motor wieder zurueck positioniert.
            if self.motor_right.center.y < self.motor_left.center.y:
                self.move_right_motor(-1, self.slow_forward)
            break

last_motor_distance = self.current_motor_to_motor_distance
while self.current_motor_to_motor_distance >
    \ self.motor_to_motor_starting_distance and last_motor_distance \
        >= self.current_motor_to_motor_distance:
    last_motor_distance = self.current_motor_to_motor_distance
    self.move_left_motor(-1, self.slow_forward)
    if last_motor_distance < self.current_motor_to_motor_distance:
        self.move_left_motor(1, self.slow_forward)
```

Dies ist der finale Code für die erste Situation. Bei der Zweiten, wenn die Seile gespannt sind und sich nicht mehr überkreuzen können, wird einfach der rechte Motor nachgeschoben, bis er die gleiche Höhe erreicht wie der linke. Danach ergibt sich aber schnell wieder das gleiche Problem, die Motoren sind zu weit auseinander. Als Lösung wird der linke Motor ein wenig hochgeschoben und der rechte wieder nachgeschoben, bis er die gleiche Höhe erreicht. Dies funktioniert, weil die Endpunkte der Seile zuoberst am nächsten beieinander sind.

5 Simulation

```
...
if speed < 0:
    if not self.ropes_intercept:
        while self.motor_right.center.y > self.motor_left.center.y:
            self.move_right_motor(1, self.slow_forward)
            while self.current_motor_to_motor_distance > \
                self.motor_to_motor_starting_distance and not self.has_rope_tension
                ():

            self.move_right_motor(1, self.slow_forward)
        while self.motor_right.center.y < self.motor_left.center.y:
            self.move_left_motor(-1, self.slow_forward)
```

Als letztes gibt es natürlich noch die Situation, wo die Motoren nachlassen und das Seil länger wird. Wenn der linke Motor hinunter gelassen wird, wird zuerst geschaut, ob die Distanz zu den Motoren kleiner als die ursprüngliche ist. Ist dies der Fall, so wird der linke Motor hinunter gelassen, bis der Abstand erreicht wird. Zum Schluss muss der rechte Motor nachgeschoben werden, da dieser vom Bewegen des linken ein wenig mitgezogen wurde.

```
...
if speed > 0:
    while self.current_motor_to_motor_distance < self.
        motor_to_motor_starting_distance:
            self.move_left_motor(1, self.slow_forward)
    while self.motor_right.center.y < self.motor_left.center.y \ 
        <= self.corner_right.center.y + self.right_rope_distance:
            self.move_right_motor(-1, self.fast_forward)
```

Nur mit diesem Code ergibt sich das Problem, dass es sich visuell nicht wirklich echt anfühlt, der Motoren Abstand zu gross wird und die Simulation in der Nähe von den unteren Ecken komplett spinnt. Visuell in der Abbildung 52

Wie man sieht, wäre dies in Echt nicht so. Der linke Motor würde gerade nach unten hängen. Um

5 Simulation

dies zu erreichen, erkenne ich den Moment, wo beide Motoren den gleichen Richtungsvektor (Länge bei beiden auf 1 berechnet) zur rechten Wandmarkierung aufweisen. Danach ist es nur noch eine Frage des Motorenabstands, welcher noch korrigiert werden muss.

...

```
if speed > 0:  
    while self.current_motor_to_motor_distance <\br/>        self.motor_to_motor_starting_distance:  
            self.move_left_motor(1, self.slow_forward)  
  
    while self.motor_right.center.y < self.motor_left.center.y \  
        <= self.corner_right.center.y + self.right_rope_distance:  
        self.move_right_motor(-1, self.fast_forward)  
  
    left_motor_to_right_corner_vec = \  
        self.calculate_vec(self.corner_right.center, self.motor_left.center)  
    right_motor_to_right_corner_vec = \  
        self.calculate_vec(self.corner_right.center, self.motor_right.center)  
  
    while self.current_motor_to_motor_distance \  
        > self.motor_to_motor_starting_distance \  
        and not left_motor_to_right_corner_vec.compare(  
            right_motor_to_right_corner_vec, 2):  
            self.move_right_motor(1, self.faster_than_fast_forward)  
  
    left_motor_to_right_corner_vec =\  
        self.calculate_vec(self.corner_right.center, self.motor_left.center)  
    right_motor_to_right_corner_vec = \  
        self.calculate_vec(self.corner_right.center, self.motor_right.center)
```

```

if self.current_motor_to_motor_distance > \
    self.motor_to_motor_starting_distance:
    motor_to_motor_forward = \
        self.calculate_vec(self.motor_right.center, self.motor_left.center)
    new_pos = Vec2(copy=self.motor_right.pos)
    new_pos.add(motor_to_motor_forward, self.
        motor_to_motor_starting_distance - 0.1)
    self.motor_left.pos = new_pos
    self.left_rope_distance = \
        self.calculate_length(self.motor_left.center, self.corner_left.center
    )

```

5.4.3 Probleme

Der Code war während der Entwicklung extrem anfällig für Endlosschleifen, dies konnte aber mit der Zeit gut behoben werden. Die Simulation ist zudem nicht sehr genau, was daran liegt, dass die Position nicht wirklich berechnet werden kann, sondern solange herum geschoben wird, bis die Position allen Anforderungen (Motoren Abstand oder zum Beispiel, dass der eine Motor nicht höher sein darf) entspricht. Der Code läuft zudem nicht immer, wegen der Bewegungsfunktion für die Motoren, ganz flüssig. Diese Funktion gebraucht nicht lange zum Berechnen der neuen Position, sondern macht dies Ungenau. Durch die Ungenauigkeit werden zum Teil Positionen erreicht, welche nicht den Anforderungen entsprechen und als Folge haben, dass der Code die Position korrigieren muss. Die neue korrigierte Position wurde aber genau gleich erreicht, nämlich mit der ungenauen Funktion. Somit sind wir wieder bei der Ausgangslage. Nun stellt man sich vielleicht die Frage, warum die Bewegungsfunktion nicht besser geschrieben wurde. Wie oben schon beschrieben wurde, funktioniert sie in dem man zuerst den Richtungsvektor zur Wandmarkierung misst, diesen um eine gewünschte Anzahl Grad dreht, in unserem Fall immer 1 Grad, diesen Richtungsvektor der Wandmarkierung hinzurechnet und danach einen neuen Richtungsvektor zwischen dem zu bewegenden Motor und dem neu kalkulierten Punkt berechnet. Danach wird die Länge des Richtungsvektors auf 1 kalkuliert und die Distanz, welche man

6 Steuerung

den Motor bewegen will, dazu gerechnet. Bei dieser letzten Bewegung geht die Rundung des Kreises verloren und man muss die Position, besser gesagt die Distanz zur Wandposition, korrigieren. Dies ist ungenau. Viel besser wäre es wie vorher den Richtungsvektor zwischen Motor und Wandmarkierung zu messen und diesen um weniger als 1 Grad zu drehen. Danach müsste man den Richtungsvektor ganz einfach wieder der Wandmarkierung hinzufügen und der neue Punkt wäre bereits perfekt getroffen worden. Das Problem liegt beim Rotieren des Vektors. Wenn der Wert des Grades kleiner als eins war, so berechnete Python die neuen Werte komplett falsch. Somit musste ich eine andere Variante finden.

5.4.4 Resultat

In der Abbildung 53 sehen Sie ein Resultat mit dem Navigationsskript angehängt.

6 Steuerung

6.1 Installation

6.1.1 Voraussetzungen

1. Python Version ≥ 3.6

<https://www.python.org/downloads/>

2. Python Bibliotheken

Nach dem Download von Python müssen wir noch folgende Bibliotheken installieren.

- tkinter
- numpy
- Pillow
- opencv-python

Der Konsolen Command funktioniert folgenderweise:

```
pip install <bibliothek>
```

Die Dateien zur Steuerung befinden sich im Abgabeordner der IDPA, oder können über ein Git Repository lokal auf den Computer gecloned werden.

6 Steuerung

```
git clone https://github.com/nilsegger/idpa.git
```

6.1.2 Skript starten

Öffnen Sie eine Konsole im Ordner und geben Sie folgenden Command ein.

```
python main.py <bild_pfad>
```

6.1.3 Visuelle Erklärung

In der Abbildung 54 sieht man die Visuelle Erklärung.

6.2 Aufbau

1. Camera

Die Camera-Klasse ist ein Interface, zuständig für die Zulieferung des Livestreams auf den Roboter in Form einer Pixelmatrix. Die Kamera-Klasse wird jedoch nur bei der Verbindung der Simulation benutzt. Für die echte Kamera wurde das Skript angepasst.

2. MotorInterface

Das MotorInterface ist dafür zuständig, die Steuerung für die Motoren bereitzustellen.

3. Vision

Die Vision-Klasse baut alles zusammen. Nimmt den Input von Camera, erkennt alle nötigen Punkte vom Roboter und steuert diesen mit dem MotorInterface.

6.3 Code

6.3.1 Ablauf

In der Abbildung 55 sieht man den Ablauf.

6.3.2 Markierungen Erkennung

Um die Markierungen zu erkennen, sucht OpenCV nach Kreisen im gegebenen Bild. Dies funktioniert am besten in einem schwarz-weiss Bild. Der schwierige Teil, bevor man das Bild in schwarz-weiss

6 Steuerung

umwandeln kann, ist das Finden des richtigen Farbbereichs. Ohne diesen Schritt würde es viel zu viele, falsche und ungewollte Kreise finden. Für die Navigation ist es wiederum sehr wichtig, dass man mit Sicherheit sagen kann, welcher Kreis zu welcher Markierung gehört.

Um dies zu Erreichen wird das Bild zuerst in den HSV-Farbraum konvertiert. Danach werden die Farben ausgegrenzt, die nicht im gesuchten Farbbereich liegen. Dies resultiert in einem schwarz-weiss Schema, dass die passenden Bildpunkte vom Rest des Bildes unterscheidet und eine perfekte Basis für die Erkennung der Kreise bildet. In der Abbildung 56 sieht man den HSV Farbspektrum, in 57 den Farbraum und in 58 die Erkennung der Kreisen.

```
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
lower_range = np.array([lb, lg, lr])
upper_range = np.array([ub, ug, ur])
mask = cv2.inRange(hsv, lower_range, upper_range)
circles = cv2.HoughCircles(mask, cv2.HOUGH_GRADIENT, 1, min_dist, param1=
    param1, param2=param2, minRadius=0,
maxRadius=max_radius)
```

Nach dem Erkennen muss ermittelt werden, welche Markierung welchem Motor entspricht. Dafür werden alle Markierungen vom kleinsten y-Wert zum grössten sortiert. Danach wird die x-Achse der ersten Markierung mit der zweiten verglichen. Hierbei entspricht der kleinere Wert der linken Position. Das gleiche wird mit der dritten und vierten Markierung wiederholt. Schlussendlich hat man ein Array, das dem folgendem entspricht:

1. Linke Wand Markierung
2. Rechte Wand Markierung
3. Linker Motor
4. Rechter Motor

6.3.3 Algorithmus

6 Steuerung

```
"""
```

```
Dieser Code entspricht nicht zu 100 Prozent der echten Version fuer  
Lesbarkeit ,  
zudem unterscheiden sich der Code fuer die Simulation und den echten  
Roboter .
```

```
"""
```

```
markers = self.camera.get_markers(frame)  
if markers is not None:  
    self.show_markers(overlay , markers)
```

```
"""
```

```
Calculate Vales berechnet Werte wie der Abstand zwischen Motoren und  
Wand Markierungen ,  
cm zu pixel multiplikator und den Canvas Rand .
```

```
"""
```

```
if self.calculate_values(markers):
```

```
"""
```

```
Wurde das Bild bereits vorbereitet ?
```

```
"""
```

```
if self.image_scaled:
```

```
    if not self.start_printing:
```

```
        self.display_message(\
```

```
            overlay , "Ready , press enter to begin printing process .")
```

```
    else:
```

```
"""
```

```
Navigierungs und Spray Funktion .
```

```
"""
```

```
    self.print(markers)
```

```
else:
```

```
"""
```

6 Steuerung

```
Manage image scale berechnet das Bild so,  
dass es in den zuvor berechneten Canvas Rand hineinpasst.  
Zudem findet es den relativen Spray Pfad.
```

```
"""
```

```
self.manage_image_scale(markers)
```

6.3.4 Steuerung

Für die Steuerung wird zuerst der Druckpfad im Bild berechnet. Durch die Funktion cv2.Canny() kann OpenCV das gewünschte Bild untersuchen. Wie oben beschrieben, bekommt man durch die Funktion ein schwarz-weiss Bild zurück, dessen weisse Pixel den Zielbereich indizieren. Damit der Roboter nicht 30 Pixel an einem Ort versucht zu drucken, werden die Abstände in der Realität vermessen. Die Zentimeter werden umgerechnet und alle Pixel innerhalb des bestimmten Radius werden gruppiert. Diese Punkte werden dann in einem Array gespeichert und dienen dem Roboter als Referenz. Damit sich der Roboter orientieren kann, werden die Abstände zwischen dem linken Motor und der linken Wandmarkierung, sowie des rechten Motors und der rechten Wandmarkierung gemessen. Zudem werden noch die Abstände der Motoren zum horizontalen Zentrum des Roboters gemessen.

Damit wir die gewünschte Position der Motoren ermitteln können, werden nun die frisch berechneten Motorenabstände zum Zentrum des nächsten Druckpunktes, für den linken Motor subtrahiert und für den rechten addiert. Dies resultiert in der gewünschten Position der Motoren. Aus dieser Position, werden die Abstände erneut gemessen und verglichen. Anhand dieses Vergleichs wird über eine weitere Motorsteuerung entschieden (korrekte Position, oder Neuberechnung). Ist der Abstand zu gross, wird der Motor zum Aufwickeln des Seiles aufgefordert. Ist der Abstand jedoch zu klein, folgt ein Kommando zum Abwickeln des Seiles.

Hat der Motor die gewünschte Position innerhalb eines gewissen Abstandes (z.B 2cm) erreicht, so stoppt dieser seine Steuerung und wartet lediglich noch bis auch der andere Motor die gewünschte Position erreicht hat. Sind beide Motoren angekommen, wird der Befehl zum Drucken aufgerufen und der nächste Punkt im Druckpfad angesteuert.

7 Schlusswort

6.3.5 Probleme

Bei unserer Navigation gab und gibt es hauptsächlich 4 Probleme:

1. Lichteinfall auf die Markierungen.

Wird das Zimmer zu stark beleuchtet, so hat unser Erkennungsalgorithmus Probleme, die Markierungen korrekt zu erkennen. Dies liegt daran, dass die Einstellungen der Kamera nicht auf Überbelichtung ausgelegt sind und sich Blende und ISO-Empfindlichkeit nicht zuversichtlich ändern. Bei Überbelichtung können die LEDs von anderen hellen Gegenständen (z.B. der Wand) nicht unterschieden werden.

2. Druckpfad richtig und schnell erkennen.

Ein Bild kann sehr schnell hochauflösend sein und besteht somit aus vielen Pixel. Jedes Pixel soll auf die Farbe weiss und gleichzeitig auf den Abstand aller anderen weissen Pixel in der Nähe abgesucht werden, damit der Roboter nicht mehrmals am gleichen Ort druckt.

3. Winkelausrichtung der Kamera.

Wenn die Kamera schräg ausgerichtet ist, so wirkt das auf die Präzision des Roboters aus.

4. Neigung des Roboters.

Je näher der Roboter am linken oder rechten Seitenrand ist, desto stärker neigt er sich. Dies wirkt sich ebenfalls negativ auf die Präzision des Druckvorgangs aus.

7 Schlusswort

7.1 Auswertung des Produktes

Ursprünglich wollten wir einen Druckroboter entwickeln, der mit drei Farben drucken kann. Zusätzlich sollte er einen Rahmen von 4m x 4m bedrucken können. Die Präzision sollte +/- 3cm betragen. Um den Druckvorgang einfach zu gestalten wollten wir eine graphische Benutzeroberfläche dazu entwickeln, diese ist Zeitmangel zum Opfer gefallen. Zeitmangel war ebenfalls der Grund, weshalb wir den als Ziel vorgesetzten, grossen Bildrahmen nicht testen konnten. Unser Testrahmen blieb 1.5m x 1m

7 Schlusswort

gross. Als erstes Bild (vgl. Abbildung 59) haben wir ein Viereck, ein Dreieck und einen Text gedruckt, um einen allfälligen Verzug zu messen. In der Abbildung 60 ist das Ergebnis des ersten Ausdrucks ersichtlich. Je seitlicher die Druckpunkte, desto markanter wird ein Verzug. Dieser Verzug entsteht durch eine Neigung, die nicht in der Steuerung berücksichtigt wird. Hingegen wurde die Präzision erfüllt, wir massen +/- 1 cm (Ausreisser nicht berücksichtigt). Die Geschwindigkeit war für unsere Testgrösse ideal.

Insgesamt haben wir mit dieser Arbeit viel gelernt. Durch das bereichsübergreifende Arbeiten konnten wir alle vieles voneinander mitnehmen. Neu für uns alle war der Bereich der Mechanik. Zudem handelte es sich bei unserem Projekt um bewegte Mechanik, die anspruchsvoller war, als gedacht. Dies bedeutet aber auch lehrreicher als gedacht. Dieses Projekt reichte bis in die Robotik, die Arbeit mit OpenCV legt sicher eine stabile Grundlage für kommende etwas überrissene Projektideen. Unser Konzept ist somit zufriedenstellend aufgegangen. Wir können drucken und erreichen dies besser, als gedacht. Um das Ganze zu perfektionieren, könnten wir die Steuerung überarbeiten, sodass wir alle drei Farben drucken könnten, unter Berücksichtigung der Neigung. Durch eine Steigerung der Motorkraft könnten wir Vektorbilder drucken. Eine grafische Benutzeroberfläche könnte die Bedienungsfreundlichkeit verbessern.

7.2 Auswertung Arbeitsprozess

Wir als Gruppe konnten beim Arbeitsprozess der Projektarbeit gemeinsam wachsen. Zu Beginn der Arbeit haben wir viel in die Planung investiert, aber teilweise in die falschen Aspekte. Wir haben viel Zeit in die Projektfindung gesteckt, bis wir uns erst einmal finden konnten. Diese Zeit hätten wir schon vorab in die wer-, was-, wann-, wie-Fragen investieren können. Nichtsdestotrotz blicken wir sehr positiv auf unseren kreativen Projektstart zurück - wir hatten viele verschiedene Ideen und sahen viele Möglichkeiten. Wir sind am Prozess gewachsen, dadurch dass wir bei Rückschlägen konstruktiv reagiert haben und gemeinsam eine Lösung gefunden haben. Die Arbeitsaufteilung war nicht ideal gewählt. Der Hardware-Teil (Mechanik und Elektronik) des Produkts wurde durch David realisiert. Die Software haben Nils und Leonie gemeinsam gestartet und aufgeteilt, durch Abwesenheiten wurde die Zusammenarbeit schwierig und endete damit, dass der Software-Teil grösstenteils an Nils zu übergeben. Dies war auch ein Problem des Wohnorts, Nils und David wohnen beide in der Region Baden, Leonie

8 Glossar

lebt im Kanton Bern. Die gemeinsamen Treffen waren dadurch erschwert, da sich der Artomat konkret bei David zu Hause befand. Die Schwierigkeit auf die wir trafen, war die Komplexität, sprich die Grösse des Projektes. Hätten wir ein einfacheres Produkt mit weniger Arbeitsschritten und Teilbereichen gewählt, wäre es sicher perfekter geworden. Insgesamt sind wir aber sehr zufrieden.

8 Glossar

Begriff / Abkürzung	Definition / Erklärung
MDF	Mitteldichte Holzplatte
Drehmoment	Drehkraft des Motors
PCB	Printed circuit board
GPIO	General purpose input output
Galvanischetrennung	Separieren von zwei Stromkreisen
SLA	Stereolithographie
FDM	Fused Deposition Modeling

Literatur

- [CG]Maxime (o. J.). *OpenCV Konturen Erkennung*. URL: <https://www.codingame.com/playgrounds/38470/how-to-detect-circles-in-images> [Stand: 02.08.2019].
- 101 (25. Nov. 2014). *OpenCV Fenster Updated*. URL: <https://stackoverflow.com/questions/27117705/how-to-update-imshow-window-for-python-opencv-cv2> [Stand: 02.08.2019].
- Adafruit (2017). *Large push-pull solenoid*. URL: <https://www.adafruit.com/product/413> [Stand: 26.07.2019].
- Alexander Mordvintsev, Abid K (2013). *Allgemeines OpenCV Tutorial (Hello World)*. URL: https://opencv-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html [Stand: 02.08.2019].

Literatur

- Bodnar, Jan (2019). *Drawing in Tkinter*. URL: <http://zetcode.com/tkinter/drawing/> [Stand: 15.07.2019].
- Dietz, Peter (Jan. 2015). *Qualität auf Knopfdruck*. Hrsg. von Guido Schiefer. URL: <https://www.c3h8-magazin.de/1-2015/qualitaet-auf-knopfdruck/?print=1> [Stand: 26.07.2019].
- Engelhardt, Erich F (2016). *Roboter mit Raspberry Pi: Mit Motoren, Sensoren, LEGO® und Elektronik eigene Roboter mit dem Pi bauen, die Spaß machen und Ihnen lästige Aufgaben abnehmen*. Franzis Verlag.
- Geerling, Jeff (2019). *Power Consumption Benchmarks*. URL: <https://www.pidramble.com/wiki/benchmarks/power-consumption> [Stand: 26.07.2019].
- Großig, Natalie (o.D.). *Die Spraydose: Aufbau, Funktionsweise und Mechanik*. Hrsg. von Tillman Strasburger. URL: <https://selbermachen.de/tools-tipps/die-spraydose-aufbau-funktionsweise-und-mechanik> [Stand: 26.07.2019].
- Hanus, Bo (2013). *Der leichte Einstieg in die Elektrotechnik & Elektronik: Bauteile der Elektrotechnik · Solartechnik · Netzgeräte · Motoren und Generatoren · Messgeräte · Beleuchtung*. Franzis Verlag.
- harshkn (8. Mai 2017). *OpenCV kleine Kreiserkennung*. URL: <https://stackoverflow.com/questions/43852023/detecting-small-circles-using-houghcircles-opencv> [Stand: 02.08.2019].
- IAmSuyogJadhav (2018). *OpenCV Bild Transparent übereinander fusionieren*. URL: <https://gist.github.com/IAmSuyogJadhav/305bfd9a0605a4c096383408bee7fd5c> [Stand: 02.08.2019].
- Ibrahim, Mohd Ikhwan (7. Jan. 2018). *Aerosol Spray*. Hrsg. von Mohd Ikhwan Ibrahim. URL: https://grabcad.com/library/aerosol-spray-2/details?folder_id=4703709 [Stand: 26.07.2019].
- Loo, Gert van (27. Mai 2019). *GPIO pads control*. Techn. Ber. University of Western Australia.
- Mathematik für Elektroniker/in für Automatisierungstechnik (2018). Europa-Lehrmittel. ISBN: 978-3-8085-3498-4.
- Motayed, Abhishek und S Noor Mohammad (2001). “Tuned performance of small-signal BJT Darlington pair”. In: *Solid-State Electronics* 45.2, S. 369–371.
- Murugavel, Manivannan (14. Dez. 2017). *OpenCV Bild mit Numpy darstellen*. URL: https://medium.com/@manivannan_data/drawing-image-using-numpy-and-opencv-565abdbb3670 [Stand: 02.08.2019].

Literatur

- Niemeyer, Clemens (2005). *Motorsteuerung mit Microcontrollern*. Techn. Ber. Technische Universität München.
- o.V. (o.J.[a]). *Capabilities*. URL: <https://jlcpcb.com/capabilities/Capabilities> [Stand:01.07.2019].
- (o.J.[b]). *How to export Altium PCB to gerber files*. URL: <https://support.jlcpcb.com/article/42-how-to-export-altium-pcb-to-gerber-files> [Stand:01.07.2019].
- (o.J.[c]). *Power Supply*. URL: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md> [Stand:12.07.10].
- (Nov. 1999). *LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator*. Techn. Ber. Texas Instruments.
- (Feb. 2000). *BD243/A/B/C*. Techn. Ber. Fairchild Semiconductor.
- (Jan. 2001). *BC237/238/239*. Techn. Ber. Fairchild Semiconductor.
- (Sep. 2002). *BDW54, BDW54A, BDW54B, BDW54C, BDW54D PNP SILICON POWER DARLINGTONS*. Techn. Ber. Bourns.
- (2005). *5 Watt Surmetic 40 Zener Voltage Regulators*. Techn. Ber. On Semiconductor.
- (März 2007). *High Voltage Transistors*. Techn. Ber. ON Semiconductor.
- (Mai 2011a). *BC307B*. Techn. Ber. ON Semiconductor.
- (19. Juli 2011b). *Elektromagnete von Gruner – kundenspezifische Ausführungen*. Techn. Ber. GRUNER AG.
- (23. Okt. 2015). *ILD205T, ILD206T, ILD207T, ILD211T, ILD213T*. Techn. Ber. Vishay.
- (12. Dez. 2016). *LM341 and LM78M05 Series 3-Terminal 500-mA Positive Voltage Regulators*. Techn. Ber. Texas Instruments.
- (Juni 2017a). *Large push-pull solenoid*. Techn. Ber. adafruit Industries.
- (6. Juli 2017b). *Small Signal Fast Switching Diodes*. Techn. Ber. Vishay.
- (Juni 2018). *Seven Darlington arrays*. Techn. Ber. STMicroelectronics.
- (2. Apr. 2019a). *236-100 - Abschlussplatte 1 mm dick anrastbar*. URL: https://wago.partcommunity.com/3d-cad-models/sso/236-400-wago?info=wago%2Fpg04%2Fserie236%2F0236-0100_0999-0962.prj&cwid=6818 [Stand:26.07.2019].

Literatur

- o.V. (16. Jan. 2019b). *Anreihbare Einzelklemme für Leiterplatten 2 Lötstifte/Pol 1-polig Rastermaß 5/5,08 mm / 0.2 in.* URL: https://wago.partcommunity.com/3d-cad-models/sso/236-743-999-950-wago?info=wago%2Fpg04%2Fserie236%2F0236-0401_0999-0962.prj&cwid=6818 [Stand: 26.07.2019].
- (17. Juli 2019c). *Raspberry Pi: Grundlagen der Energieversorgung / Stromversorgung.* URL: <https://www.elektronik-kompendium.de/sites/raspberry-pi/1912111.htm> [Stand: 26.07.2019].
- (o.D.[a]). *33GN2738-132-GV-5 312:1.* Techn. Ber. MOTRAXX ELEKTROGERAETE GmbH.
- Rosebrock, Adrian (21. Juli 2014). *OpenCV Kreis Erkennung Tutorial.* URL: <https://www.pyimagesearch.com/2014/07/21/detecting-circles-images-using-opencv-hough-circles/> [Stand: 02.08.2019].
- Sadekar, Kaustubh (11. Feb. 2019). *OpenCV Farberkennung Tutorial.* URL: <https://www.learnopencv.com/invisibility-cloak-using-color-detection-and-segmentation-with-opencv/> [Stand: 02.08.2019].
- Sanden, Jon van der (12. Dez. 2013). *Filament spool.* URL: <https://grabcad.com/library/filament-spool-1> [Stand: 26.07.2019].
- Schaerer, Thomas (26. Juni 2019a). *Bipolarer Transistor.* URL: <https://www.elektronik-kompendium.de/sites/bau/0201291.htm> [Stand: 26.07.2019].
- (26. Juni 2019b). *Spannungsstabilisierung mit Z-Diode und Transistor (Kollektorschaltung).* URL: <https://www.elektronik-kompendium.de/sites/slt/0204131.htm> [Stand: 26.07.2019].
- Schimpf, Paul H. (2013). *A Detailed Explanation of Solenoid Force.* Techn. Ber. Eastern Washington University.
- Smith, Dave W. (12. Aug. 2017). *RGB zu BRG OpenCV Verwirrung.* URL: <https://stackoverflow.com/questions/44693507/image-color-changed-after-converting-from-numpy-array-to-pil-image-python> [Stand: 02.08.2019].
- V., o. (o. J.). *OpenCV Kontur Erkennung.* URL: https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html [Stand: 02.08.2019].
- (2019d). *OpenCV Formen zeichnen.* URL: https://docs.opencv.org/3.1.0/dc/da5/tutorial_py_drawing_functions.html [Stand: 02.08.2019].

Abbildungsverzeichnis

- V., o. (o.D.[b]). *OpenCV Kreiserkennung Tutorial*. URL: https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Hough%20Circle_Transform.php [Stand: 02.08.2019].
- Vobiscum, Pax (8. Aug. 2018). *Falsche Canvas Grösse Export von Tkinter*. URL: <https://stackoverflow.com/questions/51742871/how-to-maintain-canvas-size-when-converting-python-turtle-canvas-to-bitmap?rq=1> [Stand: 30.06.2019].
- Weisenhorn, Martin (24. Mai 2016). *Laborübung, H-Brücke für DC-Motor*. Techn. Ber. Zurich University of Applied Sciences.
- Wild, David (27. Mai 2019a). *Artomat contoller v1*. Unveröffentlichtes Dokument, IDPA Artomat. Hapa ag.
- (16. Juli 2019b). *Druckmechanismus*. Techn. Ber. Hapa ag.
- (5. Juli 2019c). *Laborbericht Dimensinierung R9*. Unveröffentlichtes Dokument, IDPA Artomat. Hapa ag.
- (25. Juni 2019d). *Rollenbefestigung*. Unveröffentlichtes Dokument, IDPA Artomat. Hapa ag.

Abbildungsverzeichnis

1	Übersicht Artomat	7
2	Kennlinie 33GN2738-132-GV-5 312:1o.V. o.D.(a)	12
3	Motoren Befestigung	56
4	Motorkappe Anschluss des Motor	57
5	Kupplung von Motor zu Achse	57
6	Achse im Artomat	58
7	Erstkonstruktion von der Seilrolle Sanden 2013Wild 2019d	59
8	Seilrolle mit reduziertem Durchmesser	60
9	Achsen klemme V1	61
10	Achsen klemme V1 zerbrochen	62
11	Achsen klemme V2	63
12	Achsen klemme V2 zerbrochen	64

Abbildungsverzeichnis

13	Kerbe in Seilrolle	65
14	Seilführung	66
15	Aufbau einer BuntsprühlackdoseDietz 2015	67
16	DruckmechanismusWild 2019b	68
17	Schätzung der Sprühauslösekraft über der Eintauchtief	68
18	Sprühkopf mit blauer MarkierungIbrahim 2018	69
19	Sprühkopfhalte	70
20	Teller, teil des Druckmechanismus	71
21	Magnete halten Teller in Position	72
22	Gewicht am Seil wirkt wie ein Pendel	73
23	Drehbewegung aufgrund von Wandabstandhalter	74
24	Zeichnung der Geschätzten Drucklinie	75
25	3D Modell des Wandabstandhalters	76
26	Wandbefestigung mit abstand	77
27	UV - Marker an der Wand	78
28	Schaltbild einer H-Brücke Weisenhorn 2016	78
29	Ausschnitt aus Schema Artomat controller v1 Wild 2019a	79
30	Laborbericht Dimensionierung R9 grafisch dargestellt.Wild 2019c	79
31	Messung H-Brücke Ansteuerung Artomat controller v1 Wild 2019a	80
32	Messung H-Brücke Ansteuerung Artomat controller v1 Wild 2019a	81
33	Ausschnitt aus Schema Artomat controller v1 Wild 2019a	82
34	Messung Solenoid Ansteuerung Artomat controller v1 Wild 2019a	83
35	Kollektorschaltung	84
36	LM341-T in der Version 1	85
37	Tabelle Kriterien integrierte H-Brücke	86
38	Artomat Controllerboard V2 H-Brücken Messung	86
39	Artomat Controllerboard V2 H-Brücken Messung	86
40	Stromversorgung der Version 2	87
41	Dimensionierung Spule LM2596-5.0 o.V. 1999	88

9 Abbildungen

42	Dimensionierung Spule LM2596-12.0 o.V. 1999	89
43	12 V Ausgang von Stromversorgung V2	90
44	Funktion um den Druckmechanismus zu kontrollieren	91
45	Visuelle Erklärung	92
46	Simulation Ausgangslage	93
47	Linker Motor wird angezogen	94
48	Linker Motor wird angezogen	95
49	Linker Motor wird angezogen	96
50	Linker Motor wird angezogen	97
51	Motor Wanderlinie	98
52	Simulation Schwachpunkt	99
53	Simulation Resultat	100
54	Overlay der Vision	101
55	Vision Ablauf	102
56	HSV	103
57	Farbraum	104
58	Erkennung von Kreisen	105
59	Soll-Bild des erst Ausdruck	106
60	Erstes Bild Ergebnis	107

9 Abbildungen

9 Abbildungen

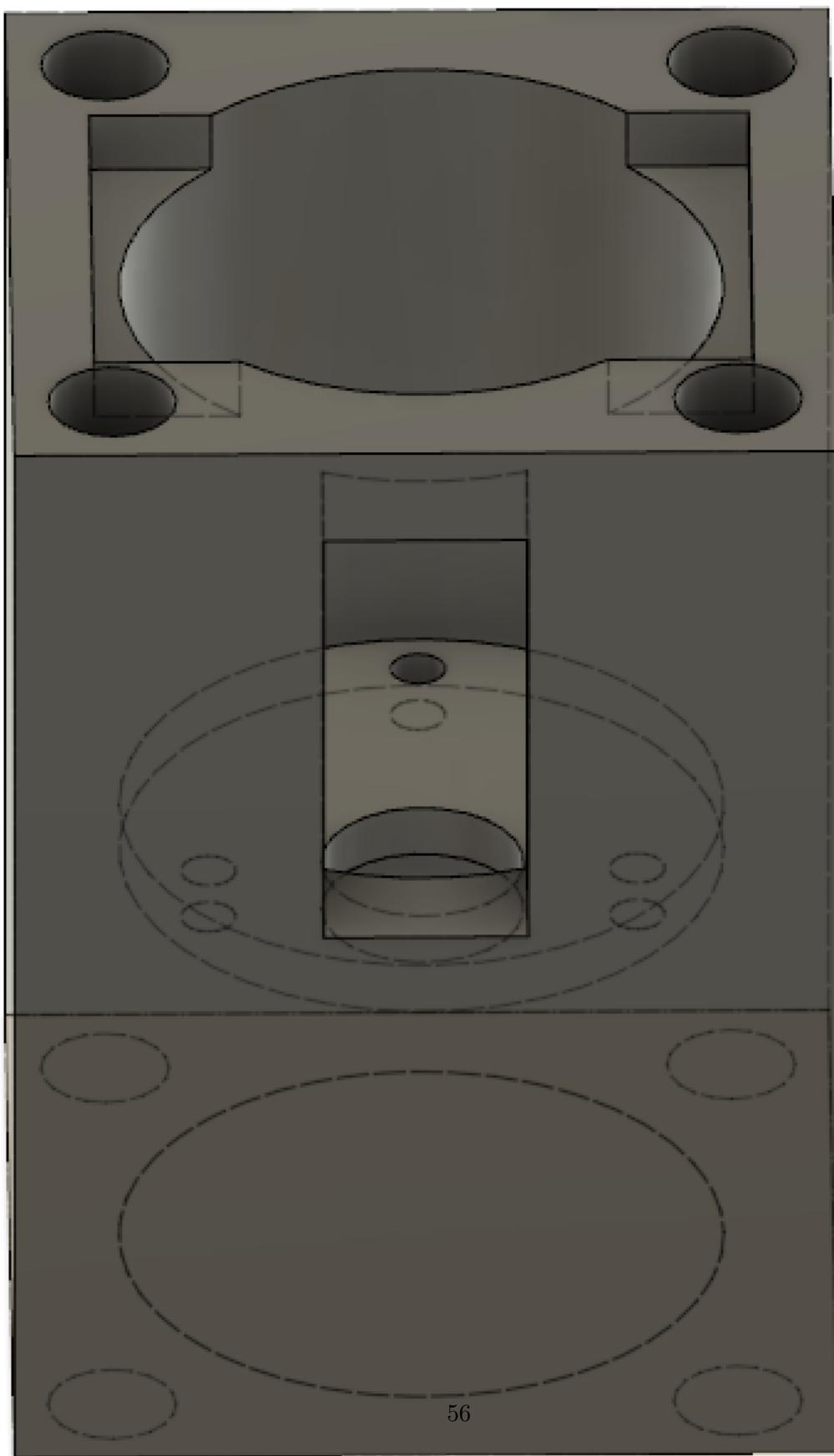


Abbildung 3: Motoren Befestigung

9 Abbildungen



Abbildung 4: Motorkappe Anschluss des Motor

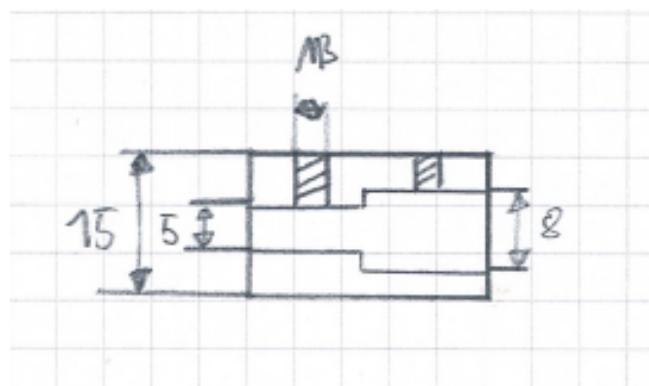


Abbildung 5: Kupplung von Motor zu Achse

9 Abbildungen



Abbildung 6: Achse im Artomat

9 Abbildungen



Abbildung 7: Erstkonstruktion von der Seilrolle Sanden 2013Wild 2019d

9 Abbildungen

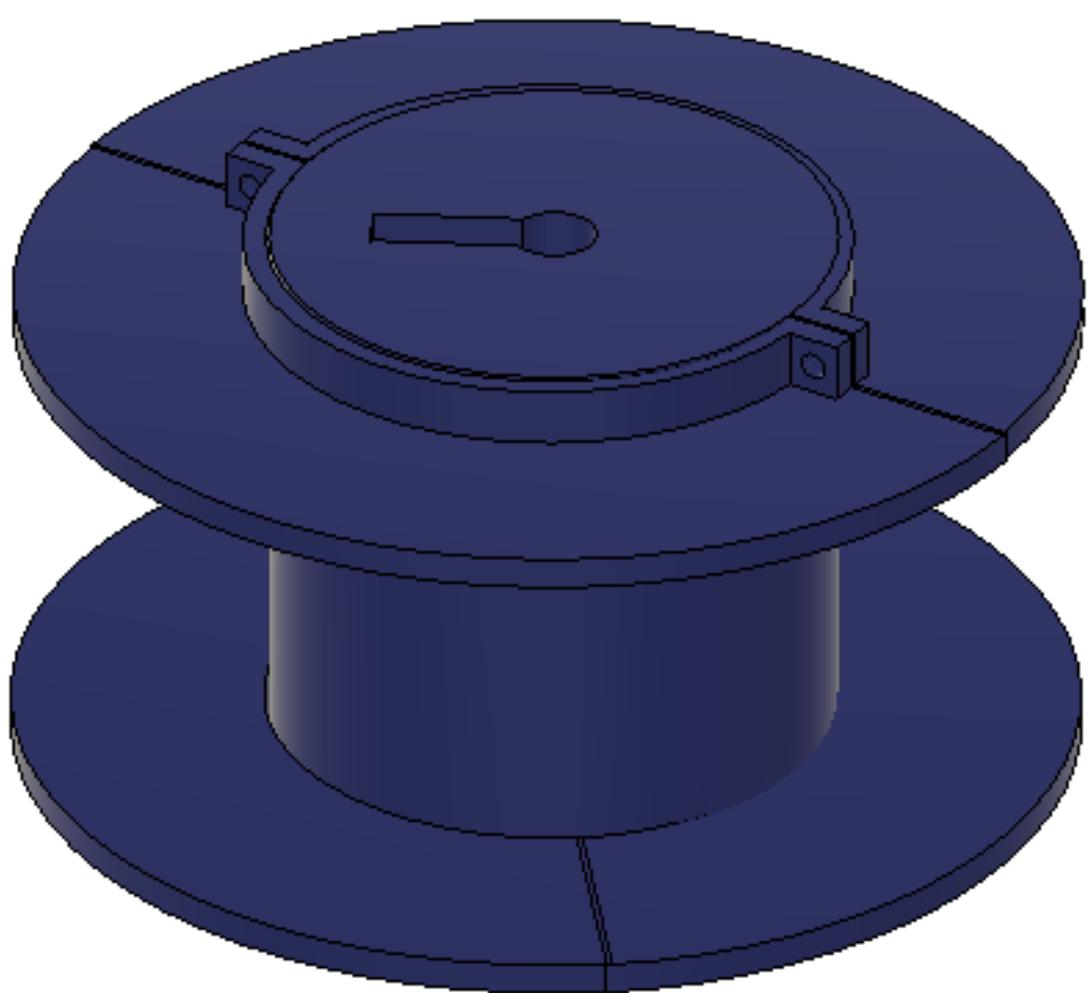


Abbildung 8: Seilrolle mit reduziertem Duchmesser

9 Abbildungen

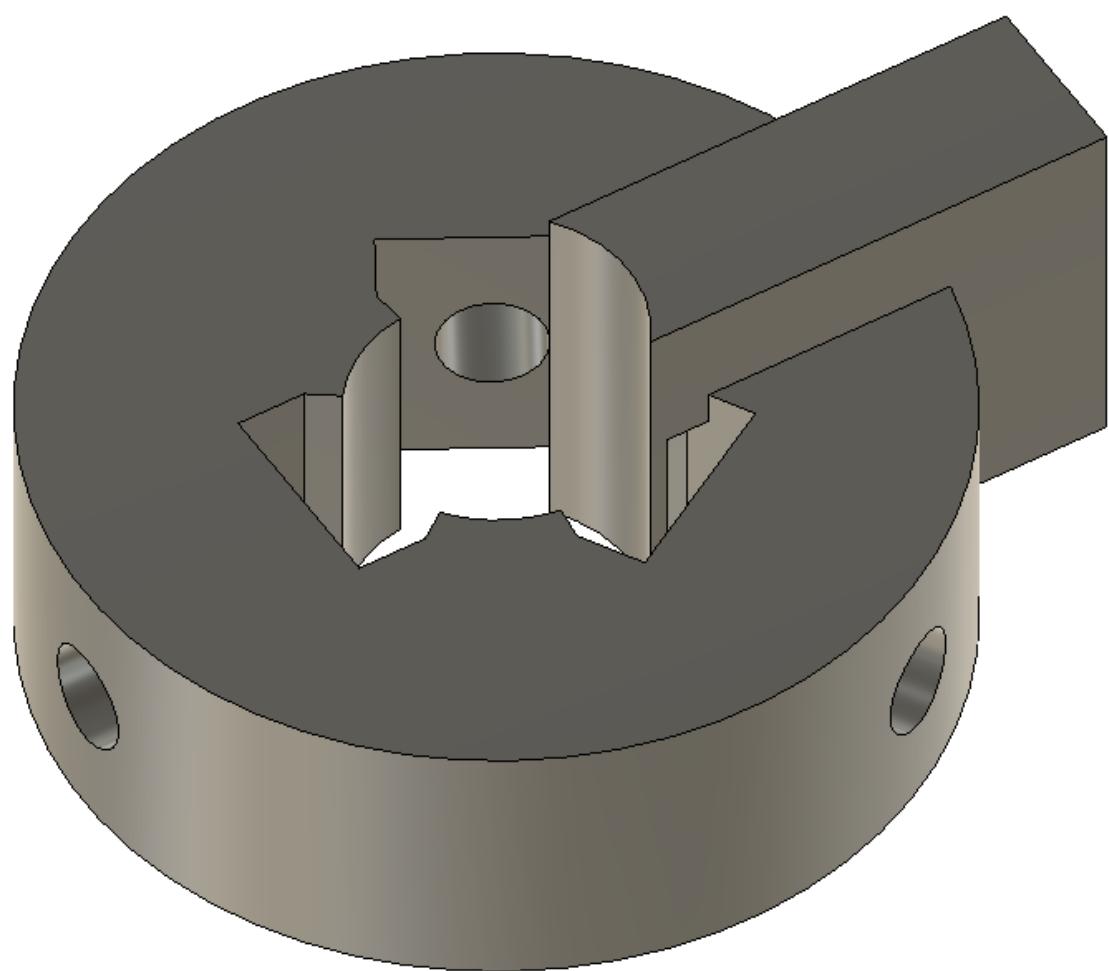


Abbildung 9: Achsen klemme V1

9 Abbildungen



Abbildung 10: Achsen klemme V1 zerbrochen

9 Abbildungen

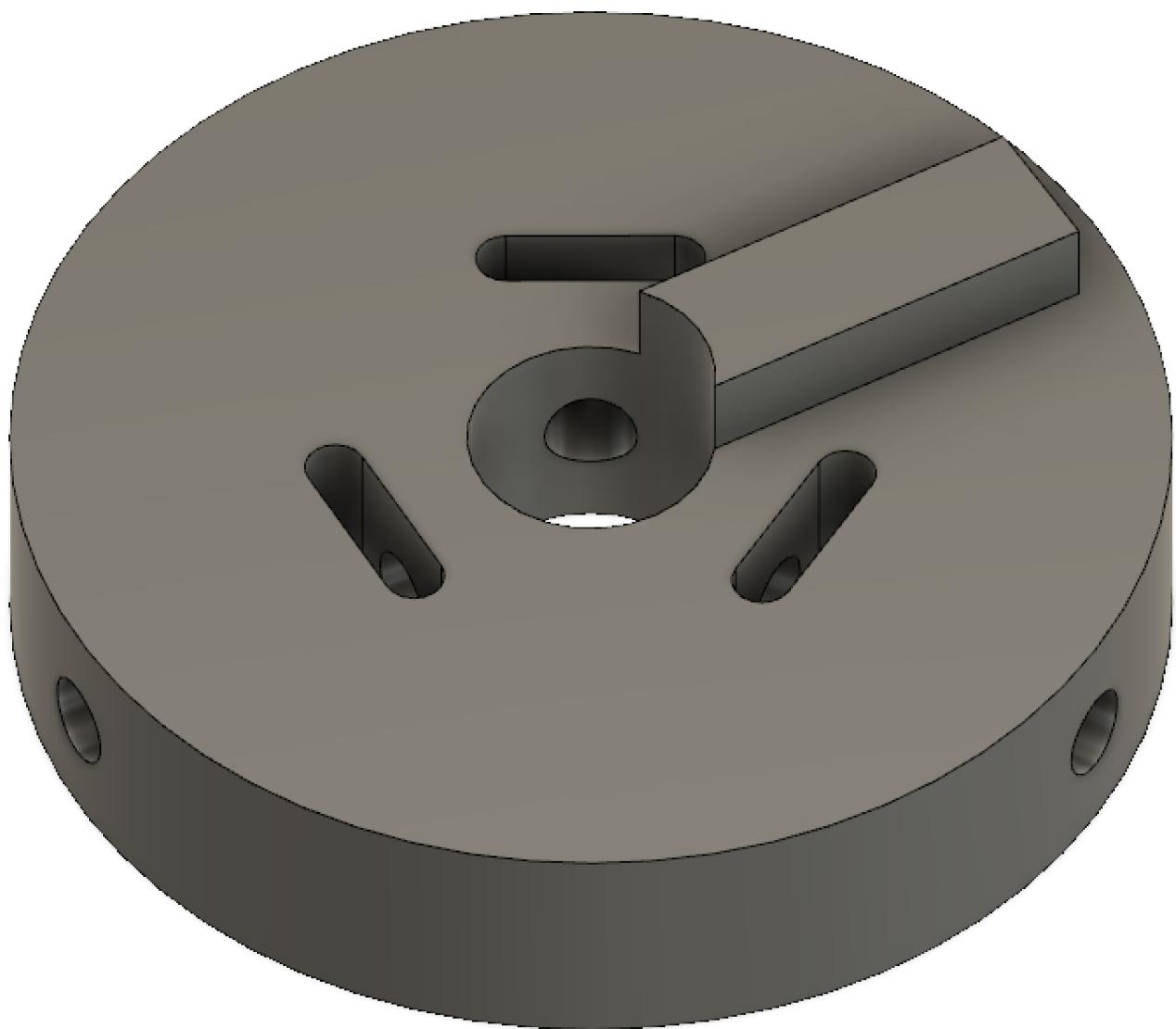


Abbildung 11: Achsen klemme V2

9 Abbildungen

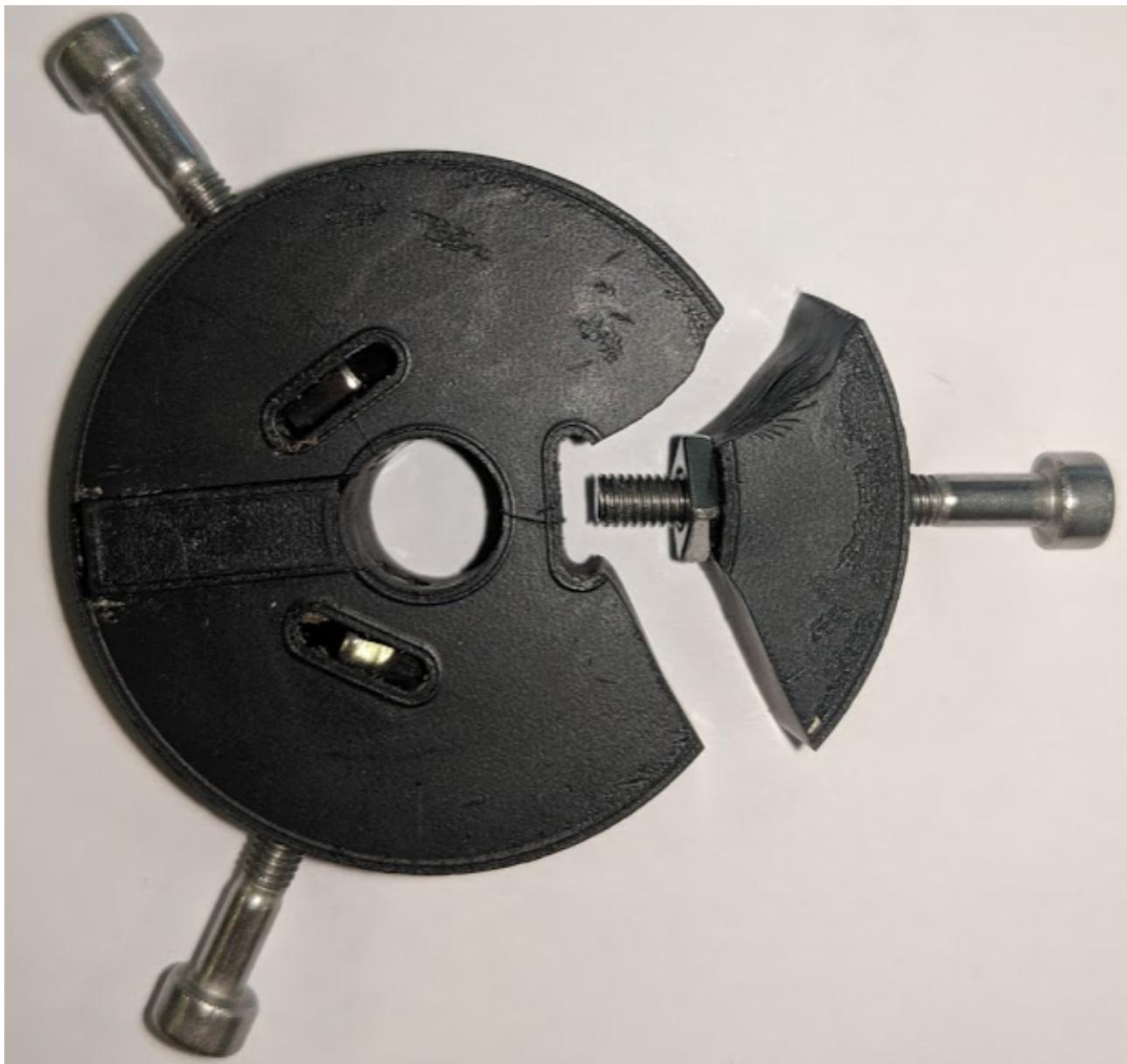


Abbildung 12: Achsen klemme V2 zerbrochen

9 Abbildungen

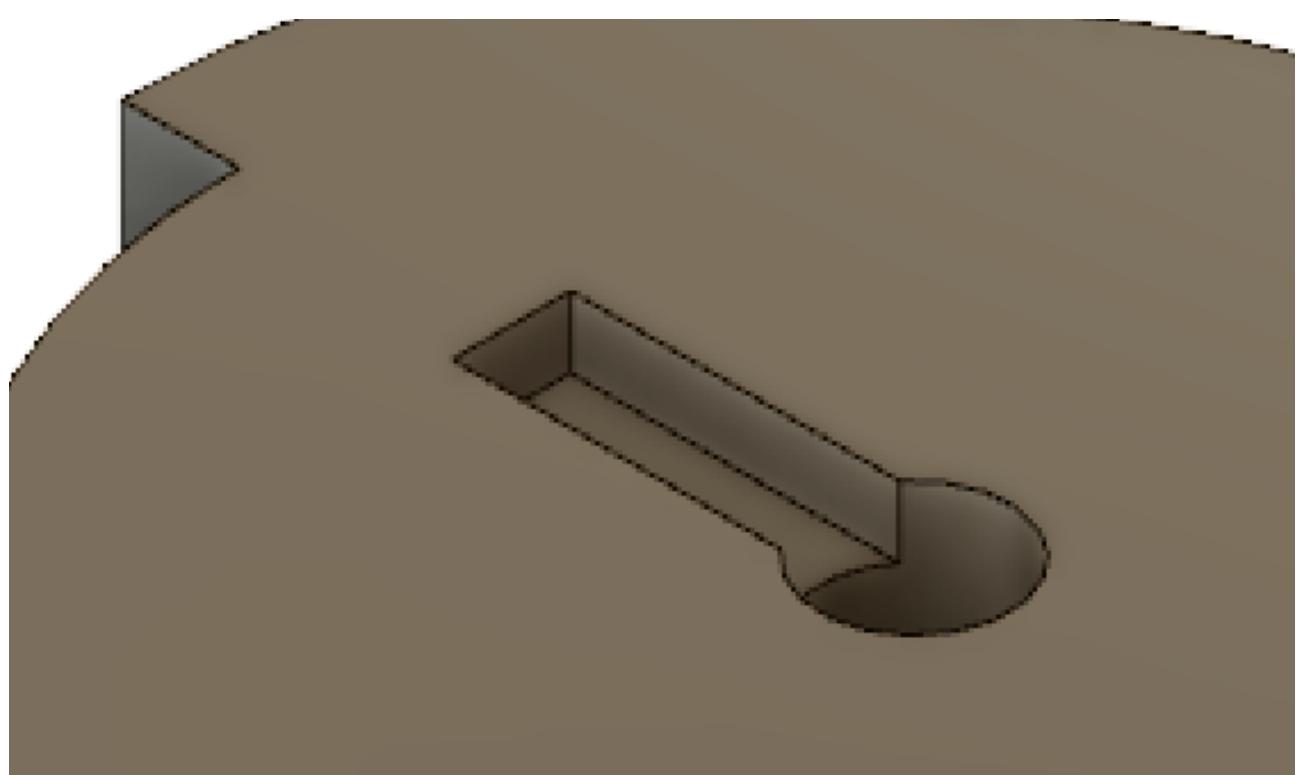


Abbildung 13: Kerbe in Seilrolle

9 Abbildungen



Abbildung 14: Seilführung

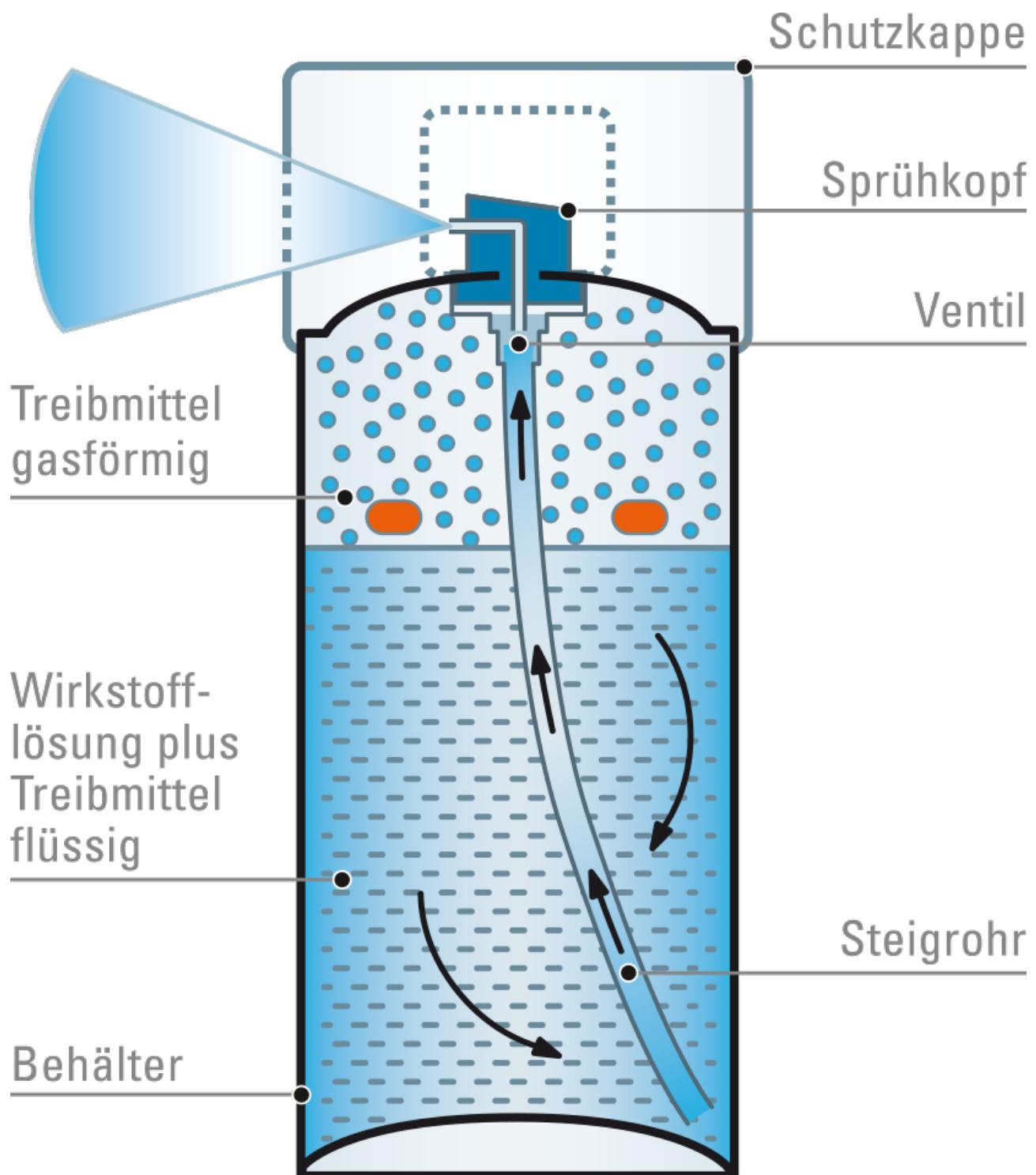


Abbildung 15: Aufbau einer Buntsprühlackdose Dietz 2015

9 Abbildungen



Abbildung 16: DruckmechanismusWild 2019b

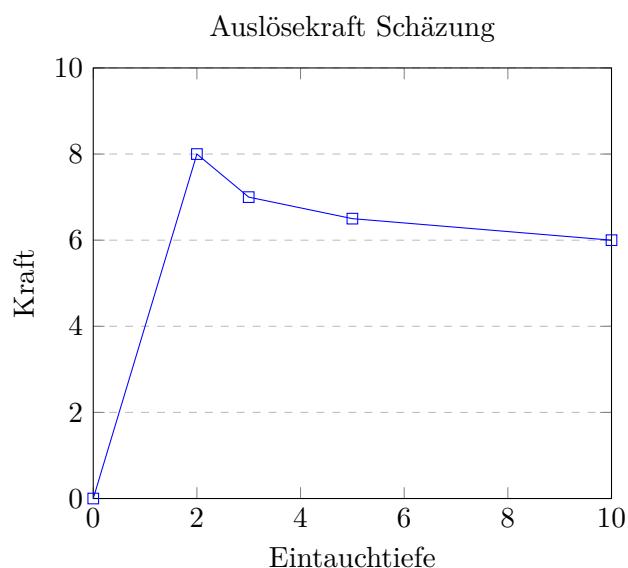


Abbildung 17: Schätzung der Sprühauslösekraft über der Eintauchtief

9 Abbildungen

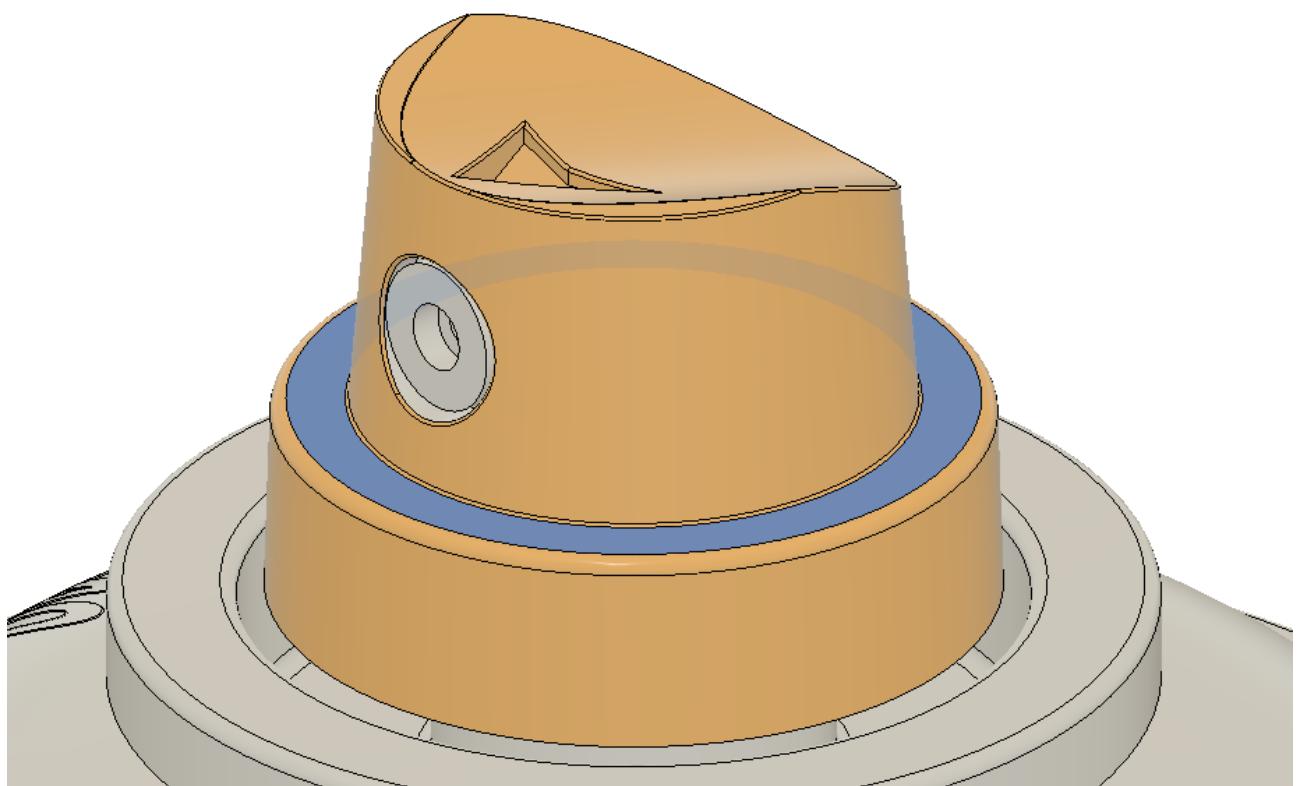


Abbildung 18: Sprühkopf mit blauer Markierung Ibrahim 2018

9 Abbildungen

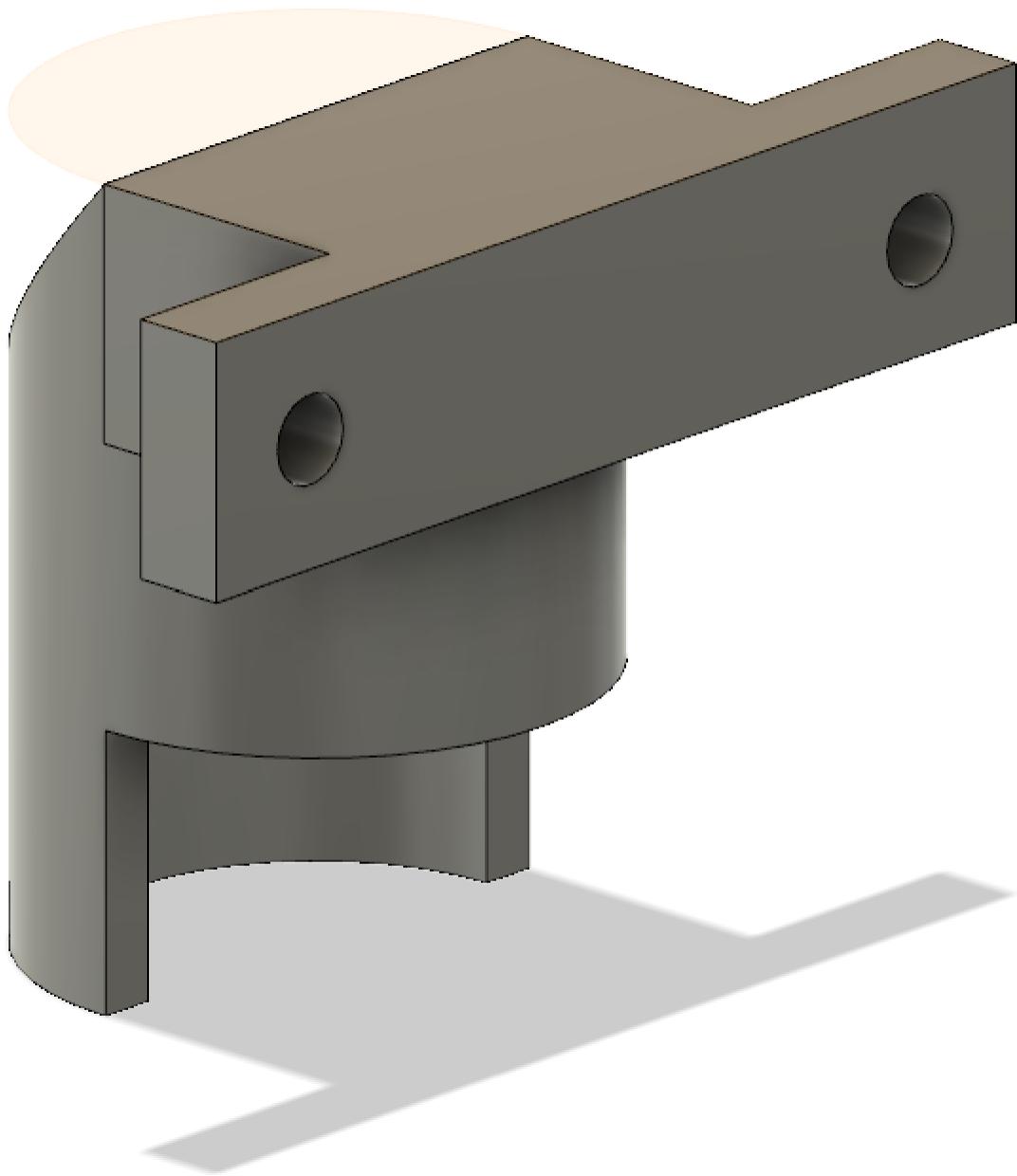


Abbildung 19: Sprühkopfhalte

9 Abbildungen

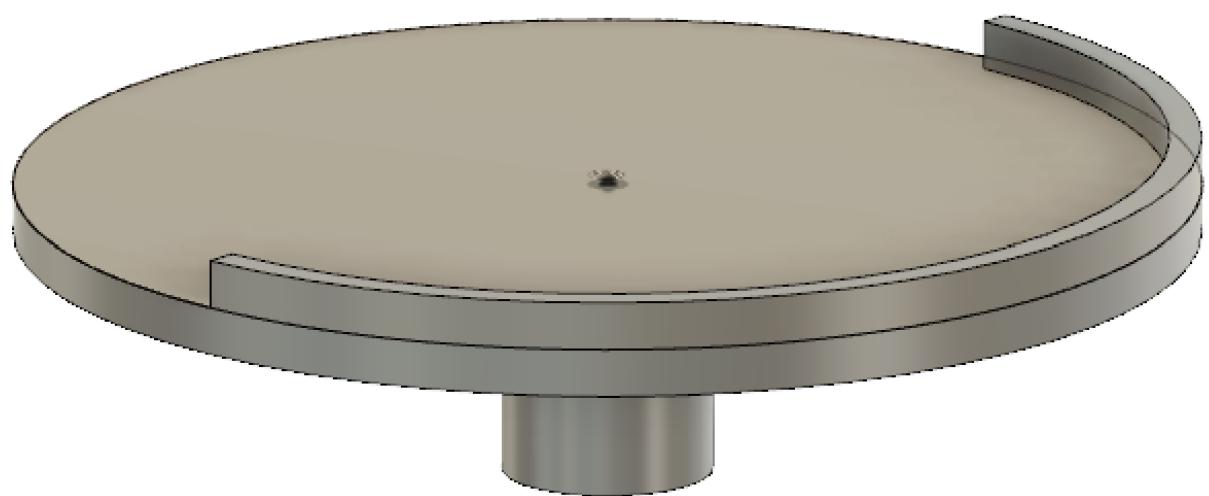


Abbildung 20: Teller, teil des Druckmechanismus

9 Abbildungen



Abbildung 21: Magnete halten Teller in Position

9 Abbildungen

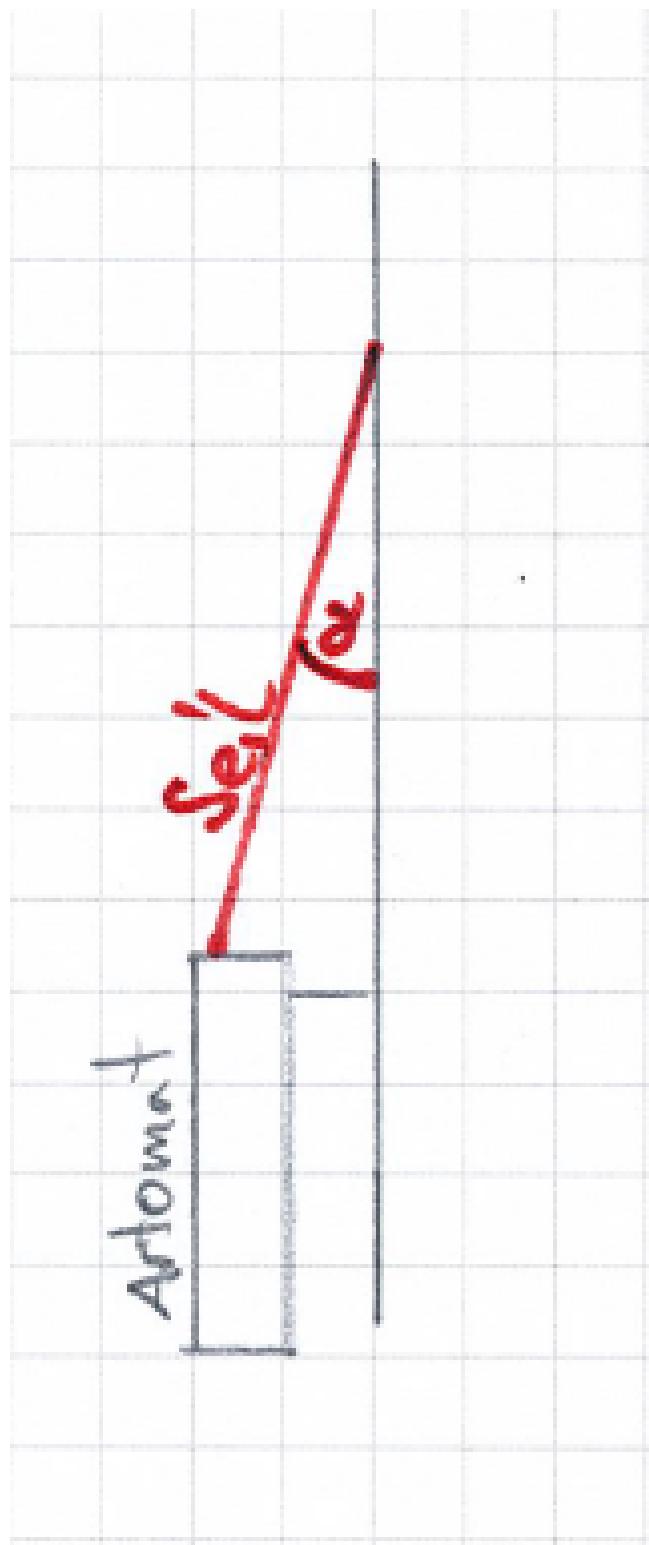


Abbildung 22: Gewicht am Seil wirkt wie ein Pendel

9 Abbildungen

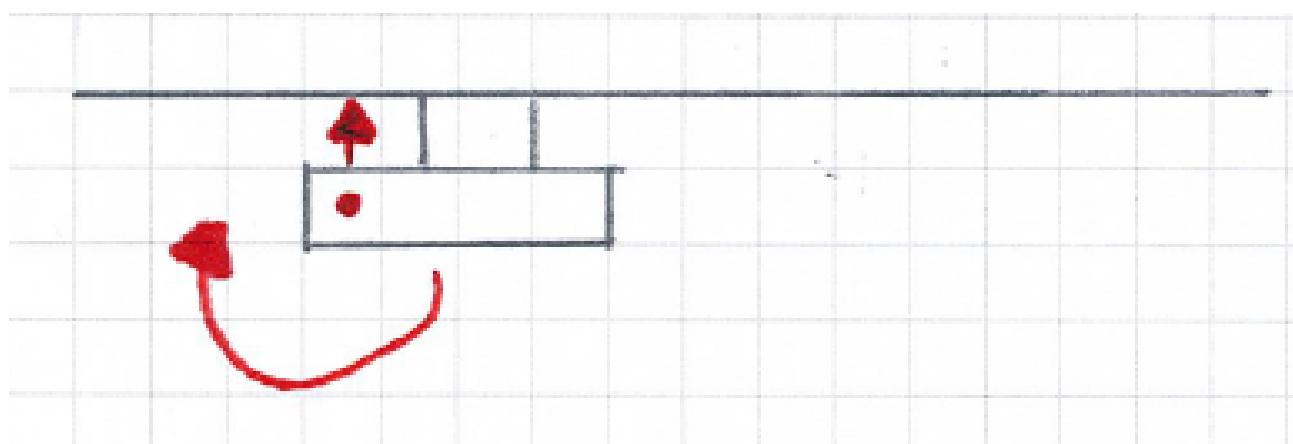


Abbildung 23: Drehbewegung aufgrund von Wandabstandhalter

9 Abbildungen

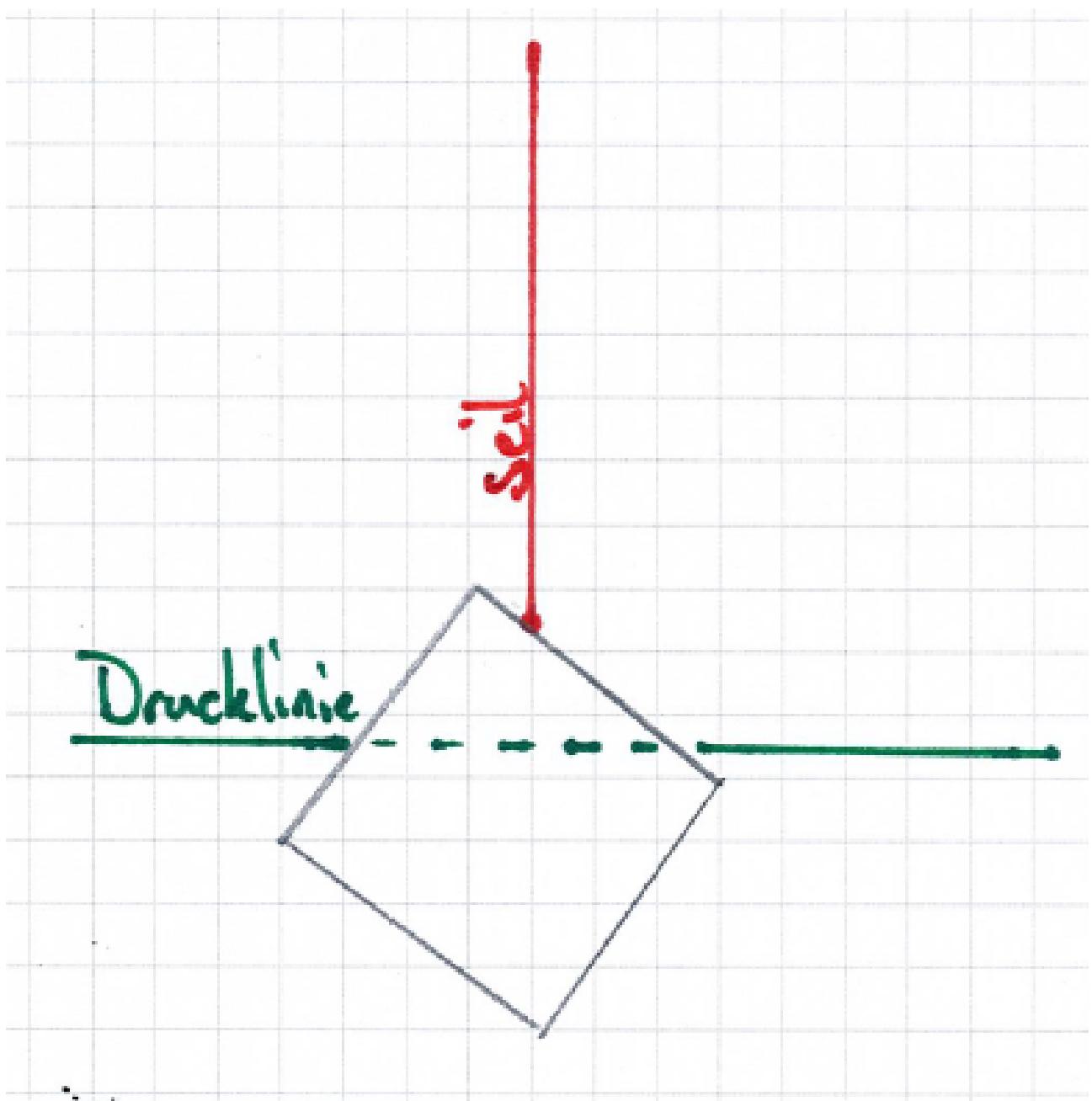


Abbildung 24: Zeichnung der Geschätzten Drucklinie

9 Abbildungen

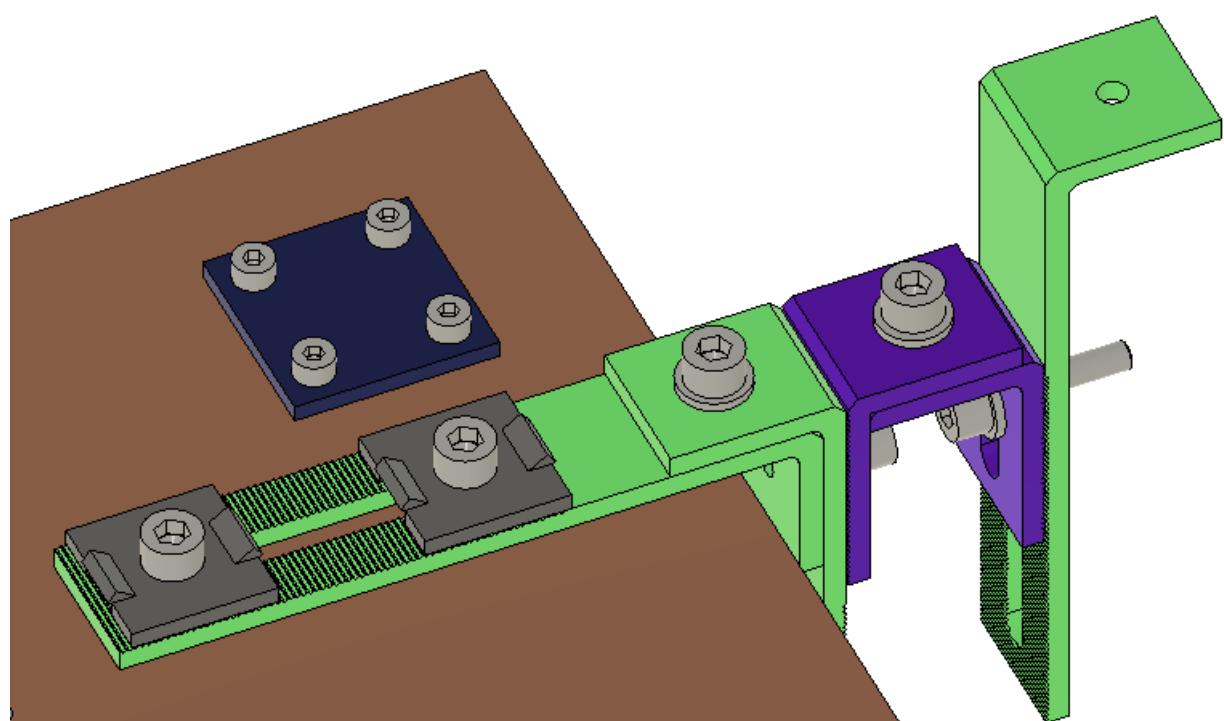


Abbildung 25: 3D Modell des Wandabstandhalters

9 Abbildungen

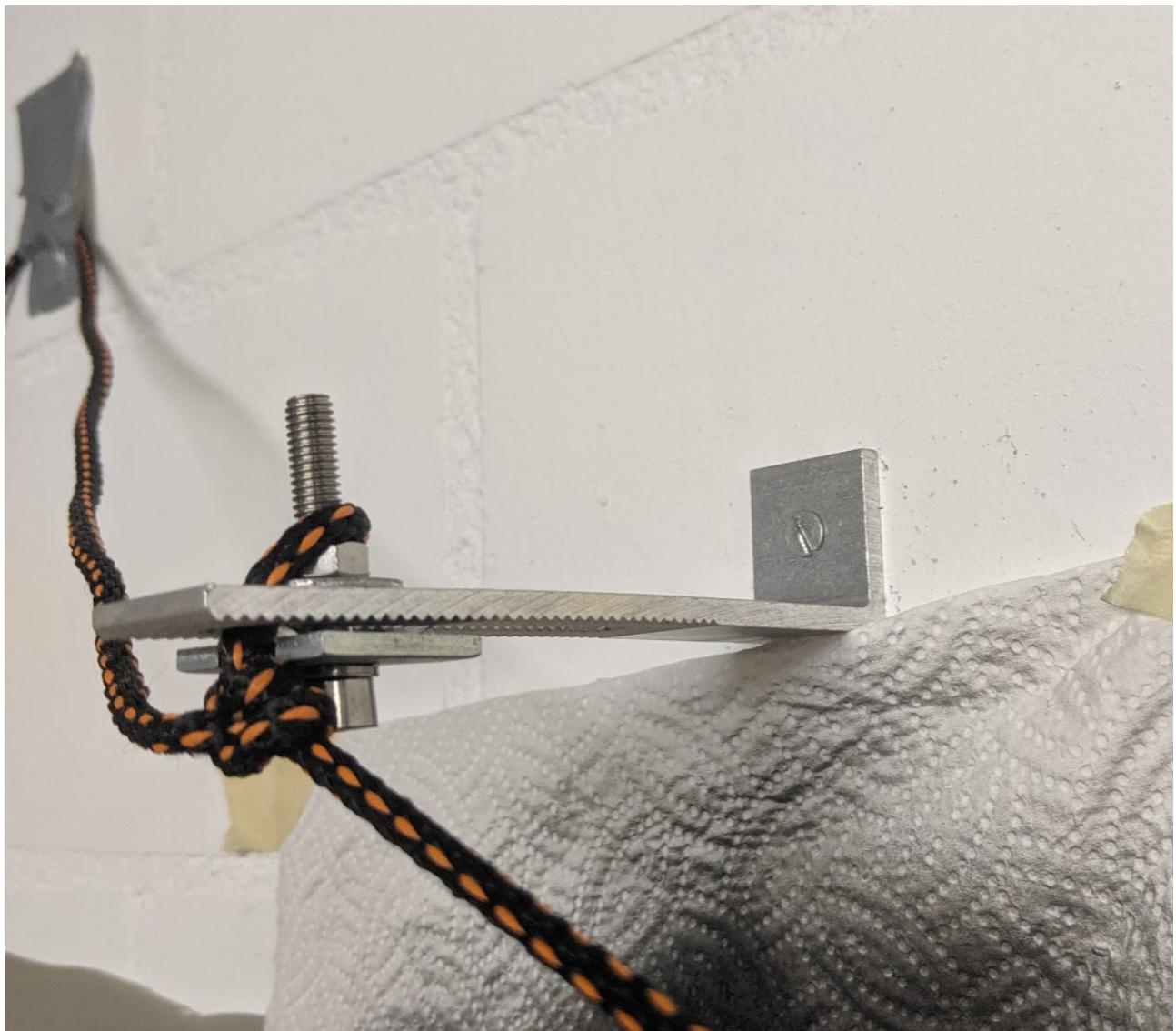


Abbildung 26: Wandbefestigung mit abstand

9 Abbildungen



Abbildung 27: UV - Marker an der Wand

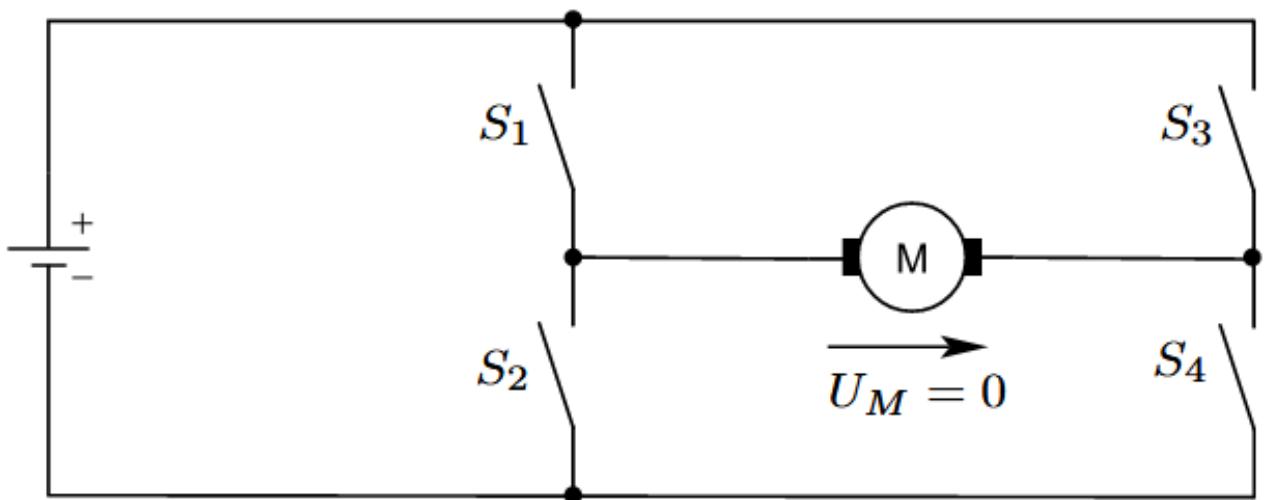


Abbildung 28: Schaltbild einer H-Brücke Weisenhorn 2016

9 Abbildungen

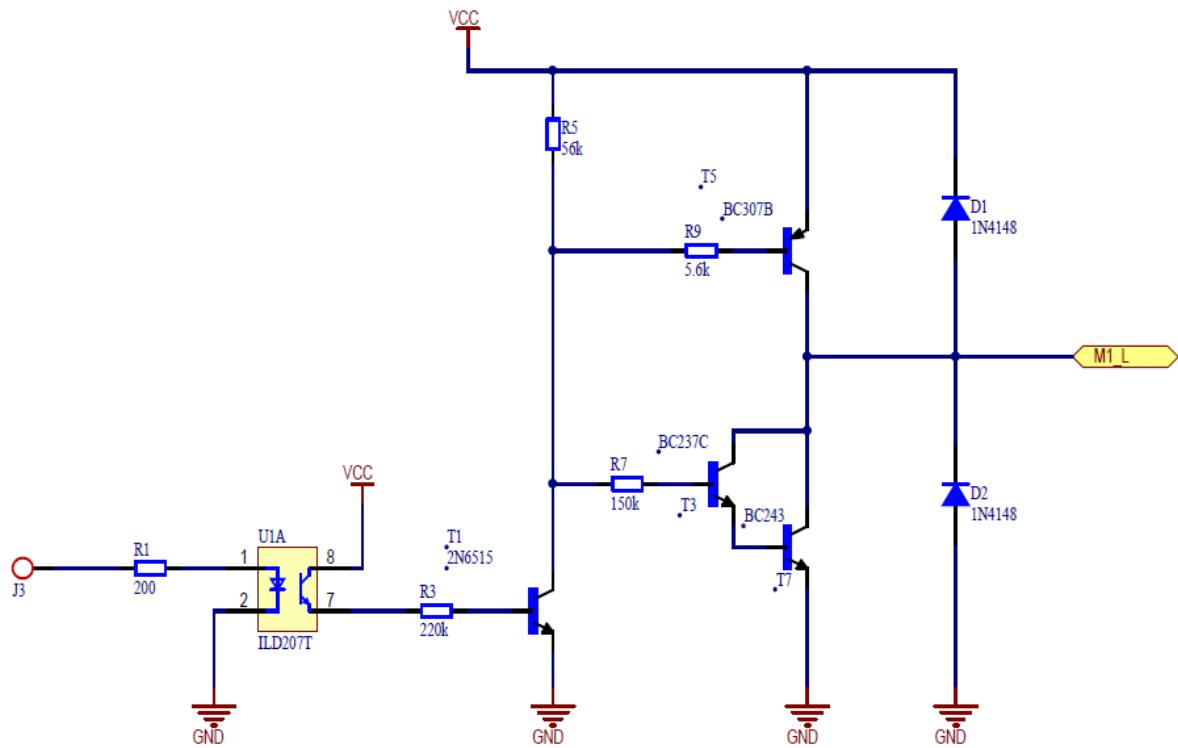


Abbildung 29: Ausschnitt aus Schema Artomat controller v1 Wild 2019a

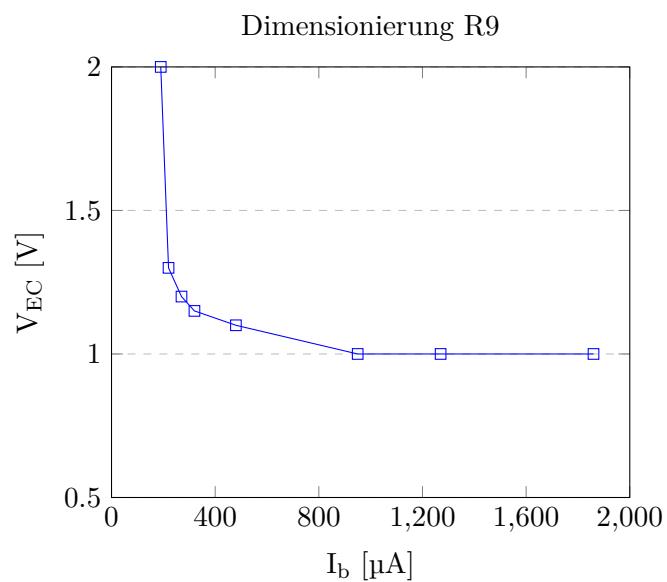


Abbildung 30: Laborbericht Dimensionierung R9 grafisch dargestellt.Wild 2019c

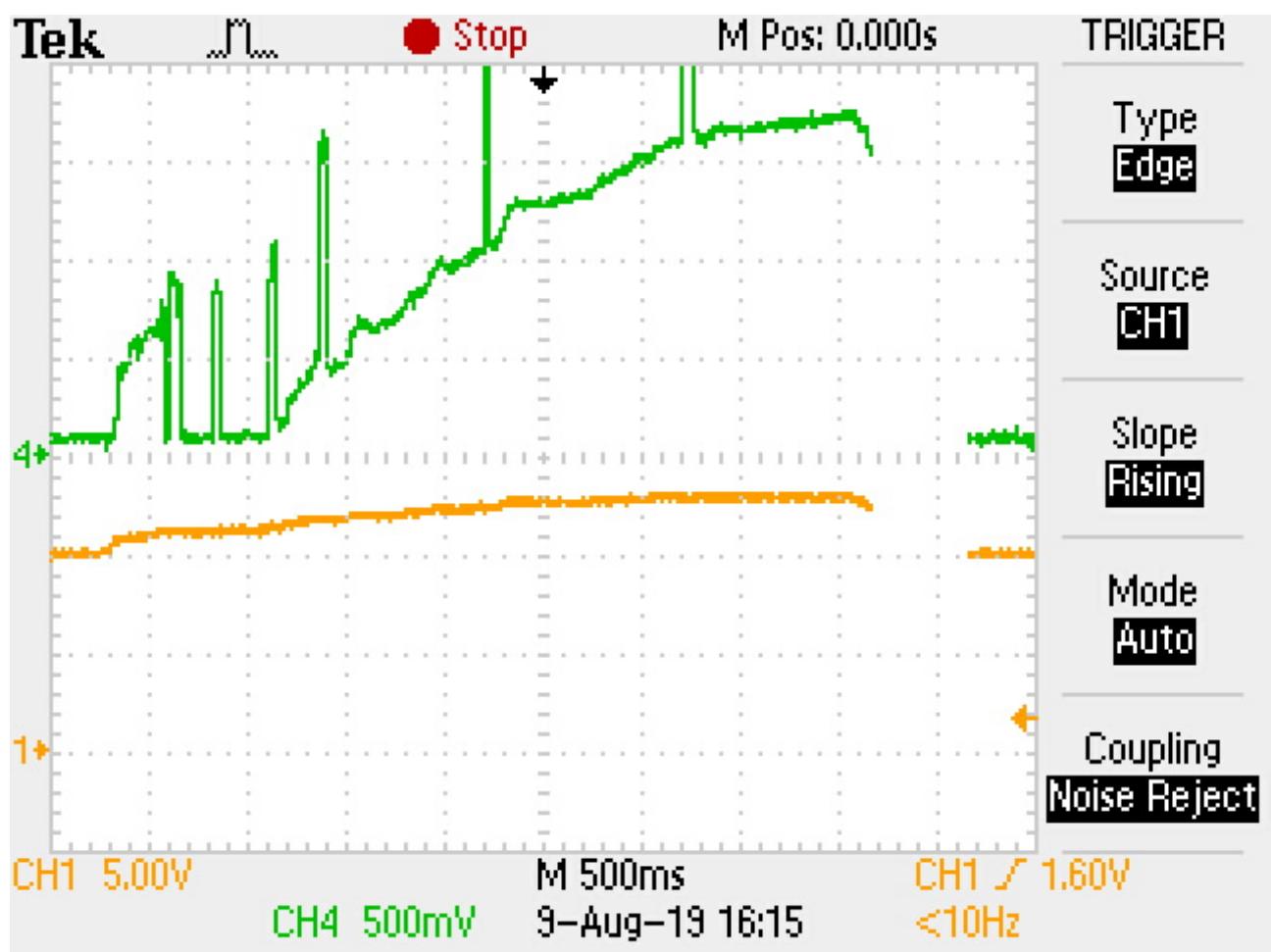


Abbildung 31: Messung H-Brücke Ansteuerung Artomat controller v1 Wild 2019a

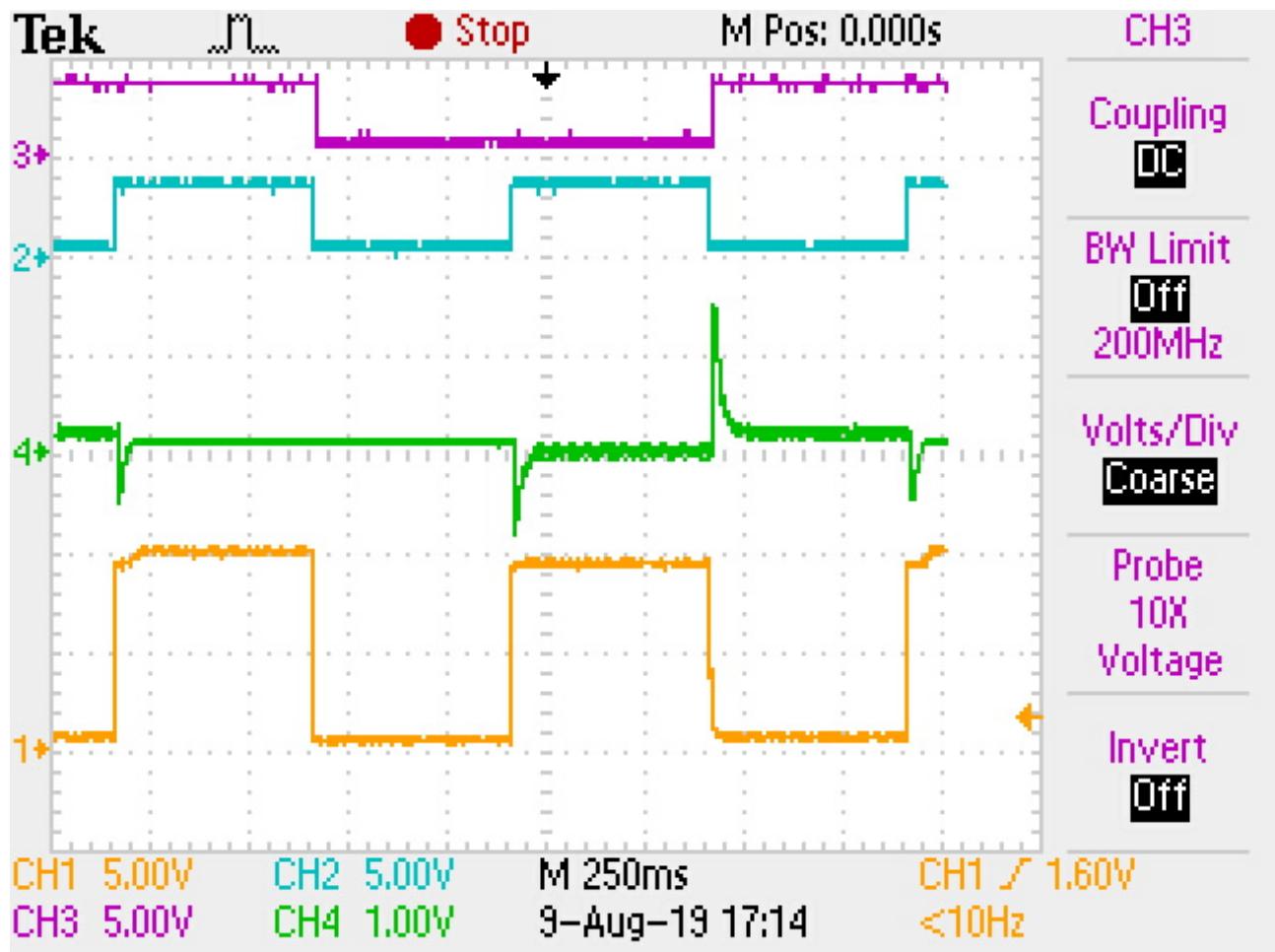


Abbildung 32: Messung H-Brücke Ansteuerung Artomat controller v1 Wild 2019a

9 Abbildungen

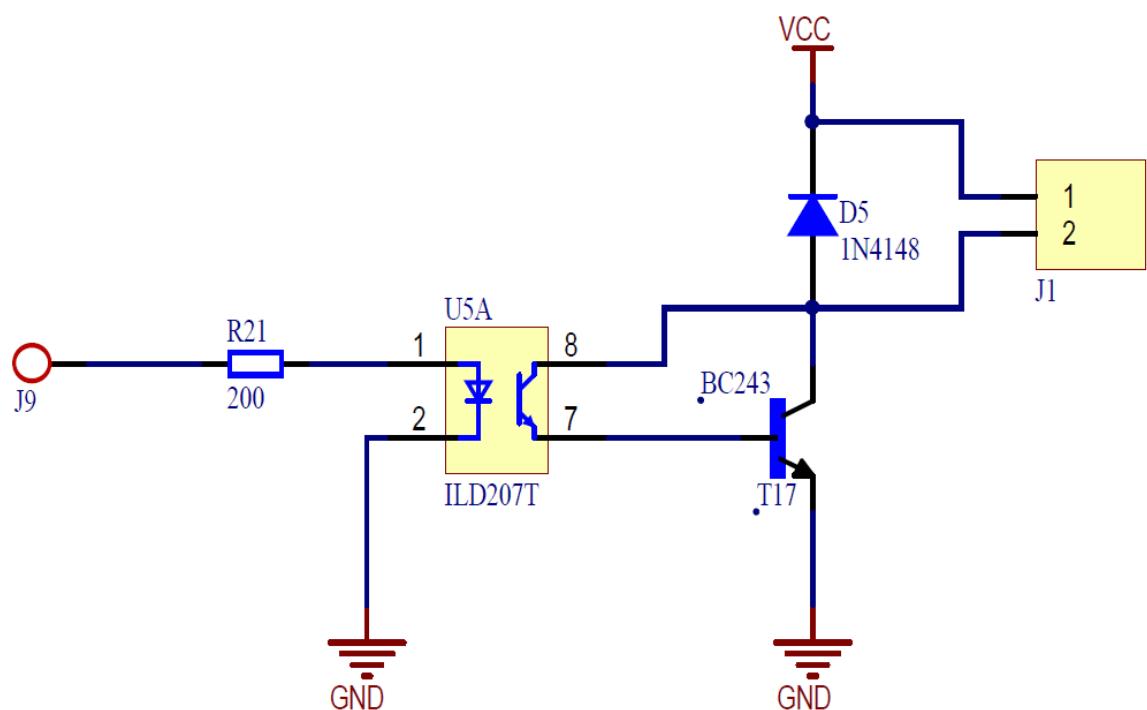


Abbildung 33: Ausschnitt aus Schema Artomat controller v1 Wild 2019a

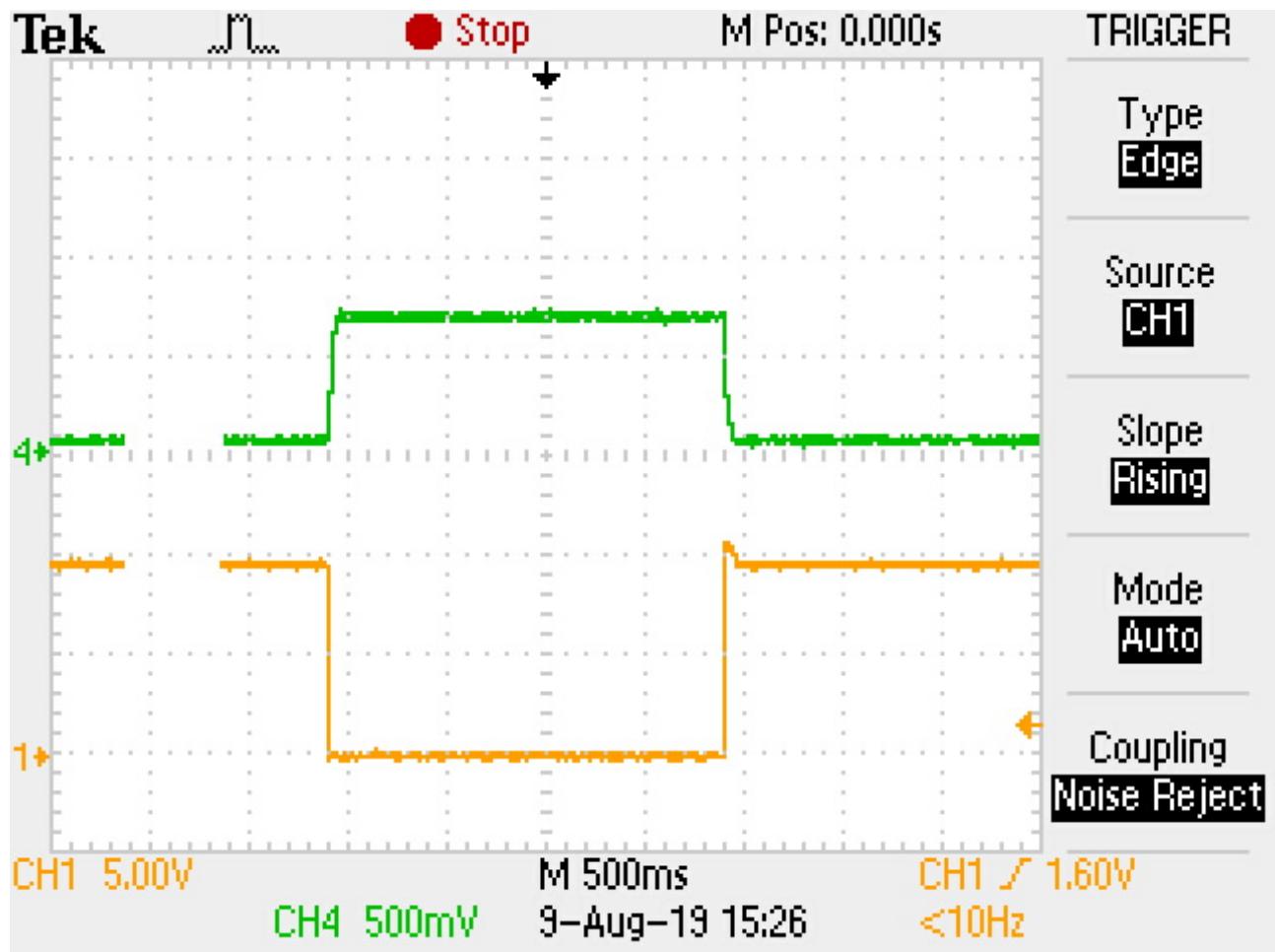


Abbildung 34: Messung Solenoid Ansteuerung Artomat controller v1 Wild 2019a

9 Abbildungen

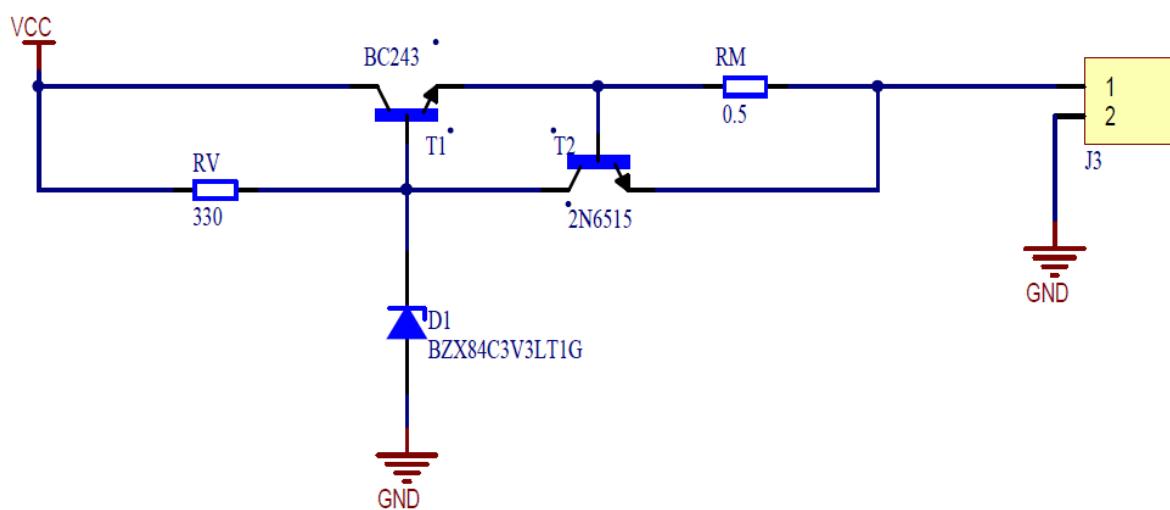


Abbildung 35: Kollektorschaltung

9 Abbildungen

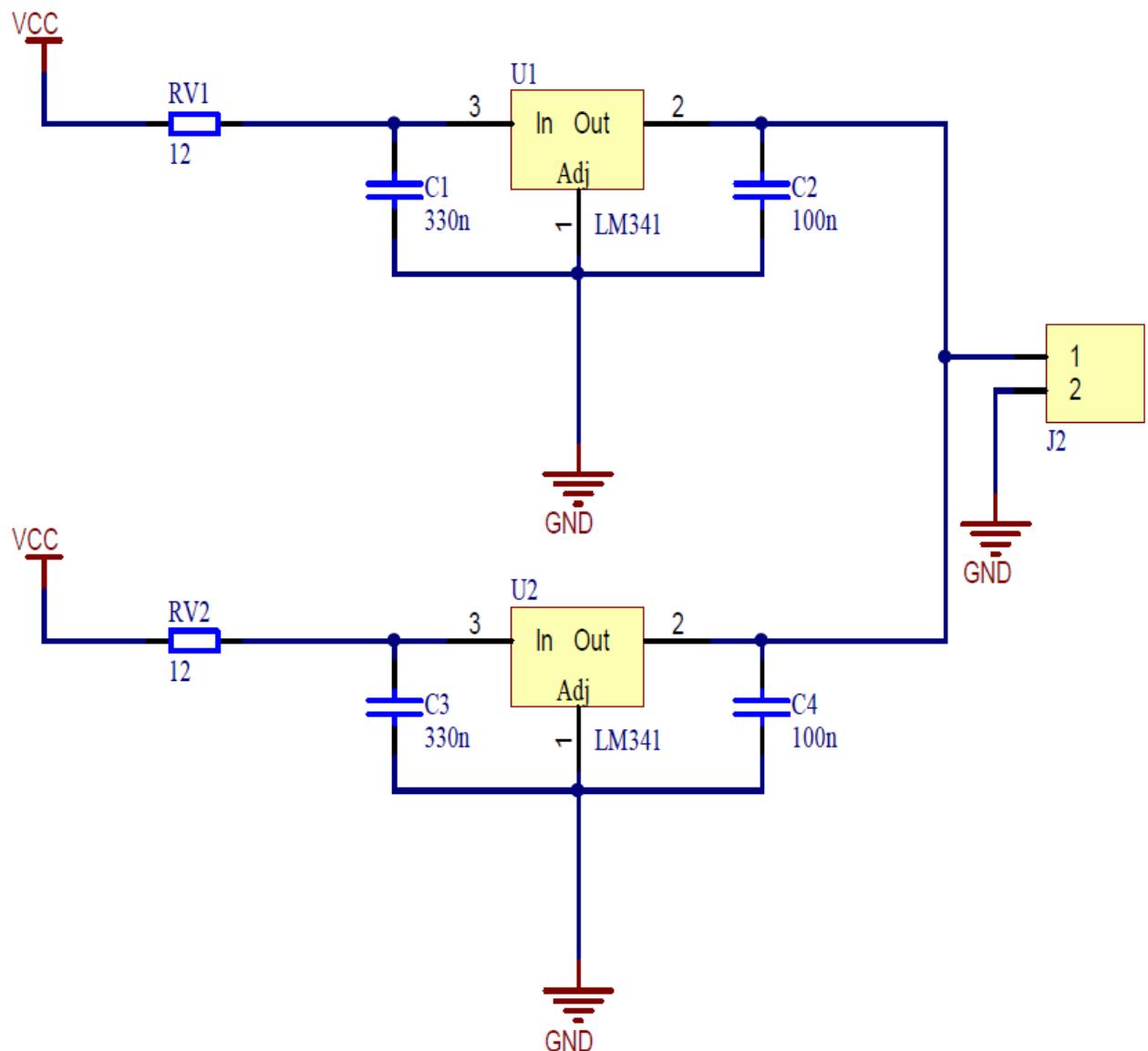


Abbildung 36: LM341-T in der Version 1

9 Abbildungen

Kriterium	Wert
Bauteiltyp	SMT
Spannung	12 V
Max. Strom	1.44 A
$R_{DS(on)}$	1 Ω

Abbildung 37: Tabelle Kriterien integrierte H-Brücke

Abbildung 38: Artomat Controllerboard V2 H-Brücken Messung

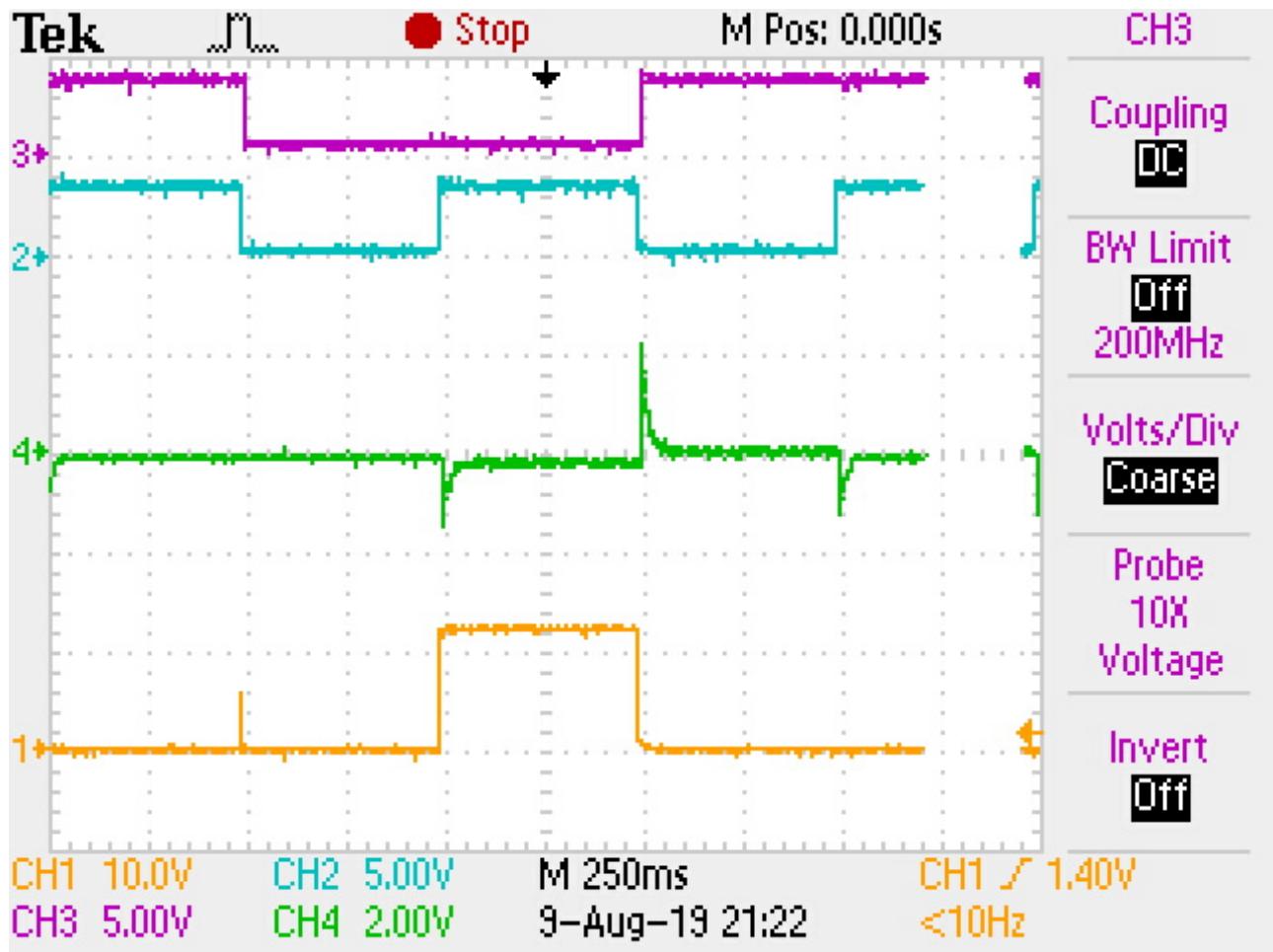


Abbildung 39: Artomat Controllerboard V2 H-Brücken Messung

9 Abbildungen

Kriterium	Wert
Bauteiltyp	SMT
Spannung	12 V
Max. Strom	1 A
$V_{CE\max}$	1 V

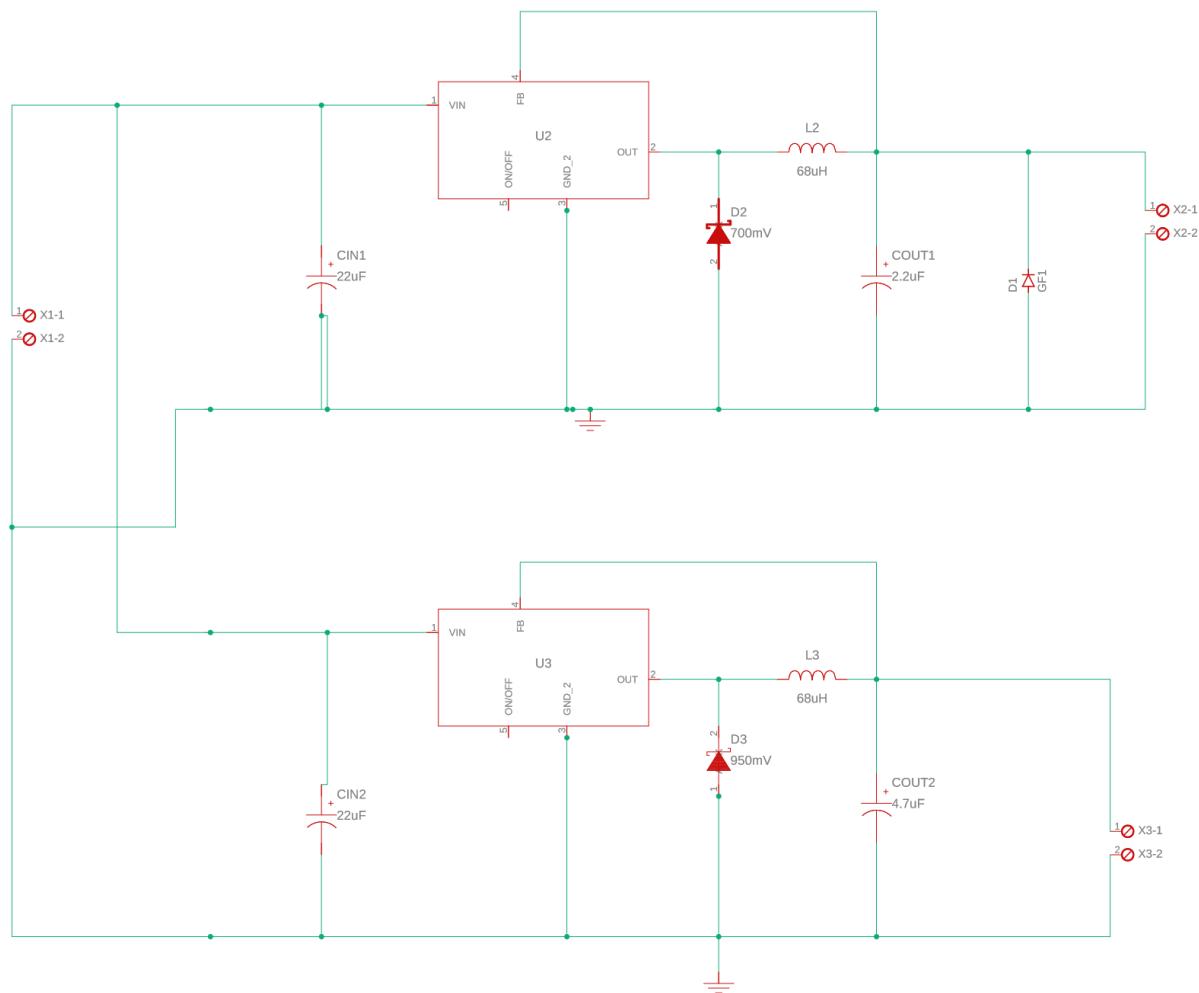


Abbildung 40: Stromversorgung der Version 2

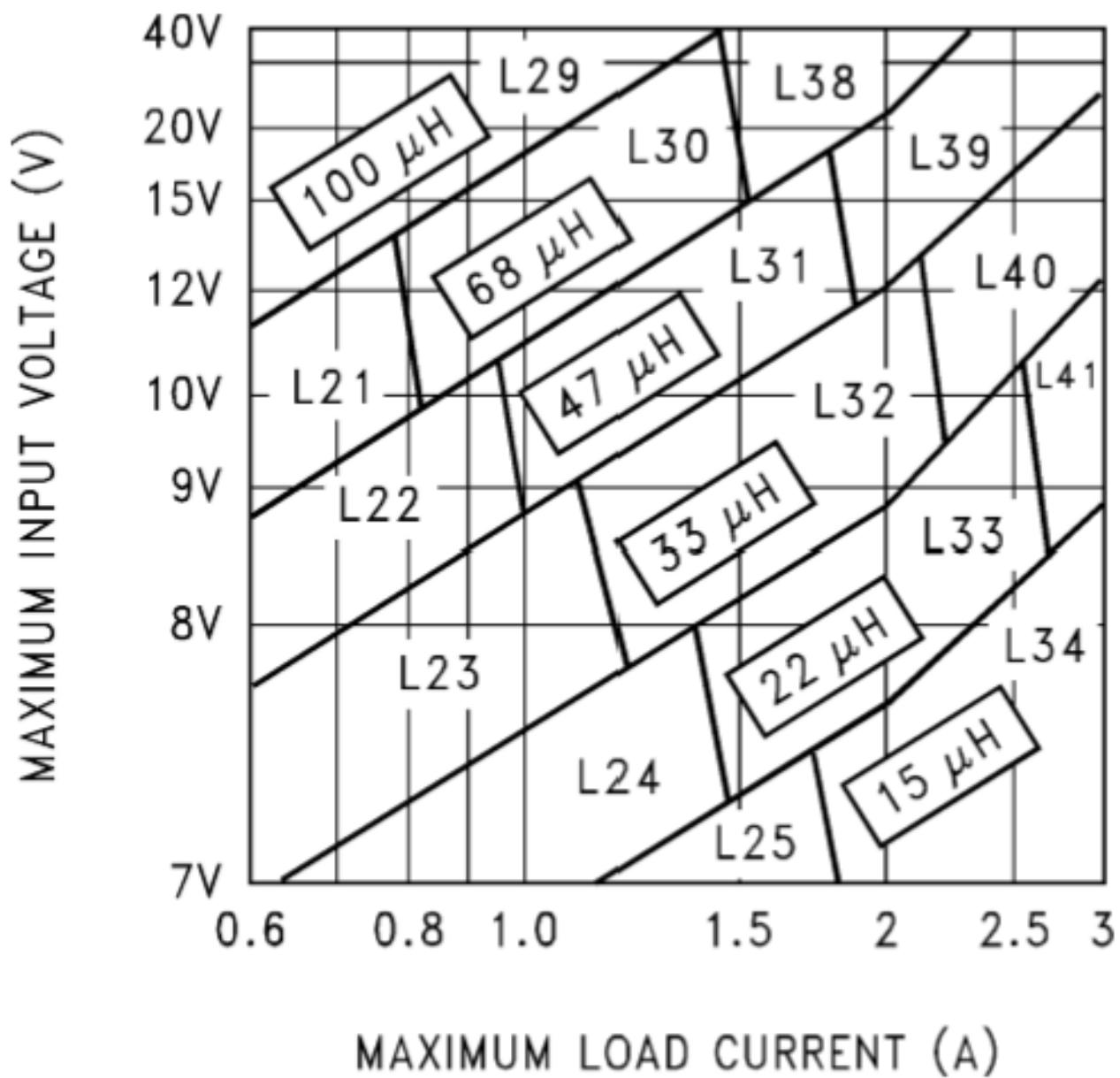


Abbildung 41: Dimensionierung Spule LM2596-5.0 o.V. 1999

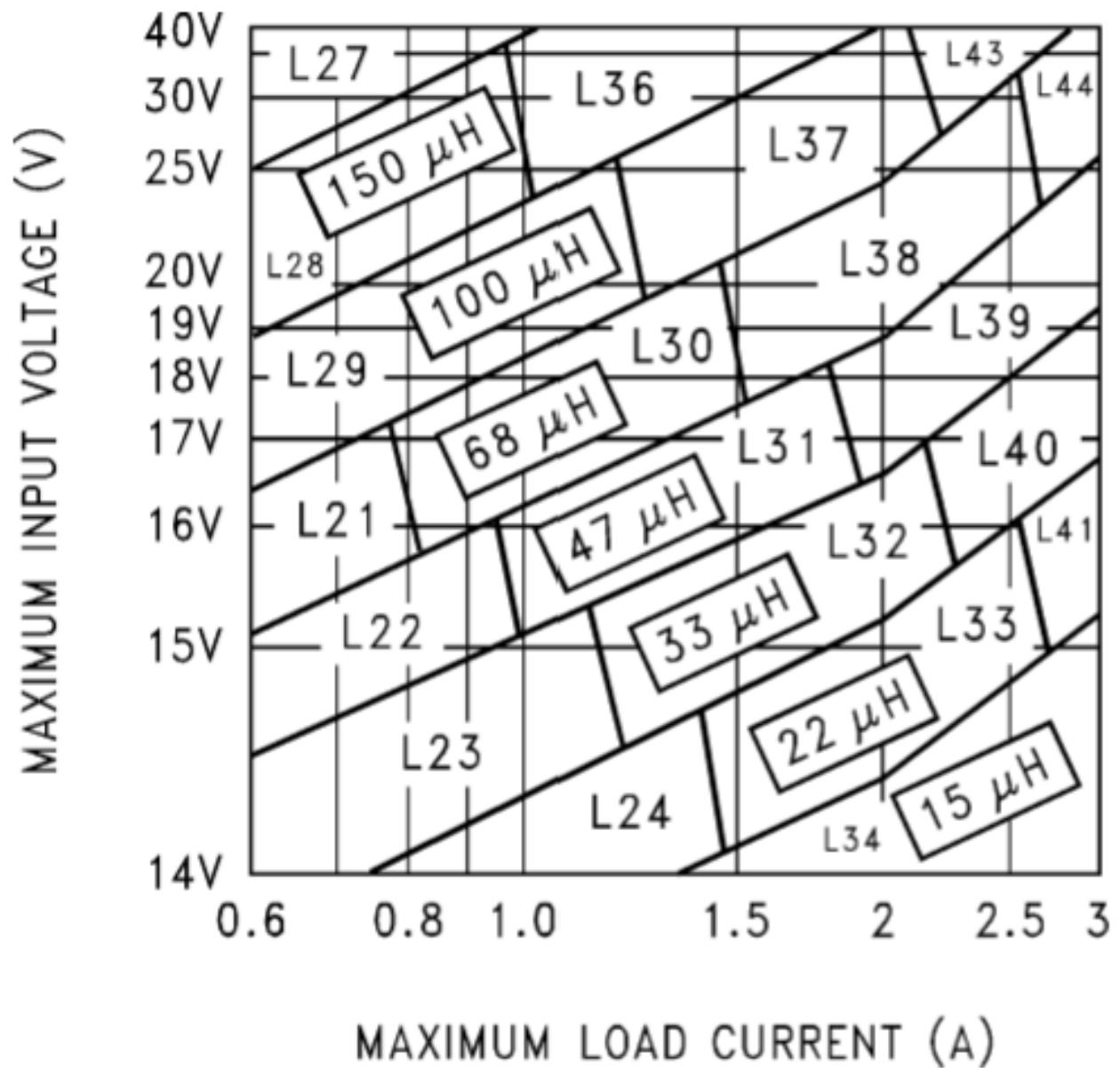


Abbildung 42: Dimensionierung Spule LM2596-12.0 o.V. 1999

9 Abbildungen

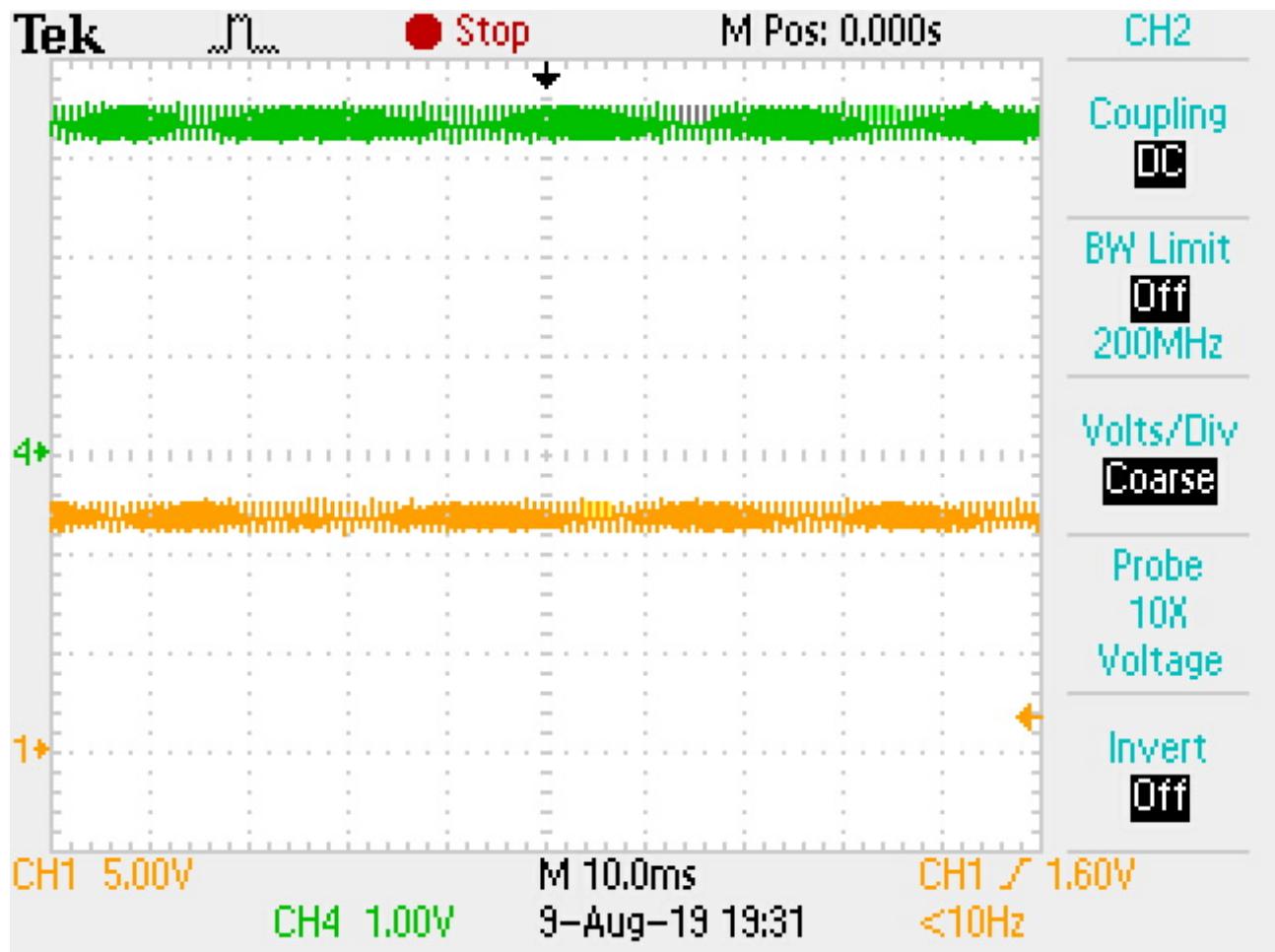


Abbildung 43: 12 V Ausgang von Stromversorgung V2

9 Abbildungen

```
def solenoid_s1(self , ontime):
    global s1Start,s1Finish,s1Enabled , tspraymax , current_on_time1
    if ontime <= 0 and not s1Enabled:
        if s1Finish:
            s1Finish = False
        elif not s1Finish:
            if s1Enabled:
                tON= time.time()- s1Start

            if current_on_time1 < tON or tON > tspraymax:
                self.dwiGpio(24, 0)
                s1Finish=True
                s1Enabled=False
            else:
                s1Enabled=True
                s1Start=time.time()
                self.dwiGpio(24, 1)
                current_on_time1 = ontime
```

Abbildung 44: Funktion um den Druckmechanismus zu kontrollieren

9 Abbildungen

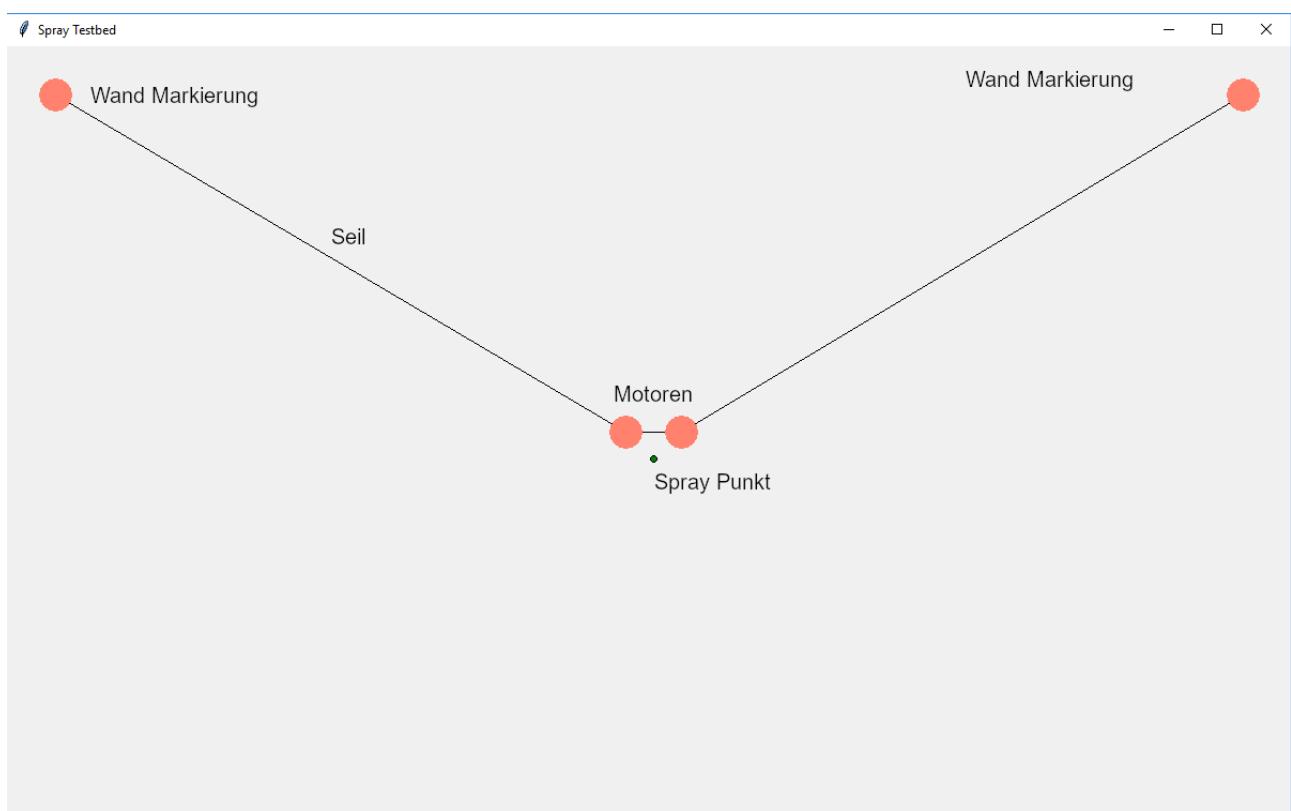


Abbildung 45: Visuelle Erklärung

9 Abbildungen

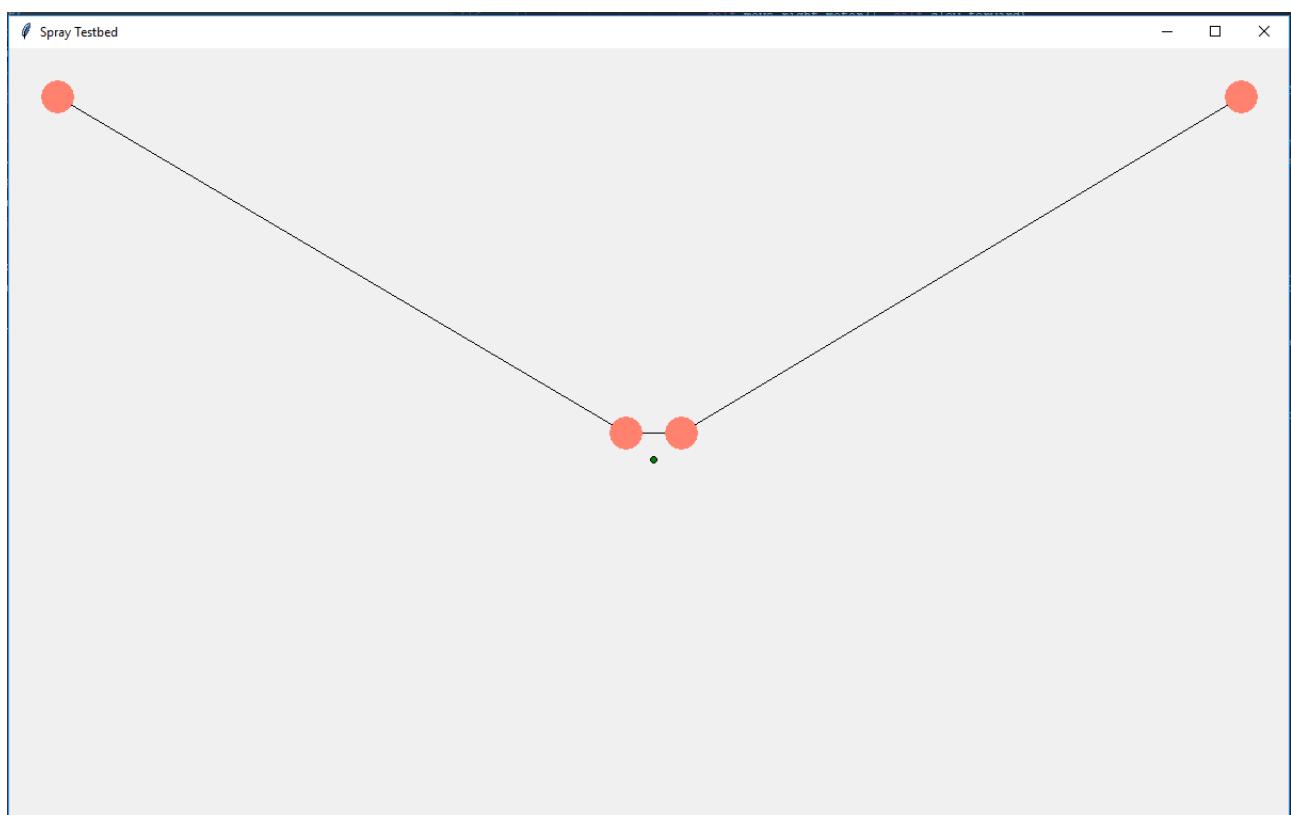


Abbildung 46: Simulation Ausgangslage

9 Abbildungen

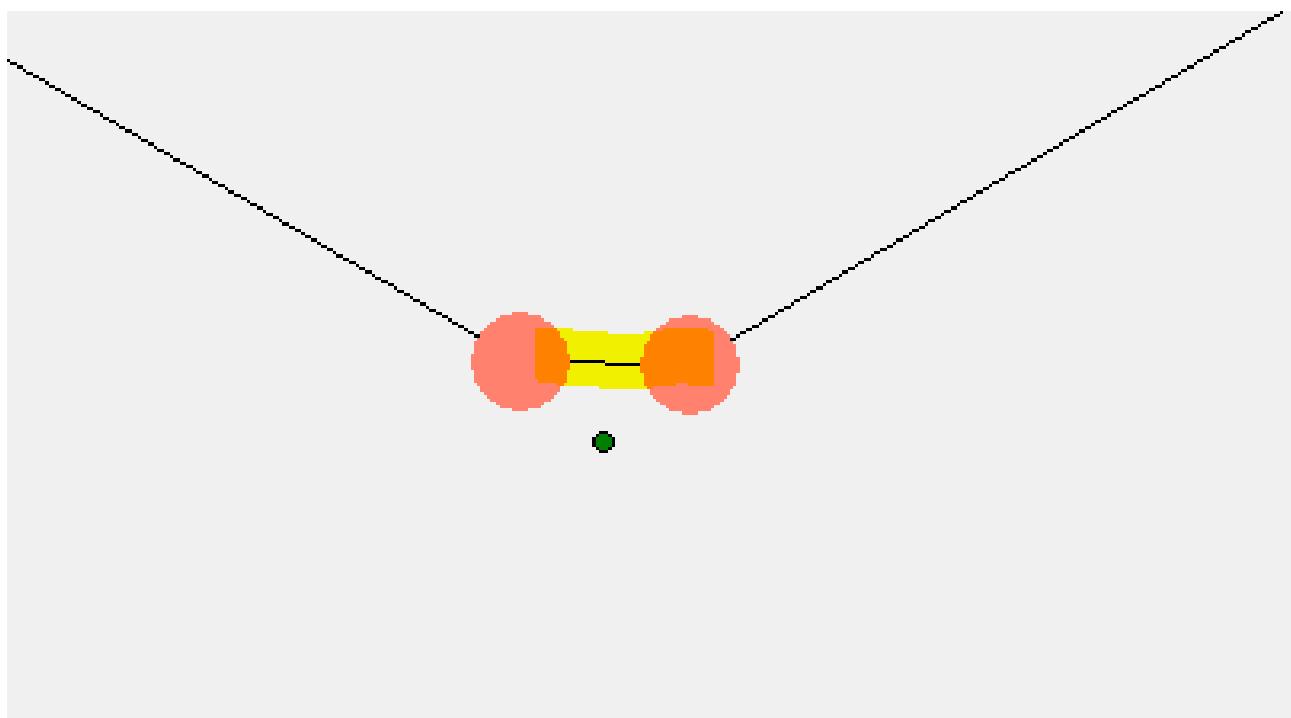


Abbildung 47: Linker Motor wird angezogen

9 Abbildungen

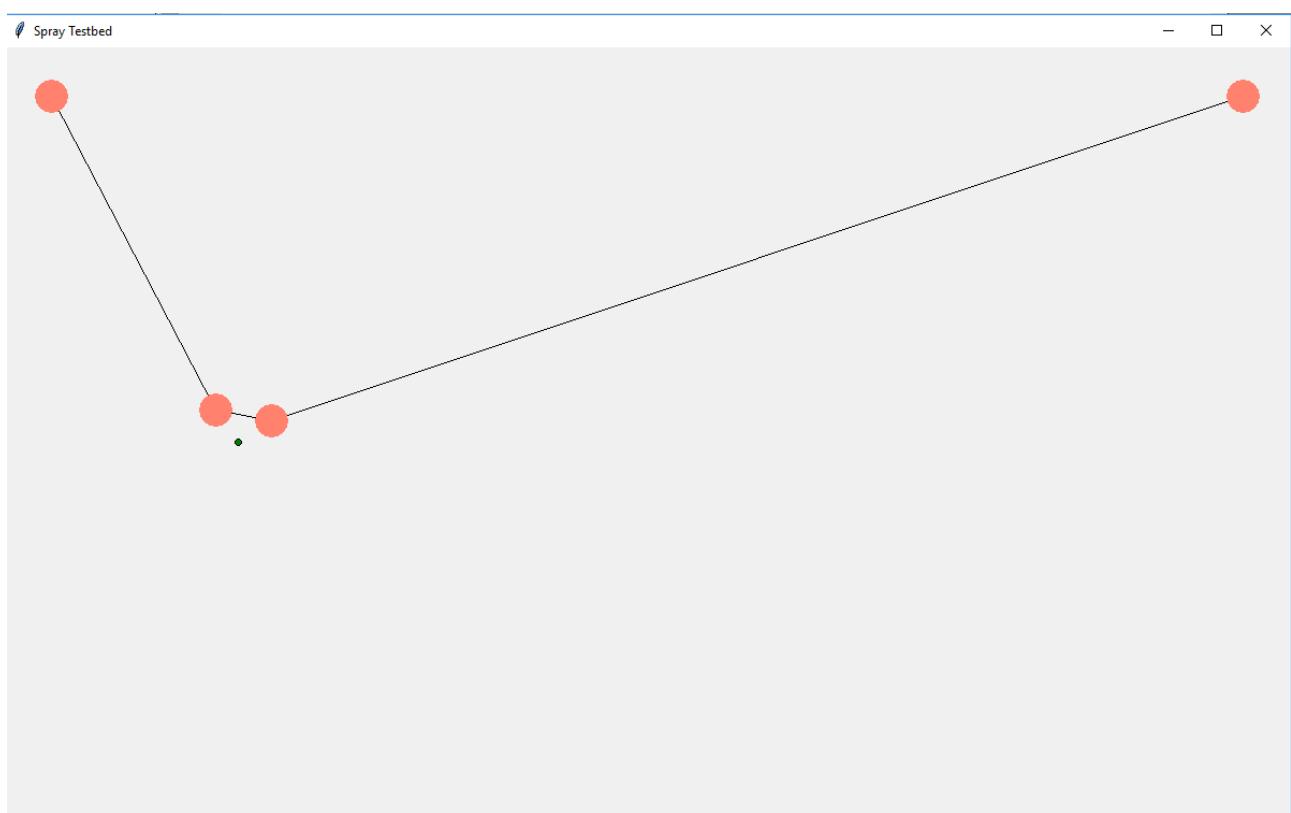


Abbildung 48: Linker Motor wird angezogen

9 Abbildungen



Abbildung 49: Linker Motor wird angezogen

9 Abbildungen



Abbildung 50: Linker Motor wird angezogen

9 Abbildungen

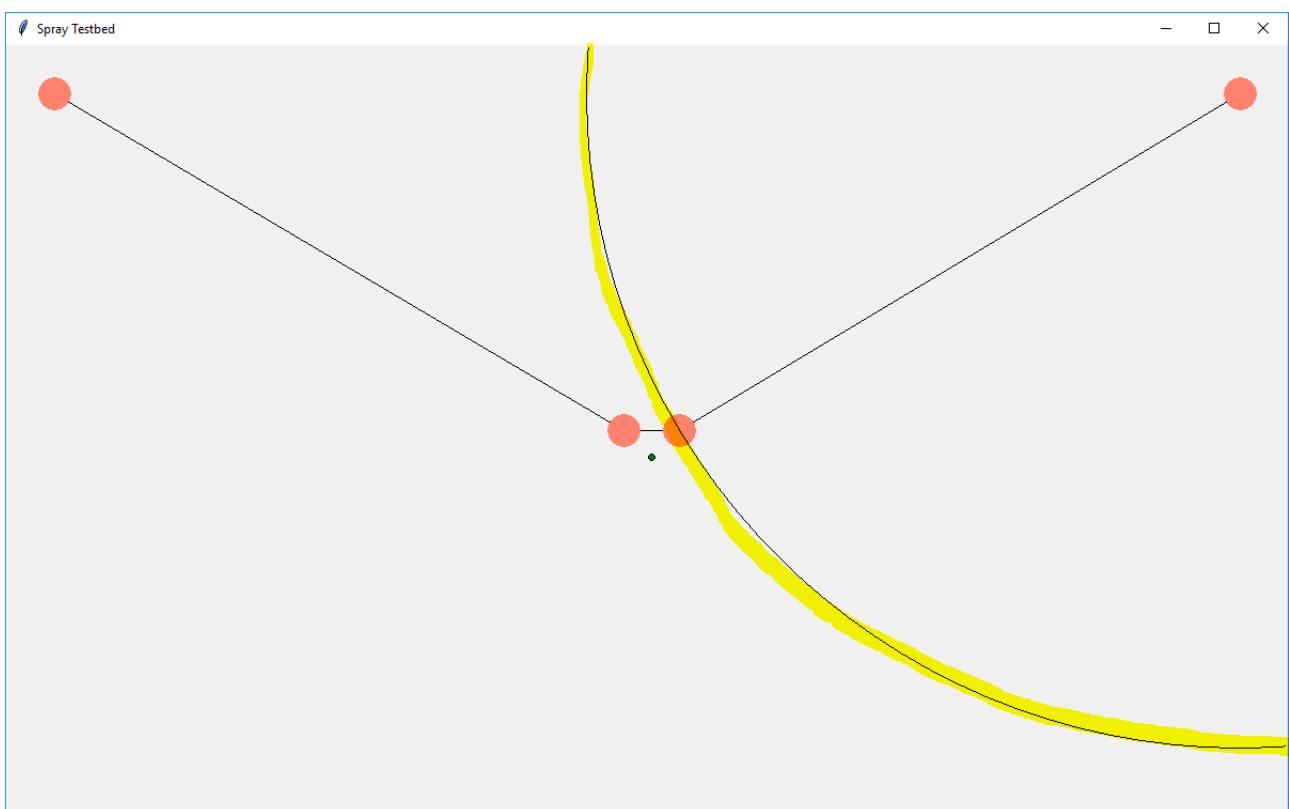


Abbildung 51: Motor Wanderlinie

9 Abbildungen



Abbildung 52: Simulation Schwachpunkt

9 Abbildungen



Abbildung 53: Simulation Resultat

9 Abbildungen

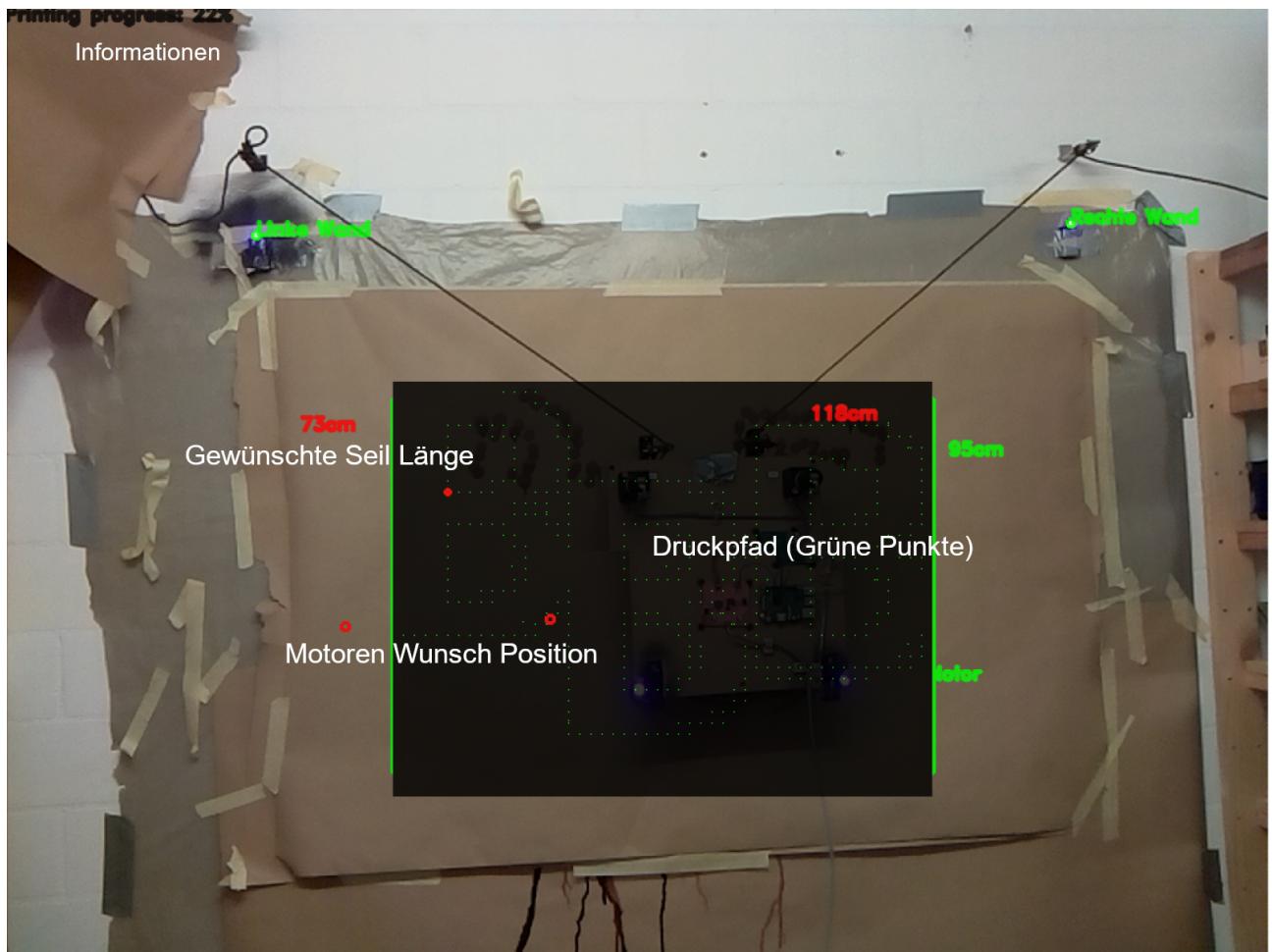


Abbildung 54: Overlay der Vision

9 Abbildungen

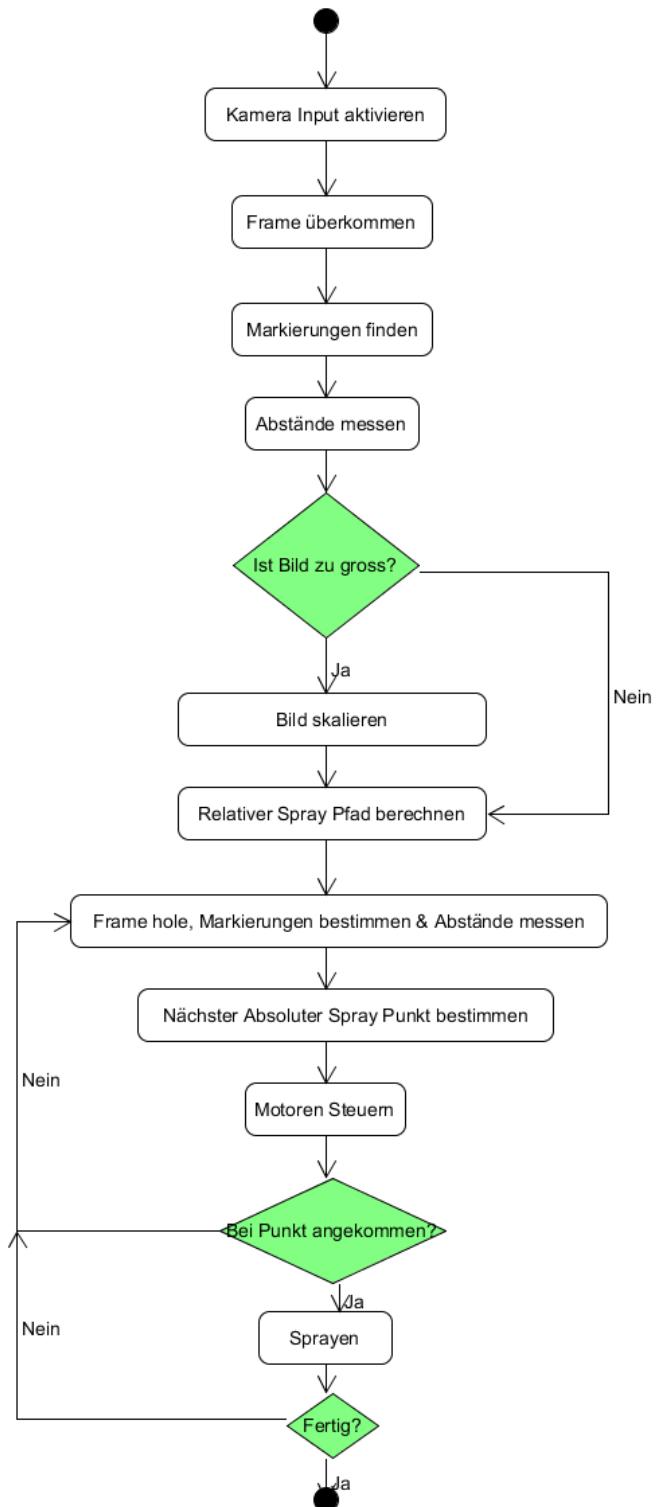


Abbildung 55: Vision Ablauf

9 Abbildungen

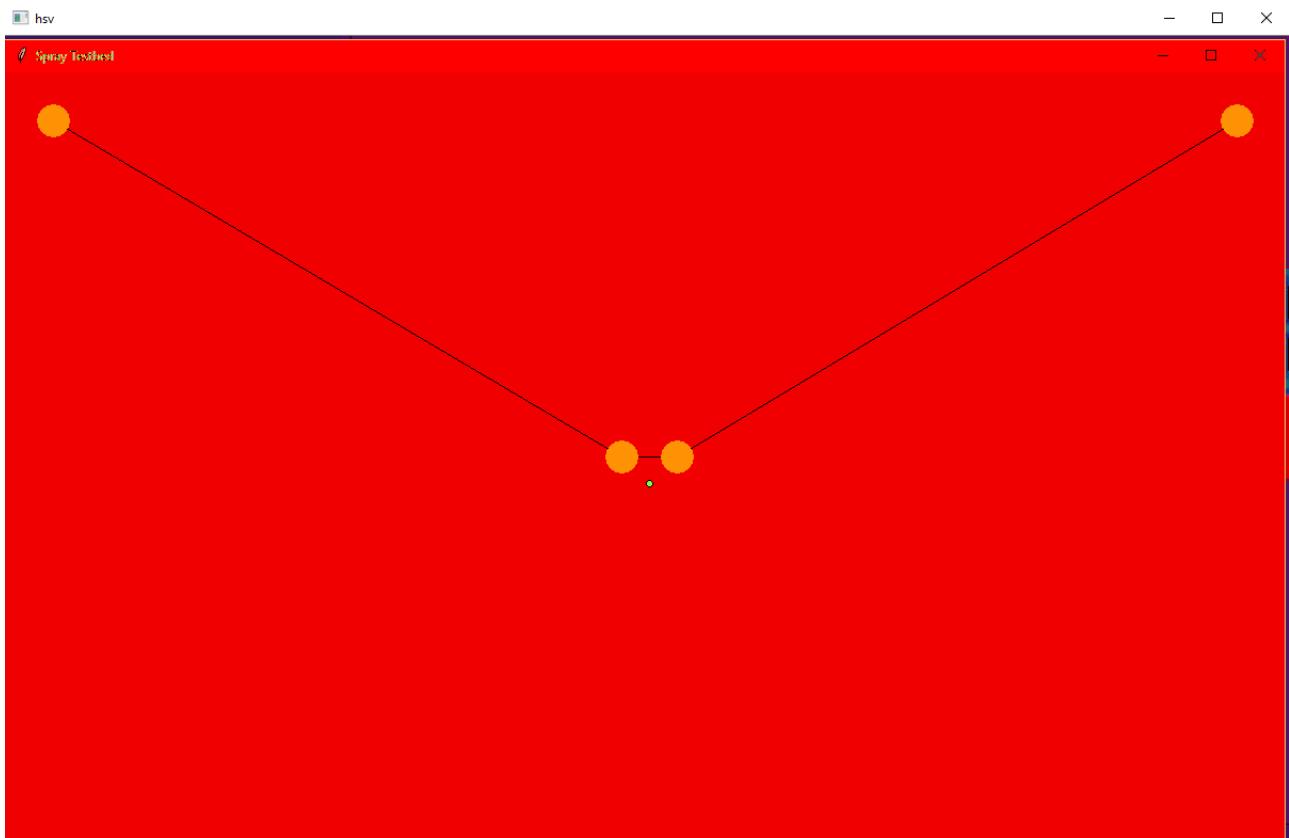


Abbildung 56: HSV

9 Abbildungen

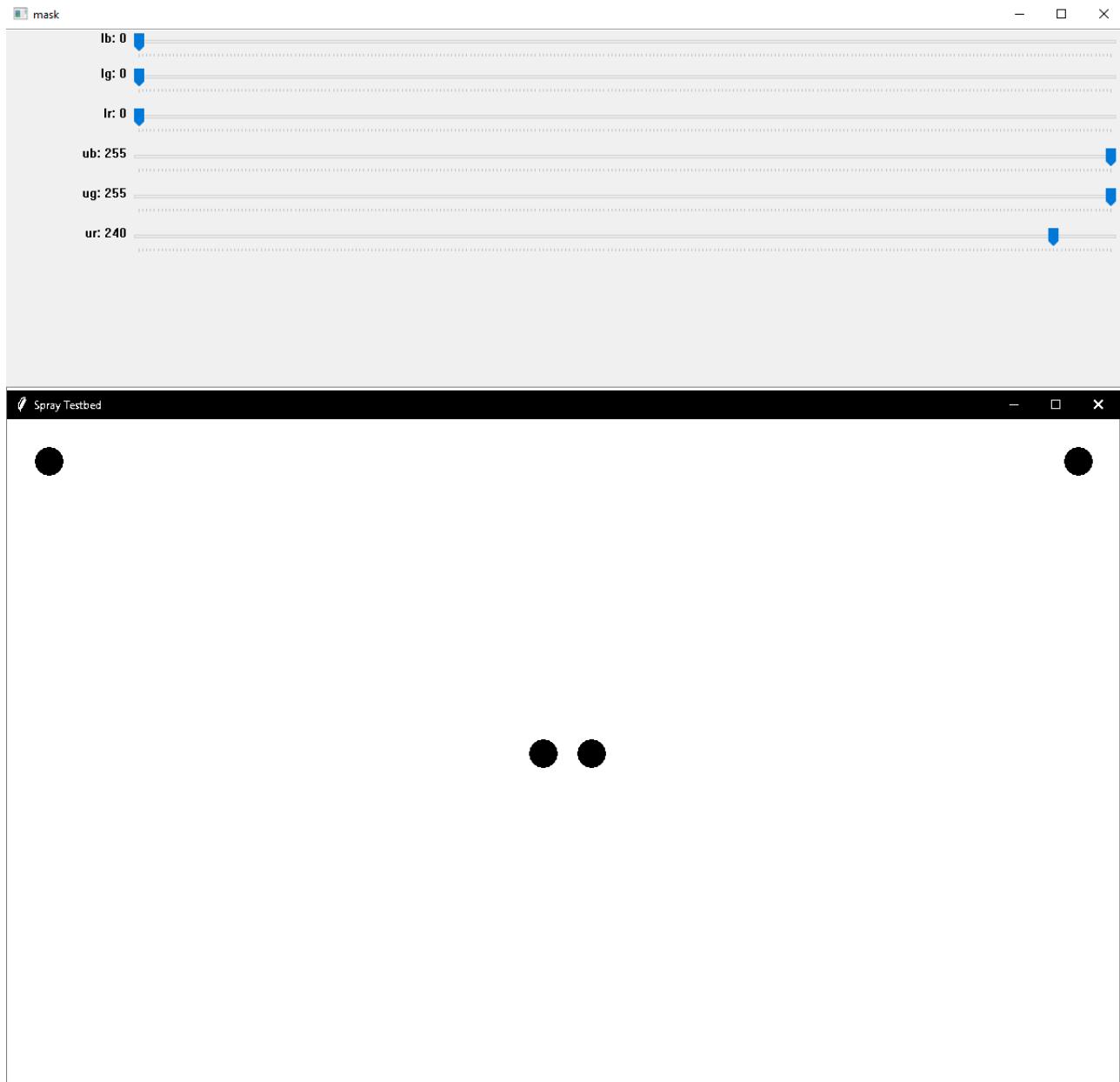


Abbildung 57: Farbraum

9 Abbildungen

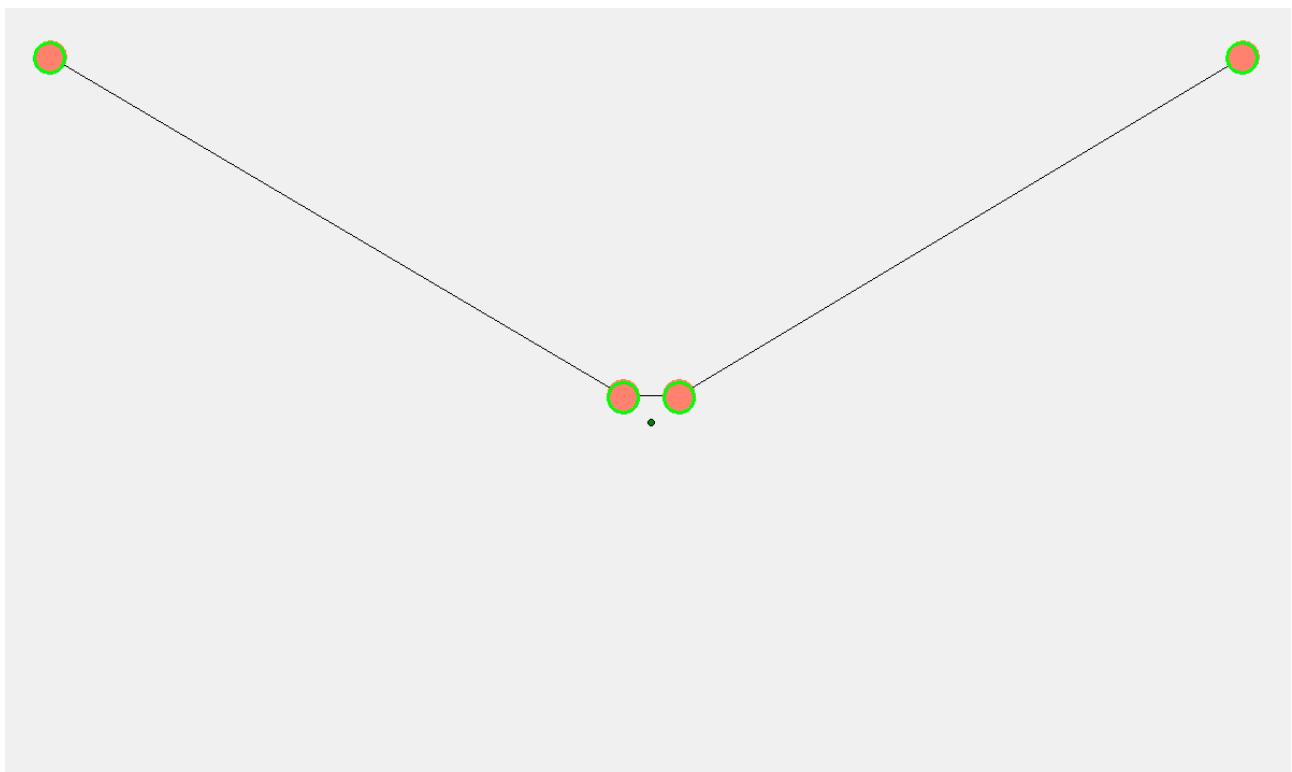


Abbildung 58: Erkennung von Kreisen

9 Abbildungen



Abbildung 59: Soll-Bild des ersten Ausdruck

9 Abbildungen

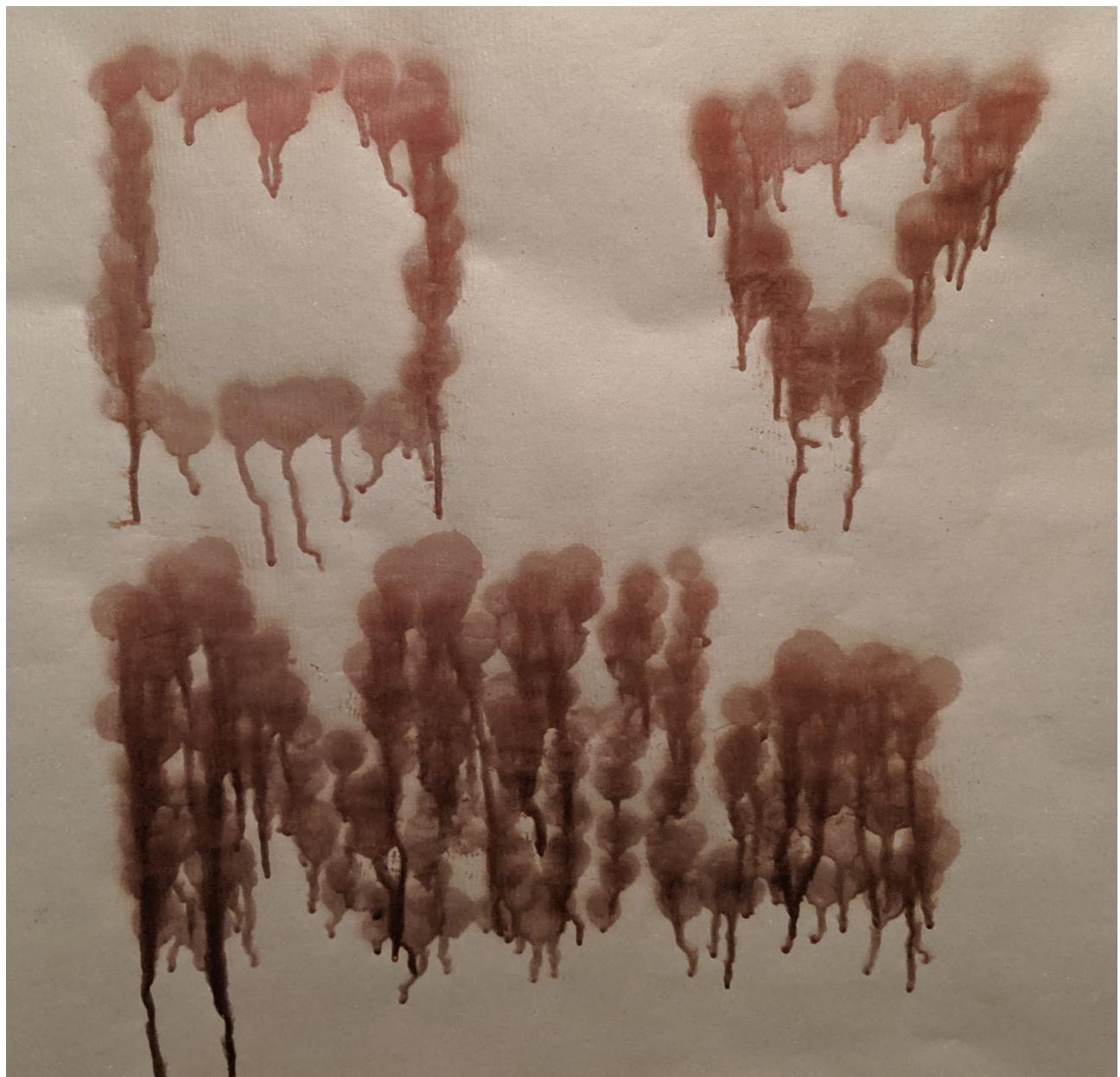


Abbildung 60: Erstes Bild Ergebnis