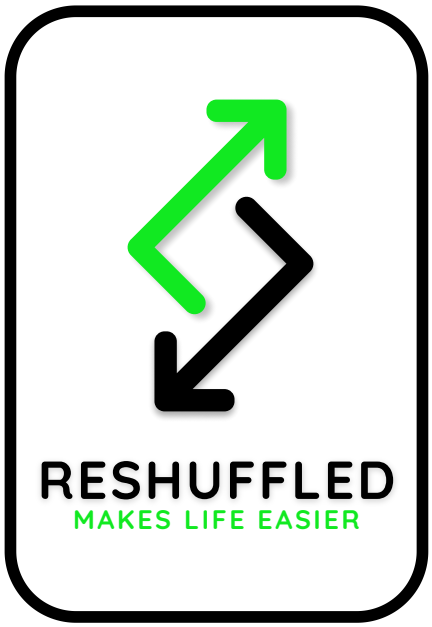


Diplomarbeit ReShuffled

HTBLA Kaindorf an der Sulm
Grazer Straße 202, A-8430 Kaindorf an der Sulm
Ausbildungsschwerpunkt Mechatronik

Vollmaier Alois Perl Nicolas Hörmann Stefan

Abgabedatum: 06.04.2020

Dr. Dipl-Ing. Gerhard Pretterhofer

Dipl-Ing. Manfred Steiner

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Arnfels, am 5. April 2018

Stefan Hörmann	Nicolas Perl
Alois Vollmaier	Mr. Mösensaft

Danksagung

An dieser Stelle möchten wir uns bei allen bedanken, die uns im Rahmen der Diplomarbeit unterstützt und betreut haben.

TODO

Abstract

TODO

Zusammenfassung

Wir setzten uns als Ziel eine Maschine zu entwickeln, welche das Mischen, sowie das Ausgeben von Spielkarten übernimmt. Die Idee ist es, diese Verfahren möglichst platzsparend, zeiteffizient und detailliert durchdacht und optimiert zu realisieren.

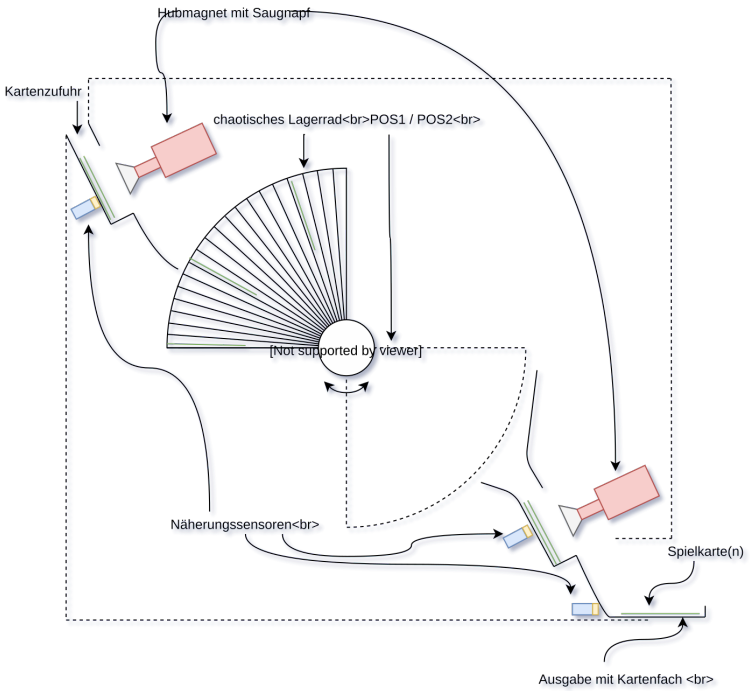


Abbildung 1: Name des Bildes

Grundsätzlich basiert das Mischprinzip auf einer Art "Fächersystem". Eingelegte Karten gelangen mithilfe eines ausgeklügelten Systems, welches aus einem Hubmagneten mit integriertem Saugnapf besteht, aus dem Einlegefach. Erfolgt die Kartenentnahme, rutscht die Karte in ein zufälliges Fach des Lagerrads. Anschließend wird dieses Lagerrad in Drehbewegung versetzt um die gelagerten Karten auszugeben.
Der Benutzer steuert diese Maschiene mithilfe einer GUI welche auf einem 7"LCD Display angezeigt wird. Systemintern steuert ein 8-Bit Mikrocontroller der AVR-Familie den Ablauf.

Gender Erklärung

Aus Gründen der besseren Lesbarkeit wird in dieser Arbeit die Sprachform des generischen Maskulinums angewendet. Es wird an dieser Stelle darauf hingewiesen, dass die ausschließliche Verwendung der männlichen Form geschlechtsunabhängig verstanden werden soll.

Über dieses Dokument

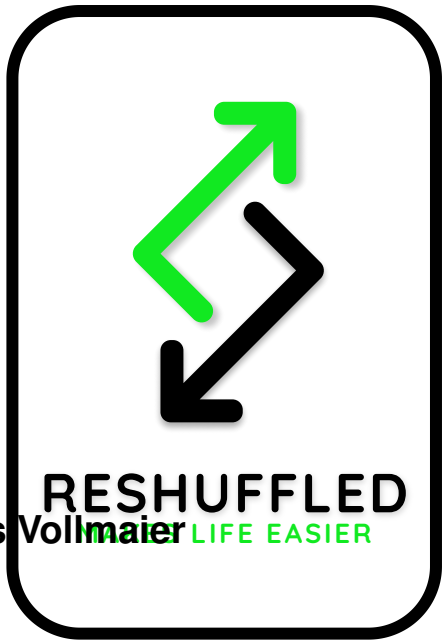
Diese Arbeit wurde in LaTeX verfasst. Diese Art der Dokumentation bietet gegenüber den normalen Textverarbeitungen gewisse Vorteile hinsichtlich der Formatierung und des Einbindens von Grafiken. Auch Formeln können sehr einfach und effizient angegegeben werden. Die Rohfassung des Dokuments befindet sich auf dem Arnfelser Gitweb Server der HTBLA Kaindorf Abteilung Mechatronik.

Projektteam

Stefan Hörmann

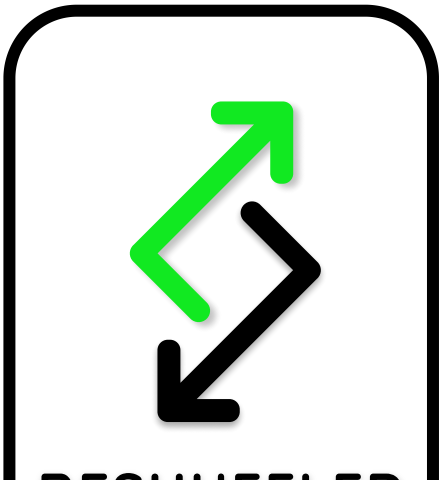
Aufgabenbereich:
Mechanik
Betreuer:
Dr. Dipl-Ing. Gerhard Pretterhofer

Nicolas Perl



Aufgabenbereich:
Elektronik
Betreuer:
Dipl-Ing. Manfred Steiner

Alois Vollmaier



Aufgabenbereich:
Informatik
Betreuer:
Dipl-Ing. Manfred Steiner

Inhaltsverzeichnis

1 Mechanik

1.1 Einleitung

1.2 Anforderung

Die Arbeit des mechanischen Teiles besteht darin, eine Maschine, die Spielkarten mischen und ausgeben kann zu entwerfen, zu konstruieren und einen Teilaufbau durchzuführen. Die Maschine sollte in der Lage sein 20 Spielkarten zu mischen und diese nach einem Spielmodus der zuvor am LCD gewählt wurde auszugeben. Das Ziel ist es, die Spielkarten optimal zu mischen, aber die Maschine dennoch kompakt und optisch ansprechen zu entwerfen. Im weiteren sollte der Mischvorgang und die Ausgabe der Karten nicht zu lange dauern. Die Teile der Maschine sollten so konstruiert werden, dass sie kostengünstig produziert werden können. Zum Schluss sollte noch ein Teilaufbau der Maschine geschehen, um die Funktionalität der einzelnen Bereiche zu testen und gegebenfalls zu verbessern.

1.3 Problemstellungen

Ein Problem ist das begrenzte Budget unseres Teams, somit sind wir auf gewisse Produktionsarten unserer Bauteile beschränkt. Dies hat zur Folge, dass die Bauteile oft sehr simpel sind um sie leichter zu konstruieren. Die Oberfläche der Karten ist ein weiteres Problem, da diese sich nicht immer separieren lassen, dies verursacht, dass oft zwei oder mehrere Karten auf einmal genommen werden und das Konzept des optimalen Mischens zerstört.

1.4 Konzepte

1.4.1 Anforderungen

1. **Kosten**
   Der Automat sollte möglichst kostengünstig produziert werden, da das vorhandene Budget gering ist. Dies hat zur Folge das keine teuren Motoren oder ähnliche Bauteile zum Einsatz kommen können und keine teuren Bauteile produziert werden können.

2. **Schnelligkeit**
   Um ein gutes Spielerlebnis zu garantieren, sollte der Automat keine langen Mischzeit besit-

zen. Die Dauer in der man die Karten einführt und auf den Mischen-Button klickt bis hin zur Ausgabe der ersten Karte sollte möglichst gering sein.

3. **Mischgenauigkeit**
   Die Mischgenauigkeit ist die am schwersten gewichtete Anforderung, da es das Ziel ist ein optimales Mischen der Spielkarten zu erreichen, sollte diese Anforderung mit größter Wichtigkeit erfüllt werden.

4. **Optik und Größe**
   Die Optik des Automaten soll schlicht gehalten werden, jedoch sollte sie dennoch auf Messen und andere Ausstellungen präsentierbar sein. Der Automat sollte jedoch auch stabil konstruiert werden, muss aber dennoch mobil bleiben und darf eine gewisse Größe nicht Überschreiten.

1.4.2 Variantenvergleich

Um alle oben angegebenen Anforderungen zu erfüllen, wurden mehrere Konzepte entworfen und diese verglichen.

1.4.2.1 Variante 1 - Linearachsen

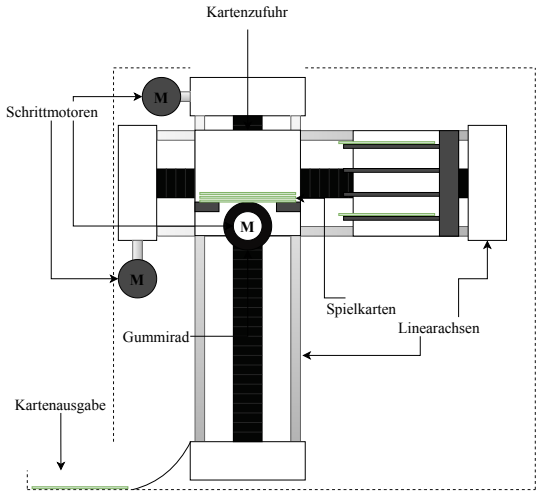


Abbildung 1.1: Variante 1

Das erste Konzept würde mit zwei Linearachsen realisiert werden, diese wären im rechten Winkel zueinander angeordnet. Die Senkrechte Linearachse ist mit einer Halterung versehen, diese Halterung ist in der Lage ein Kartendeck aufzunehmen und die unterste Karte mithilfe eines Ausgaberades weiterzubefördern. Die zweite Linearachse besitzt 4 Fächer in der die Karten von der ersten Linearachsenausgabe zufällig befördert werden. Dies wird realisiert indem die erste Linearachse bei jeder Ausgabe zufällig das Fach durch hinauf und hinabfahren wechselt. Befinden sich alle Karten im Lager, so fährt die erste Linearachse nach unten, danach fährt die zweite Linearachse impulsiv nach links, um die Karten aus dem Lager zu befördern. Diese fallen Senkrecht in das

Lager der ersten Linearachse, wo sie nun zum Ausgeben durch das Ausgaberad bereit liegen.

Durch die schnelle Bewegung der Linearachsen ist es möglich einen schnellen Mischprozess zu erreichen, auch die Tatsache das es nur ein rotierendes Rad gibt und zwei Bewegliche Linearachsen führt dazu, das Fehler bei Bewegungen nur selten Auftreten. Jedoch besteht durch den hohen Aufbau der Maschine und durch die hohe Position der zweiten Linearachse die sich horizontal bewegt die Gefahr des umkippens der Maschine, und somit ist keine stabilität mehr gegeben. Der Preis der Linearachsen ist ein weiterer Nachteil dieses Konzeptes, eine Linearachse die unsere Anforderungen entspricht, wäre mit Motor und Schlitten zu teuer für unser Budget.

Vorteile:

- schnelles Mischen
- wenige Fehlerquellen

Nachteile:

- teuer
- großer Aufbau
- instabil

1.4.2.2 Variante 2 - Lagerrad mit Asugaberäder

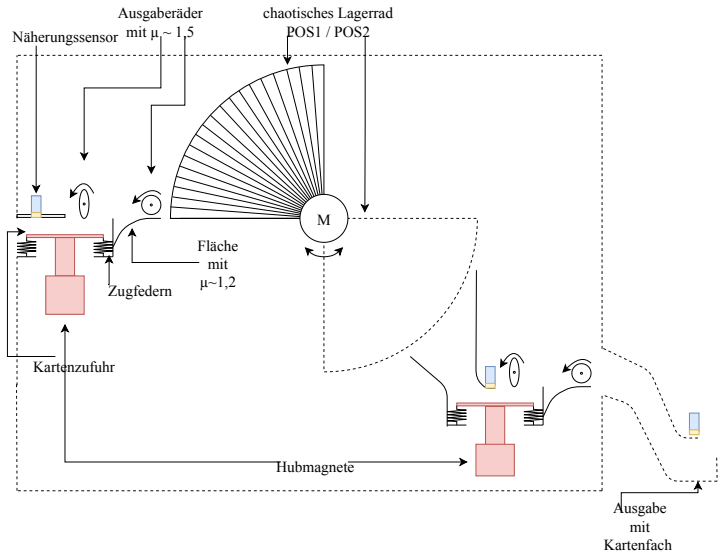


Abbildung 1.2: Variante 2

Beim zweiten Konzept wird als Lager ein virtel eines Zylinders benutzt. In diesem befinden sich verschidene Fächer in der die Karten eingelagert werden. Dieser wird mit einem Motor betrieben

und dreht sich somit in die vorgegebenen Positionen um das Einlagern und das Ausgeben der Karten zu ermöglichen. Die Karteneingabe erfolgt über einen Schlitz in der Frontplatte der Maschine, dort befindet sich ein Hubmagnet der die Karten zum weiterbefördern nach oben drückt. Ein Kapazitiver Sensor sorgt dafür, das scihergestellt werden kann, dass sich Karten auf dem Hubmagnet befinden. Um eine Karte in das Lager zu befördern wird der Hubmagnet eingeschaltet und drückt das Kartendeck auf das erste Ausgaberad, um die Kraft des Hubmagnetes zu minimieren sind Zugfedern angebracht. Das Ausgaberad befördert eine Karte weiter vor zum zweiten Ausgabrad welches weiderum sicherstellt das nur eine Karten in das Lagerrad transportiert wird. Danach dreht das Lagerrad auf eine andere zufällig ausgewählte Position. Dieser Prozess wird solange wiederholt bis alle Karten des Kartendecks sich im Lagerrad befinden. Befinden sich alle Karten im Lagerrad, so dreht sich dieses mit einer hochen Geschwindigkeit und wirft somit die Karten auf der Hinterseite der Maschine in eine Auffangführung. Diese Auffangsführung befördert die Karten in einen gleichen Mechanismus wie bei der Vorderseite der Maschine, in der sie von einem Hubmagenten nach oben gedrückt werden und von zwei Ausgaberädern zur Kartenentnahme geschoben werden. Da die Karten zum Schluss nach einem Spielmodus und somit in einer bestimmten Anzahl Ausgegeben werden, befindet sich ein Kapazitiver Sensor auch bei der Ausgabe der Karten, dieser soll überprüfen ob die Karten von dem Spiler berreits genommen wurden oder nicht.

Durch den niedrigen Aufbau der durch ein "Fließbandartiges"befördern der Karten erreicht wird, besitzt die Maschine ein hohes Maß an stabilität, jedoch ensteht dadurch auch der Nachteil dass die Maschine sehr lang wird. Da dieses Konzept 5 bewegliche Räder besitzt sowie zwei Hubmagneten ist es anfällig für Fehler beim Bewegungsablauf. Auch kann nicht garantiert werden dass nur eine Karte in das Lagerrad befördert wird, dies würd das Konzept des optimalen Mischens zerstören. Die vielen Bauteile führen auch zu teureren Anschaffungskosten, die wir stark vermeiden möchten.

Vorteile:

- stabil
- niedriger Aufbau

Nachteile:

- lange Gesamtgröße
- viele bewegliche Bauteile / Fehlerquellen

1.4.2.3 Variante 3 - Lagerrad mit Saugnäpfe

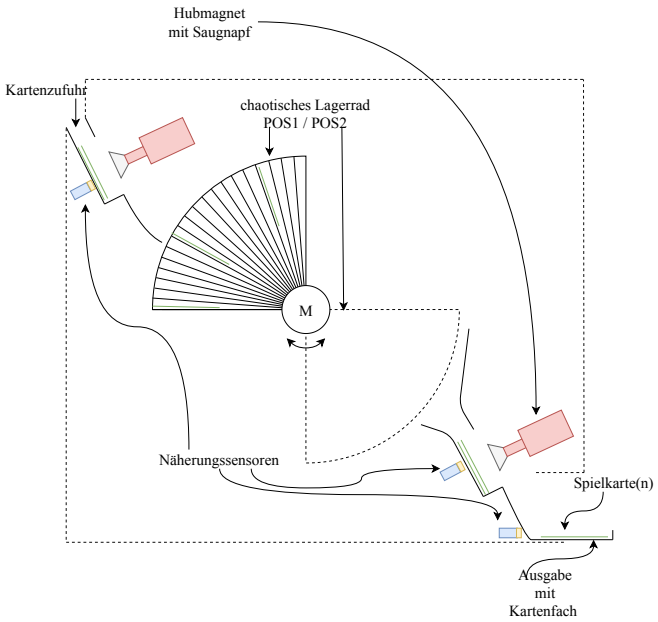


Abbildung 1.3: Variante 3

Das dritte Konzept besitzt ein identes Lagersystem wie das zweite Konzept, ein Lagerrad das in Fächer unterteilt ist und über einen Moter diverse Positionen einnehmen kann. Das Kartendeck wird in den oberen / vorderen Anfang der Maschine eingeführt. Dort liegt es schräg in einem Winkel von ca. 60°. Um sicherzustellen das sich Karten in dieser Halterung befinden ist ein Kapazitiver Sensor an der Unterseite angebracht. Ein Hubmagnet der im rechten Winkel zu den Karten über der Halterng angebracht ist, saugt jede Karte einzeln an indem ein Saugnapf der an einem Hubmagneten befestigt ist heruntergedrückt wird. Ist die Karte angesaugt, so wird der Hubmagnet von einer Feder in seine Ausgangsstellung zurückgebracht, dabei wird die Spielkarte durch eine Platte abgestreift und fliegt somit in das Lager des Lagerrades hinein. Dieser Prozess wird solange wiederholt bis sich alle Spilkarten des Kartendecks im Lagerrad befinden. Sind alle Karten im Lagerrad so dreht sich dieses und wirft die Karten auf der Rückseite der Maschine in ein Führung die die Karten in eine zweite Halterung befördern. Diese zweite Halterung ist ident Aufgebaut wie die erste. Die Karten werden nun wieder einzeln vom Saugnapf angesaugt und in ein Ausgabefach am Ende der Maschine befördert. Im Ausgabefach befindet sich ein Kapazitiver Sensor, dieser überprüft ob die Karten vom Spiler berreits genommen wurden oder nicht.

Durch die wenigen Bauteile die dieses Konzept besitzt, nämlich zwei Hubmagnete und einen Motor, ist das Konzept sehr fehlerunanfällig bei Bewegungsabläufe. Außerdem ist es durch die wenigen Bauteile im vergleich billiger als die anderen Konzepte. Ein Problem dieses Konzeptes ist seine Höhe und dass das Lagerrad sich in der Mitte der Maschine befindet, jedoch ist durch das geringe Gewicht des Lagerrades noch immer genügend stabilität vorhanden, auch wenn sich dieses mit voller Geschwindigkeit dreht.

Vorteile:

- billig
- wenig bewegliche Bauteile

Nachteile:

- hocher Aufbau

1.4.3 Ausgaberäder

Um die Karten weiterzubefödern werden bei zwei Konzepten Ausgaberäder benutzt, für diese gibt es verschiedene Konzepte dies sich in Preis, Herstellung und Funktionalität unterscheiden.
Das Ausgaberad ist mit einem Motor verbunden, dieses sorgt dafür das eine Karte von einem Kartendeck weiterbefördert wird.

- **Rundes Ausgaberad**

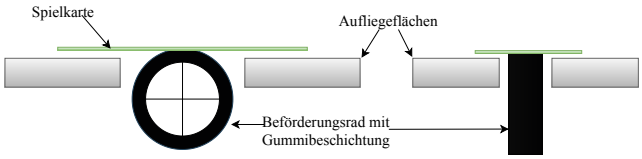


Abbildung 1.4: Rundes Ausgaberad

Die einfachste Möglichkeit dieses Rad zu entwerfen, wäre ein einfaches rundes Ausgaberad. Dieses Rad wäre mit einer Schicht umhüllt, welche die Reibung dem Rad und der Karte erhöht. Das runde Rad wäre einfach zu fertigen und würde somit wenig kosten und nur einen geringen Zeitaufwand haben. Jedoch ist das Rad in der Lage mehr als nur eine Karte mit sich mitzuziehen, da es durchgehen Kontakt mit der Spielkartenoberfläche hat, dies hätte zur Folge, das mehrere Spielkarten zugleich weiterbefördert werden.

- **Eliptisches Ausgaberad**

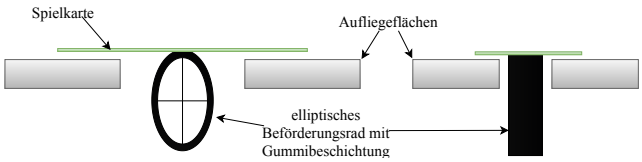


Abbildung 1.5: Eliptisches Ausgaberad

Ein elliptisches Ausgaberad wäre das nächste Konzept, dieses Rad wird auch wie beim ersten Rad mit einem Motor verbunden um somit Karten zu befördern. Durch die elliptische Form des Rades herrscht kein durchgehender Kontakt mit der Oberfläche der Spilkarte, aus diesem Grund ist die

Chance das mehrere Karten zugleich befördert werden minimiert. Jedoch setzt dies auch einen Motor vorraus der in kurzer Zeit ein hohes Drehmoment entwickelt, da die Karten schlagartig befördert werden. Das elliptische Rad würde in der Produktion auch mehr Kosten verursachen und wäre Zeitintesiver in der Hertellung.

- **Vergleich der Ausgaberäder**

Da das primäre Ziel unserer Arbeit das perfekte Mischen der Spielkarten ist, wäre es besser das elliptische Ausgaberad in betracht zu ziehen. Die Kosten die durch die aufwendigere Herstellung entstehen wären überschaulich und somit wäre es nur profitabel dieses Konzept zu wählen. Durch die Tatsache das das elliptische Rad das Kartendeck nur jede halbe umdrehung berrührt und nicht konstant, muss ein Motor für die Räder einegesetzt werden der sich schnell genug dreht um eine Karte mit einer berührung des Rades "herauszuschießen". Dies würde aber keine zusätzlichen Kosten verursachen. Aus diesen Gründen fällt die Wahl auf das **elliptisches Ausgaberad**.

1.4.4 Klemmmechanissmen

Um beim Ausgeben und beim Weiterbefördern der Karten die Chance zu minimieren das mehrere Karten auf einmal weiterbefördert werden, wird ein Klemmmechanissmus benutzt, dieser sorgt dafür das die Karten von außen Geklemmt werden, und somit nur die unterste Karte durch das Drehen des Ausgaberades weiterbefördert wird.

- **Primitiver Klemmmechanissmus**

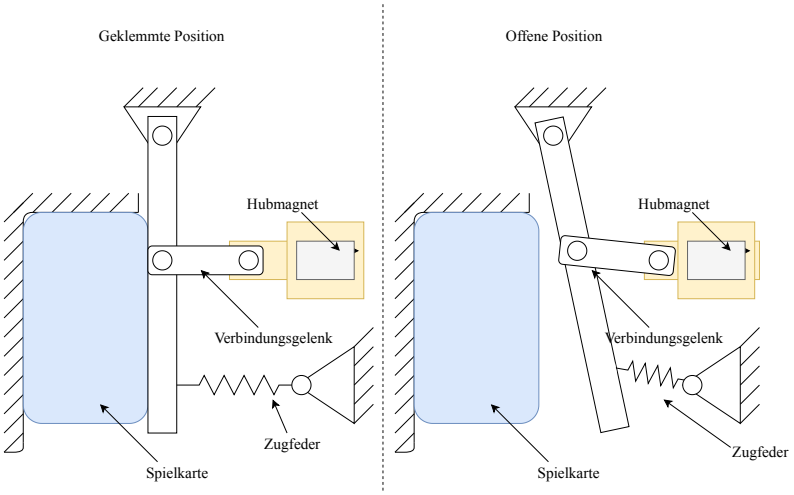


Abbildung 1.6: Primitiver Klemmmechanissmus

Dieser Klemmmechanismus ist der primitivste und einfachste, dadurch aber auch der billigste. Die Karten werden auf der einen Seite an eine feste Wand gedrückt auf der anderen Seite werden sie

durch ein bewegliches Gelenk fixiert. Dieses Gelenk wrid durch das einschalten des Hubmagneten in die geschlossene Position bewegt, soll das Gelenk wieder öffnen, so wird der Hubmagnet ausgeschalten und ein Zugfeder zieht das Gelenk wieder in seine Ausgangsposition zurück.

- **Komplexer Klemmmechanissmus**

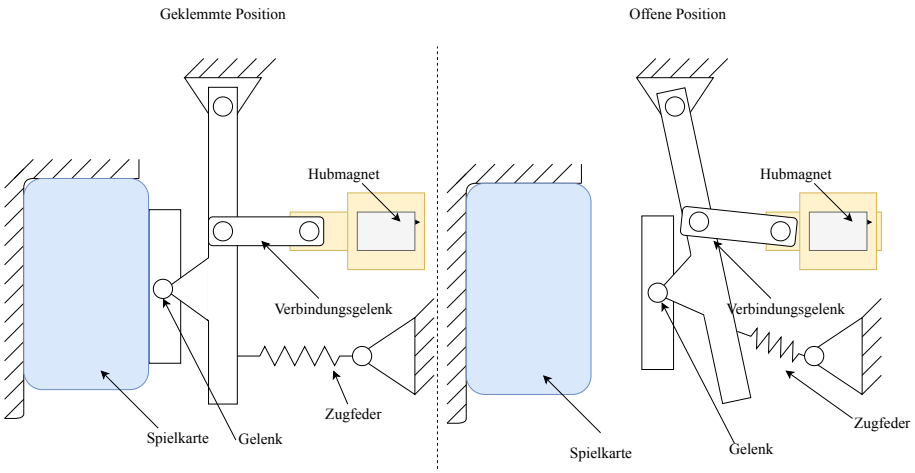


Abbildung 1.7: Komplexer Klemmmechanissmus

Bei diesem Mechanissmus werden die Karten auf der einen Seite durch eine feste Wand geklemmt, auf der anderen werden sie durch ein Gelenk geklemmt, dieses Gelenk ist beweglich und gleicht somit verschidene Kartengrößen aus. Um das Gelenk dieses Konzeptes zu schliesen wird ein Hubmagnet benötigt, wird dieser eingeschalten so schließt das Gelenk und passt sich der Kartengröße an. Soll es wieder geöffent werden, so wird der Hubmagnet ausgeschaltet und eine Zugfeder bringt den Mechanissmus wieder in den Grundzustand. Dieses Konzept ist aufwendiger zu realisieren wie das erste, jedoch kann es verschiedene Kartengrößen klemmen und gleicht sich den Karten an. Das Konzept ist dafür aber auch aufwendiger in der Produktion.

- **Gummi Klemmmechanissmus**

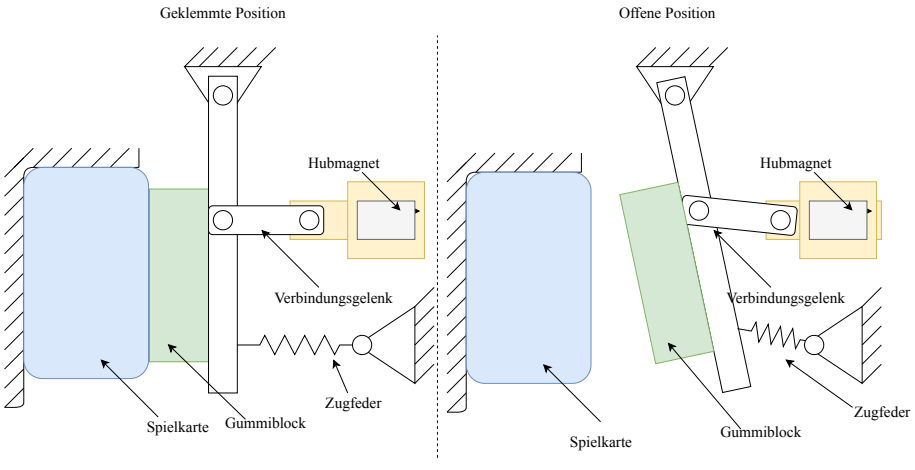


Abbildung 1.8: Gummi Klemmmechanissmus

Der letzte Mechanissmus ist ufnktionsmäsig gleich wie der primitive Klemmmecahnissmus, der einzige Unterschied besteht aus dem Gummiähnlichen Block der auf der Klemmfläche sitzt. Dieser soll sich den Karten anpassen und sorgt dafür das alle Karten gleichmäßig geklemmt werden, auch erhöht es die Reibung zwischen den Karten und der Klemmoberfläche. Dieses Konzept wäre in der Herstellung verglichsweise eifnach zu realisieren und dadurch auch billig in der Herstellung.

- **Vergleich der Klemmmechanissmen**

Da alle Mechanissmen die gleiche Anzahl an Bateile erfordern, spielt der Preis bei dieser Auswahl keine große Rolle. Aus diesem Grund sind die Kriterien Funktionalität und Aufwendigkeit in der Produktion. Der erst Mechanissmus wäre der einfachste in der Produktion, jedoch hat er durch sein nicht anpassungsfähiges Klemmgelenk Probleme mit verschiedenen Kartengrößen, aus diesem Grund fällt er bei diesem Auswahlverfahren durch. Der zweite Mechanissmus wäre der komplexe Klemmmechanissmus, dieser würde sich zwar verschiedenen Kartengrößen anpassen, wäre aber etwas aufwendiger in der Produktion und ist somit auch nicht die bevorzugte Wahl. Der letzte Klemmechanissmus der einen gummiartigen Block auf der Klemmfläche hat, wäre einfach in der Produktion und würde sich auch bei verschiedenen Kartengrößen anpassen, aus diesem Grund fätt die wahl auf den Gummi Klemmmechanissmus.

1.4.5 Seperation der Karten

Da bei "Variante 3 - Lagerrad mit SaugnäpfeSSaugnäpfe mit Unterdruckprinzip in Verwendung kommen würden, liegt das Problem vor, dass die Karten aneinander haften, dieses Problem entsteht durch den Unterdruck der beim Aufpressen des Saugnapfes entsteht. Um zu garantieren das nur eine Karten in das Lager befördert wir oder ausgegeben wird, müssen diese aneinander haftende Karten getrennt werden.

- **Selbständiges Loslösen der Karten**

Die billigste und primitivste Möglichkeit die Karten zu separieren wäre abzuwarten bis alle Karten sich durch die Schwerkraft von der eigentlich angesaugten Karte getrennt haben. Jedoch zeigten Versuche die durchgeführt wurden dass dies zu lange dauern würde und das Spielerlebniss somit beeinflussen würde.

		Sekunden					
		1. Durchgang	2. Durchgang	3. Durchgang	4. Durchgang	5. Durchgang	Mittelwert
Karten	20	9.69	9.32	8.92	8.54	4.53	8.2
	19	9.1	9.97	13.82	6.71	7.75	9.47
	18	15.39	8.93	6.51	13.05	15.25	11.826
	17	6.45	6.67	6.16	8.06	6.97	6.862
	16	10.76	6.52	10.62	11.37	6.97	9.248
	15	12.81	6.45	16.22	8.74	4.7	9.784
	14	3.46	9.78	11.72	5.85	6.39	7.44
	13	4.85	8.18	10.77	10.5	3.43	7.546
	12	15.1	5.37	12.36	8.54	12.31	10.736
	11	15.83	6.32	5.58	5.08	1.12	6.786
	10	7.14	8.4	11.21	12.76	4.43	8.788
	9	7.48	6.24	11.87	4.32	6.36	7.254
	8	4.68	5.65	8.61	5.2	4.41	5.71
	7	9.73	8.29	9.13	5.35	11.2	8.74
	6	5.41	5.16	8.05	4.98	9.73	6.666
	5	6.06	4.81	7.8	4.73	7.19	6.118
	4	4.86	9.87	7.74	9.51	7.4	7.876
	3	4.54	11.31	4.31	6.74	10.82	7.544
	2	4.83	2.19	4.13	3.78	5.71	4.128
						Mittelwert: 7.932	

Tabelle 1.1: Tabelle Haftzeit der Karten

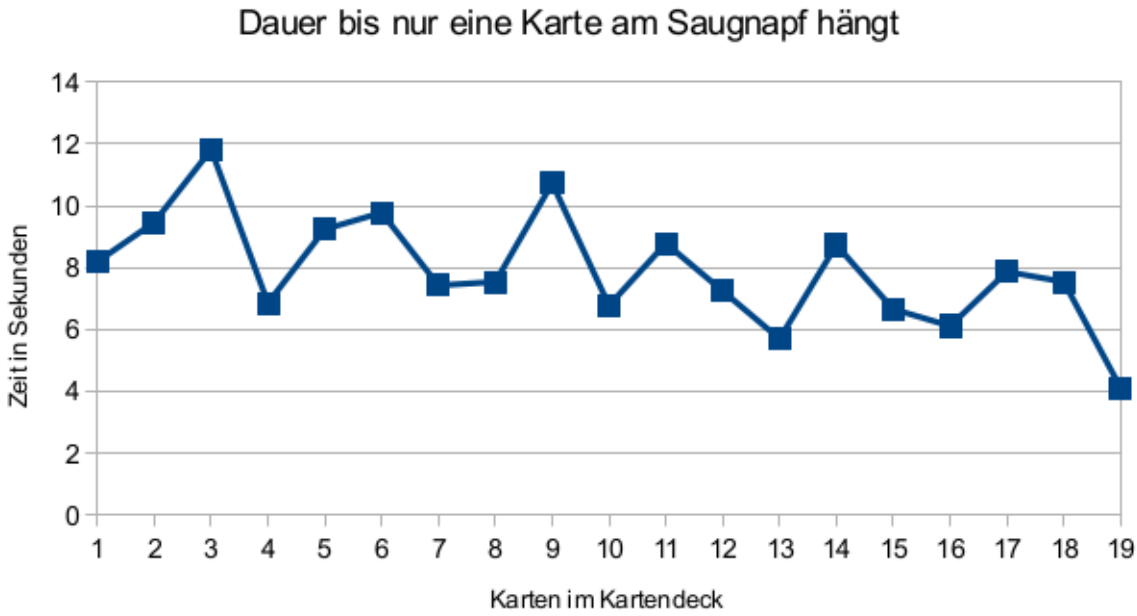


Abbildung 1.9: Diagramm Haftzeit der Karten

Wie man in Tabelle 1.1 und in Abbildung 1.9 erkennen kann dauert es im Durchschnitt 7,932 sekunden bis sich nur mehr die eigentlich angesaugte Karte in der Luft befindet. der Versuch wurde mit 1Kg Ansaugdruck durchgeführt. Es gab insgesamt 5 Durchgänge, auch wurden alle Möglichkeiten der Kartenanzahlen im Kartendeck berücksichtigt, so saugte der Saugnapf bei der ersten Durchgangsreihe die erste Karte an, die auf 19 anderen Karten liegt , bis hin zu einem Saugnapf der eine Karte ansaugt die nur auf einer einzelnen anderen Karte liegt.

Vorteile:

- billig
- keine zusätzlichen Bauteile benötigt

Nachteile:

- lange Wartezeit
- unzuverlässig

- **Rütteln**

Ein weiteres Konzept um die angesaugte Karten von den anderen zu Trennen wäre die Karten minimal zu rütteln. Da die angesaugt Karte viel stärken an dem Saugnapf haftet als die Karten, die nur an der angesaugten Karten haften, könnte der Hubmagnet oder der ansaugmechanissmus leicht gerüttelt werden, so würden die anderen Karten schneller den Unterdruck untereinander verlieren und sich somit schon in kurzer Zeit separieren lassen. Jedoch müsste für dieses Konzept ein Mechanissmus entwickelt und produziert werden der nur einen ausgewählten Berreich rüttelt, dies wäre mit weiteren Bauteilkosten und Aufwand verbunden.
Vorteile:

- zuverlässig

Nachteile:

- teurer
- zusätzliche Bauteile
- zusätzliche Größe

- **Abstreifbürsten**

Bei diesem Konzept sind Bürstenartige Abstreifvorrichtungen in der Kartenhalterung angebracht. Diese Bürsten sollen beim aufheben der angesaugeten Karte die anderen Karten die mitgehebt wurden abstreifen. Dabei ist es wichtig, dass die Bürste den richtigen Widerstand aufbringt. Ist der Widerstand zu hoch, wird auch die eigentlich angesaugt Karte mitabgestreift, ist er jedoch zu niedrig ist es möglich, dass die anderen Karten die mitgehoben werden nicht abgestreift werden.
Vorteile:

- zuverlässig
- billig

Nachteile:

- Karten werden seitlich abgenützt

- komplizierter Einbau

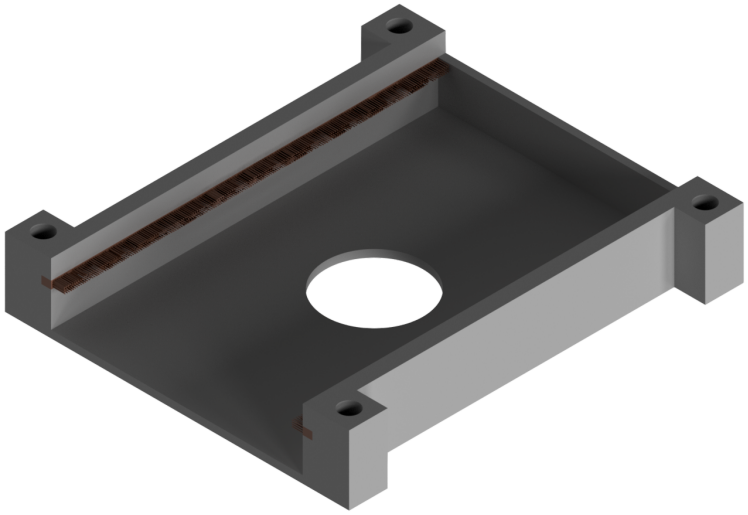


Abbildung 1.10: Ausgabe mit Bürsten

- **Gummiabsstreifer**

Dises Konzept funktioniert ähnlich wie das Äbstreifbürsten Konzept". Es besitzt seitlich Abstreifplatten die aus Gummi sind, diese sollen der angesaugten Karte helfen die die migezogenen Karten abzustreifen. Wie beim vorherigen Konzept muss hierbei das Gummi auch so dimensioniert werden, dass es einerseits nicht zu steif ist und die eigentlich angesaugt Karte abstreift, andererseits sollte es steif genug sein um die Karten die mitgehoben werden abzustreifen. Man die Steifigkeit des Gummistreifens verändern, indem man die Breite der Einschnitte erhöht oder vermindert.

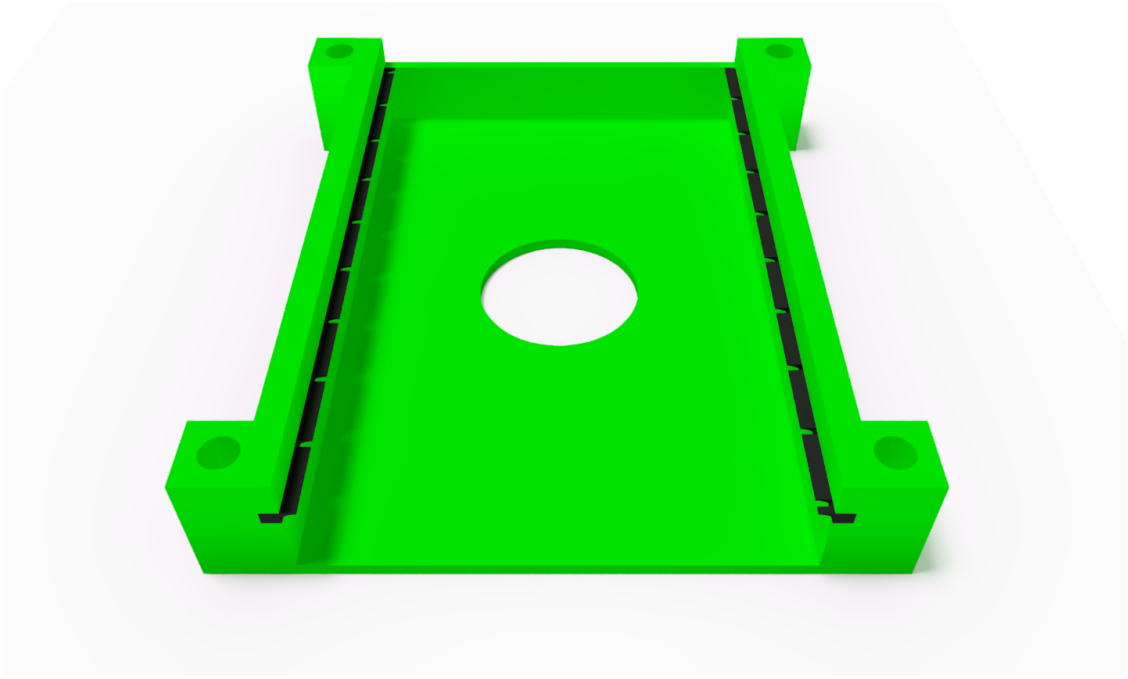


Abbildung 1.11: Ausgabe mit Bürsten

Vorteile:

- zuverlässig
- billig

Nachteile:

- Gummi wird spröde

1.4.6 Endauswahl der Varianten

Gegenüberstellung der Varianten

Variante	Vorteile	Nachteile
1 - Linearachsen	schnelles Mischen wenige Fehlerquellen	teuer großer Aufbau instabil
2 - Lagerrad mit Ausgaberäder	stabil niedriger Aufbau	lange Gesamtgröße viele Bewegliche Bauteile / Fehlerquellen
3 - Lagerrad mit Saugnäpfe	billig wenig bewegliche Bauteile	hoher Aufbau

Tabelle 1.2: Vergleich der Varianten

Begründung der Wahl

Alle drei Varianten wurden durchdacht und bieten ihre individuellen Vorteile. Durch den enormen Preis von Variante 1 ist diese aber nicht für unser Projekt geeignet. Das optimale Konzept des Mischens wird am besten durch Variante 3 realisiert, da diese die niedrigste Wahrscheinlichkeit

aufweist, mehrere Karten aufeinmal in das Lager zu befördern. Da Variante 3 im Vergleich zur Variante 2 weniger Bauteile besitzt, ist diese Variante sowohl Preislich ansprechender für uns sowie auch die Tatsache das Fehlerquellen durch bewegliche Bauteile minimiert werden.
Somit fällt die finale Wahl auf **Variante 3**.

1.5 Konstruktion

1.5.1 Konstruktion des Lagerrades

Das Lagerrad soll durch das zufällige Rotieren das Mischen der Karten ermöglichen. Im Verlauf der Diplomarbeit wurde das Lagerrad mehrfach überarbeitet. Die grundsätzliche Form dieses Rades besitzt ein virtel eines Hohlzylinders der in mehreren Fächern unterteilt ist. Die erste Version dieses Rades besaß 20 Unterteilungen.

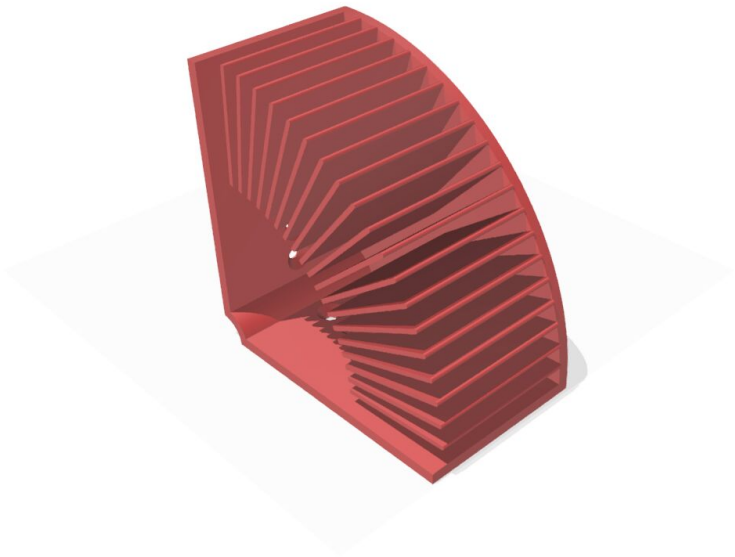


Abbildung 1.12: Lagerrad 20 Fächer (geöffnete Außenwand zur vereinfachten Ansicht)

In der oben angezeigen Grafik wurde die Außenwand entfernt um eine bessere Einsicht in das Bauteil zu bekommen. Durch die 20 unterteilungen muss der Schrittmotor genauere Positionen anfahren, außerdem wäre der Aufwand beim produzieren groß. Aus diesem Grund entschieden wir und die Unterteilungen zu minimieren, und kamen zu dem Entschluss, dass 4 Unterteilungen für ein optimales Mischen ausreichen. Diese Konstruktion wurde nach einem Stecksystem entworfen, sodass man die einzelnen Teile herstellen kann und diese dannach zusammenstecken und verkleben kann.

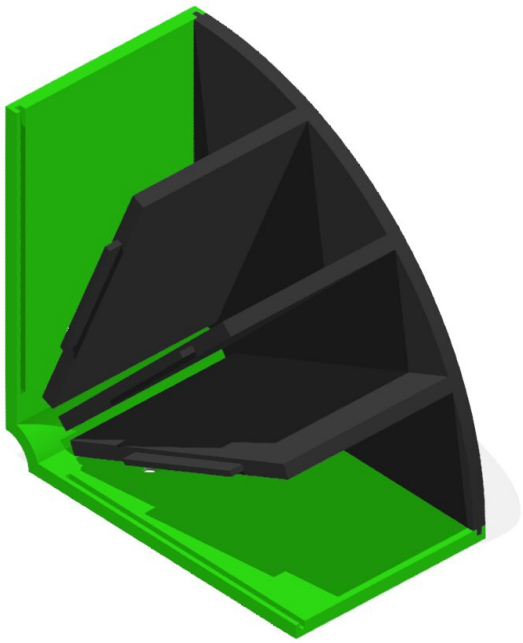


Abbildung 1.13: Lagerrad 4 Fächer (geöffnete Außenwand zur vereinfachten Ansicht)

Auch in der oben gezeigten Grafik wurde die Außenwand entfernt um das Bauteil besser zu veranschaulichen. Das Baumodul besteht aus zwei Seitenplatten sowie zwei Außenplatten die mit einem Stecksystem verbunden sind. Im inneren befinden sich 3 Trennplatten die mit einem Stecksystem mit der Außenplatte verbunden sind.

1.5.2 Konstruktion der Kartenhalterung

1.5.3 Konstruktion der Kartenentnahme

1.5.4 Konstruktion der Endschalterhalterung

1.5.5 Konstruktion des Gehäuses

1.6 Berechnungen und Dimensionierungen

1.7 Teilaufbau

1.8 Analyse

2 Elektronik

2.1 Anforderungen

Der elektrische Teil dieser Diplomarbeit umfasst das Entwerfen und Entwickeln eines Schaltplans und einer dazugehörigen Leiterplatte sowie das Layouten dieser und deren Bestückung. Die Platine, gesteuert vom Mikroprozessor Atmega324P, hat die Aufgabe drei kapazitive Sensoren einzulesen, zwei Hubmagneten anzusteuern, einen Endschalter einzulesen und einen Schrittmotor anzusteuern. Zudem hat sie die Aufgabe, jegliche elektrische Bauteile des Automaten mit Spannung zu versorgen. Außerdem soll der Mikroprozessor im Stande dazu sein, Befehle von einem Raspberry Pi 3B+ über UART zu erhalten. Die Möglichkeit, Debugging über eine zweite UART – Schnittstelle via Mini USB zu betreiben, soll ebenfalls gegeben sein.

2.1.1 Zeitplan

2.2 Variantenvergleiche und Konzeptionierungen

2.2.1 Mikrocontroller

Da unser finales Konzept Einiges an Logikansteuerungen benötigt, ist es sinnvoll die Ansteuerung der Peripherie auf einen Mikrocontroller auszulagern. Wir stellten die Anforderungen einer doppelt vorhandenen UART Schnittstelle, eine zur Kommunikation mit dem Raspberry Pi und eine, welche als Debugging-Schnittstelle genutzt werden sollte. Außerdem schien es uns sinnvoll, einen Mikroprozessor der Atmega-Familie auszuwählen, da mit dieser bereits Erfahrung gesammelt wurde.

2.2.1.1 ATmega162

Dieser Mikrochip besitzt einen Flash-Speicher von 16kiB, einen SRAM in einer Größe von 1kiB und einen E²PROM-Speicher von 512 Bytes. Der ATmega162 verfügt über eine SPI und zwei UART Schnittstellen sowie 35 I/O Register und zwei Timer. Er kann mit einer Gleichspannung von 2.7V bis 5.5V betrieben werden.

2.2.1.2 ATmega324P

Der ATmega324P ist ein Mikrochip, welcher 32kiB an Flash-Speicher, 1kiB an E²PROM-Speicher sowie 1kiB an SRAM mit sich bringt. Er verfügt über 32 I/O Register, eine SPI Schnittstelle, eine I²C Schnittstelle, zwei UART Schnittstellen sowie drei flexible Timer. Außerdem lässt er sich mit einer Gleichspannung von 1.8V bis 5V betreiben.

2.2.1.3 ATmega128

Der ATmega128 besitzt einen E²PROM-Speicher von 4kiB, einen 123kiB großen Flash-Speicher und einen SRAM von 4kiB. Er besitzt zwei UART-, eine SPI- sowie eine I²C-Schnittstelle. Zusätzlich bietet er vier Timer und 53 I/O Pins. Er kann mit einer Gleichspannung von 4.5V bis 5.5V betrieben werden.

2.2.1.4 Mikrocontroller-Auswahl

	ATmega162	**ATmega324P**	**ATmega128**
Preis	gering	gering	mittel
Flash-Speicher	16kiB	32kiB	128kiB
EEPROM-Speicher	512B	1kiB	4kiB
I/O-Pins	35	32	53
UART-Schnittstellen	2	2	2
SPI-Schnittstelle	ja	ja	ja
Versorgungsspannung	2,7V-5V	1,8V-5V	4,5V-5V

Tabelle 2.1: Vergleich der Mikrocontrollerattribute

Unsere Entscheidung fiel auf den ATmega324P, da dieser die nötigen Parameter, wie eine Versorgungsmöglichkeit mit 3,3V und einen ausreichend großen, programmierbaren Speicher, mit sich bringt.

2.2.2 Schrittmotor-Ansteuerung

Um jenen, im Automaten verbauten, Schrittmotor ansteuern zu können, ist eine Baugruppe von Nöten. Für das Ansteuern von Schrittmotoren gibt es verschiedenste Möglichkeiten. In den folgenden Unterkapiteln werden einzelne genannt, ein Variantenvergleich durchgeführt und sich für eine Variante entschieden.

2.2.2.1 DIY-H-Brücke

Mithilfe einer sogenannten H-Brücke ist es möglich, Schrittmotoren mitunter der Verwendung eines Mikrocontrollers anzusteuern. Der von uns ausgewählte Schrittmotor 12HS19-2004S1 ist bipolarer Art und weißt 2 Spulen auf, weshalb er sich von zwei H-Brücke steuern lässt. Für das Ansteuern einer H-Brücke werden drei Steuersignale benötigt: zwei digitale Signale sowie ein PWM-Signal.

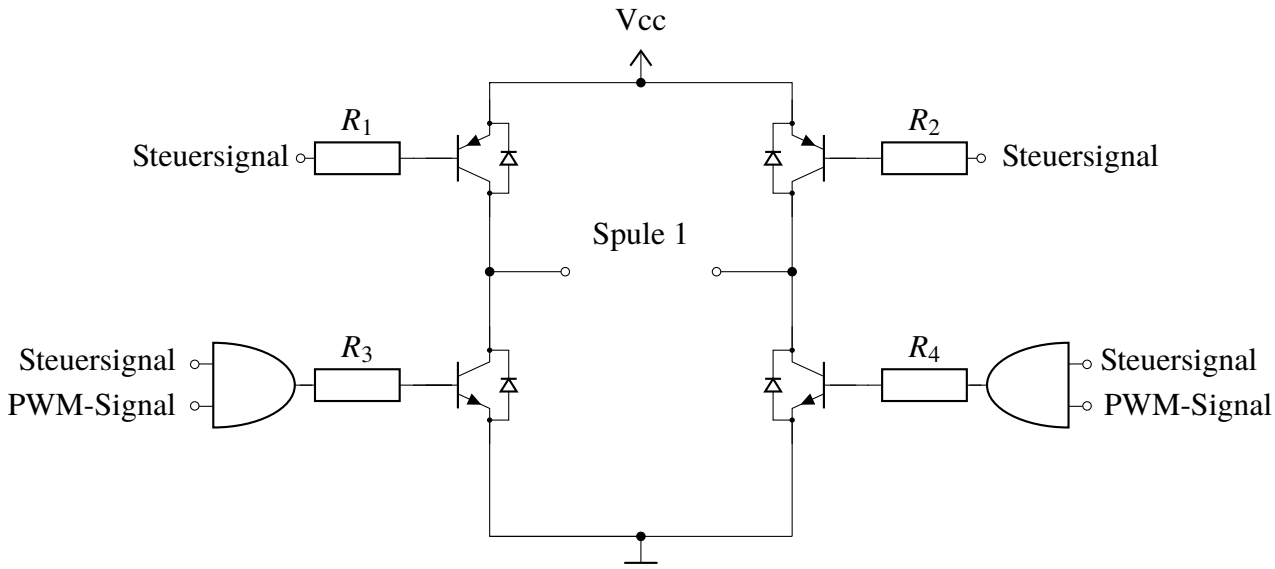


Abbildung 2.1: Aufbau einer H-Brücke

Grundsätzlich ist zu sagen, dass eine H-Brücke aus vier Transistoren und vier Schutzdioden besteht.
Ein konkreter Aufbau einer H-Brücke kann der Abbildung 2.1 entnommen werden. Bei dieser Variante einer H-Brücke sind zwei NPN-Transistoren und zwei PNP-Transistoren vorhanden, von denen jeder einen Basisvorwiderstand besitzt. Wie in Abbildung 2.1 zu erkennen ist, befindet sich vor jedem Basiswiderstand der NPN-Transistoren ein UND-Gatter, mit dem jeweils ein digitales Steuersignal und das PWM-Signal zusammengeführt werden. Zusätzlich befindet sich parallel zu jedem Bipolartransistor eine Schutzdiode. Diese haben folgende Aufgabe:

Ein Nebeneffekt der Funktionsweise eines Motors, ist die Erzeugung von Energie. Bei einem Deaktivieren der Transistoren, um den Motor zu stoppen, muss diese erzeugte Energie des Motors auf irgendeine Weise freigesetzt werden können. Um ein Beschädigen der Transistoren zu vermeiden, verwendet man diese Dioden, um dem Strom einen Pfad zu bieten, der die Energie freisetzt.

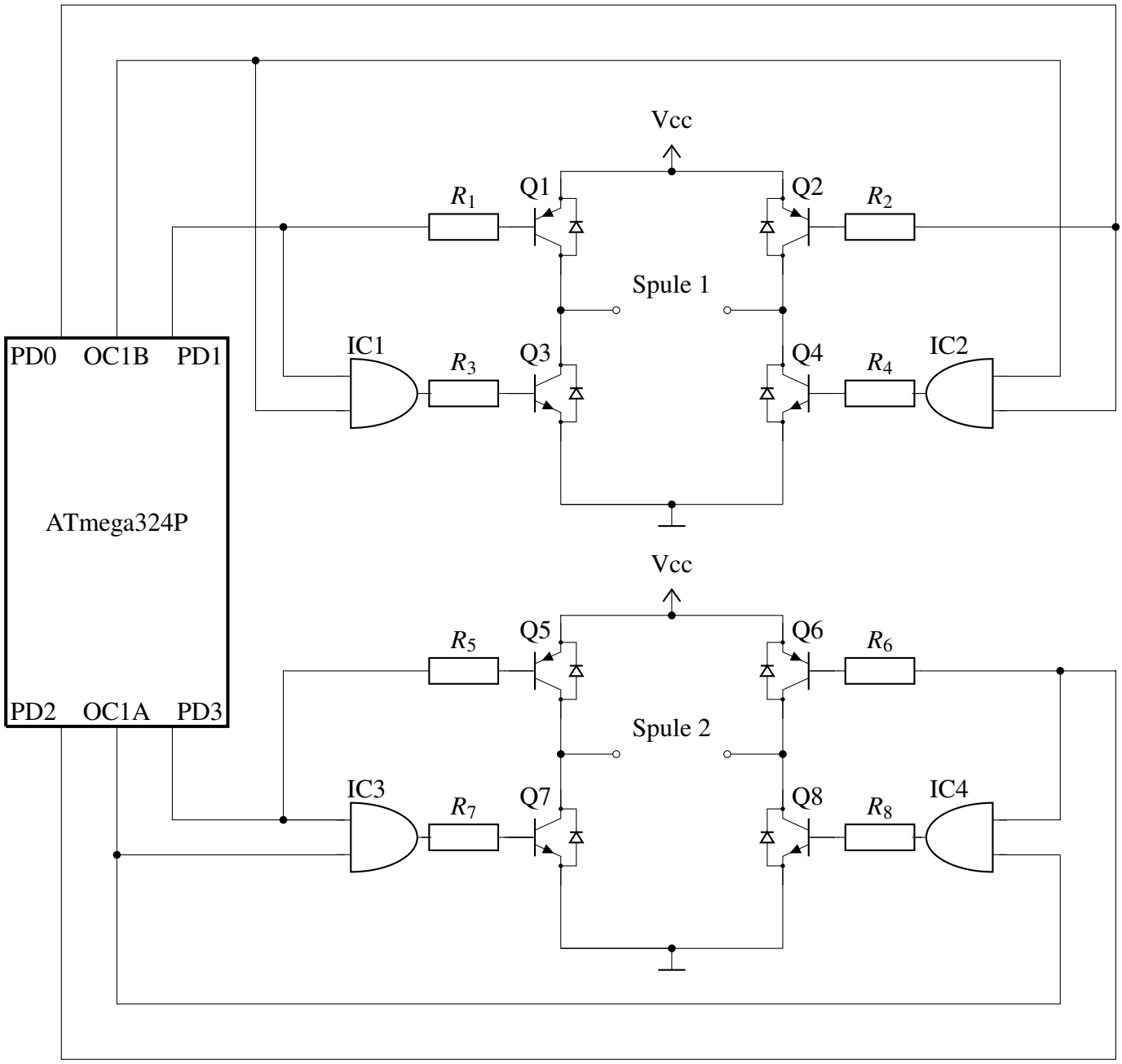


Abbildung 2.2: Anschluss einer H-Brücke an einen Mikrocontroller

In der obigen Abbildung 2.2 wird eine konkrete Verbindung von zwei H-Brücken in Kombination mit einem Mikrocontroller dargestellt. Da sich die Ansteuerung der beiden H-Brücken gleich gestaltet, wird zur vereinfachten Funktionserklärung nur die obere behandelt.

Mithilfe des Timer 0 wird am Ausgang OC1B des Mikrocontrollers ein PWM-Signal erzeugt, mithilfe dessen, abhängig vom gewählten Duty-Cycle, Spannungssignale in gewissen Zeitabständen geschickt werden. Dadurch befindet sich jeweils an einem Eingang des UND-Gatters mit der Bezeichnung IC1 und an einem Eingang des UND-Gatters mit der Bezeichnung IC2 ein Signal mit einem HIGH-Pegel (Spannung) oder LOW-Pegel (keine Spannung).

Je nach gewollter Drehrichtung, ist der Motor über die Transistoren anders anzusteuern:
Um den Schrittmotor nach Rechts drehen zu lassen, muss am Ausgang PD0 ein HIGH-Signal erzeugt werden. Am Ausgang PD1 muss ein LOW-Signal erzeugt werden, da ansonsten der PNP-Transistor Q1 sperrt und kein Stromfluss gewährleistet wäre. Durch das HIGH-Signal, ausgehend vom Ausgang OC1B, befindet sich ein HIGH-Signal am unteren Eingang des UND-Gatters IC2. Wenn nun ein HIGH-Pegel ausgehend vom Ausgang PD0 kommt, liefert das UND-Gatter IC2 ein HIGH-Signal weiter an den NPN-Transistor Q4, welcher somit leitend wird und den Stromkreis schließt.

Um den Schrittmotor nach Links drehen zu lassen, wird am Ausgang PD1 ein HIGH-Signal gesetzt. Am Ausgang PD0 muss ein LOW-Signal erzeugt werden, da ansonsten der PNP-Transistor Y sperrt und kein Stromfluss gewährleistet wäre. Durch das HIGH-Signal, ausgehend vom Ausgang PD1, befindet sich ebenfalls ein HIGH-Signal am oberen Eingang des UND-Gatters. Wenn nun ein HIGH-Pegel vom Ausgang OC1B bereitgestellt wird, liefert das UND-Gatter IC1 ein HIGH-Signal weiter an den NPN-Transistor Q3, welcher somit leitend wird und ein Fließen des Stromes durch die Motorwicklung bewirkt.

- **Vorteile**
  - günstig
- **Nachteile**
  - viele Bauteile
  - Bauteile müssen erst dimensioniert werden
  - Kosten könnten sich aufgrund hoher Verlustleistung durch einen Kühlkörper erhöhen

2.2.2.2 Schrittmotor-Treibermodule

Schrittmotor-Treiber sind Module, welche mithilfe von externen Signalen in der Lage sind, Schrittmotoren steuern zu können.

- **Vorteile**
  - platzsparend
  - leicht austauschbar
  - leicht verwendbar

- **Nachteile**
  - Kosten könnten sich, aufgrund eines Kühlkörpers erhöhen

2.2.2.3 A4988

Das Modul kann mit einer Eingangsspannung von 8V bis 35V arbeiten und kann ohne den Einsatz eines Kühlkörpers bis zu 1A pro Phase liefern. Der Treiber ist jedoch darauf ausgelegt, einen Strom von 2A mit zusätzlicher Kühlung zur Verfügung stellen zu können. Es besteht kein Bedarf daran Phasenfolgetabellen, Hochfrequenz-Steuerleitungen oder komplexe Schnittstellen zu programmieren. Das A4988 Treiber-Modul von Allegro bietet viele Möglichkeiten. Zu nennen ist die Eigenschaft einer einstellbaren Begrenzung des abgegebenen Stroms über ein Potentiometer. Dadurch können Spannungen über der Nennspannung des Schrittmotors verwendet werden, um höhere Schrittgeschwindigkeiten zu erreichen. Zusätzlich wird die Gewährleistung eines Überstromschutz und Übertemperaturschutzes sowie eine Überspannungsabschaltung versichert. Erdschluss- und Kurzschlussschutz sind ebenfalls gegeben. Der Schrittmotortreiber bietet fünf verschiedene Mikroschrittauflösungen: Voll-, Halb-, Viertel-, Achtel- und Sechzehntel-Schrittmodus.

2.2.2.4 DRV8825

Diesem Treiber-Board von Texas Instruments ist es Möglich mit einer Spannung von 8,2 V bis 45 V zu arbeiten. Er kann ohne eine bestimmte Kühlung bis zu 1,5 A pro Phase liefern. Ausgelegt ist dieses Modul auf einen Strom von 2,5A, welche mit zugeführter Kühlung auch getrieben werden können. Der DRV8825 arbeitet mit einem Logikpegel von 3,3V bis 5V und bietet die Möglichkeit einer einstellbaren Strombegrenzung. Das Modul besitzt ebenfalls die Eigenschaften eines Überstrom- und Übertemperaturschutzes sowie sechs Mikroschrittauflösungen, welche bis zu einer Zweiunddreißigstel-Auflösung reichen. Die Besonderheit eines SLEEP-MODE mit geringen Stromverbrauch und eines eingebauten Unterspannungsschutzes sind zusätzlich gegeben. Der DRV8825-Treiber verfügt über eine Schnittstelle und eine Pinbelegung, welche beinahe mit denen des A4988-Schrittmotortreibers identisch sind, sodass er in vielen Anwendungen als leistungsstärkerer Ersatz verwendet werden kann.

2.2.2.5 TB6600

Dieser Ein-Achsen-Schrittmotortreiber ist für Hybride Schrittmotoren mit 2 oder 4 Phasen geeignet. Der TB6600 arbeitet mit einer Spannung von 9V bis 40V und ist für einen Ausgangsstrom von 0,7A bis 4A ausgelegt. Das Modul bringt die Eigenschaften eines Übersitzungs-, Kurzschluss- und Überstromschutzes sowie einen großflächigen Kühlkörper mit sich. Der Ausgangsstrom ist mittels DIP-Schalter in acht Schritten wählbar. Sechs verschieden wählbare Mikroschrittauflösungen, welche bis zu einer Auflösung von Zweiunddreißigstel reichen, werden zur Verfügung gestellt. Die Kontrolllogik dieses Treibers umfasst eine Spannung von 5V. Das Gewicht von 200dag und eine Größe von 57mm x 96mm x 28mm unterscheiden sich wesentlich von den anderen Treibermodulen.

2.2.2.6 Treiber-Auswahl

	A4988	**DRV8825**	**TB6600**
Preis	gering	gering	mittel
Max. Strom	2A	2.2A	4A
Spannungsbereich	8V-35V	8.2V-35V	9V-40V
Logik-Pegel	3.3V-5V	3.3V-5V	5V
Max. Schrittauflösung	16	32	32
Größe	klein	klein	groß

Tabelle 2.2: Vergleich der Treibermodule

Da unsere Wahl auf den Schrittmotor 12HS19-2004S1 gefallen ist, wäre es ratsam einen Treiber auszuwählen, welcher 2A Strom zur Verfügung stellen kann. Unsere Treiber-Auswahl ist somit auf den DRV8825 gefallen, da dieser alle erforderlichen Kriterien, wie einen geringen Preis und den 3,3V Logikpegel, erfüllt und ein gutes Allroundpaket mit sich bringt. Ein Begründung, warum die Wahl auf das DRV8825 Modul und nicht das A4988 Modul fiel, sind die größere Spannungsbandbreite, der höhere maximale Strom pro Phase sowie die höhere maximale Schrittauflösung.

2.2.3 Detektion der Kartenposition

2.2.3.1 Kapazitive Sensoren

Die Funktion von kapazitiven Sensoren ist mit der eines offenen Kondensators zu vergleichen. Zwischen der Messelektrode und der GND - Elektrode bildet sich ein elektrisches Feld. Dringt ein Material mit einer Dielektrizitätszahl εr größer als Luft in das elektrische Feld ein, so vergrößert sich je nach dieser Zahl des Material die Kapazität des Feldes. Es wird zwischen zwei Arten von kapazitiven Sensoren unterschieden:

Sensoren mit GND-Elektrode
Ein bündiger Einbau dieser Sensoren ist möglich, da sich ihr Messfeld von der Messelektrode zur integrierten GND-Elektrode ausbreitet. Diese Variante dient sich gut zur Detektion von nicht leitenden Materialien.

Sensoren ohne GND-Elektrode
Ein bündiges Einbauen dieser Art ist nicht vorteilhaft. Eine integrierte GND-Elektrode ist nicht vorhanden, sie wird nämlich vom zu konstatierenden Objekt dargestellt. Diese Bauform weist eine geringe Empfindlichkeit gegen Verschmutzung auf. Für hohe Schaltabstände sind leitende und geerdete Gegenstände von Vorteil.

2.2.3.2 Ultraschall-Sensoren

Die meisten Ultraschall-Sensoren arbeiten nach dem Prinzip der Laufzeitmessung von hochfrequenten Schallimpulsen. Ein Sensor sendet zyklisch einen kurzen, hochfrequenten Schallimpuls aus, welcher sich in der Luft fortpflanzt und am getroffenen Gegenstand reflektiert wird. Das Echo wird vom Sensor wieder aufgenommen und aus der Zeitspanne zwischen dem Zeitpunkt des Absendens und dem Zeitpunkt des Erfassens wir der Abstand vom Sensor berechnet. So ist es diesem Art von Sensor möglich, unterschiedlichste Materialien wie Metall oder Holz aufzufassen. Lediglich schalldämpfende Materialien können nur schwer erfasst werden. Diese Art von Sensoren ist in der Lage berührungslos Objekte zu erkennen und ihre Entfernung zum Sensor zu messen. Ein fast wartungsfreier Betrieb ist möglich.

2.2.3.3 Optische Sensoren

Ein optischer Sensor sendet über seine eigene Lichtquelle einen Lichtstrahl aus. Zu unterscheiden sind hierbei Lichtschranken und Reflexionstypen.

- Optische Sensoren des Lichtschrankentyps detektieren Unterbrechungen einer Lichtachse, welche durch ein Zielobjekt hervorgerufen werden.
- Lichtsender und Empfänger sind baulich getrennt.
- Sensoren des Reflexionstyps werden zur Erfassung eines vom Zielobjekt reflektierten Lichtstrahls eingesetzt.

Optische Sensoren besitzen eine fast wartungsfreie, langfristigen Betrieb, da die Detektion kontaktlos erfolgt. Diese Art von Sensoren kann für nahezu jedes beliebige Material eingesetzt werden. Es sind große Erkennungsabstände möglich.

2.2.3.4 Sensoren-Auswahl

	Kapazitive Sensoren	**Ultraschall-Sensoren**	**Optische Sensoren**
Preis	gering	hoch	mittel
Platzbedarf	gering	gering	mittel
Genauigkeit	hoch	mittel	hoch

Tabelle 2.3: Vergleich der Sensorattribute

Unsere Wahl fiel auf einen kapazitiven Sensor da dieser für uns bereits für einen erschwinglichen Preis alles Nötige mit sich bringt: einen hohen Grad an Genauigkeit sowie einen geringen Platzbedarf. Somit fiel unsere Wahl auf den LJC18A3, welcher mit einer Versorgungspannung von 6V bis 36V arbeitet und metallische und nichtmetallische Gegenstände in einem Schaltabstand von 1 bis 10mm erfassen kann. Dieser Sensor konnte mit einem sehr geringen Preis in unserem Besitz gebracht werden und zeigte schon bei den ersten Tests, dass er seiner Aufgabe gewachsen war.

2.2.4 Erfassung der Position des Motors

2.2.4.1 Endschalter

Diese Realisierung umfasst einen Endschalter, welcher als Referenzpunkt vor jedem Mischvorgang angefahren wird, von dem aus die benötigten Schritte aufgetragen werden.

2.2.4.2 Lichtschranke

Eine Lichtschranke würde, wie der Endschalter, am Anfang angefahren werden und so ebenfalls als Referenzpunkt fungieren.

2.2.4.3 Varianten-Auswahl

	Endschalter	**Licntschranke**
Preis	sehr gering	mittel
Platzbedarf	sehr gering	mittel
Genauigkeit	mittel	hoch

Tabelle 2.4: Vergleich der Sensorattribute

Unsere Wahl fiel auf den Endschalter, da dieser leicht in kürzester Zeit zu einem geringen Preis erworben werden konnte und ausreichend Genauigkeit mit sich bringt.

2.2.5 Spannungsversorgung

2.2.5.1 Doppel-Schaltnetzteil mit zweierlei Ausgangsspannungspegeln

Diese Variante wäre mit einem Netzteil zu realisieren, welches lediglich über einen Kaltgerätestecker mit einer Netzspannung von 230V AC in Verbindung gebracht werden müsste, um die Versorgung der Peripherie zu gewährleisten. Sensoren, Schrittmotor und Hubmagneten würden über den 12V DC Ausgang versorgt werden, die Versorgung des Mikrocontrollers und des Raspberry Pi 3B+ würde über den 5V Ausgang des Schaltnetzteils realisiert werden. Es müsste jedoch ein Pegelwandler zwischen dem Raspberry Pi und dem Mikrocontroller eingebaut werden, da die Spannungslogik des Raspberry auf 3.3V basiert und die des Mikrocontrollers auf 5V.

2.2.5.2 Schaltnetzteil und Schaltregler

Bei dieser Realisierung handelt es sich um ein mit 230V Wechselspannung versorgtes Netzteil, welches diese in ein 12V Gleichspannungssignal tranferiert. Jene 12V werden von jeglichen Aktoren und Sensoren genutzt. Außerdem wird mithilfe der 12V ein Schaltregler mit Spannung versorgt, welcher die 5V Spannungsversorgung des Raspberry Pi bereitstellt. Über die GPIO Pins des Raspberry Pi wird folgend der Mikrocontroller mit 3.3V versorgt.

2.2.5.3 Schaltnetzteil und Festspannungsregler

Bei dieser Alternative wird ein Schaltnetzteil mit 230V AC versorgt, welches ein 12V DC Signal am Ausgang bereitstellt. Dieses wird für die 12V Peripherie genutzt sowie als Versorgung für einen Festspannungsregler, welcher 12V zu 5V umwandelt. Mit diesem Signal ist die Versorgung des Raspberry Pi bereitgestellt. Die Versorgung des Mikrocontroller wird über zwei 3.3V GPIO Pins gewährleistet.

2.2.5.4 Auswahl der Spannungsversorgung

	Doppel-Schaltnetzteil	**SN und Schaltregler**	**SN und Festspannungsregler**
Preis	hoch	mittel	sehr gering
Platzbedarf und Gewicht	hoch	gering	gering
Bauteilbedarf	gering	mittel	sehr gering

Tabelle 2.5: Vergleich der Spannungsversorgungsmodul-Attribute

Unsere Wahl fiel auf die Variante des 12V Schaltnetzteils mit 5V Festspannungsregler, da diese für einen geringen Preis erhältlich ist und einen sehr geringen Bedarf an Platz und Bauteilen mit sich bringt.
Den Mikrocontroller versorgen wir mit dem 3.3V Ausgang des Raspberry Pi 3B+. Aufgrund der Tatsache, dass der Mikrocontroller mit einer Versorgung von 5V auch mit einer Logik von +5V arbeiten würde und somit ein Pegelwandler zu 3.3V nötig wäre, verwarfen wir die Idee diesen mit 5V zu versorgen.

2.2.6 Blockschaltbild der gesamten Elektronik

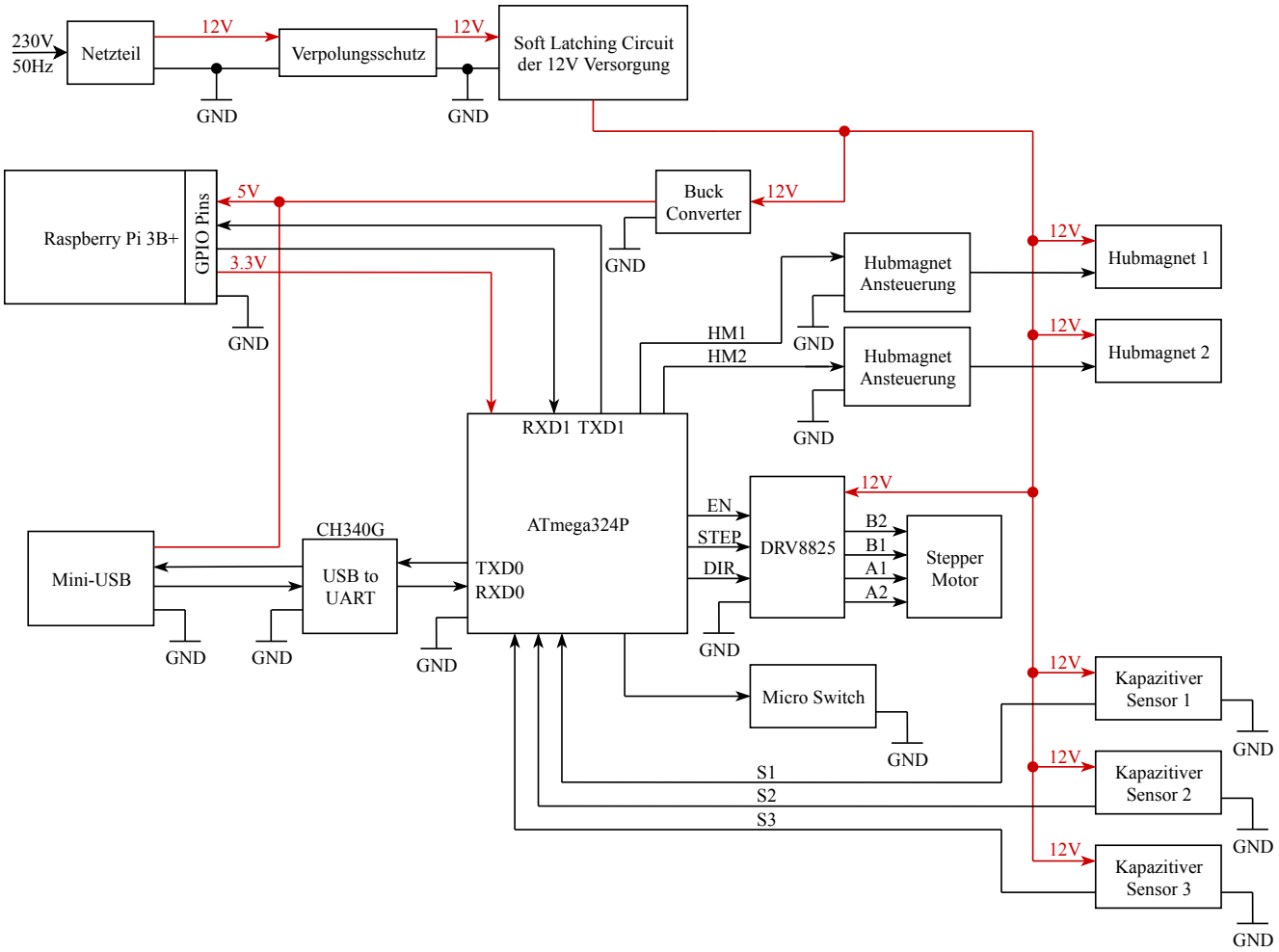


Abbildung 2.3: Blockschaltbild der gesamten Elektronik

2.3 Schaltplan

2.3.1 Verpolungsschutz

Da die Versorgungsspannung der elektrischen Baugruppen über eine Klemme eingespeist wird und eine Verwechslung der Anschlusspolaritäten nicht unwahrscheinlich ist, vermag es einen Verpolungsschutz zu verwerden um die dahinter gelegene Elektronik zu schützen.

2.3.1.1 Verpolungsschutz mit einer Diode

Die einfachste Methode diesen Schutz zu gewährleisten, ist über eine Diode. Jedoch bringt diese Methode einen erheblichen Nachteil von einem Spannungsabfall von bis zu 1V mit sich, was bei 12V immerhin 8.33 Prozent entsprechen.

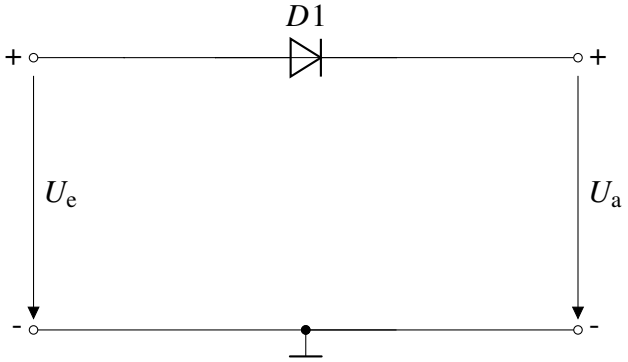


Abbildung 2.4: Variante mithilfe einer Diode

2.3.1.2 Verpolungsschutz mit einer Sicherung und einer Diode

Diese Realisierung bringt die Verwendung einer Sicherung und einer Diode mit sich.
Bei verpolter anliegender Eingangsspannung brennt die Sicherung, bei ausreichendem, von der Spannungsversorgung zur Verfügung gestelltem Strom, durch. Jedoch entsteht die negative Tatsache einer verpolten Spannung von bis zu 1V, bis die Sicherung auslöst.

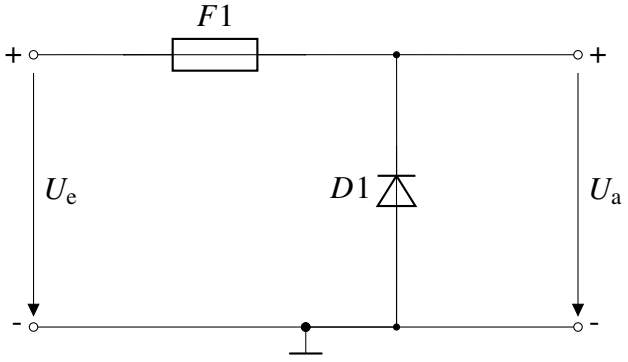


Abbildung 2.5: Zerstörerische Variante mithilfe einer Sicherung und einer Diode

2.3.1.3 Verpolungsschutz mithilfe eines P-Kanal MOSFET

Der P-Kanal MOSFET leitet, wenn das Gate um U_{GSth} negativer ist als der Spannungspegel anliegend am Source. Das Anlegen einer korrekt gepolten Spannung am Eingang des MOSFET bewirkt ein Leiten der Bulk-Diode, sodass am Sourceeingang die Eingangsspannung ankommt. Weil die am Source anliegende Spannung weitaus positiver als U_{GSth} ist, leitet der MOSFET . Die eingebaute Z-Diode begrenzt U_{GS} auf einen für den MOSFET ungefährlichen Wert. Die Tatsache, dass der MOSFET quasi verkehrt herum leitet und der Drain somit positiver als der Source ist, gilt als redondant, da es sich lediglich um einige Millivolt Unterschied handelt. Sollte man eine verpolte Spannung am Eingang anlegen, sperrt die Bulk-Diode und der MOSFET gelangt nicht in den leitenden Zustand. Bei Spannungen, die kleiner als UGSmax sind, kann die Z-Diode vernachlässigt werden. In unserem Fall handelt es sich lediglich um eine anliegende Spannung von 12V, weshalb die Diode vernachlässigt werden kann und ein beinahe verlustfreier Betrieb möglich ist.

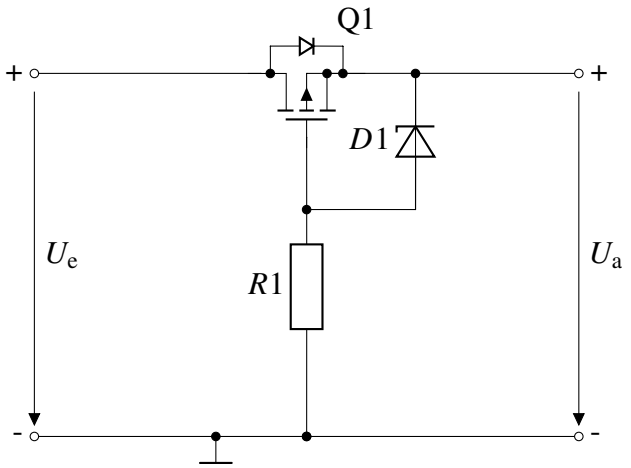


Abbildung 2.6: Verpolungsschutz mit P-Kanal MOSFET für kleine Spannungen

Aufgrund der Vorteile des fast verlustlosen und wartungsarmen Betriebs entschieden wir uns für den Gebrauch der 3.Variante.

2.3.2 Soft-Latching-Circuit

Eine Eingangsspannung von 12V, welche vom Netzteil zur Verfügung gestellt wird, soll durch die Betätigung eines Tasters durchgeschaltet werden und die Elektronik versorgen. Diese Versorgung soll auch nach der Betätigung zur Verfügung stehen. Die Versorgung soll durch ein Signal des RPI gestoppt werden, wodurch ein sicheres Herunterfahren des RPI bereitgestellt werden soll.

2.3.2.1 Realisierte Variante mit einzelnem Taster zum Ein- und Ausschalten

Durch jene Betätigung von T_1 fließt Strom durch R_2 und aktiviert somit den N-Kanal MOSFET Q_2. Dieser Feldeffekttransistor schaltet dadurch die am Drain anliegende Leitung auf Masse. Das deshalb auf Ground gezogene Gate des P-Kanal MOSFET Q_1 lässt diesen durchschalten. Über den Ausgang $12V_{OUT}$ werden somit sämtliche 12V Sensoren, Aktoren und Elektronikkomponenten versorgt. Das dadurch ebenfalls über R_3 versorgte Gate von Q_2 lässt die davon am Drain liegende Leitung auf Masse schalten, welches die Versorgung der 12V Komponenten auch nach der Betätigung des S_1 bereitstellt. Um diese Selbsthaltung wieder zu lösen, benötigt es ein digitales „High“ von 3.3V am Eingang $U_{UNLATCH}$, ausgehend von einem Pin des Mikrocontrollers. Die Schottkydiode D_1 sorgt für dafür, dass kein Strom zurück in den µC fließen kann und dafür, dass sich der aufgeladene Elektrolytkondensator nur über den Widerstand R_8 entlädt. Das dadurch versorgte Gate von Q_3 schaltet die am Drain anliegende Leitung auf Masse. Der dadurch deaktivierte Q_2 löst die Selbsthaltung und die Schaltung befindet sich wieder in ihrem Grundzustand. Ein genaue Dimensionierung von C_1 und R_8 sorgen für ein ausreichend langes „High“ der Leitung, sodass ein zuverlässiges Abschalten der 12V Ausgangsspannung bereitgestellt werden kann.

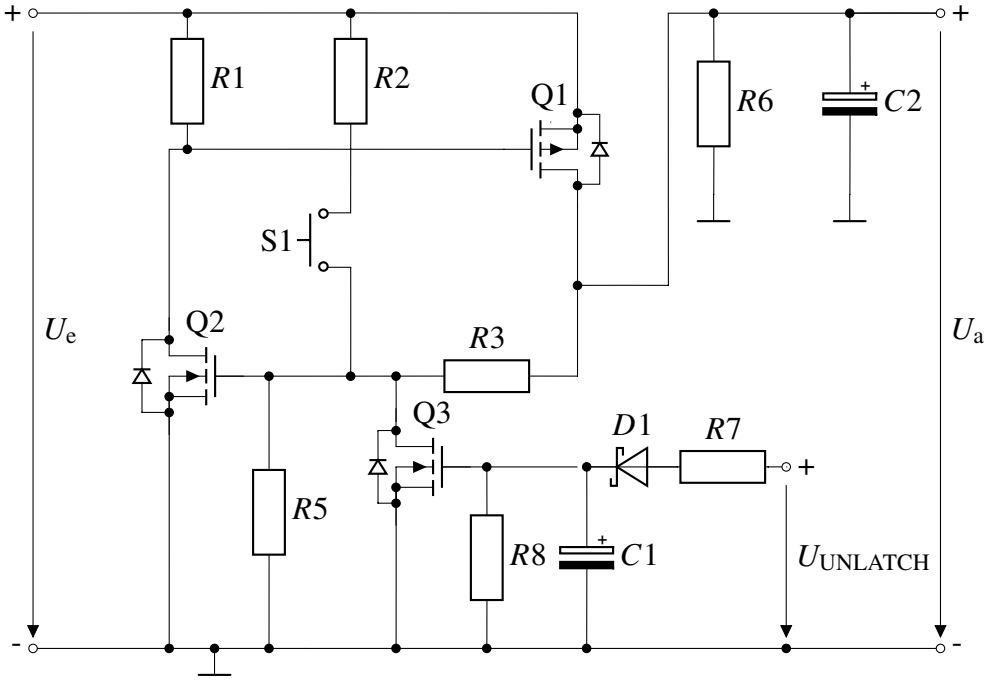


Abbildung 2.7: 12V Soft-Latching Circuit

2.3.2.2 Andere Variante

Die folgenden Variante kam ebenfalls im Laufe der Diplomarbeit auf.
Sobald S_1 betätigt wird fließt Strom durch den Widerstand R_2, durch der Bipolartransistor Q_2 die am Collector anliegende Leitung auf Masse schaltet. Das dadurch auf Masse gezogene Gate des P-Kanal MOSFET lässt diesen durchschalten und eine Ausgangsspannung von 12V ist verfügbar. Der durch den Widerstand R_3 fließende Strom sorgt für ein Durchschalten des npn-Transistors, auch wenn der Taster S1 nicht mehr gedrückt ist.
Ein Betätigen des Tasters S_2 hat die Folge, dass die Basis des Transistors Q_2 auf Masse gezogen wird. Dies hat die Folge eines positiven Pegels am Gate des MOSFETs, welches diesen sperren lässt. Somit kommt am Ausgang keine Spannung vorgefunden werden. Um ein floatendes Gate von Q_1 zu vermeiden ist der Widerstand R_1 vom Drain auf das Gate geschaltet.

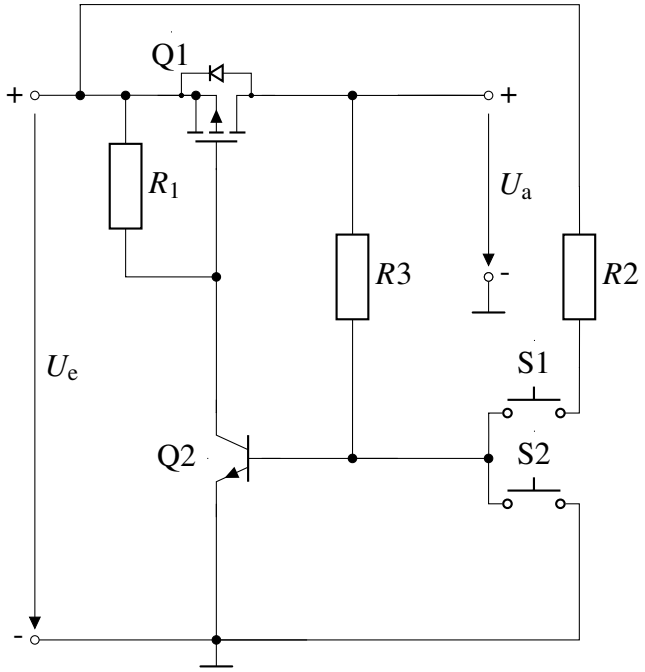


Abbildung 2.8: Realisierung einer Selbsthaltung mit zwei Tastern

Diese Variante wurde letztendlich verworfen, da wir uns dazu entschieden haben, nur einen Schalter zu verwenden, wessen Funktion das Einschalten der 12V sein soll. Außerdem soll ein sicheres Herunterfahren des Raspberry Pi gewährleistet sein, zu welchem diese Schaltung nicht in der Lage ist.

2.3.3 DC/DC Wandler

Als Buck-Converter wählten wir den LM2576-5.0 aus. Mit diesem ist es möglich eine Eingangsspannung von 7V bis 40V zu einer Spannung von 5V mit einem Strom von maximal 3A umzuwandeln. Weitere Kriterien, die unsere Wahl befürworteten, waren die niedrige Anzahl von nur vier zusätzlichen Bauteilen, der hohe maximale Ausgangsstrom sowie die hohe Effizients.

2.3.3.1 Beschaltung

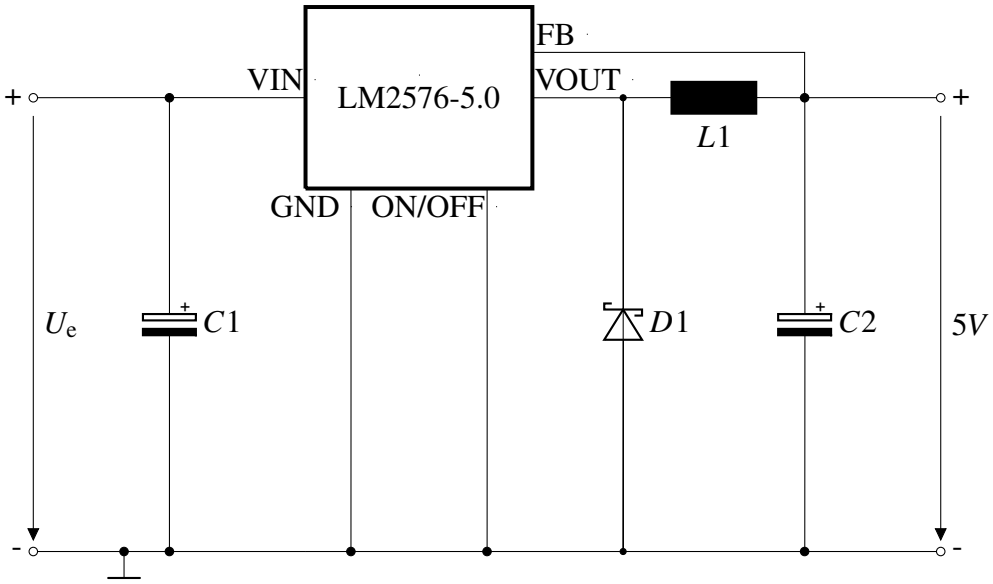


Abbildung 2.9: Beschaltung des LM2576-5.0

2.3.3.2 Kühlkörperberechnung

Um die Notwendigkeit eines Kühlkörpers zu berechnen, werden Formeln der Datenblätter zur Berechnung angewandt.
Um die Verlustleistung des LM2576 zu berechnen, wird von folgender Formel Gebrauch gemacht.

$$P_D = V_{IN} \cdot I_Q + \frac{V_{OUT}}{V_{IN}} \cdot I_{LOAD} \cdot V_{SAT} \tag{2.1}$$

wobei

- P_D der auszurechnenden Verlustleistung,
- V_{IN} der Eingangsspannung,
- I_Q dem Ruhestrom,
- V_{OUT} der regulierten Ausgangsspannung,
- I_{LOAD} dem Laststrom und V_{SAT} der Sättigungsspannung entspricht.

Setzen wir nun die bei unserer Anwendung maximal anfallenden Werte ein erhalten wir folgendes Ergebnis:

$$P_D = 12V \cdot 0.013A + \frac{12V}{5V} \cdot 3A \cdot 1.4V$$
$$= 1.906W$$

Der Anstieg der Sperrschichtübergangstemperatur lässt sich somit folgend berechnen:

$$\Delta T_j = P_D \cdot \Theta_{ja} \tag{2.2}$$
$$= 1.906W \cdot 42.6\frac{^\circ C}{W}$$
$$= 81,1946^\circ C$$

Zu diesem Anstieg ist noch die gewöhnliche Umgebungstemperatur hinzuzurechnen.

$$T_j = \Delta T_j + T_A \tag{2.3}$$
$$= 81,1946^\circ C + 25^\circ C$$
$$= 106,1956^\circ C$$

Durch dieses Ergebnis lässt es sich, trotz eines Sicherheitsabschlages der maximalen Sperrschichttemperatur (T_{jmax} = 125°C-15°C), auf einen zulässigen Betrieb schließen.

Obgleich dieser Wert recht hoch erscheint, werden wir auf einen Kühlkörper verzichten, da wir dieses maximale Potential des Gleichspannungswandlers nicht voll ausnutzen werden. (Die Verlustleistung erreicht bei einem Laststrom von 1.5A einen Wert von 0.8435W. Der daraus zu schließende Wert der Sperrschichttemperatur beträgt somit gerade einmal 60,9331°C.)

2.3.4 Raspberry Pi 3B+

Als Basis für unsere Human Machine Interface wählten wir einen Raspberry Pi 3B+ aus. Die Begründung dieser Wahl befindet sich im Infortmatik-Teil dieser Diplomarbeit. Dieser hat die Aufgabe den Mikroprozessor ATmega324P mir 3.3V zu versorgen sowie über das Touchpanel eingelesene Information mit dem Mikroprozessor auszutauschen.

2.3.4.1 Watchdog

Als Funktion der Ausfallerkennung implementierten wir eine Watchdog-Verbindung zwischen dem Raspberry Pi und dem Mikrocontroller. So kann bei einem Ausfall dennoch eine sicheres Herunterfahren beziehungsweise eine geeignete Fehlerbehebung gewähleistet werden.

2.3.5 Beschaltung des ATmega324P

Um den Mikrocontroller zu versorgen, wurde an allen VCC-Pins und am AVCC-Pin eine Verbindung zum 3.3V Ausgang des Raspberry Pi vorgesehen. Um die Versorgungsspannung von 3.3V zu glätten, wurde an allen 4 Versorgungspins je ein Kondensator mit 100nF vorgesehen. An jeglichen GND Anschlüssen des Mikrocontrollers wurde eine Leitung gegen Masse implementiert.
Um die interne Referenzspannung zu glätten, wurde, wie im Datenblatt empfohlen, ein 100nF Kondensator am Pin AREF implementiert.
Laut dem zugehörigen Datenblatt ist bei einer Versorgungspannung von 3.3V ein Quarz mit einer Taktfrequenz von 0 bis 10MHz zu arbeiten. Aufgrund unserer Entscheidung, den Mikrocontroller mit einem 10MHz Quarz zu betreiben, waren zwei Kondensatoren mit einem Wert von 12 bis 22pF zu wählen. Diese haben den Nutzen, den Quarzoszillator gut anschwingen und stabil schwingen zu lassen.
Der RESET-Pin setzt bei einem "LOWPegel-Signal den µC zurück. Um dies im Grundzustand zu unterbinden, wird ein Pull-Up Widerstand von 10kOhm implementiert. Parallel zu diesem wurde eine externe Diode zum Schutz vor Überspannungen vorgesehen. Um den RESET-Pin gegen Masse schalten zu können, wurde ein Taster implementiert. Über den Data-Terminal-Ready Pin des USB-UART-Converters und dessen Jumper-Pad oder die SPI-Schnittstelle ist es jedoch ebenfalls möglich, den ATmega324P zu resetten. Parallel zum Taster wurde ein RC-Glied vorgesehn, welches einerseits als Entprellung der Tasterbetätigung dient andererseits auch als Prevention gegen Spikes und für ein längeres, definiertes Bestehen des µC im Reset-Zustand.

2.3.6 DRV8825-Schrittmotortreiber

Dieses Modul ist dafür ausgelegt, bipolare Schrittmotoren anzusteuern. Der DRV8825 besitzt zwei H-Brücken-Treiber und einen Microstepping Indexer. Um die Motorwindungen anzusteuern, sind die Ausgangsblöcke des Treibers als volle N-Kanal Leistung-MOSFET H-Brücken realisiert. Dem DRV8825 ist es möglich einen Ausgangsstrom von bis zu 2.5A bei geeigneter Kühlung zu treiben.

2.3.6.1 Datenblattwerte

- Minimale Betriebsspannung 8.2V
- Maximale Betriebsspannung 47V
- Maximaler Strom per Phase 1.5A (2.5A*)
- Minimale Steuerspannung 2.2V
- Maximale Steuerspannung 5.25V
- Dimensionen 15.5mm x 20.5mm

* nur bei ausreichender Kühlung möglich

2.3.6.2 Beschaltung

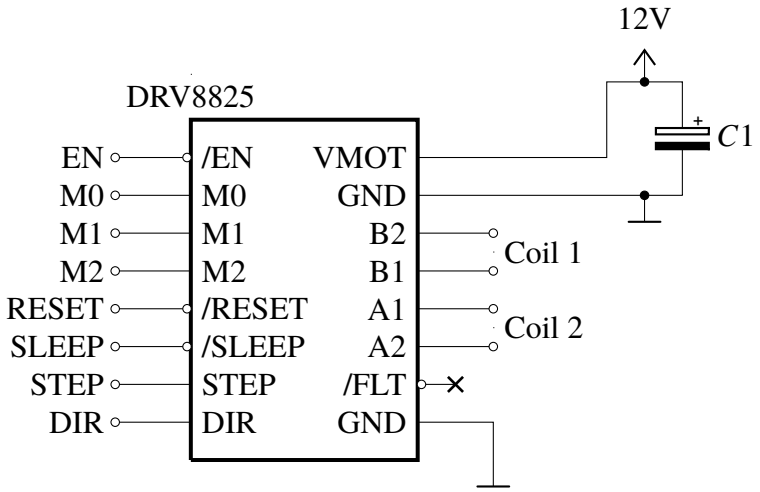


Abbildung 2.10: Gewöhnliche Beschaltung eines DRV8825

2.3.6.3 Beschaltungs- und Anschlussfunktionen

Der DRV8825 besitzt 16 Anschlüsse. Drei dieser Anschlüsse fungieren zur Spannungsversorgung: VMOT und die beiden GND Anschlüsse.

Über den Anschluss VMOT werden in unserem Fall 12V eingespeist. Bei der Beschaltung der Spannungsversorgung des DRV8825 wird vom Hersteller der Gebrauch eines Elektrolytkondensators empfohlen.

„Diese Trägerplatine benutzt niedrige-ESR Keramikkondensatoren, welche sie anfällig für zerstörerische LC-Spannungsspitzen machen. Insbesondere wenn längere Stromkabel verwendet werden. Unter den falschen Bedingungen können diese Spannungsspitzen die maximale Spannung von 45 V für den DRV8825-Treiber überschreiten und der Platine dauerhafte Schäden beisetzen. Selbst, wenn die Versorgungsspannung des Motors nur 12 V, können diese Art von Spannungsspitzen auftreten. Eine Möglichkeit diese Gefahr zu negieren, ist einen, mindestens 47µF großen Elektrolytkondensator zwischen dem Pin VMOT und Ground in der Nähe des Treibers zu schalten.“

An die Anschlüsse B2, B1, A1 und A2 werden die Windungen des Schrittmotors mit dem Treiber verbunden. Hierbei ist auf die richtige Zusammengehörigkeit zu achten: Die Anschlüsse B1 und B2 sind mit der einen Windung des Motors zu verbinden, A1 und A2 mit der anderen.

Der invertierte FLT (Fault) Anschluss kann als Statusüberwachung genutzt werden. Er liefert ein LOW-Pegel Signal, falls der Treiber eine zu hohe Temperatur oder einen zu hohen Strom detektiert.

Der DRV8825 Treiber verfügt 8 Logikanschlüsse, welche auf der linken Seite des Treibers in Abbildung X zu finden sind:

Der invertierte EN Anschluss sorgt bei einem eingehenden HIGH-Signal dafür, dass die H-Brücken deaktiviert und das STEP Eingangssignal nicht berücksichtigt wird. Bei einem eingehenden LOW-Signal ist der Treiber aktiviert. Somit sind die H-Brücken aktiviert und steigende Flanken am STEP Anschluss werden eingelesen und verarbeitet. Durch einen internen Pull-down Widerstand ist dieser Pin auch im nicht beschaltenen Zustand auf LOW.

Über die Anschlüsse M0, M1 und M2 ist es möglich, die gewünschte Mikroschrittauflösung zu wählen. Lässt man diese unbeschalten, sorgen interne Pull-down Widerstände dafür, dass diese auf LOW gezogen werden. Eine Übersicht der Mikroschrittauflösungen befindet sich im Kapitel 2.3.6.4 .

Der invertierte Anschluss RST sorgt bei einem LOW-Signal für einen Reset der internen Logik und deaktiviert die H-Brücken Treiberbausteine.

Ein Anlegen eines HIGH-Signals am invertierten Eingang SLP versetzt den Treiber in einen stromsparenden Schlafzustand.

Um den Treiber in voller Funktion verwendet zu können, müssen die Anschlüsse RST und SLP mit einem HIGH-Signal versorgt werden.

Der STEP Anschluss des DRV8825 liest ein Rechtecksignal ein. Bei jeder steigenden Flanke rückt der Indexer um einen Schritt weiter.

Die Drehrichtung des Motors wird über den DIR Anschluss gesteuert. Ein HIGH-Pegel am Anschluss bewirkt eine Drehrichtung des Motors im Uhrzeigersinn, ein LOW-Pegel sorgt für eine Drehrichtung gegen den Uhrzeigersinn.

2.3.6.4 Schrittauflösung

In der unten zu sehenden Tabelle sind die alle Pegelkombinationen der Modus-Anschlüsse dargestellt, um die gewünschte Mikroschrittauflösung zu erreichen.

2.3.6.5 Stromlimitierung

Um den durch die Motorwindungen fließenden Strom zu limitieren, kann dem Datenblatt des DRV8825 folgende Formel entnommen werden:

$$I_{max} = \frac{V_{ref}}{5 \cdot R_{sense}} \tag{2.4}$$

M2	M1	M0	Schrittauflösung
0	0	0	Vollschritt
0	0	1	½ Schritt
0	1	0	¼ Schritt
0	1	1	8 Mikroschritte pro Schritt
1	0	0	16 Mikroschritte pro Schritt
1	0	1	32 Mikroschritte pro Schritt
1	1	0	32 Mikroschritte pro Schritt
1	1	1	32 Mikroschritte pro Schritt

Tabelle 2.6: 1 signalisiert ein HIGH-Signal, 0 sein LOW-Signal

Der von uns ausgewählte Schrittmotor 17HS19-2004S1 braucht laut dessen Datenblatt einen Strom von 2A . Durch einige Tests und Messungen fanden wir jedoch heraus, dass 1.6A bei gewöhnlichem Betrieb mehr als genug sind. Um nun die Stromlimitierung auf 1.6A einzustellen, muss die Referenzspannung adaptiert werden.

$$V_{ref} = I_{max} \cdot 5 \cdot R_{sense}$$

Der R_{sense} kann dem Datenblatt entnommen werden. Beim DRV8825 beträgt der Wert 0.1 Ohm.

$$\begin{aligned} V_{ref} &= 1.6A \cdot 5 \cdot 0.1\Omega \\ &= 0.8V \end{aligned}$$

Die Berechnung ergibt, dass für einen Strom von 1.6A, eine Referenzspannung von 0.8V eingestellt werden muss. Diese Spannung kann mithilfe eines Potentiometers, welches sich auf dem Treibermodul befindet, eingestellt werden.

2.3.6.6 Kühlkörperberechnung

Um die Notwendigkeit eines Kühlkörpers zu berechnen, werden folgende Formeln benötigt:

$$P_D = \frac{(T_{jmax} - T_{ha} - Sicherheit)}{R_{thja}} \tag{2.5}$$

$$P_V = R_{DSon} \cdot I_{max}^2 \tag{2.6}$$

Sollte die Verlustleistung P_V größer als P_D ist ein Kühlkörper von Nöten.

Datenblattwerte

- T_{jmax} = 150°C
- T_{ha} = 25°C
- R_{thjc} = 15.9°C/W

- R_{thja} = 31.6°C/W
- R_{DSon} = 0,32 Ohm

Berechnung

$$\begin{aligned} P_V &= R_{DSon} \cdot I_{max}^2 \\ &= 0,32\Omega \cdot 1.6A^2 \\ &= 0.8192W \end{aligned}$$

$$\begin{aligned} P_D &= \frac{(T_{jmax} - T_{ha} - Sicherheit)}{R_{thja}} \\ &= \frac{150°C - 25°C - 25°C}{31.6\frac{°C}{W}} \\ &= 3.1646W \end{aligned}$$

Durch die Berechnung lässt sich beweisen, dass kein Kühlkörper nötig ist. Jedoch wird vom Hersteller ausdrücklich empfohlen einen Kühlkörper zu verwenden, sobald ein Strom größer als 1A getrieben wird, wodurch wir schlussendlich einen Kühlkörper auf dem Treiber platzierten.
Für die Kühlkörperberechnung wird von folgender Formel Gebrauch gemacht:

$$\begin{aligned} R_{thK} &= \frac{T_{jmax} - T_{ha} - Sicherheit}{P_V} - (R_{thiso} + R_{thjc}) \\ &= \frac{150°C - 25°C - 25°C}{0,8192W} - (0.1\frac{°C}{W} + 15.9\frac{°C}{W}) \\ &= 106.07\frac{°C}{W} \end{aligned} \tag{2.7}$$

Die Berechnung zeigt, dass die Verwendung eines Kühlkörpers nicht unvorteilhaft wäre, weshalb wir den mitgelieferten Kühlkörper auch verwenden. Zu diesem sind keinerlei Werte bekannt, jedoch konnten wir durch einige Tests seine geeignete Funktionalität unter Beweis stellen.

2.3.7 Erfassung der Motorposition

Zu Beginn wird der Schrittmotor in eine Richtung angesteuert, bis er seinen Referenzpunkt erreicht, welcher als Mikroschalter implementiert ist. Der vom Mikrocontroller auf "HIGH"gesetzte Ausgang, wird bei einer Betätigung des Mikroschalters gegen Masse geschalten. Der Zustand dieses Ausgangs wird über das zugehörige PIN-Register vom Mikrocontroller eingelesen. So können ausgehend vom Referenzpunkt Schrittzahlen zu gewünschten Positionen aufgetragen werden.

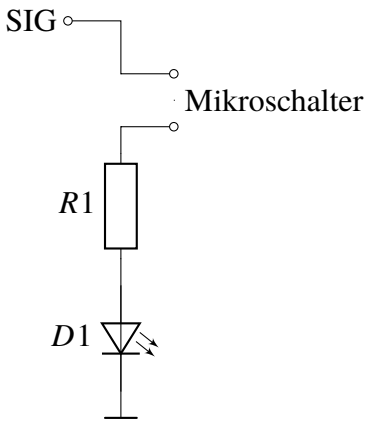


Abbildung 2.11: Erfassung der Schrittmotorposition

2.3.8 Kapazitive Sensoren

Die von uns ausgewählten Sensoren LJC18A3-BZ/BX besitzen jeweils drei Anschlüsse: einen blauen, welcher mit Ground zu verbinden ist, einen braunen, der mit einer Versorgungsspannung von 6V bis 36V zu verbinden ist sowie einem schwarzen über den das Signal vermittelt wird.

Datenblattwerte:

- Schaltabstand 1 bis 10mm, einstellbar durch Stellschraube
- Betriebsspannung 6V-36V
- Maximale Last 300mA
- Ausgang realisiert durch einen NPN-Schließer

In unserem Fall werden alle kapazitiven Sensoren mit 12V versorgt. Der Signalausgang des Sensors liefert im Grundzustand ein HIGH-Signal. Sobald ein Objekt in das Messumfeld des Sensors gelangt, liefert dieser ein Signal von 0.62V, welches als LOW-Signal gilt. Um den HIGH-Pegel für den Mikrocontroller lesbar zu gestalten, ist ein Spannungsteiler am Signalausgang des Sensors zu dimensionieren. Hierzu verwenden wir folgende Formel:

$$U_a = U_e \cdot \frac{R_1}{R_1 + R_2} \tag{2.8}$$

Durch verschiedene Tests musste ich die Erkenntnis machen, dass sich der Spannungspegel des Sensorausgangssignals je nach Spannungsteiler ändert. Durch sukzessive Annäherung an die gewünschten 3.3V Ausgangsspannung wurden folgende Werte als günstige Lösung gewählt:

$$R_1 = 5.1k\Omega, R_2 = 2k\Omega$$

Durch diese Werte ergibt sich bei einer HIGH Signalausgangsspannung von 6.92V eine Spannung am Ausgang des Spannungsteilers von 3.31V . Bei einem LOW-Pegel ist am Ausgang des Spannungsteilers eine Spannung von 0.29V zu messen.

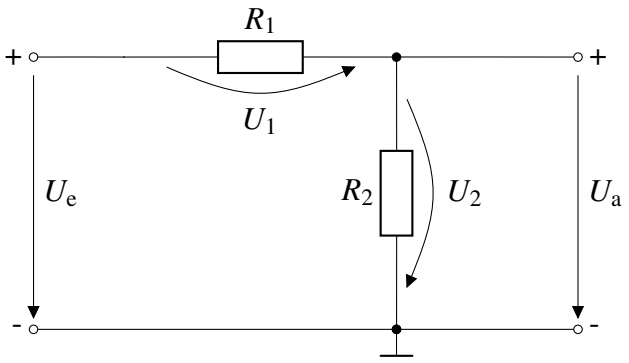


Abbildung 2.12: Aufbau einer Spannungsteilers

2.3.9 Ansteuerung der Hubmagneten

Als Ansteuerung der beiden Hubmagneten wird jeweils die unten zu sehende Schaltung angewandt. Auf einen Bipolartransistor wurde nicht zurückgegriffen, da dieser nicht in der Lage dazu wäre derartige Ströme ohne Überhitzung zu schalten.

2.3.9.1 Funktion

Sobald die Gate-Source-Schwellenspannung des MOSFETs mithilfe eines 3.3V Signals ausgehend vom Mikrocontroller überschritten wird, schaltet der N-Kanal MOSFET durch. Dieser ermöglich das Fließen des Drainstromes durch den Hubmagneten zum Massepotential. Um den fließenden Gate-Source-Strom zu verringern wurde der Widerstand R1 implementiert. Der Widerstand R2 hingegen hat den Nutzen, parasitäre Kapazitäten des MOSFETs zu entladen Da es sich bei den Hubmagneten um eine induktive Last handelt, ist es erforderlich parallel eine "Fly-Wheel-Diodeßu schalten. Diese hat die Aufgabe, den MOSFET vor einer selbsterzeugenten, rückfließenden Spannung des Hubmagneten zu schützen.

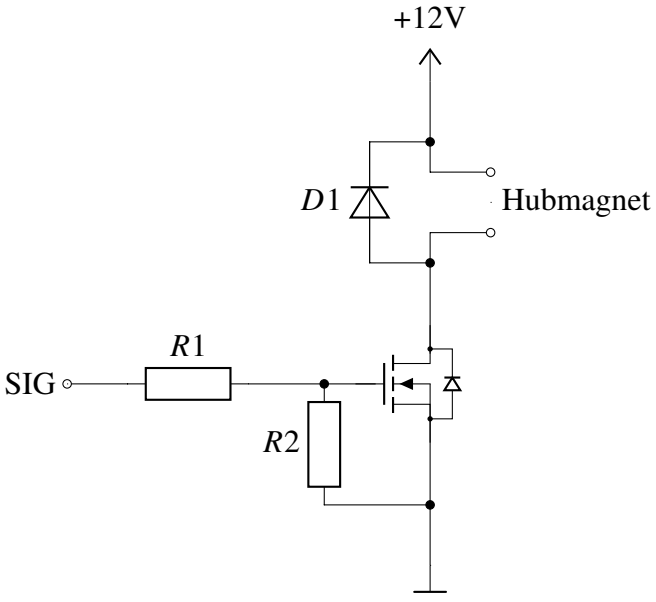


Abbildung 2.13: Ansteuerung der Hubmagneten

2.3.10 Mini-USB Schnittstelle

Um Debugging effektiv betreiben zu könnnen, ist eine Verbindung von Mini-USB zu UART implementiert worden. So können Vorgänge leicht über einen Computer via Kabel überwacht werden oder etwa ein neues Programme herübergespielt werden.
Als USB-zu-UART-Converter wählten wir den CH340G aus, für dessen Wahl seine geringe, einfache Beschaltung und Implementierung sprach. Dieser Treiberbaustein ist wie folgend zu beschalten:

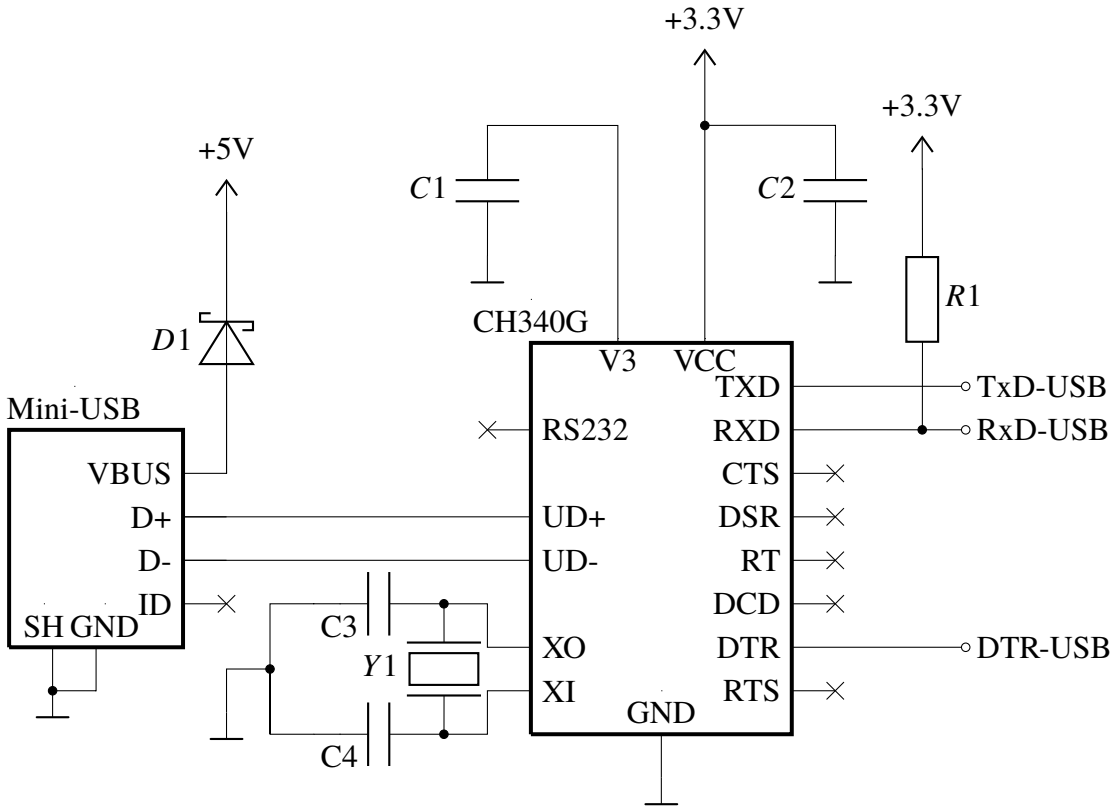


Abbildung 2.14: Beschaltung des Mini USB Treibers

Neben der vorgeschriebenen Beschaltung wurde zusätzlich die Schottkydiode D1 und ein Pull-Up Widerstand am RXD-Pin implementiert. Die Schottkydiode hat die Aufgabe eine Rückspeisung von 5V zum eingesteckten Gerät zu verhindern. Der Pull-Up Widerstand fungiert als Prevention einer floatenden Leitung.

2.3.11 Spannungspegelüberwachung

Grundsätzlich ist zu sagen, dass an den Spannungspegeln von 12V, 5V und 3.3V sowie an anderen signifikanten Stellen der Platine Messpunkte zur erleichterten Fehlersuche und Wartung realisiert wurden.
Zusätzlich wurden, um jegliche Spannungspegel visuell darzustellen, bei manchen Spannungspegeln eine grüne Leuchtdioden als visuelle, schnelle Funktionsüberwachung integriert.

Am 12V Spannungspegel realiserten wir eine Zenerdiode mit einer Zenerspannung von 8.2V . Diese ist in Serie mit einem Widerstand und einer Diode geschaltet. So vermag es der LED erst ab einer Spannung von cirka 10.4V zu leuchten.
Ein Überwachen des 5V Pegels wurde folgend realisiert: Zwei Dioden würden seriell mit einem Widerstand und einer Leuchtdiode geschalten. Die benötigte Spannung, um ein Leuchten der LED zu geährleisten, steigt somit auf etwa 4V .
An dem 3.3V Spannungspegel ist ein Widerstand und eine Leuchtiode in Serie geschaltet, welche ab einer Spannung von 2.2V leuchtet.

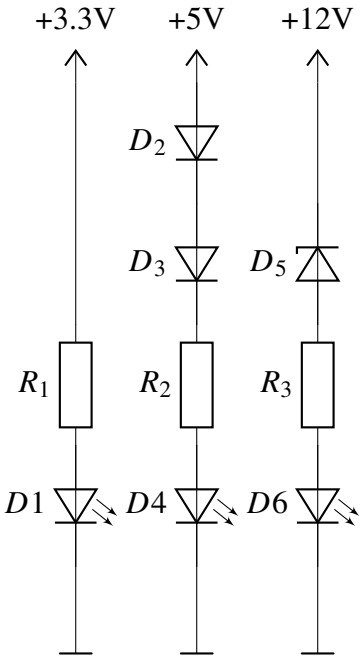


Abbildung 2.15: Realisierung der Leuchtdioden

2.3.12 Netzteil

Als Netzteil wählten wir ein Schaltnetzteil mit Kaltgerätestecker aus. Dieses hat die Aufgabe ein Wechselspannung von 230V in eine Gleichspannung von 12V umzuwandeln. Außerdem erfüllt es unsere Anforderung, einen Strom von 5A bereitstellen zu können.

2.4 Auswahl der Bauteile

2.5 Leiterplattendesign

Sogleich die Auswahl der Peripherie beendet war, war es an der Zeit ein Design für die geplante Platine in KiCad zu entwerfen. Als Anforderungen stellten wir uns ein möglichst kompaktes, strukturiertes sowie übersichtliches Design zu erstellen. Im Laufe der Diplomarbeit entstanden unzählig viele, verschiedene Layouts, welche die Gründe von ständig neu aufkommenden Ideen, Ausmerzung von Fehlerquellen und Simplifizierungen hatten. Diese Faktoren machten das Designen eines optimalen Layouts zu einer langwierigen Aufgabe. Eine falsche Dimensionierung von Bauteilen und Baugruppen, Denkfehler, die das Fertigen erschweren würden, und ungünstige Leiterbahnanordnungen waren hierbei die Hauptfaktoren.

2.5.1 Leiterplattenfertigung

2.5.1.1 Prototyping

Bevor wir den Auftrag einer Leiterplattenfertigung an ein professionielles Unternehmen aufgaben, stellten wir uns die Fertigung eines funktionierenden Prototypen als Aufgabe. Dieser sollte in der Werkstätte der HBTLA Kaindorf angefertigt werden.

Als Fertigungsschritte sind folgende zu nennen:

- Zuallererst ist das Layout der Kupferforderseite und der Kupferückseite jeweils auf eine transparentes Papier mit den bestmöglichen Einstellungen zu drucken. Für einen erfolgreichen Druck ist ein Spiegeln der Vorderseite der Platine ein Muss. Folgend ist ein Spray zum Verdichten des Toners anzuwenden. Aus den zwei Stücken Transparentpapier ist eine Tasche anzufertigen, sodass jegliche Bohrungen genau gegenüberliegen.

- Zunächst ist die Leiterplatte mithilfe der angefertigten Tasche eine Minute lang in der Belichtungsmaschine unter Vakuuum zu belichten.

- Die beleuchtete Leiterplatte ist anschließend eine Minute lang in ein Natriumhydroxid-Bad zu begeben, um sie zu entwickeln.

- Folgend ist sie mit heißem Wasser abzuspülen und in ein Natriumpersulfat-Bad zu geben. Die Zeitdauer des Verweilens variiert je nach Belastung des Ätzbades. In unserem Fall ätzten wir die Leiterplatte 17 Minuten.

- Anschließend ist die Leiterplatte mit heißem Wasser abzuspülen und mit Azeton zu reinigen.

- Als nächsten Schritt ist sie für einige Minuten in ein Zinnbad zu legen.

- Nun kann die Leiterplatte ein letztes Mal mit heißem Wasser gereinigt und mit Fluxclean gereinigt werden.
- Letztendlich kann der exakte Zuschnitt und das Bohren durchgeführt werden.

- Zuletzt kann das Anfertigen von Durchkontaktierungen sowie das Bestücken der Leiterplatte erfolgen.

2.5.1.2 Professionelle Fertigung

2.6 Resümee und Aussichten für die Zukunft

3 Informatik

3.1 Allgemeines

Im folgenden Projektteil werden die Kernbestandteile sowie der nähere Aufbau des informatischen Bereichs dieser Arbeit gezeigt. Weiters werden wesentliche Elemente der Inbetriebnahme der Maschine dokumentiert und zusammengefasst. Die Aufteilung dieses XX seitigen Kaptiels überstreckt sich über alle signifikanten programmatischen Teile der grafischen Oberfläche, bezeichnet als **Frontend**, bis hin zu den im Hintergrund arbeitenden Funktionen, welche als **Backend** zusammengefasst werden. Auch wird gezeigt, wie einige Teile der **Hardwarenahen-Programmierung** funktionieren und wie diese mit der Steuerelektronik zusammenarbeiten. Folglich werden am Schluss der Verlauf der **Testing-Phase** sowie die Zusammensetzung des **Teilaufbaus** erläutert.

3.2 Zeitplan

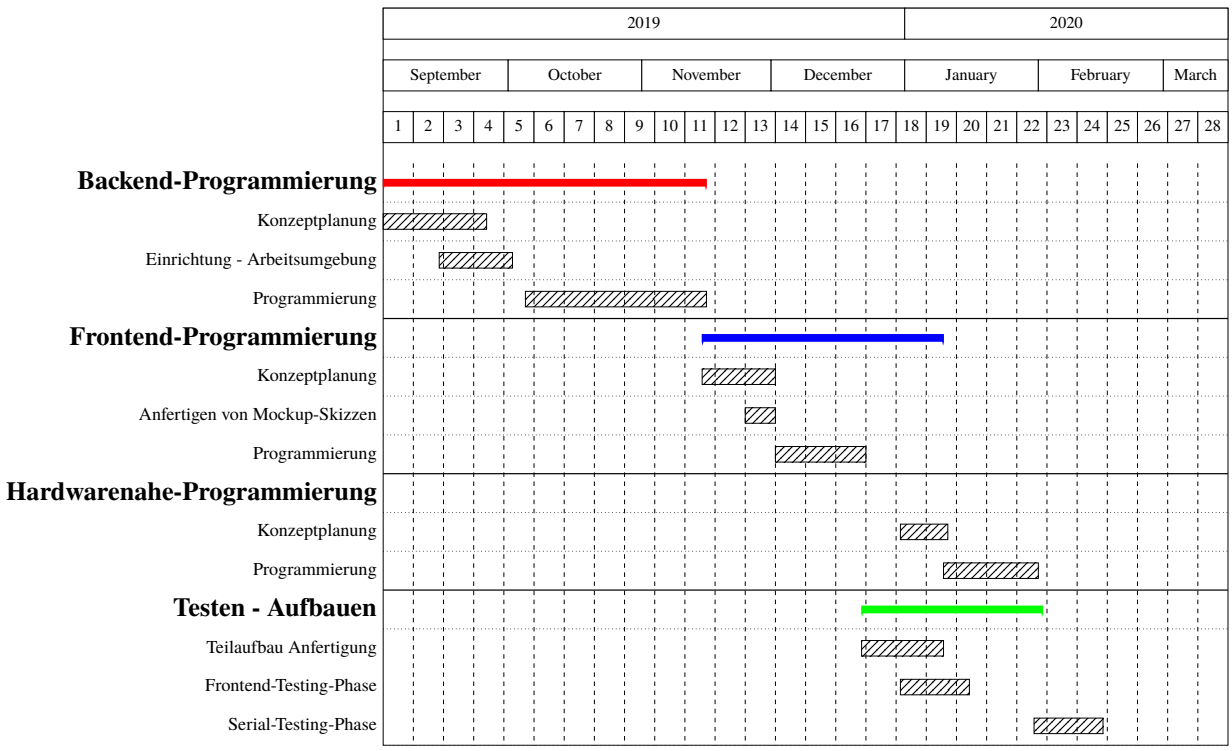


Abbildung 3.1: Zeitplanung - Informatik

Das hier gezeigte Bild illustriert, wie der Projektzeitraum aufgeteilt wird. Um das Zeitfenster der Projekts nicht zu verzögern werden folgende Meilensteine in den Zeitplan implementiert:

1. xx.xx.xxxx - Aufgabe XX
2. xx.xx.xxxx - Aufgabe XX
3. xx.xx.xxxx - Aufgabe XX

3.3 Anforderungen und Ziele

3.3.1 Fronted-Programmierung

Im Allgemeinen besteht das Ziel darin, eine benutzerfreundliche grafische Bedienoberfläche zu realisieren, welche schlussendlich im Betrieb der Maschine auf einem 7"Display angezeigt werden soll. Auf diesem Interface soll es möglich sein, die Steuerung der Maschine zu übernehmen. Die Möglichkeiten des Benutzers, die Maschine zu bedienen sollen folgende Kernpunkte beinhalten:

1. Ausgabe von einzelnen Spielkarten
2. Konfigurierung von Spielmodi, welche Einstellungen zum Spiel beinhalten
3. Das Zählen von Punkten und dessen Visualisierung am Display
4. Ausschalten der Maschine
5. Übersicht aller vergangenen Spiele

Die wesentliche Anforderung, die Oberfläche einfach und schlicht zu halten, soll im Projekt berücksichtigt werden. Grundsätzlich steht die sogenannte **User Experience**, welche alle Aspekte der Eindrücke eines Nutzers bei der Interaktion mit einem Produkt beschreibt, sowie die **Funktionalität** bei der Programmierung im Vordergrund. Um dies zu gewährleisten sollte eine Voruntersuchung vollzogen werden.

3.3.2 Backend-Programmierung

Wie in der Einleitung bereits erwähnt besteht das Backend aus Aufgaben welche im Hintergrund abgearbeitet werden. Diese Aufgaben umfassen die Kommunikation des Raspberry PI mit der Ansteuerplatine sowie dem Simulator.
Auch wird im Hintergrund eine sogenannte LOG-Datei, welche wichtige Aktionen protokolliert, automatisch vom Programm generiert. Diese Datei kann z.B. beim Debugging-Vorgang notwendig sein bzw. dem Entwickler dabei unterstützen, was eine enorme Zeitersparnis mit sich bringt.
Um etwaige Einstellungen der Seriellen Schnittstelle sowie die Definition der Pfade für verschiedene Dateien außerhalb des Programmes vorzunehmen, ist es auch nötig, eine Konfigurationsdatei zu

erstellen, welche nach jedem Start vom Programm eingelesen wird. Mit dieser sogenannten Config-Datei wird auch die Möglichkeit geschaffen, weitere Informationen der Anwendung auch nach einem Neustart der Maschine zu speichern.
Diese hiermit geschaffene Datenpersistenz kommt dem Programm auch bei der zukünftigen Speicherung der Spielmodi oder etwaigen Statistik-Dateien zugute. Bei einer Realisierung einer Statistik-Datei sollte der Name des ausgewählten Spielmodis, dessen Hintergrundinformationen, Spielernamen und Punkteanzahl Kernbestandteil sein.
Ein weiteres Ziel ist die Internationalisierung der gesamten Software. Soll eine Software in verschiedenen Ländern eingesetzt werden, ist eine Internationalisierung ein wichtiger Bestandteil der User Experience. In unserem Fall ist es lediglich notwendig, alle angezeigten Texte zu übersetzen, da keine weiteren Informationen vorhanden sind.

3.3.3 Hardwarenahe-Programmierung

Ein weiterer Punkt des Projekts ist die Hardwarenahe-Programmierung. Aufgrund der Tatsache, das die verwendete Ansteuerplatine über einen ATmega 324PA verfügt, ist es notwendig, eine Programmierung dieses Mikrocontrollers durchzuführen. Als Basis dafür dienen ankommenden Befehle über der seriellen Schnittstelle. Ziel ist es also, diese serielle Kommunikation auf dem Mikrocontroller auszuprogrammieren.

3.4 Projektspezifische Voruntersuchung

Damit dem Programmierprozess nichts mehr im Wege steht, ist es notwendig, einige Voruntersuchungen durchzuführen. Folgend werden einige grundlegende Fragen geklärt und näher erläutert.

3.4.1 Auswahl der Arbeitsumgebung

Das Bearbeiten von Code spielt bei der Programmierung eine elementare Rolle. Dadurch muss, bevor mit dem Programmierprozess gestartet wird, eine passende IDE ausgewählt werden, welche alle vom Benutzer gestellten Anforderungen erfüllt. In unserem Fall sind diese im Speziellen:

1. Möglichkeiten der Programmierung sowie dem Debuggen eines Remote-Rechners
2. Nötigsten Features beinhalten
3. Grafisch Ansprechend

Einer der wichtigsten Punkte dieser Aufzählung ist dabei jedoch die Möglichkeit, Remote-Geräte zu debuggen. Debugging selbst hilft dem Entwickler bei der Fehlersuche. Mithilfe der Möglichkeit, das Programm schrittweise auszuführen, ist es auch möglich, die aktelle Wertebelegung von Variablen zu überprüfen. Dadurch erspart dies enorme Zeit bei der Programmierung. In unserem Fall ist das Endgerät jedoch nicht der PC, sondern ein Raspberry PI, welcher mit dem Netzwerk über Ethernet verbunden ist. Es erfordert aber nun die Möglichkeit, das Gerät über das Netzwerk zu debuggen.

3.4.1.1 Apache-Netbeans

Die Netbeans-IDE, bereitgestellt von der Firma Apache, ist eine Open-Source Entwicklungsumgebung, welche selbst in der Programmiersprache Java geschrieben wurde und damit plattformunabhängig ist. Primär wurde sie entwickelt, um Programme in der Programmiersprache Java zu erstellen.
Große Vorteile bringt diese IDE im Bereich der Remote-Programmierung mit sich. Die Möglichkeit, eine Remote-Plattform einzurichten wurde einfach gelöst und das Arbeiten verläuft meist ohne Probleme. Auch das einfache Einbinden von Plugins und Bibliotheken zählt zu den Hauptvorteilen von Netbeans.

3.4.1.2 Intellij IDEA

Eine kostenpflichtige Alternative zu Netbeans ist Intellij. Aufgrund der komplizierten Einrichtung eines Remote-Systems, sowie dessen Handhabung im täglichen Arbeitsprozess wurde deutlich, dass Netbeans im wichtigsten Kriterium besser abschneidet. Jedoch bring Intellij auch einige Vorteile mit sich.

1. Flüssigere Bedienung
2. Wirkt durchdachter
3. Zentralere Verwaltung von Plugins

3.4.1.3 Fazit

Natürlich ist es meist Ansichtssache, für welche IDE man sich entscheidet doch in Anbetracht der Vorteile von Netbeans gegenüber Intellij im Bereich der Remote-Programmierung, wurde auch diese IDE für den zukünftigen Arbeitsprozess gewählt.

3.4.2 Auswahl des GUI-Toolkits

Die Auswahl des für die Anwendung am besten geeigneten GUI-Toolkits, spielt eine wesentliche Rolle in der programmierung von grafischen Anwendungen. Diese Toolkits stellen meist alle Elemente zur Erstellung einer GUI bereit. Diese sind z.B. Knöpfe, Listen und Textfelder mit denen der Benutzer interagieren kann.

3.4.2.1 Swing

Das bereits in der Standart Java Bibliothek verfügbare Java Swing bietet die Möglichkeit komplexe Oberflächen zu erstellen. Der Aufwand zur Einrichtung hält sich in Grenzen denn im Vergleich zu Java-FX muss hier nicht extern in einem eigenen Programm gearbeitet werden, um die GUI zu erstellen. Aufgrund der Verfügbarkeit von Java Swing in der Java Bibliothek bieten alle IDE's die

Möglichkeit zur erstellung von Oberflächen. Das etwas veraltete und lieblose Look and Feel von Java Swing brachte uns zur Entscheidung, den Nachfolger, nämlich Java FX, zu verwenden.

3.4.2.2 JavaFX

Der Nachfolger von Java Swing ist Java FX. Hierbei werden im Vergleich zu Java Swing nicht alle GUI Elemente in einer Datei beschrieben, was die Übersichtlichkeit beeinträchtigt, sondern die Beschreibung geschieht in externen FXML-Dateien. Die Basis dieser FXML-Dateien ist XML, eine Sprache zur Darstellung von Daten in einem von Menschen lesbaren Format. Der Code hinter der GUI befindet sich jedoch nicht in diesen Dateien sondern in eigenen Controller Klassen, welche mit der GUI verknüpft sind. Dies bringt enorme Vorteile mit sich, denn die strikte Trennung zwischen Beschreibenden Elementen der GUI und dem dahinter stehenden Code ist dadurch möglich.
Auch bieter Java FX die Möglichkeit, separate Stylesheets, also Dokumente, in welchen das Aussehen von Elementen beschrieben wird, zu erstellen.

3.4.2.3 Fazit

3.4.3 Projektorganisation

Eine einheitliche Formatierung und Namensgebung ein Kernbestandteil einer gelungenen Projektplanung. Dies führt zu einer leichteren Orientierung in fremden aber auch in eigenen Projekten. Um die Übersichtlichkeit eines Projektes zu verbessern wird nach einer klar definierten Projektstruktur gearbeitet. Anhand der folgenden Grafik kann die Projektstruktur dieses Projektes abgelesen werden:

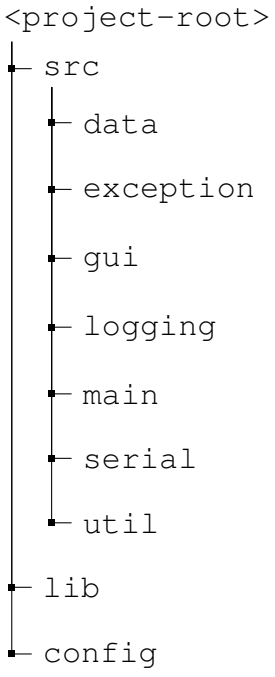


Abbildung 3.2: Projektstruktur

Der Kopf dieser Struktur ist der Ordner mit dem Namen des Projekts. Dieser wird auch `root-directory` genannt, und ist selbst kein Package. Auch die Trennung von Sourcecode und externen Dateien spielt eine wesentliche Rolle. Dadurch erfolgt darunter die Aufteilung in die Ordner `src`, `lib`, `config`.

Sourcecode
Im Ordner `src` befindet sich der gesamte Sourcecode, wobei in diesem auch wieder eine Aufteilung in Packages erfolgt. Diese benennen sich in unserem Fall `data`, `exception`, `gui`, `logging`, `main`, `serial`, und `util`.

Bibliotheken
Oft werden externe Bibliotheken verwendet, welche im Ordner `lib` gesammtelt werden. Dabei erleichtert diese Verzeichnishirachie die übersichtlichkeit aller verwendeten Bibliotheken und dessen Versionen.

Konfigurationsdateien
In unserem Fall ist es auch notwendig, einen Ordner explizit für Konfigurationsdateien vorzusehen. Dort können Textresourcen für Sprachvarianten und Konfigurationsdateien für etwaige Aufgaben abgelegt werden.

3.4.3.1 Fragestellung zur Verwendung eines Build-Tools

Das zuvor genannte Layout der Projekthrachie eignet sich gut für viele Projekte. Jedoch zeigen sich nach und nach einige Schwachstellen an diesem System.
Einerseits ist das aufwendige Erstellen dieser Hirachie ein Punkt, welcher nicht unerwähnt bleiben soll. Auch besteht ein Risiko zur Inkonstistenz bei der Namensgebung von Packages oder Bibliotheken.

Trotz alledem wurde auf die althergebrachte Version gesetzt und das Projekt so erstellt. Grund dafür ist widerum die fehlende Möglichkeit, Remote-Debugging durchzuführen. Natürlich wäre eine Umstellung in einem finalen Release denkbar.

3.5 Backend-Programmierung

3.5.1 Entwurfsmuster Singelton

3.5.1.1 Beschreibung

Singelton dient dazu, die Einzigartikeit eines Objekts sicherzustellen. Das bedeutet eine Instanz des Objektes ist nur genau einmal im Speicher vorhanden.
Auch bietet es einen globalen Zugriffspunkt

3.5.1.2 Struktur

Realisiert wird dies durch einen privaten Kontruktor, denn wo keine Zugriffsberechtigung von außen herrscht, kann auch kein Objekt erstellt werden. Dies bedeuted es liegt eine Kapselung des Konstruktionsprozesses innerhalb der Klasse vor. Die einzige Möglichkeit, eine Instanz zu erstellen, liefert die statische Methode `createInstance()`. Wurde einmal eine Instanz erzeugt, also der private Konstruktor durch `createInstance()` aufgerufen, wird diese erzeugte Instanz in einem statischen Attribut gespeichert.

Natürlich wird auch ein Zugriffspunkt auf dieses Attribut benötigt. Implementiert wird dies durch eine weitere statische Methode namens `getInstance()`. Diese Methode ist der einzige Zugriffspunkt auf die Instanz.

3.5.1.3 Beispiel

```
1
2  public class Config {
3
4  private static final Logger LOG = Logger.getLogger(Config.class.getName())
       ;
5  private static Config instance;
6
7
8  public static Config getInstance () {
9  if (instance == null) {
10 throw new IllegalStateException("Instance not created yet");
11 }
12 return instance;
13 }
14
15
16 public static Config createInstance (String configPath) {
17 if (instance != null) {
18 throw new IllegalStateException("Instance already created");
19 }
```

```
20 else {
21 instance = new Config(configPath);
22 }
23 return instance;
24 }
25
26 //
       ********************************************************************************

27 private final File configFile;
28 private ConfigModel configModel;
29
30
31 private Config (String configPath) {
32 configFile = new File(configPath);
33 readConfig();
34 }
35
36
37 public void save () {
38 writeConfig();
39 }
```

Listing 3.1: Java Codebeispiel

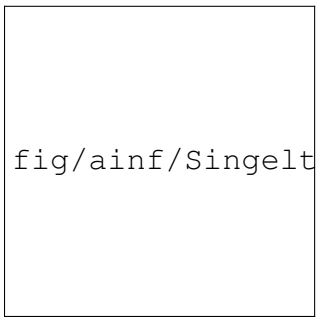


Abbildung 3.3: UML-Diagramm Singelton

3.5.2 Kommunikation - serielle Schnittstelle

3.5.2.1 Konzept

Um erfolgreich Daten zwischen dem Raspberry PI 3B+ und der Platine, welche die Ansteuerung sämtlicher Komponenten übernimmt, zu übertragen, wird ein Kommunikationsprotokoll benötigt. Dieses Protokoll stellt im engsten Sinne eine Vereinbarung dar, wie die Datenübertragung zwischen zwei oder mehreren Parteien abläuft. Anforderungen an dieses Protokoll sollen sein:

1. einfache Integration in die Zielsysteme
2. erweiterbarkeit des Protokolls mit geringem Arbeitsaufwand

3. hohe Sicherheit gegenüber Übertragungsfehler

4. schnelle Fehlererkennung sowie Fehlerbehebung

Neben standartisierten Protokollen wie Modbus, Feldbus oder CAN-Bus gibt es die Möglichkeit selbst ein sogenanntes propretäres Übertragungsprotokoll zu kreieren. Dies ist in unserem Fall nötig, um alle Anforderungen abzudecken. Basierend auf dem Master - Slave Prinzip, wobei der Rasperry PI den Master und die Ansteuerplatine den Slave darstellt, soll ein abgewandeltes Modbus ASCII Protokoll umgesetzt werden.
Zusätzlich soll, um die Anforderung des einfachen Fehlerhandlings zu erfüllen, ein Simulator ausprogrammiert werden, welcher auf Softwarebebene den Platz des Slaves bzw. der Ansteuerplatine einnimmt. Der Startvorgang des Simulators soll mit einer einfachen modifizierung der Konfigurationsdatei vonstatten gehen.

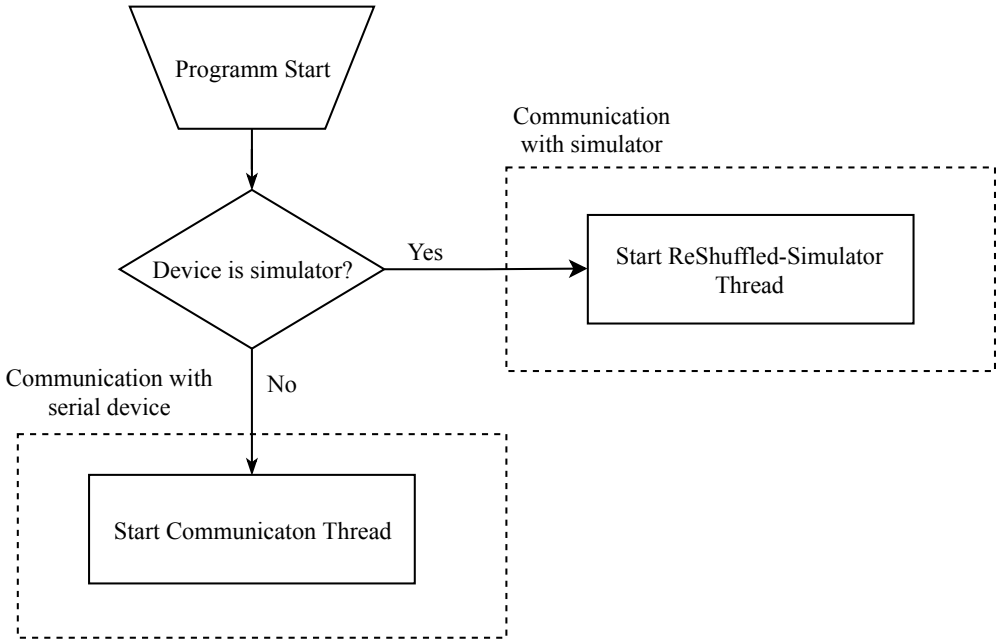


Abbildung 3.4: Schematische Darstellung der Geräteauswahl

3.5.2.2 Übertragungsprotokoll

Doppelpunkt :	**Daten (ASCII)**	**Trennzeichen #**	**CRC32-Prüfsumme**	**Semicolon \n**
8-Bit	16-Bit	8-Bit	32-Bit	8-Bit

Tabelle 3.1: Visualisierung des Datenakets

Das verbindugslose ProtokollRealisierung ist wie in der Konzeptbeschreibung Master-Slave orientiert. Die Datenübertragung erfolgt textuell wobei nur Großbuchstaben verwendet werden dürfen.

Wie aus der oben dargestellten Tabelle zu entnehmen, ist der Aufbau eines Frames klar definiert. Der eindeutige Start des Datenpakets, welcher mit einem Doppelpunkt (:) eingeleitet wird,

sowie das ebenfalls eindeutige Ende, umgesetzt mit einem Line Feed Character (), bringt eindeutige Vorteile mit sich. Im Gegensatz zu anderen Protokollen wie z.B. Modbus RTU, muss hier nicht auf das Ende des Pakets "gewartet"werden. Dies führ oft zu Einbußen im Bereich der Performance und ist für uns nicht zielführend.

Gefolgt von dem Startzeichen folgen nun die Daten. Diese beinhalten eindeutig definierte Zecichenfolgen, welche verwendet werden um verschiedenste Zustände der Ansteuerplatine auszuführen. Diese Um auch hier zu wissen, wo sich das der Nutzdaten befindet, schließt das Trennzeichen () diese ab.

Um nun die Integrität, also die Korrektheit der Daten bei einer Übertragung zu überprüfen, wird im nächsten Schritt eine Prüfsumme verwendet. Ziel dieser ist es, anhand der Nutzdaten einen Wert zu bilden, welcher danach vom Sender im Frame gespeichert bzw übertragen wird. Der Empfänger berechnet nun mit dem selben Verfahren die Prüfsumme aus den empfangenen Daten und vergleicht diese mit der Übertragenen Prüfsumme des Senders. Sind beide Prüfsummen identisch, war die Übertragung erfolgreich und die Daten sind mit großer Wahrscheinlichkeit korrekt. Stimmen diese nicht überein liegt ein Fehler vor. Die wichtigsten Arten von Übertragungsfehlern sind:

1. Einzelbitfehler (1 Bit verändert)

2. Burstfehler (ganze Folge von Bits verändert)

Neben einfachen Verfahren wie z.B. dem Paritätsbit-Verfahren gibt es auch Komplexere. Die zyklische Redundanzprüfung, auch CRC genannt, ist eines davon. Sie ist realtiv einfach zu realisieren und dennoch wirkungsvoll. Wichtig beim CRC-Verfahren ist, dass beide Teilnehmen, also Sender und Empfänger, das selbe Generator-Polynom verwenden. Der Grad des Generatorpolyoms beträgt in unserem Fall 32 (CRC-32).

3.5.2.3 Entwurdsmuster Command

3.5.2.4 Request- und Responsehandling

3.5.3 Kommunikation - Debugging-Simulator

3.5.3.1 Konzept und Gründe zur Erstellung

3.5.3.2 Datenaustausch

3.5.3.3 Verarbeitung ankommender Daten

3.5.4 Logging

3.5.4.1 Grundlagen

3.5.4.2 Integration in das Programm

3.5.5 Konfiguration

3.5.5.1 Grundlagen zu Gson

3.5.5.2 Datenmodelle

3.5.5.3 Verwahrung am Zielsystem

3.5.6 Statistiken

3.5.6.1 Konzept

3.5.6.2 Datenmodelle

3.5.6.3 Verwahrung am Zielsystem

3.5.7 Controllerklassen

3.5.7.1 Grundlagen

3.5.7.2 StartupController

3.5.7.3 MainController

3.5.7.4 HomeController

3.5.7.5 StatsController

3.5.7.6 About- und HelpController

3.6 Frontend-Programmierung

3.6.1 Designkonzept

3.6.2 Entwurfsmuster MVC

3.6.2.1 Grundlagen

Das MVC-Design Pattern, auch Model-View-Controller Design Pattern ist eines der weit verbrei-

Bereiche ab, welche strikt von einander getrennt sind. Mithilfe dieser Strukturierung fällt es dem Programmierer wesentlich leichter, spätere Änderungen bzw. Erweiterungen am Projekt durchzuführen. Auch ist es nun möglich pararell an dem Programm zu arbeiten den Ersteller von GUI und Ersteller des Controllers, also der Berechnungen dahinter, sind nun nicht mehr voneinander Abhängig.

1. **View**
   Datenrepräsentation
   Organisiert alle Kontrollelemente
2. **Controller**
   Stellt die Verbindung zwischen View und Model her
   Verwaltet Benutzerinteraktion und enthält Steuerlogik
3. **Model**
   Datenrepräsentation
   Ist von der Benutzerschnittstelle komplett abgeschirmt

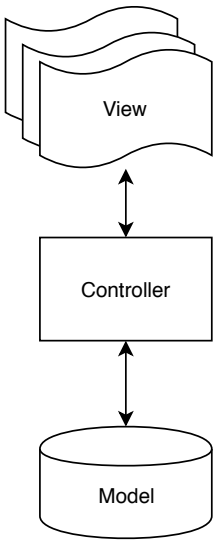


Abbildung 3.5: MVC-Design Pattern

3.6.2.2 Integration anhand eines Programms

Um zu zeigen, wie das MVC-Pattern in der Realität umgesetzt wird, ist im folgenden ein Beispiel dargestellt. Ziel des Programms ist es, einen Wert mithilfe eines Buttons hochzuzählen und auf einem Label im View anzeigen zu lassen. Sobald der Knopf İncrement by 1"betätigt wird, soll der Wert um 1 erhöht werden.

```
1     public class Main extends Application {
2     @Override
3     public void start(Stage stage) throws Exception {
4     Model model = new Model();
5     View view = new View(model, stage);
6     Controller controller = new Controller(model, view);
7     }
8     public static void main(String[] args) {
9     launch(args);
10    }
11    }
```

Listing 3.2: Java Codebeispiel

```
1  public class Model extends Observable{
2  private int counter;
3  public Model() {
4  }
5  public int getCounter() {
6  return countDown;
7  }
8  public void increment() {
9  if (counter > 0) {
10 counter++;
11 }
12 notifyObservers();
```

```
13 }
```

Listing 3.3: Java Codebeispiel

```
1  public class View implements Observer {
2  private Model model;
3  private Stage stage;
4  private Label label;
5  private Button countButton;
6  public View(Model model, Stage stage) {
7  this.model = model;
8  this.stage = stage;
9  label = new Label("Counter: " + model.getCounter());
10 btIncrement = new Button("Increment by 1");
11 stage.setScene(new Scene(new VBox(label, countButton)));
12 model.addObserver(this);
13 }
14 @Override
15 public void update(Observable o, Object arg) {
16 label.setText("Counter: " + model.getCounter());
```

Listing 3.4: Die Klasse View

3.6.3 Beschreibung der View-Elemente

3.6.3.1 Grundlagen zu SceneBuilder

3.6.3.2 Material-Design-Bibliothek-JFoenix

3.6.4 Utility Klassen

3.6.4.1 AlertUtil

3.6.4.2 GuiUtil

3.6.5 Implementierung von CSS

3.6.5.1 Grundlagen

3.6.5.2 StartupCSS

3.6.5.3 HomeCSS

3.6.6 Internationalisierung

3.6.6.1 Gründe der Umsetzung

3.6.6.2 Integration in das Interface

3.6.6.3 Ressourcen Manager

3.6.6.4 Resource Utility

3.6.6.5 Bereitgestellte Ressources Bundles

3.7 Hardwarenahe-Programmierung

3.7.1 Einrichtung des Mikrocontrollers

3.7.2 Konzept und Ablaufdiagramm zur Kartenausgabe

3.7.3 Request- und Responsehandling

3.8 Teilaufbau

3.8.1 Auswahl des Zielsystems

3.8.1.1 Arduino

3.8.1.2 Raspberry PI

3.8.2 Auswahl des Displays

3.8.3 Montage und Testaufbau

3.8.4 Konfiguration des Zielsytsems

3.8.4.1 Betriebssystem

4 Gesamtsystemtests

Anhang

A Zeitaufzeichnung

B Persönlicher Anhang 1

C Abkürzungsverzeichnis

D Abbildungsverzeichnis

E Tabellenverzeichnis

F Listings