

# Fachbericht

BACHELORTHESES  
IoT COCKTAILMASCHINE  
TEAM SCHENK & AEBI  
31. Oktober 2020



<b>Betreuer Dozent:</b>	Prof. Dr. Schleuniger, Pascal
<b>Experte:</b>	Theiler, Robert
<b>Institution:</b>	FHNW Brugg/Windisch
<b>Studiengang:</b>	Elektro- und Informationstechnik
<b>Semester:</b>	Frühlingssemester 2020

## **Danksagung**

Da diese Arbeit zur Zeit des Corona bedingten Lockdowns entstanden ist und keine Möglichkeit bestand in den Schulwerkstätten zu arbeiten, konnte nicht auf externe Hilfe verzichtet werden. Vor allem auch weil die Arbeit ein voll umfängliches Produkt darstellt, welches nicht nur elektrotechnische sondern auch mechanische Komponenten beinhaltet wurde eine gesamte Werkstatt benötigt. Diese wurde und von Claudia und Thomas Aebi zur Verfügung gestellt. Weiter wurden einige mechanische Komponenten auf der Drehbank angefertigt, was spezielles Equipment voraus setzte. Dabei wurden wir von Peter Aebi fachkräftig unterstützt. Im weiteren konnten wir uns jeder Zeit auf Herrn Christoph Biel verlassen, welcher die Studenten mit viel Herzblut jeder Zeit unterstützt. Sei es wenn man Messgeräte benötigt, Bestellungen erfassen oder Geld zurückfordern muss. Zu guter Letzt wurden wir durch das ganze Projekt hindurch von unserem Fachcoach Herrn Schleuniger unterstützt. Bei all diesen Personen wollen wir uns herzlichst bedanken. Auch jenen, die nicht direkt mit dem Projekt zu tun hatten, uns jedoch während dieser Zeit unterstützt haben und zur Seite standen, vielen herzlichen Dank. Ohne diese Mithilfe wäre es uns nicht möglich gewesen dieses Projekt auf diese Weise zu vollenden.

## Abstract

Der technologische Fortschritt ist seit Jahren am explodieren. Das Internet der Dinge, oder auf Englisch auch Internet of things, kurz IoT, ist ein Produkt aus diesem Fortschritt. Es beschreibt ein Netz aus physikalischen und virtuellen Komponenten, welche miteinander kommunizieren, um einem Grösseren ganzen zu dienen. Im Zentrum steht nebst der Interaktion zwischen Mensch und elektronischen Systemen die Interaktion zwischen verschiedenen Systemen. In dieser Bachelorthesis wird anhand der Entwicklung einer IoT-Cocktailmaschine demonstriert, welche Teilsysteme wie genutzt werden können, um ein fertiges Produkt zu entwickeln. Die Herausforderung liegt im gesamten Produktentwicklungsprozess und umfasst:

- die Projektauswahl, -Planung und -Abgrenzung
- das Erstellen der Anforderungen an das Gesamtsystem und der darin enthaltenen Teilsysteme
- die Wahl, das Testen, Implementieren der Komponenten
- die Evaluation des Gesamtsystems

Für die Umsetzung benötigt es einen Software-, einen elektronischen und einen mechanischen Teil. Der mechanische Teil beinhaltet die Unterbringung der Zutaten und der Elektronik, der Beförderung und Messung der Flüssigkeiten mittels Pumpen und Durchflusssensoren sowie die Bewegung des Glases mit einem bürstenlosen Gleichstrommotor. Die Verschalung rundet die Maschine ab und verhindert, dass sensible Teile angefasst oder nass werden können. Zum elektrotechnischen Teil gehört unter anderem ein Display, über welches der Benutzer die Cocktails auswählen, erstellen oder bearbeiten kann. Eine mikroSD-Karte ermöglicht Cocktails, Zutaten und maschinenspezifische Zustände zu speichern. Ein RFID-Reader liest Tags aus, was es ermöglicht Lieblingsgetränke zu hinterlegen und direkt in der Maschine anzuwählen. Für die Verbindung zwischen Android-Applikation und Cocktailmaschine wird ein Wireless-/Bluetoothmodul verwendet. Mittels MOSFET-Schaltungen werden während der Erstellung von Cocktails die Pumpen vom Mikrocontroller ein- und ausgeschaltet. Die Ansteuerung und Regelung des bürstenlosen Gleichstrommotors ergibt sich aus einem FOC-Treiber und einem Gate-Treiber. Der ABN-Encoder liefert das Feedback zur Lage des Rotors, welches benötigt wird als Regelparameter für den FOC-Treiber. Die entwickelte Leiterplatine gehört zum Elektronikteil und bildet das Bindeglied zwischen Mechanik und Elektronik. Der Softwareteil wird in vier Teilsoftwares geteilt. Alle Teile werden in verschiedenen Sprachen geschrieben. Die Firmware auf dem Mikrocontroller, welche in C geschrieben wurde, regelt den Programmfluss der gesamten Maschine. Die Firmware auf dem Wireless-/Bluetoothmodul, welche in Arduino geschrieben wurde, bildet die Schnittstelle zwischen den Signalen der Android-Applikation und der Firmware. Die Android-Applikation wurde mit dem Online-Tool App-Inventor entwickelt und ermöglicht dem Anwender Cocktails über Fernzugriff zu erstellen und einem RFID-Tag zu zu weisen. Bei der letzten Programmierumgebung handelt es sich um den Nextion Editor, mit welchem Nextion Displays Programmiert werden können.

Damit alle Komponenten ansteuerbar sind, mussten einige Anpassungen an der Hardware vorgenommen werden. Dazu gehören aufgrund von Störungen auf dem SPI-Bus das Umlegen der SPI-Leitungen des RFID-Readers vom Mikrocontroller auf das Wireless-/Bluetoothmodul und das Umlegen der SPI-Leitungen des FOC-Treibers auf ein separates Software-SPI. Die Motorengruppe läuft aufgrund von Defekten auf der Leiterplatine mit externen Boards. Die Firmware läuft bis auf wenige Punkte sauber. Dazu gehört zum Beispiel, dass der Abbruchprozess beim Erstellen eines Getränktes nicht wunschgemäß Funktioniert. Die gesetzten Ziele wurden alle erreicht werden. Ausserdem wurden fast alle Wunschziele erfüllt.

## Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
<b>2 Ausgangslage und Erweiterungen</b>	<b>2</b>
<b>3 Mechanik</b>	<b>4</b>
3.1 Rahmen . . . . .	4
3.2 Getränkeschlitten . . . . .	5
3.3 Verkleidung . . . . .	6
3.4 Unterbringung der Elektronik . . . . .	6
3.5 Kühlbox . . . . .	7
<b>4 Printaufbau</b>	<b>9</b>
<b>5 Teilsysteme</b>	<b>11</b>
5.1 Speisungen . . . . .	11
5.2 Motor . . . . .	16
5.3 Flüssigkeitsbeförderung . . . . .	24
5.4 Programmierschnittstellen . . . . .	26
5.5 Benutzerschnittstellen . . . . .	29
5.6 Beleuchtung . . . . .	34
5.7 Mikrocontroller . . . . .	35
5.8 SD-Karte . . . . .	36
<b>6 Inbetriebnahme</b>	<b>38</b>
6.1 Speisungen . . . . .	38
6.2 Programmierschnittstellen . . . . .	39
6.3 Mikrocontroller . . . . .	40
6.4 Benutzerschnittstellen . . . . .	41
6.5 SD-Karte . . . . .	44
6.6 Flüssigkeitsbeförderung . . . . .	45
6.7 Beleuchtung . . . . .	46
6.8 Motor . . . . .	47

<b>7 Software</b>	<b>58</b>
7.1 Mikrocontroller . . . . .	58
7.2 Wireless-/Bluetoothmodul . . . . .	62
7.3 Displaysoftware . . . . .	65
7.4 Android App . . . . .	75
<b>8 Zielerreichung</b>	<b>79</b>
8.1 Pflichtzielerfüllung . . . . .	80
8.2 Wunschzielerfüllung . . . . .	81
8.3 Zielerreichung . . . . .	82
<b>9 Persönliches Schlusswort</b>	<b>84</b>
<b>10 Ehrlichkeitserklärung</b>	<b>85</b>
<b>Literatur</b>	<b>86</b>
<b>A Programmierschnittstellen</b>	<b>90</b>
A.1 Inbetriebnahme . . . . .	90
<b>B Mikrocontroller</b>	<b>94</b>
B.1 Inbetriebnahme . . . . .	94
<b>C Wireless-/Bluetoothmodul</b>	<b>99</b>
C.1 Inbetriebnahme . . . . .	99
<b>D LED</b>	<b>101</b>
D.1 Inbetriebnahme . . . . .	101
<b>E FOC-Treiber</b>	<b>103</b>
E.1 Teilschemas Breakout-Board . . . . .	103
E.2 Inbetriebnahme . . . . .	105
E.3 Register Openloop . . . . .	106
<b>F Gate-Treiber</b>	<b>107</b>
F.1 Standard-Schaltkreis . . . . .	107
F.2 Blockdiagramm . . . . .	107
F.3 Inbetriebnahme . . . . .	108
F.4 Register Gate-Treiber . . . . .	110

<b>G ABN-Encoder</b>	<b>111</b>
G.1 Register Closed-Loop . . . . .	111
<b>H Software Ramp</b>	<b>112</b>
H.1 Implementierung . . . . .	112
H.2 Matlab-Script . . . . .	114
<b>I Print</b>	<b>118</b>
I.1 KiCad Schema . . . . .	118

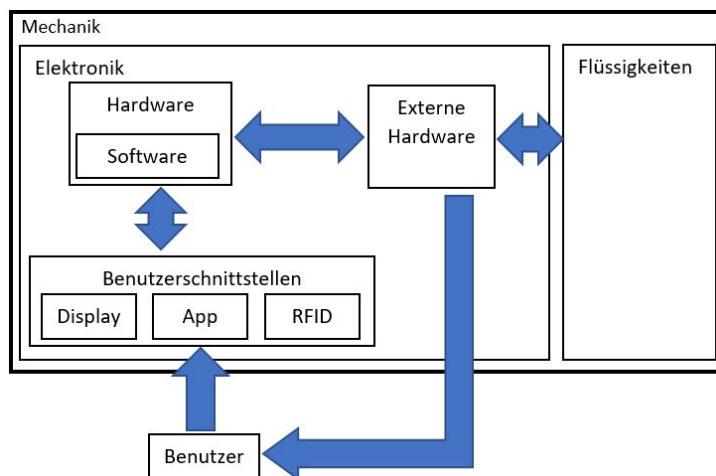
## 1 Einleitung

Eine gelungene Party auf die Beine zu stellen verlangt einem einiges ab. Vor allem kostet es eine Menge Aufwand und Zeit. Dies gilt besonders, wenn es darum geht mit vielen Freunden zusammen zu feiern. Neben der gelungenen Musikauswahl und den Snacks dürfen die Getränke auf keinen Fall fehlen. Um diese sicherzustellen, gibt es mehrere Möglichkeiten. Einerseits könnte jeder seine eigenen Getränke mitbringen, was jedoch bedeutet, dass es unter Umständen eine riesige Sauerei gibt oder viele Flaschen in der Gegend rumstehen. Andererseits könnte man als Gastgeber selber anbieten Cocktails zu mixen und so den Getränkenachschub zu gewährleisten. Da gibt es jedoch ein grosses Problem. Als Gastgeber möchte man nicht den ganzen Abend hinter der Bar stehen müssen, sondern lieber bedenkenlos mitfeiern.

Damit genau dies möglich ist, wurde der PartyMixer geschaffen. Dieses System soll dem Anwender eine einfache Cocktaillerstellung dank dem innovativen Zusammenspiel von intuitiven Komponenten bieten. Ein Mikrocontroller bildet dabei das Herz dieser Cocktailmaschine, welche mit der aus C-Code bestehenden Software alle Komponenten wie Pumpen, Durchflussmessgeräte und Motor ansteuert. Unterstützt wird dieser von einem Wireless-/Bluetoothmodul, welches für zusätzliche Schnittstellen sorgt und eine Anbindung einer App oder eines Webhostes ermöglicht. Weitere Systeme wie RFID oder Beleuchtungssysteme runden das Paket als gesamtes ab.

Ein stabiles Rahmensystem bildet das Grundgerüst einer voll umfänglich entwickelten Mechanik, welche nicht nur die Elektronik und deren Komponenten beinhaltet, sondern auch die Flüssigkeiten. Eine Kältekammer sorgt hierbei für ideale Getränketerminperaturen.

All diese Komponenten interagieren gemäss Blockbild in Abbildung 1.1 zusammen.



**Abbildung 1.1:** Konzept des PartyMixer's

In den folgenden Kapiteln ist dokumentiert, wie die Cocktailmaschine im Detail aussieht und wie die einzelnen Teilsysteme aufgebaut sind. Dazu gehören die elektronischen Teilsysteme, das dazugehörige Printdesign, die Software und die Mechanik. In Kapitel 3 wird die komplette Mechanik beschrieben, wohingegen in Kapitel 4 und 5 die Elektronik unter die Lupe genommen wird. Außerdem wird in 6 das System in Betrieb genommen. Die Software wird in Kapitel 7 beschrieben.

## 2 Ausgangslage und Erweiterungen

### Projekt 5

Das Projekt 5 hat die Basis für das Projekt 6 gebildet. In diesem wurde ein Konzept erarbeitet, welches die Hauptkomponenten der Maschine festlegte und deren Arbeitsweise. Durch diese Konzepterstellung war es möglich ein Blockschaltbild zu erstellen, welches diese Komponenten im Groben zusammen vereinte. Zu den Systemen zählten dabei:

- Pumpen zur Beförderung der Flüssigkeiten
- Durchflussmessgeräte zur Dosierung der Flüssigkeiten
- Ein Motor, welcher ein Glas befördern soll
- Ein Display, über welches die Maschine bedient werden kann

Nach der Auswahl dieser Hauptkomponenten konnte dann die Ansteuerung theoretisch aufgebaut werden. Dazu gehörten folgende Komponenten:

- Ein Mikrocontroller, welcher die Software beinhaltet und alle für die Ansteuerungen benötigten Schnittstellen bietet
- Eine 48V-Spannungsquelle zur Speisung des Motors
- Eine 12V-Spannungsquelle zur Speisung der Pumpen
- Eine 5V-Spannungsquelle zur Speisung des Mikrocontrollers, der Durchflussmessgeräte und des Displays
- Eine 3.3V-Spannungsquelle zur Speisung der Motorenansteuerung

Ein weiterer Teil des Projekt 5 war es, die gewählten Komponenten und die daraus entstandenen Teilsysteme in einem Testaufbau aufzubauen und zu evaluieren. Der Mikrocontroller wurde im Projekt 5 so ausgewählt, dass es möglich war weitere Komponenten anzubinden.

### Projekt 6

Das im Projekt 5 entstandene Blockschaltbild ist nun im Projekt 6 um weitere Teilsysteme ergänzt worden, was in Abbildung 2.1 zu sehen ist. Auf diese Teilsysteme wird mit den vorherigen Teilsystemen detailliert im Kapitel 4 und 5 eingegangen. Zu den neuen Systemen gehören die grün umrahmten Teilsysteme. Die Erweiterungen bestehen aus:

- Ein Wireless-/Bluetoothmodul, welches eine externe Ansteuerung per Web-Server oder Android App ermöglicht und eine dazugehörige Programmierschnittstelle
- Einem RFID Lesegerät
- Einem SD-Karten Slot
- Einer Maschinenbeleuchtung
- Einem ABN-Encoder zur Positionsgebung des Motors

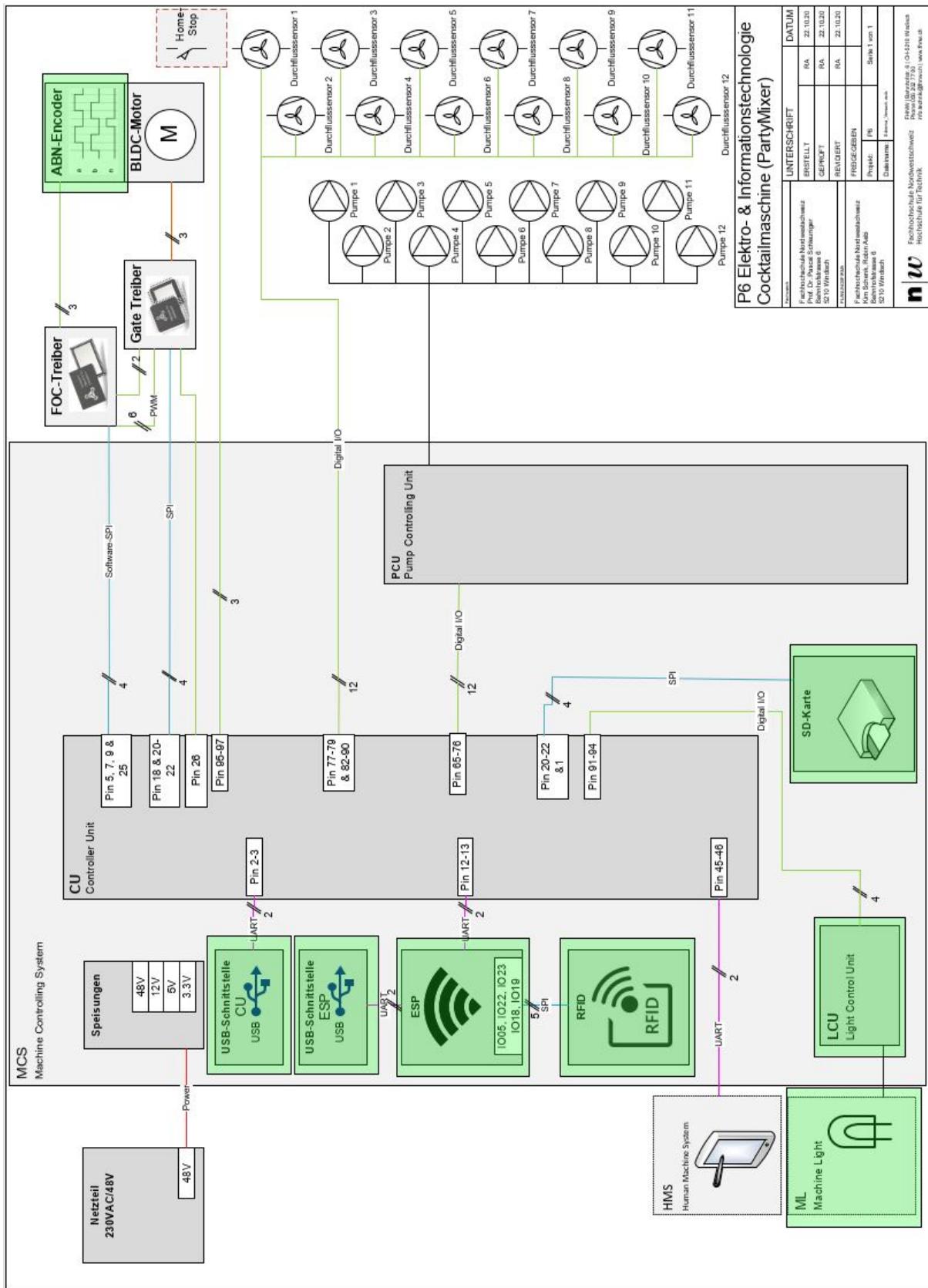


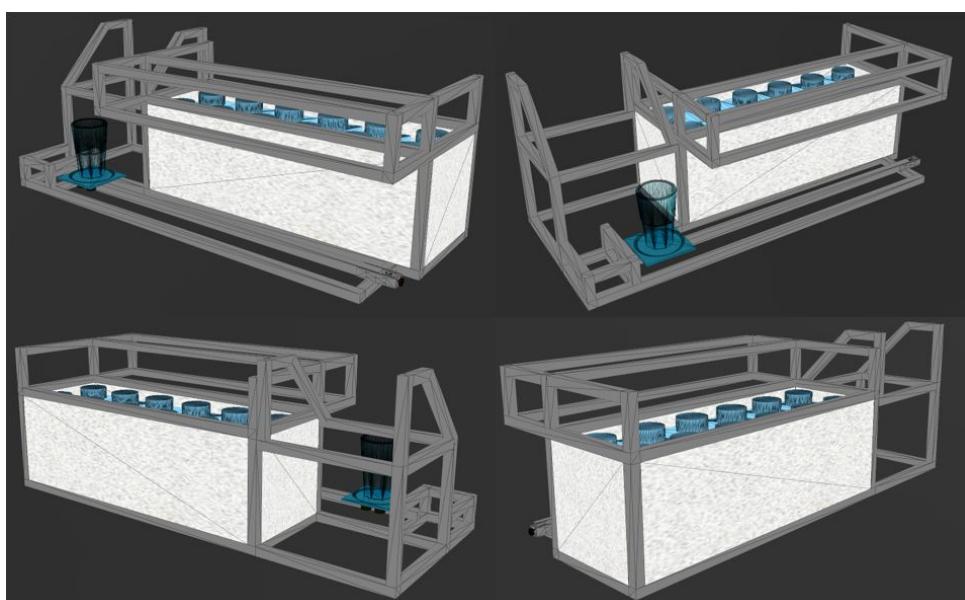
Abbildung 2.1: Blockschaltbild des PartyMixer's gemäss Projekt 6.

### 3 Mechanik

Damit die Maschine auch benutzt werden kann, wurde eine Mechanik entwickelt, welche aus folgenden Komponenten besteht:

- Rahmen (Grundgerüst)
- Getränkeschlitten
- Verkleidung
- Unterbringung der Elektronik
- Kühlbox

Nachdem alle Komponenten bekannt waren, wurde dabei vor der Bauphase ein Dreidimensionales Modell mittels des Windows Programm «3D Builder» gezeichnet, welches in Abbildung 3.1 zu sehen ist.



**Abbildung 3.1:** 3D-Modell des Partymixers

#### 3.1 Rahmen

Der Rahmen besteht aus 20x20mm Aluminium X-Profilen. Diese haben einerseits den Vorteil, dass sie sehr stabil und anderseits vielseitig einsetzbar sind. Die Profile verfügen über eine Nut, in welche sogenannte «Nutensteine» eingesetzt werden. Diese verfügen wiederum über eine Bohrung mit einem Gewinde. Durch diese Kombination kann mittels Nutenwinkel ein stabiler Rahmen aufgebaut werden. In Abbildung 3.2 sind diese Bauteile zu sehen.



**Abbildung 3.2:** Bauteile für den Rahmen des Partymixers [1][2][3]

### 3.2 Getränkeschlitten

Um ein Getränk möglichst ruckelfrei und stabil von einem zum anderen Punkt befördern zu können wurde auf ein Schlittensystem zurückgegriffen, welches so ähnlich auch im 3D-Druck Bereich eingesetzt wird. Dieses besteht ebenfalls aus einem X-Profil gemäss Kapitel 3.1, einem Schlitten, einem Riemen, einem Riemenspanner und einem Zahnrad.



**Abbildung 3.3:** Bauteile für den Schlitten des Partymixers [4] [5] [6] [7]

Der Schlitten verfügt über zwei fest angebrachte Rollen auf der einen Seite und über zwei verstellbare Rollen auf der anderen Seite des Profils. Dies ermöglicht eine saubere Zentrierung des Schlittens. Der Riemen wird innerhalb des Profils geführt, was einerseits den Riemen schützt und andererseits einen schöneren Optik dient. Am einen Ende des X-Profils befindet sich der Riemenspanner, mit welchem sich die Spannung des Riemens einstellen lässt und am anderen Ende das Zahnrad, welches vom Motor angetrieben wird.

Um den Motor und den Encoder montieren zu können, wurde eine Halterung entworfen und im 3D-Drucker ausgedruckt. Da die Achse des Motors relativ kurz ist und das Zahnrad sich in einem grösseren Abstand zum Montageort des Motors befindet, wurde eine Achse aus Aluminium auf der Drehbank gedreht. Damit keine Flüssigkeit in das Profil rein tropft, wurde der Schlitten nicht direkt unter den Pumpen installiert, sondern ein wenig versetzt. Um das Glas jedoch trotzdem mittig unter den Pumpen führen zu können, wurde eine Glasauflage im 3D-Drucker gedruckt, welche auch den Riemen sauber im Profil führt. Diese Teile sind in Abbildung 3.4 zu sehen.



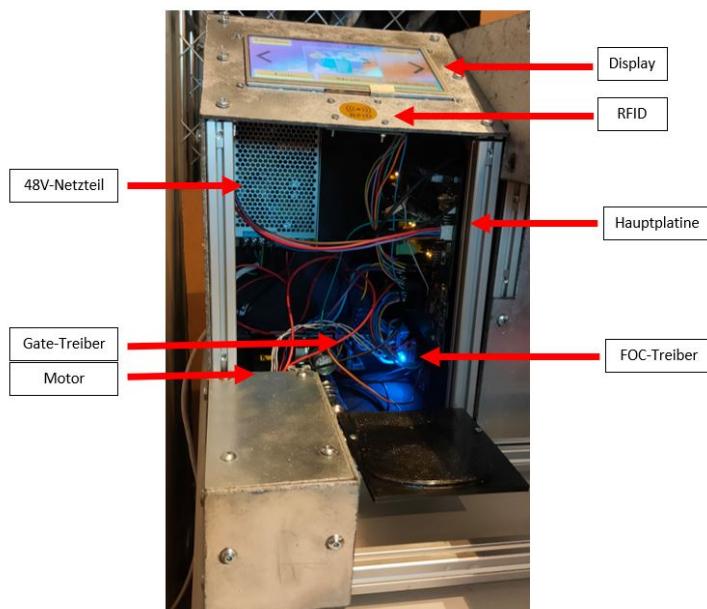
**Abbildung 3.4:** Schlitten, Motorhalterung und Achse des PartyMixers

### 3.3 Verkleidung

Um der Maschine ein Outfit verpassen zu können, wurde diese mit einfachen Spandruckplatten verkleidet, welche dann mit einem schwarzen Lack grundiert wurden und mit einem Chromstahl-lack veredelt. Eine bessere Variante wären Aluminium- oder Chromstahlplatten, welche jedoch schwieriger zu bearbeiten und einiges teurer sind. Da es sich jedoch um einen Prototyp handelt, wurden die Spandruckplatten verbaut. Diese sind wie die anderen Teile auch an den X-Profilen mittels Nutensteinen festgeschraubt.

### 3.4 Unterbringung der Elektronik

Der grösste Teil der Elektronik befindet sich auf der linken Seite der Maschine unterhalb des Displays gemäss Abbildung 3.5. Darin ist einerseits der Motor untergebracht und anderseits das Netzteil, die RFID-Antenne, das Display und das Herz der Maschine - die Hauptplatine.



**Abbildung 3.5:** Unterbringung der Elektronik des Partymixers

Über Kabelkanäle werden die Pumpen und die Durchflussmessgeräte an der Hauptplatine angeschlossen. Diese befinden sich über dem Getränkeschlitten. Um diese an einem X-Profil befestigen zu können wurden auch hier 12 Halterungen gezeichnet und im 3D-Drucker ausgedruckt. Zu sehen ist dies in Abbildung 3.6.



**Abbildung 3.6:** Montage der Pumpen und Durchflussmessgeräte des Partymixers

### 3.5 Kühlbox

Damit die Getränke möglichst lange kühl bleiben, wurde eine Kühlbox aufgebaut. Dazu wurde der Bereich, welcher für die Flüssigkeiten bestimmt ist, mit Styropor ausgekleidet. Dieser hält die Kälte länger in der Box, da Styropor viele Luftblasen enthält und diese als gute Isolation dienen. Um den Effekt noch zu verstärken wurde die Box zusätzlich mit Aluminiumtape verkleidet, was nochmals zusätzlich isoliert und auch die Box sauber verschließt. Um die Getränke zusätzlich kühl halten zu können, wurde Platz für Kuhlakkus geschaffen, welche im Gefrierfach gefroren und dann bei Gebrauch zwischen den Flüssigkeiten platziert werden können. Dieser Aufbau ist in Abbildung 3.7 zu sehen.



**Abbildung 3.7:** Kühlbox des PartyMixers.

Cocktails werden normalerweise sehr kalt serviert. Dabei spricht man bei gerührten Cocktails von 2°C - 4°C und bei geschüttelten Cocktails von 0°C - 2°C. Diese Temperaturen werden mit Eis erreicht. Dieses muss beim PartyMixer separat hinzugefügt werden. Um jedoch die Getränke auch ohne Eis kalt geniessen zu können wurde das Ziel gesetzt, dass die Kühlbox die Getränke während mindestens 8std. kühl hält. Dies soll unter folgenden Bedingungen zu Hause erreicht werden können: [8]

- Vorherige Lagerung der Getränke im Kühlschrank bei 5°C im unteren Bereich des Kühlschranks
- Einsetzen der gefrorenen Kühlakkus in die Maschine vor Gebrauch
- Betrieb der Maschine bei geschlossener Kühlbox

Diese Bedingungen können von jedem Benutzer eingehalten werden und sind somit für den Betrieb auch realistisch. Das Ziel war es, dass die Getränke in der Kühlbox über den Verlauf eines ganzen Abends von 18:00 Uhr bis 02:00 Uhr unter den vorhin genannten Bedingungen eine Temperatur von 5°C nicht überschreiten.

Um dies zu testen, wurde die Maschine unter den oben genannten Bedingungen einmal komplett befüllt und über eine Zeitdauer von 24std. ausgemessen. Die Raumtemperatur betrug dabei 22.6°C. Die Flüssigkeitsbehälter wurden mit Wasser komplett befüllt und im Kühlschrank auf exakt 5°C abgekühlt. Die Kühelemente wurden im Gefrierfach gefroren. In Abbildung 3.8 ist das Ergebnis zu sehen.

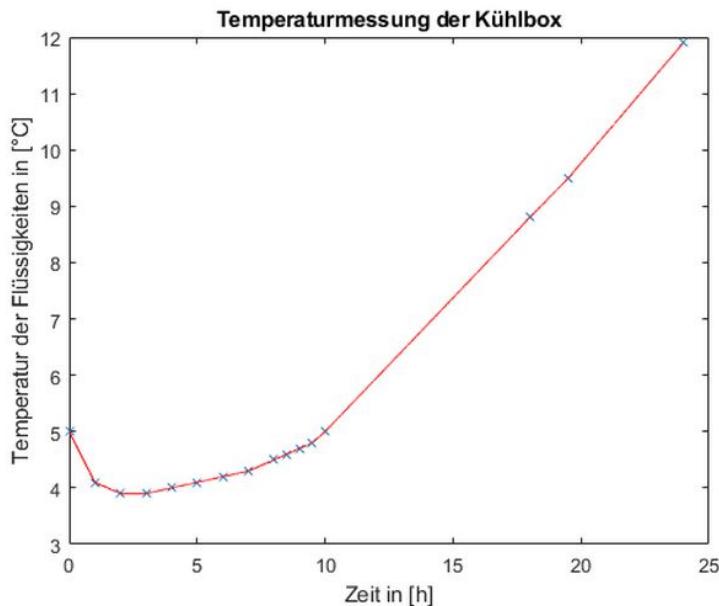


Abbildung 3.8: Kältetest des Partymixers

Der Test hat gezeigt, dass die Flüssigkeiten sich durch die Kühlakkus zuerst wie erwartet noch ein wenig abkühlen, bevor die Temperatur wieder anfängt anzusteigen. Nach 8std. erreichten die Flüssigkeiten eine Temperatur von 4.6°C und nach 10std. wurde die 5°C Marke erreicht. Es können also unter den genannten Bedingungen über eine Zeitdauer von 10std. gekühlte Getränke aus der Maschine genossen werden, welche eine Temperatur von 4°C - 5°C haben.

## 4 Printaufbau

In Abbildung 4.1 ist der Aufbau des entstandenen Prints zu sehen. Auf diesem sind die in Abbildung 2.1 gezeigte Blöcke wieder zu erkennen. In Tabelle 4.1 und Abbildung 4.2 ist zusätzlich ein Layer-Detail des Herstellers und dessen Spezifikationen zum Print zu sehen. Das Schema zum Print befindet sich im Anhang Kapitel I.1.

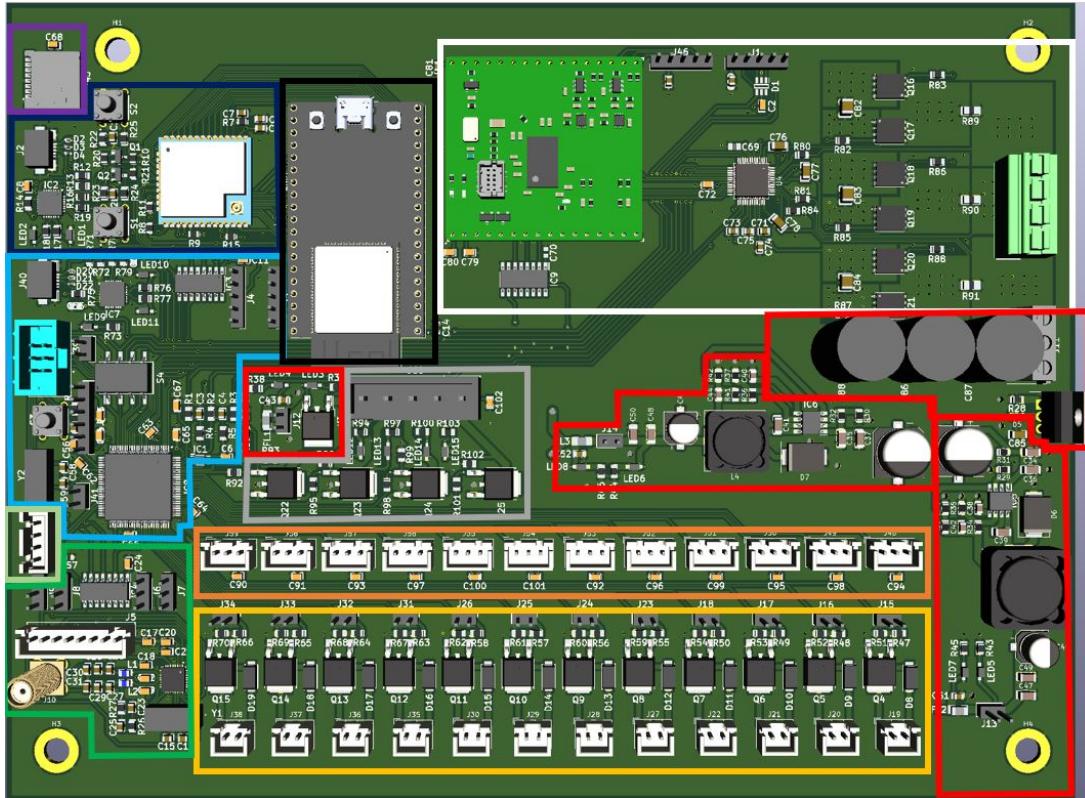


Abbildung 4.1: Aufbau des Prints als Modell

Technische Details Print	
Dimensionen	222mm x 166mm ( $\pm 0.2\text{mm}$ )
PCB Dicke	1.6mm ( $\pm 10\%$ )
Material	FR-4 (Tg 130-140C)
Oberfläche	HASL (with lead)
Anzahl Layers	4
Top / Bottom	1oz (35 $\mu\text{m}$ )
Layer 2 / 3	0.5oz (17 $\mu\text{m}$ )

Tabelle 4.1: Spezifikationen der Platine [9]

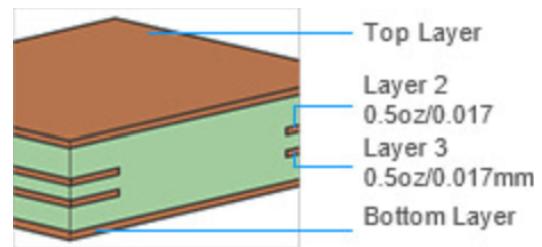


Abbildung 4.2: Querschnitt der Platine [9]

Der Print ist grundsätzlich in zwei Gruppen unterteilt. Der rechte Teil des Prints beinhaltet den Leistungsteil und auf der linken Seite des Prints ist der Logikteil zu finden. In Abbildung 4.1 sind die einzelnen Teile auf dem Print zur besseren Überschaubarkeit in Farben unterteilt. Die Aufgabenbereiche, die Funktionsweise und Details der einzelnen Teile werden in Kapitel 5 behandelt. Folgende Legende soll lediglich eine grobe Übersicht bieten.

## Legende zu Abbildung 4.1

**Speisungen:** In den rot markierten Bereichen sind die Speisungsteile zu sehen, wobei sich auf der rechten Seite des Prints die Eingangskondensatoren mit Verpolschutz, die 12V-Speisung und die 5V-Speisung befinden. Das 48V-Netzteil ist gemäss Blockschaltbild 2.1 extern angebracht. Im Logikteil des Prints auf der linken Seite ist die 3.3V-Speisung zu finden.

**Mikrocontroller:** Im hellblau markierten Bereich auf der linken Seite des Prints befindet sich das Herzstück der Elektronik. Der Mikrocontroller mit der dazugehörigen Programmierschnittstelle, welche einen Micro-USB Anschluss zu Programmierzwecken beinhaltet.

**Pumpenansteuerung:** Auf der unteren Seite des Prints sind im gelben Bereich die Anschlüsse der Pumpen zu finden, welche die Flüssigkeiten befördern. Dazu gehört jeweils eine kleine Leistungsansteuerung mittels Logik-MOSFET und Freilaufdiode.

**Durchflussmessgeräte:** Über der Pumpenansteuerung befinden sich im orange markierten Bereich die Anschlüsse der Durchflussmessgeräte, welche auch die Speisung der Durchflussmessgeräte und einen Entstörkondensator beherbergen.

**Maschinenbeleuchtung:** Um die Maschine mit einem Showeffekt auszustatten wurde eine Lichtansteuerung eines RGBW-LED-Streifen implementiert gemäss Kapitel 2. Diese ist im grau markierten Bereich zu finden und beinhaltet einen Anschluss, sowie vier Logik-MOSFET's zur Ansteuerung.

**Wireless- /Bluetoothmodul:** Um mit externen Gerätschaften kommunizieren zu können, befindet sich im Dunkelblau markierten Bereich das Wireless-/Bluetoothmodul. Auch dieses beinhaltet neben dem eigentlichen Modul eine dazugehörige Programmierschnittstelle gemäss Blockschaltbild 2.1.

**SD-Karte:** Der microSD-Kartenslot befindet sich links oben im Violett markierten Bereich.

**Motoransteuerung:** Die Motorenansteuerung ist im weissen Teil des Print's zu finden. Sie hat einen grossen Teil des Prints in Beschlag genommen und ist von links nach rechts in drei Teile unterteilt. Als Erstes der FOC-Treiber, gefolgt vom Gate-Treiber und der H-Brücke mit dem Anschluss des Motor's. Dieser Teil ist jedoch bei der fertigen Maschine durch externe Elektronik ersetzt worden.

**Display:** Das Display benötigt keine grosse Ansteuerungslogik und besteht aus diesem Grund auch nur aus einem einzelnen Stecker. Dieser befindet sich im Bereich des Mikrocontrollers, im hellgrün markierten Bereich.

**Wireless-/Bluetoothmodul 2:** Aus Sicherheitsgründen wurde für den Fall, dass Probleme bei der Inbetriebnahme auftauchen, ein Steckplatz für ein Evaluationsboard des Wireless- / Bluetoothmoduls implementiert. Dieses wird jedoch nicht verwendet. Somit generiert dies neue Möglichkeiten für weitere Anschlüsse am Wireless-/Bluetoothmodul im Dunkelblauen Teil. An diesen Anschlüssen befindet sich nun das RFID Evaluationsboard sowie die SPI-Leitungen vom Mikrocontroller zum FOC-Treiber.

**RFID-Modul:** Für den Fall, dass noch genügend Zeit übrigbleiben sollte, wurde ein eigenes RFID-Board gelayoutet. Diese ist im dunkelgrünen Bereich zu sehen. Dieses wird jedoch in diesem Projekt nicht verwendet, da sich das RFID-Modul nun im schwarz markierten Bereich als Evaluationsboard befindet.

Auf dem Print mussten im Laufe des Projektes kleinere Anpassungen vorgenommen werden. Dies gehört zum Entwicklungsprozess dazu. Somit mussten zum Beispiel Leitungen aufgetrennt werden, neue Leitungen geschaffen oder sogar neue Baugruppen implementiert werden. Die Änderungen sind im Schema Rot gekennzeichnet.

## 5 Teilsysteme

Da der Partymixer aus vielen kleineren und grösseren Teilsystemen besteht, werden diese in diesem Kapitel einzeln aufgelistet und im Detail angeschaut. Es wird dabei bei jedem Teilsystem auf drei Punkte eingegangen, die Problemstellung (Was ist der Zweck der Teilschaltung und weshalb wird sie benötigt?), das Schema (Wie sieht die Schaltung aus und welche Bauteile werden benötigt) und der Funktionsbeschrieb der Schaltung (Welche Funktion haben die Bauteile der Schaltung und wie hängen sie zusammen).

### 5.1 Speisungen

Die Speisung, welche das System mit Strom versorgt, ist ein essentieller Bestandteil des Party-Mixer's. Im System befinden sich vier verschiedene Speisungen auf verschiedenen Spannungsniveaus. Die Eingangsspannung, womit zwei andere Speisespannungen erzeugt werden und der Motor betrieben wird, wird von einem 48V Netzteil erzeugt. Mittels Step-Down Reglern wird aus der 48V Eingangsspannung eine 12V und eine 5V Speisung erzeugt. Bei der vierten Speisung handelt es sich um einen einfachen Linearregler, welcher aus den 5V eine 3.3V Speisung realisiert. In den folgenden Unterkapitel können die Details der einzelnen Spannungsquellen entnommen werden und deren Aufgabengebiet. Die Grundlage sämtlicher Berechnungen der 5V und 12V Speisungen sind dem Datenblatt entnommen [10, S.10].

#### 5.1.1 48V Speisung

Der Motor wird mit einer Spannung von 48V betrieben. Dies ist zugleich auch die höchste verwendete Speisespannung. Um diese Speisung gewährleisten zu können, wird ein fertiges Netzteil von Aliexpress gemäss **Fachbericht 5** eingesetzt. Somit entfällt das Schema für diesen Speisungsteil. Ein Anschauungsbild des eingesetzten Netzteils kann in Abbildung 5.1 begutachtet werden [11].



**Abbildung 5.1:** Anschauungsbild des 48V Netzteils [11]

Eine Leistungsabschätzung war jedoch unabdingbar. Auch diese wurde im Projekt 5 durchgeführt. Unter Berücksichtigung der Schaltungsteile welche noch im Projekt 6 ergänzt werden, wurde dieses dann ausgewählt und eingekauft. Es hat sich dabei herausgestellt, dass das Netzteil mindestens eine Leistung von 150W bieten muss. Die Leistungsabschätzung kann im **Fachbericht 5** eingesehen werden.

### 5.1.2 12V Speisung

Die Pumpen werden mit 12V betrieben, was der Grund ist, weshalb eine 12V Speisung implementiert werden musste. Dazu wird ein Schaltspannungsregler verwendet. Dieser wandelt mittels Step-Down-Prinzip die 48V des Netzteils in eine konstante Spannung von 12V. Es handelt sich hierbei um einen Regler von Monolithic Power Systems. Genauer gesagt um den MP24943DN-LF. Die Auswahl ist auf dieses Bauteil gefallen, da mit 48V eine relativ hohe Eingangsspannung verarbeitet werden muss. Der MP24943DN-LF kann am Eingang mit Spannungen von 4.5-55V arbeiten und dabei eine Ausgangsspannung von 0.8-45V erzeugen. Dies bei einem maximalen Strom von bis zu 3A. Die Realisierung der 12V Speisung ist in Abbildung 5.2 zu sehen [12] [13] [14, S.355-370].

#### Schema

Das Schema in Abbildung 5.2 kann in fünf Teile unterteilt werden. Da wären zuerst die Eingangskondensatoren, welche mit C32, C34 & C36 realisiert sind. Diese Eingangskondensatoren werden gefolgt von einem Spannungsteiler, welcher den Enablepin auf aktiv setzt. Der eigentliche Regler wird mittels des IC7, D6 & L3 realisiert. Mittels zweier Spannungsteiler, wird die gewünschte Ausgangsspannung, sowie die «Overvoltage-Protection» eingestellt. Vor dem Ausgang der Schaltung ist dann erneut eine Kondensatorstufe implementiert, welche das Ausgangssignal glättet. [13]

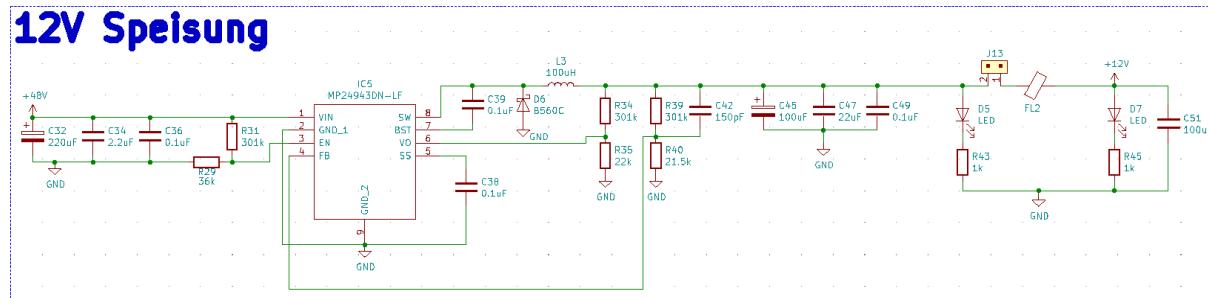


Abbildung 5.2: Schema der 12V-Speisung

#### Funktionsbeschrieb der Schaltung

Um den MP24943DN-LF auf aktiv zu setzen, wird eine minimale Spannung von 1.8V vorausgesetzt. Fällt diese unter 0.4V, so wird dieser MP24943DN-LF auf inaktiv gesetzt. Damit der Spannungsregler immer eingeschaltet bleibt, wird mittels zweier Widerstände R29 & R31 ein Spannungsteiler realisiert, welcher den Enable (EN) Pin auf 5V und somit auf aktiv setzt. Dieser Spannungsteiler musste implementiert werden, da alle Eingangspins des MP24943DN-LF außer dem  $V_{in}$  einen maximale Eingangsspannung von 6.5V verkraften können [10, S.3].

Die gewünschte Ausgangsspannung wird mittels Spannungsteiler R39 & R40 eingestellt, welche auf den Feedback Eingang (FB) rückgekoppelt werden. Diese berechnet sich laut Datenblatt gemäss Formel 5.1.

$$R40 = \frac{R39}{\frac{V_{out}}{0.8} - 1} \quad (5.1)$$

Bei einem Widerstandsverhältnis von  $R39=301\text{k}\Omega$  &  $R40=21.5\text{k}\Omega$  entspricht dies einer Ausgangsspannung von 12V. Gemäss Datenblatt solten die Widerstände  $> 10\text{k}\Omega$  sein, um die Verluste niedrig zu halten.

Um einer Überspannung vorbeugen zu können, wird am Eingang Voltage-Overshoot (VO) ein Spannungsteiler implementiert. Diese wird am VO-Eingang mit einer Referenzspannung von 0.9V verglichen. Übersteigt die Spannung an VO die Referenzspannung von 0.9V, so wird der Regler ausgeschaltet, bis die Spannung wieder unter 0.9V fällt. Als maximale Ausgangsspannung wurde hierbei eine Spannung von 13V gewählt. Diese Wahl wurde getroffen, da die 12V ausschliesslich für die Ansteuerung der Pumpen verwendet wird und diese eine Spannung von 13V verkraften können ohne Schaden zu nehmen. Der Spannungsteiler wird gemäss Datenblatt mit der Formel 5.2 berechnet [12].

$$R35 = \frac{R34}{\frac{V_{ovp}}{V_{ovref}} - 1} \quad (5.2)$$

Bei einem Widerstandsverhältnis von  $R34=301\text{k}\Omega$  &  $R35=22\text{k}\Omega$  entspricht dies einer Überspannungsschutzschwelle von 13.21V. Gemäss Datenblatt solten die Widerstände  $> 10\text{k}\Omega$  sein, um die Verluste niedrig zu halten. [13].

Der Rippel des Spulenstroms lässt sich gemäss Formel 5.3 berechnen. Dieser sollte gemäss Datenblatt ca. 30% des maximalen Ausgangsstroms von 3A betragen.

$$L3 = \frac{V_{out} \cdot (V_{in} - V_{out})}{V_{in} \cdot \Delta I_L \cdot f_{osc}} \quad (5.3)$$

Der interne Oszillatator läuft dabei bei einer Frequenz von 100kHz. Bei der ausgewählten Spule von  $100\mu\text{H}$  resultiert ein  $\Delta I_L$  von 0.9A erhalten. Ausserdem wird im Datenblatt darauf hingewiesen, dass die gewählte Spule auf mindestens 125% des maximalen Ausgangsstroms von 3A ausgelegt werden soll. Auch der Gleichstromwiderstand der Spule sollte  $\leq 200\text{m}\Omega$  sein [10, S.8-10].

Mit den Kondensatoren C45, C47 & C49 wird die Ausgangsspannung zum Abschluss noch geglättet. Bei den Eingangskondensatoren sowie den Ausgangskondensatoren sollte es sich um low ESR Typen handeln. Um die restlichen hochfrequenten Störungen herauszufiltern, ist zum Abschluss ein Ferrit implementiert worden.

Als Massnahme um die Schaltung in Betrieb nehmen zu können, wurde mittels Jumper sichergestellt, dass die Speisung vom System abgekoppelt werden kann. Ein LED zeigt an, ob die Speisung funktioniert oder nicht. So ein LED ist jeweils vor und nach dem Jumper implementiert worden. Der Ferrit FL2 filtert noch letzte Störungen heraus.

### 5.1.3 5V Speisung

Der Mikrocontroller, sowie die Durchflussmessgeräte und das Display werden mit 5V betrieben. Aus diesem Grund wurde eine 5V Speisung implementiert. Dazu wird derselbe Schaltspannungsregler wie bei der 12V Speisung in Kapitel 5.1.2 verwendet, jedoch mit anderen Komponenten. Die Realisierung der 5V Speisung kann in Abbildung 5.3 eingesehen werden.

## Schema

Das Schema in Abbildung 5.3 ist wie bei der 12V Speisung gemäss Kapitel 5.1.2, in fünf Teile unterteilt. Da sind zuerst die Eingangskondensatoren, welche mit C31, C33 & C35 realisiert sind. Diese Entstörstufe wird wiederum gefolgt von einem Spannungsteiler, welcher den Enable auf aktiv setzt. Der eigentliche Regler wird hier mittels des IC6, D5 & L4 realisiert. Mit zwei Spannungsteiler wird die gewünschte Ausgangsspannung sowie die «Overvoltage-Protection» eingestellt. Vor dem Ausgang der Schaltung ist erneut eine Kondensatorstufe implementiert, welche das Ausgangssignal glättet.

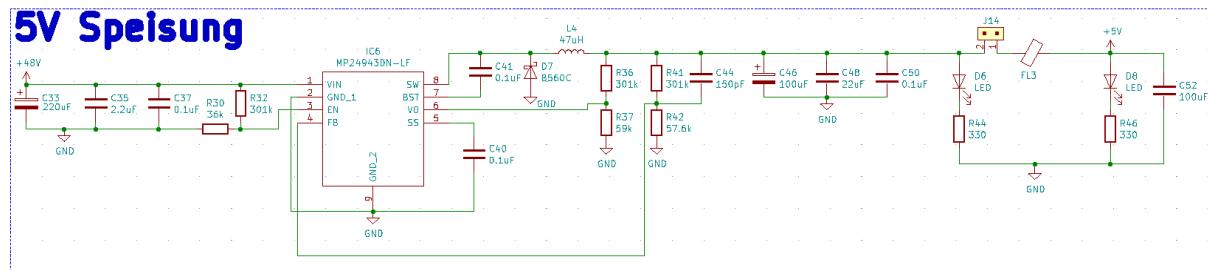


Abbildung 5.3: Schema der 5V-Speisung

## Funktionsbeschrieb der Schaltung

Bei der 5V-Speisung wurde mittels R29 & R31 ein Spannungsteiler realisiert, welcher das IC gemäss Kapitel 5.1.2 auf aktiv setzt.

Das Widerstandsverhältnis von R41 & R42, das die Ausgangsspannung definiert, ist gemäss Formel 5.1 berechnet. Daraus resultiert für  $R41=301\text{k}\Omega$  und für  $R42=57.6\text{k}\Omega$ , was einer Ausgangsspannung von 4.98V entspricht.

Beim Überspannungsschutz ist darauf geachtet worden, dass der Mikrocontroller nur in einem Spannungsbereich von 4.5V-5.5V betrieben werden darf. Die maximal verträgliche Eingangsspannung liegt laut Datenblatt bei 6V. Somit muss der Überspannungsschutz so gestaltet werden, dass die Schwelle von 6V nicht überschritten werden kann. Um dies zu erreichen, ist für  $R36=301\text{k}\Omega$  und  $R37=53\text{k}\Omega$  gewählt worden. Gemäss Formel 5.2 erhält man so eine Überspannungsschutzwelle von 6V [14, S.1].

Der interne Oszillator läuft wiederum bei einer Frequenz von 100kHz. Bei der ausgewählten Spule L4 von  $47\mu\text{H}$  erhält man mittels Formel 5.3 ein  $\Delta I_L$  von 0.953A. Hier gilt gemäss Datenblatt, dass die gewählte Spule auf mindestens 125% des maximalen Ausgangsstroms von 3A ausgelegt werden muss. Auch der Gleichstromwiderstand der Spule sollte  $\leq 200\text{m}\Omega$  sein [10, S.3].

Mit den Kondensatoren C46, C48 & C50 wird die Ausgangsspannung zum Abschluss noch geglättet. Bei den Eingangskondensatoren sowie den Ausgangskondensatoren sollte es sich gemäss Datenblatt um low ESR Typen handeln, um eine bessere Störungsunterdrückung zu gewährleisten. Zum Abschluss ist ein Ferrit implementiert worden, welcher allfällige hochfrequente Störungen herausfiltert.

Bei der 5V Speisung wurde ebenfalls ein Jumper zu Testzwecken und zwei LED's implementiert. Ausserdem findet sich auch hier wieder ein Ferrit FL3, welcher letzte Störungen beseitigt.

### 5.1.4 3.3V Speisung

Um die Treiber der Motorenansteuerung, das Wireless-/Bluetoothmodul, die SD-Karte und die RFID-Schaltung zu betreiben, wird zusätzlich eine 3,3V-Speisung verbaut. Da es sich dabei

nicht um Leistungstreibende Elemente handelt, genügt ein einfacher Linearregler, der von der 5V-Speisung aus betrieben wird.

### Schema

Bei dem Linearregler handelt es sich konkret um den LF33CDT-TRY von STMicroelectronics. Dieser hat eine fixe Ausgangsspannung von 3.3V bei einem maximalen Strom von 1A. Das dazugehörige Schema kann in Abbildung 5.4 begutachtet werden.

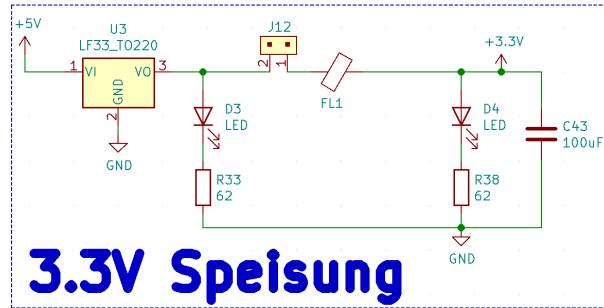


Abbildung 5.4: Schema der 3.3V-Speisung

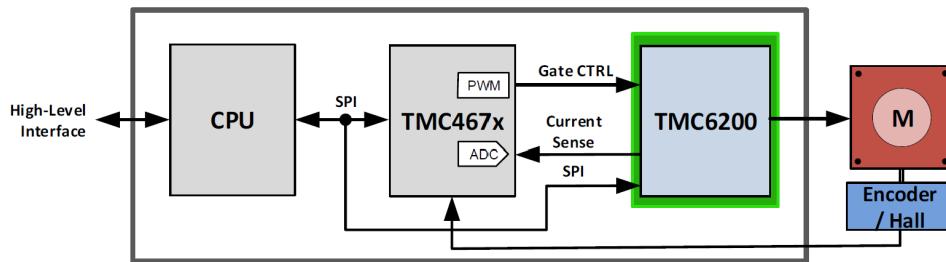
### Funktionsbeschrieb der Schaltung

Der Linearregler benötigt keine spezielle Beschaltung. Er wird lediglich an die 5V Speisung angeschlossen. Am Eingang und am Ausgang ist ein Entstörkondensator implementiert.

Wie bei der 12V-Speisung in Kapitel 5.1.2 und der 5V-Speisung in 5.1.3 wurde ein Jumper, sowie zwei LED's implementiert. Ein Ferrit FL1 filtert letzte Störungen heraus.

## 5.2 Motor

Die Antriebsgruppe wird benötigt, um die Positionierung des Glases unter die Flüssigkeitsauslässe zu gewährleisten. Abbildung 5.5 zeigt, aus welchen Komponenten die Gruppe besteht. Der Mikrocontroller ist über einen SPI-Bus mit dem FOC-Treiber (TMC4671) und dem Gate-Treiber (TMC6200) verbunden. Über diesen Bus findet die Initialisierung der Motorengruppe (Informationen zum Motor, PI-Regler, PI-Limits, PWM, ABN-Encoder) und die Positionsangabe des Motors statt. Sobald der Mikrocontroller dem FOC-Treiber eine Position vorgibt, arbeitet der FOC-Treiber autonom und berechnet anhand der Initialisierungen den Modulationsindex für die Kommutierung des Motors. Anhand des Modulationsindex werden die PWM-Signale erzeugt, sie bilden die Gate-CTRL-Signale in der Abbildung. Der Gate-Treiber (TMC6200) stellt die benötigte Energie zur Verfügung, welche benötigt wird, um die Gates der H-Brücken-MOSFETs zu laden und entladen. Weiter erhöht die Bootstrap-Schaltung des Gate-Treibers die Gate-Spannung der High-side-MOSFETS auf die Motorspannung plus Schaltspannung. Außerdem überwacht der Gate-Treiber die Spannungen an der H-Brücke, was ermöglicht Fehlschaltungen zu detektieren. Die mit der H-Brücke erzeugten Schaltsignale magnetisieren die Spulen des Motors und lassen den Rotor drehen. Der dabei fließende Strom durch die Spulen wird mittels Shunts vom Gate-Treiber gemessen, verstärkt und an den FOC-Treiber geleitet. Der ABN-Encoder (AMT332S-V) gibt das Feedback über die Lage des Rotors an den FOC-Treiber. Anhand des Strommesssignals vom Gate-Treiber und des Lagesignals des ABN-Encoders regelt der FOC-Treiber den Modulationsindex nach, was auch bei sich ändernden Umgebungsbedingungen eine exakte Positionierung des Rotors und somit des Glases unter den Flüssigkeitsauslässen ermöglicht.



**Abbildung 5.5:** Blockschaltbild Konfiguration IC's mit BLDC und Encoder. [15, S.1]

### 5.2.1 FOC-Treiber

Beim FOC-Treiber handelt es sich um den TMC4671. Abbildung 5.6 zeigt die interne Logik des ICs. Kern des FOC-Blocks bildet die Clarke- und Park-Transformation. Die Clarke-Transformation transformiert die drei Teilflüsse ( $I_U$ ,  $I_V$ ,  $I_W$ ) der Spulen im dreiphasigen System zu einem Gesamtfluss im zweiphasigen System. Die Park-Transformation transformiert die gemessenen Wechselgrößen im stehenden, statorbezogenen Koordinatensystem in einfach regelbare Gleichgrößen im sich drehenden, rotorbezogenen Koordinatensystem. So kann interpretiert werden, wie der Rotor zum Fluss steht. Ziel der FOC-Logik ist, dass der Fluss so ausgerichtet ist, dass nur eine tangentiale Kraft und somit ein Drehmoment auf den Rotor wirkt, was in Abbildung 5.7 veranschaulicht wird. Im FOC-Block wird dies durch die Null vor dem  $PID_D$ -Block verdeutlicht. Die Stärke des Drehmoments wird mit dem  $PID_Q$ -Block geregelt. Die Abweichung ergibt sich durch die Differenz der Eingangssignale auf die PID-Blöcke. Die Blöcke  $PID_D$  und  $PID_Q$  berechnen anhand der Abweichung eine neue Soll-Spannung, damit der Fluss und somit das Drehmoment gemäß den Sollwerten am Rotor anliegt. Die iPark-Transformation

transformiert das sich drehende, rotorbezogene Koordinatensystem zurück in ein stehendes, statorbezogenes Koordinatensystem und die iClarke-Transformation transformiert wiederum das zweiphasige System in ein dreiphasiges System. Anhand dieser Sollspannungen wird von der eingebauten PWM-Engine das Gate-CTRL-Signal erzeugt und an den Gate-Treiber (Power Stage) weitergeleitet.

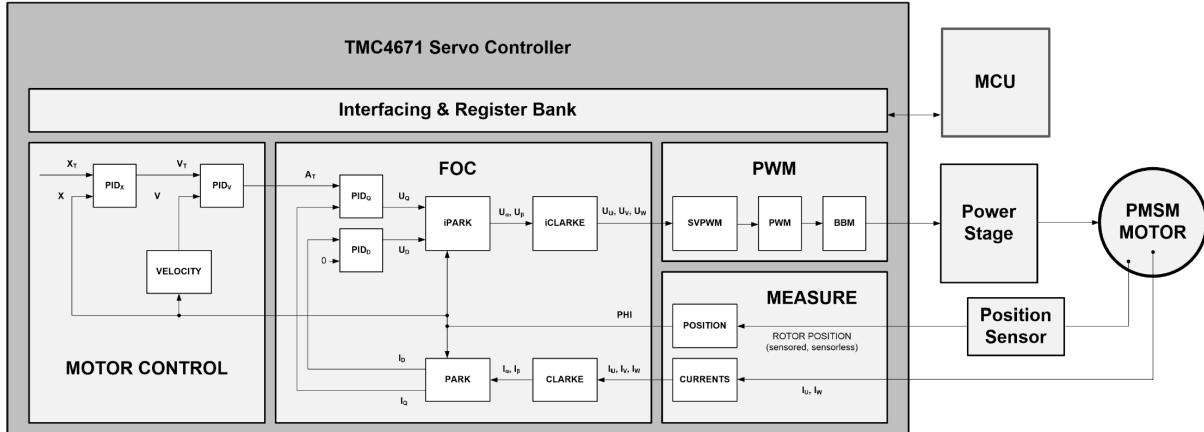


Abbildung 5.6: Logik des FOC-Treibers. [15, S.14]

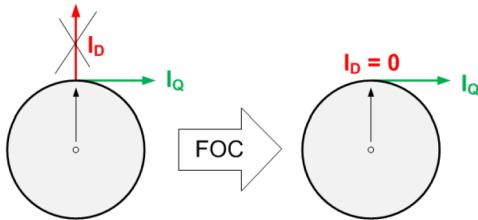


Abbildung 5.7: Veranschaulichung FOC-Regelung. [15, S.9]

Die Sollwerte für die Stärke des Drehmoments kann im Drehmoment-Modus manuell vorgegeben werden oder im Geschwindigkeits- oder Positionsmodus vom überlagerten Geschwindigkeitsregelkreis. Im Gegensatz zu einem Motor, dem nur eine Spannung vorgegeben wird, kann bei diesem Regelverfahren bei einer Abweichung des vorgegebenen Parameters nachgeregelt werden. So kann beispielsweise bei gröserer Belastung die angelegte Spannung erhöht werden, sodass die vorgegebene Geschwindigkeit gehalten werden kann.

Dies führt zur Kaskadenregelung, der PI-Regelstruktur im FOC-Treiber. Eine Kaskadenregelung besteht in der Regel aus drei überlagerten Regelkreisen. Der innerste Regelkreis ist der Stromregelkreis. Dieser ist dem Geschwindigkeitsregelkreis unterlagert. Der Geschwindigkeitsregelkreis ist wiederum dem Positionsregelkreis unterlagert. Bei einer Kaskadenregelung ist die Ausgangsgrösse des überlagerten Regelkreises die Eingangsgrösse des unterlagerten Regelkreises. Aus Stabilitätsgründen ist darauf zu achten, dass die Nachstellzeit der äusseren Regelkreise grösser ist, als die der Inneren.

Mit der Begrenzung der PI-Werte wird die Mechanik geschont und der Motor vor Überlast geschützt. Dies wird erreicht, indem die Ausgangsgrößen der PI-Regler mit einem Limit versehen werden. So kann der Strom durch die Spulen begrenzt werden, eine Maximalgeschwindigkeit definiert werden und die Enden der Positionen vorgegeben werden. Die Limits sind auf den Motor abgestimmt [16].

## Schema

Der Treiber und dessen Beschaltung ist als Breakoutboard erhältlich. Daran sind diverse Anschlussmöglichkeiten vorhanden. In Abbildung 5.8 ist erkennbar, welche Pins für den PartyMixer verwendet werden:

- SPI Input
- Phasenströme Input
- Encoder Input
- Motorspannung Input
- Gate-CTRL-Signale Output

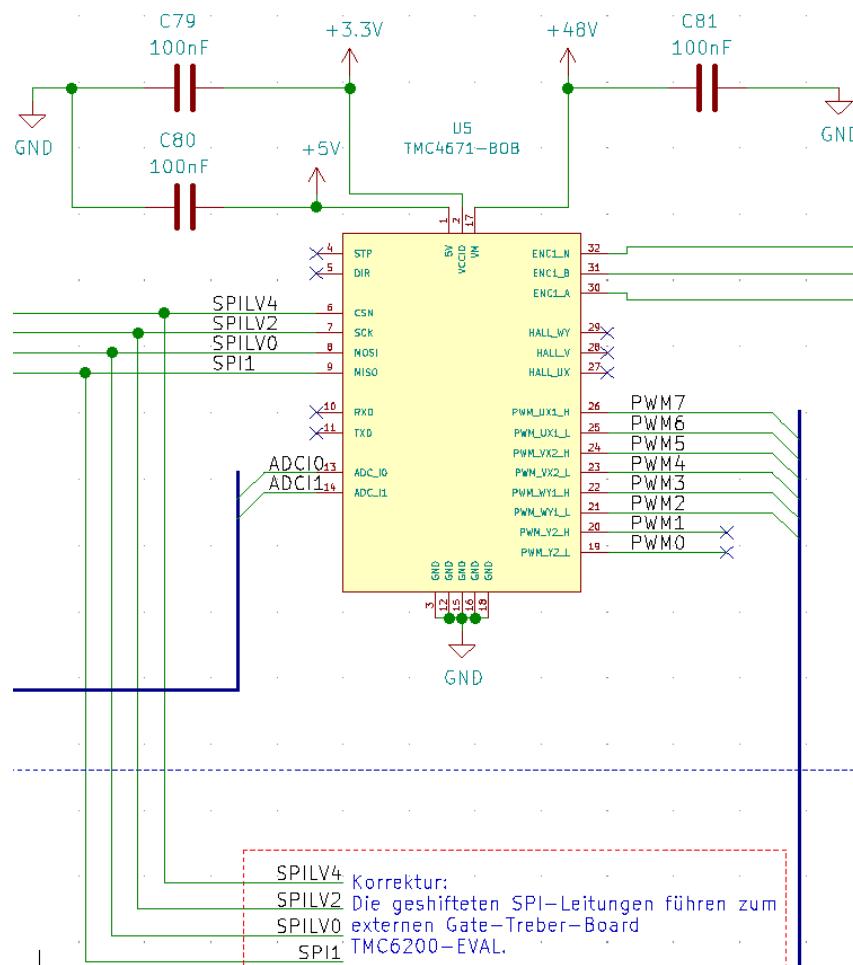


Abbildung 5.8: Schema FOC-Treiber.

Im Schema sind rote Korrekturen zu sehen. Da das Breakout-Board extern montiert wurde, sind die geshifteten SPI-Leitungen an den Header-Pins nicht belegt und werden deswegen für das externe Breakout-Board des Gate-Treibers verwendet.

### Funktionsbeschrieb der Schaltung

Im Folgenden werden kurz die Teilsysteme auf dem TMC4671-BOB beschrieben. Sämtliche Teilschemas aus dem Datenblatt sind im Anhang Kapitel E.1 aufgezeigt.

Der **SPI**-Kommunikationseingang ist nicht speziell geschützt. Im Layout wurde darauf geachtet, dass die angelegte Spannung nicht über 3.3V steigt. Dies wird mit einem Level-Shifter zwischen Mikrocontroller und FOC-Treiber gewährleistet.

Der Eingang zur **Strommessung** ist mit einem Tiefpass geschützt. Hier werden hochfrequente Störsignale, welche auf der Übertragungsstrecke induziert werden, gefiltert. Das Filter hat die Zeitkonstante:

$$\tau = R308 \cdot C300 = 100\Omega \cdot 100pF = 10ns \quad (5.4)$$

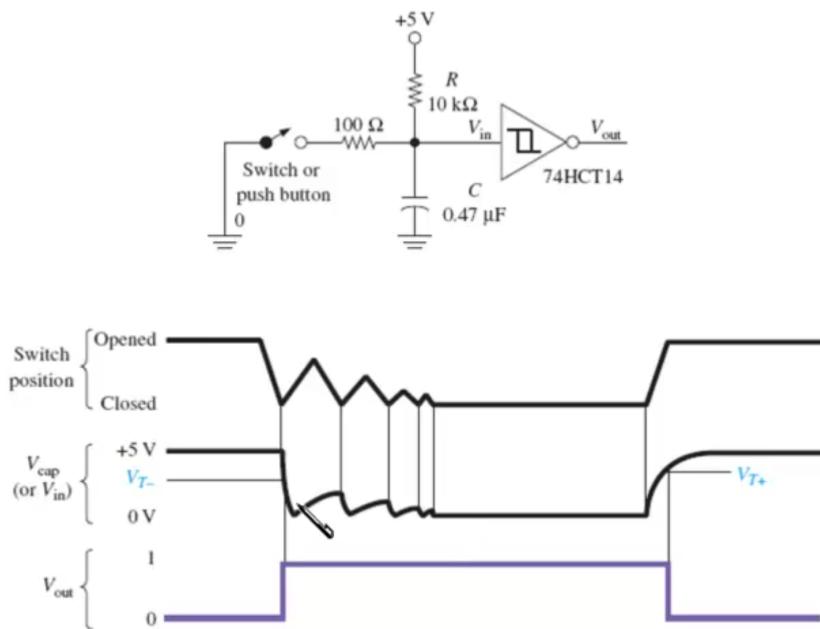
Um den Eingang der **Encoder**-Pins zu schützen, wird ein Schmitt-Trigger mit eingebautem Level-Shifter verwendet. Zudem wird mit dem vorgeschalteten Kondensator- und Widerstandsnetzwerk ein Entprellen der Encoder-Signale erreicht. Wird der Input auf 0V gezogen, so entlädt sich der Kondensator über die Widerstände R200-R202 mit der Zeitkonstante:

$$\tau = R200 \cdot C204 = 1k\Omega \cdot 100pF = 100ns \quad (5.5)$$

Sobald der Eingang vom Encoder nicht mehr auf 0V gezogen wird, so lädt sich der Kondensator über die Widerstände R200-R202 sowie R206,R208,R210. Somit ergibt sich eine längere Zeitkonstante:

$$\tau = (R200 + R210) \cdot C204 = (1k\Omega + 4.7k\Omega) \cdot 100pF = 570ns \quad (5.6)$$

Ohne diese Schaltung kann es vorkommen, dass bei einem schwingenden Signal die Auswertungslogik unabsichtlich einen Schritt misst, ohne dass einer vorgekommen ist. In Abbildung 5.9 ist die Funktion verdeutlicht.



**Abbildung 5.9:** Schmitt-Trigger Debounce-Schaltung. Achtung, die Abbildung entspricht nicht direkt der Schaltung auf dem Breakout-Board, sie dient lediglich der Veranschaulichung. [17, 3:00]

Die **Motorspannung** ist wichtig, um den Kommutierungsvorgang zu berechnen. Wichtiger als die Zeitkonstante ist hier ein Spannungsteiler, welcher die Motorspannung auf unter 3.3V bringt. Dazu wird folgende Formel angewendet:

$$U_{TMC} = U_M \cdot \frac{R310}{R310 + R311} = 48V \cdot \frac{1k\Omega}{1k\Omega + 100k\Omega} = 0.7V \quad (5.7)$$

Ausserdem ergibt sich aus der Schaltung ein Eingangsfilter [18]:

$$\tau = C302 \cdot \frac{R310 \cdot R311}{R310 + R311} = 10nF \cdot \frac{1.5k\Omega \cdot 100k\Omega}{1.5k\Omega + 100k\Omega} = 147us \quad (5.8)$$

Die **Gate-CTRL-Signale** für den Gate-Treiber gehen direkt auf die Header-Pins des BOBs.

### 5.2.2 Gate-Treiber

Die Gate-CTRL-Signale werden vom Gate-Treiber aufbereitet. Das entsprechende Bauteil ist der TMC6200. Er verarbeitet die Signale so, dass die H-Brücke angesteuert werden kann. Die H-Brücke besteht aus sechs MOSFETs, die gemäss den Gate-CTRL-Signalen die Zwischenkreisspannung (48VDC) schalten.

Um ein optimales Schaltverhalten zu erreichen, die Umschaltverluste zu verkleinern und somit die daraus entstehende Abwärme zu verringern, ist es vorteilhaft, die Gates so schnell wie möglich zu laden und zu entladen. Ein Gate-Treiber löst dieses Problem, indem die benötigte Energie zur Verfügung stellt. Ausserdem hebt der Gate-Treiber mit einer Bootstrap-Schaltung die Gate-Spannungen der High-Level-MOSFETS auf die Motorspannung plus Gatespannung, damit der MOSFET sicher durchschaltet.

Das Blockdiagramm und eine Beispielschaltung befinden sich im Anhang Kapitel F.1 und F.2. Das Blockdiagramm zeigt, dass der Gate-Treiber aus einer Treiber-Logik für den Motor, einem SPI-/Pinsettings-Interface, einer Diagnosenlogik, Strommessschaltung und diversen unterstützenden Schaltungen wie Gate-Spannungsversorgung besteht. Weiter hat der ausgewählte Gate-Treiber folgende Features integriert:

- Strommessung und -Verstärkung
- Kurzschlussdetektion
- Schaltkontrolle
- Thermische Begrenzung

### Schema

Der Treiber und dessen Beschaltung ist als Breakoutboard erhältlich. Daran sind diverse Anschlussmöglichkeiten vorhanden. In Abbildung 5.10 ist zu sehen, welche Pins verwendet werden:

- SPI Input
- Phasenströme Input und Output
- Motorspannung Input
- Gate-HIGH und -Low-Signale Output
- Messung Schaltspannung Input

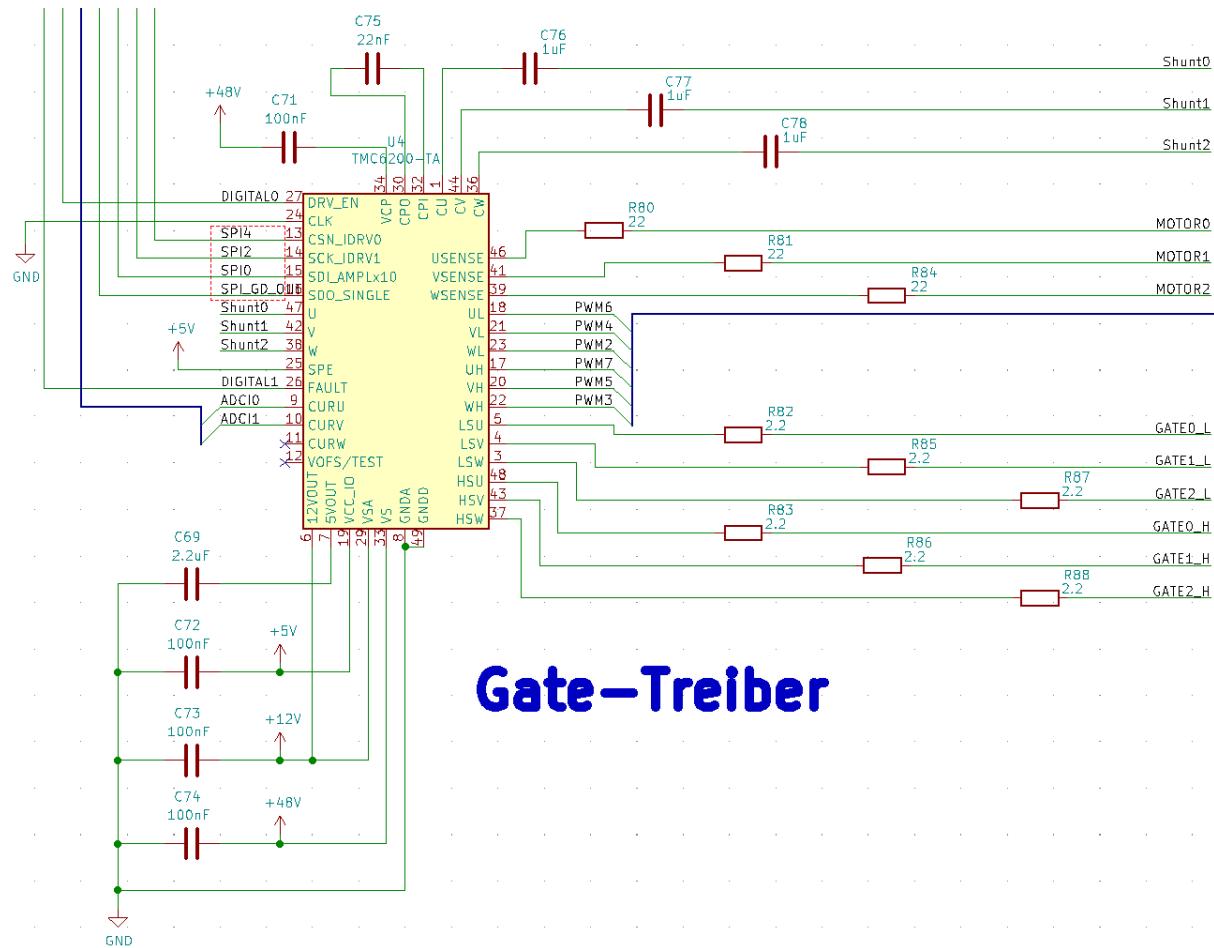


Abbildung 5.10: Schema Gate-Treiber.

Im Schema sind die Korrekturen Rot gekennzeichnet. Im Layout wurden die Pins fälschlicherweise vertauscht. Weiter ist zu beachten, dass die Gate-Treiber-Schaltung sich nicht mehr auf der Leiterplatine befindet, sondern auf einem externen Breakout-Board. Die SPI-Pins werden nicht mehr über die vorgesehenen Leitungen angesteuert, sondern über die ursprünglichen Header-Pins des FOC-Treibers.

### Funktionsbeschrieb der Schaltung

Da das Gate ein kapazitives Verhalten zeigt, ist der Strom, welcher zu Beginn ins Gate fliesst, sehr hoch. Die **Gate-Vorwiderstände** R82, R83 und R85 bis R88 in Abbildung 5.10 begrenzen diesen. Die Dimensionierung der Gate-Widerständen lehnt an die MOSFET Gate-Drain-Ladung (Miller charge) an. Aus dem Datenblatt ist einer Tabelle zu entnehmen, wie gross der Widerstand bei gegebener Miller charge sein muss. Die Tabelle mit den ermittelten Parameter ist im Anhang Kapitel F.3.2 zu finden. Die Gate-Ladung des ausgewählten MOSFET's beträgt 61nC. Gemäss der Tabelle muss der Vorwiderstand mit  $R_{Gate} \leq 2.5\Omega$  dimensioniert werden. Dies bedingt, dass programmierbare Register DRV\_STRENGTH auf 1 bis 3 gesetzt ist [15, S.13].

Wird von einem High-Zustand in einen Low-Zustand gewechselt, kann aufgrund von Induktivitäten die Spannung unterschiessen. Die **Schutzwiderstände** R80, R81 und R84 in Abbildung 5.10 schützen die Messeingänge des Gate-Treibers vor diesem Effekt. Die Dimensionierung dieses Widerstands wird im Datenblatt mit einem Widerstandswert zwischen  $10\Omega$  und  $22\Omega$  angegeben [15, S.10].

Die **Bootstrap-Kondensatoren** C76 - C78 in Abbildung 5.10 werden eingesetzt, um die Gate-Spannung am High-Side-MOSFET auf die anliegende Schaltspannung plus die Gatespannung anzuheben. Die Dimensionierung dieser Kondensatoren wird im Datenblatt mit einem Kapazitätswert zwischen 470nF und 1 $\mu$ F angegeben, bei einer Nennspannung von 16V oder 25V. Weiter gilt gemäss Datenblatt, dass bei MOSFET's mit einem  $Q_G \geq 40nC$  die Gatekapazität 1 $\mu$ F sein soll [15, S.10]. Da die Kapazität 61nF beträgt, ist dies der Fall.

Aufgrund der hohen Versorgungsspannung (48V), treten im IC über den Gate- und 5V-Spannungsreglern erhebliche Verluste auf. Um diese Verluste zu reduzieren, wird laut Datenblatt geraten, eine externe Gate-Spannung anzuhängen. Für beste Resultate wird eine Spannung von  $12V \pm 1V$  empfohlen. Aus dem Datenblatt des Gate-Treibers ist zu entnehmen, dass für den Kondensator C69 2.2 $\mu$ F und für den Kondensator C73 100nF verwendet werden. Beide sind in Abbildung 5.10 zu sehen. Der Kondensator C73 sowie die Kondensatoren C72 und C74 sind **Stützkondensatoren** und glätten die Versorgungsspannungen auf den jeweiligen Spannungsebenen.

### 5.2.3 H-Brücke

Das Bindeglied zwischen Kraft (Motor) und Steuersignalen wird von der H-Brücke gebildet. Durch Laden-/Entladen der MOSFET-Gates wird die Spannung am Motor kommutiert.

#### Schema

Der Schaltungsaufbau ergibt sich durch den dreiphasigen Aufbau des BLDCs. Es werden so drei Stränge gebildet, woran jeweils eine Spule verbunden wird. Der Energiefluss führt dabei über einen Strommesswiderstand. Die Eingänge der H-Brücke werden zusätzlich mit Stützkondensatoren bestückt, um eine saubere 48V-Netzspannung zu gewährleisten.

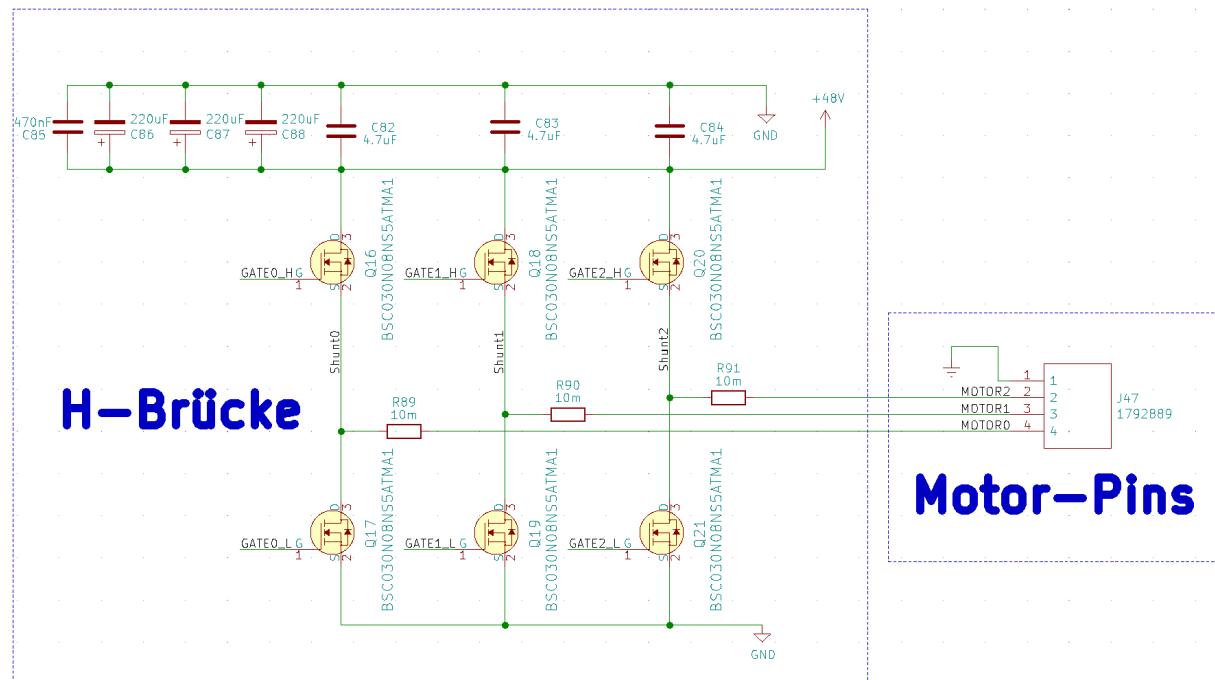


Abbildung 5.11: Schema H-Brücke.

### Funktionsbeschrieb der Schaltung

Für die Messung des Stromes wird ein **Shunt** in Serie mit der Spule des BLDC-Motors geschaltet. Die Shunts für die Strommessung R89 - R91 sind in Abbildung 5.11 zu sehen. Durch den fliessenden Strom liegt eine Spannung über dem Shunt an, welche dann vom Gate-Treiber verstärkt wird und aufbereitet an den FOC-Treiber ausgegeben wird. Die Dimensionierung der Widerstände ist an den Maximalstrom plus 25-50% Reserve angelehnt. Im Falle des AKM22h sind dies 5A. Aus dem Datenblatt ist einer Tabelle zu entnehmen, wie gross der Shunt bei gegebenem Strom sein muss. Die Schaltung für den PartyMixer wird auf 10A ausgelegt. Gemäss der Tabelle muss der Widerstand  $10\text{m}\Omega$  sein, was bedingt, dass ein Verstärkungsfaktor der Phasenströme von 10x eingestellt wird. Die zugehörige Tabelle aus dem Datenblatt ist im Anhang Kapitel F.3.1 zu finden [15, S.31].

Die Eingangsspannung der gesamten H-Brücke wird mit den **Stützkondensatoren** C85-C88 in Abbildung 5.11 geglättet. Bei C86-C88 handelt sich um Low-ESR Kondensatoren, jeweils einen pro Phase. Die Kondensatoren C82-C84 sind Stützkondensatoren, welche direkt zwischen Ein- und Ausgang eines H-Brücken-Strangs, nahe der MOSFETs platziert sind.

Die **MOSFETs** Q16-Q21 in Abbildung 5.11 bilden die eigentliche H-Brücke. Sie Schalten den Leistungsfluss gemäss den Gate-Ctrl-Signalen. Sie sind für 100A Nennstrom und 100V Sperrspannung ausgelegt. Die Gate-Source-Spannung beträgt  $\pm 20\text{V}$ .

Der Sockel J47 in Abbildung 5.11 bildet die Schnittstelle zwischen Leiterplatine und Motorleitungen. Der Motor wird direkt auf die vorgesehenen Anschluss-Pins geführt. Obwohl die Versorgungsspannung unterhalb der maximalen Berührungsspannung liegt, ist für den Fall eines Defekts im Motor zur Personensicherheit ein Pin für die Erdung des Motorengehäuses vorgesehen.

#### 5.2.4 BLDC

Ein Brushless DC-Motor<sup>1</sup> (BLDC) zeichnet sich dadurch aus, dass der Rotor mit Permanentmagneten bestückt ist. Somit ähnelt er vom Aufbau her einer Synchronmaschine mit permanent erregten Rotorwicklungen. Zur Ansteuerung hat der BLDC drei Phasenleitungen, die auf die Spulen führen. Die Spulen sind innerhalb des BLDC in Stern geschaltet. Der Motor hat drei Polpaare, womit die magnetische Winkelgeschwindigkeit drei Mal schneller ist als die mechanische.

### Schema

Zum BLDC gibt es kein Schema. Die Anschlusspins für die Motorleitungen sind in Abbildung 5.11 zu sehen.

### Funktionsbeschrieb der Schaltung

Im Folgenden werden die Eigenschaften des verwendeten BLDCs aufgelistet. So wurde beispielsweise die Versorgungsspannung nach der Nennspannung  $V_M$  ausgewählt, das Limit des Geschwindigkeitsregelkreis auf  $1500\text{min}^{-1}$  gesetzt und der Strom auf 5A begrenzt [19, S.36].

---

<sup>1</sup>Bürstenloser Gleichstrommotor

Beschreibung	Var	Wert	Einheit	Benötigt für
Nennnetzspannung	$V_M$	= 48	[V]	Versorgungsspannung
Stillstanddrehmoment	$M_0$	= 0.88	[Nm]	-
Nenndrehmoment	$M_n$	= 0.85	[Nm]	-
Spitzendrehmoment	$M_{max}$	= 2.8	[Nm]	-
Nenndrehzahl	$n_n$	= 1500	[min <sup>-1</sup> ]	Limit: Geschwindigkeit (PI-Regler)
Nennleistung	$P_n$	= 0.13	[kW]	Power Management
Stillstandstrom	$I_0$	= 5.41	[A]	-
Nennstrom	$I_N$	= 5.21	[A]	Limit: Strom (PI-Regler)
Spitzenstrom	$I_{max}$	= 21.6	[A]	-
Drehmomentkonstante	$k_T$	= 0.1632	[Nm/A]	Reglerauslegung
Rototrägheitsmoment	$J$	= 0.16	[kg·cm <sup>2</sup> ]	Reglerauslegung
Gewicht	$m$	= 1.1	[kg]	Mechanik

**Tabelle 5.1:** Charakteristische Eigenschaften des BLDC

### 5.2.5 ABN-Encoder

Für eine Geschwindigkeits- und Positionsregelung ist ein Positionsgeber unabdingbar. Diese Aufgabe übernimmt ein ABN-Encoder, welcher dem FOC-Treiber mittels digitalen Signalen die Positionsänderung in kleinen Schritten mitteilt. Die absolute Position wird vom FOC-Treiber berechnet. Gewählt wurde der AMT332S-V, weil er unempfindlich auf Staub, Schmutz und Öl ist, und weiler einfach zu montieren ist. Als Feature hat er eine über UART einstellbar hohe Genauigkeit, auf welche beim Funktionsbeschrieb eingegangen wird.

#### Schema

Nebst der 5V-Spannungsversorgung sind die Signalleitungen (A, B und N) auf den Encoder geführt. Die Leitungen gehen auf den FOC-Treiber. Dies ist im Schema des FOC-Treibers zu sehen, welches in Abbildung 5.8 dargestellt ist. Während A und B die Codierung für die relative Wegänderung sind, gibt N an, sobald eine gesamte Umdrehung erfolgte. Auf die Schaltung innerhalb des Encoders wird nicht weiter eingegangen.

#### Funktionsbeschrieb der Schaltung

Die einstellbar hohe Genauigkeit zeigt sich in Form der Bitanzahl pro Umdrehung. Die maximale Auflösung liegt bei 4096 Schaltvorgänge pro Umdrehung, was einer Auflösung von 12 Bit entspricht [20, S.1]. Der Encoder ist defaultmäßig mit diesem Wert initialisiert und wird nicht verändert. Die Auflösung ist wichtig für die Implementierung in den FOC-Treiber. Die Auflösung bezieht sich auf eine Signalleitung. Für den FOC-Treiber entspricht dies einer gesamten Auflösung von 8192 Schaltvorgängen pro Umdrehung ( $A \& B = 2 \cdot 4096\text{Bits}$ ). Die Ausgangspins sind normale Header-Pins. Der Spannungsausgang wurde mit einem Stützkondensator C89 versehen um die Versorgungsspannung zu glätten.

## 5.3 Flüssigkeitsbeförderung

Für die Flüssigkeitsbeförderung sind zwei essentielle Komponenten erforderlich. Einerseits muss die Flüssigkeit befördert werden, andererseits muss diese auch dosierbar sein. Diese zwei Hauptaufgaben übernehmen die Pumpen und die Durchflussmessgeräte.

### 5.3.1 Pumpen

Um die Flüssigkeiten sauber befördern zu können, werden Vakuummembranpumpen eingesetzt. Diese werden oft in der Lebensmittelindustrie verwendet, da diese einerseits hygienisch sind und zum andern einen guten Durchfluss bieten. Ein weiterer Vorteil dieser Pumpen ist, dass durch die Erstellung eines Vakuums auch Luft gepumpt werden kann. Dies ermöglicht einen «Kaltstart» ohne Flüssigkeit in den Schläuchen. [12]

#### Schema

Die Schaltung zur Ansteuerung der Pumpen ist einfach aufgebaut, wie in Abbildung 5.12 zu sehen ist. Sie besteht aus einer kleinen Leistungsstufe, welche mit einem Logik-MOSFET realisiert ist, der die 12V für die Pumpen ein- und ausschaltet. Außerdem beinhaltet die Schaltung einen Begrenzungswiderstand, eine Freilaufdiode und einen Stecker für den Anschluss der Pumpe. Die Schaltung ist für alle zwölf Pumpen umgesetzt.

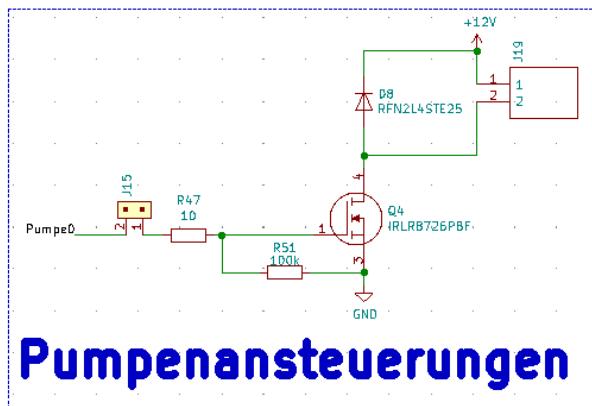


Abbildung 5.12: Schema der Pumpenansteuerung.

#### Funktionsbeschrieb der Schaltung

Die Pumpen werden jeweils direkt über den Mikrocontroller angesteuert. Dabei wird für jede Pumpe ein Digitalpin gemäss Kapitel 5.7 verwendet. Da der Mikrocontroller nicht genügend Ausgangsstrom bietet, um die Pumpen ansteuern zu können, ist eine kleine Leistungsstufe mit einem MOSFET (hier Q4) implementiert gemäss Abbildung 5.12. Es handelt sich bei diesem MOSFET um einen Logik-Level-Mosfet IRLR8726 von Infineon Technologies. Dieser hat den Vorteil, dass er direkt über einen Mikrocontroller angesteuert werden kann und trotz niedrigem Spannungsniveau voll durchsteuern kann. Da das Gate jedes MOSFET's auch eine Kapazität darstellt, wird der Anfangsstrom beim Einschalten des Pin's durch einen 10 Ohm (hier R47) Widerstand begrenzt. Ein weiterer Grund für diesen Widerstand ist, dass manche MOSFET's beim Ein- und Ausschalten, mit der steilen Flanke dazu neigen kurzzeitig zu oszillieren. Dies wird durch diesen Widerstand auch unterdrückt. Zu Messzwecken ist vor diesem Widerstand ein Jumper implementiert. Dies ermöglicht es, die Ansteuerung vom Mikrocontroller abzukoppeln. Da es sich bei dieser Pumpe um einem DC-Motor und somit um eine Induktion handelt, ist parallel zur Pumpe am Stecker eine Freilaufdiode implementiert. Diese schützt den MOSFET vor Spannungsspitzen beim Ausschalten. [14, S.362] [13]

### 5.3.2 Durchflussmessgeräte

Um die Flüssigkeiten wie gewünscht abfüllen zu können, müssen diese sauber dosiert werden können. Zu diesem Zweck wurden Durchflussmessgeräte eingesetzt. Bei diesen Durchflüssmessgeräten handelt es sich um volumetrische Durchflussmessgeräte von Sea, welche bei Durchfluss eine Pulsfolge ausgeben. Diese können dann mittels Elektronik ausgewertet werden. [21]

#### Schema

Das Schema für diese Schaltung ist in Abbildung 5.13 zu sehen. Es beinhaltet lediglich einen Stützkondensator an der Speisung und einen Stecker, an welchem die Durchflussmessgeräte angeschlossen werden können.

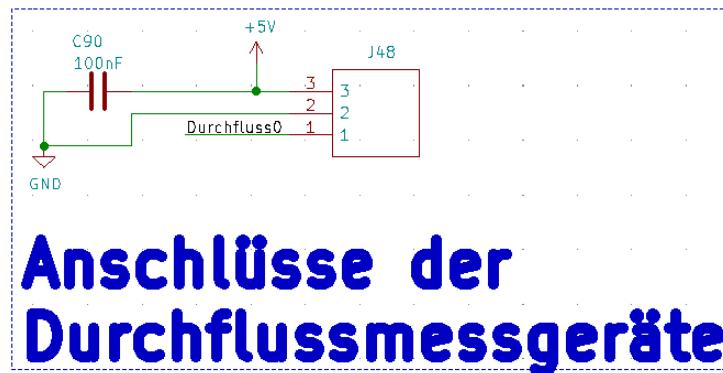


Abbildung 5.13: Schema der Durchflussmessgeräte.

#### Funktionsbeschrieb der Schaltung

Die Durchflussmessgeräte haben drei Anschlüsse. Zum einen ist dies ein Speisungseingang von 5V und eine Groundleitung. Der dritte Pin der Durchflussmessgeräte ist ein Signalpin, welcher bei Durchfluss ein gepulstes Signal mit 5V und 50% Duty-Cycle ausgibt. Da dieses Signal direkt vom Mikrocontroller ausgewertet werden kann, ist keine weitere Elektronik erforderlich. Zur Auswertung der Signale werden gemäss Kapitel 5.7 wiederum Digitalpins als Eingänge verwendet [21].

## 5.4 Programmierschnittstellen

Auf der Leiterplatte des PartyMixers sind der Mikrocontroller und das Wireless-/Bluetooth- modul (Targets), welche programmiert werden müssen. Damit die Software vom Computer (Host) auf die Komponenten geladen werden kann, braucht es eine Programmierschnittstelle. Computerseitig geschieht dies über USB. Die Übermittlung von Daten über die USB-Schnittstelle geschieht über ein differentielles Verfahren und benötigt zwei Kommunikationsleitungen (D+ und D-), welche innerhalb des USB-Kabels verdrillt sind. Das differentielle Verfahren zeichnet sich dadurch aus, dass elektromagnetische Störungen von außerhalb zu einem grossen Teil eliminiert werden können, da sie sich auf beide Leitungen gleich auswirken. Die Targets werden jedoch über eine serielle Schnittstelle programmiert und benötigen zusätzliche Steuerleitungen um in einen Programmiermodus zu kommen. Deswegen wird ein USB-UART-Konverter verwendet, der das USB-Signal in ein UART-Signal wandelt und die entsprechenden Steuerleitungen zur Verfügung stellt.

Die beiden Programmierschnittstellen benötigen unterschiedlich viele Steuerleitungen, da sie sich im Verfahren zum Aufruf des Download-Boot-Modus unterscheiden. Die Verfahren sind vom Hersteller des Targets gegeben und können nicht beeinflusst werden. Die vom Verfahren vorgegebene Signalfolge, welche an den vordefinierten Pins angelegt werden muss, um den Programmier-Modus zu starten, wird auch Handshake genannt. Dieser findet immer statt, bevor ein Programm vom Host auf ein Target geladen wird. Er wird vom Host gesteuert und muss deswegen im Voraus bekannt sein, sodass die gewünschte Signalfolge berücksichtigt werden kann.

### **Handshake Mikrocontroller**

Um Download-Boot-Modus des Mikrocontrollers zu starten, reicht es, die Reset-Leitung per Handshake auf 0V zu ziehen. Dies wird erreicht, indem das Wiring-Protokoll verwendet wird, das auf den verwendeten Bootloader auf dem Mikrocontroller angepasst ist und bei Ausführen des Handshakes die DTR- bzw. Reset-Leitung toggelt. Nachdem der Handshake ausgeführt wurde und das Programmiertool AVRdude die Device-Signatur des Mikrocontrollers (0x1E9801) verifiziert hat, sendet es das kompilierte HEX-File an den Mikrocontroller.

### **Handshake Wireless-/Bluetoothmodul**

Um Download-Boot-Modus des Wireless-/Bluetoothmodul zu starten, sind die Strapping Pins relevant. Sie werden nach einem Neustart des Wireless-/Bluetoothmoduls als Erstes gelesen und bestimmen den Grundzustand des Wireless-/Bluetoothmoduls. Die Tabelle 5.2 zeigt die Strapping-Pins und deren Einfluss auf den Boot-Modus. Aufgrund der defaultmässigen Pull-up und -down Widerstände kann aus dieser interpretiert werden, dass wenn U0TXD, IO2 und IO5 "floating" sind, IO0 den Boot-Modus bestimmt [22, S.12-S.14]. Mittels Handshake wird die Resetleitung getoggelt und die Strapping-Pins so gesetzt, dass der Download-Boot-Modus gestartet wird.

<b>Boot-Mode Konfiguration</b>			
<b>Pin</b>	<b>Default</b>	<b>Boot</b>	<b>Download</b>
IO0	1	1	0
U0TXD	1	1	Don't care
IO2	0	Don't care	0
IO15	1	Don't care	Don't care
IO5	1	1	Don't care

**Tabelle 5.2:** Wenn U0TXD, IO2, IO5 floating sind, bestimmt IO0 den Boot-Modus.

Im Folgenden wird die Hardware für die Programmierschnittstellen beschrieben. Es ist darauf zu achten, dass in Abbildung 5.14 nur das Schema für die Programmierschnittstelle des Wireless-/Bluetoothmoduls zu sehen ist. Vorab: Bei den Leitungen DTR, DSR, RTC, CTS, RXD und TXD handelt es sich um die Kommunikationsleitungen zwischen dem USB-UART-Konverter und dem Target. In Tabelle 5.3 sind die Unterschiede tabellarisch aufgelistet.

Leitung			Konverter	Target
RXD	<==	über Widerstand	====	TX
TXD	====	über Widerstand	==>	RX
IO_0 / Reset	<==	über Transistor	====	DTR
EN	<==	über Transistor	====	RTS
IO_13	<==	über Widerstand	====	RTS
IO_15	<==	über Widerstand	====	CTS

Tabelle 5.3: Verbindung zwischen USB und Wireless-/Bluetoothmodul.

### Schema

Das USB-Kabel wird an der Buchse J2 angeschlossen. Sobald ein USB-Kabel eingesteckt ist, sind die Leiterplatine und der Computer auf gleichem Ground-Potential und die Leitungen D+ und D- mit dem Konverter verbunden. Über das USB-Kabel werden ebenfalls 5V am Spannungsteiler R12 und R13 angelegt, was vom Konverter als bestehende USB-Verbindung interpretiert wird. Da beim Einsticken die Gefahr von Spannungsdifferenzen besteht, wird ein ESD-Schutz benötigt, welcher vom den Dioden D2-4 gewährleistet wird. Die LED's können verwendet werden, um einen Datenfluss zu signalisieren. Die Widerstände in den Kommunikations- und Steuerleitungen schützen die Pins vor Ausgleichsströmen, die entstehen wenn die Pins auf unterschiedlichem Potential liegen.

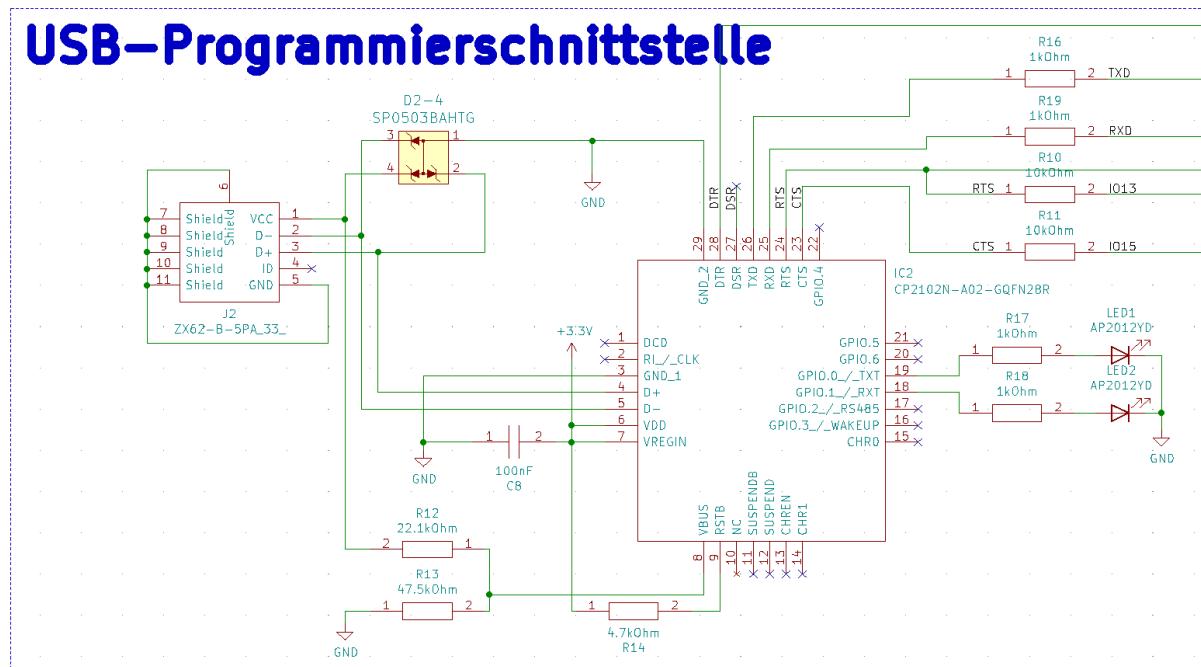


Abbildung 5.14: Schema USB-B.

## Funktionsbeschrieb der Schaltung

Der dazugehörige ESD-Schutz bildet die Diodenschaltung D2-D4. Dabei handelt es sich um ESD-Schutzdiode des Typs SP0503BAHTG. Sie werden ab einer Spannung von 3.3V leitend und reduzieren so die maximale Spannung auf den Eingangspins des Konverters auf 3.3V. Der Konverter hat im Schema die Bezeichnung IC2. Es ist ein CP2102N des Herstellers Silicon-Labs. Die Widerstände R12 und R13 bilden einen Spannungsteiler, welcher an die 5V-Eingangsspannung der Buchse angeschlossen ist. Der Spannungsteiler hat eine 3.3V Zwischenspannung und ist an dem VBUS-Eingang des Konverters angeschlossen. Die Widerstände R16, R19, R10 und R11, welche die Ausgleichsströme zwischen dem USB-UART-Konverter und den Targets begrenzen, sind mit  $1\text{k}\Omega$  dimensioniert. Der Widerstand R14 ist ein Pull-Up Widerstand, welcher den Konverter permanent aktiviert und ist mit  $4.7\text{k}\Omega$  dimensioniert. Die Widerstände R17 und R18 reduzieren den Strom durch die LEDs vom Typ AP2012YD.

## 5.5 Benutzerschnittstellen

Über die Benutzerschnittstellen geschehen die Interaktionen zwischen Mensch und Maschine. Im Gesamten gibt es drei verschiedene Schnittstellen, das Display, das Wireless-/Bluetoothmodul und das RDIF-Modul. Die Hauptinteraktion geschieht über das Display, über das Wireless-/Bluetoothmodul lassen sich mit der Android-Applikation einige Funktionen der Maschine von extern aufgerufen werden, mit dem RFID-Modul kann ein Zugewiesener Cocktail direkt ausgewählt werden. Die zugehörigen Schaltungen werden im Folgenden beschrieben.

### 5.5.1 Display

Der Benutzer bedient den PartyMixer via dem Nextion Touch-Display. Dabei handelt es sich um das Display NX8048T070. Das GUI basiert dabei auf den Funktionen, die in der Software Nextion Editor zur Verfügung gestellt werden. Die Funktionen umfassen:

- Erstellen von Seiten, die auf dem Display angezeigt werden.
- Setzen von Buttons, Slider und Textfeldern auf den Seiten.
- Speichern von Grafiken, die auf den Seiten angezeigt werden.
- Kommunizieren über UART um Aktionen auszulösen.

### Schema

Um das Display mit der Leiterplatine zu verbinden, braucht es einen Stecker und einen Stützkondensator nahe der Pins am Spannungsausgang, um die Versorgungsspannung zu glätten. Auf eine Abbildung wird verzichtet, der Stecker für das Display ist jedoch im Schema auf Seite 7 zu sehen, welches im Anhang Kapitel I.1 eingefügt ist.

## Funktionsbeschrieb der Schaltung

Die Funktionen werden durch das Stützen der Versorgungsspannung mit dem Kondensator und der Anbindung des Displays an die Leiterplatine gegeben.

### 5.5.2 Wireless-/Bluetoothmodul

Das Wireless-/Bluetoothmodul ist in die Maschine eingebettet, damit der drahtlose Zugriff via Android-Applikation gewährleistet ist. Es handelt sich bei dem Bauteil um das ESP32 vom Hersteller Espressif. Da sich das Bauteil in einer Box befindet, ist es vorteilhaft ein ESP32-Typ zu verwenden, der die Möglichkeit bietet, eine abgesetzte Antenne anzuschliessen, im Falle das Signal zu schwach ist. Dazu eignet sich das Espressif ESP32-32U.

Über den entfernten Zugriff werden folgende Funktionen zur Verfügung gestellt:

- Applikation per Bluetooth mit dem PartyMixer verbinden.
- Erstellen von Cocktails.
- Zuweisung eines Cocktails zu einem Tag.
- Informationen zur Maschine lesen.

### Schema Wireless-/Bluetoothmodul

Empfängt das Wireless-/Bluetoothmodul Daten über Bluetooth, so werden diese interpretiert und aufbereitet über die zweite serielle Schnittstelle an den Mikrocontroller weitergeleitet. Dieser Vorgang wird mittels Firmware gesteuert.

In Abbildung 5.15 ist das Schema des Wireless-/Bluetoothmoduls zu sehen. Es beinhaltet Stützkondensatoren, die die Versorgungsspannung glätten sowie die Pull-up- und Pull-down-Widerstände, welche dazu da sind, einen eindeutigen Zustand an den Strapping-Pins zu behalten, während das Wireless-/Bluetoothmodul bootet.

In Abbildung 5.16 ist eine Backup-Lösung zu sehen, wo ein ESP32-Devkit in Header-Pins eingesteckt werden kann. Da die Backup-Lösung nicht gebraucht wird, können die Header-Pins in Abbildung 5.16 für Korrekturen verwendet werden und dienen so als Backup-Pins für andere Systeme.

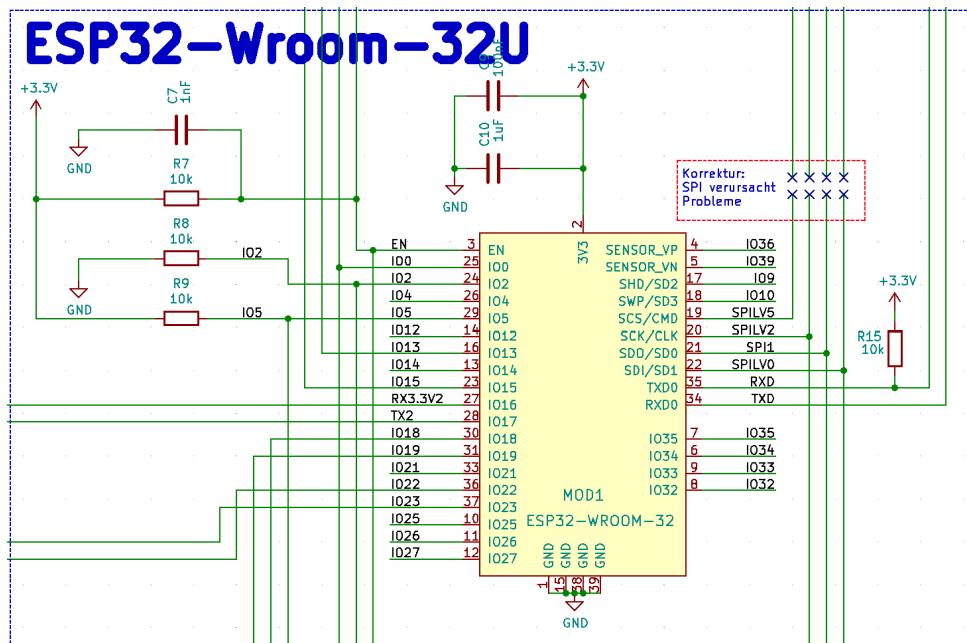
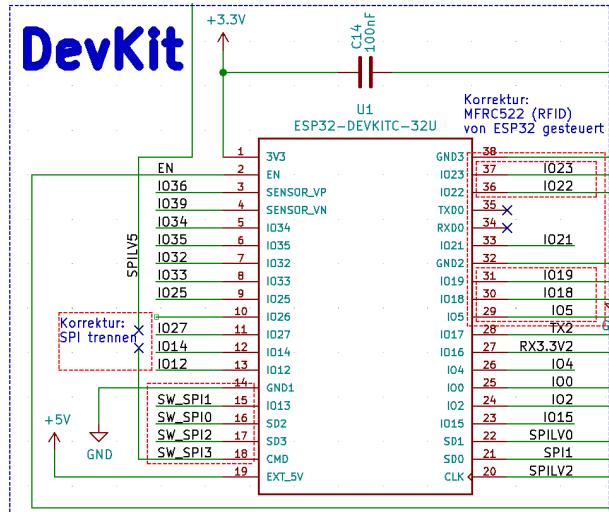


Abbildung 5.15: Schema ESP32-Wroom-32U.



**Abbildung 5.16:** Header-Pins des DevKits werden für das RFID-Modul und TMC4671 benutzt.

Die Korrekturen an der Hardware wurden Rot eingezzeichnet. Die SPI-Verbindung vom Mikrocontroller zum Wireless-/Bluetoothmodul, welche in Abbildung 5.15 zu sehen ist, ist aufgetrennt, da die Leitungen den Bootvorgang des Wireless-/Bluetoothmoduls stören. Aufgrund der funktionierenden Schaltung des ESP32-Wroom und den Störungen, die das RFID-Modul auf dem SPI-Bus verursachte, ist das RFID-Modul am ESP32 über einen separaten SPI-Bus eingesteckt. Deswegen wird die Kommunikation mit dem RFID-Modul vom ESP32 übernommen. Aus dem selben Grund werden die Header-Pins dazu verwendet, eine steckbare Schnittstelle zwischen Mikrocontroller und FOC-Treiber zu bilden. Dazu sind die Leitungen zum ESP32 aufgetrennt und die Pins mit dem Mikrocontroller verbunden.

### Funktionsbeschrieb der Schaltung (Wireless-/Bluetoothmodul)

Das Wireless-/Bluetoothmodul ESP32 ist mit MOD1 beschriftet. Die Kondensatoren C9 und C10 dienen zur Glättung der Eingangsspannung. Über den EN-Pin wird das Modul ein- und ausgeschaltet (active high), der Pull-Up-Widerstand R7 zieht diese Leitung defaultmäßig auf 5V. Die Widerstände R8, R9, R10 und R11 sind an die Strapping-Pins angeschlossen.

Aus Tabelle 5.2 ist ersichtlich, dass U0TXD für den normalen Boot-Modus auf HIGH sein muss, weswegen der Widerstand R15 platziert wurde. Mit dem Kondensator C7 wird sichergestellt, dass nach einem Reset automatisch der Download-Boot-Modus gestartet werden kann.

### Schema (Automatische Boot-Logik)

Die automatischen Bootlogik, welche in Abbildung 5.17 zu sehen ist, steuert die Pins EN und IO0 so an, dass das Wireless-/Bluetoothmodul beim Hochladen eines Programms automatisch neu gestartet wird und die Spannung an den Pins so anliegt, dass der Download-Boot-Modus gestartet wird.

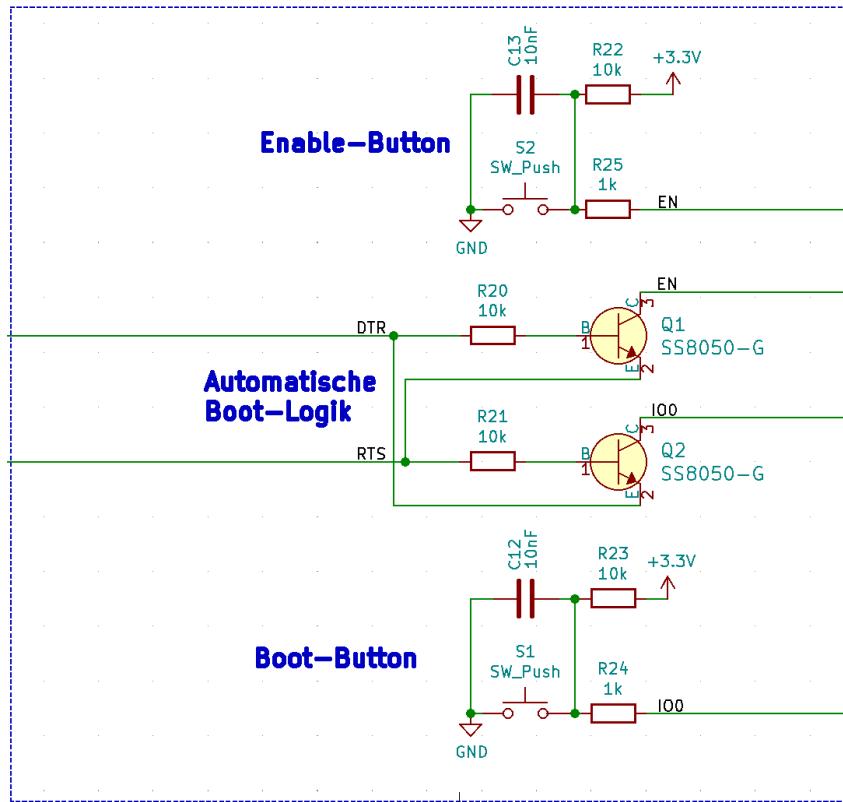


Abbildung 5.17: Schema automatische Bootlogik.

### Funktionsbeschrieb der Schaltung (Automatische Bootlogik)

Für die Beschaltung der automatischen Boot-Logik benötigt es die Leitungen DTR und RTS, welche vom USB-UART-Konverter gesteuert werden und EN und IO0 als Outputs auf das Wireless-/Bluetoothmodul. Die Buttons sind dazu da, das ESP manuell in den Download-Boot-Modus zu setzen. Die Widerstände R20 und R21 sind Vorwiderstände an der Basis der Transistoren Q1 und Q2. R22 und R23 sind Pull-Up-Widerstände für die EN- und IO0-Leitung. Die Kondensatoren C13 und C12 dienen dem Entprellen der Buttons. Die Widerstände R25 und R24 begrenzen den Strom bei Drücken der Buttons S1 oder S2.

### 5.5.3 RFID

Eine Funktion der Maschine ist, dass der User ohne Suchen sein Lieblingsgetränk zubereiten lassen kann. Dafür wird auf ein System zurückgegriffen, welches beispielweise auch für Zutrittskontrollen verwendet wird, RFID<sup>2</sup>.

Ein Eingrenzungskriterium für das Bauteil war, dass es für den ausgewählten IC eine bestehende Library gibt, welche mit Arduino oder besser C kompatibel ist und dass der IC auf einem Breakout-Board erhältlich ist. Der Chip, welcher diese Anforderungen erfüllt, ist der **Mifare MFRC522**. Das Breakout-Board ist an der Verschalung der Maschine angeschraubt. Das Modul arbeitet auf 13.56MHz und gilt somit als kurzwelliges System (HF). [23]

Das RFID-Modul ist der Reader im RFID-System. Er kann Daten vom RFID-Tag lesen. Dazu erzeugt er ein hochfrequentes, elektromagnetisches Wechselsignal mit einer Frequenz von

<sup>2</sup>Radio Frequency IDentification

13.56MHz. Das Wechselfeld induziert beim Empfänger eine Spannung, welche als Energieversorgung dient. Durch Kurzschliessen der Tag-Antenne wird ein Teil der Energie des vom Reader ausgehenden Wechselfeldes verbraucht. Diese Energiedifferenz kann der Reader detektieren.

### Schema

In Bild 5.18 ist der Schaltungsaufbau des RFID-Moduls zu sehen. Darin ersichtlich sind folgende Teilbereiche:

- MFRC522 RFID IC
- Antenne
- Anpassnetzwerk für Antenne
- Kommunikationsschnittstellen

Die Schaltung ist auf dem Print mitgelayoutet. Allerdings ist das Breakout-Board die bessere Lösung, da die Bestimmung des Anpassnetzwerkes noch nicht korrekt ist und so kein Antennenkabel vom Stecker nach aussen geführt werden muss, da das verwendete Breakout-Board extern angebracht wird und die Antenne schon vorhanden ist.

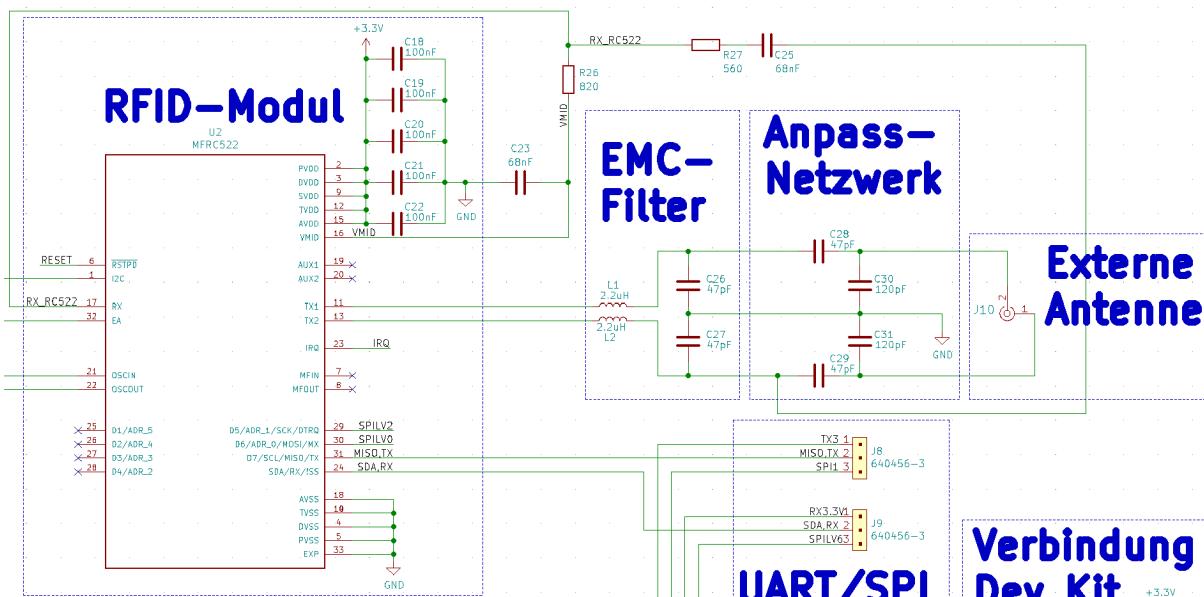


Abbildung 5.18: Schema des RFID Sender/Empfänger

### Funktionsbeschrieb der Schaltung

Das RFID-Modul steuert den Ablauf, damit die Datenübertragung stattfinden kann. Er verbindet die RFID-Antenne mit dem PartyMixer und schreibt bzw. liest die Informationen auf das bzw. aus dem Trägersignal. Die Herausforderung besteht im Design des Anpassnetzwerkes, welches an das IC und die Antenne angepasst werden muss. Wie die Bauteile dimensioniert werden, ist in der Annotation Note 1445 von NXP Semiconductors beschrieben. Da das Breakout-Board zum Einsatz kommt, wird nicht weiter auf die Dimensionierung der Bauteile eingegangen.

Die Kommunikationsanschlüsse sind so gestaltet, dass im Falle einer zukünftigen Revision, die Kommunikation zwischen UART und SPI gewählt werden kann. Das Breakout-Board kommuniziert nur über SPI.

## 5.6 Beleuchtung

Damit die Maschine von weitem erkennbar ist und auch Eindruck schindet, eignet sich ein LED-Band. Am Besten, wenn es noch verschiedenfarbig ist. Der Aufbau eines LED-Bandes ähnelt dann dem in Abbildung 5.19 gezeigten Schaltung. Der Streifen ist Rot eingerahmmt, die LED's und Widerstände befinden sich auf dem Band und die zu sehenden Fähnchen für R, G, B und W führen über die Leistungs-MOSFETS. Die MOSFETs wiederum werden vom Mikrocontroller angesteuert.

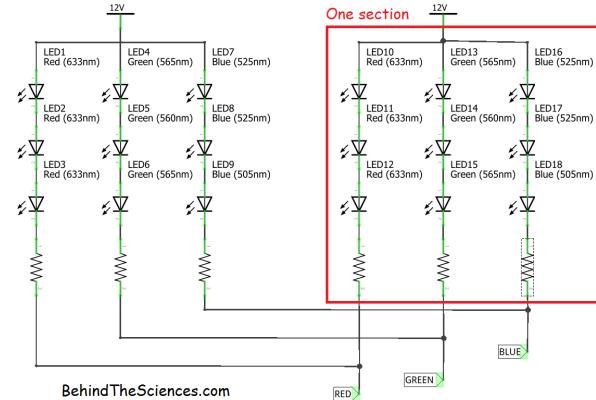


Abbildung 5.19: LED Beispiel. [24]

## Schema

Bei Licht handelt es sich um elektromagnetische Wellen, welche im sichtbaren Wellenlängen-Bereich liegen. Dabei gibt es vier Hauptfarben: Rot, Grün, Blau und Weiss. Zwar kann Weiss auch aus einer Kombination aller drei Farben erstellt werden, es kommt aber besser mit einer separaten Diode. Das resultierende Licht des Bandes ist eine Überlagerung der Wellenlängen. Diese Überlagerung kann vom Mikrocontroller über den Duty-Cycle eines PWM-Signals gesteuert werden. So lassen sich mit dem Licht aus den Grundfarben praktisch alle Farben mischen. Abbildung 5.20 zeigt den Schaltungsaufbau der LED-Steuerung mit den MOSFETs und zugehörigen passiven Bauteilen.

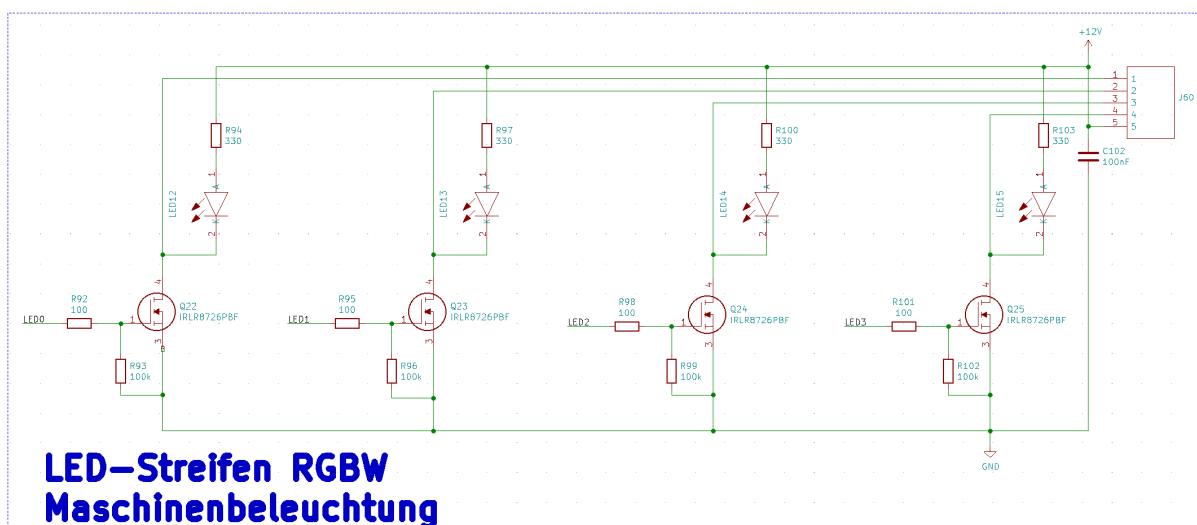


Abbildung 5.20: Schema der LED-Ansteuerung

## Funktionsbeschrieb der Schaltung

Damit die LED's angesteuert werden können, braucht es ein Bauteil welches mit einer 5V-Eingangssignal 12V schalten können. Dazu wird der selbe MOSFET Q22 bis Q25 verwendet wie schon bei den Pumpen in Kapitel 5.3.1. Über die Widerstände an den Gates wird der Strom zum Schutz des Gates begrenzt. Ausserdem verhindern sie ein allfälliges Oszillieren. Die Leitungen führen direkt auf den Klemmblock für die LED-Streifen. Die parallel geschalteten LEDs LED12 bis LED15 sind zum Debuggen ohne LED-Streifen.

## 5.7 Mikrocontroller

Der Mikrocontroller ist das Gehirn der Maschine. Er muss in der Lage sein mit allen Komponenten auf der Platine zu kommunizieren, die Tasks zu verarbeiten und abzuhandeln. Der ATMega2560 übernimmt diese Aufgabe.

### Schema

Damit der Mikrocontroller funktionieren kann, sind folgende Funktionen implementiert:

- Erzeugung des System-Clocks mit 16Mhz mittels Quarz.
- In-System-Program-Connector (ISP) für die Grundkonfiguration des Mikrocontrollers.
- Abtrennung des ISP von der SPI-Leitung.
- Reset-Button für Neustart des Mikrocontrollers.
- Kontroll-Led für Spannung am Mikrocontroller.

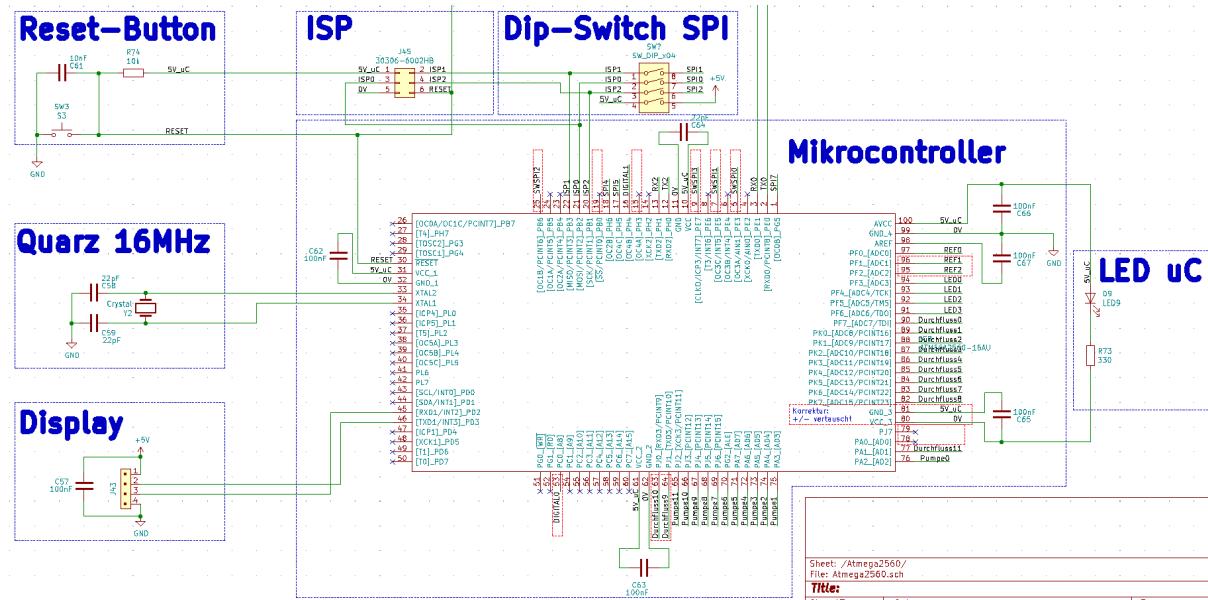


Abbildung 5.21: Schema Mikrocontroller

Die Korrekturen sind Rot im Schema in Abbildung 5.21 eingezeichnet. Dazu gehören zum Ersten die zusätzlichen Software-SPI Pins für den FOC-Treiber, welche auf die Header-Pins des Wireless-/Bluetoothmodul DevKits führen und zum Zweiten das Umlegen der DIGITAL0-Leitung, welche die Enable-Leitung des Gate-Treibers darstellt. Weiter musste die Spannungsversorgung an einer Seite der Mikrocontrollers getauscht werden und zwei Leitungen für die Flüssigkeitsbeförderung umgelegt werden.

## Funktionsbeschrieb der Schaltung

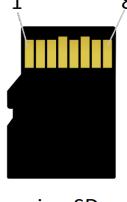
Das Schema besteht aus dem Mikrocontroller IC8 mit den fünf Stützkondensatoren C62, bis C66 an den Spannungseingängen. Mit dem Full Swing Oscillator Crystal Y2 und den Kondensatoren C58 und C59 wird eine Frequenz von 16MHz generiert. Der Reset-Button SW3 dient dazu, den Mikrocontroller manuell neu zu starten. Der dazugehörige Kondensator C81 entprellt den Button und der Widerstand R74 zieht die Leitung mittels Pull-up auf VCC. Mittels dem ISP-Stecker kann über einen Programmer, z.B AVR MKII, auf den Mikrocontroller zugegriffen werden. Der DIP-Switch ist dazu da, die SPI-Leitungen von der ISP-Schnittstelle zu trennen. Die LED D9 gibt Auskunft ob Spannung am Mikrocontroller anliegt.

## 5.8 SD-Karte

Die SD-Karte speichert die Getränke, Zutaten und Maschinenzustände, damit diese bei einem Neustart wieder aufgerufen werden können und die Maschine auf den letzten Stand gesetzt wird. Nebst dem Erhalten der Informationen ist die SD-Karte wichtig für den Betrieb der Maschine. Füllt sich der Speicher des Mikrocontrollers zu sehr, führt dies zu einem Absturz.

Abbildung 5.22 zeigt die Pinbelegung einer mikroSD-Karte. Es ist erkennbar, dass sie über SPI kommuniziert. Für den Anschluss wird nur ein mikroSD-Sockel benötigt. Ein Level-Shifter wandelt die 5V-Spannungssignale des Mikrocontrollers mit in 3,3V-Spannungssignale für die SD-Karte.

microSD Pinout		
Pin	Pin Name	Function
1	NC	Not used
2	CS	Chip Select
3	MOSI	Master-out, Slave-in
4	VDD	Positive Supply
5	CLOCK	Serial Clock
6	GND	Ground
7	MISO	Master-in, Slave-out
8	NC	Not used



microSD

Abbildung 5.22: Pinout des mikroSD-Sockels. [25]

Die sdKarte ist mit FAT32 formatiert. FAT32 steht für File Allocation Table mit 32Bit Datenbreite. Es ist ein von Microsoft entwickeltes System, dessen Wurzeln bis ins Jahr 1977 zurückreichen und heute noch der Industriestandard unter den Dateisystemen ist. [26]

Der Speicherbereich einer FAT32-formatierten Partition besteht aus fünf Bereichen. [27]

1. Master Boot Record (LBA Sektor 0 des Laufwerks, Startprogramm)
2. Volume Boot Record, Partition Boot Sektor (LBA Sektor 0 der Partition, Bootcode, Beginn der Daten, Fehlermeldungen)
3. File Allocation Table (i.d.R 2 mal hintereinander vorhanden, direkt nach dem PBR, Organisation der belegten oder freien Cluster oder Partitionen)
4. Directory Table (mit den Ordner und File-Einträgen, Wurzelverzeichnis)
5. Datenbereich (Fileinhalt)

## Schema

In Abbildung 5.23 ist das Schema der SD-Karte zu sehen. Darin erkennbar ist der mikroSD-Adapter J45 mit einem Stützkondensator C68 am Spannungseingang.

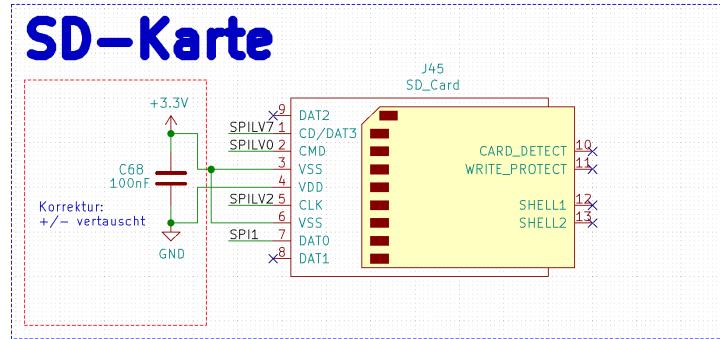


Abbildung 5.23: Schema SD-Karte.

## Funktionsbeschrieb der Schaltung

In der Schaltung ist erkennbar, dass die SPI-Leitungen an die Pins des Adapters führen, genauso die Spannungsleitungen. Da die SD-Karte stets vom Mikrocontroller angesteuert wird, gibt es keine weiteren erwähnenswerte Funktionen zu beschreiben.

## 6 Inbetriebnahme

Ein essentieller Teil der Entwicklung ist die Inbetriebnahme. Dabei wurden die Teilsysteme nacheinander in Betrieb genommen. Die einzelnen Systeme wurden dann in ihren wichtigsten Funktionen geprüft und für den Einsatz vorbereitet. In den folgenden Kapiteln wurden die einzelnen Teilsysteme des PartyMixers in Betrieb genommen.

### 6.1 Speisungen

Um die Speisungen in Betrieb nehmen zu können, ist gemäss Kapitel 5.1 ein Jumper in die Schaltung implementiert worden, welcher die Speisungen von der übrigen Hardware trennt. Ausserdem sind Status-LED's verbaut, welche den korrekten Betrieb anzeigen sollen. Da es sich bei der 48V-Speisung um ein fertiges Netzteil handelt, musste dieses nicht grossartig in Betrieb genommen werden. Die einzige Einstellung, welche vorgenommen werden musste, ist die Feinjustierung der Ausgangsspannung mittels Drehregler am Netzteil.

#### 6.1.1 12V Speisung

Die 12V Speisung wurde für die Inbetriebnahme vom übrigen System abgekoppelt. Da diese aus der 48V-Speisung generiert wird, wurde lediglich diese vorgeschaltet. Beim Einschalten des Netzteils war der korrekte Betrieb am Status-LED sichtbar. Eine Messung mit dem Voltmeter bestätigte dies. Um allfällige Rippel sehen zu können, wurde noch mit dem Oszilloskop gemessen.

#### 6.1.2 5V Speisung

Simultan zur 12V-Speisung wurde die 5V-Speisung in Betrieb genommen. Auch bei dieser Speisung wurde lediglich das Netzteil vorgeschaltet und das restliche System abgeschnitten. Auch hierbei leuchtete das Status-LED sofort auf. Um zu verifizieren ob die richtige Spannung anliegt, wurde dies auch hier mit einem Voltmeter geprüft. Um allfällige Rippel sehen zu können, wurde noch mit dem Oszilloskop gemessen.

#### 6.1.3 3.3V Speisung

Die 3.3V-Speisung wurde als letzte Speisung in Betrieb genommen, da diese von der 5V-Speisung gemäss Kapitel 5.1.4 abhängig ist. Der Linearregler hat am Ausgang auch eine Status-LED zur Verifizierung verbaut. Auch diese leuchtete nach dem Start des Netzteils und dem Zuschalten der 5V-Speisung wie gewünscht auf. Das Voltmeter bestätigte anschliessend die Richtigkeit der Ausgangsspannung.

## 6.2 Programmierschnittstellen

Die Programmierschnittstellen werden jeweils mithilfe eines USB-UART-Controller gesteuert. Im Folgenden wird die Inbetriebnahme dieser Schnittstelle für den Mikrocontroller und das Wireless-/Bluetoothmodul beschrieben. Beide werden über eine USB-B-Buchse realisiert.

Für die Inbetriebnahme der USB-B-Schnittstelle wurde vorerst geprüft, ob der USB-UART-Converter über die USB-Buchse und Kabel vom Computer erkannt wird. Danach wurde geprüft, wie der Handshake beim Programmieren der Komponenten funktioniert. Die Inbetriebnahme der Programmierung geschieht parallel zur Inbetriebnahme der Programmierschnittstelle, da die Programme zum Schreiben, Kompilieren und Hochladen des Codes schon vor dem Prüfen der USB-B-Schnittstelle installiert sein müssen. Da ein erfolgreicher Handshake als Voraussetzung gilt, dass überhaupt ein Programm hochgeladen werden kann, wurde entschieden, dass der Handshake zur Programmierschnittstelle gehört. Die Programmierung der Targets (Mikrocontroller und Wireless-/Bluetoothmodul) wird in den Kapitel 6.4.2 und Kapitel 6.3 beschrieben.

Für die Inbetriebnahme der Programmierschnittstellen wurde mit der entsprechenden Programmierumgebung ein HEX-File mit dem darin enthaltenen Code übertragen.

Vorgehen:

1. Platine mit Spannung versorgen und Computer mit zwei USB-Kabeln an die USB-B-Buchsen anschliessen.  
[Systemeinstellungen→Geräte-Manager](#)  
Dort sind die Devices sichtbar unter dem Name: *Silicon Labs CP210x USB to UART Bridge*.
2. Überprüfen der Handshakes, welche definiert sind in den Tools zum Hochladen der Software.
  - (a) Mikrocontroller

Im Anhang Kapitel A.1.2 ist der Handshake und die Übertragung der ersten Bytes zu sehen. Der Handshake wird in einer weiteren Abbildung des selben Kapitels genauer aufgezeigt. Darin ist erkennbar, dass sobald die DTR-Leitung auf GND gezogen wird und die Spannung über dem Reset-Pin aufgrund des Kondensators zwischen DTR und Reset nur kurzzeitig auf 0V fällt. Dies reicht jedoch aus, dass der Mikrocontroller in den Boot-Modus fällt. Nach ca. 50ms beginnt die Datenübertragung vom Computer in den Flash-Speicher des Mikrocontrollers.

- (b) Wireless-/Bluetoothmodul

Im Anhang Kapitel A.1.3 wurden die ersten 100ms des Hochladen eines Programms auf das Wireless-/Bluetoothmodul gemessen. Gesucht wird nach einem gleichzeitigen Flankenwechsel der RTS-Leitung von 0 auf 1 und der DTR-Leitung von 1 nach 0. Eine weitere Abbildung im selben Kapitel zeigt eine genauere Aufnahme zum Zeitpunkt des Flankenwechsels. Bei genauerem Betrachten der Messungen fällt auf: Entgegen der Erwartung ist der Flankenwechsel der beiden Leitungen leicht verzögert (um ca. 80 µs). Auch das Signal des EN-Pins erreicht wesentlich schneller einen HIGH-Zustand als erwartet. Denn gemäss der Theorie müssen die Pins EN und IO0 zum gleichen Zeitpunkt auf LOW sein, um in den Download-Boot-Modus zu kommen, und trotz der Abweichung der Praxis zur Theorie funktioniert die Übertragung des Codes. Eine kleine Erläuterung zu dieser Abweichung ist im Anhang Kapitel A.1.3 zu finden.

Schlussfolgerung davon ist, dass die Strapping-Pins wahrscheinlich erst nach einer gewissen Zeit gelesen werden und so die angelegten Zustände erst zu einem späteren Zeitpunkt von Bedeutung sind.

Über LED's kann angezeigt werden, ob eine Übertragung zwischen den Komponenten stattfindet. Pro Schnittstelle jeweils ein LED für RX und ein LED für TX. Diese funktionieren leider nicht. Dazu müsste wahrscheinlich ein Register des USB-UART-Converters gesetzt werden oder ggf. die Vorwiderstände der LED's angepasst werden.

### 6.3 Mikrocontroller

Als Programmierumgebung wird aufgrund des AVR-Chips die Software Atmel Studio 7.0 ausgewählt. Das Hochladen des Anwenderprogramms auf den Mikrocontroller kann direkt aus dieser Umgebung geschehen. Atmel Studio kompiliert den geschriebenen C-Code in Maschinencode und schreibt diesen in ein HEX-File. Das HEX-File wird dann von einem Programmiertool namens AVRdude hochgeladen.[28]

Damit der Mikrocontroller sich mit avrdude programmieren lässt, müssen einige Grundeinstellungen vorgenommen werden. Dazu gehören ein einem ersten Schritt das Setzen der Fuse-Bits, in einem zweiten Schritt das Hochladen des Bootloaders und in einem dritten Schritt das Setzen der Lock-Bits. Dies geschieht über die ISP schnittstelle. Eine detaillierte Beschreibung, wie diese gesetzt wurden, ist im Anhang Kapitel B.1.1 zu finden. Aus der Beschreibung folgt für die Fuse- und Lock-Bits die Einstellungen gemäss Tabelle 6.1.

Extended	High	Low	Lock
0xFF	0xD0	0xF7	0xCF

**Tabelle 6.1:** Tabelle Fuse- und Lock-Bits.

Das Setzen der Fuse- und Lock-Bits sowie das Brennen des Bootloaders kann mit einem AVR MKII Programmer in Atmel Studio gemacht werden. Alternativ gibt es einen Weg, den USB-Treiber (ATMega16U2) eines Arduino Uno mit einer entsprechenden Firmware zu laden, sodass dieser als Programmer verwendet werden kann [29].

#### 6.3.1 AVRdude in Atmel Studio einbinden

Von <http://savannah.gnu.org/> kann eine Datei heruntergeladen welche [avrdude-6.3-mingw32.zip](#) heisst [30]. Der gleichnamige Ordner wird im Ordner [C:\Tools](#) gespeichert.

In AtmelStudio den Reiter ["Tools→External Tools"](#) ausgewählen und ein neues Tool hinzufügen. Im Falle des ATMega2560 werden die Commands gemäss Tabelle 6.2 eingegeben:

Title	:	PartyMixer
Command	:	C:\Tools\avrdude-6.1-mingw32\avrdude.exe
Arguments:	:	-D -P <b>COMx</b> -p ATMEGA2560 -c wiring -b 115200 -U flash:w:\${TargetDir}\${TargetName}.hex:i

**Tabelle 6.2:** AVRdude Commands [31]

Der entsprechende **COMx**-Port des zu flashenden Gerätes (Silicon Labs CP210x USB to UART Bridge) muss mit dem Geräte-Manager ermittelt werden.

Nach der Grundkonfiguration des Mikrocontrollers und dem Einbinden von avrdude in Atmel-Studio wird erwartet, dass der Mikrocontroller programmiert werden kann. Der Erfolg zeigt sich im Output-Fenster von AtmelStudio, wenn das Programm geschrieben und gelesen wird.

Folgende Schritte wurden befolgt:

1. Als Erstes wurden die Fuse-Bits gesetzt. Dies geschah über den Reiter:

[AtmelStudio → Tools → Device programming → Fuses](#)

Es wurde darauf geachtet, dass der AVR mkII ausgewählt wurde und der Gerätecode des ATMega2560 ausgelesen werden konnte.

2. Als Zweites wurde der Bootloader installiert. Dies geschah unter:

[AtmelStudio → Tools → Device programming → Memory](#)

Hier wird ein stk500v2-BL verwendet, welcher im folgenden Ordner zu finden ist:

[Software → Atmega → 6\\_HEX-Files → 1\\_Bootloader\\_STK500V2](#)

3. Als Drittens wurden die Lock-Bits gesetzt unter:

[AtmelStudio → Tools → Device programming → Lock-Bits](#)

Diese sollten nicht mehr geändert werden. Bei jedem Brennen des BL wieder zu setzen.

4. Mikrocontroller mit der kompilierten Software (Mikrocontroller.HEX) programmieren.

[AtmelStudio → Tools → PartyMixer](#)

Ergebnis: AtmelStudio zeigt, dass das Programm hochgeladen wird und der Mikrocontroller startet das Programm.

Befindet sich ein Bootloader im Flash-Speicher, dauert es nach einem Reset zwei Sekunden, bis der Programmcode gestartet wird. Während dieser Zeit wartet der Bootloader auf einkommende Daten, welche auf den Flash-Speicher geschrieben werden sollen. Falls keine Daten kommen, wird das Anwenderprogramm gestartet (sofern ein Programm hochgeladen wurde).

Ergänzende Tabellen aus dem Datenblatt des Mikrocontrollers und Screenshots aus der Programmierumgebung, welche mit dem Setzen der Fuse- und Lock-Bits oder Programmierung des Mikrocontrollers zusammenhängen, sind im Anhang Kapitel B angefügt.

## 6.4 Benutzerschnittstellen

Im Folgenden wird die Inbetriebnahme der Benutzerschnittstellen erklärt. Dazu gehören das Display, das Wireless-/Bluetoothmodul und die RFID-Schnittstelle.

### 6.4.1 Touch-Display

Für die Inbetriebnahme des Displays wird auf den Fachbericht 5 verwiesen. Ergebnis: Über die UART-Schnittstelle können die Button- und Page-IDs bei Klick auf das Display gesendet werden. Der Mikrocontroller empfängt und verarbeitet die ID und löst einen entsprechenden Event aus. Bei der Inbetriebnahme im Projekt 5 wurde ein neues Bild auf dem Display gesetzt.

### 6.4.2 Wireless-/Bluetoothmodul

Die Inbetriebnahme des Wireless-/Bluetoothmodul geschieht mit der Programmierumgebung Arduino IDE. Damit das Wireless-/Bluetoothmodul programmiert werden kann, muss es in der Arduino IDE bekannt gemacht werden. Das Programmer-Tool *esptool.py* ist von espressif und muss ebenfalls in die IDE gelinkt werden.

Im Folgenden wird beschrieben wie vorgegangen wird, damit sich das Wireless-/Bluetoothmodul programmieren lässt. Es wird ein Code hochgeladen mit dem sich das ESP als Webhost im vorgegebenen Netz anmeldet. Es wurde getestet, ob ein Klicken auf ein Button im Explorer ein Event auslöst und ob die Kommunikation zwischen Mikrocontroller und Wireless-/Bluetoothmodul funktioniert.

Folgende Schritte wurden befolgt:

1. Benötigte ESP32-Bibliotheken vom *espressif*-Github-Account runterladen.[32]  
[Github search: espressif/arduino-esp32](#) [32]
2. Speichern der heruntergeladenen Daten unter:  
[.../Dokumente/Arduino/hardware/espressif/esp32/ \(Inhalt: cores, docs, libraries, ...\)](#)  
 Danach sollte das Wireless-/Bluetoothmodul in der Arduino-IDE erscheinen (im sketchbook). Dort kann nun das ESP32 Dev Module angewählt werden.  
[Werkzeuge →Board](#)
3. Unter dem selben Reiter können noch weitere Einstellungen getätigt werden.  
[Arduino IDE →Werkzeuge →...](#)

Upload Speed	:	921600
CPU Frequency	:	240MHz
Flash Frequency	:	80MHz
Flash Mode	:	QIO
Flash Size	:	4MB (32Mb)
Partition Scheme	:	Default 4MB wit spifss
Core Debug Level	:	none
PSRAM	:	disabled
Port	:	<b>COMx</b>

 Wobei der Port **COMx** im Geräte-Manager ermittelt werden muss. Das ESP32 ist jetzt computerseitig flashbar.
4. Testprogramm aus dem Blog-Post von Randomnerdtutorials.com herunterladen, leicht modifizieren und hochladen. [33]  
[Arduino IDE →Verify and Upload](#)  
 Für die Inbetriebnahme wurde das Testprogramm so modifiziert, dass das Wireless-/ Bluetoothmodul gleich getestet werden kann wie das Touch-Display. Dies bedeutet, dass das Wireless-/Bluetoothmodul eine Page-ID, eine Button-ID und das Abschlusszeichen 0xFF 0xFF 0xFF sendet. Der Unterschied ist, dass das Wireless-/Bluetoothmodul über einen anderen UART-Port des Mikrocontrollers kommuniziert. Die Testsoftware ist bei den anderen Firmwares auf dem USB-Stick zu finden.

5. Die "Debug Kommunikation" über den ersten Seriellen Port des Wireless-/ Bluetoothmoduls testen.

[Arduino IDE →Tools →Serial Monitor](#)

Im Testprogramm wird hier angezeigt, wenn ein neuer Client die IP-Adresse aufruft und wenn im Internetexplorer ein Button gedrückt wird. Erste Versuche nach dem Hochladen

waren erfolgreich. Das Wireless-/ Bluetoothmodul hat eine IP-Adresse zugeordnet bekommen.

Ergebnis: Der Klick auf das Display bewirkt, dass eine Kommunikation zwischen Wireless-/Bluetoothmodul und Mikrocontroller stattfindet. Der Test zeigt, dass das Drücken auf den Button im Internetexplorer die gleiche Aktion auslöst, wie das Drücken auf das Touch-Display. Es werden Bilder neu gesetzt und Texte geändert. Die Implementierung des EPS ist folglich erfolgreich. Im Anhang Kapitel C.1.1 sind ergänzende Screenshots aus der Arduino IDE zu sehen.

#### 6.4.3 RFID

Beim RFID-Leser handelt es sich wie in Kapitel 5.5.3 beschrieben um das MFRC522 Evaluierungsboard (Breakout-Board). Dieses kommuniziert über SPI mit einem gewünschten Controller. Die Pinbelegung dazu ist in Abbildung 6.1 zu sehen. Der grosse Vorteil dieses Evaluierungsboard ist es, dass es dazu eine Arduino Library gibt mit vielen Beispielen. Wie schon in Kapitel 4 beschrieben, wurde das RFID-Modul nicht an den Mikrocontroller angeschlossen, sondern an das Wireless-/Bluetoothmodul, welches dann wiederum mit dem Mikrocontroller kommuniziert. Um den Leser in Betrieb nehmen zu können, wurde dieser wie folgt angeschlossen:

- Vcc: An 3.3V vom Wireless-/Bluetoothmodul
- GND: An GND vom Wireless-/Bluetoothmodul
- SDA: An IO5 vom Wireless-/Bluetoothmodul
- SCK: An IO18 vom Wireless-/Bluetoothmodul
- MISO: AN IO19 vom Wireless-/Bluetoothmodul
- Reset: AN IO22 vom Wireless-/Bluetoothmodul
- MOSI: AN IO23 vom Wireless-/Bluetoothmodul

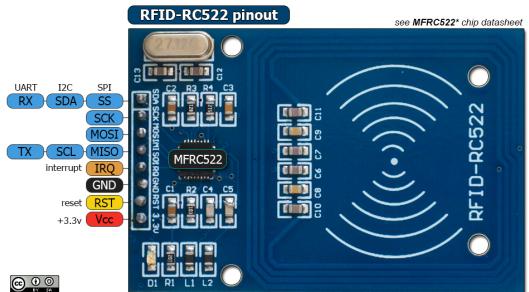


Abbildung 6.1: Anschauungsbild MFRC522 Evaluierungsboard [34]

Diese Pinbelegung ist von der Arduino-Library gegeben und wird auch so bei der Cocktailmaschine eingesetzt, da es keinen Grund gibt diese direkt in der Library zu ändern. Lediglich die beiden Pin's «SDA» und «Reset» können flexibel festgelegt werden.

Um das Lesegerät in Betrieb nehmen zu können, wurden nach dem Anschliessen des Wireless-/Bluetoothmodul an den Computer folgende Schritte unternommen:

1. Einbinden der MFRC522 Library unter:  
[Werkzeuge → Bibliotheken verwalten](#)
2. Öffnen des Beispiels DunmInfo unter:  
[Datei → Beispiele → MFRC522 → DumpInfo](#)

3. Einstellen des Reset und des SDA (SS) Pin's in den defines des Codes
4. Einbinden des ESP32-Boards unter:  
[Werkzeuge →Board →Boardverwalter →esp32](#)
5. Einstellen des verwendeten Boards unter:  
[Werkzeuge →Board →ESP32 Arduino →ESP32 Dev Module](#)
6. Einstellen des richtigen COM-Ports unter:  
[Werkzeuge →Port](#) (Kann im Geräte-Manager des Computers nachgeschaut werden)
7. Upload des Pogrammes mittels Upload-Button
8. Öffnen des Serial Monitors unter:  
[Werkzeuge →Serieller Monitor](#)

Nun konnten die verschiedenen RFID-Tag's eingelesen werden und die gelesenen Informationen wurden am seriellen Monitor ausgegeben. [35]

## 6.5 SD-Karte

Die Inbetriebnahme der mikroSD-Karte geschieht mit einem Testprogramm. Das Programm initialisiert die benötigte Hardware (SPI, UART, Pins), enthält eine FAT32-Library und ermöglicht das Debugen über die serielle Schnittstelle.

Sobald die interne Hardware der Mikrocontrollers initialisiert ist, wird die SD-Karte gebootet. Hat dies funktioniert, wird ein File namens *TEST.txt* erstellt, welches den String *Hallo File* enthält.

Vorgehen:

1. Benötigte Applikation, welche im Software-Ordner auf dem USB-Stick oder Github [36] zu finden ist, in Atmel Studio öffnen.  
[Software→Atmega→2\\_SD\\_Karte →SD\\_Karte](#)
2. Software hochladen:  
[AtmelStudio →Tools →PartyMixer](#)
3. Programm für Kommunikation über die serielle Schnittstelle downloaden (z.B HTerm 0.8.1beta.)[37]
4. Verbindung mit Mikrocontroller herstellen und Neustart über Button DTR auslösen. Damit wird der DTR-Pin getoggelt und der Mikrocontroller neu gestartet

Port	=	<b>COMx</b>
Baudrate	=	57600
Data	=	8
Stop	=	1

Der Port **COMx** ist aus dem Geräte-Manager zu entnehmen. Es ist derselbe Port, welcher verwendet wird um die Software hochzuladen.

Ergebnis: Die mikroSD-Karte wird erfolgreich gebootet, was im seriellen Monitorfenster mit "Boot...OK!" angezeigt wird. Das File namens *TEST.txt* mit dem Textinhalt *Hello File* ist jetzt auf der SD-Karte vorhanden.

## 6.6 Flüssigkeitsbeförderung

Die Inbetriebnahme der Flüssigkeitsbeförderung wurde in zwei Etappen durchgeführt. Als erstes wurden die Pumpen in Betrieb genommen. Danach konnten die Durchflussmessgeräte angeschlossen und ausgelesen werden.

### 6.6.1 Pumpen

Um die Flüssigkeitsbeförderung zu testen, wurde ein Testprogramm erstellt, welches die 12 Pumpen der Reihe nach für eine kurze Zeitdauer einschaltet. Dies funktionierte auf Anhieb und es konnten einige Tests durchgeführt werden.

Einer der wichtigsten Tests war es dabei, den maximalen Durchfluss der Pumpen zu bestimmen. Dies ermöglichte eine Abschätzung, wie lange es benötigt um einen Cocktail herzustellen. Um das Pflichtziel von unter einer Minute erfüllen zu können, müssen daher die Pumpen genügend schnell Pumpen können.

Im Schnitt ergab sich dabei eine Abfüllzeit von 26 Sekunden für eine Menge von 5dl. Erstaunlich ist jedoch, dass die Pumpen unterschiedlich lange benötigen für die selbe Menge. Somit benötigte die langsamste Pumpe 28 Sekunden, wobei die schnellste Pumpe lediglich 24 Sekunden benötigte. Das Zusammenspiel von Pumpen und Motor wird in der Zielerreichung in Kapitel 8 aufgezeigt.

### 6.6.2 Durchflussmessgeräte

Die Durchflussmessgeräte in Betrieb zu nehmen gestaltete sich ein wenig schwieriger als bei den Pumpen in Kapitel 6.6.1. Da die Durchflussmessgeräte bei Durchfluss Pulse ausgeben, müssen die positiven Flanken gezählt werden. Dafür wurde das Testprogramm für die Pumpen erweitert. Auch dies funktionierte auf Anhieb und es konnte mit ersten Testläufen begonnen werden.

Da die Durchflussmessgeräte auf dem volumetrischen Prinzip basieren, spielt es keine Rolle ob die 12 Pumpe exakt gleich stark Pumpen oder nicht. Es wurde jedoch schnell festgestellt, dass die Durchflussmessgeräte kleine Toleranzen zueinander aufweisen, jedoch für sich selber relativ exakt arbeiten. Dies hatte zur Folge, dass die einzelnen Durchflussmessgeräte jeweils separat mit der Software kalibriert werden mussten. Dabei ist aufgefallen, dass die Anzahl der Pulse im Verhältnis zum Volumen für die einzelnen Durchflussmessgeräte auch nicht ganz linear ist. Somit wurden zwei Kalibrierungen pro Durchflussmessgerät vorgenommen. Einmal wurden die Durchflussmessgeräte auf 3dl und einmal auf 5dl kalibriert. Dies wurde aus dem einfachen Grund gemacht, dass die Getränke in diesen zwei Größen hergestellt werden. Um dies sicherstellen zu können, wurde jedes der 12 Durchflussmessgeräte in jeweils 6 Durchgängen für 3dl und 5dl kalibriert und Softwaremäßig erfasst. Dabei wurde das abgefüllte Gewicht von Wasser mit einer Küchenwaage gemessen. Wasser hat dabei den Vorteil, dass es die Dichte von nahezu  $1000\text{kg/m}^3$  besitzt, was so viel bedeutet, dass 1g Wasser 1ml entspricht [38].

Als sichergestellt wurde, dass die Durchflussmessgeräte kalibriert sind, wurde mit der eigentlichen Testreihe begonnen. Es ging dabei um die Abfüllgenauigkeit der einzelnen Pumpstationen. Besser gesagt um die Beständigkeit. Dazu wurden erneut einige Testreihen aufgesetzt. Einerseits wurde für jede Pumpe 10 Mal 3dl und 10 Mal 5dl gepumpt und geschaut, wie gross die Toleranz ist. In einem weiteren Schritt wurde dann das Zusammenspiel der einzelnen Pumpen gemessen.

#### **Test 1:**

In diesem Test wurden alle 12 Pumpen 10x Durchgemessen. Dabei wurde für jede Pumpe 10x 3dl abgefüllt und mit einer Küchenwaage gemessen, wie sehr das Ergebnis schwankt. Die grösste festgestellte Toleranz lag hierbei bei 2g Abweichung, was erstaunlich ist, da dies einer Genauigkeit von 0.7% entspricht.

#### **Test 2:**

Das Selbe wie in Test 1 wurde auch in Test 2 durchgeführt. Allerdings wurde die Abfüllmenge auf 5dl erhöht. Hierbei ergab sich eine grösste Abweichung von 5g, was einer Genauigkeit von 1% entspricht.

#### **Test 3:**

Im letzten Test wurden noch einmal 5dl abgefüllt, jedoch nicht von einer einzelnen Pumpe sondern von jeder Pumpe ein zwölftel, wie es bei der Cocktaillerstellung der Fall ist. Dies ist der "Worst-Case". In diesem Test wurde eine maximale Toleranz bei 10 Durchläufen von 18g gemessen, was 3.6% entspricht.

Diese Testreihe ist für die Zielerreichung in Kapitel 8 essentiell.

## 6.7 Beleuchtung

Die Inbetriebnahme der Beleuchtung geschieht mit einem Testprogramm. Das Programm initialisiert die benötigte Hardware des Mikrocontrollers (UART, Pins, Interrupts), enthält eine LED-Library und ermöglicht das Debugen über die serielle Schnittstelle.

Die Farbe und Helligkeit der Beleuchtung wird mit einem PWM-Signal geregelt. Das PWM-Signal wird erzeugt mittels Timer Interrupts, damit Regelung parallel neben dem Programfluss geschehen kann. Insgesamt benötigt es fünf Timer um das LED-Band anzusteuern. Jeweils einen für jede Farbe und einen um durch den Rainbow-Modus zu interieren.

Die Inbetriebnahme wurde anhand des Datenblatts und des Tutorials von Atmel Mikrochip gemacht, welches auf Youtube zu finden ist.[39][40][41]. Im Anhang Kapitel D.1.1 sind Auszüge aus dem Datenblatt und eine Beschreibung des Auszugs zu finden. Das Kapitel zeigt, dass das PWM-Signal mittels Timer-Interrupts erzeugt wird. Um das Signal mit 10kHzr und variablem Duty-Cycle zu Taktten, wird der Timer im CTC-Mode betrieben.

Es wird erwartet, dass das LED-Band zu Beginn Rot leuchtet und dann durch alle Farben wechselt. Wird über den seriellen Port eine '1' geschrieben, so muss das Licht vom Rainbow-Modus zu Weiss wechseln. Wird eine '0' geschrieben, so muss das Licht von Weiss zum Rainbow-Modus wechseln.

Vorgehen:

1. Benötigte Applikation, welche im Software-Ordner auf dem USB-Stick oder Github [36] zu finden ist, in Atmel Studio öffnen.

Software→Atmega→6\_LED\_Control→1\_LED\_Testsoftware→LED\_Control

2. Software hochladen:  
[AtmelStudio](#)→Tools→PartyMixer

Ergebnis: Die LEDs leuchten zu Beginn Rot und wechseln dann durch die Farben. Wird eine '1' eingegeben, wechselt das Licht von Rainbow zu Weiss, wird eine '0' eingegeben, wechselt das Licht von Weiss nach Rainbow. Die Inbetriebnahme hat gezeigt, dass der Duty-Cycle nicht über 95% gehen sollte, da die LEDs ansonsten flimmern.

## 6.8 Motor

Der Motor wird in folgender Reihenfolge in Betrieb genommen:

1. FOC-Treiber
2. Gate-Treiber mit H-Brücke und BLDC
3. ABN-Encoder mit PI-Regler
4. Software-Ramp

Da der FOC-Treiber in einem Openloop-Modus betrieben werden kann, eignet es sich, diesen als Erstes in Betrieb zu nehmen. Im Openloop werden Schaltsignale erzeugt, welche den Motor mit einer vorbestimmten Drehzahl laufen lassen, ohne das Verhalten des Motors zu messen, weder den Strom durch die Spulen noch Lage des Rotors. Werden diese Gatesignale korrekt ausgegeben, so können diese verwendet werden, den Gate-Treiber in Betrieb zu nehmen. Hand in Hand mit dem Gate-Treiber wird die H-Brücke in Betrieb genommen. Sobald die Openloop-Kette (Mikrocontroller, FOC-Treiber, Gate-Treiber, H-Brücke) funktioniert und der angeschlossene Motor gemäss der Vorgegebenen Geschwindigkeit dreht, kann der ABN-Encoder in Betrieb genommen werden und die Closed-Loop-Modi implementiert werden, welche einen Torque-, Velocity- und Positions-Regelkreis beinhalten.

### 6.8.1 FOC-Treiber

Der FOC-Treiber wird mit einem Testprogramm in Betrieb genommen. Das Programm initialisiert die benötigte Hardware des Mikrocontrollers (UART, SPI, Pins), enthält die Library des FOC-Treibers und ermöglicht das Debuggen über die serielle Schnittstelle. Die Software TMCL-IDE hilft bei der Bestimmung der Parameter für den FOC-Treiber. Welche Register wie beschrieben sind ist im Anhang Kapitel E.3 zu sehen. Die Initialisierung sowie das Auslesen gewisser Register ist mit der Testapplikation "*3\_Motor\_Openloop*" möglich. Sie ist im Softwareordner auf dem USB-Stick oder auf Github [36] zu finden.

Es wird erwartet, dass der FOC-Treiber die Gate-CTRL-Signale erzeugt. Der Duty-Cycle muss sich ändern, sobald der FOC-Treiber die Open-Loop-Drehung startet.

## Setup

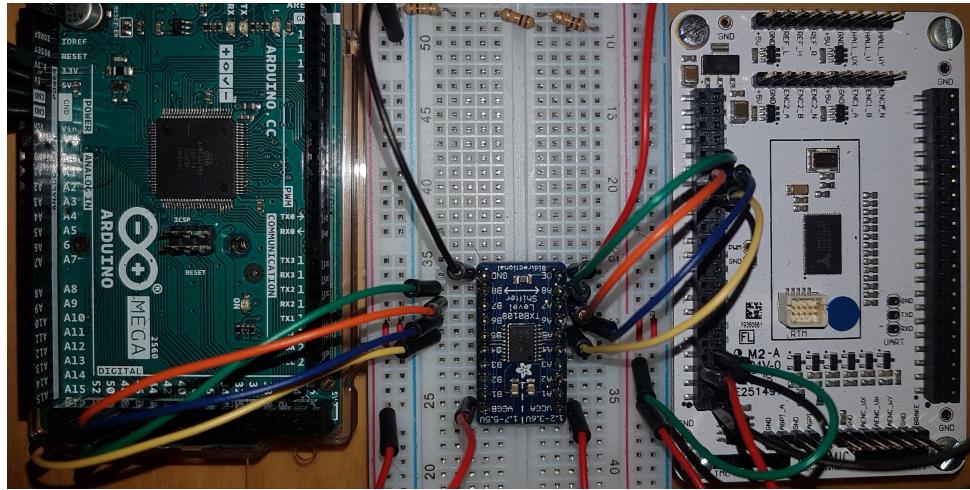


Abbildung 6.2: Gesamtansicht Setup.

**Achtung**, wenn die Scripts auf den Mikrocontroller des PartyMixers geladen werden ist darauf zu achten, dass die Achse des Motors frei beweglich ist und das Förderband nicht mit der Achse mitdreht, da ansonsten Schäden an der Maschine entstehen können. Es wird empfohlen die Scripts mit Mikrocontroller und EVAL-Boards zu testen oder den Motor aus dem Partymixer auszubauen.

Vorgehen:

1. Benötigte Applikation, welche im Software-Ordner auf dem USB-Stick oder Github [36] zu finden ist, in Atmel Studio öffnen.  
[Software→Atmega→3\\_Motor\\_Openloop→1\\_Motor\\_Testsoftware→Motor](#)

2. Software anpassen: Zeile 23 bis 32  
initTMC4671\_Openloop();

```
while (1)
{
    _delay_ms(10000);
    read_registers_TMC4671();
}
```

3. Software hochladen:  
[AtmelStudio→Tools→PartyMixer](#)

Ergebnis: Nach Hochladen der Software liegen die gewünschten Signale an. Da kein Motor angeschlossen ist, müssen die Signale gemessen werden. Nach dem Testdrive werden im 10 Sekunden-Takt diverse aus dem FOC-Treiber ausgelesene Register ausgegeben. Abbildung 6.3 zeigt die High-Side-GateCTRL-Signale und Abbildung 6.4 die Low-Side-GateCTRL-Signale. Beide entsprechen den Erwartungen.

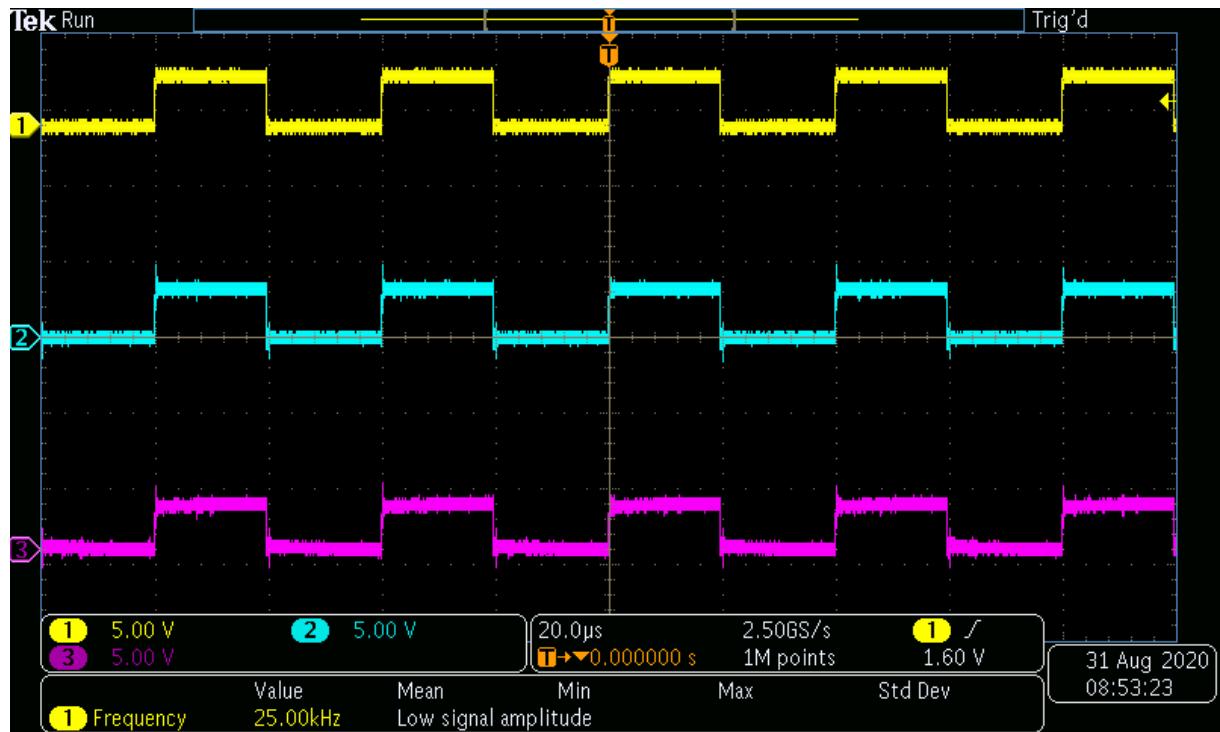


Abbildung 6.3: High-Side-GateCTRL-Signale. Gelb = U, Blau = V, Magenta = W

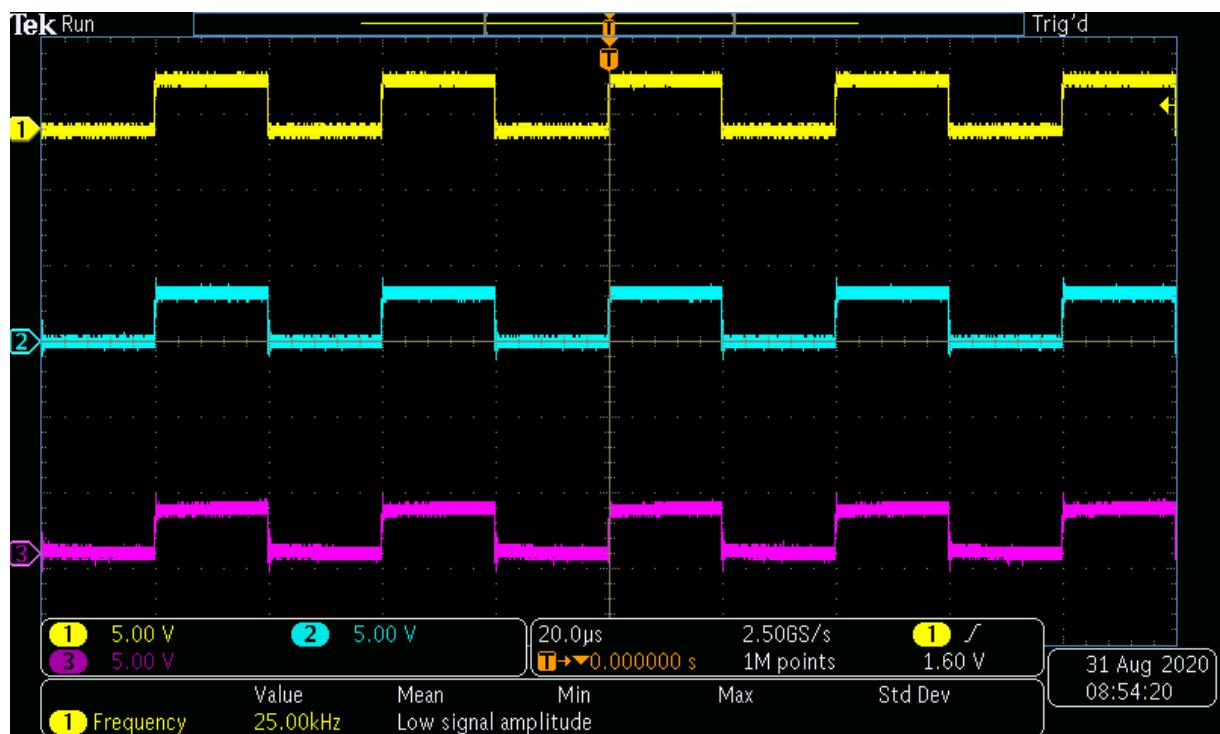


Abbildung 6.4: Low-Side-GateCTRL-Signale. Gelb = U, Blau = V, Magenta = W

Im Anhang Kapitel E.2.1 sind ergänzende Bilder zur Inbetriebnahme der SPI- Kommunikation zu finden.

### 6.8.2 Gate-Treiber, H-Brücke und BLDC

Der Gate-Treiber wird mit einem Testprogramm in Betrieb genommen. Das Programm initialisiert die benötigte Hardware des Mikrocontrollers (UART, SPI, Pins), enthält die Library des FOC- und Gate-Treibers und ermöglicht das Debugen über die serielle Schnittstelle. Die Software TMCL-IDE hilft bei der Bestimmung der Parameter für den Gate-Treiber. Welche Register wie beschrieben wurden ist im Anhang Kapitel F.4 zu sehen. Die Initialisierung sowie das Auslesen gewisser Register ist ebenfalls mit der Testapplikation "3\_Motor\_Openloop" möglich. Dazu müssen einige Zeilen auskommentiert werden.

Es wird erwartet, dass die Gate-CTRL-Signale vom Gate-Treiber aufbereitet werden und eine Spannung an den High- und Lowside-MOSFETs anliegt und die H-Brücke den Motor steuert.

#### Setup

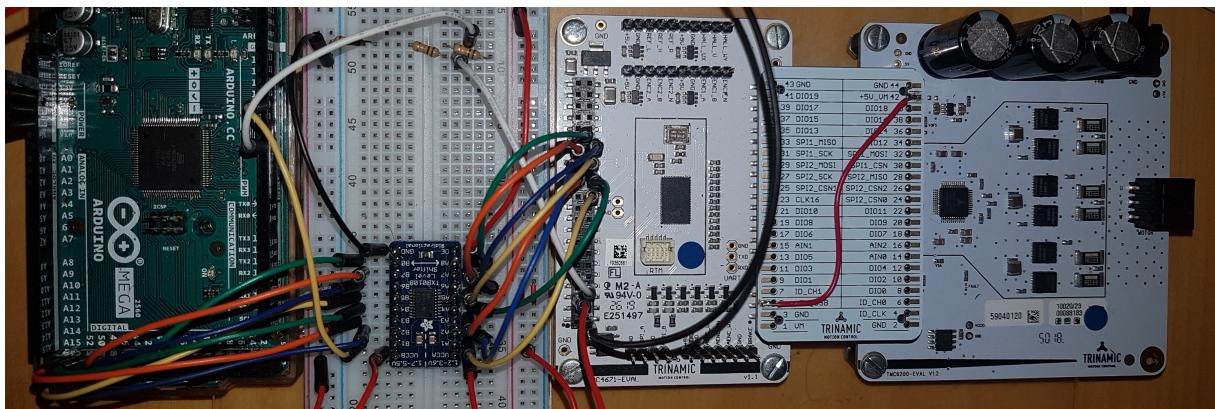


Abbildung 6.5: Gesamtansicht Setup.

**Achtung**, wenn die Scripts auf den Mikrocontroller des PartyMixers geladen werden ist darauf zu achten, dass die Achse des Motors frei beweglich ist und das Förderband nicht mit der Achse mitdreht, da ansonsten Schäden an der Maschine entstehen können. Es wird empfohlen die Scripts mit Mikrocontroller und EVAL-Boards zu testen.

Vorgehen:

1. Benötigte Applikation, welche im Software-Ordner auf dem USB-Stick oder Github [36] zu finden ist, in Atmel Studio öffnen.

Software→Atmega→3\_Motor\_Openloop→1\_Motor\_Testsoftware→Motor

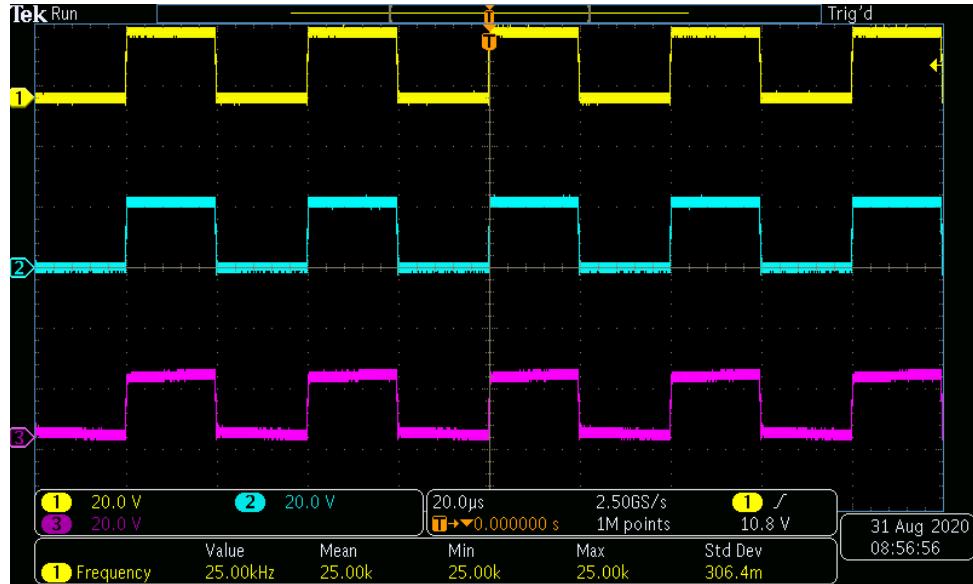
2. Software anpassen:

```
initTMC6200;
initTMC4671_Openloop();
```

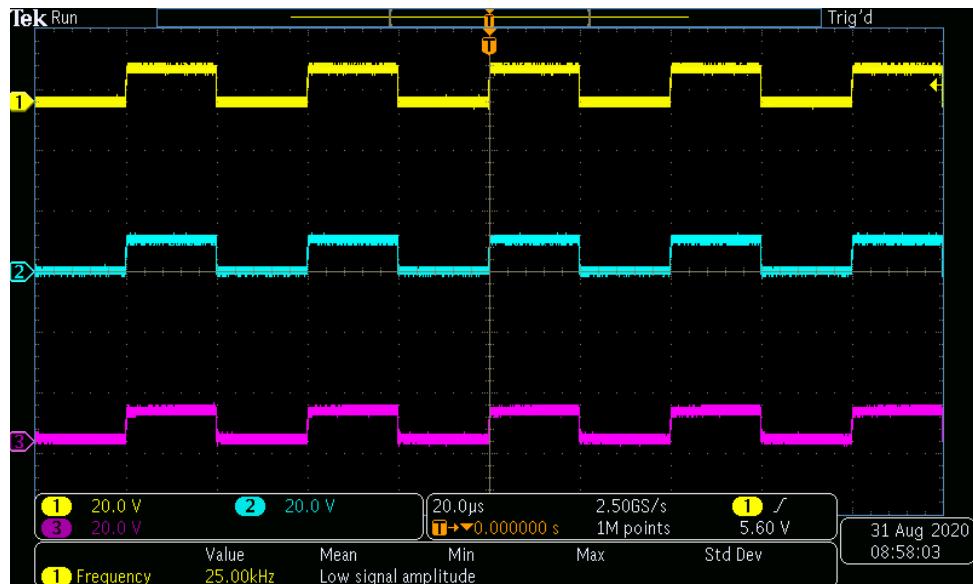
```
while (1)
{
    _delay_ms(5000);
    read_registers_TMC6200();
    _delay_ms(10000);
    read_registers_TMC4671();
}
```

3. Software hochladen:  
[AtmelStudio](#)→Tools→PartyMixer

Ergebnis: Die Gate-CTRL-Signale liegen an. Abbildung 6.6 zeigt die High-Side-Gatesignale, Abbildung 6.7 die Low-Side-Gatesignale. Sie führen auf die MOSFET-Gates, welche die H-Brücke bilden und den Leistungsteil schalten.

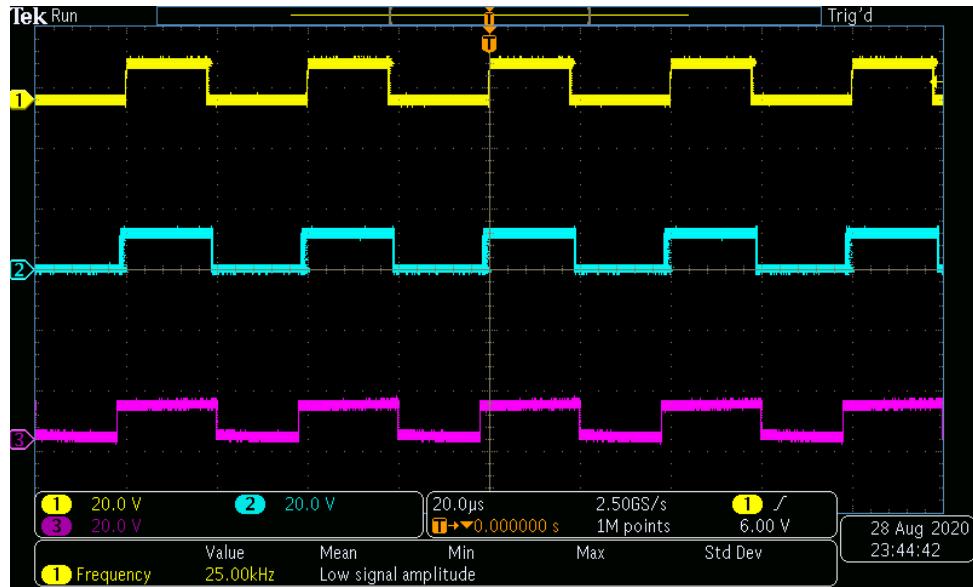


**Abbildung 6.6:** High-Side-Gatesignale von Gate-Treiber auf H-Brücke. Gelb = U, Blau = V, Magenta = W



**Abbildung 6.7:** Low-Side-Gatesignale von Gate-Treiber auf H-Brücke. Gelb = U, Blau = V, Magenta = W

Die Abbildung 6.8 zeigt die Schaltsignale am Ausgang der H-Brücke.



**Abbildung 6.8:** Schaltsignale während dem Openloop Testdrive. Gelb = U, Blau = V, Magenta = W

Wird der Motor angeschlossen und der Mikrocontroller neu gestartet, der Motor sich mit der vorgegebenen Open-Loop-Geschwindigkeit.

### 6.8.3 ABN-Encoder

Der ABN-Encoder wird mit einem Testprogramm in Betrieb genommen. Das Programm initialisiert die benötigte Hardware des Mikrocontrollers (UART, SPI, Pins), enthält die Library des FOC- und Gate-Treibers und ermöglicht das Debugen über die serielle Schnittstelle. Da der ABN-Encoder am FOC-Treiber angeschlossen ist, wird die Schnittstelle für den ABN-Encoder am FOC-Treiber initialisiert. Dazu müssen weitere Register gesetzt werden, worunter auch die Register für die PI-Regler fallen. Die Software TMCL-IDE hilft bei der Bestimmung der Parameter für den ABN-Encoder und PI-Regler. Welche Register wie beschrieben wurden ist im Anhang Kapitel G.1 zu sehen. Die Initialisierung sowie das Auslesen gewisser Register ist mit der Testapplikation „*4\_Motor\_ABN\_Encoder*“ möglich.

Es wird erwartet, dass der FOC-Treiber im Positionsmodus im ersten Moment stehen bleibt. Wird jetzt der Rotor gedreht, muss der FOC-Treiber die Abweichung nachkorrigieren. Wird dem Motor eine andere Position vorgegeben, muss der Motor sich an die gewünschte Position fahren und dort halten.

Die Werte der PI-Regler wurden experimentell bestimmt. Folgende Werte wurden dabei ermittelt:

Die Limits wurden folgendermassen gesetzt:

Limit	Wert	Limit	Wert
Drehmoment	2500	Fluss	2500
Geschwindigkeit	1500		

Die Regler wurden so bestimmt, dass die Regelkreise in unbelasteten Zustand ihre Sollwerte schnellst möglich erreichen, ohne überzuschliessen. Dies betrifft den Regler für den Stromregel-

Regelkreis	P-Anteil	I-Anteil
Drehmoment	700	1000
Fluss	700	1000
Geschwindigkeit	2000	300
Position	2000	0

Tabelle 6.3: PI-Werte

kreis und den Regler für den Geschwindigkeitsregelkreis. Die zugehörigen Parameter wurden in der TMCL-IDE ermittelt.

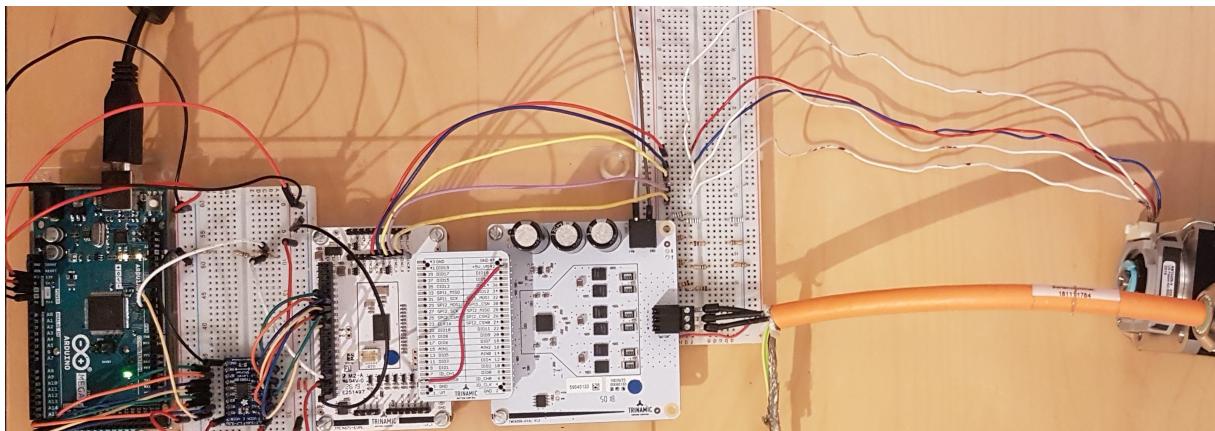


Abbildung 6.9: Komplettansicht mit Mikrocontroller, Level-Shifter, FOC-Treiber, Gate-Treiber, H-Brücke, ABN-Encoder und Motor.

**Achtung**, wenn die Scripts auf den Mikrocontroller des PartyMixers geladen werden ist darauf zu achten, dass die Achse des Motors frei beweglich ist und das Förderband nicht mit der Achse mitdreht, da ansonsten Schäden an der Maschine entstehen können. Es wird empfohlen die Scripts mit Mikrocontroller und EVAL-Boards zu testen.

Vorgehen:

1. Benötigte Applikation, welche im Software-Ordner auf dem USB-Stick oder Github [36] zu finden ist, in Atmel Studio öffnen.

[Software→Atmega→4\\_Motor\\_ABK\\_Encoder→1\\_Motor\\_Testsoftware→Motor](#)

2. Software hochladen:

[AtmelStudio→Tools→PartyMixer](#)

Ergebnis: Nach der Initialisierung wird zuerst ein Testdrive ausgeführt, welcher den Rotor mit hoher Geschwindigkeit eine Sekund nach links und eine Sekunde nach Rechts drehen lässt. Danach wird der Motor in den Positionsmodus versetzt, wordurch der FOC-Treiber bei einer Störung von ausserhalb nachkorrigiert, um wieder auf die Ursprungsposition zu kommen. Wird über den seriellen Monitor '1' eingegeben, fährt der Motor 12 verschiedene Positionen an und dann wieder zurück zur Ursprungsposition. Wird Enter gedrückt, wird die Position inkrementiert. Wird '0' gedrückt, fährt der Motor an die Ursprungsposition zurück. Abbildung 6.10 zeigt die Signale des Encoders während der Bewegung.

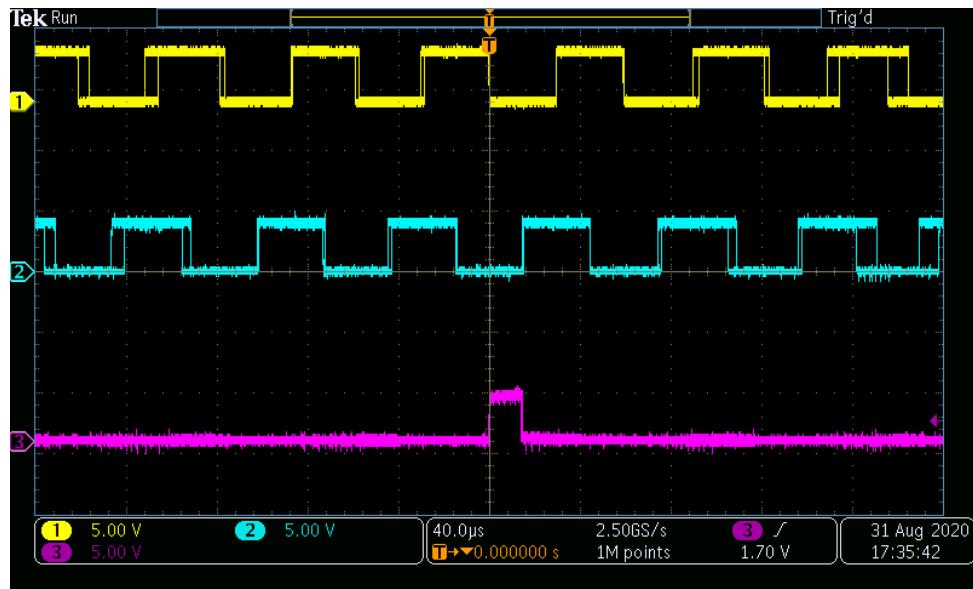


Abbildung 6.10: Encoder-Interface Pinout-Table. Gelb = A, Blau = B, N = Magenta

Die Regler wurden in der TMCL-IDE ermittelt. Eine Tabelle mit den genauen Parametern und Registern ist im Anhang Kapitel G.1 zu finden. Die zu diesem Zeitpunkt ermittelten PI-Werte waren gut für eine erste Inbetriebnahme, mussten jedoch für den Partymixer angepasst werden. Tabelle 6.3 zeigt die Werte, wie sie im Partymixer sind.

Für die folgenden Bilder ist zu beachten, dass sie mit den Werten aus dem Anhang Kapitel G.1 gemacht wurden, so wurde an die Werte herangetastet. Abbildung 6.13 zeigt einen Sprung des Stromes von 0mA nach 2000mA. Der aktuelle Strom wird so vom Stromregelkreis geregelt, dass er schnell in die Höhe schießt und keine Überschwinger hat. In Abbildung 6.12 ist das selbe Verhalten für die Geschwindigkeit zu sehen. Der Geschwindigkeitsregelkreis lässt die Geschwindigkeit hochschnellen und hat keine Überschwinger. Da der Geschwindigkeitsregelkreis ein reiner P-Regler ist, können gar keine Überschwinger entstehen, was sich in Abbildung 6.13 zeigt. Je näher der Rotor der Endposition kommt, desto weniger hat er das Verlangen nachzukorrigieren.

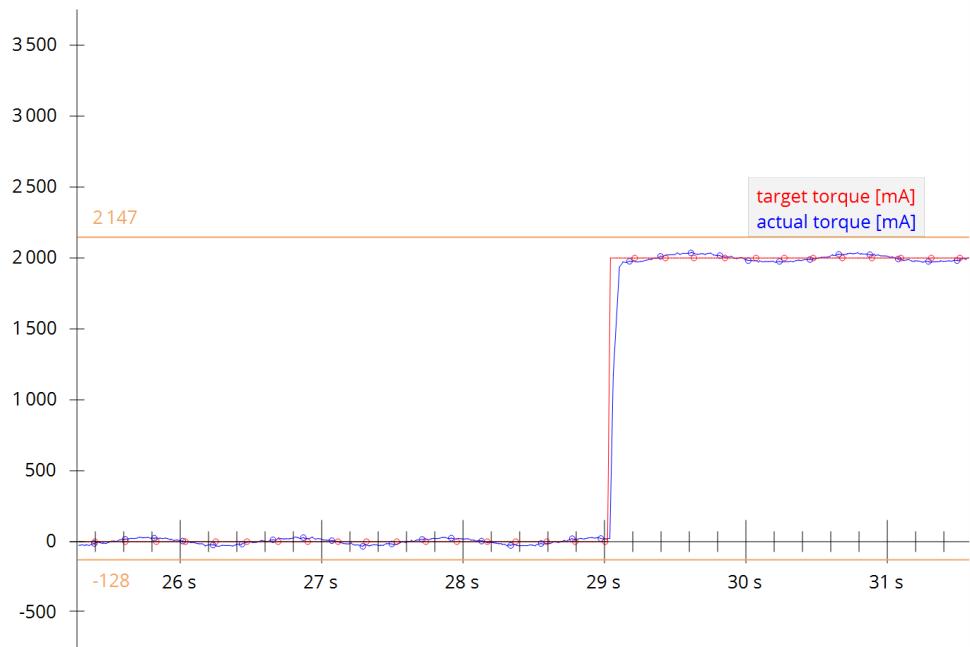


Abbildung 6.11: Drehmoment-Graph bei einem Sprung von 0 bis 2000 mA.

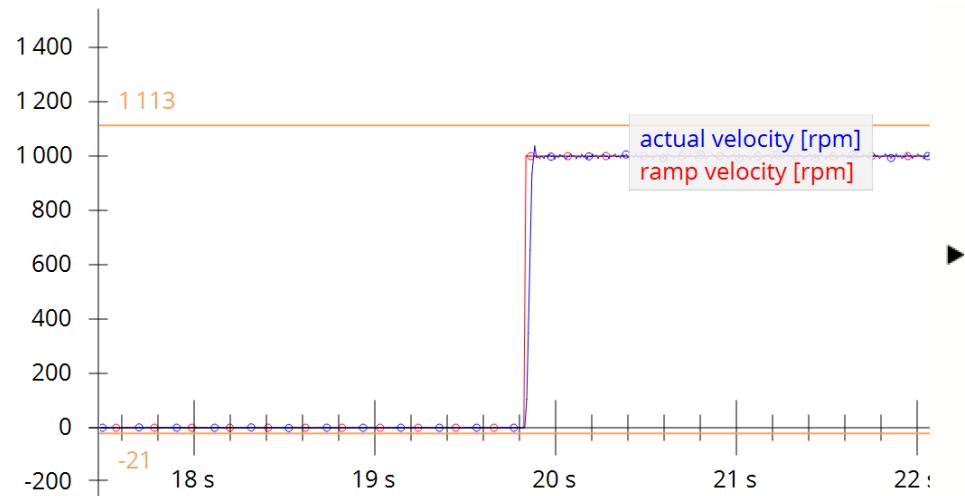


Abbildung 6.12: Geschwindigkeits-Graph bei einem Sprung von 0 bis 1000 Umdrehungen pro Minute.

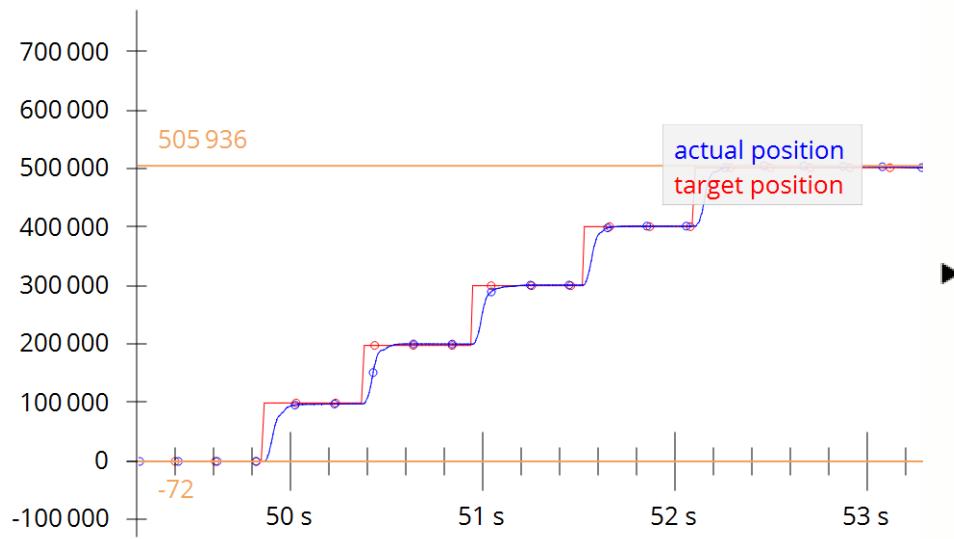


Abbildung 6.13: Positions-Graph bei einem Sprung von jeweils 100000 Schritten.

Die Problematik besteht in der Kontrolle des Motors. So wie die Regler ausgelegt sind, kann der Schlitten, welcher vom Motor bewegt wird, nicht sorgfältig genug bewegt werden. Abhilfe schafft eine Software-Ramp, welche den Weg unter Vorgabe einer Höchstgeschwindigkeit und -Beschleunigung berechnet. Die Ramp gibt dem FOC-Treiber schrittweise die Positionen vor, wodurch der Motor für den Bruchteil einer Sekunde wartet, bis die nächste Position vorgegeben wird.

#### 6.8.4 Software-Ramp

Da der ABN-Encoder am FOC-Treiber angeschlossen ist, wird die Schnittstelle für den ABN-Encoder am FOC-Treiber initialisiert. Dazu müssen weitere Register gesetzt werden, worunter auch die Register für die PI-Regler fallen. Die Software TMCL-IDE hilft bei der Bestimmung der Parameter für den ABN-Encoder und PI-Regler. Welche Register wie beschrieben wurden ist im Anhang Kapitel G.1 zu sehen. Die Initialisierung sowie das Auslesen gewisser Register ist mit der Testapplikation "4\_Motor\_ABN\_Encoder" möglich.

Die Software-Ramp wird mit einem Testprogramm in Betrieb genommen. Das Programm initialisiert die benötigte Hardware des Mikrocontrollers (UART, SPI, Pins), enthält die Library des FOC-Treibers, des Gate-Treibers und der Software-Ramp und ermöglicht das Debugen über die serielle Schnittstelle. Durch eine periodische Iteration über die Zeit gibt die Ramp dem FOC-Treiber schrittweise die berechnete Position vor. Die Endposition, die Geschwindigkeit und die Beschleunigung der Ramp sind einfach skalierbar, was eine schnellere Einbettung in das Gesamtsystem ermöglicht. Das Matlab-Script *GenTraj.m* von Richard Bearee war Grundlage für die Software Ramp. Das Script enthält Funktionen, welche nur schwer mit dem Mikrocontroller umzusetzen sind. Deshalb wurde zur Vorbereitung ein eigenes Matlab-Script geschrieben, mit welchem der Algorithmus in C übersetzt werden kann. Dieses ist im Anhang Kapitel H.2 zu sehen. Es ist zu beachten, dass der Jerk nicht in die Software des Mikrocontrollers implementiert wurde. Einen theoretischen Teil zur Software Ramp ist im Anhang Kapitel H.1 zu finden.

Es wird erwartet, dass sich der Motor mit der Software-Ramp langsam und kontrolliert dreht. Dazu muss über den seriellen Monitor bei leerem Textfeld Enter eingegeben werden.

Vorgehen:

1. Benötigte Applikation, welche im Software-Ordner auf dem USB-Stick oder Github [36] zu finden ist, in Atmel Studio öffnen.

[Software→Atmega→5\\_Motor\\_Linear\\_Ramp→1\\_Motor\\_Testsoftware→Motor](#)

2. Software hochladen:

[AtmelStudio→Tools→PartyMixer](#)

3. Über die serielle Schnittstelle kann nun der Motor mit diversen Eingaben bewegt werden.  
Im File sind weitere Informationen zu finden.

Ergebnis: Nach Übermitteln der Symbole aus dem seriellen Port an den Mikrocontroller beginnt der Motor langsam und kontrolliert mit kontinuierlicher Beschleunigung zu drehen, bis die Maximalgeschwindigkeit erreicht wurde. Vor Erreichen der Endposition verlangsamt der Motor mit kontinuierlicher negativer Beschleunigung.

## 7 Software

Im Folgenden Kapitel werden die beiden Softwareteile beschrieben. Die Software für den Mikrocontroller beinhaltet die komplette Ansteuerung der Teilsysteme, die Führung durch das Menü auf dem Display sowie die Ausführung der Funktionen bei Auswahl auf dem Display. Außerdem wird hier der gesamte Stand der Maschine gespeichert.

Die Software für das Wireless-/Bluetoothmodul erweitert die Funktionalität der Maschine, indem einige Funktionen auch über eine Android-Applikation auf dem Handy aufgerufen werden können. Die Software auf dem Wireless-/Bluetoothmodul übernimmt dabei eine Zwischenfunktion, indem es die eingehende Bluetooth-Kommunikation der Android-Applikation übersetzt und die Daten an den Mikrocontroller sendet. Außerdem werden Daten vom Mikrocontroller abgefragt und auch das RFID-System wurde in das Wireless-/Bluetoothmodul implementiert.

### 7.1 Mikrocontroller

Die Software für den Mikrocontrollers ist wie folgt aufgebaut: Im **Init/Main** werden die Variablen deklariert, welche im Programm benutzt werden, um die aus den Kommunikationsbuffern geholten Daten zur Verarbeitung zu Speichern. Zudem werden hier die Adressen auf die Funktionen deklariert, welche die Module (z.B IO, Speicher, Devices, Interfaces) initialisieren und die Buffer in der main-Schleife prüfen. Durch Aufrufen des h-Files "Cocktail\_Statemachine.h" werden die Adressen der projektspezifischen Funktionen deklariert und somit die **User- Applikation** eingebunden. Darunter befinden sich die Libraries, welche die Registernummern oder Befehlssätze beinhalten, um die Devices anzusteuern. Sie bilden die Schnittstelle zwischen User-Applikation und Kommunikationsinterface. Dies ist vergleichbar mit einem **Application- Programming-Interface**<sup>3</sup>, einem Programmteil, welcher eine Verbindung eines Programms zu einem anderen Programm ermöglicht. Die Informationen werden dabei standardisiert zwischen den Anwendungen ausgetauscht. Die Daten oder Befehle werden strukturiert nach einem definierten Syntax übergeben. Der Zugriff auf die Hardware des Mikrocontrollers (SPI- und UART-Interface) erfolgt mittels den AVR-Registern. Die Libraries, die dafür verwendet werden befinden sich folglich zwischen dem "API"-Layer und der Hardware und können mit der **Hardware Abstraction Layer**<sup>4</sup> verglichen werden. Es wird nur über diese Funktionen auf die entsprechende Hardware zugegriffen. [42]

#### 7.1.1 Strukturplan

In Abbildung 7.1 wird aufgezeigt, wie die Software aufgebaut ist und welcher Teil der Software von welchen Libraries abhängig ist. Wie in Kapitel 7.1 beschrieben kann die Software in folgende Teile gegliedert werden:

- Init/Main
- User Application
- API - Application-Programming-Interface
- HAL - Hardware-Abstraction-Layer
- Interfaces / Pins

---

<sup>3</sup>API, Schnittstelle zur Anwendungsprogrammierung

<sup>4</sup>HAL, Hardwareabstraktionsschicht

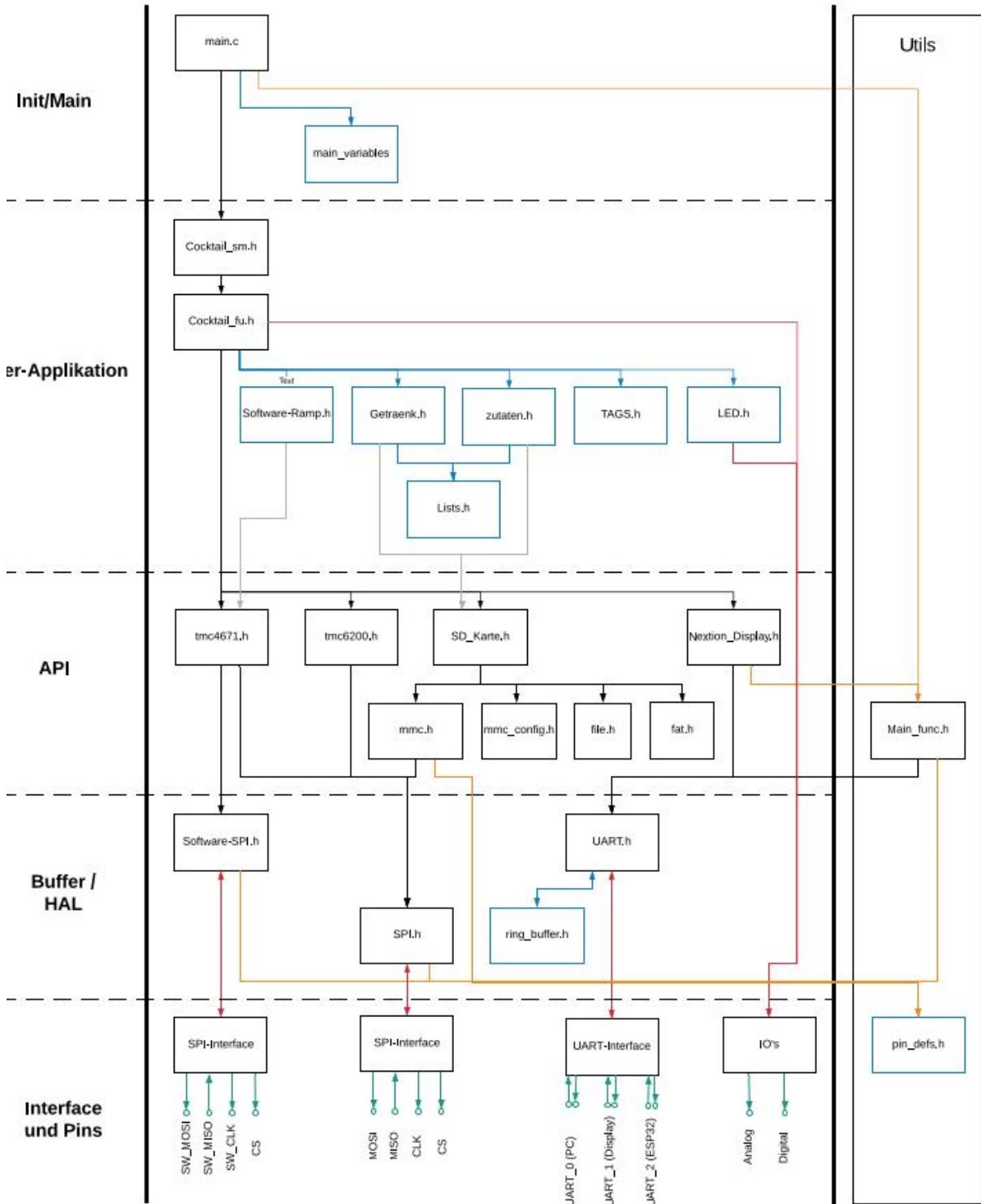


Abbildung 7.1: Softwareübersicht Mikrocontroller

### 7.1.2 Programmflussdiagramm

#### Bootloader-Section

In Abbildung 7.2 wird der Programmfluss der Mikrocontroller-Software aufgezeigt. Beim einschalten des Mikrocontrollers wird in der Bootloader-Section zuerst der Bootloader gestartet (Boot Loader Flash Section). Sofern keine neue Software über die UART0-Schnittstelle kommt, so beginnt das Cocktail spezifische Programm (Application Flash Section). Sofern eine neue Software über die UART0-Schnittstelle gesendet wird, wird die kommende Software in diese Application Flash Section geschrieben und das Programm nach der Übertragung gestartet.

#### Init-Section

Wird das Programm gestartet, beginnt die Init-Section. Hier werden die zuerst die Main-Variablen initialisiert. Danach werden die Adressen der Funktionen ins Programm geladen mittels den h-Files. Die Interfaces (SPI, Uart, I/O's) müssen als erstes initialisiert werden, damit die Initialisierung der SPI-Devices stattfinden kann und Boot-Informationen über eine gewünschte Schnittstelle angezeigt werden können (z.B Uart0 ==> Computer). Sobald die Schnittstellen initialisiert wurden, werden die Devices initialisiert. Dazu gehören die SD-Karte, der FOC-Treiber und der Gate-Treiber. Daraufhin werden die Speicher-Strukturen initialisiert, welche für die User-Applikation (Tabelle 7.1 in Kapitel 7.1.1) gebraucht werden. Zu guter Letzt wird die Startanzeige des Displays geladen.

#### Main-Loop

Da die Maschine nur auf Inputs reagieren muss, werden im Main-Loop nur die Buffer der Devices abgefragt. Sobald ein Terminator-Zeichen ankommt (z.B ein carriage return '\r' der UART0-Schnittstelle oder 0xFF 0xFF 0xFF der UART1-Schnittstelle), werden die zuvor empfangenen Daten interpretiert und verarbeitet.

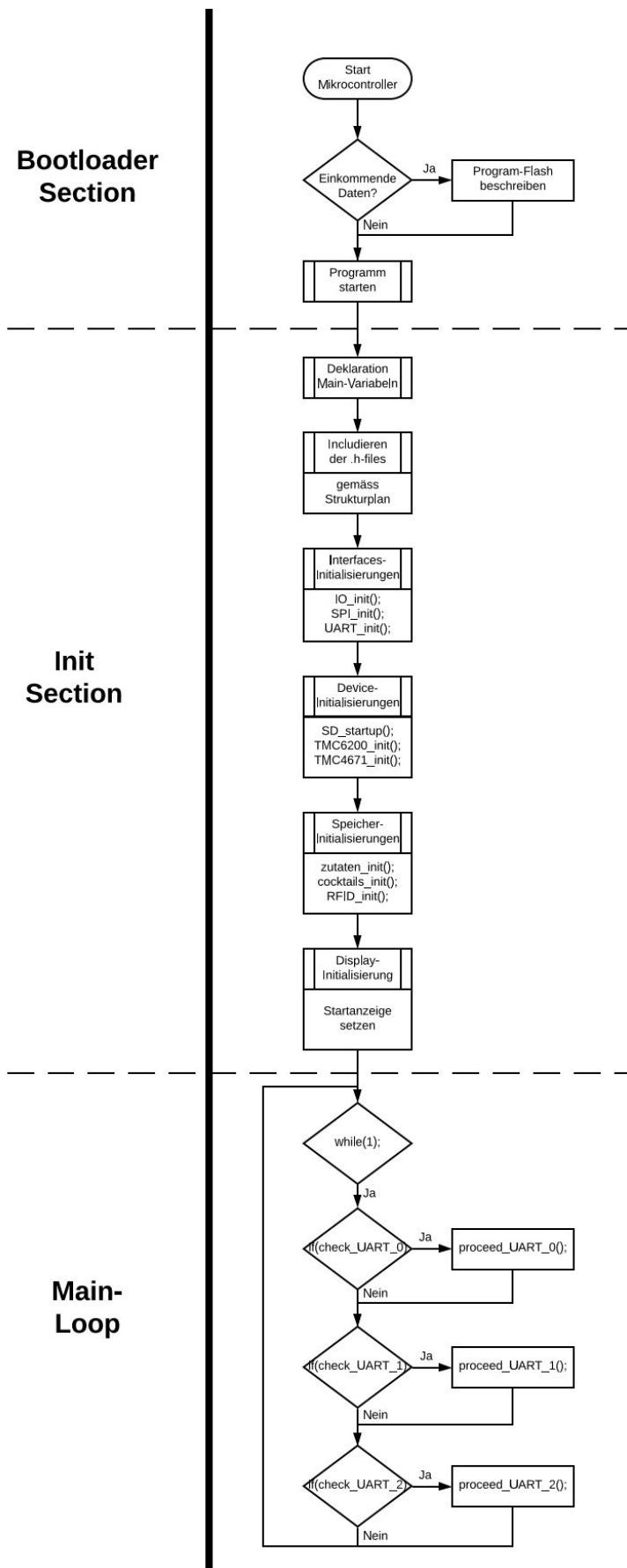


Abbildung 7.2: Programmfluss Mikrocontroller

### 7.1.3 Speicherorganisation

Da im Programmfluss viel mit Listen gearbeitet wird, wird auf *Doubly Dynamic Linked Lists* zurückgegriffen. Eine Doppelt verkettete Liste ist eine gängige Datenstruktur, um sich einfach und effizient in einer Liste hoch und runter zu bewegen. Weiter können der Liste ohne grossen Aufwand sogenannte Nodes<sup>5</sup> hinzugefügt werden. Im Folgenden wird von Elementen statt von Nodes gesprochen. Für jedes Element werden zwei Pointer initialisiert, welche auf das nächste (next) oder auf das vorherige (prev) Element zeigen. Nebst den Pointern auf die umliegenden Elemente enthält das Element die gewünschten Daten. Um zu wissen, ob das Ende oder der Beginn erreicht wurde, werden zwei zusätzliche Pointer initialisiert, welche auf das erste und/oder letzte Element zeigen (Head und Tail). Abbildung 7.3 zeigt eine Struktur der beschriebenen Liste mit next, prev, head, tail und Elementen. [43]

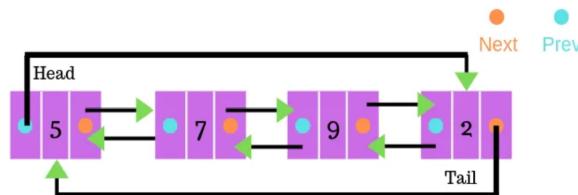


Abbildung 7.3: Doppelt verkettete Liste mit zwei Elementen.[44]

Die Elemente sind in Structs organisiert, welche die Pointer und die Daten enthalten. In der Software werden verschiedene Struct-Typen verwendet. Für jeden Struct-Typ lässt sich eine Liste mit mehreren Structs erstellen. Es werden für die folgenden fünf Elemente Listen angelegt:

- Cocktail-Liste (SD-Files)
- Zutaten-Liste (SD-Files)
- Tag-Liste (Maschine)
- Zutaten-in-Maschine-Liste (Maschine)
- Listen-Pointer

Die Cocktail- und die Zutaten-Liste beinhalten nur die Nummer eines zugehöriges Files, welches sich auf der SD-Karte befindet. Wird ein Cocktail oder eine Zutat benötigt, so werden die Daten temporär von der SD-Karte in einen Struct im Programmspeicher geladen und überschrieben, sobald eine andere Zutat oder ein anderer Cocktail geladen wird.

Die Cocktails sind folglich separat gepseichert, genauso die Zutaten. Die Liste, welche Informationen zu den Zutaten in der Maschine beinhaltet, wird als ein einziges File auf der SD-Karte gespeichert.

## 7.2 Wireless-/Bluetoothmodul

Bei der Inbetriebnahme in Kapitel 5.5.2 wurde das Wireless-/Bluetoothmodull mittels eines Web-Servers getestet und in Betrieb genommen. Da jedoch dies für einen Benutzer relativ umständlich ist, wurde eine Android-App erstellt, mit welcher Befehle an das Wireless-/Bluetoothmodul gesendet werden können. Diese kommuniziert über Bluetooth und hat den Vorteil, dass eine App einiges benutzerfreundlicher ist, da App's im Alltag fast jeder Person integriert sind [33].

Die Software für das Wireless-/Bluetoothmodull wurde komplett in Arduino IDE geschrieben und beinhaltet folgende Bereiche:

---

<sup>5</sup>Nodes = Knoten

- Daten von der App empfangen
- Daten an die App senden
- Daten vom uP abfragen
- Daten an den uP senden
- Daten vom RFID-Leser empfangen

Im Programm werden zuerst die notwendigen Bibliotheken eingebunden. Dabei werden folgende Bibliotheken verwendet:

- Arduino.h
- BluetoothSerial.h
- SPI.h
- MFRC522.h

Dabei beinhaltet die «Arduino.h» Bibliothek die Standardbefehle von Arduino, die «BluetoothSerial.h» Bibliothek beinhaltet alle Kommunikationsbefehle um über Bluetooth kommunizieren zu können, die «SPI.h» Bibliothek beinhaltet alle Kommunikationsbefehle um über SPI kommunizieren zu können und die «MFRC522.h» Bibliothek beinhaltet alle Befehle, um mit dem RFID-Modul arbeiten zu können.

Es wurde bei der Programmierung darauf geachtet, dass mit möglichst einfachen Mitteln gearbeitet wird. Um zu Kommunizieren wurden die Datentypen String und Char verwendet. Für Zählvariablen wurde der Datentyp int verwendet.

### 7.2.1 Programmflussdiagramm

#### Bootloader-Section

In Abbildung 7.4 wird der Programmfluss des Wireless-/Bluetoothmoduls (ESP32) aufgezeigt. Beim Einschalten der Maschine wird das ESP aufgestartet, bevor es in die Init-Section wechselt.

#### Init-Section

Bei Programmstart werden zuerst die Headerfiles geladen und die Variablen initialisiert, sowie die I/O Schnittstellen und die angeschlossenen Devices. Dabei handelt es sich um das MFRC522-Evalboard (RFID).

#### Main-Loop

Das Wireless-/Bluetoothmodul reagiert jeweils auf einkommende Daten. Entweder treffen Daten vom App über Bluetooth ein, vom MFRC522 über SPI oder vom Mikrocontroller via UART. Sobald Daten aus einer dieser Quellen empfangen werden, so reagiert das Programm darauf und löst eine entsprechende Routine aus.

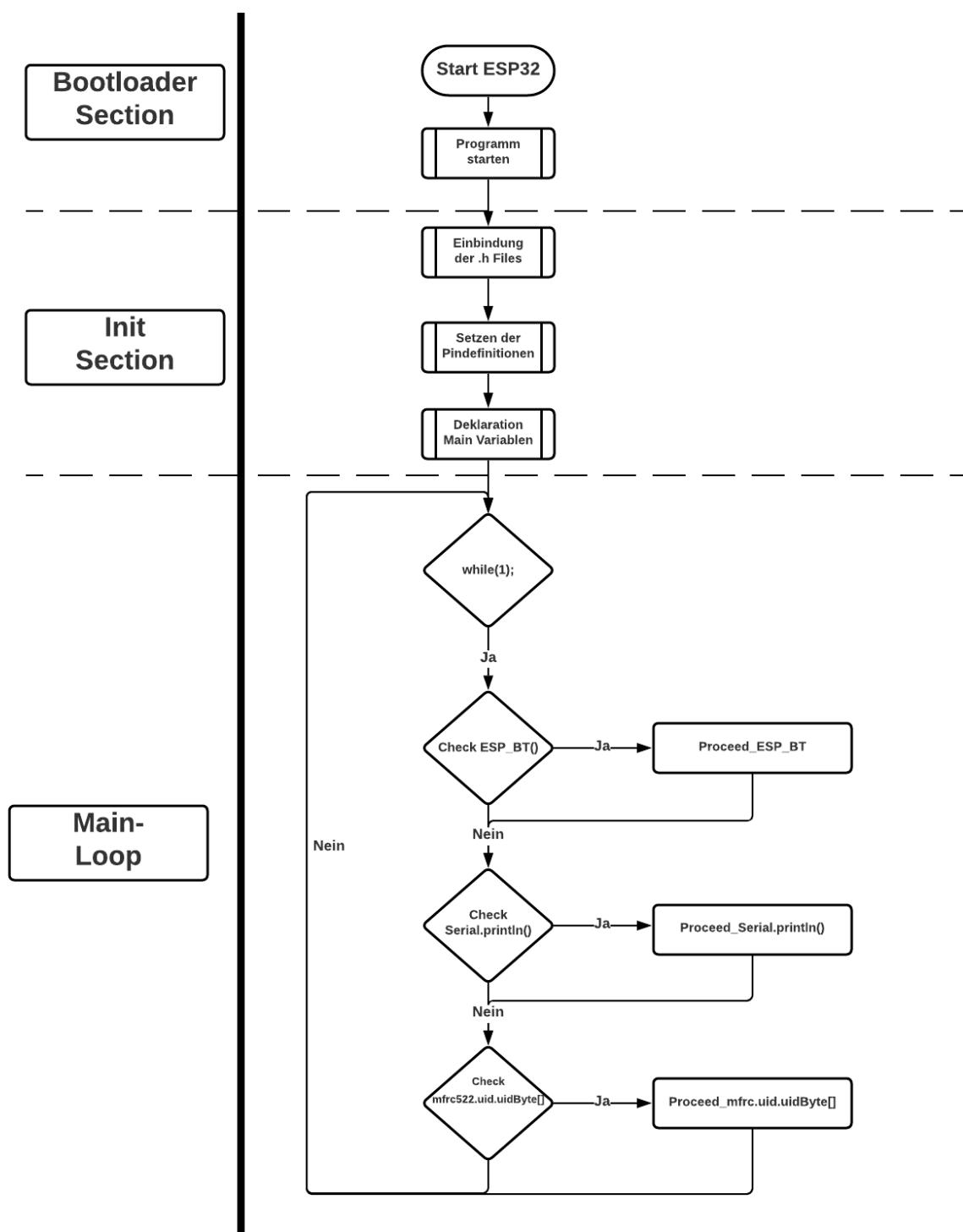


Abbildung 7.4: Programmflussdiagramm des ESP32

### 7.3 Displaysoftware

Das NX8048T070 7-Zoll Display von Nextion wurde mit der dazugehörigen Software «Nextion Editor» programmiert. Diese wurde eigens für die angebotenen Displaytypen entwickelt und erleichtern dem Benutzer die Erstellung von grafischen Oberflächen und deren Funktionen. Die Softwareoberfläche ist in Abbildung 7.5 zu sehen. Der Aufbau ist simpel gestaltet. In der Mitte der Software befindet sich eine grosse Ansicht der bereits erstellten Displayseite. Darauf sind alle Komponenten zu sehen, welche schon platziert wurden und deren Formatierung. Auf der linken Seite können im oberen Bereich Elemente per Drag and Drop in die Displayseite gezogen werden. Unterhalb der Elementauswahl befindet sich ein Medienfenster, in welchem Bilder für Hintergründe der Elemente, Audiodateien und andere Medien abgespeichert werden können. Auf der rechten Seite im oberen Bereich befindet sich die Seitenansicht, worin alle erstellen Displayseiten sichtbar sind. Die ausgewählte Displayseite erscheint in der Mitte der Software. Um die gewählten Elemente formatieren zu können, befindet sich im unteren Bereich der rechten Seite ein Fenster mit den möglichen Formatierungen des jeweils gewählten Elementes. Unter der Displayseite gibt es noch ein Output-Fenster und ein Event-Fenster, in welchem die gewünschte Programmierung festgelegt werden kann. Ein cooles Feature des Nextion Editor's ist der Debug-Modus. Mit diesem Tool kann auch ohne Bildschirm die Funktion getestet werden. [45] [46]

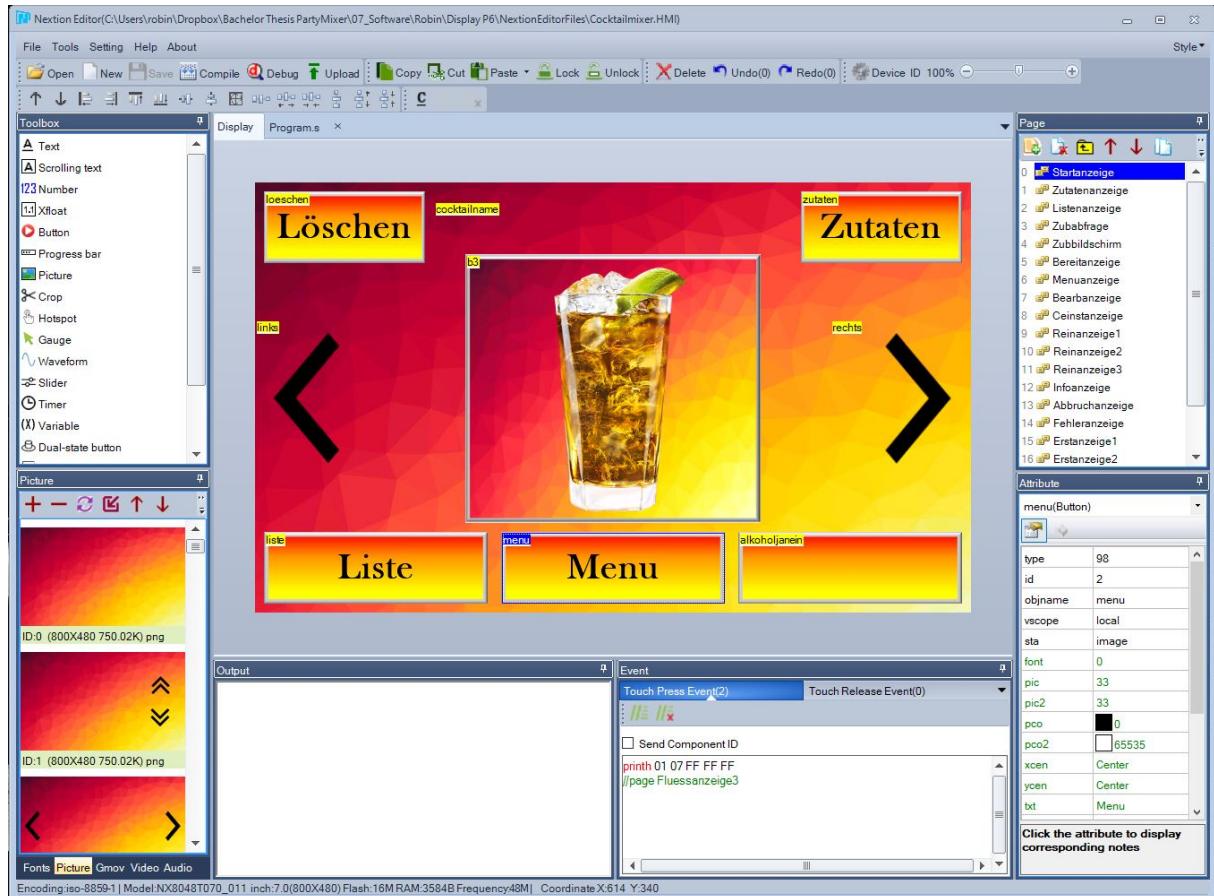


Abbildung 7.5: Programmieroberfläche von Nextion [46] [47]

### 7.3.1 Displaymenu

Das Bedienmenü am Display bietet dem Benutzer viele Freiheiten, was ein umfangreiches und möglichst intuitives Bedienkonzept erforderte. Folgende Einstellungen und Anpassungen können vom Benutzer vorgenommen werden:

- Getränkeauswahl gemäss Bild oder Listenansicht
- Zutatenabfrage
- Bearbeitung von bereits existierenden Cocktails
- Erstellen neuer Cocktails
- Individuelle Zusammenstellung und Positionierung von Flüssigkeiten
- Zuweisung eines Getränktes zu einem RFID-Tag
- Anzeige einer Maschineninfo
- Individuelle Beleuchtung der Maschine
- Maschinen Reinigungsmodus

Im folgenden wird auf die gesamte Menüstruktur eingegangen und erklärt, wie diese zu bedienen ist.

#### Hauptansicht

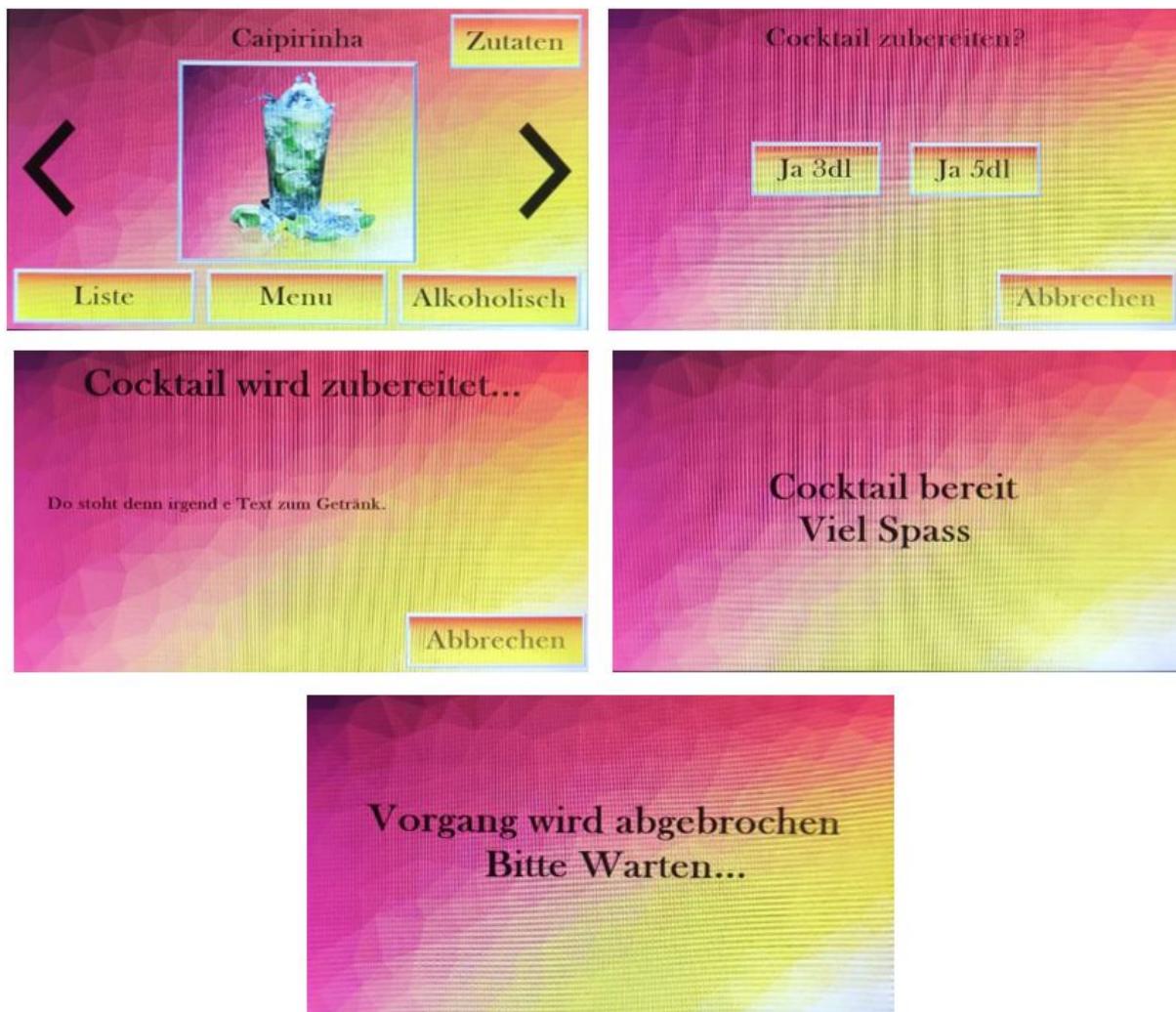
In der Hauptansicht gemäss Abbildung 7.6 wird die Getränkeauswahl getroffen. Dabei gibt es zwei verschiedene Auswahlmöglichkeiten. Mit den Pfeiltasten nach links oder rechts kann ein Getränk aus der Maschine ausgewählt werden. Es wird dabei immer der Getränkename und eine dazugehörige Abbildung angezeigt, damit dem Kunden der Cocktail auch schmackhaft gemacht werden kann. Will man schneller suchen, so kann man mittels «Liste» eine Listenansicht öffnen, wobei man mit den Pfeiltasten nach oben und unten scrollen und den gewünschten Cocktail auswählen kann. Wird eine Auswahl getroffen, so gelangt man zurück in die Hauptansicht und der gewählte Cocktail erscheint mit dem dazugehörigen Bild. Mit «Zurück» gelangt man ebenfalls zurück in die Hauptansicht. Es gibt auch Filtermöglichkeiten. Auf der rechten unteren Seite des Display's kann per Klick entschieden werden, ob nur alkoholische, alkoholfreie oder alle Cocktails angezeigt werden. Hat man ein Getränk gefunden und möchte wissen von welcher Zutat welche Menge darin enthalten ist, so kann man mittels Klick auf «Zutaten» ein Fenster öffnen, welches einem anzeigt was und wie viel davon sich darin befindet. Möchte man Einstellungen vornehmen, so gelangt man mittels «Menü» in das Maschinenmenü.



**Abbildung 7.6:** Hauptansicht der Menüstruktur des Display's [48]

### Getränkeauslösung

Wurde die Auswahl für ein Getränk getroffen, so kann man mittels Klick auf das Bild das Getränk auslösen. Dabei gelangt man in eine Abfrage, welche dem Kunden die Wahl lässt, ob er gerne 3dl oder 5dl haben möchte. Mit «Abbrechen» gelangt man in die Hauptanzeige zurück. Bei Klick auf den entsprechenden Button 3dl oder 5dl wird die gewünschte Grösse des Getränks ausgelöst und am Bildschirm erscheint ein kleiner Text, welcher einem das Warten versüßst. Dazu muss natürlich ein Glas vom Kunden auf den Schlitten gelegt werden. Der Schlitten befördert nun das Glas zu den einzelnen Positionen und füllt es dort ab. Sobald das Getränk fertig ist, fährt der Schlitten an die Ausgangsposition zurück und es wird am Bildschirm angezeigt, dass das Getränk nun fertig ist. Es erscheint danach erneut der Hauptbildschirm und es kann von neuem gestartet werden. Wird während des Erstellungsvorgangs auf «Abbrechen» gedrückt, so wird der Prozess abgebrochen, was vom Display angezeigt wird. In Abbildung 7.7 sind die dazugehörigen Bildschirme zu sehen.



**Abbildung 7.7:** Auslösevorgang der Menüstruktur des Display's [48]

### Maschineninfo und LED-Beleuchtung

Im Maschinenmenü kann unten links eine Info angezeigt werden, welche dem Kunden eine Information gibt, wer diese Maschine gebaut hat und wieso. Die Maschine verfügt über eine eigene Beleuchtung mittels RGBW-LED's. Diese kann vom Kunden individuell eingestellt werden. Klickt man auf «LED», so erscheint das Beleuchtungsmenü. In diesem Menü kann die Beleuchtung entweder auf Weiss, Rainbow oder Custom eingestellt werden. Wird auf «Weiss» gedrückt, so leuchtet die Maschine Weiss. Bei «Rainbow», was standardmäßig eingestellt ist, werden die Farben langsam abwechselnd durchgespielt. Wird «Custom» gedrückt, so können mittel Schieberegler eigene Farben eingestellt werden. Am Anfang stellen sich die Schieberegler auf die zuvor eingestellte Farbe ein. So kann auch mittels Rainbow eine Farbe belassen werden, wenn einer eine Farbe sehr gut gefällt. Wird «Zurück» betätigt, so gelangt man erneut in das Maschinenmenü. Die dazugehörigen Menübildschirme sind in Abbildung 7.8 zu sehen.



**Abbildung 7.8:** Infomenü und LED Einstellungen der Menüstruktur des Display's

### Cocktails bearbeiten

Um dem Kunden Flexibilität gewährleisten zu können, ist es möglich gemäss Abbildung 7.9 bereits vorhandene Getränke zu bearbeiten. Somit kann ein Cocktail bei Bedarf hochprozentiger eingestellt werden oder umgekehrt. Um in diesen Bearbeitungsmodus zu gelangen muss «Cocktails bearbeiten» im Hauptmenü gewählt werden. Es erscheint nun eine Liste, ähnlich der Listenansicht in Abbildung 7.6. Hier kann entweder das zu bearbeitende Getränk ausgewählt werden oder mittels «Standardeinst.» alle Zutaten in der Maschine auf den Ursprungszustand gesetzt werden. Eine Anpassung der momentanen Zutaten in der Maschine hat eine automatische Aktualisierung der mit den Zutaten herstellbare Cocktails zur Folge. Mit «Zurück» gelangt man wieder in das Hauptmenü. Wird ein Getränk ausgewählt, so erscheint eine Liste mit den vorhandenen Flüssigkeiten und einem jeweils dazugehörigen Schieberegler. In diesem Anpassungsmenü sind die momentan eingestellten Dosierungen bereits eingestellt. Diese können nun angepasst werden. Es kann jedoch niemals 100% über- oder unterschritten werden. Versucht man dies, so kann die Einstellung nicht abgespeichert und es erscheint eine Meldung. Mit «Speichern» werden die angepassten Einstellungen abgespeichert und es erscheint kurz ein Speicherbildschirm. Danach gelangt man zurück in die Hauptansicht mit dem angepassten Cocktail als Anzeige.



Abbildung 7.9: Cocktail Bearbeitungsmenü der Menüstruktur des Display's

### Cocktails erstellen

Um dem Kunden noch mehr Flexibilität bieten zu können, ist es auch möglich komplett neue Cocktails zu erstellen gemäss Abbildung 7.10. Dazu muss im Hauptmenü «Cocktail erstellen» getätigert werden. Man gelangt nun in ein Tastaturmenü, wo man den gewünschten Namen des Cocktails eingeben kann. Dabei kann mit der Taste «Fs» zwischen Gross- und Kleinschreibweise gewechselt werden. Leuchtet «Fs» grün, so wird Gross geschrieben und ist «Fs» schwarz, so wird klein geschrieben. Mit «Del.» kann ein Zeichen gelöscht und mit Leerzeichen ein Leerzeichen eingesetzt werden. Mit «Abbrechen» gelangt man wieder in die Hauptansicht. Mit «OK» wird der Name zwischengespeichert und man gelangt in eine Listenansicht der vorhandenen Flüssigkeiten, wo gemäss dem Menü in «Cocktails bearbeiten» die Dosierung eingestellt werden kann. Auch hier kann 100% nicht über- oder unterschritten werden. Mit «Speichern» wird der Cocktail abgespeichert und man gelangt in die Hauptansicht mit dem eben erstellten Cocktail. Wird «Abbrechen» betätigert, so gelangt man auch in die Hauptansicht zurück und der Cocktail wird verworfen. Selbst erstellte Cocktails erhalten alle dasselbe Bild in der Hauptansicht. Außerdem ist es möglich selbst erstellte Cocktails zu löschen. Dazu ist in der Hauptansicht bei den selbst erstellten Cocktails links oben ein Button «Löschen» zu finden. Wird dieser betätigert, so kommt man in eine Abfrage, welche mit «Ja» bestätigt und mit «Nein» dementiert werden kann.

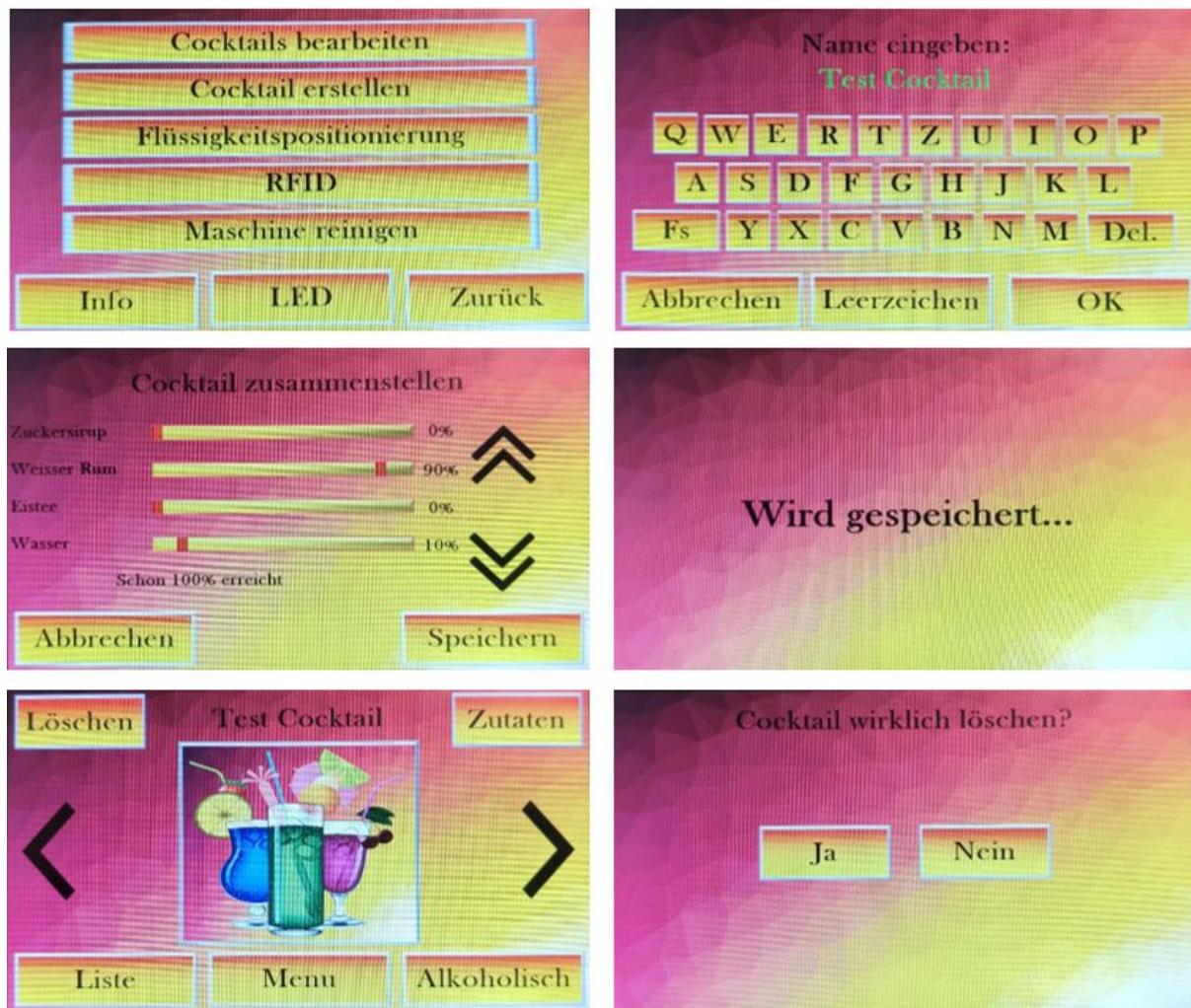


Abbildung 7.10: Cocktail Erstellungsmenü der Menüstruktur des Display's [49]

### Flüssigkeitspositionierung

Um dem Kunden absolute Freiheit bei der Bedienung geben zu können, können auch die 12 Flüssigkeiten selbst positioniert und angepasst werden. Wird im Hauptmenü auf «Flüssigkeitspositionierung» gedrückt, so erscheint eine Zeichnung der Maschine im Grundgerüst, wobei die 12 Flüssigkeitsbehälter zu sehen sind, welche mit Nummern versehen sind. Diese Leuchten grün, wenn eine Flüssigkeit zugewiesen ist. Bei Klick auf eine der 12 Nummern kann die darin enthaltene Flüssigkeit angepasst werden. Es erscheint dabei eine Listenansicht, in welcher die bereits zugewiesene Flüssigkeit grün aufleuchtet. Bei Klick auf eine Zutat aus der Liste wird diese auf der gewählten Position abgespeichert und man gelangt zurück zum Grundgerüst. Im oberen Teil des Bildschirms wird zudem angezeigt, was gerade eingestellt wurde. Wird anstelle einer Flüssigkeit «Keine Flüssigkeit» gedrückt, so gelangt man zum Grundgerüst zurück und die entsprechende Positionsnummer erscheint schwarz. Es ist nun keine Flüssigkeit der Position zugewiesen. Auch diese Änderung wird im Titel angezeigt. Mit «Zurück» gelangt man wieder ohne Änderung zum Grundgerüst. Die dazugehörigen Bildschirme sind in Abbildung 7.11 zu sehen.



Abbildung 7.11: Flüssigkeitspositionierung der Menüstruktur des Display's

### Neue Flüssigkeit erstellen und Positionieren

Der Benutzer ist auch bei den Flüssigkeiten nicht auf die vorhandenen Flüssigkeiten reduziert. Auch hier können absolut neue Flüssigkeiten hinzugefügt werden. Dazu kann man bei der Flüssigkeitsauswahl mit dem Button «Neue Flüssigkeit» eine neue Flüssigkeit erstellen und hinzufügen. Diese Bildschirme sind in Abbildung 7.12 zu sehen. Wird auf den Button gedrückt, so erscheint erneut eine Tastaturansicht wie bei «Cocktail erstellen». Danach kommt eine Abfrage, ob das Getränk kohlensäurehaltig ist. Dies hat den Grund, dass Flüssigkeiten mit Kohlensäure beim Pump- und Messvorgang ihre Kohlensäure verlieren. Flüssigkeiten mit Kohlensäure können erstellt, aber nicht zugewiesen werden. Diese Getränke müssen extern gelagert werden. Drückt man auf «mit Kohlens.» gemäß Abbildung 7.11 so erscheint eine Liste, in welcher alle kohlensäurehaltigen Getränke ersichtlich sind. Ist eine Flüssigkeit Grün, so ist sie eingebunden. Ist eine Flüssigkeit schwarz, so ist sie nicht eingebunden. Mit einem Klick auf die Flüssigkeit kann dies eingestellt werden. Wird ein Cocktail mit Kohlensäure erstellt, so werden alle Komponenten abgefüllt welche in der Maschine sind und nicht kohlensäurehaltig sind. Nach dem Vorgang wird das Glas zurück gebracht und dem Kunden wird am Display angezeigt, mit welchen kohlensäurehaltigen Flüssigkeiten und mit wie viel davon der Cocktail extern aufgefüllt werden muss. Nach der Abfrage der Kohlensäure kommt eine weitere Abfrage, ob die Flüssigkeit Alkohol enthält. Dies ist wichtig für den Filter in der Hauptanzeige rechts unten. Nach dieser Abfrage wird die Flüssigkeit eingelesen und erstellt. Es wird auch eine neue Cocktailliste erstellt,

denn es werden dem Kunden jeweils nur die Cocktails angezeigt, welche mit den abgespeicherten Flüssigkeiten auch wirklich machbar sind. Die anderen bereits erstellten Cocktails bleiben gespeichert, werden jedoch ausgeblendet. Das Selbe geschieht auch, wenn man beim Grundgerüst auf «Zurück» drückt. Die Änderungen werden übernommen und eine neue Cocktailliste wird erstellt.



**Abbildung 7.12:** Flüssigkeitserstellung der Menüstruktur des Display's

### RFID-Zuweisung

Im Hauptmenü können auch die RFID-Tag's zugewiesen werden. Dabei muss der Button «RFID» gedrückt werden. Wird dies gemacht, so erscheint eine Liste mit den 10 Tag-Nummern. Mit den Pfeiltasten kann nach unten oder oben gescrollt werden. Mit «Zurück» gelangt man erneut in das Hauptmenü. Bei Klick auf eine Nummer erscheint die Cocktailliste, wobei ein Getränk ausgewählt werden kann, welches dem vorherig ausgewählten Tag zugewiesen werden soll. Wird ein Getränk ausgewählt, so wird dieses auf dem Tag abgespeichert und man gelangt zur Hauptansicht. Mit «Zurück» gelangt man erneut zu der Tag-Nummer-Liste. Die dazugehörigen Bildschirme sind in Abbildung 7.13 zu sehen.



Abbildung 7.13: RFID-Zuweisung der Menüstruktur des Display's

### Maschinenreinigung

Um ein Verkleben der Schläuche und Pumpen nach der Party zu verhindern, wurde ein Reinigungsvorgang abgespeichert. Dazu kann im Hauptmenü «Maschine reinigen» gewählt werden. Man wird dabei durch einen Reinigungsvorgang geführt gemäss den Bildschirmen in Abbildung 7.14.



Abbildung 7.14: Reinigungsmodus der Menüstruktur des Display's

## 7.4 Android App

Um den Aufwand der Erstellung einer App möglichst gering halten zu können, wurde bei der Erstellung auf ein Web-Tool zurückgegriffen, welches mittels einer einfach gestalteten Benutzeroberfläche App-Erstellungen ermöglicht. Bei dem Programm handelt es sich um AppInventor. Es gibt einige weitere solche Anbieter wie auch Kodular. Bei diversen Programmiersversuchen stellte sich jedoch heraus, dass AppInventor trotz einiger schwächen flüssiger und zuverlässiger läuft als seine Konkurrenz. Die beiden grössten Schwächen von AppInventor sind, dass mit dem Tool keine dynamischen Buttons erstellt werden können, was es schwierig gestaltet Buttonlists oder ähnliches zu erstellen. Als weiterer grosser Schwachpunkt zählt, dass zwar verschiedene Seiten erstellt werden können, jedoch ein Bluetooth-Client zum Beispiel jeweils nur auf einer Seite aktiv sein kann. Somit müsste man bei einem Seitenwechsel jedes Mal den Bluetooth-Client neu verbinden. Beide Schwächen können jedoch mit einem Trick umgangen werden. Anstelle von Buttonlists oder dynamischen Schiebereglern kann man mit Dropdownisten arbeiten, welche sich bei der Öffnung jeweils aktualisieren und an Stelle eines Seitenwechsels kann mit der Sichtbarkeit von Elementen gearbeitet werden. Dies macht jedoch den Programmaufbau komplexer und unübersichtlicher. [50] [51]

AppInventor arbeitet mit zwei verschiedenen Benutzeroberflächen. Einerseits mit dem Designer, in welchem die App mit relativ einfachen Bausteinen gestaltet werden kann und anderseits mit den Blocks, wo der Programmablauf blockmäßig erstellt werden kann.

Der Designer, welcher in Abbildung 7.15 zu sehen ist, besteht aus vier Einheiten. Ganz links befinden sich die Bausteine in der Palette, welche per Drag and Drop in den Viewer gezogen werden können, welcher sich gross in der Mitte befindet. Links neben dem Viewer befinden sich dann die schon rein gezogenen Komponenten als Übersicht und ganz rechts die Einstellungen und Formatierungsmöglichkeiten zu den gewählten Komponenten.

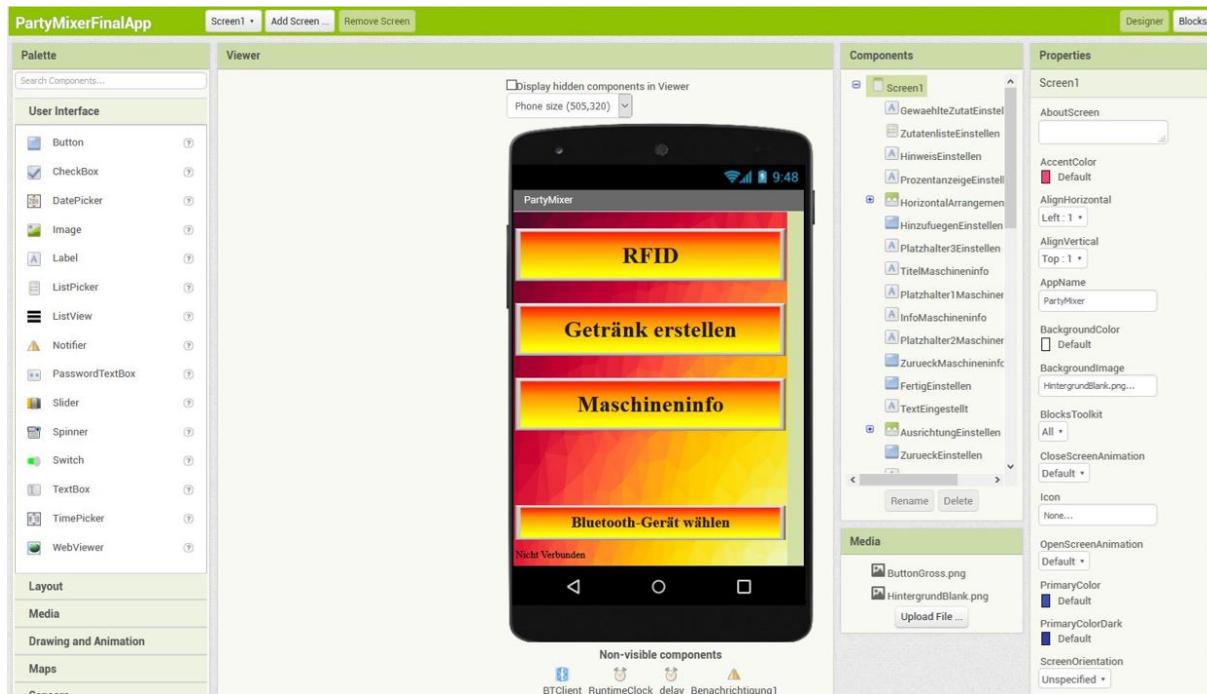


Abbildung 7.15: Designer von AppInventor [50]

In den Blocks gemäss Abbildung 7.16 wird der eigentliche Programmablauf erstellt. Dabei ist der Programmablauf jedoch nicht handschriftlich zu erstellen, sondern kann mittels Blöcke zusammengesetzt werden. Dies erfordert zwar auch eine gewisse Programmierkenntnis, ist jedoch einfacher zu verstehen. Außerdem sind viele Hilfeleistungen gegeben. Ein wichtiger Knackpunkt sind jedoch wie bei fast allen Programmiersprachen die Datentypen, welche sauber deklariert werden müssen.

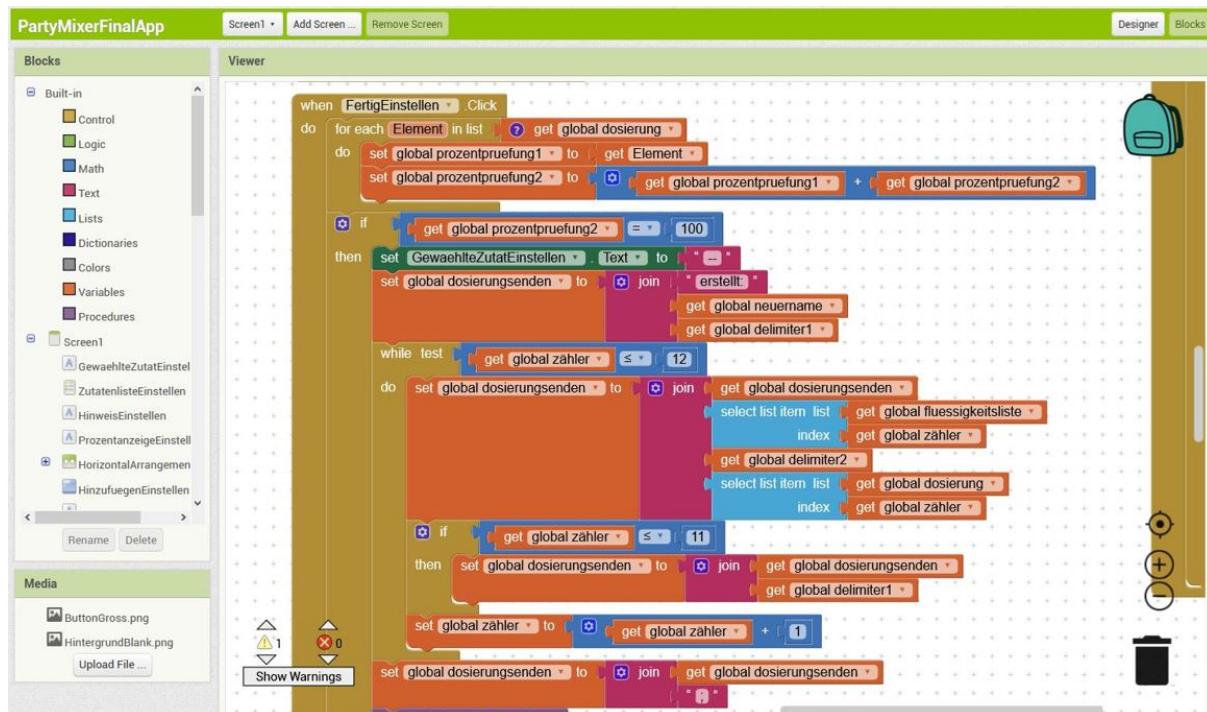


Abbildung 7.16: Blocks von AppInventor [50]

Die App funktioniert auf allen Android-Geräten, welche den Grossteil des weltweiten Marktes ausmachen. AppInventor ist jedoch an einer Kooperation mit Apple dran, um die erstellten App's auch auf iOS benutzen zu können. Dies befindet sich momentan in der Betaphase. Somit sollte die App in naher Zukunft auch auf iOS verfügbar sein können. [52]

#### 7.4.1 PartyMixer App

Damit die App so benutzerfreundlich wie möglich ist, wurde sie in einem sehr schlichten Design erstellt. Das Design und die Funktionen erinnern an das Display des PartyMixers und helfen somit dem Kunden intuitiver damit arbeiten zu können. Es können in der App drei Dinge gemacht werden. Dabei wurde darauf geschaut, dass es sich niemals um Punkte handelt, welche vom App und Display gleichzeitig beeinflusst werden können. Somit wäre es zum Beispiel unpraktisch mit der App bereits erstellte Cocktails bearbeiten zu können, da man sich so mit der App und dem Display hinein pfuschen kann. Grundsätzlich gibt es zwei mögliche Einstellungen, welche vorgenommen werden können in der App und eine Information.

Als Erstes muss eine Bluetooth-Verbindung mit dem PartyMixer hergestellt werden. Dazu kann man mit dem Button «Bluetooth-Gerät wählen» ein Bluetoothgerät aus der eigenen Bluetooth-Liste gewählt werden. Beim Loslassen des Buttons erscheint somit eine Auswahlliste. Achtung: Das erste Mal muss das Gerät in den Geräteeinstellungen verbunden werden, bevor es in der Liste erscheint. Nach dem das Gerät mit der Maschine verbunden ist, gelangt man mittels Tastendruck auf «RFID» in die RFID-Einstellung, wo man einem spezifischen RFID-Tag ein Getränk zuweisen kann. Per Klick auf «RFID-Tag wählen» erscheint eine Scrollbare Liste mit 10 Tag's, wobei man seinen erhaltenen RFID-Tag auswählen kann. Nach der Auswahl schliesst sich die Liste wieder und dieser wird über dem Button angezeigt. Als nächstes muss man das gewünschte Getränk auswählen, welches man dem Tag zuordnen möchte. Dazu erscheint beim Klicken auf «Getränk wählen» die aktuelle Cocktailliste, welche in der Maschine gespeichert ist. Auch hier verschwindet die Liste beim Klick auf einen Cocktail und man gelangt zurück. Das Getränk wird wie der Tag über dem Button angezeigt. Ist man mit der Auswahl zufrieden, so kann man den Vorgang mit Speichen abschliessen und das Getränk ist dem Tag zugeordnet. Man kann nun weitere Tag's zuordnen oder mit dem Button «Zurück» zurück ins Hauptmenü gehen. Die Menübildschirme sind in Abbildung 7.17 zu sehen.



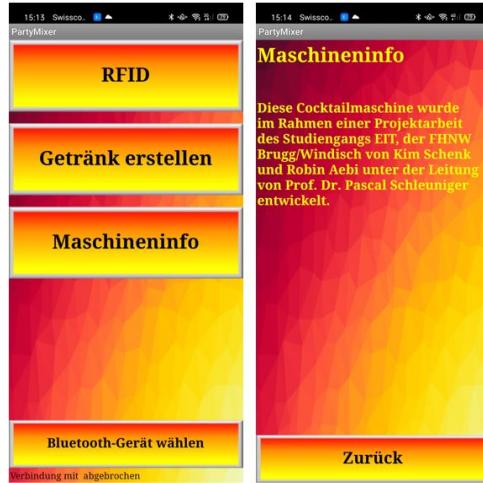
Abbildung 7.17: RFID Einstellungen der App

Eine weitere Einstellung, welche man vornehmen kann, ist die Erstellung eines eigenen Getränks, welches auf der Maschine gespeichert werden soll. Möchte man dies umsetzen, so gelangt man mit den Klick auf «Getränk erstellen» in die Namensvergabe, wo man dem neuen Cocktail einen Namen geben kann. Wenn kein Name eingetippt wird, so ist der Button «Weiter» ohne Funktion. Mit «Zurück» gelangt man erneut ins Hauptmenü. Per Klick auf das Textfeld erscheint die Telefontastatur und es kann ein Name eingegeben werden. Mit «Weiter» wird dieser gespeichert und man kommt in die Erstellungsanzeige. Da mit App Inventor keine dynamischen Buttons oder Slider erstellbar sind, musste nach einer anderen Lösung gesucht werden. Deshalb werden die Zutaten einzeln Ausgewählt. Mit Klick auf «Zutaten» erscheint die aktuelle Zutatenliste, wobei man seine erste Zutat auswählen kann, welche im Cocktail enthalten sein soll. Bei Auswahl verschwindet die Liste und die Zutat wird über dem Button angezeigt. Mit dem Slider kann nun in 5% Schritten die gewünschte Menge eingestellt werden, welche im Cocktail enthalten sein soll. Bei Klick auf «Gewählte Zutat Hinzufügen» wird die Auswahl abgespeichert, der Slider fällt auf 0 und es wird bei «Gewähltes Mischverhältnis» eine Liste angezeigt mit dem bereits getroffenen Einstellungen. Diesen Vorgang kann man so oft wiederholen, bis man 100% eingestellt hat. Wird weniger als 100% eingestellt, so erscheint bei Klick auf «Fertig» eine Meldung, dass bitte 100% eingestellt werden soll. Wird mit dem Slider mehr hinzugefügt, so dass mehr als 100% entsteht, so fällt dieser auf die verbleibende Menge bis 100% zurück und man kann diesen Wert hinzufügen. Will man eine Auswahl ändern, so muss man lediglich die zu ändernde Flüssigkeit auswählen und den neuen Wert abspeichern. Sind 100% eingestellt, so kann man mit «Fertig» den Cocktail abspeichern und man gelangt wieder ins Hauptmenü. Die Menübildschirme sind hierbei in Abbildung 7.18 zu sehen.



Abbildung 7.18: Getränk erstellen in der App

Mit einem Klick auf «Maschineninfo» im Hauptmenu gelangt man in die Infoanzeige, wo einem Informationen zur Maschine angezeigt werden. Mit einem Klick auf «Zurück» gelangt man erneut ins Hauptmenu. Die Menubildschirme dazu sind in Abbildung 7.19 zu sehen.



**Abbildung 7.19:** Infoanzeige der App

## 8 Zielerreichung

In diesem Kapitel werden die Ziele rekapituliert und aufgezeigt, was erreicht wurde. Dazu werden zuerst die gesetzten Pflichtziele und Wunschziele aufgelistet und mit Farben markiert, was erreicht wurde, was teilweise erreicht wurde und was nicht erreicht wurde. In einem weiteren Schritt wird dann detailliert auf diese Ziele eingegangen und aufgezeigt, was zusätzlich noch erreicht wurde und weshalb ein Ziel nut teilweise oder nicht erreicht wurde.

## 8.1 Pflichtzielerfüllung

Nummer	Pflichtziele	Anforderungen
1	Detailkonzept	<p>Das Detailkonzept besteht aus folgenden Komponenten:</p> <ul style="list-style-type: none"> <li>• Speisungen (48V, 12V, 5V, 3.3V)</li> <li>• Motor</li> <li>• ABN-Encoder</li> <li>• Endschalten</li> <li>• Motorentreiber</li> <li>• Gate-Treiber</li> <li>• Durchflussmessungen</li> <li>• Pumpen</li> <li>• Display</li> <li>• Mikrocontroller</li> <li>• Programmierschnittstellen</li> <li>• Bluetooth-/Wirelessmodul</li> <li>• Beleuchtung</li> </ul>
2	Design der Leiterplatte	Soll alle Teile des Detailkonzeptes umfassen. Für das Wifi-, RFID- und Motorentreibermodul wird eine Development-Board verwendet. Zusätzlich zu WiFi- und RFID-Modul wird eine eigen gelayoutete Variante miteinbezogen, welche bei genügend Kapazität implementiert wird anstelle des Moduls.
3	Mechanischer Aufbau der Maschine inkl. Achsensystem.	Der mechanische Aufbau beinhaltet folgende Teile: <ul style="list-style-type: none"> <li>• Rahmen</li> <li>• Getränkehälterung</li> <li>• Flüssigkeitsbeförderung</li> <li>• Gehäuse für Elektronik</li> <li>• Befestigung für Display</li> <li>• Glasbeförderungssystem</li> <li>• Überlaufwanne</li> <li>• Beleuchtung</li> </ul>
4	Regler Parametrisierung des Achsensystems	Die Regelung des Achsensystems wird mit dem FOC-Treiber gewährleistet. Die Regler werden so ausgelegt, dass das Glas während dem Fahren nicht überläuft. Die Bewegungsgeschwindigkeit soll trotzdem genügend schnell sein, dass der Drink in unter einer Minute hergestellt wird.
5	Bediensoftware	<p>Die Bediensoftware auf dem Mikrocontroller ermöglicht dem Benutzer folgende Eingaben:</p> <ul style="list-style-type: none"> <li>• Getränkliste mit 5 alkoholischen und 5 nicht alkoholischen Getränken, welche zur Auswahl stehen.</li> <li>• Infos zu den Getränken</li> <li>• Auswahl der Zubereitungsgröße von 3 oder 5dl</li> <li>• Nachfüllen des per Webservers (oder vorzugsweise per Android-App gemäß Wunschziel) eingestellten Getränkemittels RFID.</li> <li>• Reinigungsmodus</li> </ul> <p>Über einen Webserver (oder Android-App gemäß Wunschziel) kann der User folgende Einstellungen vornehmen:</p> <ul style="list-style-type: none"> <li>• Zuweisung eines RFID-Tags zu einem Benutzer</li> <li>• Auswahl des nächsten Getränkemäß der Getränkliste</li> </ul>

Nummer	Pflichtziele	Anforderungen
6	Funktionstest und Analyse bezüglich der Skalierbarkeit	In einer ersten wird der Print in Betrieb genommen. Dies bedeutet, dass die einzelnen Systeme mit Sonderprogrammen auf ihre Funktion geprüft werden. Dies beinhaltet die Systeme des Detailkonzeptes. In einer zweiten Phase wird die Maschine auf ihre Funktion geprüft. Dies soll die Funktionen beinhalten, welche in der Bediensoftware aufgelistet sind.
7	Software	Die Software für den Mikrocontroller soll in C geschrieben sein, für das ESP wird vorerst Arduino verwendet.
8	Getränkezubereitung	Die Abweichung der Flüssigkeitsausgabe darf höchstens 4% betragen.

## 8.2 Wunschzielerfüllung

Nummer	Wunschziele	Anforderungen
1	Lichtkonzept	Die Maschine bietet einen gewissen Showeffekt. Dazu wird ein LED-Band montiert, welcher die Maschine beleuchtet. Für die Maschine werden RGB-LEDs verwendet, was eine entsprechende Ansteuerung Hard- und Softwareseitig benötigt.
2	Software	<ul style="list-style-type: none"> <li>Die Software für das ESP soll in C geschrieben sein.</li> <li>Es soll vom Nutzer konfigurierbar sein, wo welches Getränk steht.</li> <li>Der Benutzer soll selber Cocktails individuell erstellen können.</li> <li>Individuelle Anpassungen der Mischverhältnisse der gespeicherten Getränke.</li> </ul>
3	Android-Applikation	<ul style="list-style-type: none"> <li>Anstelle des Webservers soll eine Android-App erstellt werden, welche über Bluetooth kommuniziert.</li> <li>Individuelle Erstellung von Cocktails in der App, gemäss Flüssigkeitsliste.</li> </ul>
4	Regler Parametrisierung des Achsensystems	<ul style="list-style-type: none"> <li>Das gewünschte Getränk soll in unter 40s zubereitet werden.</li> </ul>
5	Getränkezubereitung	<ul style="list-style-type: none"> <li>Die Abweichung der Flüssigkeitsabweichung darf höchstens 1% betragen.</li> </ul>

## 8.3 Zielerreichung

### Detailkonzept:

Alle im Detailkonzept erwähnten Komponenten wurden erfolgreich implementiert. Es wurde eine zusätzliche Komponente in das System integriert - Die SD-Karte.

### Design der Leiterplatte:

Die Leiterplatte umfasst alle im Detailkonzept erwähnten Schaltungen. Ausserdem sind für das WiFi-, RFID-, und FOC-Treibermodul Development-Boards verwendet worden. Das gelayoutete RFID-Modul wurde nicht in Betrieb genommen. Jedoch funktioniert das Development-Board einwandfrei. Das eigens gelayoutete Wireless-/Bluetoothmodul funktioniert reibungslos, wodurch auf dem DevKit verzichtet werden kann. Der FOC-Treiber sowie der Gate-Treiber inkl. H-Brücke mussten aufgrund layouttechnischer Problemen extern platziert werden. Die Funktion ist jedoch voll umfänglich gewährleistet.

### Mechanischer Aufbau der Maschine inkl. Achsensystem:

Alle mechanischen Komponenten sind implementiert worden. Für die Überlaufwanne wurde Platz geschaffen. Ausserdem wurde ein Kühlsystem entwickelt, welches die Getränke für 8 Stunden konstant bei 4-5°C kühl hält gemäss 3.5. Weiter wurde eine Wartungsklappe für die Pumpen und Durchflussmessgeräte implementiert. Das Schlittensystem verfügt zur besseren Justierung über einen Riemenspanner.

### Regler, Parametrisierung des Achsensystems:

Die Regler im System sind so ausgelegt, dass der Motor schnellst möglich eine Änderung annehmen kann. Dies ist für das System nicht optimal. Die Software-Ramp, welche den Weg unter Berücksichtigung einer Maximalgeschwindigkeit und -Beschleunigung im Voraus berechnet, schafft hier Abhilfe. Durch diese Software-Ramp überläuft das Glas beim Transport nicht.

### Bediensoftware:

Die Software beinhaltet standardmäßig 23 alkoholische und 6 alkoholfreie Cocktails. Diese Liste kann beliebig im Menü oder per App auf 100 Cocktails ergänzt werden. Die Getränkeinfo zeigt nicht nur das beinhaltende Getränk, sondern auch die vorhandene Menge in dl. Es kann zwischen 3dl und 5dl ausgewählt werden. Per Android-App oder Display kann ein RFID-Tag einem Getränk zugewiesen werden, welches am PartyMixer direkt herausgelassen werden kann.

### Funktionstest und Analyse bezüglich der Skalierbarkeit:

Der Print wurde erfolgreich in Betrieb genommen und die Funktionen getestet.

### Software:

Die Software ist in C geschrieben und das ESP wurde mit Arduino programmiert.

### Getränkezubereitung:

Die maximale Abweichung der Flüssigkeitsausgabe beträgt gemäss Kapitel 5.3.2 3.8%.

**Lichtkonzept:**

Die Maschine verfügt über einen RGB-LED-Controller, welcher am Display individuell vom Benutzer parametrisiert werden kann.

**Software:**

Die Software für das ESP32 wurde nicht in C geschrieben. Der Benutzer kann am Display selber entscheiden, welches Getränk wo steht und kann auch neue Flüssigkeiten erstellen und zuweisen. Außerdem kann der Benutzer selbst Cocktails am Display erstellen.

**Android-App:**

Es wurde erfolgreich eine Andoid-App implementiert, mit welcher der Benutzer RFID-Tags zuweisen und eigene Cocktails erstellen kann. Außerdem können die Maschineninfos aufgerufen werden.

**Regler Parametrisierung Achsensystems:**

Wenn alle Pumpen eingesetzt werden und die maximale Zeit für die Erstellung eines Cocktails ausgereizt wird, benötigt die Maschine 57 Sekunden. Wenn weniger Pumpen eingesetzt werden, sinkt diese Zeit. Da es jedoch keinen Cocktail gibt, der alle Zutaten enthält, ist dies kein realistischer Fall. Die Maschine kann jedoch einen Cocktail mit maximal vier Zutaten in unter 40 Sekunden erstellen.

**Getränkezubereitung:**

Die Abweichung der Flüssigkeitsausgabe beträgt bei einer Cocktaillerstellung mit mehr als einer Flüssigkeit über 1%.

## 9 Persönliches Schlusswort

Die ausführliche Recherchearbeit, welche im **Fachbericht 5** getätigter wurde, zahlte sich im Projekt 6 aus. Durch diese Recherche war es möglich viele Systeme sauber und relativ schnell einzubinden. Es wurde uns jedoch erst später bewusst, welchen Aufwand wir auf uns genommen hatten und was es bedeutet ein voll umfängliches Gerät mit Elektronik, Software und Mechanik zu konstruieren. Schlussendlich hatten wir einiges mehr als doppelt so viel Zeit und Arbeit hineingesteckt, als es für eine Bachelor Thesis geplant wäre. Dies ist jedoch nicht nur der Arbeit selbst zu verschulden, sondern auch weil wir uns das Ziel gesetzt hatten möglichst sauber und penibel zu arbeiten. Besonders die Entwicklung der Software und der Bedienmenüs, sowie der Mechanik nahmen sehr viel Zeit in Beanspruchung. Dafür wurden jedoch auch viele Extras eingebunden und es wurde über das eigentliche Ziel hinausgeschossen. Wir haben nicht nur alle Pflicht- und fast alle Wunschziele erreicht, sondern auch noch einiges mehr. Es wurde geschafft eine voll funktionsfähige Maschine zu entwickeln mit coolen Features. Daher denken wir, dass dies ein Punkt ist, auf den wir auch ein wenig stolz sein können.

Doch auch mit viel Frust mussten wir klarkommen. Der Motor funktionierte bis kurz vor Ende des Projektes nicht wirklich und wir spielten schon länger mit dem Gedanken den Motor vollkommen aufzugeben. Dies schaffte viel Unmut und wir mussten auch lernen miteinander in schwierigen Situationen umzugehen. Unsere Hartnäckigkeit zahlte sich jedoch am Schluss aus.

## 10 Ehrlichkeitserklärung

Mit der Unterschrift bestätigen die Verfasser dieser Bachelorthesis, dass die vorliegende Projektdokumentation selbstständig, im Team und ohne Verwendung anderer, als der angegebenen Hilfsmittel verfasst wurde, sämtliche verwendeten Quellen erwähnt und die gängigen Zitierregeln eingehalten wurden. Eine Überprüfung der Arbeit auf Plagiate mithilfe elektronischer Hilfsmittel darf vorgenommen werden.

Unterschrift:

---

Ort, Datum:

---

Unterschrift:

---

Ort, Datum:

---

## Literatur

- [1] C. Store. (). US \$1.43 |2020 aluminium Profil 6mm T Slot 2020 Aluminium Extrusion Eloxiert 100 200 300 400 500 600 800 100 0mm CNC 3D Drucker Teile 1m|Eckverbinder| - AliExpress, aliexpress.com. Library Catalog: de.aliexpress.com, Adresse: [https://de.aliexpress.com/item/33037478306.html?src=ibdm\\_d03p0558e02r02&sk=&aff\\_platform=&aff\\_trace\\_key=&af=&cv=&cn=&dp=](https://de.aliexpress.com/item/33037478306.html?src=ibdm_d03p0558e02r02&sk=&aff_platform=&aff_trace_key=&af=&cv=&cn=&dp=) (besucht am 15. Okt. 2020).
- [2] lientec-led. (). Nutenstein für design-profil, Lientec LED. Library Catalog: www.lientec-led.com, Adresse: <https://www.lientec-led.com/products/nutenstein-fur-designprofil-und-easymount> (besucht am 15. Okt. 2020).
- [3] LED-GLASS. (). L-Form Eckverbinder Winkelverbinder Winkel Aluminium 2020 20x20 mit Zubehör für 20 mm Konstruktionsprofile Systemprofile: Amazon.de: Baumarkt, Adresse: <https://www.amazon.de/Winkelverbinder-Zubeh%C3%B6r-Aluminium-Construction-Profiles/dp/B07BTHWS5D> (besucht am 15. Okt. 2020).
- [4] S. Store. (). 2020 neue 2 M/5 M GT2 Öffnen Rubber Zahnriemen 2GT 6mm Breite Für CNC 3D Drucker für Reprap Prusa i3|3D Druckerteile & Zubehör| - AliExpress, Adresse: <https://de.aliexpress.com/item/32972050265.html?spm=a2g0s.9042311.0.0.27424c4dyTFZ70> (besucht am 15. Okt. 2020).
- [5] L. SUPER 3D TECHNOLOGY CO. (). 2020 X achse V Slot 2040 Y achsen zahnriemen Stretch Begradiigen spanner Für Creality Ender 3 CR 10 10S tronxy 3d drucker Teile|3D Druckerteile & Zubehör| - AliExpress, Adresse: <https://de.aliexpress.com/item/4000127067311.html?spm=a2g0s.9042311.0.0.27424c4dyTFZ70> (besucht am 15. Okt. 2020).
- [6] ——, (). 1set 3D Drucker teile Openbuilds V gantry plat set spezielle rutsche platte pulley für 2020 /2040 V slot aluminium profile räder|3D Druckerteile & Zubehör| - AliExpress, Adresse: <https://de.aliexpress.com/item/32987326971.html?spm=a2g0s.9042311.0.0.27424c4dyTFZ70> (besucht am 15. Okt. 2020).
- [7] ——, (). 2GT/GT2 Timing Pulley16/20/30/40/60 zähne Bohrung 5/6. 35/8mm Synchron Räder Getriebe Breite 6mm 3D Drucker Reprap CNC Teile|3D Druckerteile & Zubehör| - AliExpress, Adresse: <https://de.aliexpress.com/item/32961817706.html?spm=a2g0s.9042311.0.0.27424c4dyTFZ70> (besucht am 15. Okt. 2020).
- [8] Mangomix, Cocktail, in Wikipedia, 2020-08-16 14:03, Page Version ID: 202831582, 16. Aug. 2020. Adresse: <https://de.wikipedia.org/w/index.php?title=Cocktail&oldid=202831582> (besucht am 14. Sep. 2020).
- [9] JLCPCB. (10. Sep. 2020). JLCPCB Capabilities, Adresse: <https://jlcpcb.com/capabilities/Capabilities> (besucht am 10. Sep. 2020).
- [10] M. P. Systems. (13. Sep. 2011). MP24943, Adresse: <https://www.mouser.ch/datasheet/2/277/MP24943-1384201.pdf>.
- [11] Aliexpress. (). US \$1.94 35% OFF|DC12V 13,8 V 15V 18V 24V 27V 28V 30V 32V 36V 42V 48V 60V 360W 600W 1000W Schaltnetzteil Quelle Transformator AC DC SMPS-in Schaltnetzteil aus Heimwerkerbedarf bei AliExpress, aliexpress.com, Adresse: [https://de.aliexpress.com/item/32905696401.html?src=ibdm\\_d03p0558e02r02&sk=&aff\\_platform=&aff\\_trace\\_key=&af=&cv=&cn=&dp=](https://de.aliexpress.com/item/32905696401.html?src=ibdm_d03p0558e02r02&sk=&aff_platform=&aff_trace_key=&af=&cv=&cn=&dp=) (besucht am 11. Jan. 2020).
- [12] A. Store. (). US \$6.37 15% OFF|AIYIMA 12 V DC mini wasserpumpe Trinkwasser brunnen elektrische pumpen Korrosion widerstand für tee maschine Wasserfilter-in Pumpen aus Heimwerkerbedarf bei AliExpress - 11.11\_Doppel-11Tag der Singles, aliexpress.com, Adresse: <https://de.aliexpress.com/item/32836659386.html?spm=a2g0s.9042311.0.0.27424c4d9xQTPG> (besucht am 18. Okt. 2019).

- [13] Mouser. (). MP24943DN-LF Monolithic Power Systems (MPS) | Mouser, Mouser Electronics, Adresse: <https://www.mouser.ch/ProductDetail/946-MP24943DNLF> (besucht am 11. Jan. 2020).
- [14] Atmel. (Feb. 2014). Atmel ATmega640/v-1280/v-1281/v-2560/v-2561/v, Adresse: [https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561\\_datasheet.pdf](https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf) (besucht am 14. Sep. 2020).
- [15] TRINAMICMotion Control GmbH & Co. KG. (12. Nov. 2019). TMC6200 Datasheet Rev. 1.05, Adresse: [https://www.mev-elektronik.com/files/MEV%20Elektronik%20Service/produkt-downloads/motion-control/TMC6200\\_datasheet.pdf](https://www.mev-elektronik.com/files/MEV%20Elektronik%20Service/produkt-downloads/motion-control/TMC6200_datasheet.pdf) (besucht am 11. Sep. 2020).
- [16] P. Stahl. (4. Aug. 2014). Simulation und Implementierung einer asynchronen Point to Point Steuerung, Simulation und Implementierung einer asynchronen Point to Point Steuerung, Adresse: [https://edoc.sub.uni-hamburg.de//haw/volltexte/2015/2884/pdf/Bachelorarbeit\\_Patrick\\_Stahl.pdf](https://edoc.sub.uni-hamburg.de//haw/volltexte/2015/2884/pdf/Bachelorarbeit_Patrick_Stahl.pdf) (besucht am 17. Sep. 2020).
- [17] B. Kleitz, sec 11.4 Switch Debouncing, 2011. Adresse: <https://www.youtube.com/watch?v=uQTPZoDbfUs> (besucht am 11. Sep. 2020).
- [18] Andy aka. (10. Juni 2013). Capacitor - what is tau in this very simple circuit?, Electrical Engineering Stack Exchange, Adresse: <https://electronics.stackexchange.com/questions/72185/what-is-tau-in-this-very-simple-circuit> (besucht am 21. Okt. 2020).
- [19] S. Akm, „Servomotoren AKM“, S. 258, 23. Juli 2020. Adresse: [https://www.sigmatek-automation.com/fileadmin/user\\_upload/downloads/Servomotoren-AKM.pdf](https://www.sigmatek-automation.com/fileadmin/user_upload/downloads/Servomotoren-AKM.pdf) (besucht am 11. Sep. 2020).
- [20] CUI Devices. (10. Apr. 2019). CUI AMT33S-V Datasheet Rev 1.01, Adresse: <https://www.cuidevices.com/product/resource/amt33.pdf> (besucht am 11. Sep. 2020).
- [21] F. +. T. Store. (). US \$1.97 28% OFF|C18 Neue Heiße 1pc Wasser Kaffee Flow Sensor Schalter Meter Durchflussmesser Zähler 0,3 6L/min-in Durchfluss-Sensoren aus Werkzeug bei AliExpress - 11.11\_Doppel-11Tag der Singles, aliexpress.com, Adresse: [https://de.aliexpress.com/item/32905517966.html?src=ibdm\\_d03p0558e02r02&sk=&aff\\_platform=&aff\\_trace\\_key=&af=&cv=&cn=&dp=](https://de.aliexpress.com/item/32905517966.html?src=ibdm_d03p0558e02r02&sk=&aff_platform=&aff_trace_key=&af=&cv=&cn=&dp=) (besucht am 21. Okt. 2019).
- [22] Espressif Systems. (17. Okt. 2016). ESP32 Schematic, Adresse: [https://dl.espressif.com/dl/schematics/ESP32-Core-Board-V2\\_sch.pdf](https://dl.espressif.com/dl/schematics/ESP32-Core-Board-V2_sch.pdf).
- [23] NXP Semiconductors N.V. (27. Apr. 2017). MFRC522 datasheet rev. 3.9, Adresse: <https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf> (besucht am 11. Sep. 2020).
- [24] behindthesciences. (27. Jan. 2017). RGB LED STRIP tutorial, Behind The Sciences, Adresse: <https://behindthesciences.com/electronics/rgb-led-strip-tutorial/> (besucht am 11. Sep. 2020).
- [25] theorycircuit.com. (8. Jan. 2018). Arduino micro SD card data logger, theoryCIRCUIT - Do It Yourself Electronics Projects, Adresse: <https://theorycircuit.com/arduino-micro-sd-card-data-logger/> (besucht am 11. Sep. 2020).
- [26] ionos.de. (15. Apr. 2020). FAT32, IONOS Digitalguide, Adresse: <https://www.ionos.de/digitalguide/server/knowhow/fat32/> (besucht am 11. Sep. 2020).
- [27] E. Milsch. (Feb. 2009). Aufbau des FAT32 Dateisystems, Adresse: <https://docplayer.org/78333139-Aufbau-des-fat32-dateisystems.html> (besucht am 11. Sep. 2020).
- [28] Verschiedene Autoren. (16. Juli 2019). AVRDUDE – Mikrocontroller.net, AVRDUDE, Adresse: <https://www.mikrocontroller.net/articles/AVRDUDE> (besucht am 1. Mai 2020).

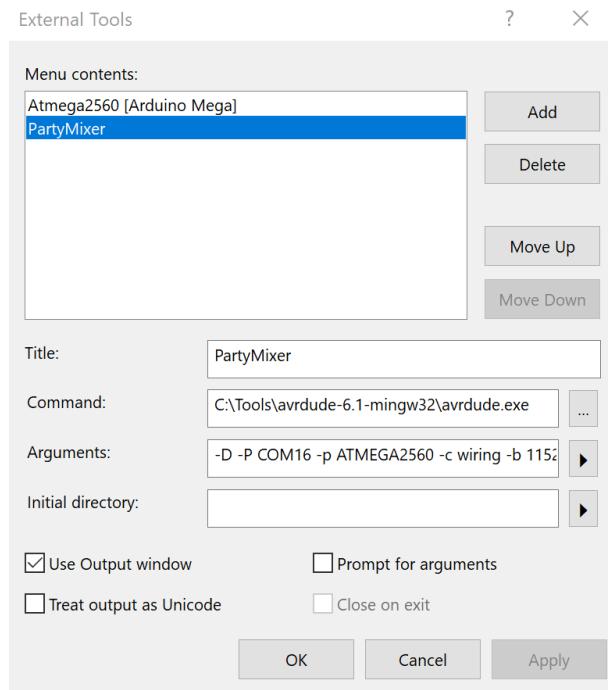
- [29] VidmoFollow. (10. Jan. 2017). Turn arduino's serial converter into AVRISP MkII clone, Instructables, Adresse: <https://www.instructables.com/id/Turn-Ardubinos-Serial-Converter-Into-AVRISP-MkII-C1/> (besucht am 12. Sep. 2020).
- [30] savannah.gnu.org. (17. Feb. 2016). Index of /releases/avrdude/, Index of /releases/avrdude/, Adresse: <http://download.savannah.gnu.org/releases/avrdude/> (besucht am 12. Sep. 2020).
- [31] M. Meier, mc1-Skript, 31. Okt. 2017.
- [32] I. Grokhovskiy, K. Sovani, M. Jain, M. N. Dev, A. Gratton und J. Domburg, espressif/arduino-esp32, original-date: 2016-10-06T06:04:20Z, 12. Sep. 2020. Adresse: <https://github.com/espressif/arduino-esp32> (besucht am 12. Sep. 2020).
- [33] R. Santos. (16. Juli 2018). ESP32 web server - arduino IDE | random nerd tutorials, Adresse: <https://randomnerdtutorials.com/esp32-web-server-arduino-ide/> (besucht am 12. Sep. 2020).
- [34] NXP B.V. 2010. (11. Okt. 2010). Antenna design guide for MFRC52x, PN51x and PN53x, Adresse: <https://my.eng.utah.edu/~mlewis/ref/NFC/AN1445.pdf>.
- [35] pcbreflux, ESP32 #33: RFID Read and Write with MFRC522 Module + ESP32 Giveaway, 18. Apr. 2017. Adresse: [https://www.youtube.com/watch?v=waYM37\\_fUsg](https://www.youtube.com/watch?v=waYM37_fUsg) (besucht am 14. Sep. 2020).
- [36] R. Aeby und K. Schenk. (21. Okt. 2020). Projekt-6/Software/Atmega at master · vwgolf-3/Projekt-6, Adresse: <https://github.com/vwgolf-3/Projekt-6/tree/master/Software/Atmega> (besucht am 21. Okt. 2020).
- [37] T. Hammer. (). HTerm, der-hammer, Adresse: <http://www.der-hammer.info/pages/terminal.html> (besucht am 12. Sep. 2020).
- [38] W. Wagner und A. Prüß, „The IAPWS formulation 1995 for the thermodynamic properties of ordinary water substance for general and scientific use“, Journal of Physical and Chemical Reference Data, Jg. 31, Nr. 2, S. 387–535, Juni 2002, ISSN: 0047-2689, 1529-7845. DOI: 10.1063/1.1461829. Adresse: <http://aip.scitation.org/doi/10.1063/1.1461829> (besucht am 14. Sep. 2020).
- [39] Mikrochip Makes, Getting Started with AVR: Using PWM to Dim an LED (#9), 12. März 2015. Adresse: [https://www.youtube.com/watch?v=FE4KfnEHPGU&list=PLtQdQmNK\\_ODRhBWYZ32BEIL0ykXLpJ8tP&index=9](https://www.youtube.com/watch?v=FE4KfnEHPGU&list=PLtQdQmNK_ODRhBWYZ32BEIL0ykXLpJ8tP&index=9) (besucht am 14. Sep. 2020).
- [40] —, Getting Started with AVR: Updating PWM Duty Cycle Using a Millisecond Timer (#10), 12. März 2015. Adresse: [https://www.youtube.com/watch?v=eXwv7e1B2QI&list=PLtQdQmNK\\_ODRhBWYZ32BEIL0ykXLpJ8tP&index=10](https://www.youtube.com/watch?v=eXwv7e1B2QI&list=PLtQdQmNK_ODRhBWYZ32BEIL0ykXLpJ8tP&index=10) (besucht am 14. Sep. 2020).
- [41] —, Getting Started with AVR: A More Complete PWM Driver (#11), 12. März 2015. Adresse: [https://www.youtube.com/watch?v=uA410-EC\\_Mo&list=PLtQdQmNK\\_ODRhBWYZ32BEIL0ykXLpJ8tP&index=11](https://www.youtube.com/watch?v=uA410-EC_Mo&list=PLtQdQmNK_ODRhBWYZ32BEIL0ykXLpJ8tP&index=11) (besucht am 14. Sep. 2020).
- [42] O. Geißler. (31. Juli 2018). Was ist ein Application-Programming-Interface (API)?, Was ist ein Application-Programming-Interface (API)?, Adresse: <https://www.datacenter-insider.de/was-ist-ein-application-programming-interface-api-a-735797/> (besucht am 17. Sep. 2020).
- [43] M. Lenz. (5. März 2016). Artikel | Doppelt verkettete Listen in C, Doppelt verkettete Listen in C, Adresse: <https://perlgeek.de/de/artikel/doppelt-verkettete-listen> (besucht am 17. Sep. 2020).
- [44] P. Yadav. (14. Okt. 2019). Circular doubly linked list in javascript, LearnersBucket, Adresse: <https://learnersbucket.com/tutorials/data-structures/circular-doubly-linked-list-in-javascript/> (besucht am 17. Sep. 2020).
- [45] Patrick. (). NX8048t070, Nextion, Adresse: <https://nextion.tech/datasheets/nx8048t070/> (besucht am 11. Jan. 2020).

- [46] B. Zhou. (). Nextion software, Nextion. Library Catalog: nextion.tech, Adresse: <https://nextion.tech/nextion-editor/> (besucht am 3. Okt. 2020).
- [47] pngimg.com. (). Cocktail PNG image with transparent background, pngimg.com, Adresse: <https://pngimg.com/download/44711> (besucht am 14. Sep. 2020).
- [48] ——, (). Cocktail PNG image with transparent background, pngimg.com, Adresse: <https://pngimg.com/download/44699> (besucht am 14. Sep. 2020).
- [49] (). Cocktail PNG image with transparent background, pngimg.com, Adresse: <https://pngimg.com/download/44729> (besucht am 14. Sep. 2020).
- [50] AppInventor. (). MIT App Inventor for iOS version 0.9 takes off in TestFlight, Adresse: <https://appinventor.mit.edu/blogs/evan/2019/09/25/mit-app-inventor-ios-version-0-9> (besucht am 3. Okt. 2020).
- [51] Kodular. (). Kodular, Kodular. Library Catalog: www.kodular.io, Adresse: <https://www.kodular.io> (besucht am 3. Okt. 2020).
- [52] AppInventor. (). MIT App Inventor | Explore MIT App Inventor, Adresse: <https://appinventor.mit.edu/> (besucht am 3. Okt. 2020).
- [53] liudr. (17. Okt. 2017). Trying to design an ESP32 dev board - MicroPython Forum, Adresse: <https://forum.micropython.org/viewtopic.php?t=4607> (besucht am 17. Sep. 2020).
- [54] H. Haftmann. (). Programmierung des Speichers, Adresse: <https://www-user.tu-chemnitz.de/~heha/viewchm.php/hs/ATmegaX8.chm/28.htm> (besucht am 17. Sep. 2020).
- [55] TRINAMICMotion Control GmbH & Co. KG. (10. Apr. 2020). TMC4671-BOB Datasheet Rev1.2 HW1.4, Adresse: [https://www.trinamic.com/fileadmin/assets/Products/Eval\\_Documents/TMC4671-BOB\\_datasheet\\_Rev1.2\\_HW1.4.pdf](https://www.trinamic.com/fileadmin/assets/Products/Eval_Documents/TMC4671-BOB_datasheet_Rev1.2_HW1.4.pdf) (besucht am 11. Sep. 2020).
- [56] ABB. (2012). Application noteLinear and s-ramped velocity profiles, Adresse: <http://www.abbmotion.com/support/SupportMe/Downloads/DocsLib/AN00118-002%20Linear%20and%20s-ramped%20velocity%20profiles.pdf> (besucht am 20. Okt. 2020).
- [57] D. Collins. (2. Mai 2017). How to reduce jerk in linear motion systems, Linear Motion Tips, Adresse: <https://www.linearmotiontips.com/how-to-reduce-jerk-in-linear-motion-systems/> (besucht am 20. Okt. 2020).
- [58] C. Lewin. (). S-curve motion profiles - a deep dive, S-Curve Profiles Deep Dive, Adresse: <https://store.pmdcorp.com/resources/get/s-curve-profiles-deep-dive-article> (besucht am 20. Okt. 2020).

## A Programmierschnittstellen

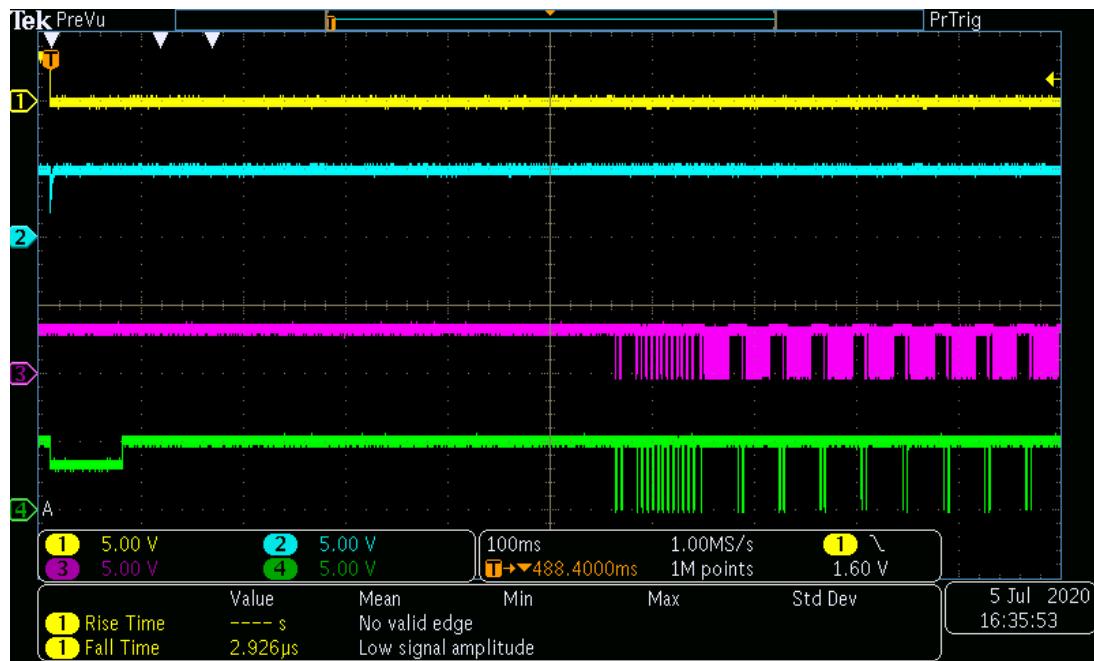
### A.1 Inbetriebnahme

#### A.1.1 AVRdude und stk500v2 (wiring)



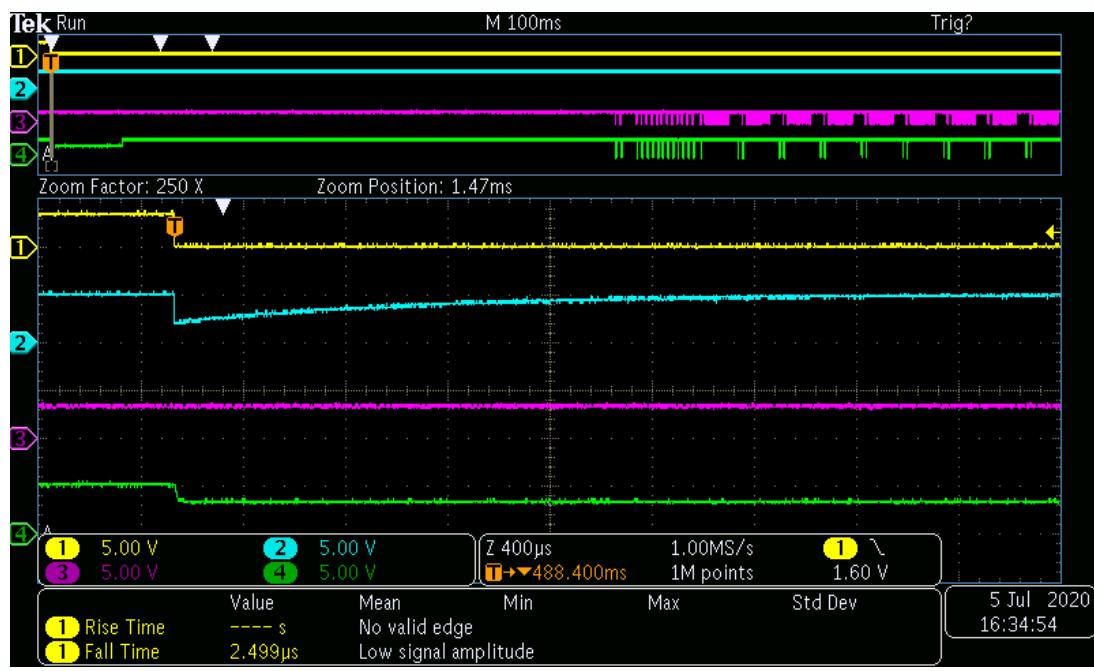
**Abbildung A.1:** External Tools ATMega2560 (Atmel Studio).

### A.1.2 Programmierschnittstelle Mikrocontroller



**Abbildung A.2:** Hochladen des Programmcodes auf den Mikrocontroller.

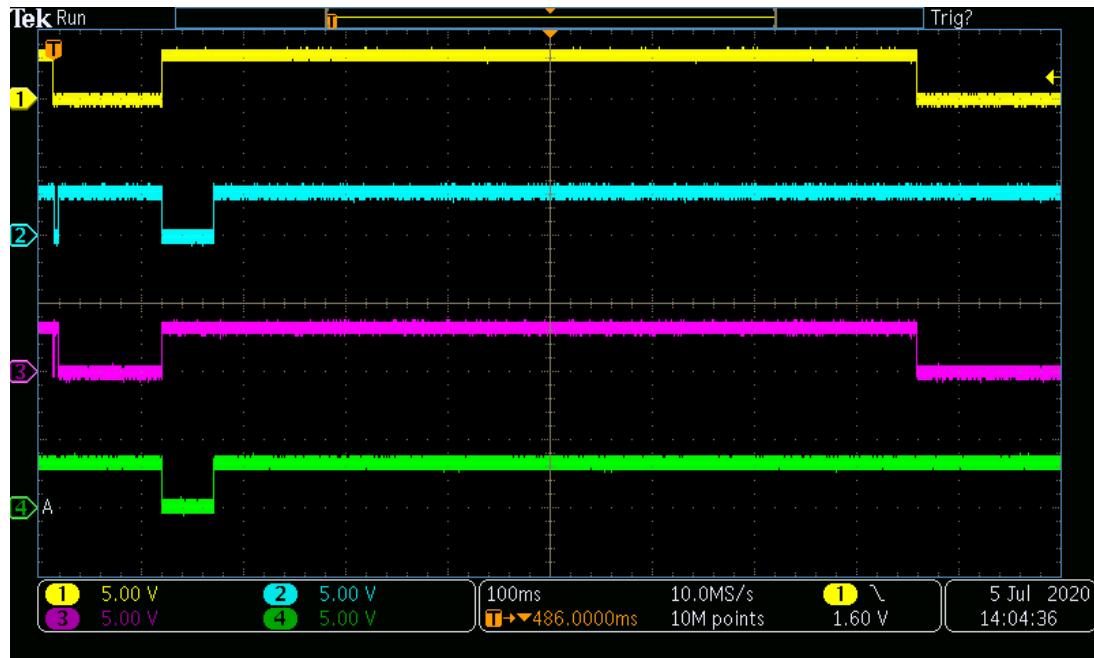
1: Gelb = DTR; 2: Blau = Reset; 3: Violett = RX; 4: Grün = TX



**Abbildung A.3:** Handshake zum Hochladen des Programmcodes auf den Mikrocontroller. Zoom auf den Moment des Resets.

1: Gelb = DTR; 2: Blau = Reset; 3: Violett = RX; 4: Grün = TX

### A.1.3 Programmierschnittstelle Wireless-/Bluetoothmodul



**Abbildung A.4:** Handshake zum Hochladen des Programmcodes auf das Wireless-/Bluetoothmodul.

1: Gelb = RTS; 2: Blau = DTR; 3: Violett = EN; 4: Grün = IO0;



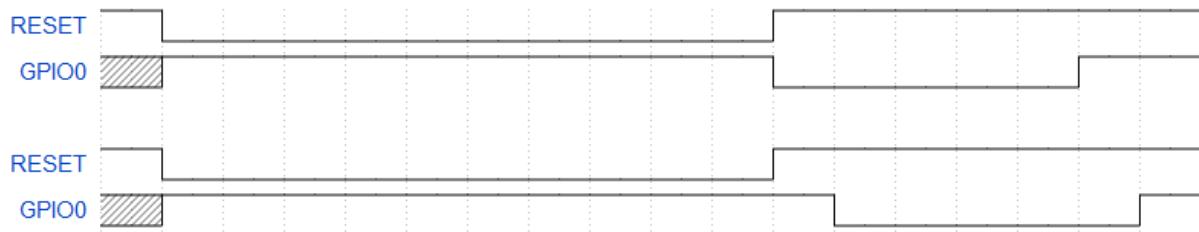
**Abbildung A.5:** Handshake zum Hochladen des Programmcodes auf das Wireless-/Bluetoothmodul. Zoom auf gleichzeitiger Flankenwechsel RTS und DTR (Schritt 2 und 3).

1: Gelb = RTS; 2: Blau = DTR; 3: Violett = EN; 4: Grün = IO0;

In Kapitel 6.2 wurde erwähnt, dass beim Handshake die Praxis der Theorie widerspricht. Eine Recherche ergab, dass einige User zum Ergebnis kamen wie bei der Inbetriebnahme. Nämlich,

dass es nach dem Reset eine Zeit dauert, bis im Startprozess die Pins geprüft werden. Dazu gehört auch der Pin IO0. Somit ist es möglich, den Pin IO0 kurz nach dem Reset auf 0 zu ziehen. Im selben Forum wurde auch die in Abbildung A.6 gezeigte Darstellung gefunden. Ein Kommentar weist darauf hin, das mit dem esptool.py der EN-Pin des Wireless-/Bluetoothmoduls direkt auf RTS gehängt werden kann. So lassen sich die Zeitpunkte, zu der die Pins auf 0 sind, näher zusammenschieben. [53]

Die Messung nach dem Einlöten der Brücke bestätigt dies. Abbildung A.7 zeigt, dass das Umschalten von EN und GPIO0 gleichzeitig passiert. Allerdings spielt der Kondensator jetzt nicht mehr so eine grosse Rolle, was auch nicht nötig ist. Auf das Hochladen des Codes hat die Brücke keinen Einfluss. Die Funktioniert wie bei der Schaltung ohne Brücke einwandfrei.



**Abbildung A.6:** Handshake zum Hochladen des Programmcodes auf das Wireless-/Bluetoothmodul. Zoom auf gleichzeitiger Flankenwechsel RTS und DTR. [53]



**Abbildung A.7:** Handshake zum Hochladen des Programmcodes auf das Wireless-/Bluetoothmodul. Zoom auf gleichzeitiger Flankenwechsel RTS und DTR.

## B Mikrocontroller

### B.1 Inbetriebnahme

#### B.1.1 Fuse- und Lock-Bits

Da während die Möglichkeit bestehen soll, den Flash-Speicher per USB zu beschreiben, muss beim Aufstarten der BL aufgerufen werden. Dazu muss das BOOTRST-Bit aktiviert werden. Der Speicherplatz für den BL wird auf 4096 words gesetzt, was mit dem Bootloader (SKT500v2) übereinstimmen muss. Das EEPROM soll beim Löschen des µC geschützt bleiben, weshalb das EESAVE-Bit aktiviert wird. Da die ISP-Schnittstelle benötigt wird, um den BL zu schreiben und Fuse-Bits zu setzen, wird das SPIEN-Bit gesetzt. Für den Mikrocontroller wird ein 16MHz Full-Swing-Crystal-Oszillatior verwendet, weshalb die Bits CKSEL3:1 auf 111 stehen müssen. Die Aufstartzeit wird vorsorglich auf die längst mögliche Zeit eingestellt. Dies führt dazu, dass das Register CKSEL0 auf 0 und die Register SUT0:1 auf 00 gesetzt werden. Der Brown-out-Detektor setzt den internen Reset, sobald die Versorgungsspannung unter einen Wert fällt. Wenn der Mikrocontroller ausfällt, während der FOC-Treiber noch aktiv ist, wird der Motor weiter gefahren. Dieses Risiko soll eingeschränkt werden, indem dieser Modus ausgeschaltet wird.[54]

Sind die Fuse-Bits gesetzt, wird via ISP-Programmer der Bootloader hochgeladen.

Die Lock-Bits müssen gesetzt werden, nachdem der BL in den Speicher geschrieben wurde. "SPM ist nicht erlaubt, in den Anwenderbereich zu schreiben und LPM ist nicht erlaubt aus dem Applikationssktor zu lesen, wenn LPM aus dem Urlader-Bereich ausgeführt wird." [54]

BODLEVEL 2:0 Fuses	Min. V <sub>BOT</sub>	Typ. V <sub>BOT</sub>	Max. V <sub>BOT</sub>	Units	
111	BOD Disabled				
110	1.7	1.8	2.0	V	
101	2.5	2.7	2.9		
100	4.1	4.3	4.5		
011					
010					
001	Reserved				
000					

Abbildung B.1: Tabelle Brown-out-Detection.[14, S.361]

Frequency Range [MHz]	CKSEL3:1	Recommended Range for Capacitors C1 and C2 [pF]
0.4 - 16	011	12 - 22

Abbildung B.2: Tabelle Frequenzbereich Crystal Oszillatior.[14, S.43]

Oscillator Source / Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset ( $V_{CC} = 5.0V$ )	CKSEL0	SUT1:0
Ceramic resonator, fast rising power	258 CK	14CK + 4.1ms <sup>(1)</sup>	0	00
Ceramic resonator, slowly rising power	258 CK	14CK + 65ms <sup>(1)</sup>	0	01
Ceramic resonator, BOD enabled	1K CK	14CK <sup>(2)</sup>	0	10
Ceramic resonator, fast rising power	1K CK	14CK + 4.1ms <sup>(2)</sup>	0	11
Ceramic resonator, slowly rising power	1K CK	14CK + 65ms <sup>(2)</sup>	1	00
Crystal Oscillator, BOD enabled	16K CK	14CK	1	01
Crystal Oscillator, fast rising power	16K CK	14CK + 4.1ms	1	10
Crystal Oscillator, slowly rising power	<u>16K CK</u>	<u>14CK + 65ms</u>	<u>1</u>	<u>11</u>

Abbildung B.3: Tabelle Aufstartzeit.[14, S.43]

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (Start Boot Loader Section)
1	1	512 words	4	0x0000 - 0x7DFF	0x7E00 - 0x7FFF	0x7DFF	0x7E00
1	0	1024 words	8	0x0000 - 0x7BFF	0x7C00 - 0x7FFF	0x7BFF	0x7C00
0	1	2048 words	16	0x0000 - 0x77FF	0x7800 - 0x7FFF	0x77FF	0x7800
0	0	<u>4096 words</u>	32	0x0000 - 0x6FFF	0x7000 - 0x7FFF	0x6FFF	0x7000

Abbildung B.4: Tabelle Bootloader Speicherplatz.[14, S.320]

Memory Lock Bits			Protection Type
BLB1 Mode	BLB12	BLB11	
1	1	1	No restrictions for SPM or (E)LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and (E)LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	(E)LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

Abbildung B.5: Tabelle Memory Lock.[14, S.326]

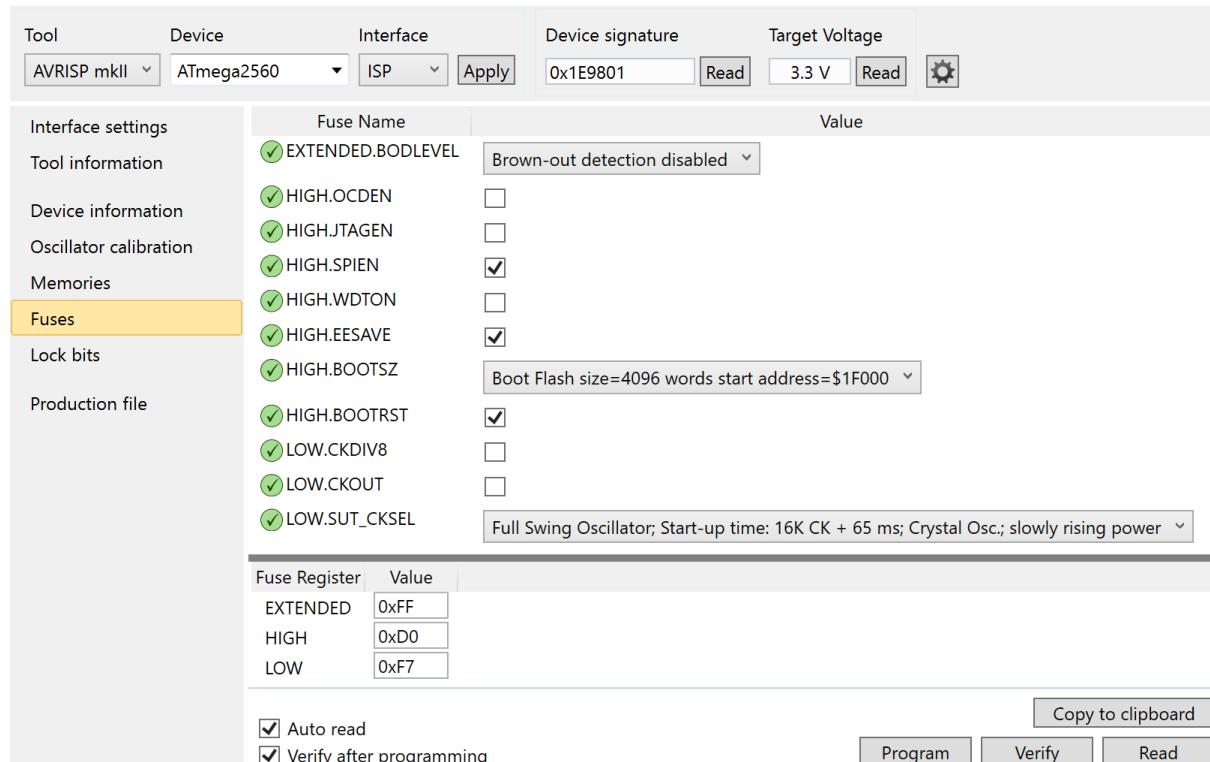


Abbildung B.6: Fuse-Bits ATMega2560 (Atmel Studio).

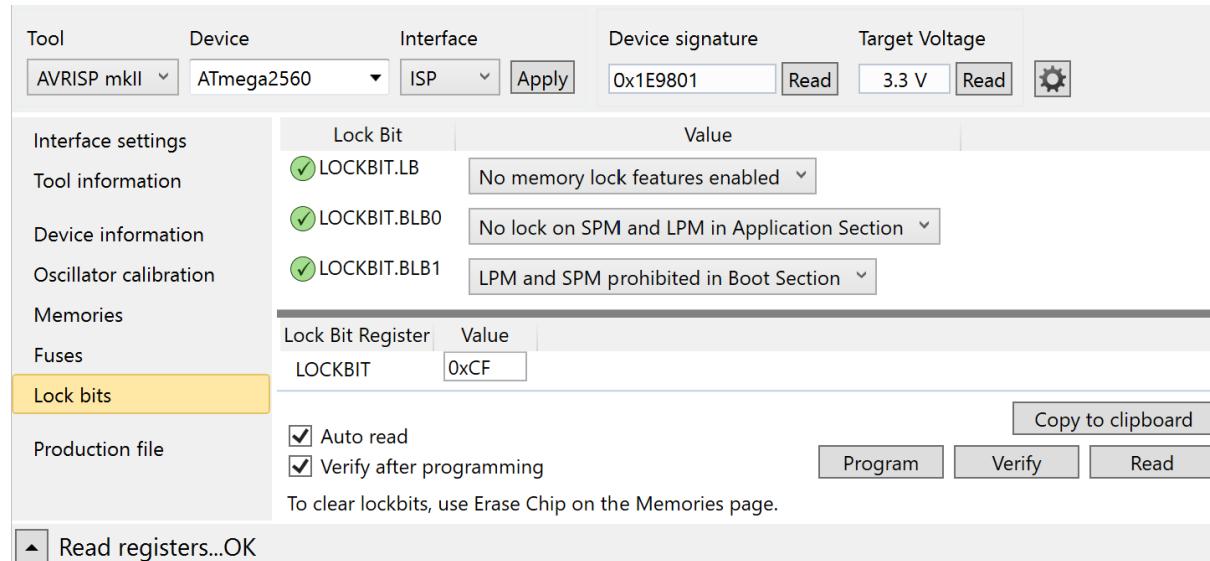


Abbildung B.7: Lock-Bits ATMega2560.

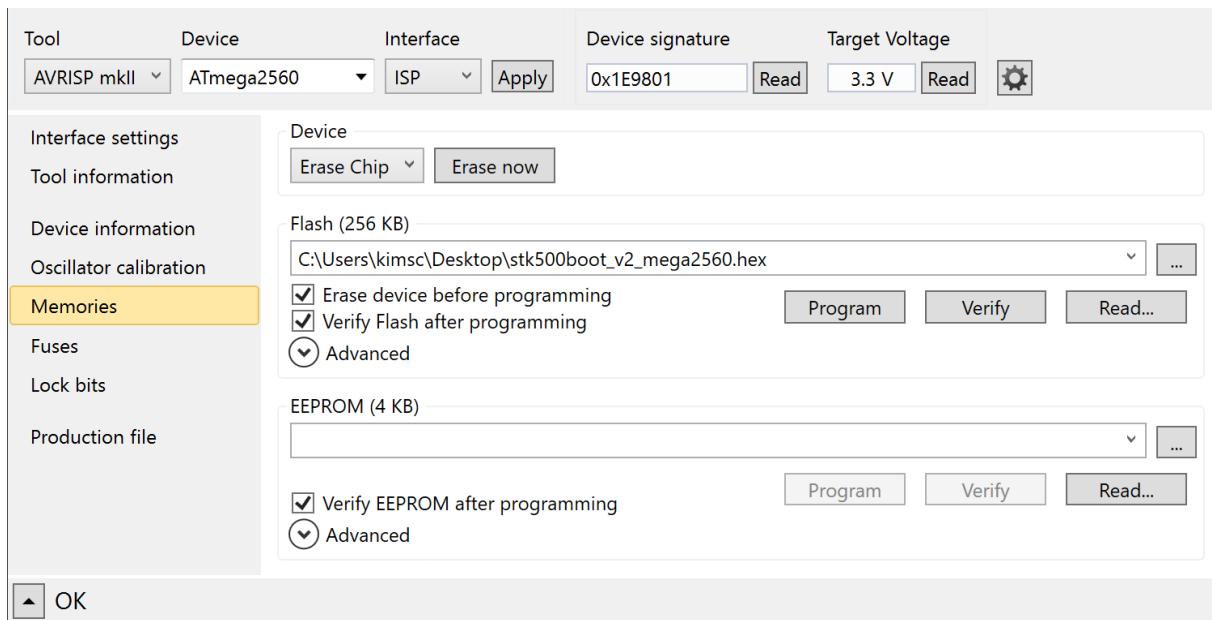


Abbildung B.8: Bootloader brennen (Atmel Studio).

### B.1.2 Pinout



## C Wireless-/Bluetoothmodul

### C.1 Inbetriebnahme

#### C.1.1 Arduino IDE

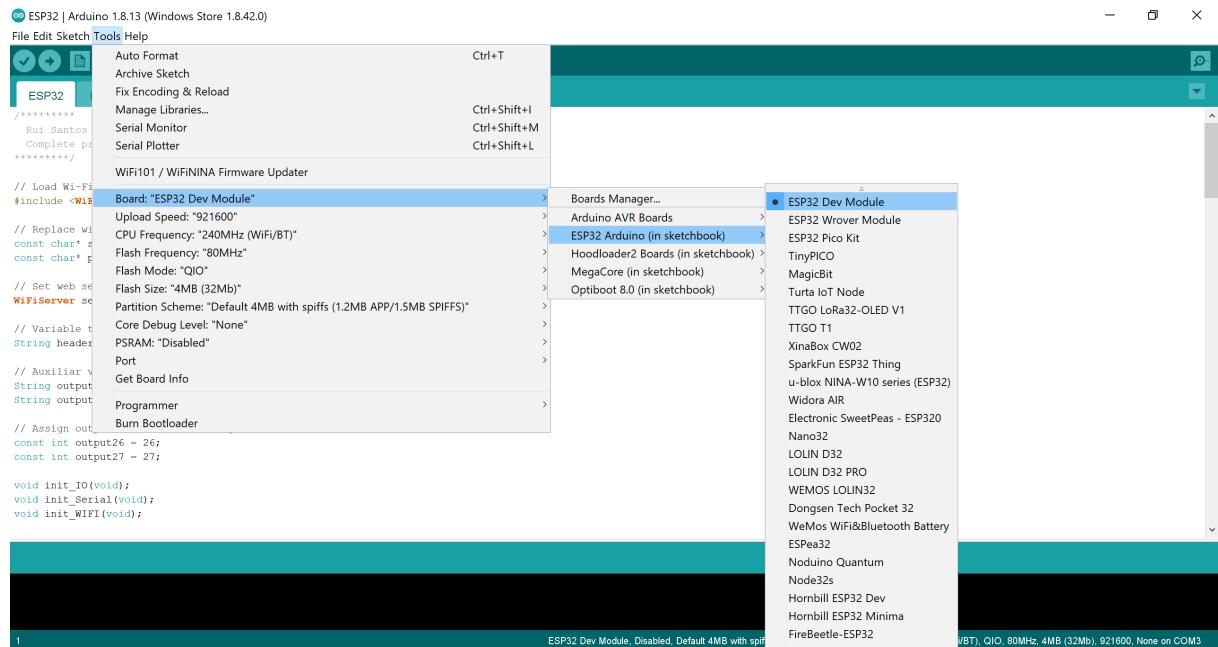


Abbildung C.1: ESP32 auswählen in der Arduino IDEs.

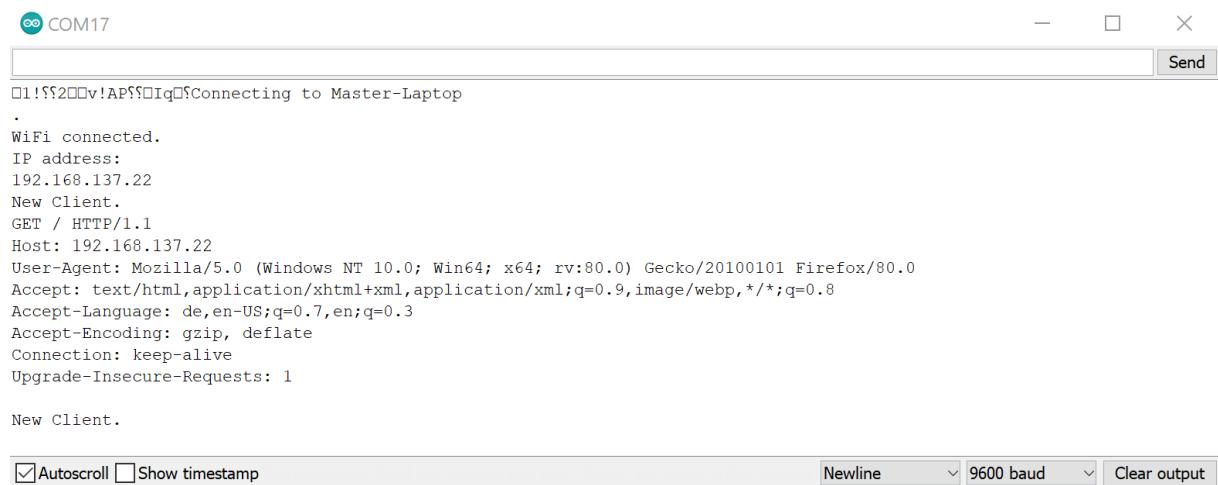
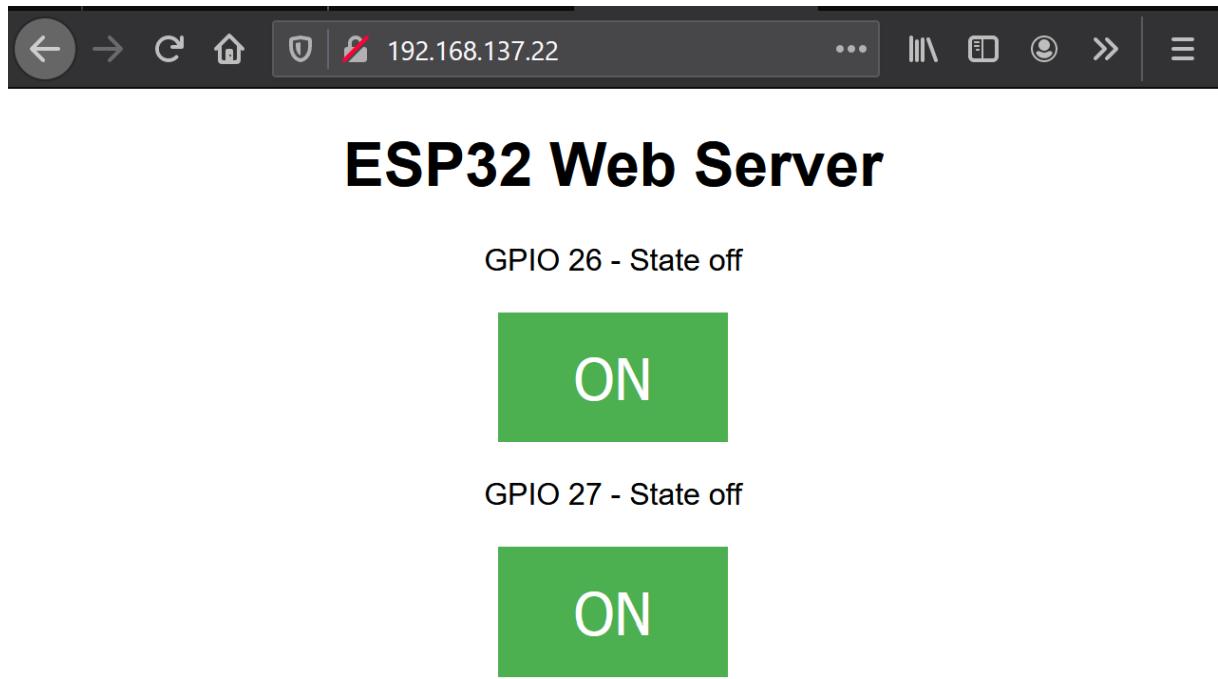


Abbildung C.2: Wireless-/Bluetoothmodul erkennt Netzwerk und Client (Browser).



**Abbildung C.3:** Wireless-/Bluetoothmodul als Host (Browseransicht).

## D LED

### D.1 Inbetriebnahme

#### D.1.1 PWM-Signale

Das PWM-Signal für die LEDs wird mit einer Frequenz von 10 kHz initialisiert. Um dies zu erreichen, wird Formel D.1 umgestellt nach D.2. Der Prescaler ist  $N = 1$ . Das Ergebnis wird im Register OCRnA gespeichert, wodurch der Timer  $1/10kHz = 100\mu s$  benötigt, bis ein Timer\_nA Compare-Interrupt ausgelöst wird. [14, S.148]

$$f_{OC_{nA}PWM} = \frac{f_{clk_{I/O}}}{N \cdot (1 + OCR_{nA})} \quad (D.1)$$

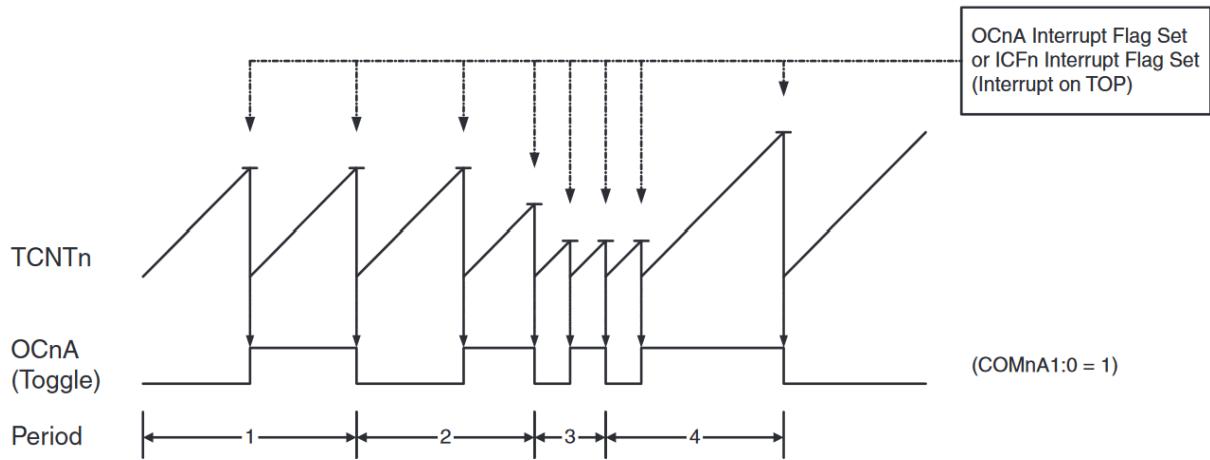
umgestellt nach OCR\_nA:

$$OCR_{nA} = \frac{f_{clk_{I/O}}}{N \cdot f_{OC_{nA}PWM}} - 1 = \frac{16MHz}{1 \cdot 10kHz} - 1 = 1599 \quad (D.2)$$

Damit das Hochzählen des Counters nach erreichen von OCRnA wieder bei null beginnt, wird der Timer im CTC-Mode betrieben. Das Register OCRnA stellt so den maximalen Wert des Counters dar. Dies ist in Abbildung D.1 und D.2 zu sehen.

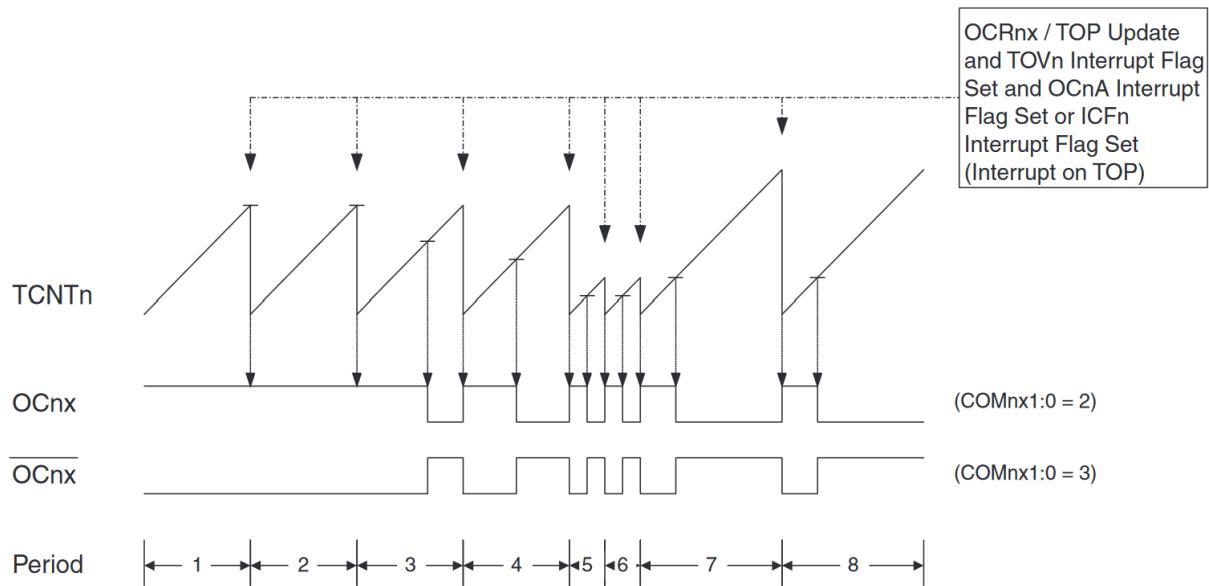
Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMMn1)	WGMn0 (PWMMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	-	-	-
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

**Abbildung D.1:** Timing-Diagramm CTC-Mode. OCnx nicht angeschlossen, Pin wird softwaremäßig getoggelt.[14, S.145]



**Abbildung D.2:** Timing-Diagramm CTC-Mode. OCnx nicht angeschlossen, Pin wird softwaremäßig getoggelt.[14, S.146]

Der Duty-Cycle wird Prozentual zum OCRnA-Register gesetzt. Wird für den berechneten Wert ein Duty-Cycle von 50% vorgegeben, ergibt sich für das OCRnB-Register den Wert  $(1600/2) - 1 \approx 799$ . So wird nach der Hälfte der Hochzählzeit ein zweiter Compare-Interrupt ausgelöst, welcher jedoch kein Einfluss auf das Counter-Register hat. Das Verhalten ist dann wie im Fast-PWM-Mode, welcher in Abbildung D.3 zu sehen ist. Nur dass anstelle von toggeln des OCnx-Pins softwaremäßig ein Pin getoggelt wird. Die Inbetriebnahme hat gezeigt, dass ein Duty-Cycle von 95% nicht überschritten werden sollte.



**Abbildung D.3:** Timing-Diagramm Fast-PWM-Mode. OCnx nicht angeschlossen, Pin wird softwaremäßig getoggelt.[14, S.147]

Nun muss in beiden der erwähnten Interrupt-Routinen das gewünschte LED getoggelt werden. In der ersten Interruptroutine mit dem OCRnA-Compare-Register wird die LED eingeschaltet, in der zweiten Interruptroutine mit dem OCRnB-Compare-Register wird die LED ausgeschaltet.

Möchte man nun die Helligkeit angepasst werden, kann ein Wert zwischen 0 und OCRnA ausgewählt werden und damit das Register OCRnB beschrieben werden.

## E FOC-Treiber

### E.1 Teilschemas Breakout-Board

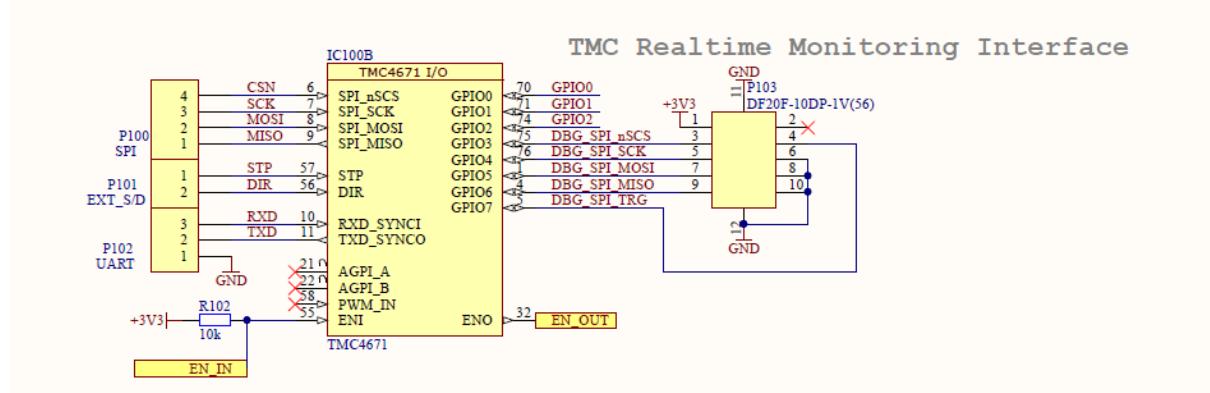


Abbildung E.1: SPI-Input des TMC4671-BOB. [55, S.2]

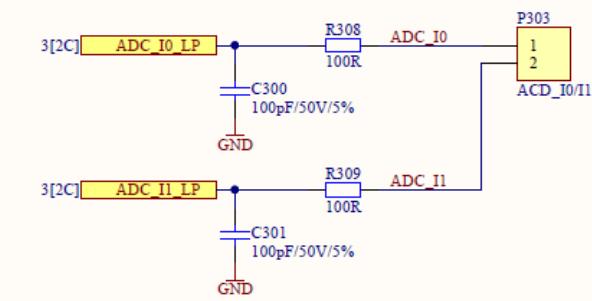


Abbildung E.2: Phasenstroeme-Input des TMC4671-BOB. [55, S.4]

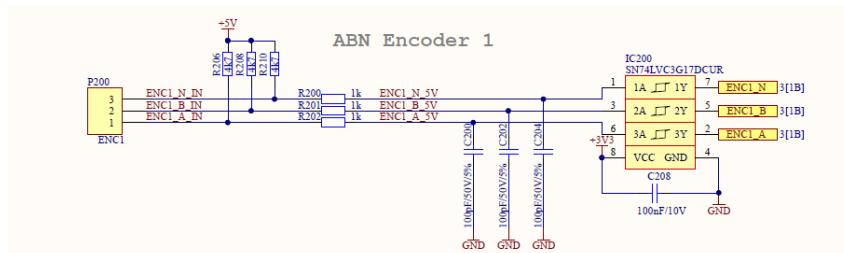


Abbildung E.3: ABN-Encoder-Input des TMC4671-BOB. [55, S.3]

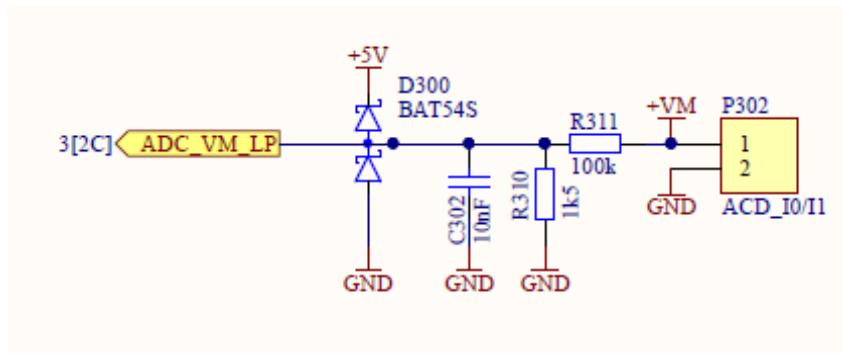


Abbildung E.4: Motorspannung-Input des TMC4671-BOB. [55, S.4]

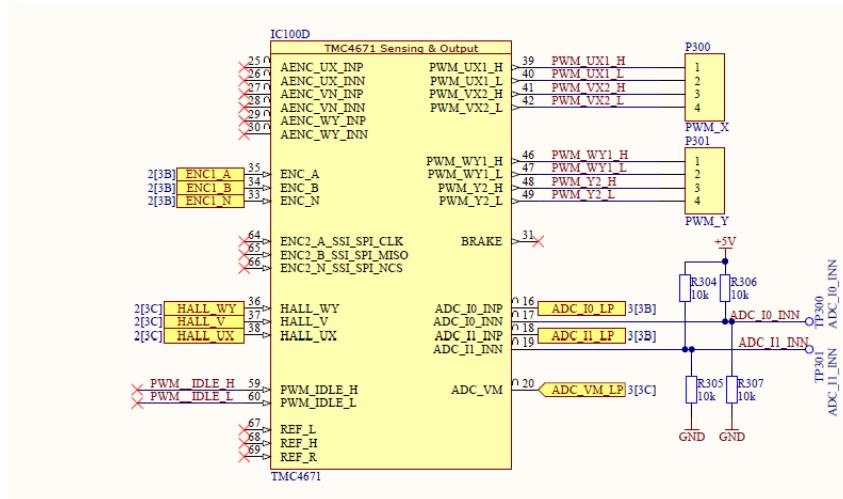


Abbildung E.5: PWM-Output des TMC4671-BOB. [55, S.3]

## E.2 Inbetriebnahme

### E.2.1 Inbetriebnahme SPI-Kommunikation

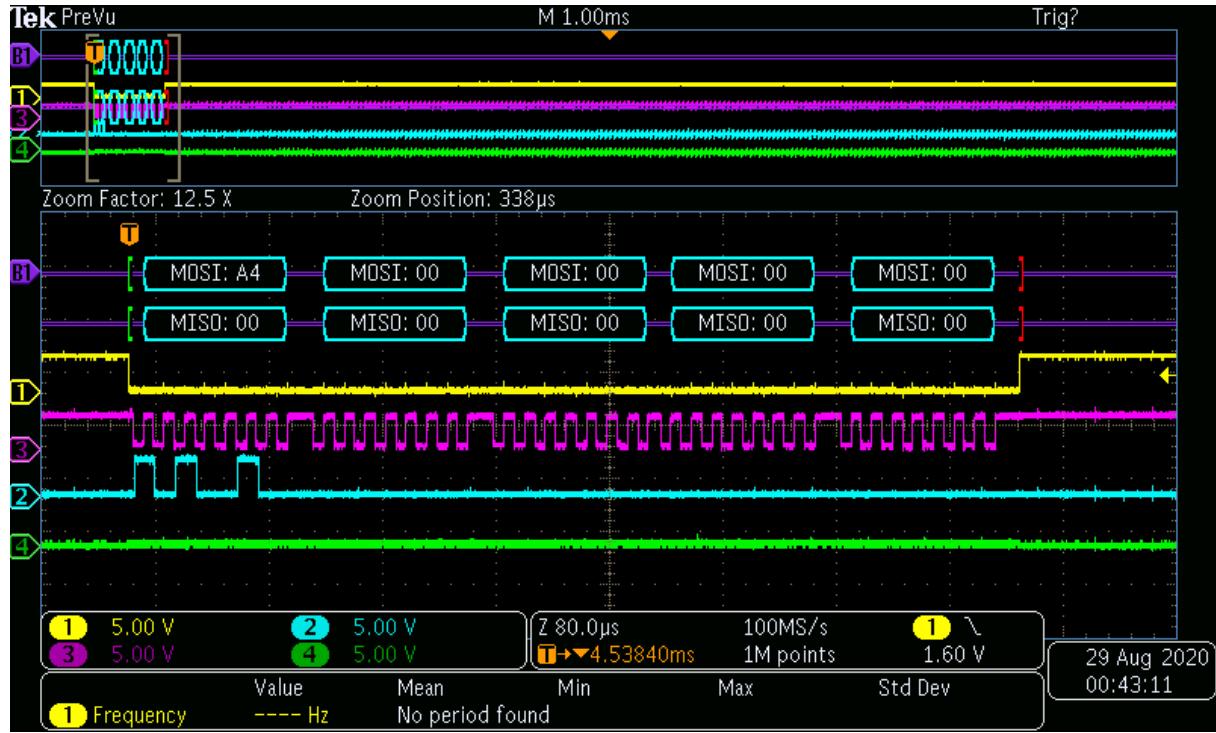


Abbildung E.6: Übertragung mit Zoom (Testdrive).

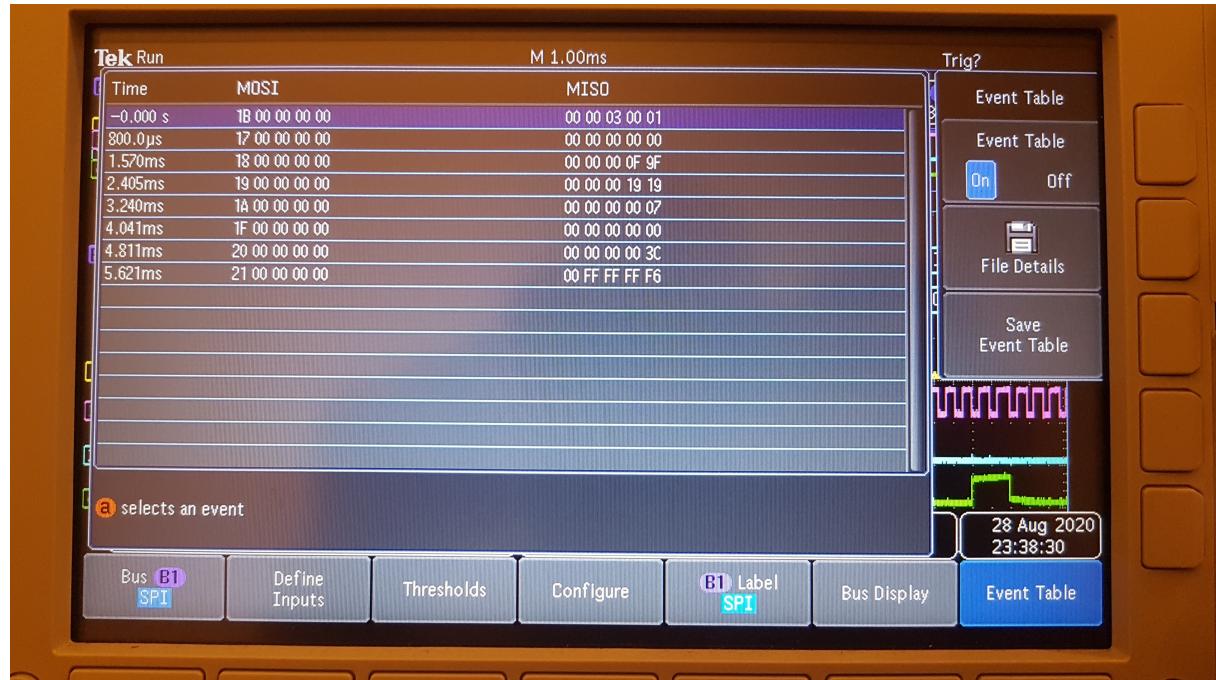


Abbildung E.7: Event-Table Inbetriebnahme FOC-Treiber.

### E.3 Register Openloop

FOC-Treiber Register			
Nr.	Register-Name	Wert	Was
<b>Motor type &amp; PWM configuration</b>			
0x17	PWM_POLARITIES	0x00000000	MOSFETs polarity "off"
0x1B	MOTOR_TYPE_N_POLE_PAIRS	0x00030003	
0x18	PWM_MAXCNT	0x00000F9F	PWM frequency 25kHz = 100MHz/(3999+1)
0x19	PWM_BBM_H_BBM_L	0x00001919	Break-before-Make-Time[10ns] MOSFET gate control
0x1A	PWM_SV_CHOP	0x00000007	PWM chopper mode: centtered PWM for FOC
<b>ADC configuration</b>			
0x0A	ADC_I_SELECT	0x24000100	ADC_I0_(RAW), ADC_I1_(RAW)
0x04	dsADC_MCFG_B_MCFG_A	0x00100010	(TMCL-IDE)
0x05	dsADC_MCLK_A	0x20000000	(TMCL-IDE)
0x06	dsADC_MCLK_B	0x00000000	(TMCL-IDE)
0x07	dsADC_MDEC_B_MDEC_A	0x014E014E	(TMCL-IDE)
0x09	ADC_I0_SCALE_OFFSET	0xFF0080CF	(TMCL-IDE)
0x08	ADC_I1_SCALE_OFFSET	0xFF008E04	(TMCL-IDE)
<b>Open loop settings</b>			
0x1F	OPENLOOP_MODE	0x00000000	Openloop Phi Direction positive
0x20	OPENLOOP_ACCELERATION	0x0000003C	Openloop Phi Beschleunigung = 60
0x21	OPENLOOP_VELOCITY_TARGET	0xFFFFFFF8	Openloop Phi Geschwindigkeit = 5
<b>Feedback selection</b>			
0x52	PHI_E_SELECTION	0x00000002	Phi-E selector = Phi-E Openloop
0x24	UQ_UD_EXT	0x00000FA0	FOC q/d = 4000
<b>Switch to open loop velocity mode</b>			
0x63	MODE_RAMP_MODE_MOTION	0x00000008	Modulation gem. UQ/UD ext.
<b>Rotate right</b>			
0x21	OPENLOOP_VELOCITY_TARGET	0x0000003C	Openloop Phi Geschwindigkeit = 60

## F Gate-Treiber

### F.1 Standard-Schaltkreis

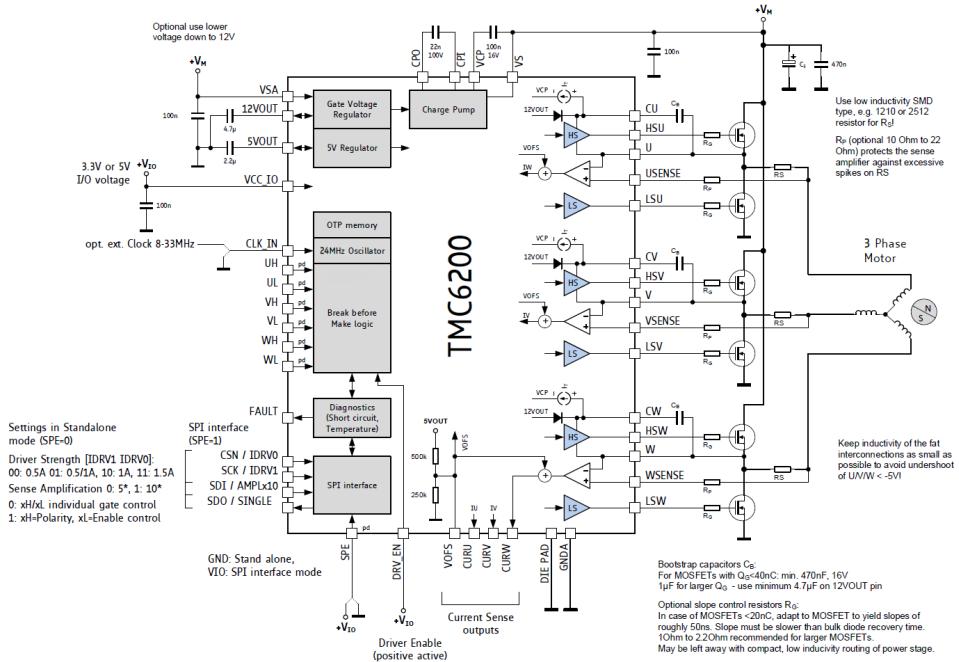


Abbildung F.1: Standard-Anwendungs-Schaltung Gate-Treiber.[15, S.1]

### F.2 Blockdiagramm

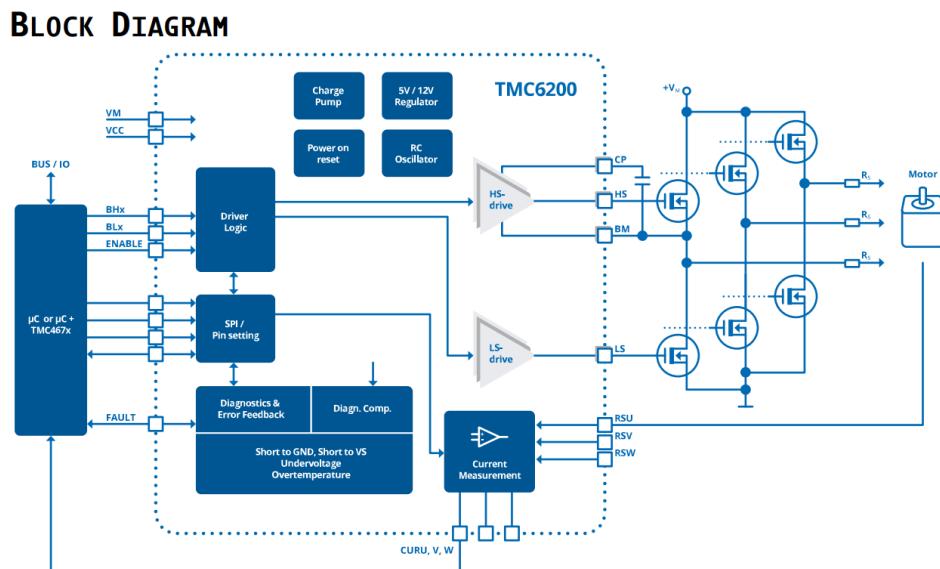


Abbildung F.2: Blockdiagramm Gate-Treiber.[15, S.5]

## F.3 Inbetriebnahme

### F.3.1 Verstärkungsfaktor, Strommessung, Strommesswiderstand

CHOICE OF $R_{SENSE}$ AND AMPLIFICATION DEPENDING ON MAX. COIL CURRENT				
$R_{SENSE}$ [ $m\Omega$ ]	Amplification factor	Current range [A]	RMS motor current limit [A]	Max. power dissipation of $R_{SENSE}$ [W]
150	10	0.7	0.5	0.05
150	5	1.3	1	0.15
100	5	2	1.5	0.23
75	5	2.6	2	0.3
33	10	3	2.2	0.16
25	10	4	3	0.23
50	5	4	3	0.45
33	5	6	4.5	0.67
15	10	6.5	5	0.38
25	5	8	6	0.9
10	10	10	7.5	0.56
5	10	20	15	1.1
2.5	20	20	15	0.56
2.5	10	40	30	2.3
1	20	50 (40@1.65V ofs.)	37	1.4

**Abbildung F.3:** Tabelle zur Bestimmung des Strommesswiderstandes aus dem Datenblatt von Trinamic.[15, S.31]

### F.3.2 Gate-Vorwiderstand

MOSFET MILLER CHARGE VS. DRVSTRENGTH AND $R_G$		
Miller Charge [ $nC$ ] (typ.)	DRVSTRENGTH setting	Value of $R_G$ [ $\Omega$ ]
<10	0 or 1	$\leq 10$ (recommended)
10...20	0 to 2	$\leq 5$ (optional)
20...80	1 to 3	$\leq 2.5$ (optional)
>80	3	$\leq 1$ (optional)

**Abbildung F.4:** Tabelle zur Bestimmung der Gatewiderstände aus dem Datenblatt von Trinamic.[15, S.13]

### F.3.3 Inbetriebnahme SPI-Kommunikation

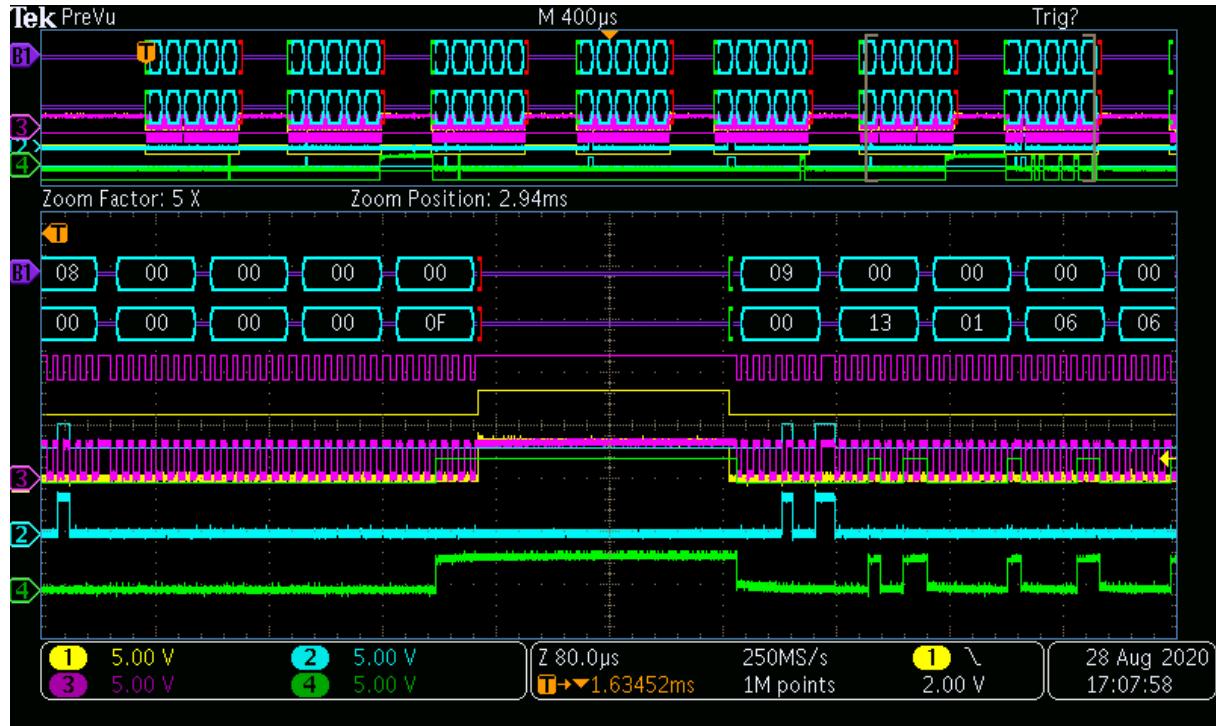


Abbildung F.5: SPI-Übertragung Read.



Abbildung F.6: Event-Table Inbetriebnahme Gate-Treiber read.

#### F.4 Register Gate-Treiber

Gate-Treiber Register			
Nr.	Register-Name	Wert	Was
0x00	GCONF	0x00000010	Current Amplification *10
0x01	GSTAT	0x00000000	Resets eventual faults from restart.
0x06	OTP_PROG	0x00000000	(TMCL-IDE)
0x08	FACTORY_CONF	0x0000000F	Clock Frequency (TMCL-IDE)
0x09	SHORT_CONF	0x13010606	Security settings (TMCL-IDE)
0x0A	DRV_CONF	0x00080004	Treiberstärke medium (TMCL-IDE)

## G ABN-Encoder

### G.1 Register Closed-Loop

FOC-Treiber Register			
Nr.	Register-Name	Wert	Was
<b>Motor type &amp; PWM configuration</b>			
0x17	PWM_POLARITIES	0x00000000	MOSFETs polarity "off"
0x1B	MOTOR_TYPE_POLE_PAIRS	0x00030003	BLDC / 3 Polapairs
0x18	PWM_MAXCNT	0x00000F9F	PWM frequency 25kHz = 100MHz/(3999+1)
0x19	PWM_BBM_H_BBM_L	0x00001919	Break-before-Make-Time[10ns] MOSFET gate control
0x1A	PWM_SV_CHOP	0x00000007	PWM chopper mode: centtered PWM for FOC
<b>ADC configuration</b>			
0x0A	ADC_I_SELECT	0x24000100	ADC_I0_(RAW), ADC_I1_(RAW)
0x04	dsADC_MCFG_B_MCFG_A	0x00100010	(TMCL-IDE)
0x05	dsADC_MCLK_A	0x20000000	(TMCL-IDE)
0x06	dsADC_MCLK_B	0x00000000	(TMCL-IDE)
0x07	dsADC_MDEC_B_MDEC_A	0x014E014E	(TMCL-IDE)
0x09	ADC_I0_SCALE_OFFSET	0xFF0080CF	(TMCL-IDE)
0x08	ADC_I1_SCALE_OFFSET	0xFF008E04	(TMCL-IDE)
<b>ABN-Encoder settings</b>			
0x25	ABN_DECODER_MODE	0x0000100A	Pins = A,B,N, polarities = 0,1,0, direction = 1
0x26	ABN_DECODER_PPR	0x00002000	Pulses per mechanical revolution = 8192
0x27	ABN_DECODER_COUNT	0x00000000	Decoder count start = 0
0x29	PHI_E_PHI_M_OFFSET	0x00000000	Angle = 0
<b>Limits</b>			
0x5C	PID_TARGET_DDT_LIMITS	0x00007FFF	$\frac{di}{dt}$ Änderung Drehmoment/Fluss
0x5D	PIDOUT_UQ_UD_LIMITS	0x00005A81	Regelfehler UQ/UD 23'169
0x5E	PID_TORQUE_FLUX_LIMITS	0x000009C4	Drehmoment/Fluss = 2500mA
0x5F	PID_ACCELERATION_LIMIT	0x000000C8	Beschleunigung = 200
0x60	PID_VELOCITY_LIMIT	0x0000005DC	Geschwindigkeit = 1500
0x61	PID_POSITION_LIMIT_LOW	0x80000001	Unten = -2'147'483'649
0x62	PID_POSITION_LIMIT_HIGH	0x7FFFFFFF	Oben = 2'147'483'648
<b>PI-Settings</b>			
0x56	TORQUE_P_TORQUE_I	0x00642EE0	Drehmoment: P = 100 / I = 12'000
0x54	FLUX_P_FLUX_I	0x00642EE0	Fluss : P = 100 / I = 12'000
0x58	VELOCITY_P_VELOCITY_I	0x07D001F4	Geschwindigkeit: P = 2000 / I = 500
0x5A	POSITION_P_POSITION_I	0x00C80000	Position : P = 200 / I = 0

## H Software Ramp

### H.1 Implementierung

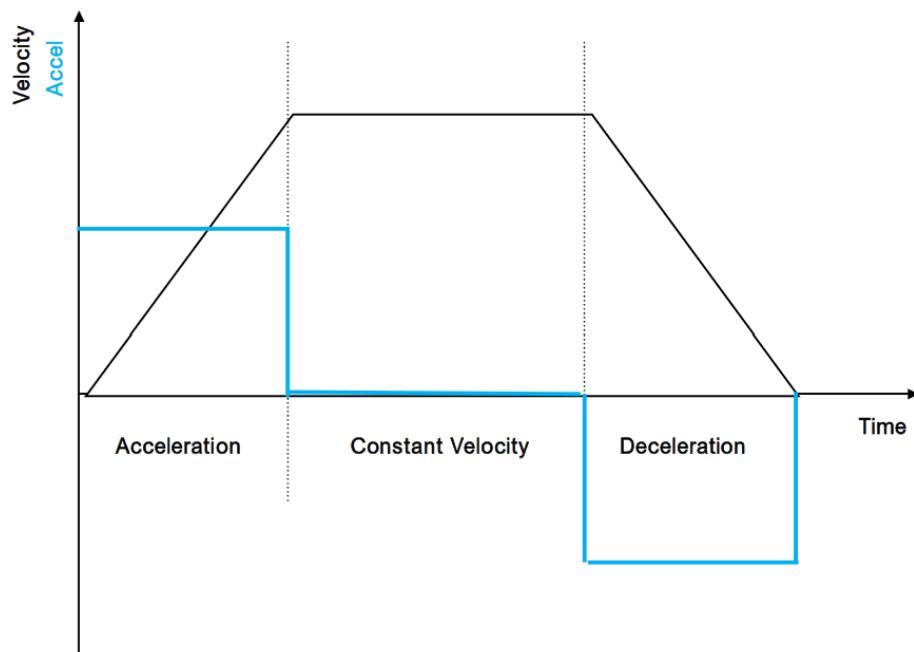
In einer triangulären/trapezodialen Geschwindigkeits-Ramp ist die Bewegung in drei Phasen unterteilt. Die erste Phase besteht aus einer Beschleunigung in eine Richtung. Die Beschleunigung dauert so lange, bis das angetriebene Objekt die Maximalgeschwindigkeit erreicht. Sobald die Maximalgeschwindigkeit erreicht wurde, beginnt die zweite Phase. Während dieser Phase bewegt sich das Objekt mit konstanter Geschwindigkeit in die selbe Richtung wie die Beschleunigung. In der dritten Phase wird das Objekt wieder abgebremst mit einer Beschleunigung in die entgegengesetzte Richtung.

Die Formeln zur Berechnung der mechanischen Zustände sind für die Beschleunigung H.1, für die Geschwindigkeit H.2 und für die Position H.3. Je nach dem, in welche Richtung gefahren werden soll, wird das Vorzeichen der Beschleunigung geändert. Dafür wird beim FOC-Treiber die aktuelle Position abgefragt und berechnet, ob der Zielpunkt sich vor oder nach der aktuellen Position befindet.

$$a(t) = a_0 \quad (\text{H.1})$$

$$v(v) = v_0 + a_0 \cdot t \quad (\text{H.2})$$

$$x(t) = x_0 + v_0 \cdot t + a_0 \cdot t^2 \quad (\text{H.3})$$



**Abbildung H.1:** Trapezodiale Geschwindigkeits-Ramp. [56]

Phase	$a_0$	$v_0$	$x_0$
1 ( $T_0 - T_1$ )	$+a_{max}$	0	0
2 ( $T_1 - T_2$ )	0	$v_{max}$	$x(t[n-1])$
3 ( $T_2 - T_3$ )	$-a_{max}$	$v_{max}$	$x(t[n-1])$

**Tabelle H.1:** Parameter für trapezodiale Ramp.

Eine weitere Möglichkeit, den Motor saft anlaufen zu lassen ist die S-Curve-Ramp. Diese besteht im Gegensatz zur trapezodialen Ramp statt aus drei Phasen aus sieben Phasen.

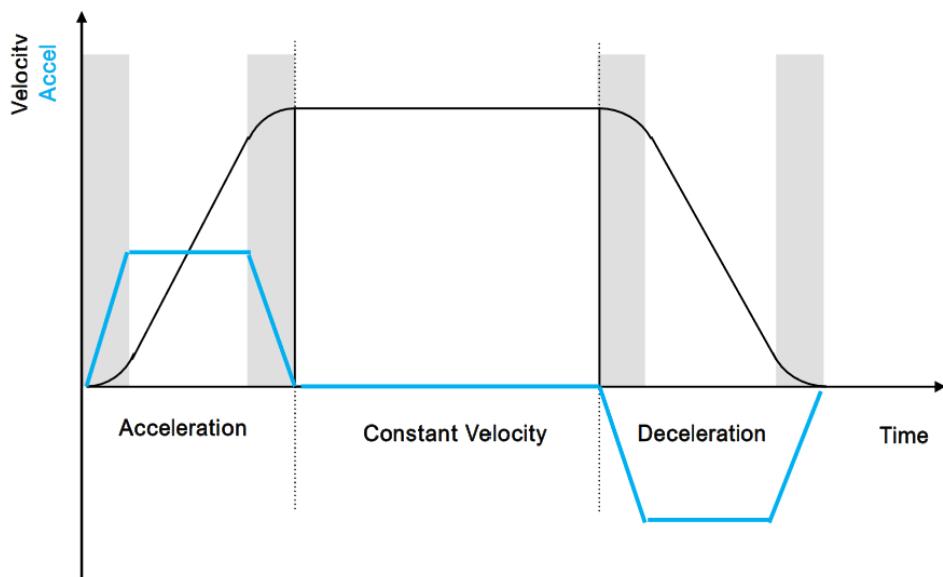
Die vier zusätzlichen Phasen kommen daher, dass jetzt die Beschleunigung nicht von Beginn weg voll anliegt, sondern auch linear hochgefahren wird. Die Zunahme der Beschleunigung wird auch Jerk genannt. [57]

$$J(t) = J_0 \quad (\text{H.4})$$

$$a(t) = a_0 + J_0 \cdot t \quad (\text{H.5})$$

$$v(t) = v_0 + a_0 \cdot t + J_0 \cdot t^2 \quad (\text{H.6})$$

$$x(t) = x_0 + v_0 \cdot t + a_0 \cdot t^2 + J_0 \cdot t^3 \quad (\text{H.7})$$

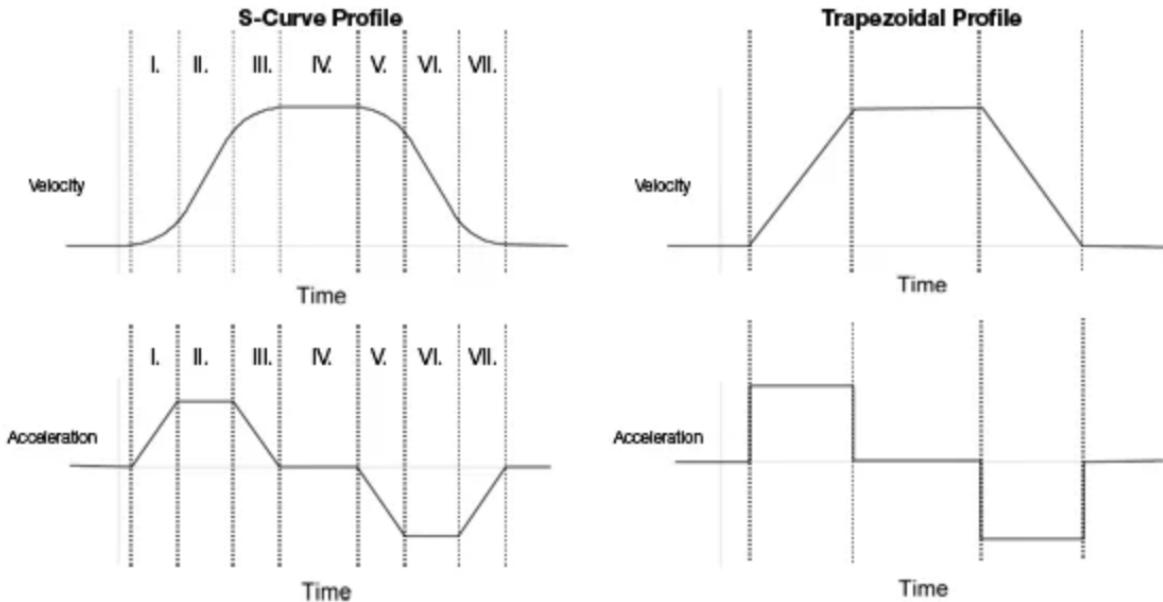
**Abbildung H.2:** S-Curve Geschwindigkeits-Ramp.[56]

Phase	$J_0$	$a_0$	$v_0$	$x_0$
1 ( $T_0 - T_1$ )	J	0	0	0
2 ( $T_1 - T_2$ )	0	a	$v_{endPhase1}$	$x_{endPhase1}$
3 ( $T_2 - T_3$ )	-J	a	$v_{endPhase2}$	$x_{endPhase2}$
4 ( $T_3 - T_4$ )	0	0	$v_{endPhase3}$	$x_{endPhase3}$
5 ( $T_4 - T_5$ )	-J	0	$v_{endPhase4}$	$x_{endPhase4}$
6 ( $T_5 - T_6$ )	0	-a	$v_{endPhase5}$	$x_{endPhase5}$
7 ( $T_6 - T_7$ )	J	-a	$v_{endPhase6}$	$x_{endPhase6}$

**Tabelle H.2:** Parameter für S-Curved Ramp.

Die S-Curved-Ramp ist nicht implementiert und stellt einen Ansatz zur Verbesserung dar. Folgende Schritte können zu einer Implementierung führen:

- Strecke und Zeit messen für gesamten Beschleunigungsvorgang mit Jerk.
- Daraus kann ermittelt werden, ob das zu bewegende Objekt schon über das Ziel herauschiesst.
- Falls es darüber herauschiesst, muss der Beschleunigungsvorgang angepasst werden.
- Der Vorgang erreicht die Höchstgeschwindigkeit dann nicht.
- Danach kann der Algorithmus in die Software eingepasst werden.

**Abbildung H.3:** Vergleich S-Curve und trapezodial. [58]

## H.2 Matlab-Script

# Ramp Calculation

```
clear  
close
```

## Inputs

```
V_max = 200;  
X_end = 5000;  
A_max = 20;  
Jerk = 10;  
Te=1e-2; % interpolation time
```

## Funktionen

```
% Jerk  
j = @(J) J  
  
j = function_handle with value:  
    @(J)  
  
% Beschleunigung  
a = @(a_0, J, t) a_0 + J * t  
  
a = function_handle with value:  
    @(a_0,J,t)a_0+J*t  
  
% Geschwindigkeit  
v = @(v_0, a_0, J, t) v_0 + a_0 * t + 1/2 * J * t.^2  
  
v = function_handle with value:  
    @(v_0,a_0,J,t)v_0+a_0*t+1/2*J*t.^2  
  
% Position  
x = @(x_0, v_0, a_0, J, t) x_0 + v_0 * t + 1/2 * a_0 * t.^2 + 1/6 * J * t.^3  
  
x = function_handle with value:  
    @(x_0,v_0,a_0,J,t)x_0+v_0*t+1/2*a_0*t.^2+1/6*J*t.^3  
  
% Zeit während Jerk  
a1_0 = 0;  
v1_0 = 0;  
x1_0 = 0;  
  
if Jerk > 0  
    T_j_1 = A_max/Jerk;  
else  
    T_j_1 = 0;  
end  
  
t(1) = 0;  
t(2) = T_j_1;  
t1 = t(1):Te:t(2);  
x_jerk_1 = vpa(x(x1_0, v1_0, a1_0, Jerk, t1));  
v_jerk_1 = v(v1_0, a1_0, Jerk, t1);  
a_jerk_1 = vpa(a(a1_0, Jerk, t1));  
j_jerk_1 = Jerk + zeros(1,length(t1));  
  
% Zeit volle Beschleunigung  
x2_0 = x_jerk_1(length(x_jerk_1));  
v2_0 = v_jerk_1(length(v_jerk_1));  
a2_0 = A_max;  
  
T_a_1 = (V_max - 2 * v_jerk_1(length(v_jerk_1)))/A_max;  
t(3) = T_a_1;  
t2 = t(1):Te:t(3);  
  
x_acceleration_1 = x(x2_0, v2_0, a2_0, 0, t2);  
v_acceleration_1 = v(v2_0, a2_0, 0, t2);  
a_acceleration_1 = a(a2_0, 0, t2);  
j_acceleration_1 = 0 + zeros(1,length(t2));  
  
% Zeit Jerk2  
x3_0 = x_acceleration_1(length(x_acceleration_1));  
v3_0 = v_acceleration_1(length(v_acceleration_1));
```

```

a3_0 = A_max;

T_j_2 = T_j_1;
x_jerk_2 = x(x3_0, v3_0, a3_0, -Jerk, t1);
v_jerk_2 = v(v3_0, a3_0, -Jerk, t1);
a_jerk_2 = a(a3_0, -Jerk, t1);
j_jerk_2 = -Jerk + zeros(1,length(t1));

% Zeit volle Geschwindigkeit
x4_0 = x_jerk_2(length(x_jerk_2));
v4_0 = v_jerk_2(length(v_jerk_2));
a4_0 = vpa(a_jerk_2(length(a_jerk_2)));
T_v = (X_end - 2 * x4_0)/V_max;
t(4) = T_v;
t3 = t(1):Te:t(4);
x_fullspeed = x(x4_0, v4_0, 0, 0, t3);
v_fullspeed = v(v4_0, 0, 0, t3);
a_fullspeed = a(0, 0, t3);
j_fullspeed = 0 + zeros(1,length(t3));

% Zeit Jerk3
x5_0 = x_fullspeed(length(x_fullspeed));
v5_0 = v_fullspeed(length(v_fullspeed));
a5_0 = 0;

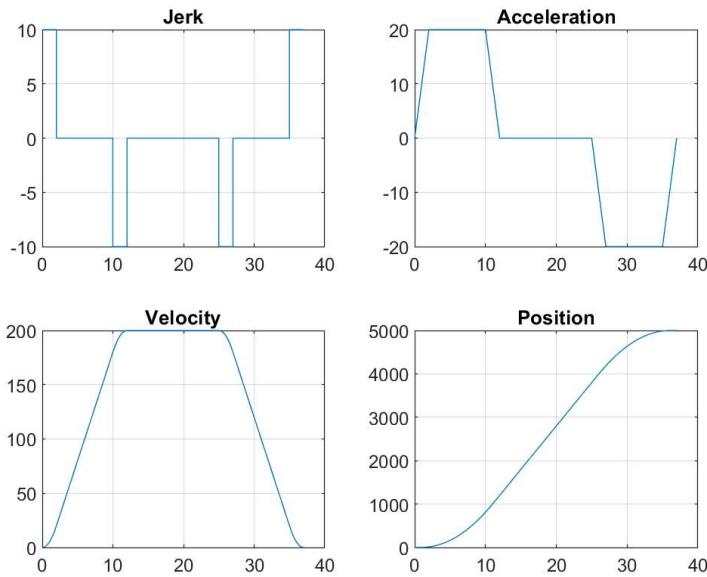
T_j_3 = T_j_2;
x_jerk_3 = x(x5_0, v5_0, a5_0, -Jerk, t1);
v_jerk_3 = v(v5_0, a5_0, -Jerk, t1);
a_jerk_3 = a(a5_0, -Jerk, t1);
j_jerk_3 = -Jerk + zeros(1,length(t1));

% Zeit Entschleunigen
x6_0 = x_jerk_3(length(x_jerk_3));
v6_0 = v_jerk_3(length(v_jerk_3));
a6_0 = -A_max;
T_a_2 = T_a_1;
x_acceleration_2 = x(x6_0, v6_0, a6_0, 0, t2);
v_acceleration_2 = v(v6_0, a6_0, 0, t2);
a_acceleration_2 = a(a6_0, 0, t2);
j_acceleration_2 = -0 + zeros(1,length(t2));

% Zeit Jerk4
x7_0 = x_acceleration_2(length(x_acceleration_2));
v7_0 = v_acceleration_2(length(v_acceleration_2));
a7_0 = -A_max;
T_j_4 = T_j_3;
x_jerk_4 = x(x7_0, v7_0, a7_0, Jerk, t1);
v_jerk_4 = v(v7_0, a7_0, Jerk, t1);
a_jerk_4 = a(a7_0, Jerk, t1);
j_jerk_4 = Jerk + zeros(1,length(t1));

T_out = [t1, t(2)+t2, t(2)+t(3)+t1, 2*t(2)+t(3)+t3, 2*t(2)+t(3)+t4+t1, 3*t(2)+t(3)+t4+t2, 3*t(2)+2*t(3)+t4+t1];
J_out = [j_jerk_1,j_acceleration_1,j_jerk_2,j_fullspeed,j_jerk_3,j_acceleration_2,j_jerk_4];
A_out = [a_jerk_1,a_acceleration_1,a_jerk_2,a_fullspeed,a_jerk_3,a_acceleration_2,a_jerk_4];
V_out = [v_jerk_1,v_acceleration_1,v_jerk_2,v_fullspeed,v_jerk_3,v_acceleration_2,v_jerk_4];
X_out = [x_jerk_1,x_acceleration_1,x_jerk_2,x_fullspeed,x_jerk_3,x_acceleration_2,x_jerk_4];
figure();
subplot(2,2,1)
plot(T_out,J_out);
title('Jerk');
grid;
subplot(2,2,2)
plot(T_out,A_out);
title('Acceleration');
grid;
subplot(2,2,3)
plot(T_out,V_out);
title('Velocity');
grid;
subplot(2,2,4)
plot(T_out,X_out);
title('Position');
grid;

```



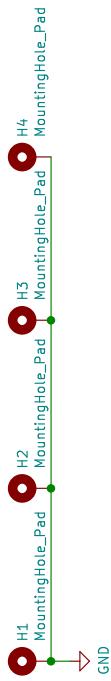
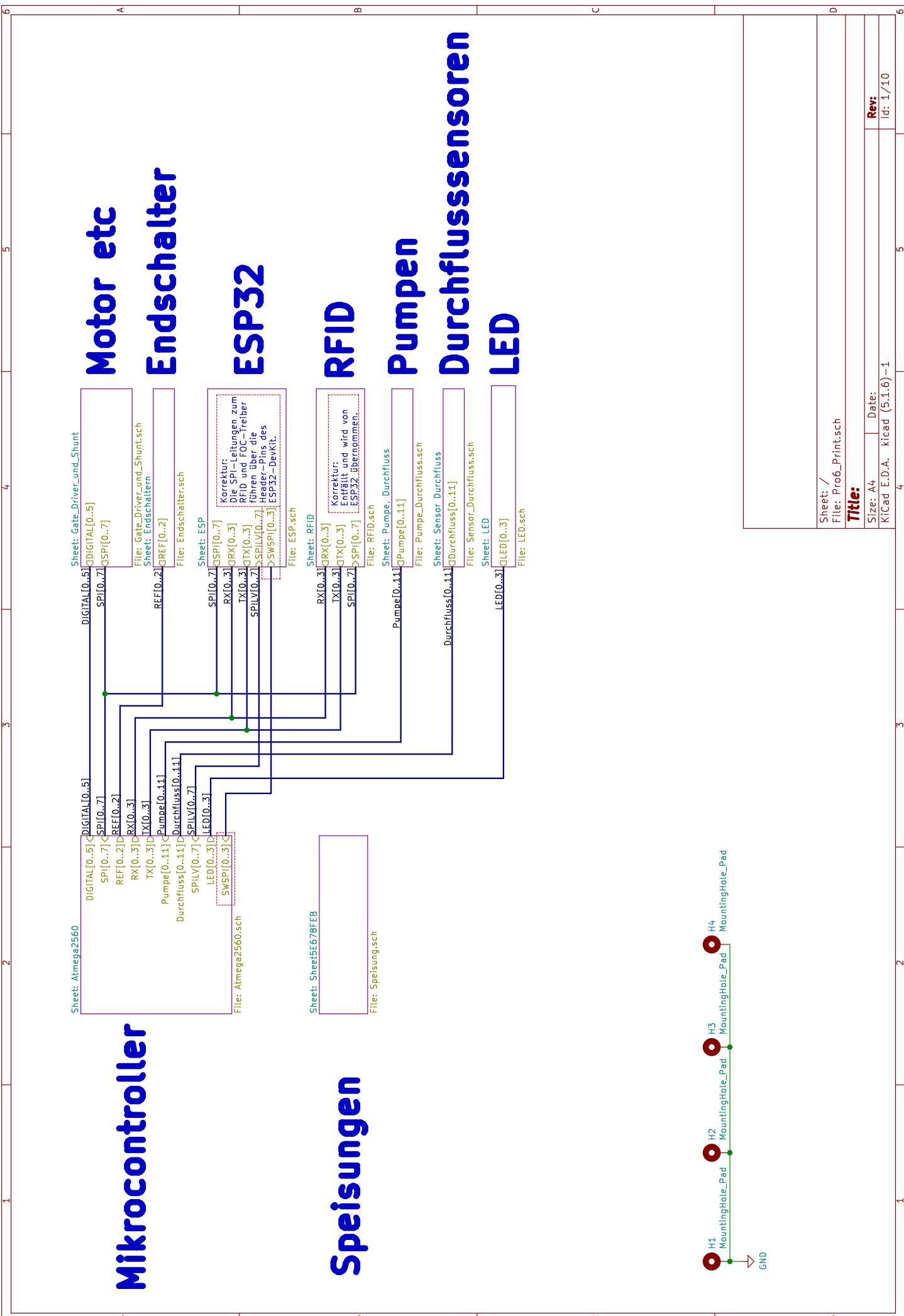
```
% Totale Zeit
T = 4 * T_j_1 + 2 * T_a_1 + T_v
```

T = 37.0

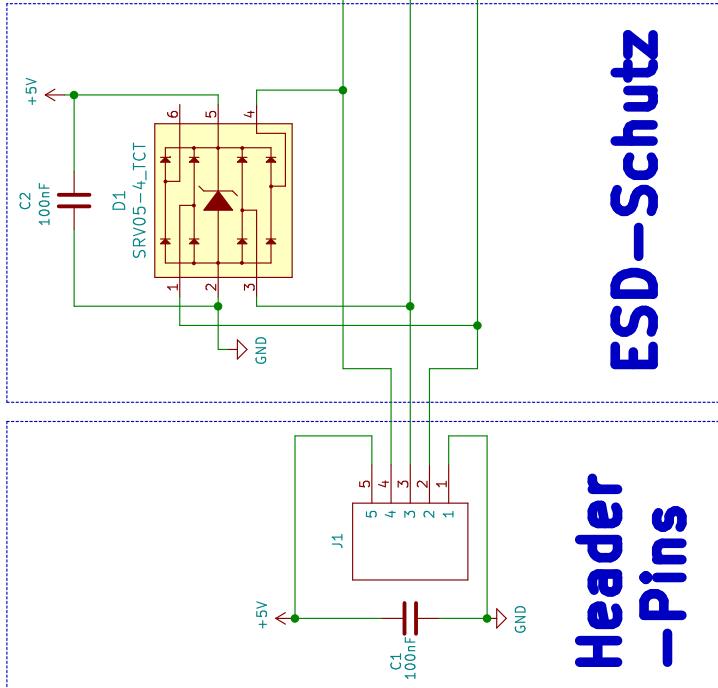
```
if X_end<=(T_a_1 + 2 * T_j_1)*V_max % Triangular velocity profile
    T_v=0;
    T_j_1 =  0;
    T_a_1=sqrt(X_end/A_max);
end
```

## I Print

### I.1 KiCad Schema



# Endschalter



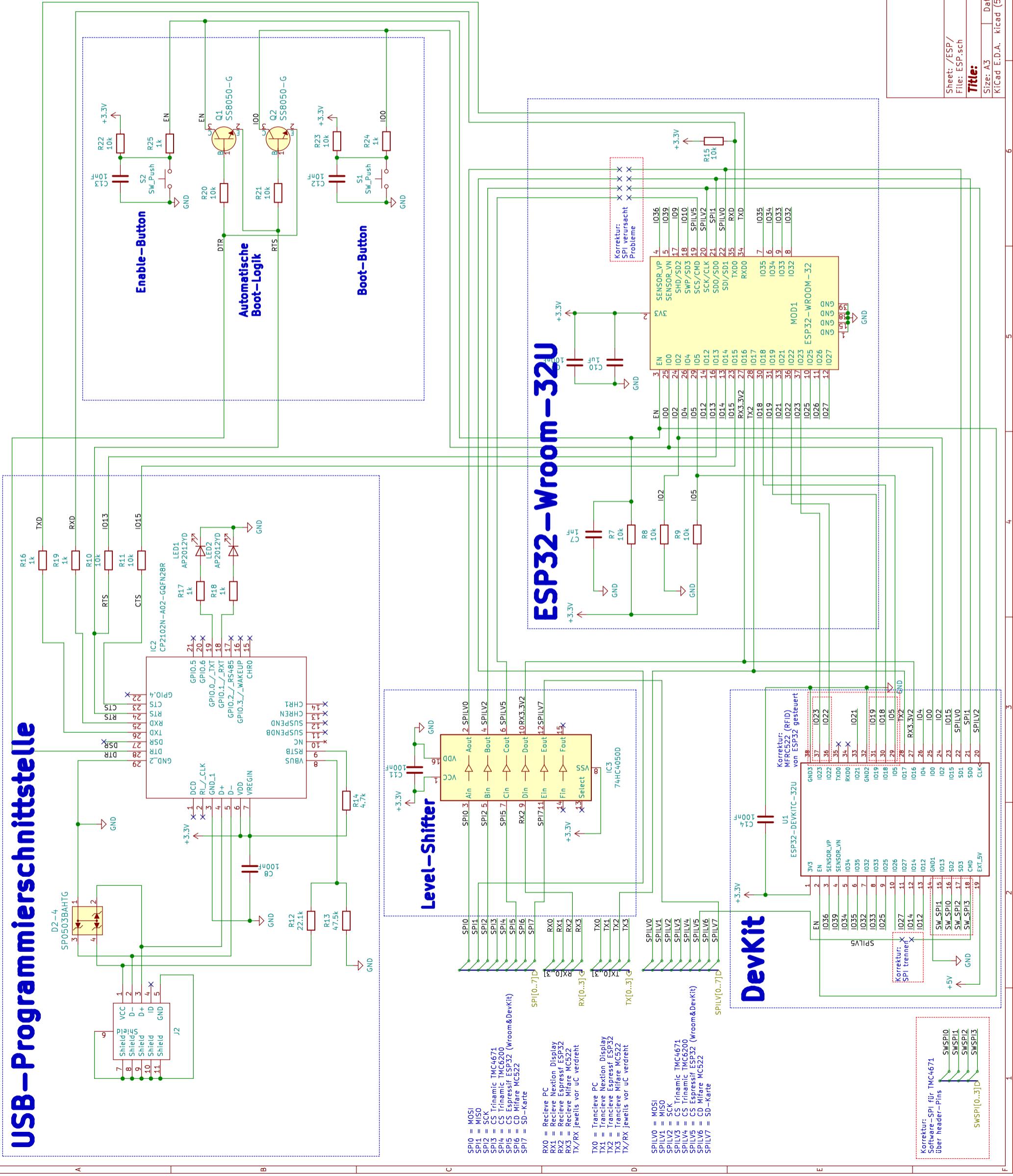
Sheet: /Endschaltern/  
File: Endschalter.sch

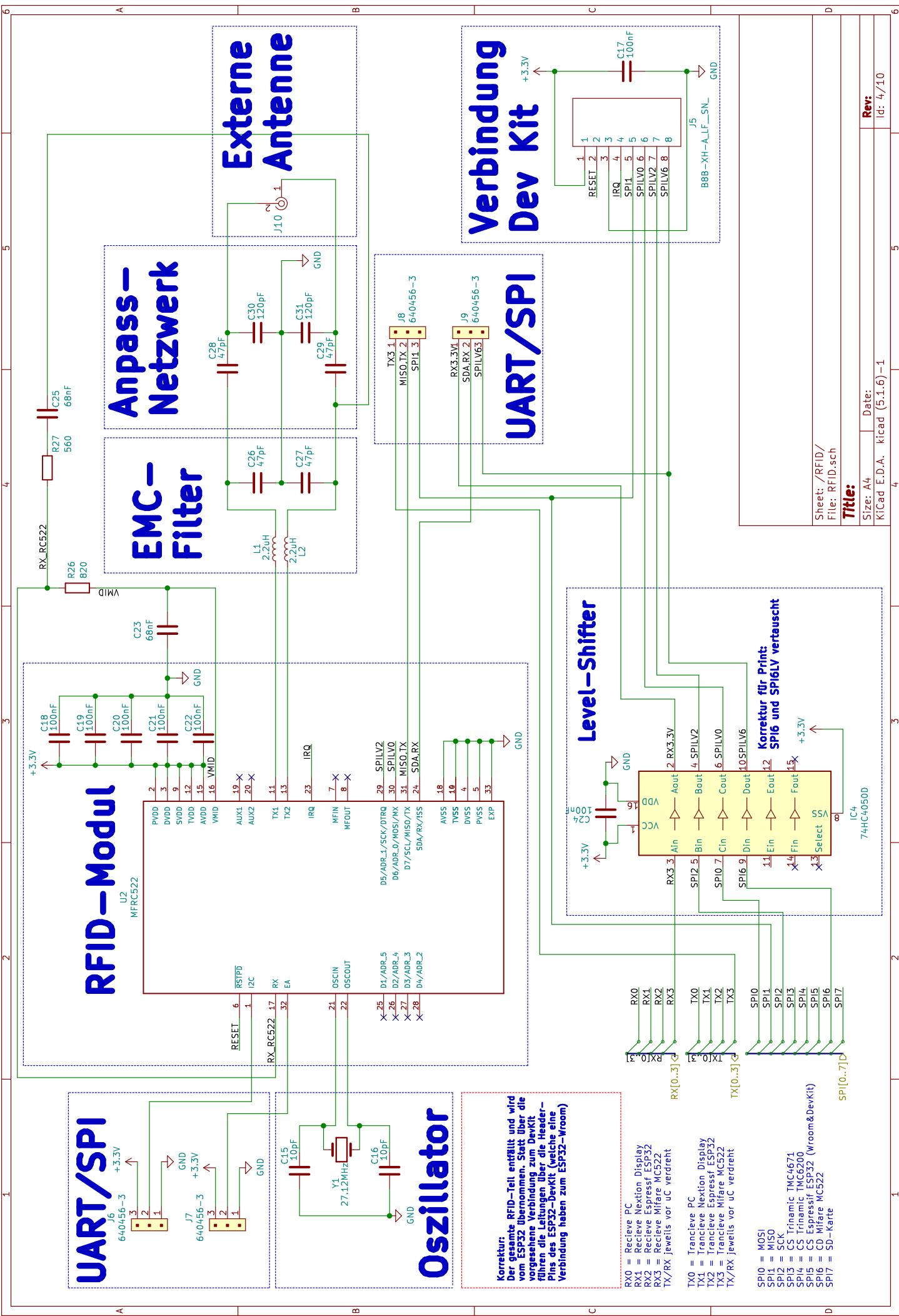
**Title:**  
Size: A4  
KICad E.D.A. kicad (5.1.6)-1

**Rev:**  
Id: 2/10

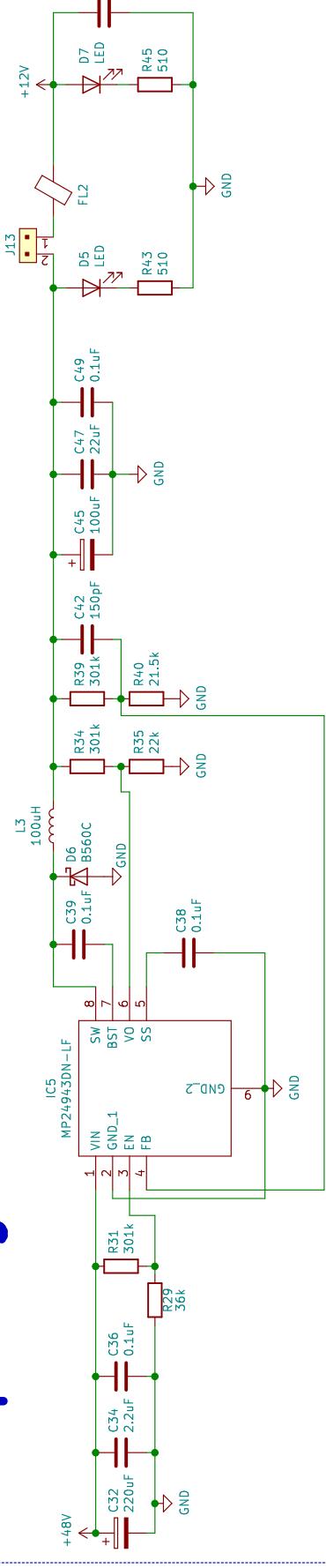
Date:  
Title:  
Size: A4  
KICad E.D.A. kicad (5.1.6)-1

# USB-Programmierschnittstelle

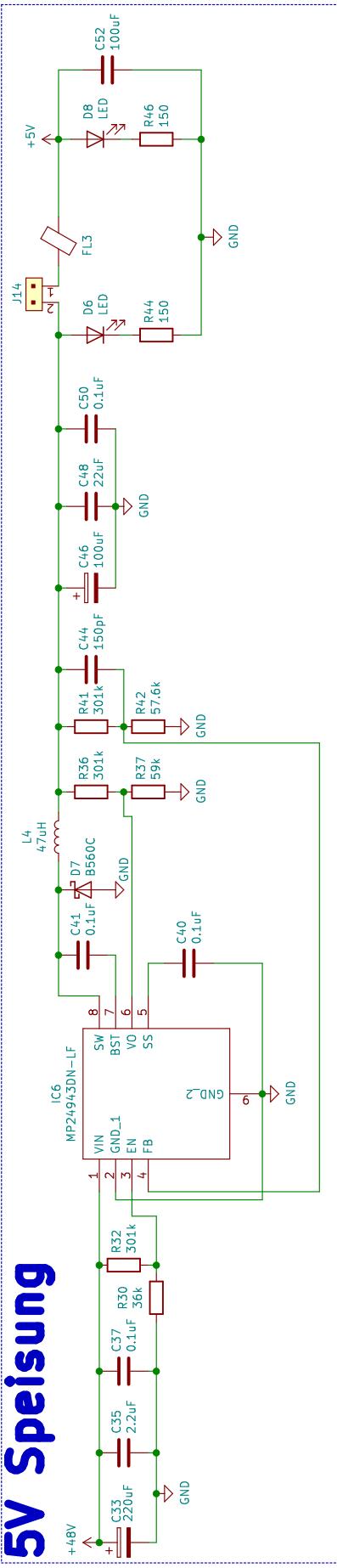




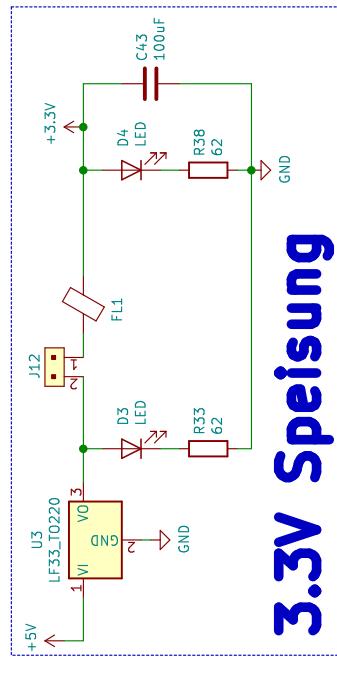
# 12V Speisung



# 5V Speisung



# 3.3V Speisung

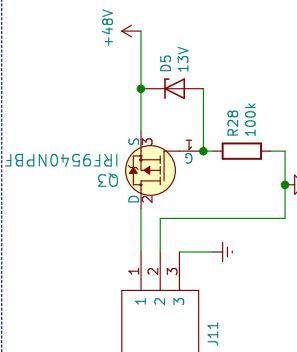


Sheet: /Sheet5E678FEB/  
File: Speisung.sch

Title:

Size: A4  
KiCad E.D.A. kicad (5.1.6)-1  
Rev: 5/10  
Id: 5/10

# Verpolungsschutz



# Pumpenansteuerungen

Sheet: /Pumpe\_Durchfluss/  
File: Pumpe\_Durchfluss.sch

**Title:**

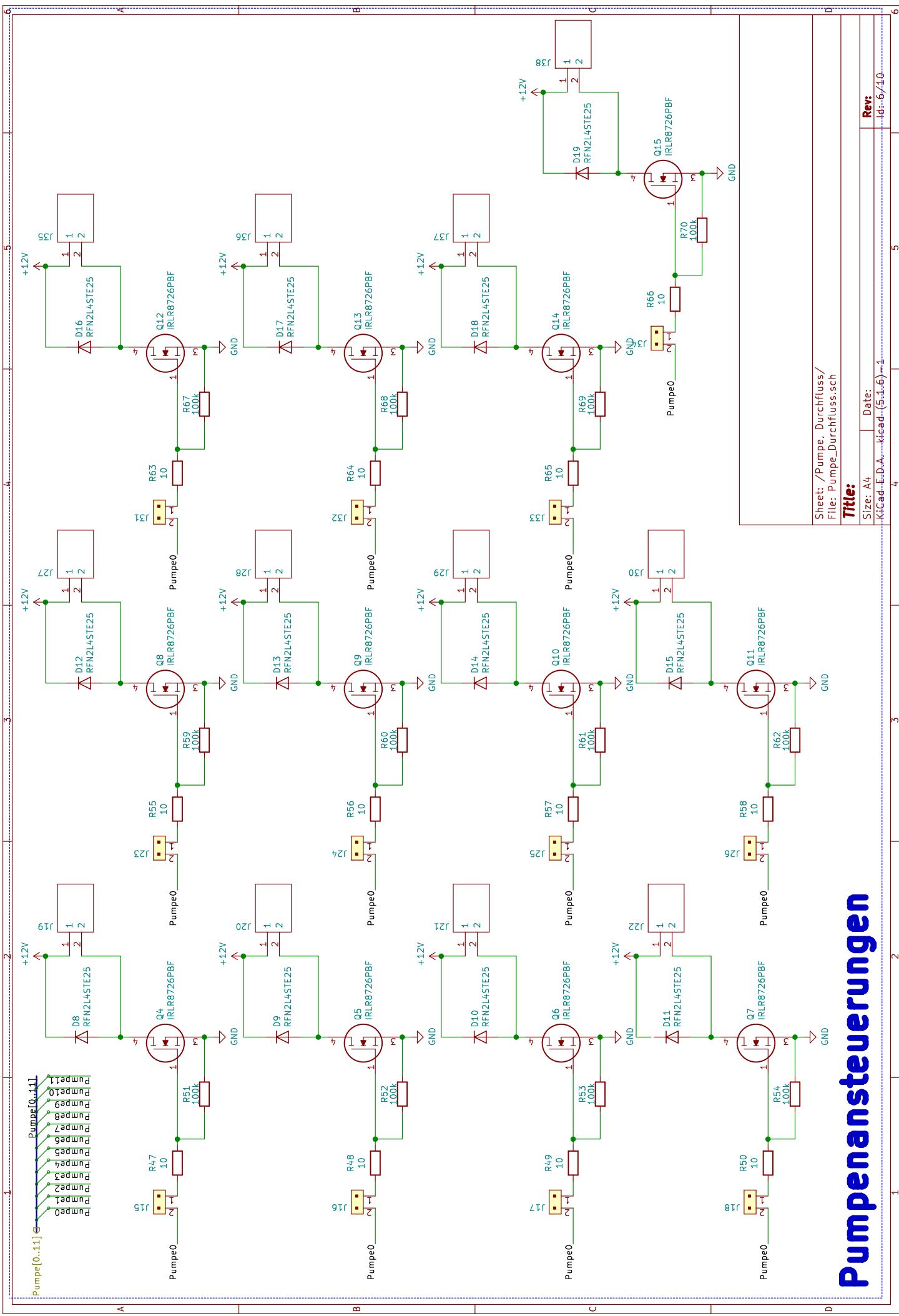
**Size:** A4

**Date:**

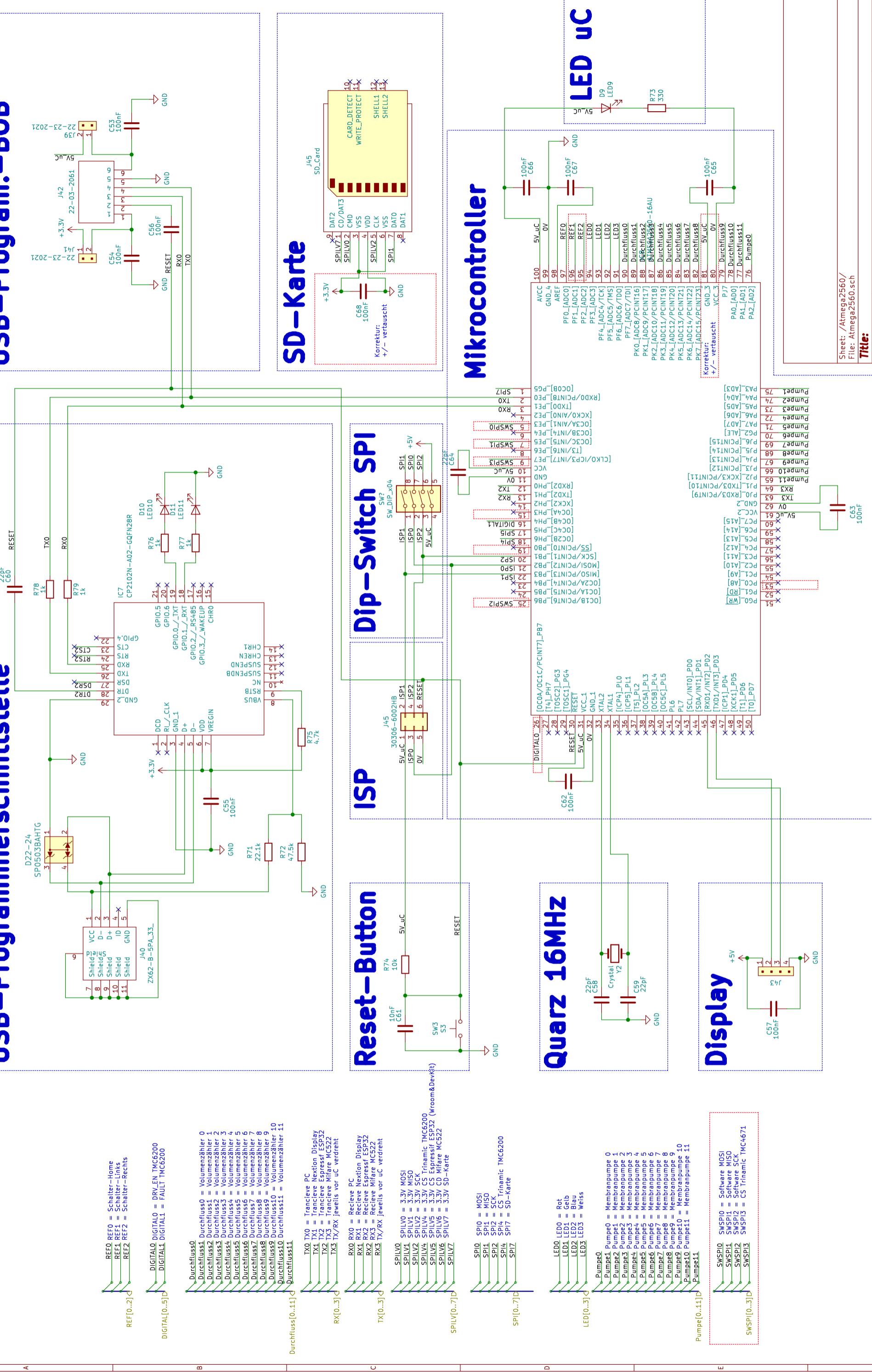
KiCad-E.DA...-kicad-(5.1.4)-1

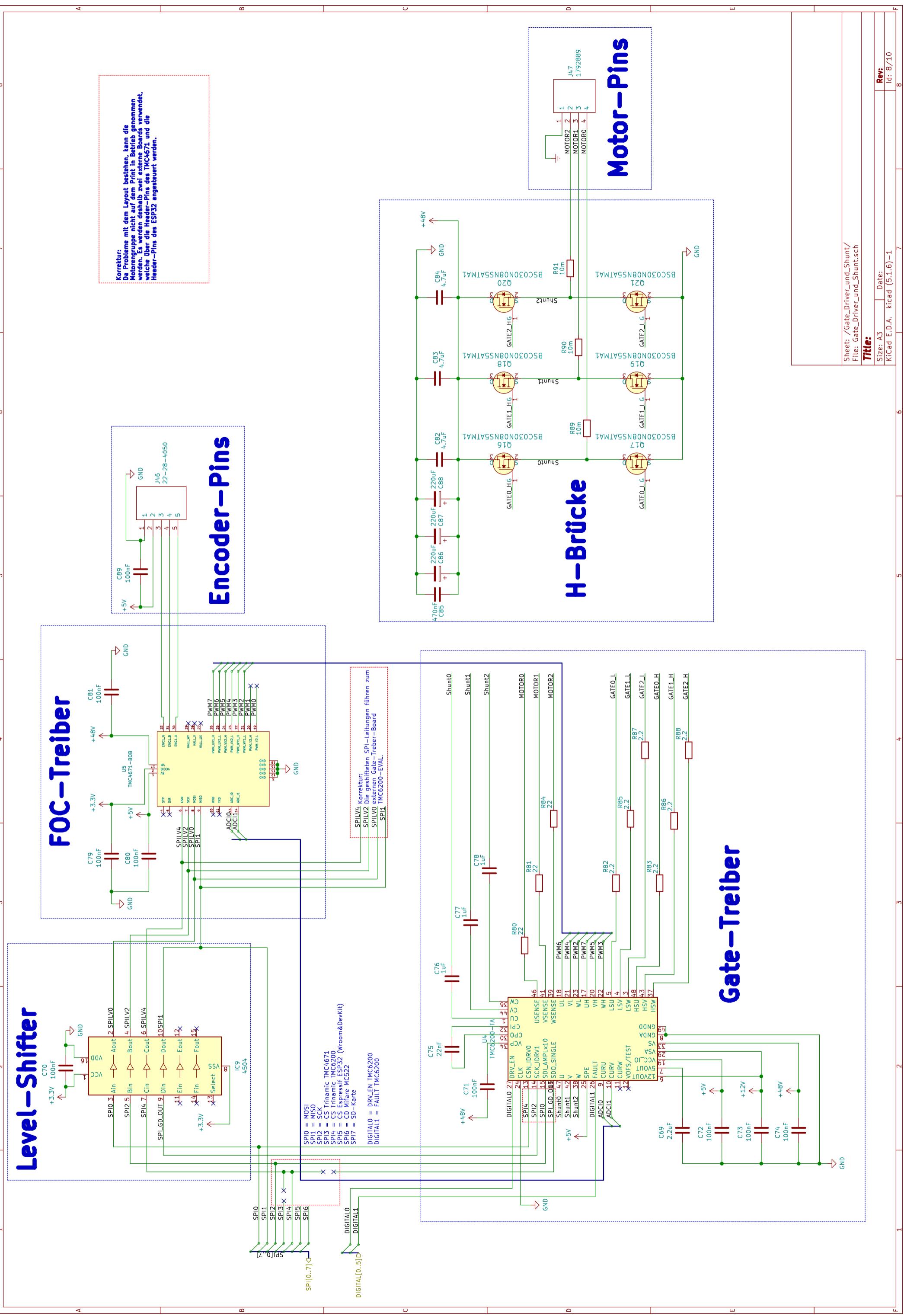
**Rev:**

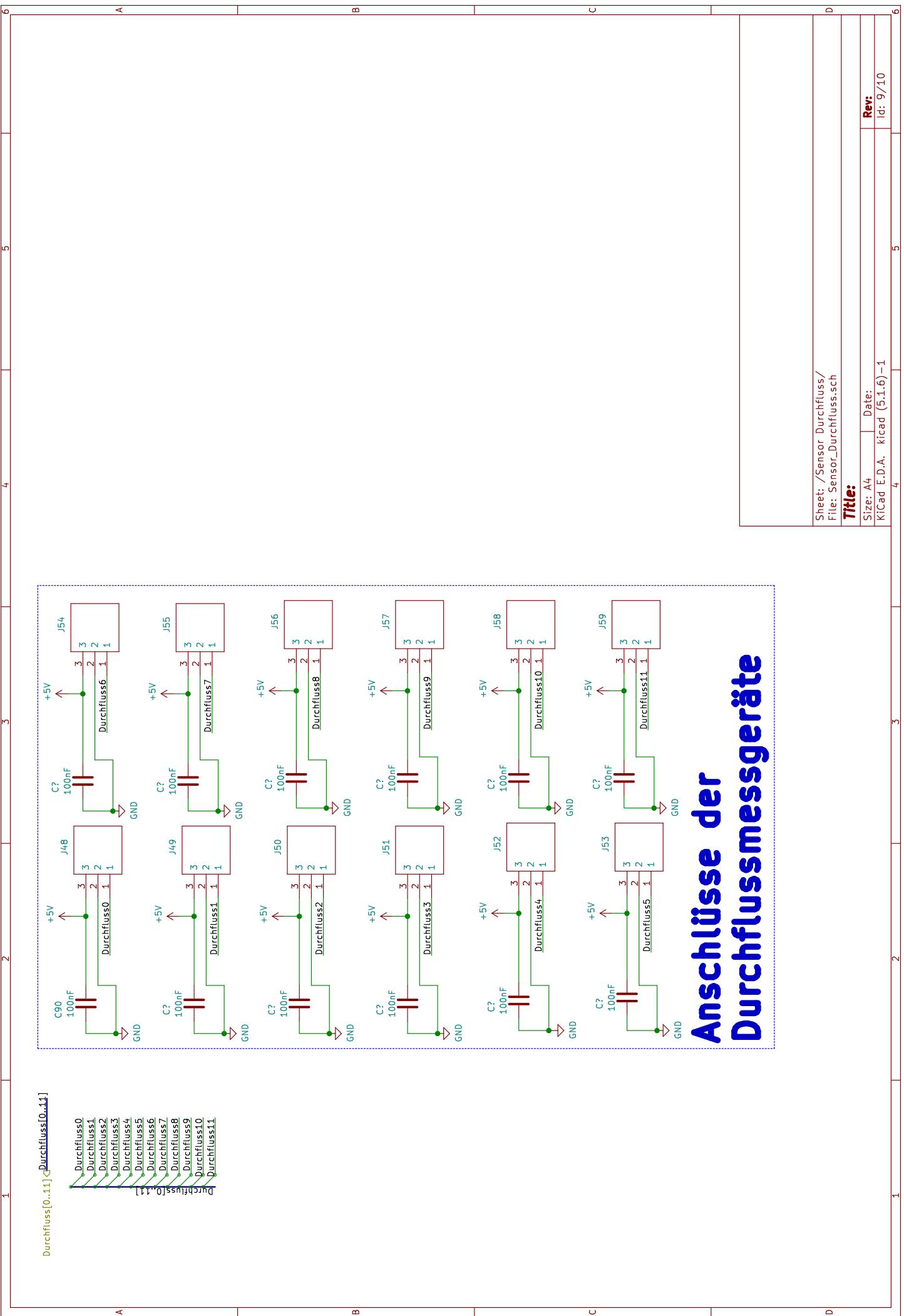
Id: 6/10



# USB-Programmierschnittstelle







# Anschlüsse der Durchflussmessgeräte

Sheet: /LED/  
File: LED.sch

**Title:**  
Size: A4  
Date:  
KiCad E.D.A. kicad (5.1.6)-1  
Rev: Id: 10/10

