

# Fachbericht

PROJEKT 6 COCKTAILMASCHINE - TEAM SCHENK & AEBI  
13. September 2020

<b>Betreuer Dozent:</b>	Prof. Dr. Schleuniger, Pascal
<b>Team:</b>	Schenk, Kim Aebi, Robin
<b>Studiengang:</b>	Elektro- und Informationstechnik
<b>Semester:</b>	Frühlingssemester 2020

## **Abstract**

In diesem Projekt wurde ein Konzept erstellt, um eine Cocktailmachine zu bauen. Dies reicht von der Analyse, was für Cocktailmaschinen es bereits gibt, über die Erstellung eines Grob- und eines Detailkonzeptes bis hin zur Evaluation der Komponenten. Der Aufbau wurde so gewählt, dass ein Glas mittels eines Linearantriebes auf einem Schlitten hin- und her gefahren wird und unter dem gewünschten Flüssigkeitsauslass stehen bleibt, wo es dann befüllt wird. Die Bedienung soll über ein Touch-Display geschehen. Die Verarbeitung der Daten wird ein Mikrocontroller übernehmen. Als mechanische Komponente wird pro Zutat eine Pumpe und ein Durchflusssensor verwendet sowie ein einzelner Motor, welcher den Linearantrieb mit dem Schlitten betreibt. Als Motor wurde ein bürstenloser Gleichstrommotor verwendet, da dieser ein sehr gutes Leistungs-/Gewicht-Verhältnis aufweist und in seiner Ansteuerung sehr interessant ist. Ziel des Projekt 5 war es, anhand des Konzeptes die einzelnen Teilsysteme aufzubauen und deren Funktion zu verifizieren und zu dokumentieren. Softwaremässig wurde die Basis für den Mikrocontroller geschrieben. Dies bedeutet, dass die Teilsysteme kontrollierbar sind und im Projekt 6 ausgebaut und zusammengeführt verwendet werden können. Die Software wurde komplett in C geschrieben und ausgiebig dokumentiert. Das Resultat zeigt, dass die Komponenten zusammenpassen und der Cocktailmachine im Projekt 6 nichts im Weg steht.

## Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
<b>2 Ausgangslage</b>	<b>2</b>
2.1 Blockschaltbild . . . . .	3
<b>3 Printaufbau</b>	<b>4</b>
<b>4 Teilsysteme</b>	<b>6</b>
4.1 Speisungen . . . . .	6
4.2 Motor . . . . .	11
4.3 Flüssigkeitsbeförderung . . . . .	18
4.4 Programmierschnittstellen . . . . .	20
4.5 Benutzerschnittstellen . . . . .	22
4.6 Beleuchtung . . . . .	27
4.7 Mikrocontroller . . . . .	28
4.8 SD-Karte . . . . .	29
<b>5 Inbetriebnahme</b>	<b>31</b>
5.1 Speisungen . . . . .	31
5.2 Programmierschnittstellen . . . . .	32
5.3 Benutzerschnittstellen . . . . .	34
5.4 Mikrocontroller . . . . .	37
5.5 Datenspeicherung . . . . .	39
5.6 Flüssigkeitsbeförderung . . . . .	39
5.7 Beleuchtung . . . . .	40
5.8 Motor . . . . .	44
<b>6 Software</b>	<b>48</b>
6.1 Atmega2560 . . . . .	48
6.2 ESP32 . . . . .	52
6.3 Displaysoftware . . . . .	53
6.4 Android App . . . . .	63

<b>7 Mechanik</b>	<b>67</b>
7.1 Rahmen . . . . .	68
7.2 Getränkeschlitten . . . . .	68
7.3 Verkleidung . . . . .	69
7.4 Unterbringung der Elektronik . . . . .	70
7.5 Kühlbox . . . . .	70
<b>8 Evaluation</b>	<b>72</b>
<b>9 Fazit</b>	<b>72</b>
9.1 Zielerreichung . . . . .	72
9.2 Kosten . . . . .	72
<b>10 Schlusswort</b>	<b>72</b>
<b>11 Ehrlichkeitserklärung</b>	<b>72</b>
<b>Literatur</b>	<b>74</b>
<b>A Programmierschnittstellen</b>	<b>76</b>
A.1 Geräte-Manager . . . . .	76
A.2 Atmega2560 . . . . .	76
A.3 ESP32 . . . . .	78
<b>B SD-Karte</b>	<b>82</b>
B.1 FAT32 . . . . .	82
<b>C ESP32</b>	<b>84</b>
C.1 Tabellarischer Vergleich zwischen ESP32 und ESP8266 . . . . .	84
C.2 Strapping-Pins . . . . .	84
C.3 Inbetriebnahme . . . . .	85
<b>D Mikrocontroller</b>	<b>88</b>
D.1 Inbetriebnahme . . . . .	88
D.2 Speicherorganisation . . . . .	92
<b>E TMC4671</b>	<b>94</b>
E.1 Standard-Schaltkreis . . . . .	94
E.2 BOB . . . . .	94
E.3 Inbetriebnahme . . . . .	96

<b>F TMC6200</b>	<b>103</b>
F.1 Standard-Schaltkreis . . . . .	103
F.2 Blockdiagramm . . . . .	103
F.3 Externe Gate-Spannungsversorgung . . . . .	104
F.4 Inbetriebnahme . . . . .	104
<b>G H-Brücke</b>	<b>111</b>
G.1 Referenzschema . . . . .	111
G.2 Inbetriebnahme . . . . .	111
<b>H CUI AMT332S-V ABN-Encoder</b>	<b>113</b>
H.1 Pinout and Interface . . . . .	113
H.2 Inbetriebnahme . . . . .	114
<b>I PI-Regler</b>	<b>117</b>
I.1 TMCL-IDE . . . . .	117

## 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 darf eines auf gar keinen Fall fehlen, die Getränke. 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. Anderseits 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 haben wir uns dazu entschieden eine automatisierte Cocktailmaschine zu entwerfen. Diese soll vollkommen autonom arbeiten und sollte problemlos von jeder beliebigen Person und in fast jedem Zustand bedient werden können.

In den folgenden Kapiteln ist dokumentiert, wie die Cocktailmaschine im Detail aussieht. Dazu gehören die elektronischen Teilsysteme, das dazugehörige Printdesign, die Software, die Mechanik und die Evaluierung.

## 2 Ausgangslage

Die Basis für das Projekt 6 hat das Projekt 5 gebildet, in welchem ein Konzept erstellt wurde, welches die Hauptkomponenten der Maschine festlegte und dessen Arbeitsweise. Aus dieser Entscheidungsfindung wurde dann ein Blockschaltbild erstellt, welches in Abbildung 2.1 zu sehen ist. Ein weiterer Teil des Projekt 5 war es, die gewählten Komponenten und die daraus entstandenen Teilsysteme in einem Testaufbau aufzubauen und zu evaluieren. Dazu gehörten die Speisungen (48V, 12V, 5V und 3.3V), der Mikrocontroller, das Display, der Motor, die Pumpenansteuerung und die Durchflussmessgeräte.

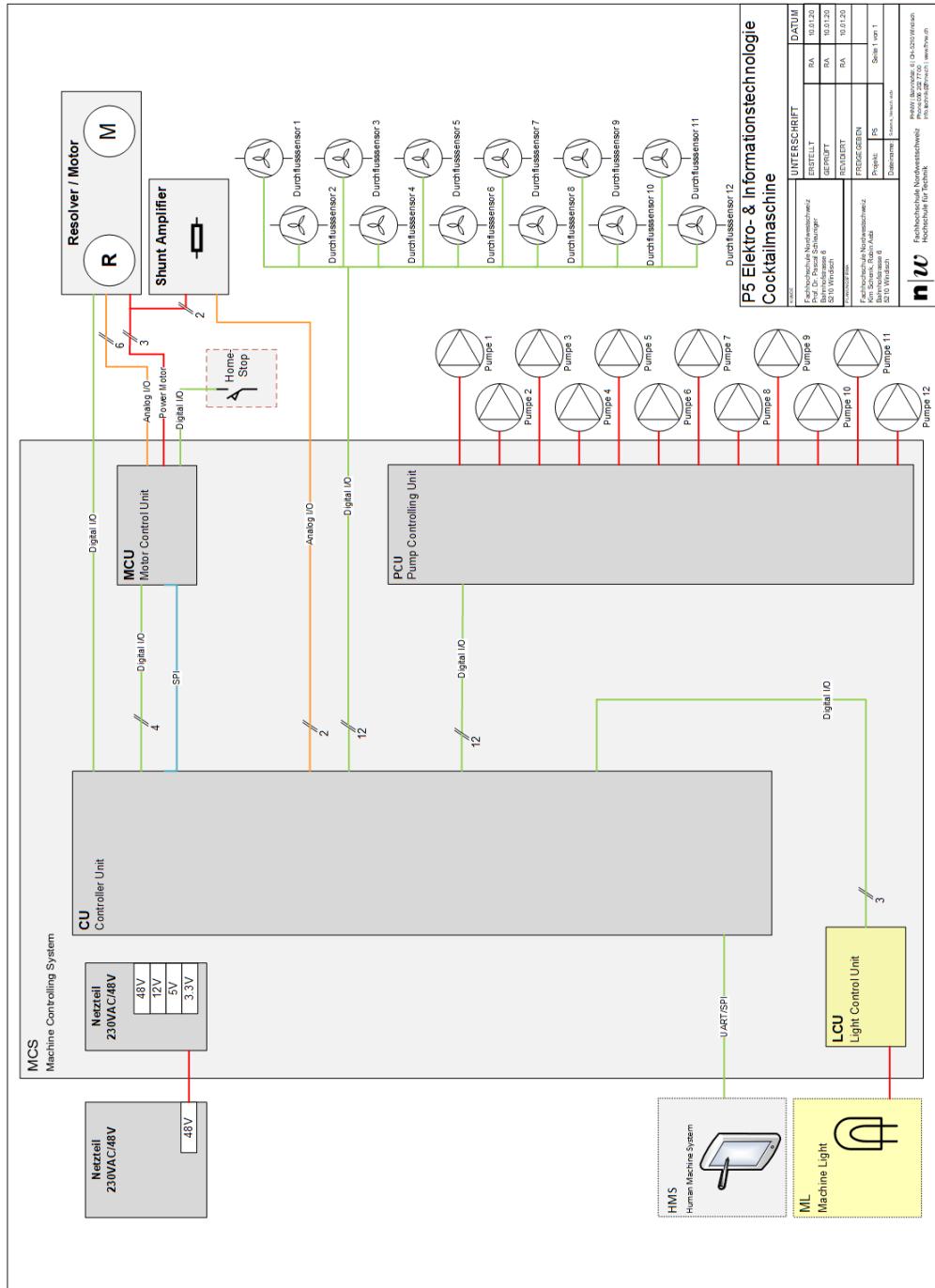
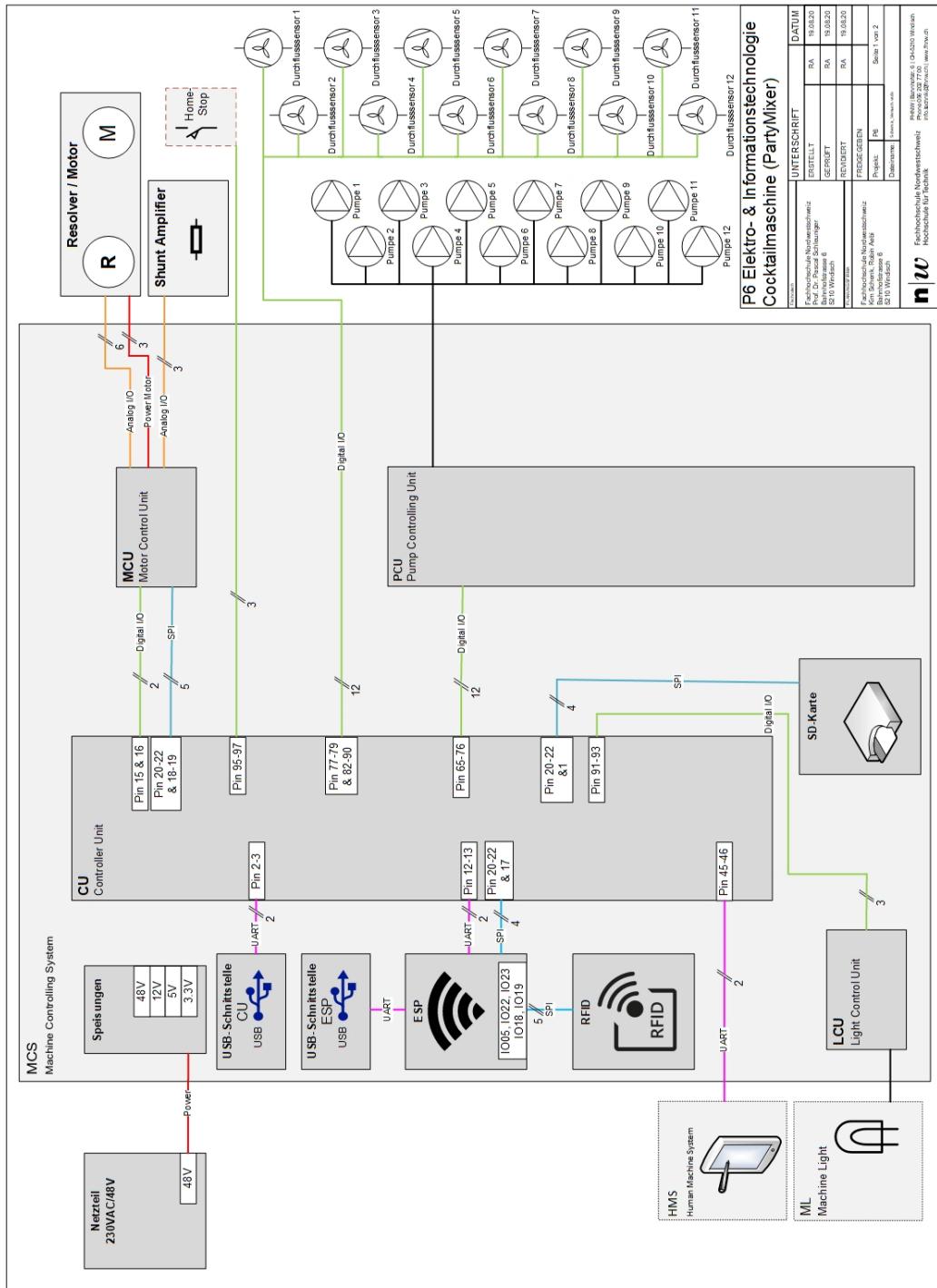


Abbildung 2.1: Blockschaltbild des CocktailMixer's gemäß Projekt 5.

## 2.1 Blockschaltbild

Das im Projekt 5 entstandene Blockschaltbild ist nun um weitere Teilsysteme ergänzt worden, auf welche im folgenden Kapitel ?? eingegangen wird. Zu den neuen Systemen gehört ein Bluetooth- / Wirelessmodul, welches eine externe Ansteuerung per Web-Server oder Android App ermöglicht, eine dazugehörige Programmierschnittstelle, ein RFID Lesegerät, ein SD-Karten Slot und eine Maschinenbeleuchtung.



**Abbildung 2.2:** Blockschaltbild des CocktailMixer's gemäss Projekt 6.

### 3 Printaufbau

In Abbildung 3.1 ist der Aufbau des entstandenen Prints zu sehen. Auf diesem sind die in Abbildung 2.2 gezeigte Blöcke wieder zu erkennen.

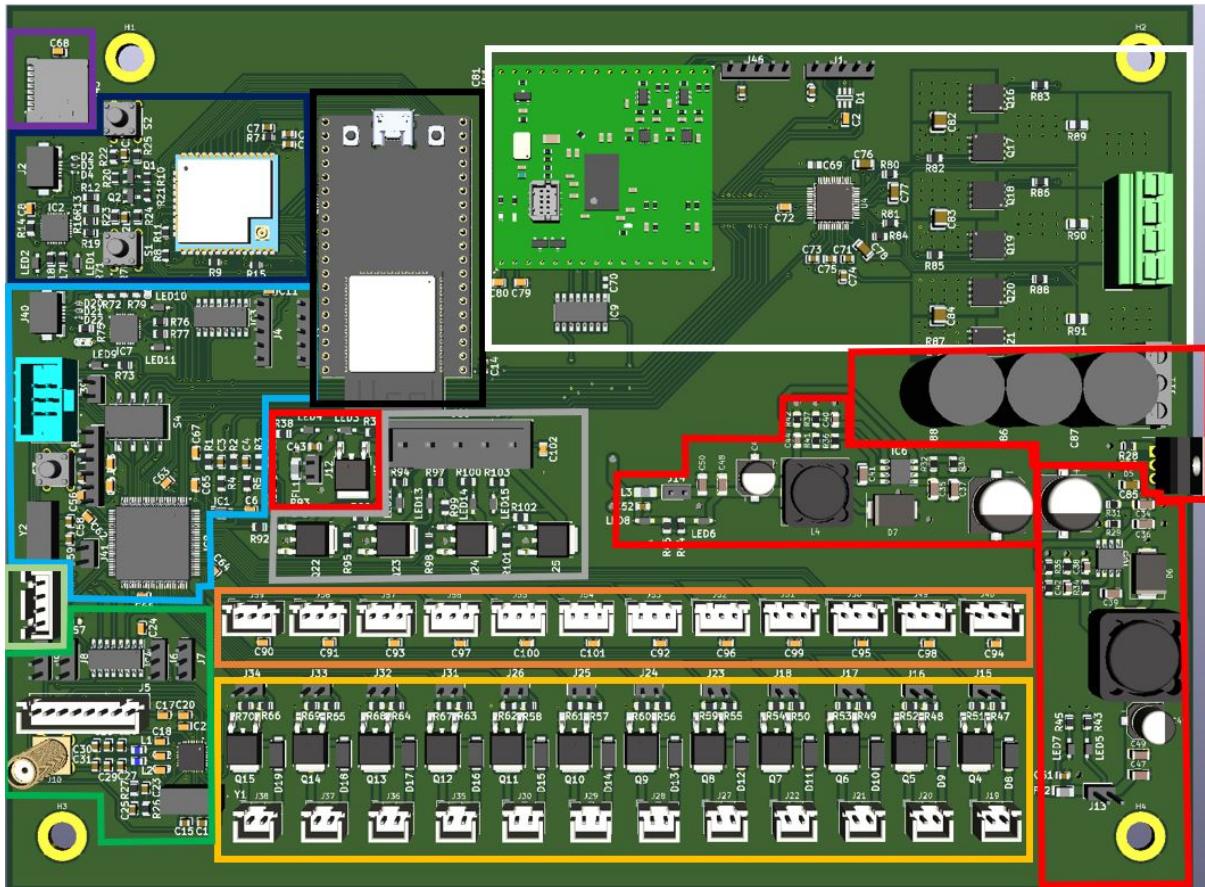


Abbildung 3.1: Print 3D

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 3.1: JLCPCB [1]

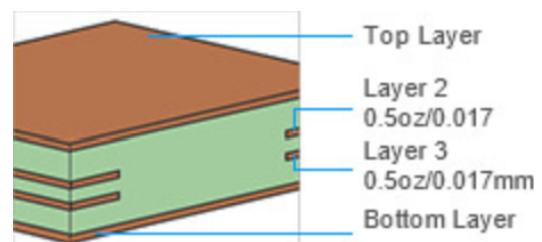


Abbildung 3.2: JLCPCB [1]

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 Logiteil zu finden. In Abbildung 3.1 sind die einzelnen Teile in Farben unterteilt. Die genauen Aufgabenbereiche und die Funktionsweise der einzelnen Teile werden in Kapitel 4 behandelt.

**Speisungen:** In den rot markierten Bereichen sind die Speisungsteile zu sehen, wobei sich auf der rechten Seite des Prints der Eingangsfilter 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 Printes befindet sich das Herzstück der Elektronik. Der Mikrocontroller mit der dazugehörigen Programmierschnittstelle, welche einen Micro-USB Anschluss beinhaltet.

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

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

**Maschinenbeleuchtung:** Um die Maschine mit einem Showeffekt auszustatten wurde eine Lichtansteuerung eines RGBW-LED-Streifen implementiert gemäss Kapitel 2.1. Diese ist im grau markierten Bereich zu finden und beinhaltet einen Anschluss, sowie vier Steuer-MOSFET's.

**Wireless- /Bluetoothmodul:** Um mit externen Gerätschaften kommunizieren zu können, befindet sich im Dunkelblau markierten Bereich des Wireless- / Bluetoothmodul. Auch dieses Beinhaltet neben dem eigentlichen Modul eine dazugehörige Programmierschnittstelle gemäss Blockschaltbild 2.2.

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

**Motoransteuerung:** Einen grossen Teil des Print's hat die Motorenansteuerung in Beschlagsnahmung genommen. Diese ist im weissen Teil des Print's zu finden. Diese ist in drei Teile unterteilt von links nach rechts. Als erstes der FOC-Treiber, gefolgt vom Gate-Treiber und der H-Brücke mit dem Anschluss des Motor's.

**Display:** Das Display benötigt keine grosse Ansteuerungslogik und besteht aus diesem Grund auch nur aus einem einzelnen Stecker, 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 Anschläussen befindet sich nun das RFID Evaluationsboard.

**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 befindet.

## 4 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 und der Funktionsbeschrieb der Schaltung. Das komplette Schema kann in **Anhang Schema** begutachtet werden.

### 4.1 Speisungen

Ohne Speisung funktioniert keine elektronische Schaltung. Sie bildet daher einen essentiellen Bestandteil des Partymixer's. In dem System befinden sich vier verschiedene Speisungen. Die Ausgangsspannung für alle anderen Speisespannungen bildet dabei ein 48V Netzteil. Aus dieser Spannung wird mittels Step-Down Reglern eine 12V und eine 5V Speisung erzeugt. Bei der vierten Spannung handelt es sich um einen einfachen Linearregler, welcher aus den 5V eine 3.3V Speisung realisiert.

#### 4.1.1 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 gemäss **Fachbericht 5** eingesetzt.

Es wurde im Projekt 5 entschieden, dass die 48V Speisung extern als fertiges Netzteil eingekauft wird. Somit entfällt das Schema für diesen Speisungsteil. Ein Anschauungsbild des eingesetzten Netzteils kann in Abbildung 4.1 begutachtet werden.



**Abbildung 4.1:** Anschauungsbild des 48V Netzteils

Es musste jedoch unbedingt eine Leistungsabschätzung gemacht werden. Auch diese wurde im Projekt 5 durchgeführt. Unter Berücksichtigung der Schaltungsteile welche noch im Projekt 6

ergänzt werden, wurde dieses dann ausgewählt und eingekauft. Die Leistungsabschätzung kann im **Fachbericht 5** eingesehen werden.

#### 4.1.2 12V Speisung

Die Pumpen werden mit 12V betrieben, was zur Folge hat, dass eine 12V Speisung implementiert werden musste. Dazu wird ein Schaltspannungsregler verwendet. Dieser wandelt mittels Step-Down Prinzip die 48V des Netzteils in eine Konstantspannungsquelle 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 kann in Abbildung 4.2 betrachtet werden.

#### Schema

Das Schema in Abbildung 4.2 kann in fünf Teile unterteilt werden. Da wäre zuerst der Eingangsfilter, welcher mit C32, C34 & C36 realisiert ist. Dieser Eingangsfilter wird gefolgt von einem Spannungsteiler, welcher den Enable 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 Filterstufe implementiert, welche das Ausgangssignal glättet.

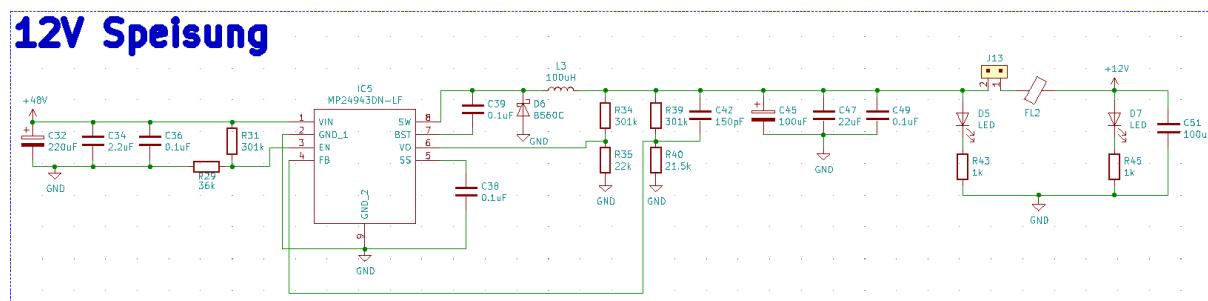


Abbildung 4.2: Schema der 12V Speisung

#### 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 auf inaktiv gesetzt. Damit der Spannungsregler immer eingeschaltet ist, 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 außer dem  $V_{in}$  einen maximalen Eingangspiegel von 6.5V verkraften können.

Die gewünschte Ausgangsspannung wird mittels Spannungsteiler R39 & R40 eingestellt, welche auf den Feedback Eingang (FB) rückgekoppelt werden. Diese berechnet sich laut Datenblatt gemäß Formel 4.1.

$$R40 = \frac{R39}{\frac{V_{out}}{0.8} - 1} \quad (4.1)$$

Bei einem Widerstandsverhältnis von  $R39=301\text{k}\Omega$  &  $R40=21.5\text{k}\Omega$  entspricht dies einer Ausgangsspannung von 12V.

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 4.2 berechnet.

$$R35 = \frac{R34}{\frac{V_{ovp}}{V_{ovref}} - 1} \quad (4.2)$$

Bei einem Widerstandsverhältnis von  $R34=301\text{k}\Omega$  &  $R35=22\text{k}\Omega$  entspricht dies einer Überspannungsschutzschwelle von 13.21V.

Der Rippel des Spulenstroms lässt sich gemäss Formel 4.3 berechnen. Dieser sollte gemäss Datenblatt ca. 30% des maximalen Ausgangsstroms von 3A betragen.

$$L3 = \frac{V_{out} \cdot (V_{in} - V_{out})}{V_{in} \cdot \Delta I_L \cdot f_{osc}} \quad (4.3)$$

Der interne Oszillator läuft dabei bei einer Frequenz von 100kHz. Bei der ausgewählten Spule von  $100\mu\text{H}$  erhalten wir ein  $\Delta I_L$  von 0.9A. 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.

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.

Um die Schaltung einfacher 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 soll noch letzte Störungen herausfiltern.

#### 4.1.3 5V Speisung

Der Mikrocontroller, sowie die Durchflussmessgeräte und das Display werden mit 5V betrieben. Aus diesem Grund wurde eine 5V Speisung implementiert. Dazu wird der selbe Schaltspannungsregler wie bei der 12V Speisung in Kapitel 4.1.2 verwendet. Die Realisierung der 5V Speisung kann in Abbildung 4.3 betrachtet werden.

## Schema

Das Schema in Abbildung 4.3 kann wie bei der 12V Speisung gemäss Kapitel 4.1.2 in fünf Teile unterteilt werden. Da wäre zuerst der Eingangsfilter, welcher mit C31, C33 & C35 realisiert ist. Dieser Eingangsfilter wird wiederum gefolgt von einem Spannungsteiler, welcher den Enable auf aktiv setzt. Der eigentliche Regler wird auch hier mittels des IC6, D5 & L4 realisiert. Mittels zweier Spannungsteiler, wird die gewünschte Ausgangsspannung, sowie die Übervoltageschutz eingestellt. Vor dem Ausgang der Schaltung ist dann erneut eine Filterstufe implementiert, welche das Ausgangssignal glättet.



Abbildung 4.3: Schema der 5V Speisung

## Funktionsbeschrieb der Schaltung

Auch bei der 5V Speisung wurde mittels R29 & R31 ein Spannungsteiler realisiert, welcher das IC gemäss Kapitel 4.1.2 auf aktiv setzt.

Das Widerstandsverhältnis von R41 & R42, welches die Ausgangsspannung definiert, wurde gemäss Formel 4.1 berechnet. Somit ergeben sich für  $R41=301\text{k}\Omega$  und für  $R42=57.6\text{k}\Omega$ , Was einer Ausgangsspannung von 4.98V entspricht.

Beim Überspannungsschutz musste darauf geachtet werden, dass der Mikrokontroller AtMega2560-16AU 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 erreichen zu können, wurde für  $R36=301\text{k}\Omega$  und  $R37=53\text{k}\Omega$  gewählt. Gemäss Formel 4.2 erhält man so eine Überspannungsschutzwelle von 6V.

Der interne Oszillator läuft wiederum bei einer Frequenz von 100kHz. Bei der ausgewählten Spule L4 von  $47\mu\text{H}$  erhält man mittels Formel 4.3 ein  $\Delta I_L$  von 0.953A. Auch hier gilt gemäss Datenblatt, 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.

Mit den Kondensatoren C46, C48 & C50 wird die Ausgangsspannung zum Abschluss auch noch geglättet. Bei den Eingangskondensatoren, sowie den Ausgangskondensatoren sollte es sich um low ESR Typen handeln. Auch hier wurde noch zum Abschluss ein Ferrit implementiert, welcher allfällige hochfrequente Störungen herausfiltern soll.

Auch bei der 5V Speisung wurde ein Jumper zu Testzwecken und zwei LED's implementiert. Außerdem findet sich auch hier wieder ein Ferrit FL3, welcher letzte Störungen beseitigen soll.

#### 4.1.4 3.3V Speisung

Um die Treiber der Motorenansteuerung, das Wirelessmodul und die RFID-Schaltung betreiben zu können, wird zusätzlich eine 3,3V Speisung verbaut. Da es sich dabei nicht um enorm Leistungstreibende Elemente handelt, wurde entschieden einen einfachen Linearregler einzusetzen, welcher 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 Schama kann in Abbildung 4.4 begutachtet werden.

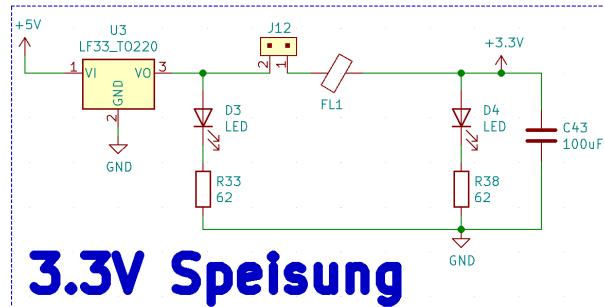


Abbildung 4.4: Schema der 3.3V Speisung

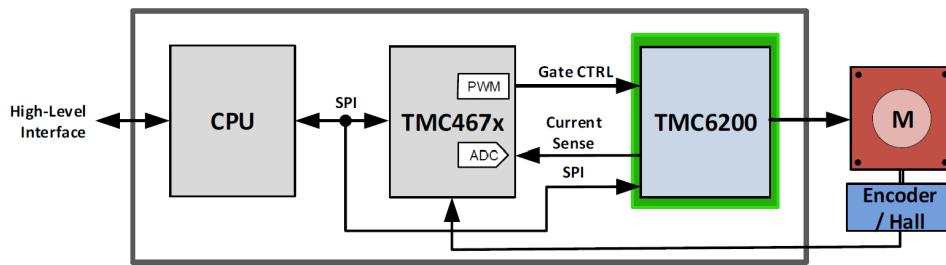
#### Funktionsbeschrieb der Schaltung

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

Wie bei der 12V Speisung in Kapitel 4.1.2 und der 5V Speisung in 4.1.3 wurde ein Jumper, sowie zwei LED's zu Testzwecken implementiert und auch hier Filtert ein Ferrit FL1 letzte Störungen heraus.

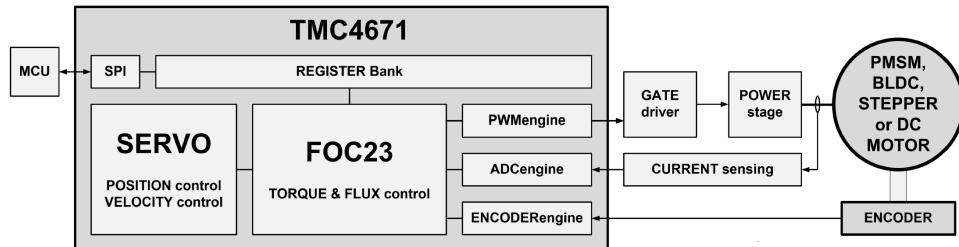
## 4.2 Motor

Um ein Glas während der Zubereitung hin- und her zu bewegen, wird eine Antriebsgruppe benötigt. Die Auswahl der Motorengruppe wurde mit Dozenten ausgewählt. Der Entscheid fiel dabei auf den Brushless DC-Motor AKM22h von SigmaTec. Auch dessen Ansteuerung ergab sich durch die schon vorhandenen EVAL-Boards mit TMC4671 und UPS 10A70V. Im Projekt 6 wird anstelle des UPS 10A70V ein TMC6200 verwendet. Das benötigte Feedback über die Lage des Rotors wird vom ABN-Encoder AMT33 von CUI devices geliefert. Zwei Shunts geben Auskunft über die Bestromung der Spulen. Auf die erwähnten Komponenten wird im Folgenden eingegangen. Abbildung 4.5 zeigt, wie die Komponenten zusammenhängen.



**Abbildung 4.5:** Blockschaltbild Konfiguration IC's mit BLDC und Encoder. [2, S.1]

Es wurde darauf geachtet, dass der Aufbau des Prints dem Testaufbau entspricht. In Abbildung 4.6 wird ein detaillierteres Blockschaltbild gezeigt, welches den Aufbau eher nach Funktionen beschreibt.



**Abbildung 4.6:** Blockschaltbild Motorengruppe nach Funktionen. [3, S.1]

### 4.2.1 FOC-Treiber TMC4671

Der FOC-Treiber berechnet den Modulationsindex für die H-Brücke anhand des Drehmoments, der Geschwindigkeit oder der Position, welche der Mikrocontroller dem FOC-Treiber vorgibt. Als Feedback benötigt der FOC-Treiber den momentanen Strom durch zwei der drei Spulen und eine Information über die momentane Lage des Rotors. Die vom FOC-Treiber ausgehenden PWM-Leitungen gehen direkt auf den Gate-Treiber, welcher dann wiederum die MOSFETs ansteuert.

Da das Gehäuse des IC's nur schwierig lötbar ist, wird ein Breakout-Board (BOB) verwendet. Auf das Board und die darauf vorhandenen Schaltungen wird im Funktionsbeschrieb eingegangen.

## Schema

Auf dem Breakout-Board sind diverse Anschlussmöglichkeiten vorhanden. Für den Cocktailmixer werden folgende Schnittstellen verwendet:

- SPI Input
- Phasenströme Input
- Encoder Input
- Motorspannung Input
- Steuersignale Output

In der schon gezeigten Abbildung 4.6 ist zu erkennen, welche Funktionen im Treiber vorhanden sind und wie diese zusammenhängen. Im wesentlichen kann daraus entnommen werden, dass der TMC4671 aus einer FOC-Logik<sup>1</sup>, einer Servo-Logik, einem SPI-Interface und diversen Engines (PWM, ADC, Encoder) besteht. Im Anhang Kapitel E.1 wird eine Standard-Anwendungsschaltung gezeigt, worin erkennbar ist, dass der Treiber sehr universell aufgebaut ist und für mehrere Motorentypen verwendet werden können. Die für den Partymixer essentiellen Ein- und Ausgänge wurden schon genannt.

In Abbildung 4.7 ist erkennbar, welche Pins für den Cocktailmixer verwendet werden. Es handelt sich dabei um die schon aufgelisteten Komponenten, nämlich Kommunikationsleitung zwischen Treiber und Mikrocontroller (SPI), Phasenströme (ADC), Encoder Input (ENC), Motorspannung (48V) und das Gate-Ctrl-Signal (PWM).

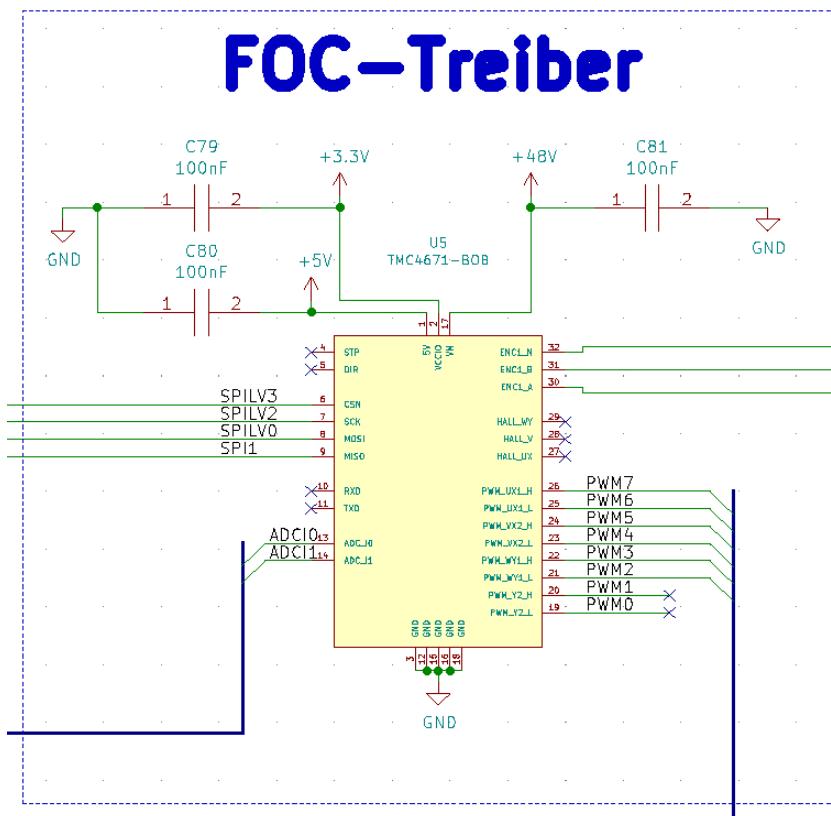


Abbildung 4.7: Schema FOC-Treiber.

<sup>1</sup>FOC= Field Oriented Control

## Funktionsbeschrieb der Schaltung

Im Folgenden werden kurz die Teilsysteme auf dem TMC4671-BOB beschrieben, welche für den Cocktailmixer gebraucht werden. Dazu wird auf das Layout vom Trinamic TMC4671-BOB zurückgegriffen. Sämtliche Teilschemas sind im Anhang Kapitel E.2 aufgezeigt. [4]

Der SPI-Kommunikationseingang ist nicht speziell geschützt. Hier ist für das eigene Layout darauf zu achten, dass die angelegte Spannung nicht über 3.3V steigt. Dies wird mit einem Level-Shifter zwischen Mikrocontroller und TMC4671 gewährleistet.

Der Eingang zur Strommessung ist mit einem Tiefpass geschützt. Dieser hat die Zeitkonstante:

$$\tau = R308 \cdot C300 = 100\Omega \cdot 100pF = 10ns \quad (4.4)$$

Um den Eingang der Encoder-Pins zu schützen, wird ein Schmitt-Trigger mit eingebautem Level-Shifter verwendet. Zudem wird mit dem 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:

$$\tau = R200 \cdot C204 = 1k\Omega \cdot 100pF = 100ns \quad (4.5)$$

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:

$$\tau = (R200 + R210) \cdot C204 = (1k\Omega + 4.7k\Omega) \cdot 100pF = 570ns \quad (4.6)$$

Jetzt geht es bedeutend länger, bis der Kondensator wieder geladen wird. Sobald die Spannung wieder einen bestimmten Wert erreicht, wird der Schmitt-Trigger aktiv und springt auf Versorgungsspannung (3.3V). Im Anhang Abbildung E.5 ist die Funktion verdeutlicht.

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

$$U_{TMC} = U_M \cdot \frac{R310}{R310 + R311} = 48V \cdot \frac{1k\Omega}{1k\Omega + 100k\Omega} = 0.7V \quad (4.7)$$

Filter vorhanden,  
siehe datenblatt

Die Ausgangssignale für den Gate-Treiber gehen direkt auf die Header-Pins des TMC4671-BOB.

### 4.2.2 Gate-Treiber

Der Motor wird über eine H-Brücke gesteuert. Dies bedingt pro Spule zwei MOSFET's, um diese entsprechend magnetisieren zu können. Um einen MOSFET in einen leitenden Zustand zu bringen, muss das Gate des MOSFET's mit einer elektrischen Ladung gefüllt werden. Während diesem Vorgang hat am Gate ein kapazitives Verhalten, was bedeutet, dass bei jedem Schaltvorgang Ströme fliessen. Um ein optimales Schaltverhalten zu erreichen, die Umschaltverluste zu verringern und somit die daraus entstehende Abwärme zu verhindern, ist es vorteilhaft die Gates so schnell wie möglich laden und entladen zu können. Da kommt der Gate-Treiber ins Spiel. Dieser ladet und entladet das Gate schnell genug und stellt die dazu benötigte Energie für das Gate zur Verfügung.

## Schema

Damit ein Aufbau mit mehreren Gate-Treibern vermieden werden kann und einige Zusatzfunktionen wie Strommessung, Messverstärkung, Kurzschlussdetektion etc. genutzt werden können, wurde während des Entwicklungsprozesses ein Gate-Treiber von Trinamic ausgewählt. Das entsprechende Bauteil ist der TMC6200. Das Blockdiagramm und eine Beispielschaltung befinden sich im Anhang Kapitel F.1 und F.2. Das Blockdiagramm zeigt, dass der TMC6200 aus einer Treiber-Logik für den Motor, einem SPI-/Pinsettings-Interface, einer Diagnosenlogik, Strommessschaltung und diversen unterstützenden Schaltungen wie Spannungsversorgung besteht.

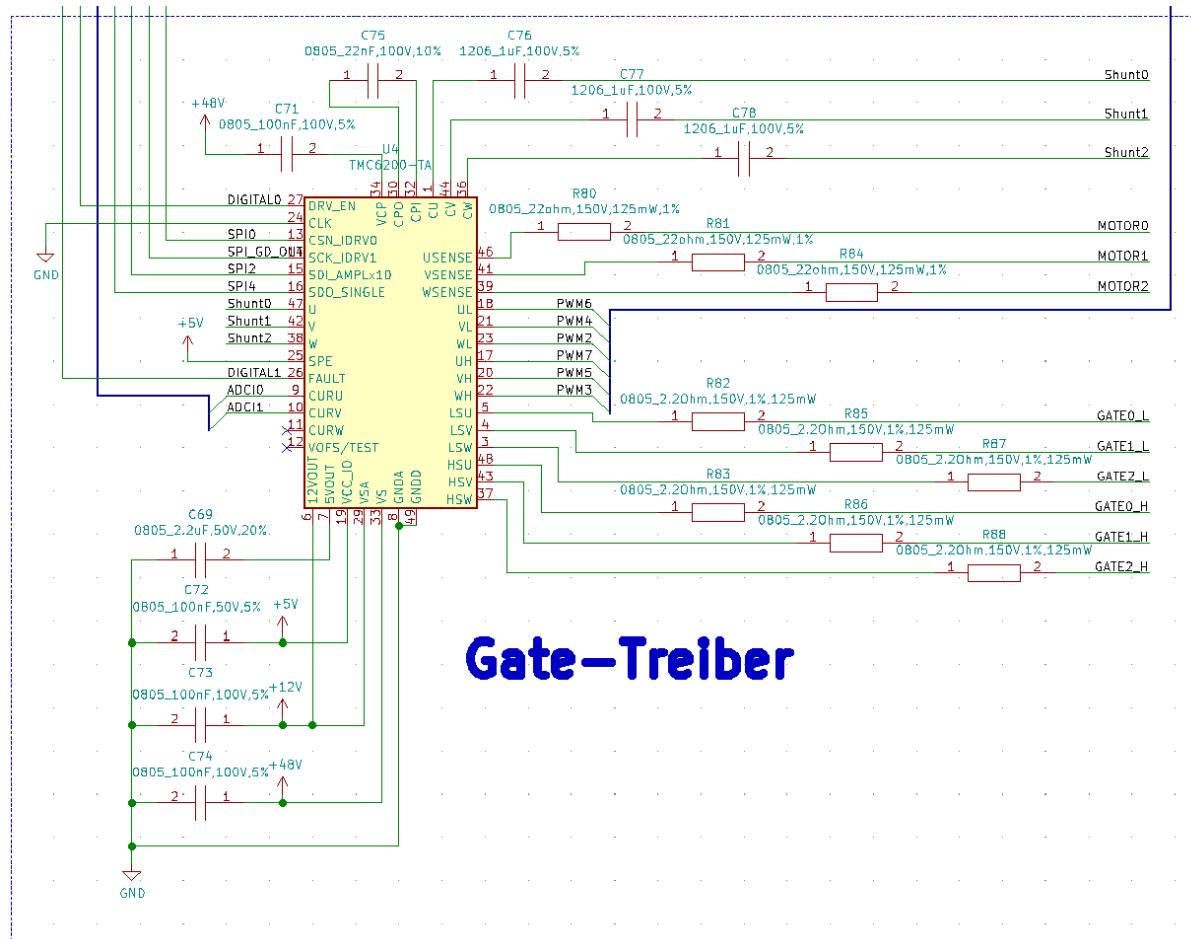


Abbildung 4.8: Schema Gate-Treiber.

## Funktionsbeschrieb der Schaltung

Für die Messung des Stromes wird ein Shunt in Serie mit der Spule des BLDC-Motors geschaltet. Die Shunts R89 - R91 sind in Abbildung 4.9 zu sehen. Durch den fliessenden Strom liegt eine Spannung über dem Shunt, welche dann vom TMC6200 verstärkt wird und aufbereitet an den Kontroll-Chip TMC4671 ausgegeben wird. Die Verstärkung sowie Offset auf das Signal sind über SPI programmierbar. Die Dimensionierung der Widerstände und die darauf folgende Programmierung des TMC6200 sind schon von Trinamic ermittelt worden. Es wird empfohlen, die Widerstände nach dem Maximalstrom des Motors auszuwählen. Im Falle des AKM22h sind dies 5A. Darüber hinaus wird empfohlen, eine Überdimensionierung von 25-50% vorzunehmen. Die Schaltung für den Partymixer wird auf 10A ausgelegt. Eine von Trinamic erstellte Tabelle ergibt

empfiehlt einen Widerstand von  $10m\Omega$  für die Shunt-Widerstände und einen Verstärkungsfaktor der Phasenströme von 10. Die Tabelle ist im Anhang Kapitel F.4.3 zu finden. [2, S.31]

Da das Gate ein kapazitives Verhalten zeigt, ist der Strom, welcher zu Beginn ins Gate fliesst, sehr hoch. Der Gate-Vorwiderstand begrenzt diesen, um das Gate und den Pin des TMC6200 vor Überströmen zu schützen. Die Vorwiderstände R82 - R83 und R85 - R88 sind in Abbildung 4.8 zu sehen. Die Dimensionierung der Gate-Widerständen wurde an die MOSFET Gate-Drain-Ladung (Miller charge) angelehnt. Auch in diesem Falle wurden von Trinamic schon einige Parameter ermittelt und im Anhang F.4.4 dargestellt. Die Gate-Ladung des ausgewählten MOSFET's beträgt  $61nC$ . Aus der Tabelle kann somit ausgelesen werden, dass der Vorwiderstand  $R_{Gate} \leq 2.5\Omega$  sein sollte und das programmierbare Register DRV\_STRENGTH auf 1 bis 3 gesetzt wird. [2, S.13]

Wird von einem High-Zustand in einen Low-Zustand gewechselt, kann aufgrund von Induktivitäten der Shunts oder deren Verbindungen die Spannung unterschiessen. Der Schutzwiderstand schützt den Messeingang des TMC6200 vor diesem Effekt. Die Schutzwiderstände R80, R81 und R84 sind in Abbildung 4.8 zu finden. Die Dimensionierung dieses Widerstands wird im Datenblatt mit einem Widerstandswert zwischen  $10\Omega$  und  $22\Omega$  angegeben. [2, S.10]

Die Bootstrap-Kondensator werden benutzt, um die Gate-Spannung am High-Side-MOSFET auf die Schaltspannung (48V) plus die Gatespannung anzuheben. Die Kondensatoren C76 - C78 in Abb. 4.8 unterstützen diesen Effekt. 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. Da die Kapazität  $61nF$  beträgt, ist dies der Fall. Die Bootstrap-Kondensatoren werden deswegen mit  $1\mu F$  dimensioniert. [2, S.10]

Aufgrund der hohen Versorgungsspannung (48V), treten im IC über den Gate- und 5V-Spannungsreglern erhebliche Verluste auf. Um diese Verluste zu reduzieren, wird gemäss Datenblatt [2, S.11] geraten, eine externe Gate-Spannung anzuhängen. Für beste Resultate wird eine Spannung von  $12V \pm 1V$  empfohlen. Die Dimensionierung der benötigten Kondensatoren ist im Anhang Kapitel F.3 zu finden. Für den Kondensator C69 sollten  $2.2\mu F$  verwendet werden und für den Kondensator C73  $100nF$ .

#### 4.2.3 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.

#### Schaltungsaufbau

Für den Aufbau und die Dimensionierung wurde das Referenzschema des Evaluationsboard verwendet, welches schon im Projekt 5 zum Einsatz gekommen ist (Trinamic UPS 10A70V) [5]. Das Schema ist im Anhang Kapitel G.1 zu finden. Die Dimensionierung der Shunts wurde in Kapitel 4.2.2 abgehandelt.

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.

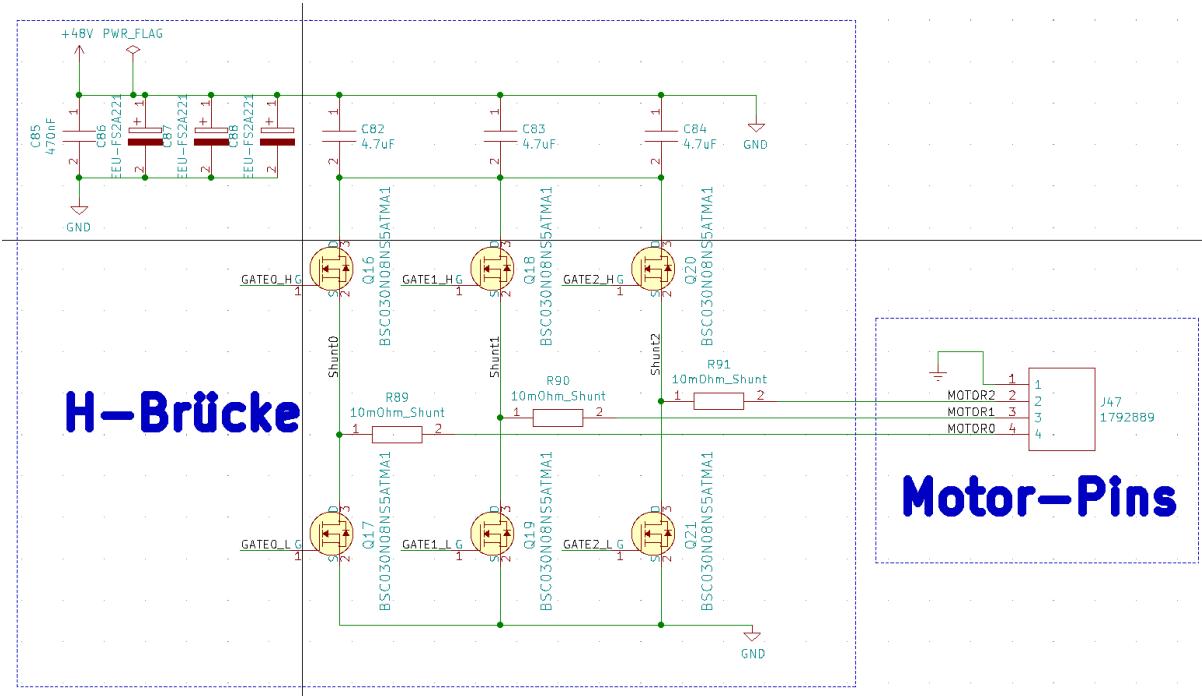


Abbildung 4.9: H-Brücke.

### Funktionsbeschrieb der Schaltung

Die Eingangsspannung der gesamte H-Brücke wird mit den Kondensatoren C85-C88 gestützt. Bei C86-C88 handelt sich um Low-ESR Stützkondensatoren, jeweils ein Kondensator pro Phase.

Die Kondensatoren C82-C84 sind Kondensatoren, welche direkt zwischen Ein- und Ausgang eines H-Brücken-Strangs, nahe der MOSFETs platziert wurden. Sie dienen auch zur Erhaltung der 48V-Versorgungsspannung.

Die MOSFETS Q16-Q21 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  $\pm 20V$ .

#### 4.2.4 BLDC

Ein BLDC zeichnet sich dadurch aus, dass der Rotor mit Permanentmagneten bestückt ist. Somit ähnelt er vom Aufbau her einer Synchronmaschine mit permanenterregten Rotorwicklungen. Zur Ansteuerung hat der BLDC drei Phasenleitungen, welche 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.

### Schaltungsaufbau

Der Motor wird direkt auf die vorgesehenen Anschluss-Pins geführt. Obwohl die Versorgungsspannung unterhalb der maximalen Berührungsspannung liegt wurde im Falle eines Defekts im Motor zur Personensicherheit ein Pin für die Erdung des Motorengehäuses vorgesehen. Die Motoranschlüsse auf dem Print sind in Abbildung 4.9 sichtbar.

## Funktionsbeschrieb der Schaltung

Hier gibt es nicht viel zu beschreiben, der Anschluss des Motors ist selbsterklärend. Lediglich die Phasenfolge ist zu beachten, damit das Drehfeld stimmt.

Der Motor an sich lässt sich wie folgt beschreiben [6, S.36]

Nennnetzspannung	$V_M$	=	48	[V]
Stillstanddrehmoment	$M_0$	=	0.88	[Nm]
Nenndrehmoment	$M_n$	=	0.85	[Nm]
Spitzendrehmoment	$M_{0max}$	=	2.8	[Nm]
Nenndrehzahl	$n_n$	=	1500	[min <sup>-1</sup> ]
Nennleistung	$P_n$	=	0.13	[kW]
Stillstandstrom	$I_0$	=	5.41	[A]
Nennstrom	$I_N$	=	5.21	[A]
Spitzenstrom	$I_{max}$	=	21.6	[A]
Drehmomentkonstante	$k_T$	=	0.1632	[Nm/A]
Rotorträgheitsmoment	$J$	=	0.16	[kg·cm <sup>2</sup> ]
Gewicht	$m$	=	1.1	[kg]

### 4.2.5 ABN-Encoder

Der ABN-Encoder wird verwendet, um dem FOC-Regler die momentane Lage des Rotors mitzuteilen. Dazu wurde der AMT33 im Verlaufe des Projektes 5 ausgesucht und hier beschrieben. Er ist unempfindlich auf Staub, Schmutz und Öl. Die Montage und Zentrierung gestaltet sich sehr einfach. Ausserdem hat er eine einstellbar hohe Genauigkeit. [7, S.1]

## Schaltungsaufbau

Nebst der 5V-Spannungsversorgung sind die Signalleitungen auf den Encoder geführt. Nämlich A, B und N. 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 eingegangen.

## Funktionsbeschrieb der Schaltung

Der ABN-Encoder kann wie gesagt eine einstellbar genaue Genauigkeit ausgeben. Dies zeigt sich in einer Form der Bitanzahl pro Umdrehung. Die maximale Auflösung liegt bei 4096 Schaltvorgänge pro Umdrehung, was einer Auflösung von 12 Bit entspricht. Dies ist wichtig für die Implementierung in den FOC-Treiber. Die Ausgangspins sind normale Header-Pins. Der Spannungsausgang wurde mit einem Stützkondensator C89 versehen.

### 4.3 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.

#### 4.3.1 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 relativ guten Durchfluss bieten können. Ein weiterer Vorteil dieser Pumpen ist, dass durch die Erstellung eines Vakuums auch Luft gepumpt werden kann. Dies ermöglicht auch einen «Kaltstart» auch ohne Flüssigkeit in den Schläuchen.

#### Schema

Die Schaltung zur Ansteuerung der Pumpen ist relativ simpel aufgebaut und in Abbildung 4.10 zu sehen. Sie besteht aus einer kleinen Leistungsstufe, welche mit einem Mosfet realisiert wurde, welcher 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 wurde für alle zwölf Pumpen umgesetzt.

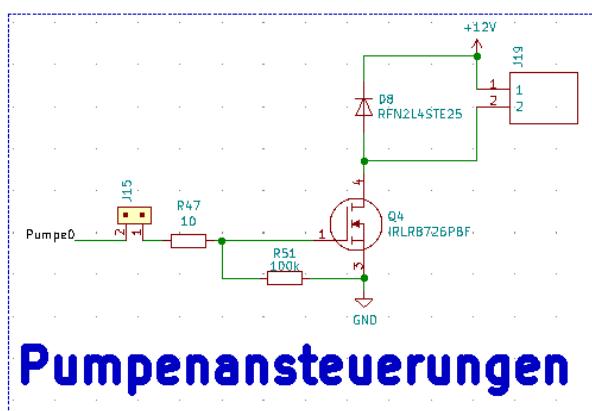


Abbildung 4.10: Schema der Pumpenansteuerung.

#### Funktionsbeschrieb der Schaltung

Die Pumpen werden jeweils direkt über den Mikrocontroller angesteuert. Dabei wird für jede Pumpe ein Digitalpin gemäss 4.7 verwendet. Da der Mikrocontroller nicht genügend Ausgangstrom bietet, um die Pumpen ansteuern zu können, wurde eine kleine Leistungsstufe mit einem Mosfet (hier Q4) implementiert gemäss Abbildung 4.10. Es handelt sich bei diesem Mosfet um einen Logic-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 10Ohm (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 wurde vor diesem Widerstand ein Jumper implementiert.

Dies ermöglicht es die Ansteuerung vom Mikrocontroller ab zu koppeln. Da es sich bei dieser Pumpe um einem DC-Motor und somit um eine Induktion handelt, wurde parallel zur Pumpe am Stecker eine Freilaufdiode implementiert. Diese schützt den Mosfet vor Spannungsspitzen beim Ausschalten.

### 4.3.2 Durchflussmessgeräte

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

#### Schema

Das Schema für diese Schaltung ist in Abbildung 4.11 zu sehen. Es beinhaltet lediglich einen Filterkondensator an der Speisung und einen Stecker, an welchem die Durchflussmessgeräte angeschlossen werden können.

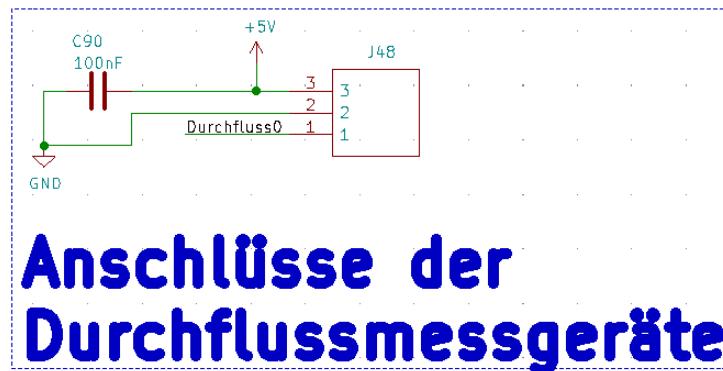


Abbildung 4.11: Schema der Durchflussmessgeräte.

#### Funktionsbeschrieb der Schaltung

Die Durchflussmessgeräte haben drei Anschlüsse. Zum einen ist dies einen 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 4.7 wiederum Digitalpins als Eingänge verwendet.

## 4.4 Programmierschnittstellen

Auf der Leiterplatte des PartyMixer's gibt es zwei Komponenten, welche programmiert werden müssen. Der Mikrocontroller und das WiFi-Modul. Um diese zu programmieren, braucht es eine entsprechende Schnittstelle, welche im Falle des Partymixers mit einer USB-B-Schnittstelle realisiert wird.

Die USB-B-Schnittstelle benötigt nur zwei Kommunikationsleitungen (D+ und D-). Die zu programmierenden Komponenten (ESP32 und ATMega2560) benötigen zusätzliche Steuerleitungen um in einen Programmiermodus zu kommen und statt einem differenziellen Verfahrens ein serielles Verfahren. Deswegen wird ein USB-UART-Converter verwendet, welcher das Signal wandelt und die Steuerleitungen zur Verfügung stellt. [8]

Die beiden Programmierschnittstellen (ATMega2560 und ESP32) benötigen unterschiedlich viele Steuerleitungen, da sie sich im Verfahren zum Aufruf des Download-Boot-Modus unterscheiden. Im Anhang Kapitel A.2 und A.3 wird dargestellt, welche Leitungen für das jeweilige Flash-Interface benötigt werden. Die Signalfolge, welche angelegt wird um den Programmier-Modus zu starten, wird auch Handshake genannt.

### 4.4.1 Handshake ATMega2560

Der Handshake zwischen Computer und ATMega2560 ist einfacher zu verstehen als der Handshake zwischen Computer und ESP32. Beim ATMega2560 wird jedes Mal, nachdem der Mikrocontroller neu gestartet wird, der Bootloader aufgerufen. Deswegen reicht es für die Übermittlung des Programmcodes die Reset-Leitung auf 0V zu ziehen. Dies wird erreicht, indem der wiring-Programmer verwendet wird, welcher auf den verwendeten Bootloader angepasst ist und vor dem Hochladen die DTR-Leitung toggelt. Nachdem das Programmiertool AVRdude die Device-Signatur (0x1E9801) verifiziert hat, sendet es das kompilierte HEX-File an den ATMega2560. Mehr zum Bootloader in Kapitel ??.

### 4.4.2 Handshake ESP32

Nach einem Neustart des ESP32 werden die Zustände der Strapping-Pins gelesen. Anhand dieser Zustände werden die Grundzustände des ESP32 gesetzt, bevor der Programmcode gestartet wird. Anhand der Strapping-Pins wird auch der Download-Boot-Modus gestartet. Mit einem Handshake zwischen Computer und ESP32 wird genau dies erreicht. Das ESP32 wird neu gestartet und der Download-Boot-Modus gesetzt. Um den Handshake zwischen Computer und ESP32 zu verstehen, muss zuerst Tabelle 4.1 betrachtet werden, worin die Strapping-Pins aufgelistet sind und welchen Einfluss diese auf denn Boot- oder Download-Modus haben. 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. [9, S.12-S.14]

Boot-Mode Konfiguration			
Pin	Default	Boot	Download
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 4.1: Wenn U0TXD, IO2, IO5 floating sind, bestimmt IO0 den Boot-Modus.

### 4.4.3 USB-B

Im Folgenden wird der in Kapitel 4.4 erwähnte USB-Anschluss beschrieben. Es handelt sich dabei um einen USB-UART-Converter mit zugehöriger Beschaltung wie z.B USB-B-Buchse, ESD-Schutz und weitere passive Komponenten.

#### Schema

Auf dem Schema ist die USB-Buchse mit J2 gekennzeichnet. Der dazugehörige ESD-Schutz bildet die Diodenschaltung D2-4. Der Converter hat die Bezeichnung IC2. Die Widerstände R12 und R13 bilden einen Spannungsteiler, welcher an die 5V-Eingangsspannung der Buchse und an dem VBUS-Eingang des Converters hängt. Die Widerstände R16, R19, R10 und R11 reduzieren den Kurzschlussstrom zwischen dem USB-UART-Converter und dem ESP32. Der Widerstand R14 ist ein Pull-Up Widerstand, welcher verhindert, dass der Reset-Pin einen ungewünschten Zustand annimmt. Die Widerstände R17 und R18 reduzieren den Strom durch die LED's.

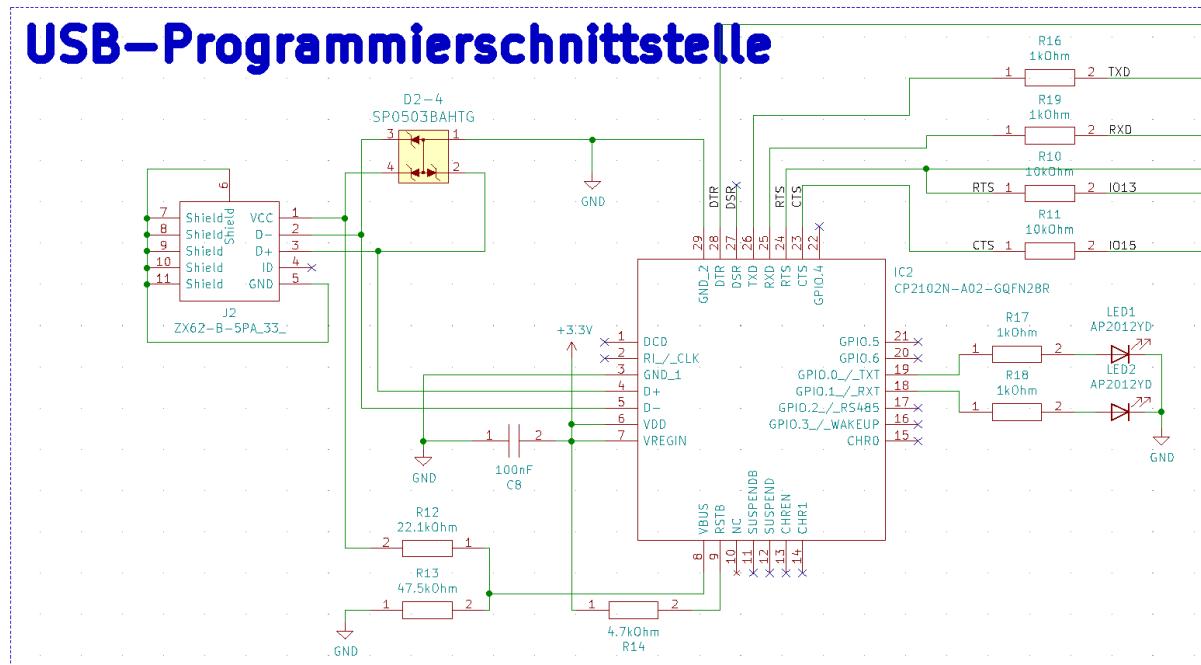


Abbildung 4.12: Schema USB-B.

#### Funktionsbeschrieb der Schaltung

Sobald ein USB-Kabel vom PC zum Converter angeschlossen wird, liegt 5V am Spannungsteiler, was vom IC als bestehende Netzwerk-Verbindung interpretiert wird. Die USB-Kommunikation zwischen PC und Converter findet über die Leitungn D+ und D- statt. Aufgrund des differenziellen Verfahrens und Verwendung von verdrillten Drähten werden Störungen von ausserhalb zu einem grossen Teil eliminiert. Der Vorteil der asynchronen serielle Schnittstelle (UART) zeigt sich im Falle der Cocktailmaschine vor allem im flexiblen Protokoll und dem einfach zu implementierenden Handshake, welcher benutzt wird, um in den Boot-Modus der Empfänger (ATMega2560 und ESP32) zu gelangen. Die LED's zeigen an, wenn eine Kommunikation stattfindet. Die Widerstände in den Kommunikations- und Steuerleitungen schützen die Pins vor hohen Strömen im Falle von Spannungsunterschieden.

## 4.5 Benutzerschnittstellen

Im Folgenden werden die Benutzerschnittstellen, welche sich auf der Leiterplatine befinden erklärt. Die vorkommenden Interfaces sind mit ihren Funktionen tabellarisch aufgelistet:

Interface	Funktion
Display	Direkte Steuerung zur Cocktailherstellung und Konfiguration der Maschine mit GUI auf dem Display.
ESP32	Indirekte Steuerung zur Cocktailherstellung und Konfiguration der Maschine mit GUI im Browser.
RFID	Abrufen des per Webhost oder Display hinterlegten Getränkes mit Badge. Men genauswahl per Display.

### 4.5.1 Display

Über das Display geschehen die Hauptinteraktionen mit dem Benutzer. Für die Bereitstellung des GUIs wird auf die Software Nextion Editor zurückgegriffen. Über diese Software lassen sich die Seiten, welche im Voraus mit viel Aufwand durchdacht wurden, sehr gut und einfach erstellen. Die Kommunikation, bestehend aus den Page- und Button-IDs, welche bei Drücken der Buttons gesendet werden, findet über UART statt.

#### Schema

Das Schema gestaltet sich eher einfach, da die Logik auf dem Display schon vorhanden ist. Es braucht lediglich einen Stecker, welcher mit J43 beschriftet ist und einen Stützkondensator nahe der Pins am Spannungsausgang.

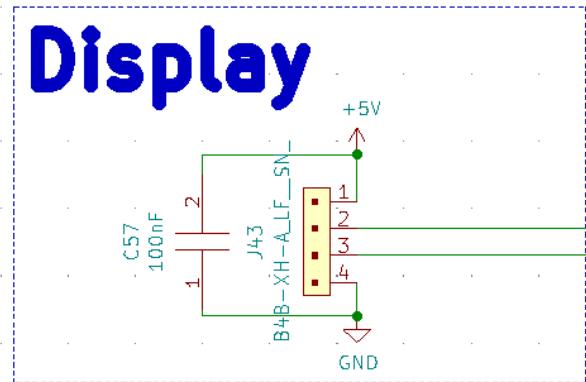


Abbildung 4.13: Schema Display-Stecker.

#### Funktionsbeschrieb der Schaltung

Die mehr oder weniger einzige Funktion, welche im Schema erkennbar ist, wird durch das Stützen der Versorgungsspannung mit dem Kondensator gegeben. Der Stecker ermöglicht den Anschluss des Displays an die Leiterplatine.

### 4.5.2 ESP

Für die Implementierung des kabellosen Zugriffs wird das ESP32 verwendet. Die Hauptfunktion auf dem Print ist die Kommunikation mit dem Mikrocontroller über die zweite serielle Schnittstelle. Die Ansteuerung zum Schreiben des Programmspeichers wird so implementiert, dass der Boot-Modus automatisch gestartet werden kann, wenn ein Code hochgeladen werden soll. Dies hat eine zusätzliche Schaltung zur Folge.

Aufgrund schon bestehender Erfahrungen wurde ein Espressif ESP-Modul ausgewählt. Grundsätzlich standen zwei Modelle zur Auswahl. Das ESP8266 und das ESP32. Für die Cocktailmachine wurde das ESP32 ausgewählt, da dies Leistungsstärker ist. Die genauen Datenvergleiche sind im Anhang Kapitel C.1 ersichtlich. Das ESP32 unterstützt das Protokoll nach ISO 802.11 b/g/n und kann somit auf 2.4GHz sowie 5GHz arbeiten. Mit dem n-Protokoll und einer Antenne kann so bis zu 150MBit übertragen werden bei einer Bandbreite von 20MHz.

Wichtig ist, dass ein ESP ausgewählt wird, welches einen Anschluss für eine abgesetzte Antenne hat, da die Leiterplatte, worauf das Modul verbaut wird, im Gehäuse platziert wird. Dazu eignet sich der Espressif ESP32-32U.

Im Anhang Kapitel C.2 sind die Strapping-Pins aufgelistet, welche während dem Startvorgang einen Einfluss auf die Konfiguration des ESP32 haben. Mit diesen Pins kann die Ausgangsspannung des internen Spannungsregler für VDD\_SDIO, der Bootmodus, der Debug-Log-Print während dem Booten sowie das SPI-Timing der Kommunikation mit der SDIO-Schnittstelle des ESP32 konfiguriert werden. Eine genauere Beschreibung der Pins befindet sich im selben Kapitel.

#### Schema (WiFi-Modul)

In Abbildung 4.14 wird das Schema rund um das ESP32 an sich gezeigt. Es beinhaltet Stützkondensatoren sowie einige Pull-up- und Pull-down-Widerstände, welche verwendet werden, um einen vordefinierten Grundzustand beim Booten des ESP32-Moduls zu erreichen (Strapping-Pins). Weiter gibt es einen Kondensator, welcher dazu da ist, bei gewünschter Zeit in den Download-Boot-Modus zu gelangen. In diesem Modus kann die User-Applikation übertragen werden.

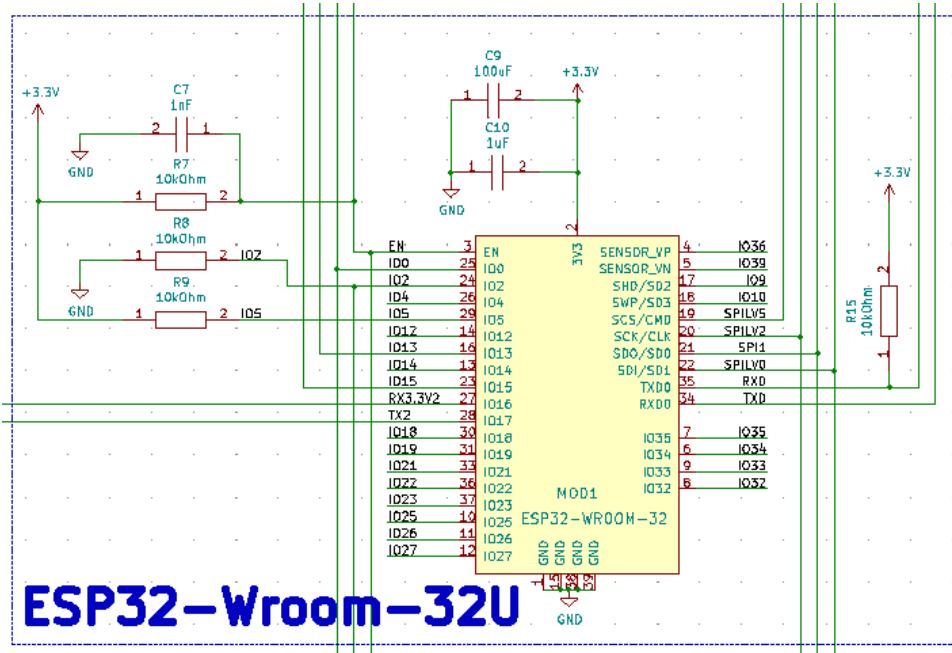


Abbildung 4.14: Schema ESP32-Wroom-32U.

### Funktionsbeschrieb der Schaltung (WiFi-Modul)

Das ESP32 an sich 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 Widerstand R7 zieht diese Leitung auf 5V. Die Widerstände R8, R9, R10 und R11 sind an die Strapping-Pins angeschlossen. Details zu den Konfigurationen sind in Tabelle C.2 aufgelistet<sup>2</sup>. Der Widerstand R15 zieht U0TXD auf HIGH. Aus Tabelle 4.1 ist ersichtlich, dass dieser für den normalen Boot-Modus auf HIGH sein muss. Für den Download-Boot-Modus hat dieser kein Einfluss. Mit dem Kondensator C7 wird sichergestellt, dass nach einem Reset der Download-Boot-Modus gestartet werden kann. Die Interaktion zwischen Computer und ESP32 wird auch Handshake genannt. Wie dieser genau funktioniert, wird in Kapitel 4.4.2 erklärt.

### Schema (Automatische Boot-Logik)

In Abbildung 4.15 wird die Schaltung gezeigt, welche verwendet wird, das ESP32-Modul in den gewünschten Boot-Zustand zu bringen. Für die Beschaltung der automatischen Boot-Logik benötigt es eine Schaltung mit DTR und RTS als Inputs vom USB-UART-Converter her und EN und IO0 als Outputs auf das ESP32. Die Buttons können bei Bedarf verwendet werden, sind für den automatischen Boot-Modus jedoch nicht zwingend nötig. Sie könnten dazu verwendet werden, manuell in den Bootmodus zu gelangen.

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 zum entprellen. Die Widerstände R25 und R24 begrenzen den Strom bei Drücken der Buttons S1 und S2.

<sup>2</sup>[https://www.espressif.com/sites/default/files/documentation/esp32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf) S.13

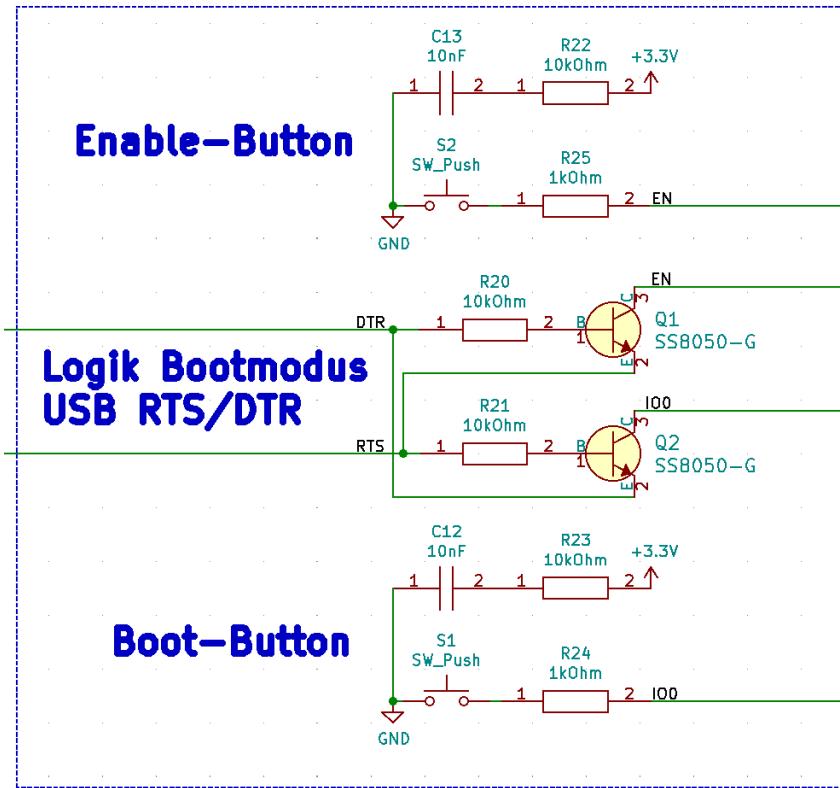


Abbildung 4.15: Schema ESP32-Wroom-32U.

### Funktionsbeschrieb der Schaltung (Automatische Bootlogik)

Mit der automatischen Bootlogik werden die Pins EN und IO0 so angesteuert, sodass das ESP32 automatisch neu gestartet wird und der Download-Boot-Modus gestartet wird. Das Verhalten des "Auto Program Circuit" und eine detaillierte Beschreibung des Vorgangs ist im Anhang Kapitel A.3.2 zu finden. Darin wird erklärt, auf welche Weise die DTR- und RTS-Leitung geschaltet werden müssen, damit die EN- und IO0-Leitung richtig gesetzt werden.

#### 4.5.3 RFID

Eine vorgesehene Funktion der Maschine ist, dass der User ohne Suchen sein Lieblingsgetränk zubereiten lassen kann. Dafür wird auf ein System zurückgegriffen, welches öfters für Zutrittskontrollen oder Ähnliches verwendet wird. Die Rede ist von RFID<sup>3</sup>.

Um schnell ein funktionierendes Teilsystem testen zu können, soll ein Breakout-Board verwendet werden. Eine Recherche hat ergeben, dass es für unseren Anwendungszweck nicht allzu viele Produkte gibt. Ein Eingrenzungskriterium war, dass es für den Ausgewählten IC eine bestehende Library gibt, welche mit Arduino oder besser C kompatibel ist. Der Chip, welcher diese Anforderungen erfüllt, ist der **Mifare MFRC522**. Dieses Breakout-Board kann außerdem an der Verschaltung der Maschine angeschraubt werden. Somit kann er sich an einer Stelle befinden, welche weiter weg von der Hauptsplatinne ist. Das Modul arbeitet auf 13.56MHz und gilt somit als kurzwelliges System (HF). [10]

Der MFRC522 ist der Reader im RFID-System. Er kann Daten lesen und schreiben. Er erzeugt ein hochfrequentes, elektromagnetisches Wechselsignal mit einer Frequenz von 13.56MHz.

<sup>3</sup>Radio Frequency IDentification

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 ein Reader detektieren.

Der Sender kann jedoch auch ein Datensignal über das Energiesignal modulieren. Die Informationen werden im Tag demoduliert und verwertet. Bei den Befehlen für den Tag geht es hauptsächlich um Lese- und Schreibmethoden. Bei einer Lesemethode des RFID wird ein Teil des Speichers abgefragt, bei Schreibmethoden wird der Speicher beschrieben. Die Reichweite für ein solches System, welches als Passivsystem deklariert ist, beträgt maximal 5cm.

Alternativ könnte ein Tag mit einer Batterie gespiesen werden. Somit erhöht sich die Reichweite des Readers, die Latenzzeiten werden kürzer und der Anwendungsbereich würde grösser. Solch ein System wird als Aktivsystem deklariert. [11]

## Schema

Bild 4.16 zeigt den Schaltungsaufbau des RFID-Moduls. Darin ersichtlich sind folgende Teilbereiche:

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

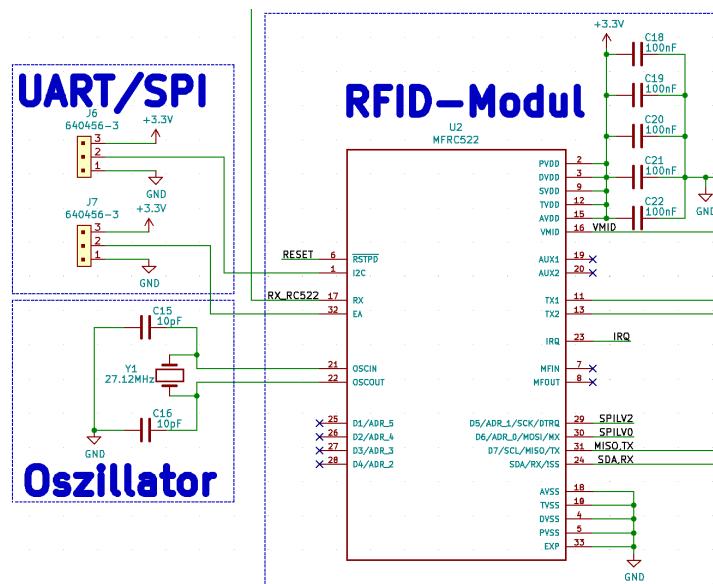


Abbildung 4.16: Schema des RFID Sender/Empfänger

## Funktionsbeschrieb der Schaltung

Der IC steuert den Ablauf, damit die Datenübertragung stattfinden kann. Er verbindet die Antenne mit dem Cocktailmixer und liest bzw. schreibt die Informationen auf das bzw. aus dem Trägersignal. Die Herausforderung besteht im Design des Anpassnetzwerkes an das IC und die Antenne. Wie die Bauteile dimensioniert werden, kann in der Annotation Note 1445 von NXP Semiconductors nachgesehen werden. [12].

Die Anschlüsse sind so gestaltet, dass falls eine einens gelayoutete Antenne verwendet wird, die Kommunikation zwischen UART und SPI gewählt werden kann. Aus Zeit- und Backupgründen wurde für die Cocktailmaschine ein Breakout-Board verwendet, welches nur einen SPI-Anschluss hat.

## 4.6 Beleuchtung

Damit die Maschine schon von Weitem erkennbar ist, wird ein auffallendes Mittel benötigt. Dazu eignet sich ein LED-Band hervorragend, am besten wenn es noch verschiedenfarbig ist. Dazu ist einerseits das LED-Band von Nöten und die geeignete Ansteuerung dazu. Für die Cocktailmaschine wurde festgelegt, dass der LED-Streifen die benötigten Widerstände für die LED's schon aufgeklebt hat und die Ansteuerung folglich direkt über FET's geschehen kann. Der Aufbau ähnelt dann dem in Abbildung 4.17 gezeigten Schaltung. Der Streifen wäre Rot eingerahmt, die LED's und Widerstände befinden sich auf dem Band und die zu sehenden Fähnchen für R, G, B und W führen zum Mikrocontroller. Es können auch mehrere Bänder parallel geschaltet werden, wie in Abbildung 4.17 ersichtlich ist.

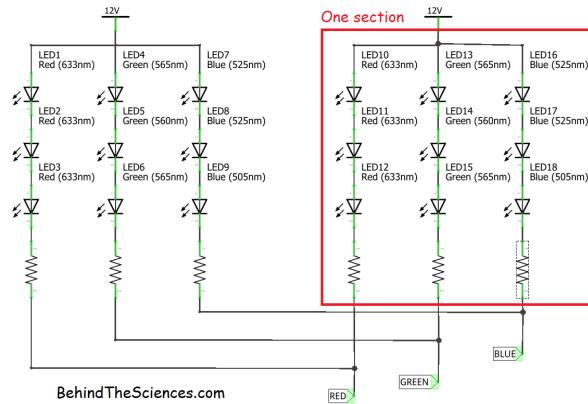


Abbildung 4.17: LED Beispiel. [13]

## Schema

Abbildung 4.18 zeigt den Schaltungsaufbau der LED-Steuerung. Damit die LED's angesteuert werden können, braucht es ein Bauteil, welches mit einer 5V-Ansteuerung 12V schalten können. Dazu wird ein MOSFET verwendet. Über die Widerstände an den Gates wird der Strom zum Schutz des Gates begrenzt. Die Leitungen führen direkt auf den Klemmblock für die LED-Streifen. Parallel dazu wurden noch für jede Lichtfarbe ein Kontroll-LED installiert, welche es ermöglicht auch ohne LED-Band etwas zu programmieren.

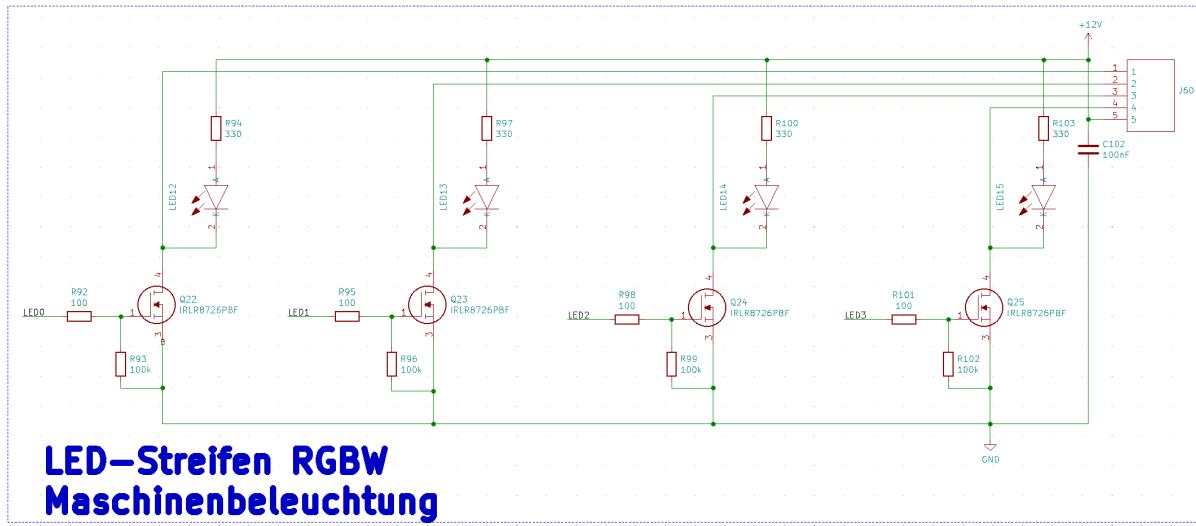


Abbildung 4.18: Schema der LED-Ansteuerung

### Funktionsbeschrieb der Schaltung

Licht sind bekanntlich elektromagnetische Wellen im sichtbaren Wellenlängen-Bereich. Dabei gibt es vier Hauptfarben: Rot, Grün, Blau und Weiss. Zwar kann Weiss auch aus einer Kombination aller drei Farben erstellt werden, 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 praktisch aus den Grundfarben praktisch alle Farben mischen.

## 4.7 Mikrocontroller

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

### Schema

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.

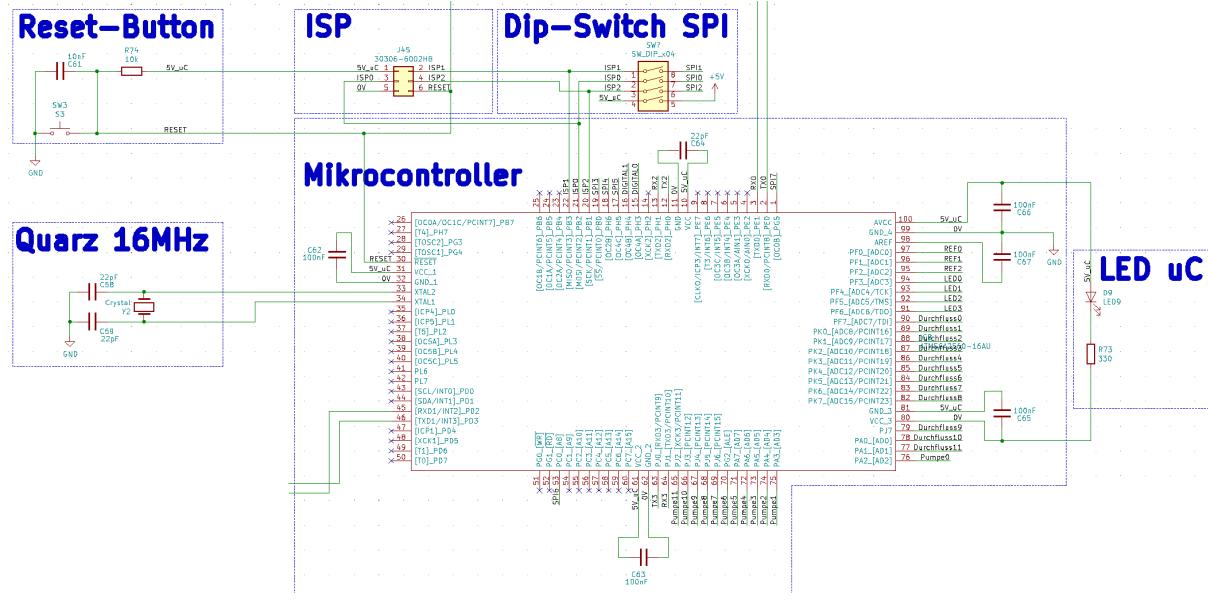


Abbildung 4.19: Schema Mikrocontroller

### Funktionsbeschrieb der Schaltung

Die Stützkondensatoren halten die Spannung am Mikrocontroller konstant. Die Quarz-Schaltung wird gebraucht, da der Mikrocontroller nur eine 8MHz-RC-Clock integriert hat, jedoch mit 16MHz gearbeitet wird. Der Reset-Button zieht in gedrücktem Zustand die Reset-Leitung auf GND. Dabei entlädt sich der Kondensator schnell, da er kurzgeschlossen wird. In offenem Zustand lädt sich der Kondensator über den Widerstand R74 mit einer Zeitkonstante von  $\tau = R_{74} \cdot C_{81} = 100\mu s$ . Die Trennung von SPI und ISP wird gemacht, sodass gewährleistet ist, dass bei der Inbetriebnahme des Mikrocontrollers die restlichen SPI-kommunikationsfähigen Komponenten nicht mitreden können. Zudem kann die 5V-Versorgungsspannung des Mikrocontroller vom Rest der Platine getrennt werden, sodass dieser auch bei ausgeschalteter Board versorgt werden kann, ohne die anderen Komponenten zu speisen. Deswegen auch die Betriebs-LED.

## 4.8 SD-Karte

Damit gespeicherte Getränke, Zutaten und Maschinenzustände bei einem Neustart der Cocktailmachine erhalten bleiben, werden diese auf der SD-Karte abgelegt. Neben dem Erhalten der Informationen ist die SD-Karte wichtig für den Betrieb der Maschine. Werden zu viel Getränke und Zutaten gespeichert, so führt dies zu einem Absturz des Mikrocontrollers. Es wurde entschieden, eine mikroSD zu verwenden. Im Folgenden wird erklärt, was es Hard- und Softwaretechnisch zu beachten gibt.

Abbildung 4.20 zeigt die Pinbelegung einer mikroSD-Karte. Es ist erkennbar, dass sie über SPI kommuniziert. Für den Anschluss wird nur ein mikroSD-Sockel benötigt. So kann die mikroSD über den Level-Shifter vom Mikrocontroller angesteuert werden.



Abbildung 4.20: Pinout des mikroSD-Sockels. [14]

Als Dateisystem wird FAT32 verwendet. 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. [15]

Der Speicherbereich einer FAT32-formatierten Partition besteht aus fünf Bereichen. [16]

1. Master Boot Record (LBA Sektor 0 des Laufwerks)
2. Volume Boot Record, Partition Boot Sektor (LBA Sektor 0 der Partition)
3. File Allocation Table (i.d.R 2 mal hintereinander vorhanden, direkt nach dem PBR)
4. Directory Table (mit den Ordner und Fileeinträgen)
5. Datenbereich (Fileinhalt)

Im Anhang Kapitel B.1.1 sind die einzelnen Bereiche detailliert beschrieben.

Für die Cocktailmaschine wird softwareseitig eine Library benötigt, welche die FAT32-Commands implementiert und gleichzeitig die Operationen auf der SD-Karte handelt. So ist die SD-Karte auch am Computer editierbar und die Befehlsliste reduziert sich auf readFile, writeFile und deleteFile.

## Schema

In Abbildung 4.21 ist das Schema der SD-Karte zu sehen. Darin erkennbar ist der mikroSD-Adapter J45 mit einem Stützkondensator C68 am Spannungseingang.

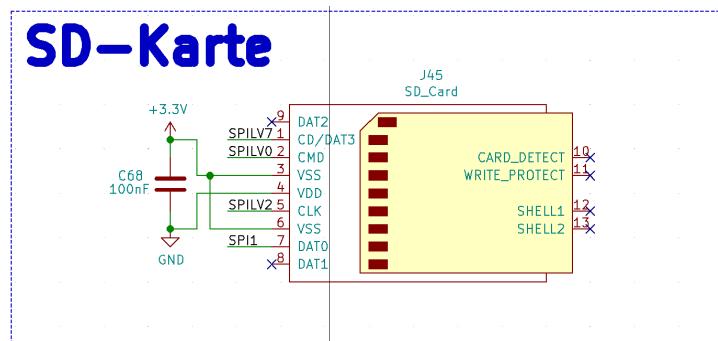


Abbildung 4.21: Schema SD-Karte.

## 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.

# 5 Inbetriebnahme

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

## 5.1 Speisungen

Um die Speisungen in Betrieb nehmen zu können ist gemäss Kapitel 4.1 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.

### 5.1.1 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.

### 5.1.2 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.

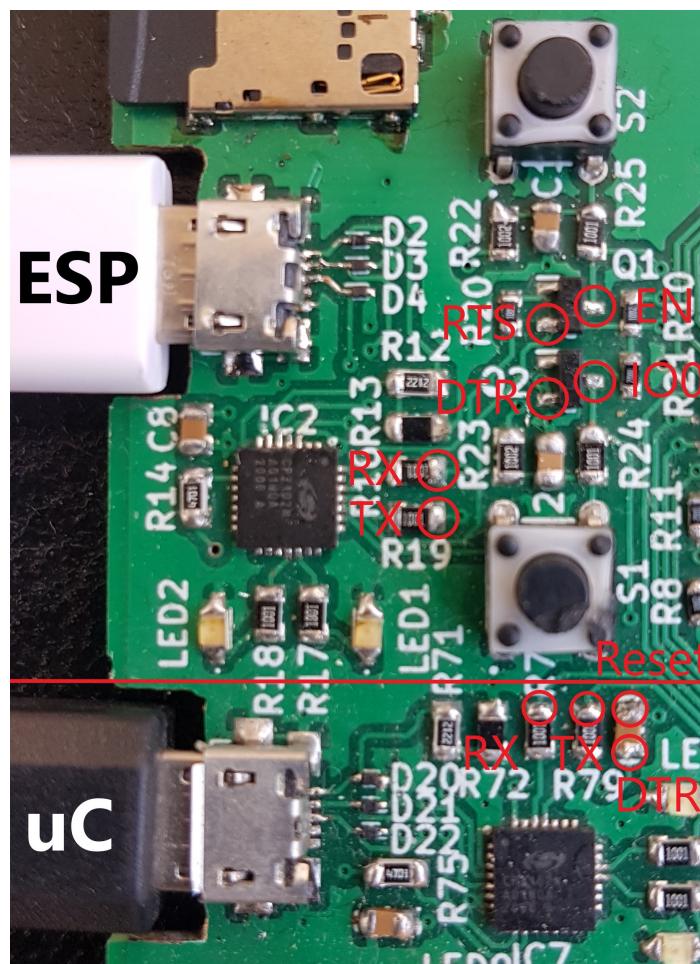
### 5.1.3 3.3V Speisung

Die 3.3V-Speisung wurde als letzte Speisung in Betrieb genommen, da diese von der 5V-Speisung gemäss Kapitel 4.1.4 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.

## 5.2 Programmierschnittstellen

Die Programmierschnittstellen werden jeweils mithilfe einen USB-UART-Controller gesteuert. Im Folgenden wird die Inbetriebnahme dieser Schnittstelle für den Mikrocontroller ATMega2560 und das WiFi-Modul ESP32 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 wird in den Kapitel 5.3.2 (WiFi-Modul) und Kapitel ?? (Mikrocontroller) beschrieben. Dies geschieht ein wenig parallel, denn die Programme zum Schreiben, Kompilieren und Hochladen des Codes müssen schon vor dem Prüfen der USB-B-Schnittstelle installiert sein. Für die Inbetriebnahme wurde entschieden, dass der Handshake noch zum USB-B-Teil gehört und als Voraussetzung zählt, sodass das Programm überhaupt hochgeladen werden kann. Außerdem wird dieser Handshake unabhängig davon gemacht, ob ein Device angehängt ist oder nicht. Erst nach einer Überprüfung der Device-Nummer kann bestimmt werden, ob das Target (ESP32 und ATMega2560) vorhanden ist oder nicht. Um den Handshake zu kontrollieren sind die in Abbildung 5.1 eingekreisten Punkte interessant:



**Abbildung 5.1:** Ausschnitt aus Platine (USB-Schnittstellen) mit eingekreisten Messpunkten.

Für die Inbetriebnahme der Programmierschnittstellen wurde mit der entsprechenden Programmierumgebung ein HEX-File mit dem darin enthaltenen Code übertragen und die in Abbildung 5.1 markierten Punkte gemessen.

Kim bitte bessere Bildmarkierungen einfugen, welche besser sichtbar sind. Danke

Folgende Schritte wurden befolgt:

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*. Die Verifizierung ist im Anhang Kapitel A.1 zu finden.

2. Überprüfen der Handshakes, welche definiert sind in den Tools zum Hochladen der Software.

- (a) ATMega2560

Um zu evaluieren, wie der Handshake durchgeführt wird, wurde beim Hochladen eines Programms an den markierten Punkten auf der unteren Seite in Abbildung 5.1 gemessen.

Im Anhang Kapitel A.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 uC.

- (b) ESP32

Um zu evaluieren, wie der Handshake durchgeführt wird, wurde beim Hochladen eines Programms an den markierten Punkten auf der oberen Seite in Abbildung 5.1 gemessen.

Im Anhang Kapitel A.3.2 wurden die ersten 100ms des Hochladens eines Programms auf das ESP32 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. Was bei genauerem Betrachten der Messungen auffällt: 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.3.2 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.

### 5.3 Benutzerschnittstellen

Die Benutzerschnittstellen sind sehr Unterschiedlich. Sie gemeinsames haben sie indem sie praktisch alle über die UART-Schnittstelle funktionieren.

#### 5.3.1 Touch-Display

Der Test für das Display gestaltet sich am schnellsten. Die Schnittstelle wird mit dem schon im P5 geschriebenen Code getestet. Für den Test wird erwartet, dass ein Druck auf das Bild in der Mitte mit dem Cocktail eine Funktion auslöst. In dieser Funktion wird auf dem Display ein anderes Bild geladen und die Texte in den Buttons geändert. Eine andere Funktion simuliert eine Zubereitung eines Cocktails. Beide Funktionen haben gleich wie im P5 einwandfrei funktioniert.

#### 5.3.2 ESP

Die Inbetriebnahme des ESP32-WROOM-32U geschieht mit der Programmierumgebung Arduino IDE. Um diesen in Betrieb nehmen zu können, waren einige Einstellungen und Downloads nötig, welche dann in der Arduino IDE eingebunden werden müssen. Im Folgenden wird beschrieben, wie vorgegangen werden kann damit sich im vorgegebenen Netz anmeldet und als Webhost dient. Es wird getestet, ob bei Klicken auf ein Button im Explorer ein Event auslöst wird und ob die Kommunikation zwischen µC und Mikrokontroller funktioniert.

Folgende Schritte wurden befolgt:

1. Benötigte ESP32-Bibliotheken vom *espressif*-Github-Account runterladen.[17]  
[Github search: espressif/arduino-esp32](#)
2. Speichern der heruntergeladenen Daten unter:  
[.../Dokumente/Arduino/hardware/espressif/esp32/ \(Inhalt: cores, docs, libraries, ...\)](#)  
Danach sollte das ESP32 Arduino erscheinen (in sketchbook). Dort kann nun das ESP32 Dev Module angewählt werden.  
[Werkzeuge →Board](#)
3. Unter demselben 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 flashbar.

4. Testprogramm aus dem Blog-Post von Randomnerdtutorials.com herunterladen, leicht modifizieren und hochladen. [18]

#### [Arduino IDE → Verify and Upload Button](#)

Für die Inbetriebnahme wurde das Testprogramm so modifiziert, dass das ESP32 gleich getestet werden kann wie das Touch-Display. Dies bedeutet, dass das ESP eine Page-ID, eine Button-ID und das Abschlusszeichen 0xFF 0xFF 0xFF sendet. Der Unterschied ist, dass das ESP über einen anderen UART-Port des µC kommuniziert. Die Testsoftware ist bei den anderen Firmwares auf dem USB-Stick zu finden.

5. Die "Debug Kommunikation" über den ersten seriellen Port des ESP32 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 ESP hat eine IP-Adresse zugeordnet bekommen.

Über die zweite serielle Schnittstelle findet die Kommunikation zwischen ESP32 und Atmega2560 statt. Sobald über den Webserver eine Aktion ausgelöst wird, sendet das ESP die gleiche Zeichenfolge, welche auch das Display ausgibt. 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.3.1 sind einige Auszüge aus der Arduino IDE aufgelistet.

### 5.3.3 RFID

Beim RFID-Leser handelt es sich wie in Kapitel ?? um das MFRC522 Evaluierungsboard (Breakout-Board). Dieses kommuniziert über SPI mit einem gewünschten Controller. Die Pinbelegung dazu ist in Abbildung 5.2 zu sehen. Der grosse Vorteil dieses Evaluierungsboard ist es, dass es dazu eine Arduino Library gibt mit vielen Beispielen.

Um den Leser in Betrieb nehmen zu können, wurde dieser wie folgt angeschlossen:

- Vcc: An 3.3V vom ESP32
- GND: An GND vom ESP32
- SDA: An IO5 vom ESP32
- SCK: An IO18 vom ESP32
- MISO: AN IO19 vom ESP32
- Reset: AN IO22 vom ESP32
- MOSI: AN IO23 vom ESP32

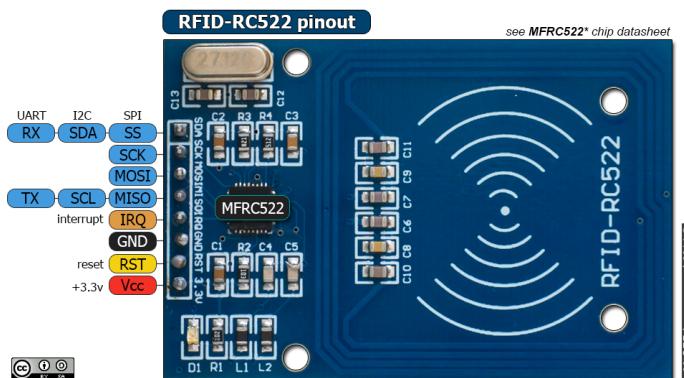


Abbildung 5.2: Anschauungsbild MFRC522 Evaluierungsboard

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 ESP32 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 Programmes 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.

Quelle:  
<https://www.RC522-pinout-5a5263ece3c12>

## 5.4 Mikrocontroller

Um das Anwenderprogramm auf dem Mikrocontroller ( $\mu$ C) speichern zu können, ist es am angenehmsten, wenn dies direkt aus der Programmierumgebung geschehen kann. Als Programmierumgebung wird aufgrund des AVR-Chips die Software Atmel Studio 7.0 ausgewählt.

Atmel Studio kompiliert den geschriebenen C Code in Maschinencode und schreibt diesen in ein HEX-File. Das HEX-File wird von einem ProgrammierTool namens AVRdude hochgeladen. <https://www.avrdude.com/>

Damit der Mikrocontroller sich Programmieren lässt, müssen einige Grundeinstellungen vorgenommen werden. Dazu gehört das Setzen der Fuse- und Lock-Bits sowie das Schreiben des Bootloaders. Eine detaillierte Beschreibung, wie diese gesetzt wurden, wird im Anhang Kapitel D.1 erklärt. Aus der Beschreibung folgt für die Fuse- und Lock-Bits die Einstellungen gemäss Tabelle 5.1.

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

**Tabelle 5.1:** 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. Für die Inbetriebnahme des Mikrocontrollers wurde der Alternativweg gewählt. Eine ausführliche Anleitung findet sich auf [instructables.com](http://instructables.com). [19]

Von seitens Software muss AVRdude in Atmel Studio eingebunden werden, was im folgenden Kapitel erklärt wird.

### 5.4.1 AVRdude in Atmel Studio einbinden

Von <http://savannah.gnu.org/> kann eine Datei heruntergeladen welche [avrduude-6.3-mingw32.zip](http://savannah.gnu.org/download/avr/avr-6.3-mingw32.zip) heisst. Der gleichnamige Ordner wird im Ordner <C:\Tools> gespeichert. [20]

Nachdem dies gemacht wurde, wird in AtmelStudio der Reiter ”Tools→External Tools” ausgewählt und ein neues Tool hinzugefügt. Im Falle des Atmega2560 geben wir die Commands gemäss Tabelle 5.2 ein:

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

**Tabelle 5.2:** AVRdude Commands

Der entsprechende **COMx**-Port des zu flashenden Gerätes (Atmega2560) muss mit dem Gerätemanager ermittelt werden.

Nun kann aus Atmel Studio das kompilierte HEX-File hochgeladen werden.

cite Herr  
Meier  
Skript mc1

Folgende Schritte wurden befolgt:

1. Als Erstes wurden die Fuse-Bits gesetzt. Dies geschah über den Reiter:  
[AtmelStudio → Tools → Device programming → Fuses](#) (siehe Abbildung D.5)  
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](#) (siehe Abbildung D.6)  
Hier wird ein stk500v2-BL verwendet.
3. Als Drittens wurden die Lock-Bits gesetzt unter:  
[AtmelStudio → Tools → Device programming → Lock-Bits](#) (siehe Abbildung D.8)  
Diese sollten nicht mehr geändert werden. Bei jedem Brennen des BL wieder zu setzen.
4. Mikrocontroller mit der kompilierten Software (Cocktailmixer.HEX) programmieren.  
[AtmelStudio → Tools → Cocktailmixer](#)

Wird der Mikrocontroller per Reset-Button neu gestartet, dauert es aufgrund des Bootloaders 2s, bis der Programmcode gestartet wird. Während dieser Zeit wartet der Bootloader auf einkommende Daten, welche auf den Flash-Speicher geschrieben werden sollen. Danach startet das Programm, sollten keine Daten kommen.

Sämtliche Tabellen aus dem Datenblatt und Screenshots aus Programmierumgebung, welche mit dem Setzen der Fuse- und Lock-Bits oder Programmierung des µC zusammenhängen, sind im Anhang Kapitel D.1 angefügt.

## 5.5 Datenspeicherung

Im Folgenden wird die Datenspeicherung in Betrieb genommen. Dies geschieht mit dem Programm, welches in der Library vorhanden ist. Damit kann auf das Directory Table zugegriffen werden und die darauf vorhandenen Files gelesen und bearbeitet werden. Es können auch neue Files generiert werden.

Die benutzte Library ist eine ältere Version. Die neueren Versionen sind auf der Homepage gezeigt und bieten einen noch breiteren Verwendungszweck. Ein wirklich gelungenes Projekt. [21]

### 5.5.1 mikroSD-Karte

Vorgehen:

1. Benötigte Applikation aus dem Software-Ordner auf dem USB-Stick in Atmel Studio öffnen.  
[Software→Atmega→2\\_Testapplikation\\_SD\\_Karte →SD\\_Karte\\_Testapplikation](#)
2. Software hochladen:  
[AtmelStudio →Tools →Cocktailmixer](#)
3. Programm für Kommunikation über die serielle Schnittstelle downloaden (z.B HTerm 0.8.1beta. )[22]
4. Verbindung mit Mikrocontroller herstellen und Neustart über Button DTR auslösen.

Port	=	<b>COMx</b>
Baudrate	=	9600
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.

5. Ob eine SD-Karte gefunden wurde, ist erkennbar, wenn die Version der SD-Karte angezeigt wird. Jetzt sollte sich die SD-Karte lesen und beschreiben lassen.

## 5.6 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.

### 5.6.1 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 es uns abzuschätzen wie lange es benötigt um einen Cocktail herzustellen.

In der folgenden Tabelle sind die durchschnittlichen Zeiten aus jeweils 5 Testläufen der einzelnen Pumpen zu sehen. Da es sich nicht um ein hochpräzises System handelt, wurde auf Sekunden gerundet.

### 5.6.2 Durchflussmessgeräte

Die Durchflussmessgeräte in Betrieb zu nehmen gestaltete sich ein wenig schwieriger als bei den Pumpen in Kapitel 5.6.1. 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.

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 gepump und geschaut, wie gross die Toleranz ist. In einem weiteren Schritt wurde dann das Zusammenspiel der einzelnen Pumpen gemessen.

## 5.7 Beleuchtung

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.

### 5.7.1 PWM-Signale LED

Das PWM-Signal für die LEDs wird mit einer Frequenz von 10 kHz initialisiert. Um dies zu erreichen, wird Formel 5.1 umgestellt nach 5.2. Der Prescaler ist  $N = 1$ . Das Ergebnis wird im Register OCRnA gespeichert, wodurch der Timer  $1/10\text{kHz} = 100\mu\text{s}$  benötigt, bis ein Timer\_nx Compare-Interrupt ausgelöst wird.

$$f_{OC_{max}} = \frac{f_{clk_{I/O}}}{2 \cdot N \cdot (1 + OCR_{nA})} \quad (5.1)$$

Robin Zeitmessung der einzelnen Pumpen einfügen, wie lange es benötigt um 5dl abzufüllen

Quelle:  
<https://www.lexikon/daten/dichtetabelle.pdf>

Tabelle mit den Ergebnissen einfügen. Muss nochmals im Bezugspunkt besprochen werden

umgestellt nach OCR\_nx:

$$OCR_{nx} = \frac{f_{clk_{I/O}}}{2 \cdot N \cdot f_{OC_{max}}} - 1 = \frac{16MHz}{2 \cdot 1 \cdot 10kHz} - 1 = 799 \quad (5.2)$$

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 5.3 und 5.4 zu sehen.

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	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 5.3: Timing-Diagramm CTC-Mode. OCnx nicht angeschlossen, Pin wird softwaremässig getoggelt.

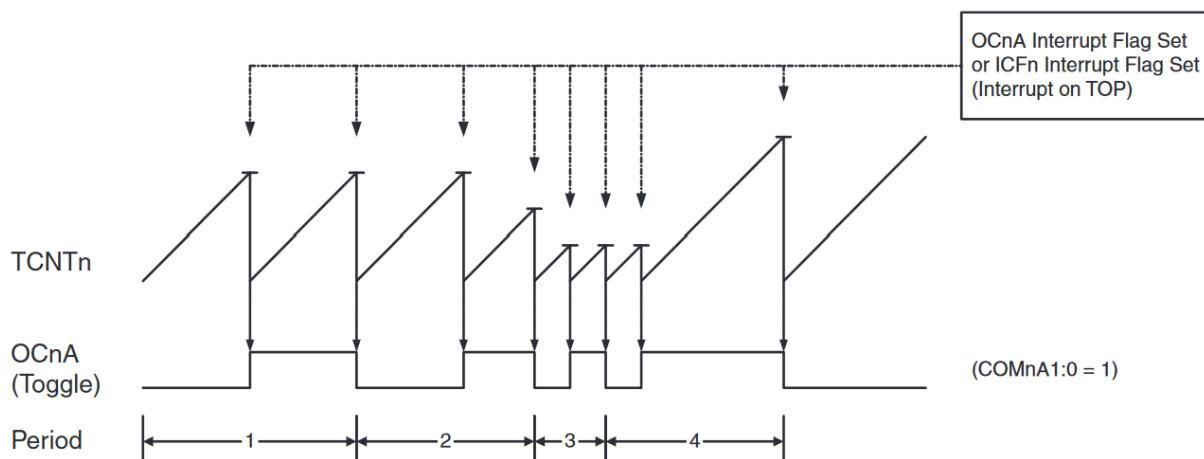


Abbildung 5.4: Timing-Diagramm CTC-Mode. OCnx nicht angeschlossen, Pin wird softwaremässig getoggelt.

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  $(800/2) - 1 \approx$

399. So wird nach der Hälfte der Hochzählzeit ein zweiter Compare-Interrupt ausgelöst, welcher jedoch kein Einfluss auf das Counter-Register hat.

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.

Der Wert für das OCRnA-Register ist folglich 799. Ein OCRnB-Register wird nicht benötigt, da die Iteration nur ausgeführt wird, sobald ein OCRnA-Compara-Match-Interrupt ausgelöst wird.

### 5.7.2 Custom-Funktion

Bei der Custom-Funktion können die Farbwerte manuell definiert werden. So wird für eine bestimmte Farbe für jede LED-Farbe der entsprechende Wert in das OCRnB-Register geschrieben. Darf sich eine Farbe nicht verändern, so muss das OCRnB-Register den gleichen Wert behalten. Die Iteration durch den Rainbow-Algorithmus wird im Custom-Mode nicht benötigt.

### 5.7.3 Rainbow-Funktion

Die Farbe der RGB-LED soll nun jeweils fünf Sekunden brauchen, um einen Farbteil komplett ein- oder auszuschalten, was einen sanften Übergang im Farbkreis ermöglicht.

Im Rainbow-Loop gibt es sechs States. Zubeginn muss die grüne LED schon voll leuchten.

1. Start ==> Grün
2. Hochzählen des Rot-Anteils ==> Yellow
3. Runterzählen des Grün-Anteils ==> Rot
4. Hochzählen des Blau-Anteils ==> Magenta
5. Runterzählen des Rot-Anteils ==> Blau
6. Hochzählen des Grün-Anteils ==> Cyan
7. Runterzählen des Blau-Anteils ==> Grün
8. Repeat 2 - 7

Es benötigt 800 Schritte um einen Farbteil komplett ein- und auszuschalten. Pro Interrupt wird ein Schritt hochgezählt. Mit Formel 5.3 kann direkt berechnet werden, mit welchem Wert das Compare-Register beschrieben werden muss, sodass es fünf Sekunden geht, bis 800 Schritte hochgezählt wurden.

$$OCR_{nx} = \frac{f_{clk_{I/O}}}{2 \cdot N \cdot f_{OC_{max}}} - 1 = \frac{16MHz}{2 \cdot 1024 \cdot \frac{5s}{800Schritte}} - 1 \approx 48 \quad (5.3)$$

Die Iteration durch den Rainbow-Modus wird folglich mit einer Frequenz von 160Hz initialisiert. In der Routine wird demnach alle 6.25ms der Duty-Cycle einer Farbe hoch oder runtergezählt, und das Timer-Compare-Register des entsprechenden LED-Timers angepasst.

cite: Formel  
- Atmega  
datenblatt  
S.121

cite: Prescaler - Atmega  
datenblatt  
S.129

cite: CTC-  
Mode - Atmega  
datenblatt  
S.145

cite: CTC-  
Timing-  
Diagramm  
- Atmega  
datenblatt  
S.146

#### 5.7.4 Vorgehen

1. Benötigte Applikation aus dem Software-Ordner auf dem USB-Stick in Atmel Studio öffnen.

Software→Atmega→3\_LED\_Control→1\_LED\_Testsoftware→LED

2. Software hochladen:

AtmelStudio→Tools→Cocktailmixer

## 5.8 Motor

Der Motor wird in folgender Reihenfolge in Betrieb genommen:

1. FOC-Treiber
2. Gate-Treiber
3. H-Brücke
4. ABN-Encoder
5. PI-Regler

Da der FOC-Treiber in einem Openloop-Modus betrieben werden kann, einget 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 Position-Regelkreis beinhalten.

### 5.8.1 FOC-Treiber

Der FOC-Treiber wird über die SPI-Schnittstelle in Betrieb genommen. Dazu werden die Parameter verwendet, welche aus der TMCL-IDE verwendet werden. Die Standardparameter beinhalten Informationen zum Motor , zwei Sekunden Linksdrehung im Open-Loop, 2 Sekunden Rechtsdrehung im Open-Loop und dann Stop. Die Initialisierung sowie das Auslesen gewisser Register ist mit der Testapplikation "3\_Motor\_Openloop" möglich.

Das Setup, mit dem der Treiber softwareseitig in Betrieb genommen wurde, ist im Anhang Kapitel E.3.1 gezeigt.

Kurz beschreiben, was die Register beinhalten

Vorgehen:

1. Benötigte Applikation aus dem Software-Ordner auf dem USB-Stick 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→Cocktailmixer](#)

4. SPI-Kommunikation und ausgehende Gate-Signale vom FOC-Treiber zum Gate-Treiber mit Oszilloskop überprüfen. Im Anhang Kapitel E.3.2 sind die Messbilder zur SPI-Kommunikation zu finden und in Anhang E.3.3 die Messbilder zur Gate-Control vom TMC4671 zum TMC6200.

### 5.8.2 Gate-Treiber

Der Gate-Treiber wird ebenfalls über die SPI-Schnittstelle in Betrieb genommen. Dazu werden die Parameter verwendet, welche in der TMCL-IDE verwendet wurden. Mit den Standardparametern muss am Motoranschluss die selbe Signalfolge anliegen wie am sie auch am FOC-Treiber anliegen (Verstärkt durch H-Brücke). Die Initialisierung sowie das Auslesen gewisser Register ist mit der Testapplikation "3\_Motor\_Openloop" möglich.

Das Setup, welches in Kapitel 5.8.1 erwähnt wurde, ist jetzt um einen Baustein erweitert worden und in Anhang Kapitel E.3.1 ersichtlich.

Vorgehen:

1. Benötigte Applikation aus dem Software-Ordner auf dem USB-Stick 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→Cocktailmixer](#)

4. SPI-Kommunikation und Signale an der H-Brücke überprüfen. Im Anhang Kapitel F.4.5 sind die Messbilder zur SPI-Kommunikation zu finden und in Anhang F.4.6 die Bilder zur Gate-Ctrl vom TMC6200 zur H-Brücke.

### 5.8.3 BLDC und H-Brücke

Die H-Brücke wird mit dem selben Setup und der selben Software wie in Kapitel 5.8.2 gemacht. Es komm. Der Vorgang ist auch derselbe, weshalb dieser weggelassen wird. Bei der H-Brücke geht es darum, die Schaltsignale der FETs zu überprüfen, bevor der Motor angeschlossen wird. Sind die Signale gut, kann der Motor angeschlossen werden.

Das Setup mit dem Motor ist im Anhang Kapitel G.2.1 und die Schaltsignale im Anhang Kapitel G.2.2 ersichtlich.

Wird jetzt der Motor angeschlossen, dreht sich dieser mit der vorgegebenen Geschwindigkeit.

### 5.8.4 ABN-Encoder

Der ABN-Encoder wird an den FOC-Treiber angeschlossen. Daher wird die Schnittstelle am TMC4671 initialisiert. Dazu werden die Parameter verwendet, welche in der TMCL-IDE verwendet wurden. Das Einlesen wird getestet, indem der Treiber in den Positionsmodus versetzt wird, wozu das Feedback des Rotors nötig ist. Hat die Initialisierung funktioniert, dreht sich der Motor an eine gewünschte Position und hält dort. Zur Initialisierung gehören auch einige Einstellungen zu den PI-Reglern. Die Initialisierung sowie das Auslesen gewisser Register ist mit der Tesapplikation "4\_Motor\_ABN\_Encoder" möglich.

Das bisherige Setup hat sich nun um den ABN-Encoder erweitert und ist im Anhang Kapitel H.2.1 ersichtlich.

Vorgehen:

1. Benötigte Applikation aus dem Software-Ordner auf dem USB-Stick in Atmel Studio öffnen.

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

2. Software hochladen:

[AtmelStudio→Tools→Cocktailmixer](#)

3. Ausgehende Signale des ABN-Encoders überprüfen. Im Anhang Kapitel H.2.2 ist das Messbild des ABN-Encoder-Ausgangs zu sehen.

### 5.8.5 PI-Regler

Die Regler, welche im FOC-Treiber integriert sind, werden verwendet, um einen Motor an eine bestimmte Position zu fahren, mit einer bestimmten Geschwindigkeit drehen oder ein bestimmtes Drehmoment anzulegen. Im Gegensatz zu einem Motor, dem nur eine Spannung vorgegeben wird, kann bei einer Abweichung des vorgegebenen Parameters nachgeregelt werden. So kann beispielsweise bei grösserer Belastung die angelegte Spannung erhöht werden, sodass die vorgegebene Geschwindigkeit gehalten werden kann.

cite:  
<https://edoc.hamburg.de/>

Für die Regelung eines Motors können verschiedene aufgebaut sein. Die Regelstruktur, wie sie im FOC-Treiber TMC4671 integriert ist, ist eine Kaskadenregelung. 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 den 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.

Die Regelung im TMC4671 besteht aus einem innersten, schnellen Stromregelkreis. Dieser soll in der Lage sein, schnellst möglich mit einer Erhöhung des Drehmomentes auf eine Abweichung zu reagieren. Ausserdem gibt der Stromregelkreis die Kommutierung des Motors vor.

Der Geschwindigkeitsregelkreis ist dem Stromregelkreis überlagert. Weicht die aktuelle Geschwindigkeit von der vorgegebenen Geschwindigkeit ab, so gibt der Geschwindigkeitsregelkreis dem Stromregelkreis eine Abweichung vor, wodurch der Strom durch die Spulen und somit das Drehmoment erhöht oder gesenkt wird.

Der Positionsregelkreis ist dem Geschwindigkeitskreis überlagert. Weicht die aktuelle Position von der vorgegebenen Position ab, so gibt der Positionsregelkreis dem Geschwindigkeitsregelkreis eine Abweichung vor, wodurch der Geschwindigkeitsregelkreis in die erforderliche Richtung korrigiert.

Mit der Begrenzung der Eingangsgrößen 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.

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

Regelkreis	P-Anteil	I-Anteil	Nachstellzeit
Drehmoment	100	1200	
Fluss	100	1200	
Geschwindigkeit	2000	300	
Position	100	0	

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 überzuschissen. Dies betrifft den Regler für den Stromregelkreis und den Regler für den Geschwindigkeitsregelkreis. Die zugehörigen Parameter wurden in der TMCL-IDE ermittelt.

Mit diesen Werte ist es nur schwierig realisierbar, den Schlitten mit einer kontrollierten Bewegung fortzubewegen. Um dennoch gezielt Positionen anzufahren, wurde eine Software-Ramp geschrieben. Diese berechnet den gewünschten Weg unter Berücksichtigung der maximalen Beschleunigung und der maximalen Geschwindigkeit. Durch eine periodische Iteration in Millisekunden-Schritten 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 skallierbar, was eine schnellere Einbettung in das Gesamtsystem ermöglicht.

## 6 Software

Im Folgenden Kapitel werden die beiden Softwareteile beschrieben. Die Software für den Mikrocontroller beinhaltet die komplette Ansteuerung der Teilsysteme, die Führung durch das Menu 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 ESP32 erweitert die Funktionalität der Maschine, indem einige Funktionen auch über eine Android-Applikation auf dem Handy aufgerufen werden können. Die Software auf dem ESP32 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 wird in das ESP32 implementiert.

### 6.1 Atmega2560

Die Software für den Atmega2560 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>4</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>5</sup> verglichen werden. Es wird nur über diese Funktionen auf die entsprechende Hardware zugegriffen.

cite:  
<https://www.insider.de/was-ist-ein-application-programming-interface-api-a-735797/>

---

<sup>4</sup>API, Schnittstelle zur Anwendungsprogrammierung

<sup>5</sup>HAL, Hardwareabstraktionsschicht

### 6.1.1 Strukturplan

In Abbildung 6.1 wird aufgezeichnet, wie die Software aufgebaut ist und welcher Teil der Software von welchen Libraries abhängig ist. Wie in Kapitel ?? beschrieben kann die Software in folgende Teile gegliedert werden:

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

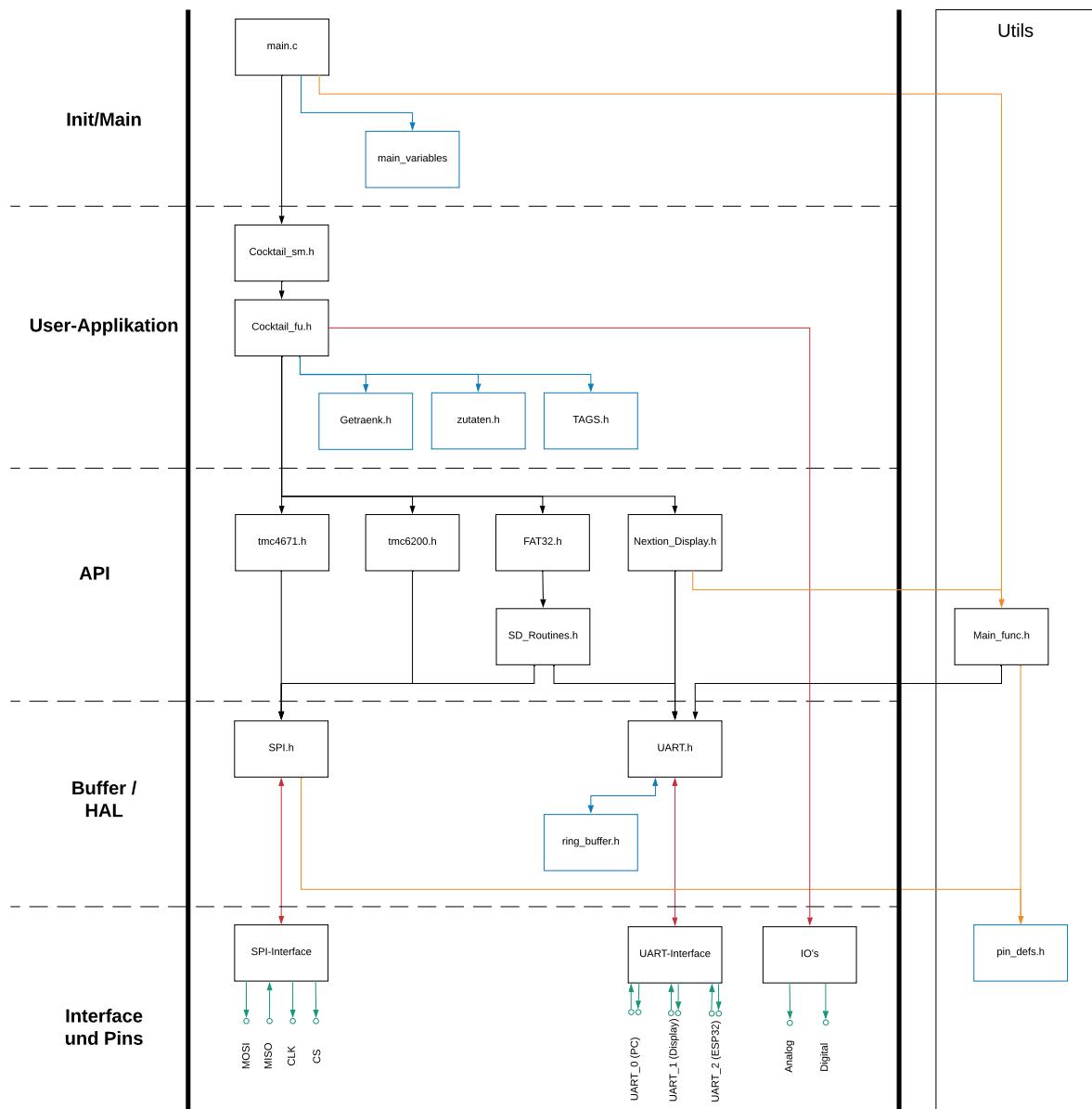


Abbildung 6.1: Softwareübersicht Atmega2560.

### 6.1.2 Programmflussdiagramm

**Bootloader-Section** In Abbildung 6.2 wird der Programmfluss der Atmega2560-Software aufgezeigt. Bei 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 cocktailspezifische 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 6.1 in Kapitel 6.1.1) gebraucht werden. Die Struktur und Anwendung der Speicherplätze wird in Kapitel ?? erklärt. 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.

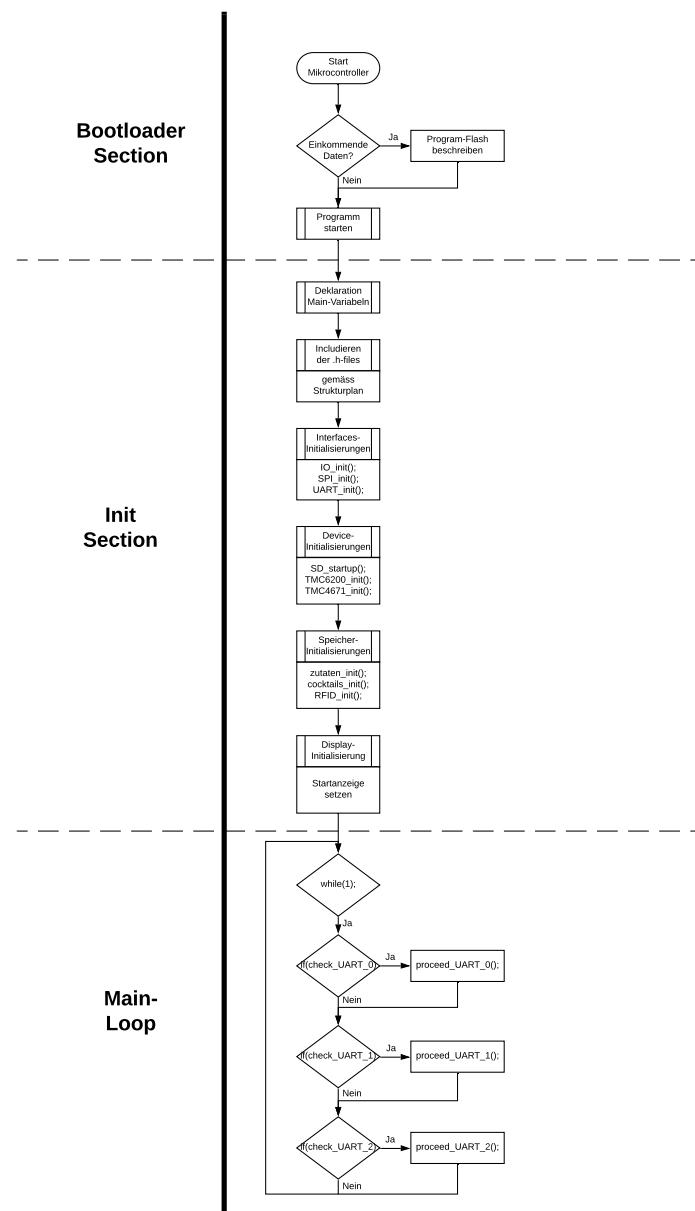


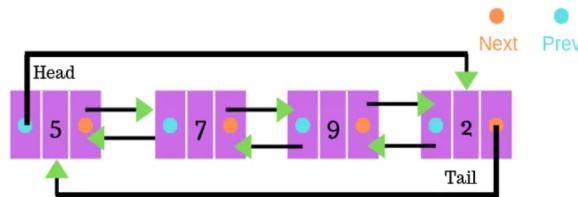
Abbildung 6.2: Programmflussdiagramm Atmega2560.

### 6.1.3 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>6</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 Zeigern auf die umliegenden Elemente enthält das Element die gewünschten Daten. Um zu wissen, ob das Ende oder der Beginn erreicht wurde, werden zwei zusätzliche Zeiger initialisiert, welche auf das erste und/oder letzte Element zeigen (Head und Tail). Im Anhang Kapitel D.2.1 können weitere Details zu Dynamic Linked Lists entnommen werden.

Abbildung 6.3 zeigt eine Struktur der beschriebenen Liste mit next, prev, head, tail und Elementen.

cite Text:  
<https://perlguru.de/verkettete-listen>



**Abbildung 6.3:** Doppelt verkettete Liste mit zwei Elementen.

Die Elemente sind in Structs organisiert, welche die Zeiger 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 vier Elemente Listen angelegt:

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

cite Bild:  
<https://learncode.academy/circulating-doubly-linked-list-in-javascript/>

Da während der Entwicklung der Programmspeicher mit zu vielen Daten gefüllt wurde, stürzte der Mikrocontroller ab. Die Cocktail- und die Zutaten-Liste beinhalten deshalb nur die Nummer eines zugehörigen 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 verschwinden wieder, sobald ein neues Element geladen wird.

Die Tag-Liste sowie die Liste der Zutaten in der Maschine werden komplett in den Programmspeicher geladen. Die beiden Listen werden bei einer Veränderung als Backup jeweils in einem File gespeichert, damit diese bei einem Neustart der Maschine wieder in den Programmspeicher geladen werden können.

## 6.2 ESP32

Bei der Inbetriebnahme in Kapitel 4.5.2 wurde das ESP32 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 ESP32 gesendet werden können. Diese

<sup>6</sup>Nodes = Knoten

kommuniziert über Bluetooth und hat den Vorteil, dass eine App einiges Benutzerfreundlicher ist, da App's im Alltag fast jeder Person integriert sind.

Die Software für das ESP32 wurde komplett in Arduino IDE geschrieben und beinhaltet folgende Bereiche:

- 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

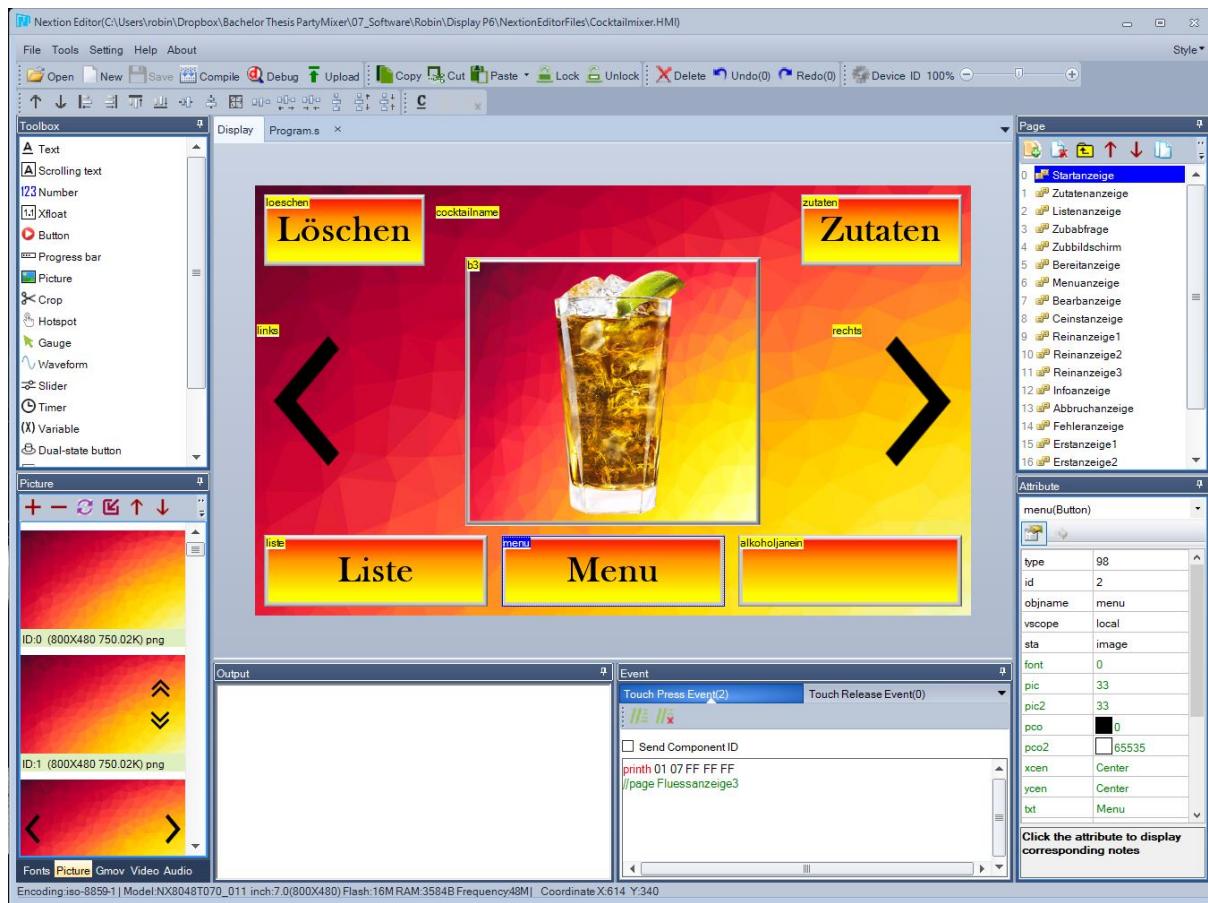
Dabei beinhaltet die «Arduino.h» Bibliothek die Standardbefehle von Arduino, die «BluetoothSerial.h» Bibliothek beinhaltet alle Kommunikationsbefehle um über Bluetooth kommunizieren zu können, die «SPI.h» Bibliothek beinhaltet alle Kommunikationsbefehle um über SPI kommunizieren zu können und die «MFRC522.h» 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.

### 6.2.1 Programmflussdiagramm

## 6.3 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 Funktion. Die Softwareoberfläche ist in Abbildung 6.4 zu sehen. Der Aufbau ist relativ 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.



**Abbildung 6.4:** Programmieroberfläche von Nextion

### 6.3.1 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 Getränkes 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 6.5 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 Getränename 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 was sich darin befindet und wie viel, 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 6.5: Hauptansicht der Menüstruktur des Display's

### 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 versüßt. 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 und das Display zeigt dies an. In Abbildung 6.6 sind die dazugehörigen Bildschirme zu sehen.

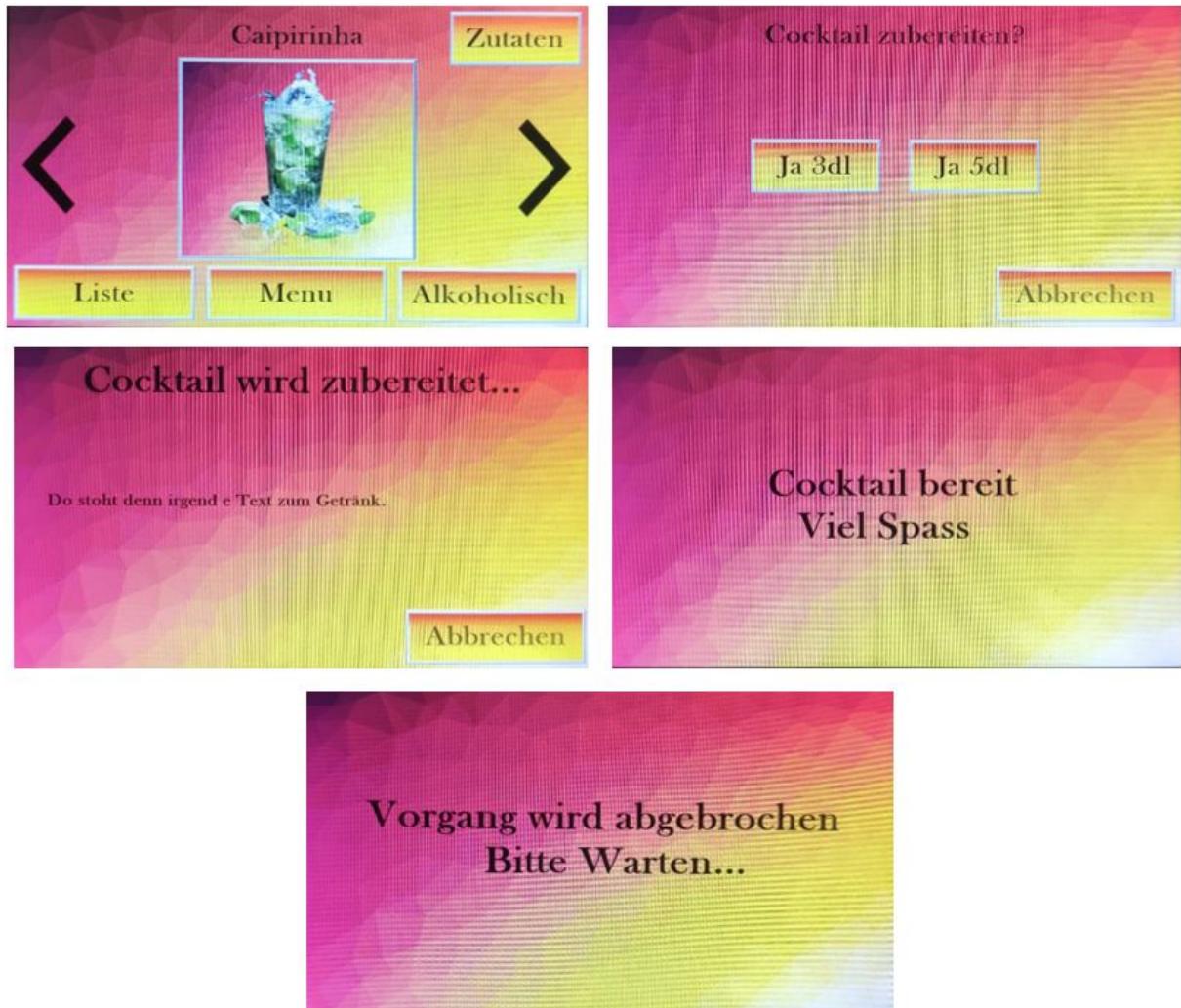


Abbildung 6.6: Auslösevorgang der Menüstruktur des Display's

### 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 standartmäßig 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 6.7 zu sehen.

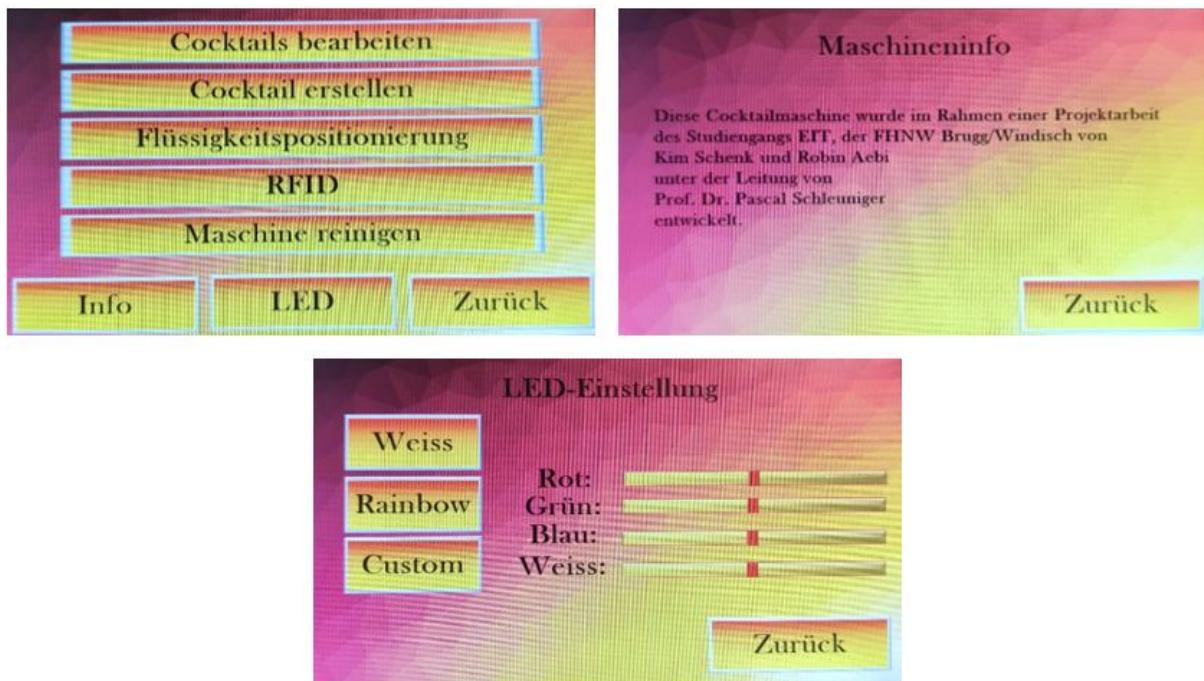


Abbildung 6.7: Infomenü und LED Einstellungen der Menüstruktur des Display's

### Cocktails bearbeiten

Um dem Kunden Flexibilität gewährleisten zu können, ist es möglich gemäß Abbildung 6.8 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 6.5. Hier kann entweder das zu bearbeitende Getränk ausgewählt werden oder mittels «Standardeinst.» alle Getränke auf den Ursprungszustand gesetzt werden. 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 6.8:** 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 6.9. 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 werden 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 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.

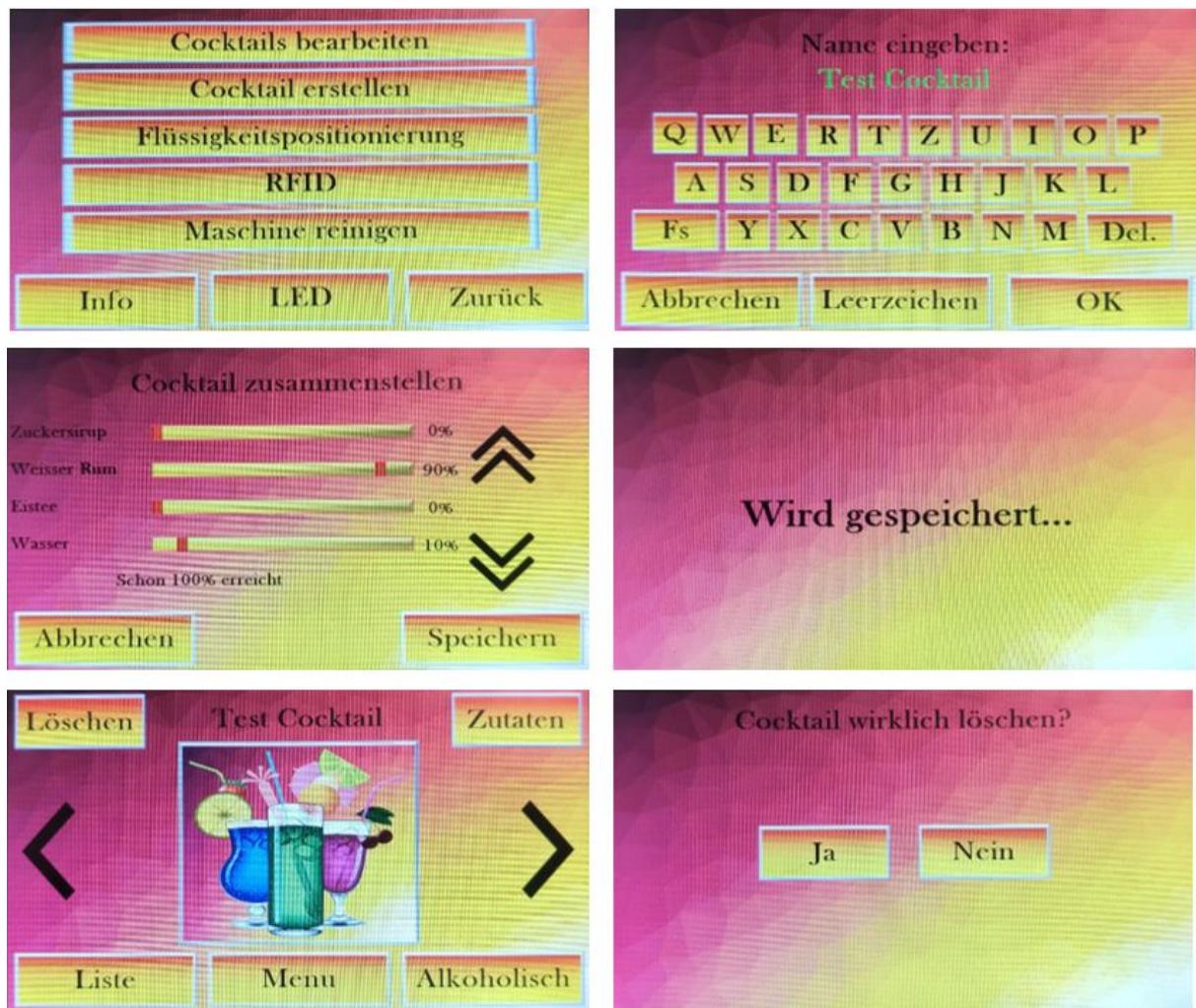


Abbildung 6.9: Cocktail Erstellungsmenü der Menüstruktur des Display's

### 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 6.10 zu sehen.



Abbildung 6.10: 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 6.11 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. 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 bleiben gespeichert, werden jedoch ausgeblendet. Das Selbe geschieht auch, wenn man beim Grund-

gerüst auf «Zurück» drückt. Die Änderungen werden übernommen und eine neue Cocktailliste wird erstellt.



**Abbildung 6.11:** 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. Wir 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 6.12 zu sehen.



Abbildung 6.12: 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 6.13.



Abbildung 6.13: Reinigungsmodus der Menüstruktur des Display's

## 6.4 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 Programmierversuchen stellte sich jedoch heraus, dass AppInventor trotz einiger schwächen flüssiger und zuverlässiger läuft als seine Konkurrenz. Die beiden grössten Schwächen von AppInventor sind es, dass mit dem Tool keine Dynamischen Buttons erstellt werden können, was es schwierig gestaltet Buttonlists oder ähnliches zu erstellen. Als weiterer grosser Schwachpunkt gilt es, das 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 Listen 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.

---

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 6.14 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 reingezogenen Komponenten als Übersicht und ganz rechts die Einstellungen und Formatierungsmöglichkeiten zu den gewählten Komponenten.

Quelle:<http://>

Quelle:<https://>

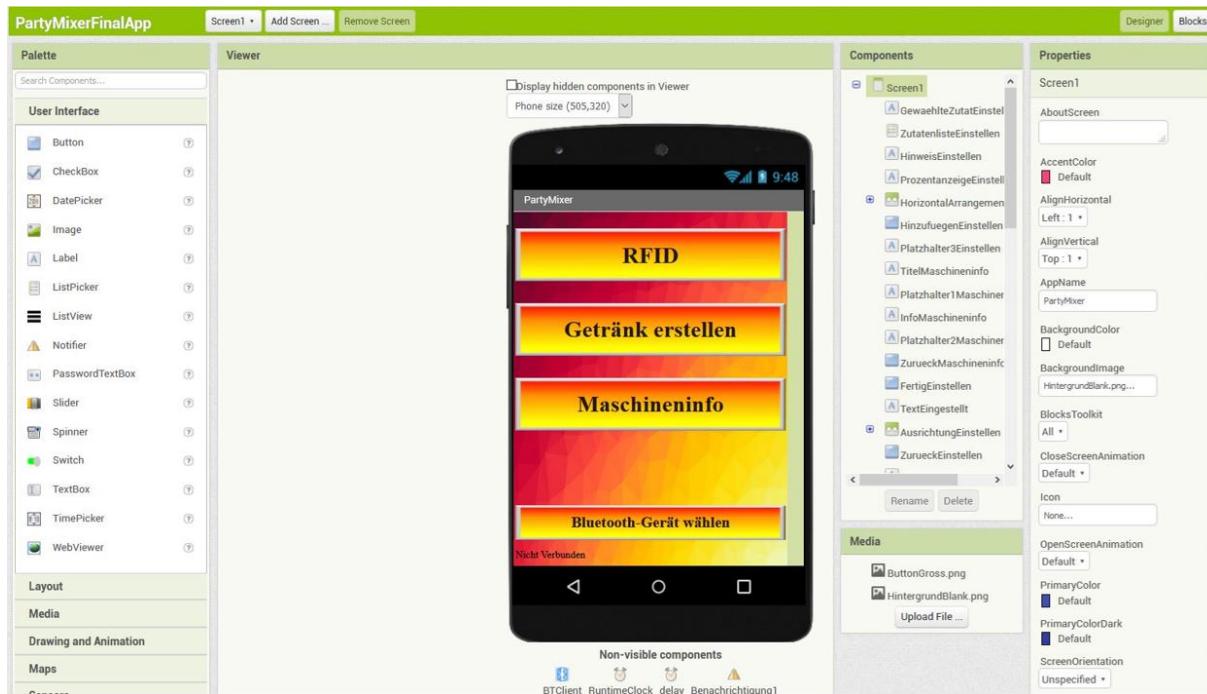


Abbildung 6.14: Designer von AppInventor

In den Blocks gemäss Abbildung 6.15 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 gemanagt werden müssen.

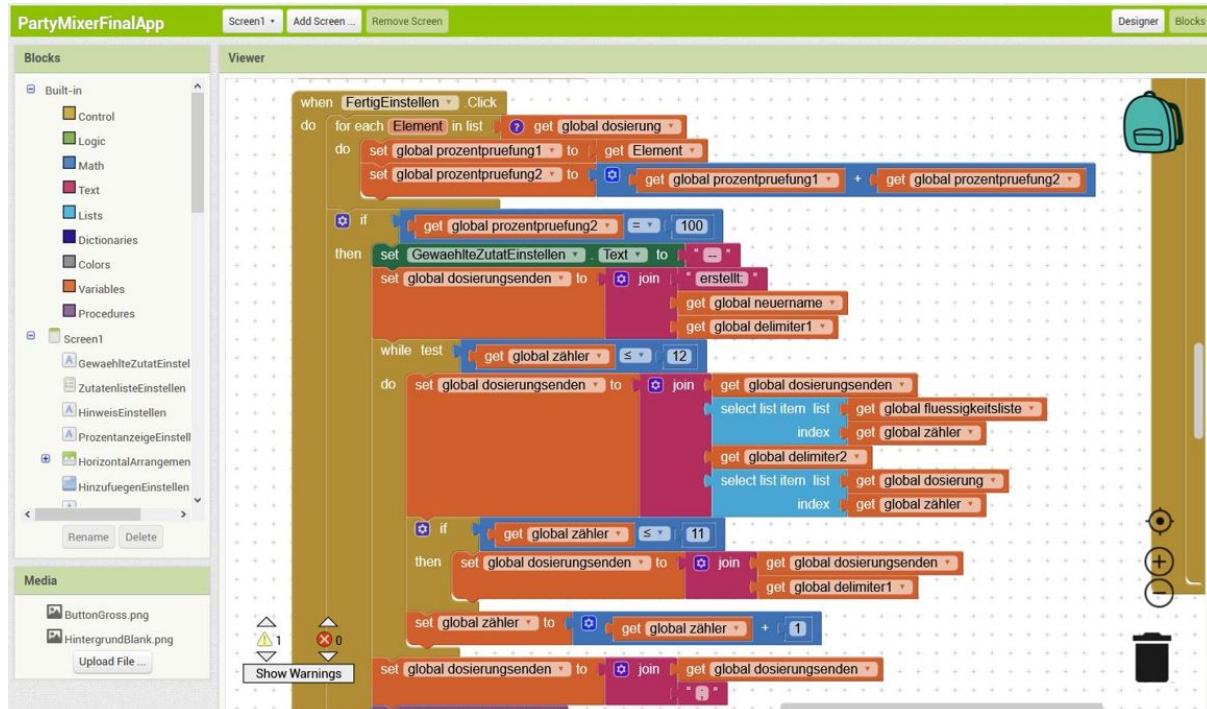


Abbildung 6.15: Blocks von AppInventor

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.

Quelle:  
<https://appinventor.mit.edu/app-inventor-ios-version-0-9>

#### 6.4.1 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 dabei 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 Hauptmenu gehen. Die Menubildschirme sind in Abbildung 6.16 zu sehen.

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 Hauptmenu. 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

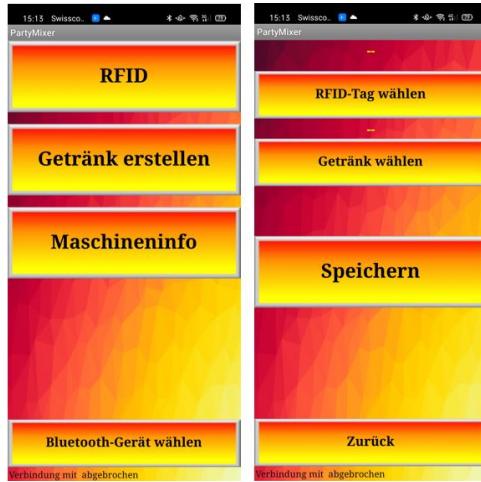


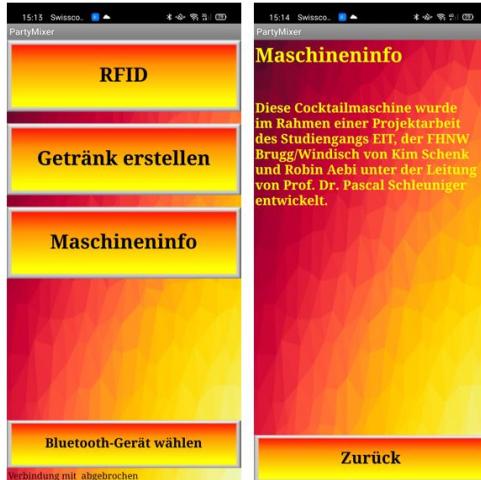
Abbildung 6.16: RFID Einstellungen der App

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 Kick 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 Menubildschirme sind hierbei in Abbildung 6.17 zu sehen.



Abbildung 6.17: 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 6.18 zu sehen.



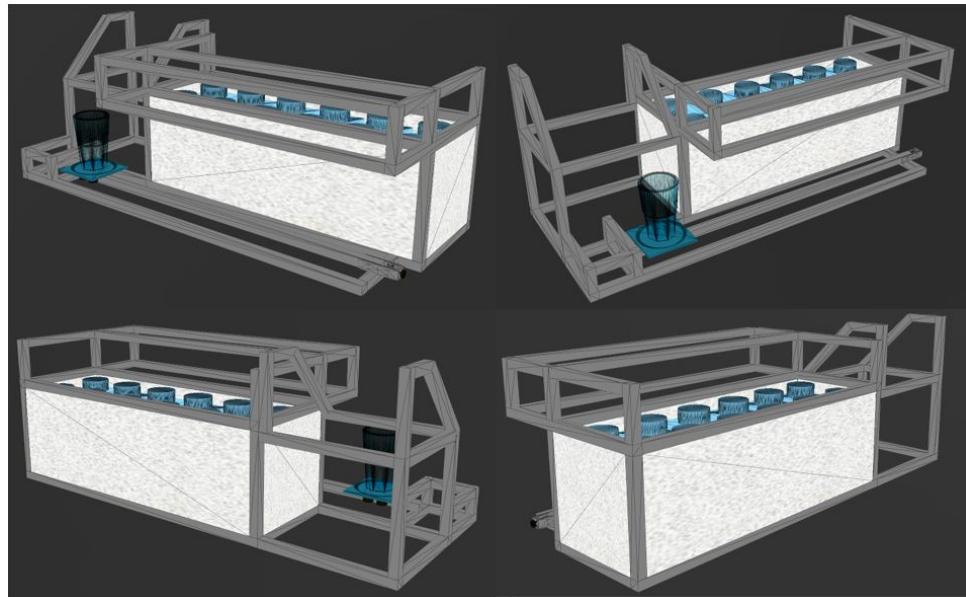
**Abbildung 6.18:** Infoanzeige der App

## 7 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

Es wurde dabei vor der Bauphase ein Dreidimensionales Modell mittels des Windows Programms «3D Builder» gezeichnet, welches in Abbildung 7.1 zu sehen ist. Dieses Modell Basiert auf den evaluierten Bauteilen, auf welche in den nächsten Kapitel eingegangen wird.



**Abbildung 7.1:** 3D-Modell des Partymixers

## 7.1 Rahmen

Der Rahmen besteht aus 20x20mm Aluminium X-Profilen. Diese haben einerseits den Vorteil, dass sie sehr stabil sind und andererseits sind diese vielseitig einsetzbar. 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 7.2 sind diese Bauteile zu sehen.



**Abbildung 7.2:** Bauteile für den Rahmen des Partymixers

## 7.2 Getränkeschlitten

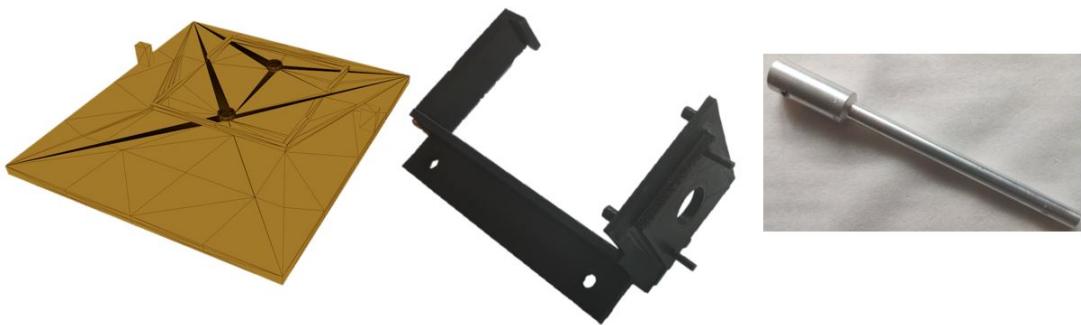
Um ein Getränk möglichst flüssig und stabil von einem zum anderen Punkt befördern zu können wurde auf ein Schlittensystem zurückgegriffen, welches so ähnlich auch im 3D-Druck Bereich eingesetzt wird. Dieses besteht ebenfalls aus einem X-Profil gemäß 7.1, einem Schlitten, einem Riemen, einem Riemenspanner und einem Zahnrad.



**Abbildung 7.3:** Bauteile für den Schlitten des Partymixers

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-Profiles 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. Damit das Glas jedoch trotzdem mitig unter den Pumpen geführt werden kann, wurde eine Glassauflage im 3D-Drucker gedruckt, welche auch den Riemen sauber im Profil führt. Diese Teile sind in Abbildung 7.4 zu sehen.



**Abbildung 7.4:** Motorhalterung des Partymixers

### 7.3 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.

bessere Bilder der Motorhalterung machen und der Ablage montiert

## 7.4 Unterbringung der Elektronik

Der grösste Teil der Elektronik befindet sich auf der linken Seite der Maschine unterhalb des Displays. Darin ist einerseits der Motor untergebracht und anderseits das Netzteil, die RFID-Antenne und das Herz der Maschine - die Steuerplatine. In Abbildung XY ist dies zu sehen.

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 7.5.

Bild der unterbringung der Elektronik einfügen



**Abbildung 7.5:** Montage der Pumpen und Durchflussmessgeräte des Partymixers

## 7.5 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 zusätzlich Platz für Kuhlakkus geschaffen, welche im Gefrierfach gefroren und dann bei Gebrauch zwischen den Flüssigkeiten platziert werden können. Dieser Aufbau ist in Abbildung 7.6 zu sehen.



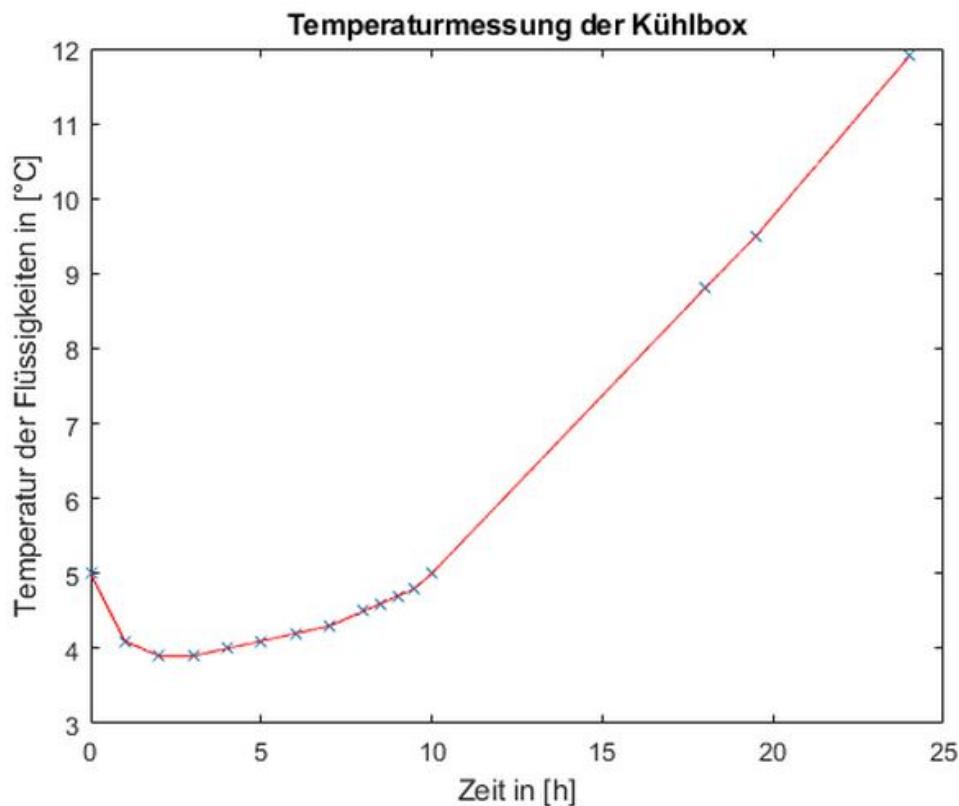
Abbildung 7.6: Kühlbox des Partymixers

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:

- 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 7.7 ist das Ergebnis zu sehen.



**Abbildung 7.7:** 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.

Flaschen  
beschreiben  
und Bild  
dazufügen

## 8 Evaluation

## 9 Fazit

### 9.1 Zielerreichung

### 9.2 Kosten

## 10 Schlusswort

## 11 Ehrlichkeitserklärung

Mit der Unterschrift bestätigt der Unterzeichnende Projektleiter, 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:

---

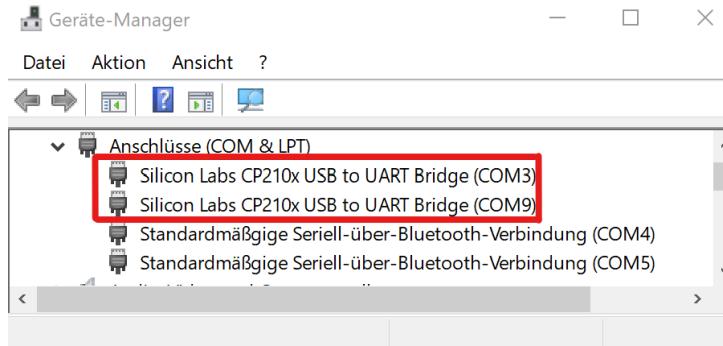
## Literatur

- [1] JLCPBC. (10. Sep. 2020). „JLCPCB Capabilities“, Adresse: <https://jlcpcb.com/capabilities/Capabilities> (besucht am 10. Sep. 2020).
- [2] 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).
- [3] ——, (6. Feb. 2019). „TMC4671 Datasheet V1.06“, Adresse: [https://www.trinamic.com/fileadmin/assets/Products/ICs\\_Documents/TMC4671\\_datasheet\\_v1.06.pdf](https://www.trinamic.com/fileadmin/assets/Products/ICs_Documents/TMC4671_datasheet_v1.06.pdf) (besucht am 11. Sep. 2020).
- [4] ——, (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).
- [5] ——, (18. Okt. 2017). „TMC-UPS-xAxV-EVAL“, Trinamic, Adresse: <https://www.trinamic.com/support/eval-kits/details/tmc-ups-xaxv-eval/> (besucht am 11. Sep. 2020).
- [6] 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).
- [7] 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).
- [8] V. H. Zenkner, „Das USB Interface aus EMV Sicht“, S. 15, 28. Juli 2014. Adresse: <https://www.all-electronics.de/wp-content/uploads/2015/11/ANP024-DE-Das-USB-Interface-aus-EMV-Sicht-35k-37images.pdf> (besucht am 11. Sep. 2020).
- [9] 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).
- [10] 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).
- [11] RFID-basis.de. (15. Sep. 2018). „Aufbau und Funktionsweise von RFID-Systemen - RFID Technik“, Adresse: <https://www.rfid-basis.de/rfid-technik.html> (besucht am 11. Sep. 2020).
- [12] 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>.
- [13] 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).
- [14] 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).
- [15] ionos.de. (15. Apr. 2020). „FAT32“, IONOS Digitalguide, Adresse: <https://www.ionos.de/digitalguide/server/knowhow/fat32/> (besucht am 11. Sep. 2020).
- [16] E. Milsch. (Feb. 2009). „Aufbau des FAT32 Dateisystems“, Adresse: <https://docplayer.org/78333139-Aufbau-des-fat32-dateisystems.html> (besucht am 11. Sep. 2020).
- [17] I. Grokhov, K. Sovani, M. Jain, M. N. Dev, A. Gratton und J. Domburg, [espressif/arduino-esp32](https://github.com/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).

- [18] 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).
- [19] 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).
- [20] 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).
- [21] C. C. Dharmani. (31. Jan. 2009). „SD/SDHC Card Interfacing with ATmega8 /32 (FAT32 implementation)“, Adresse: <https://www.dharmanitech.com/2009/01/sd-card-interfacing-with-atmega8-fat32.html> (besucht am 12. Sep. 2020).
- [22] T. Hammer. (). „HTerm“, der-hammer, Adresse: <http://www.der-hammer.info/pages/terminal.html> (besucht am 12. Sep. 2020).
- [23] B. Kleitz, sec 11 4 Switch Debouncing, 2011. Adresse: <https://www.youtube.com/watch?v=uQTPZoDbfUs> (besucht am 11. Sep. 2020).

## A Programmierschnittstellen

### A.1 Geräte-Manager



**Abbildung A.1:** Geräte-Manager mit den aufgelisteten USB-UART-Converter (Mikrocontroller und WiFi-Modul).

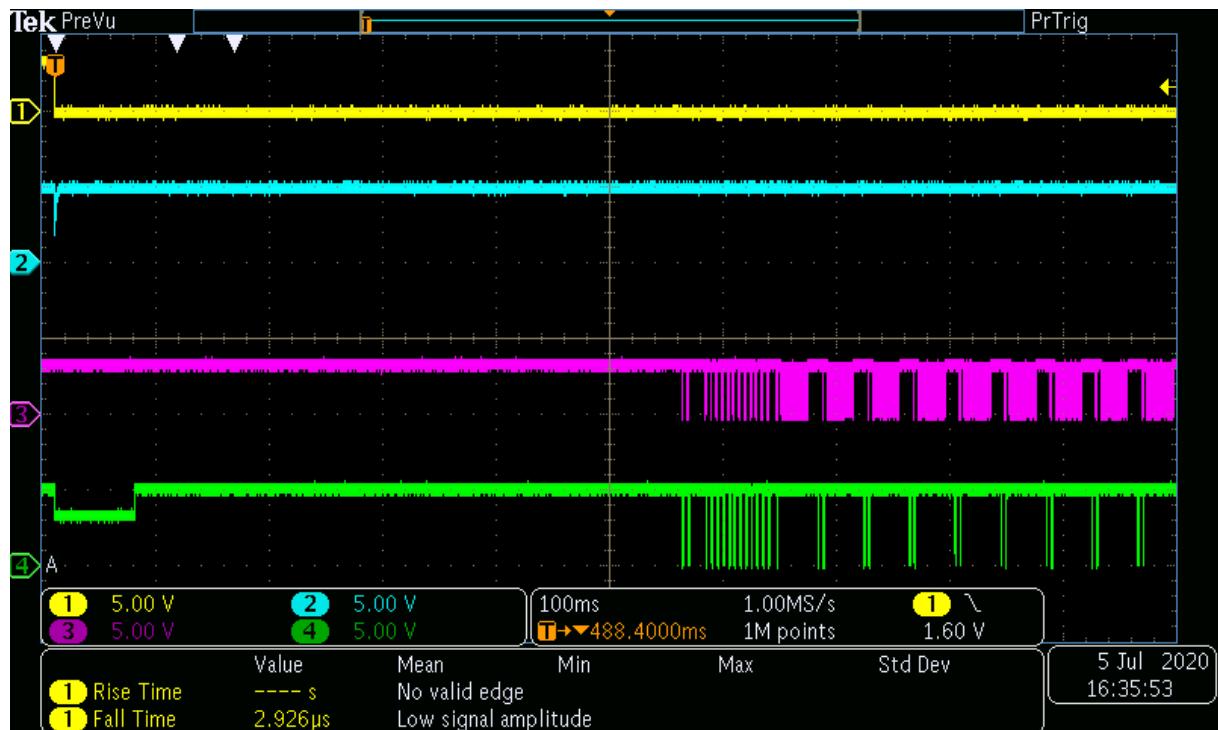
### A.2 Atmega2560

#### A.2.1 Wiring

Mikrocontroller				USB-Flash-Device
RX	<==	direkt	====	TX
TX	====	direkt	==>	RX
Reset	<==	Kondensator	====	DTR

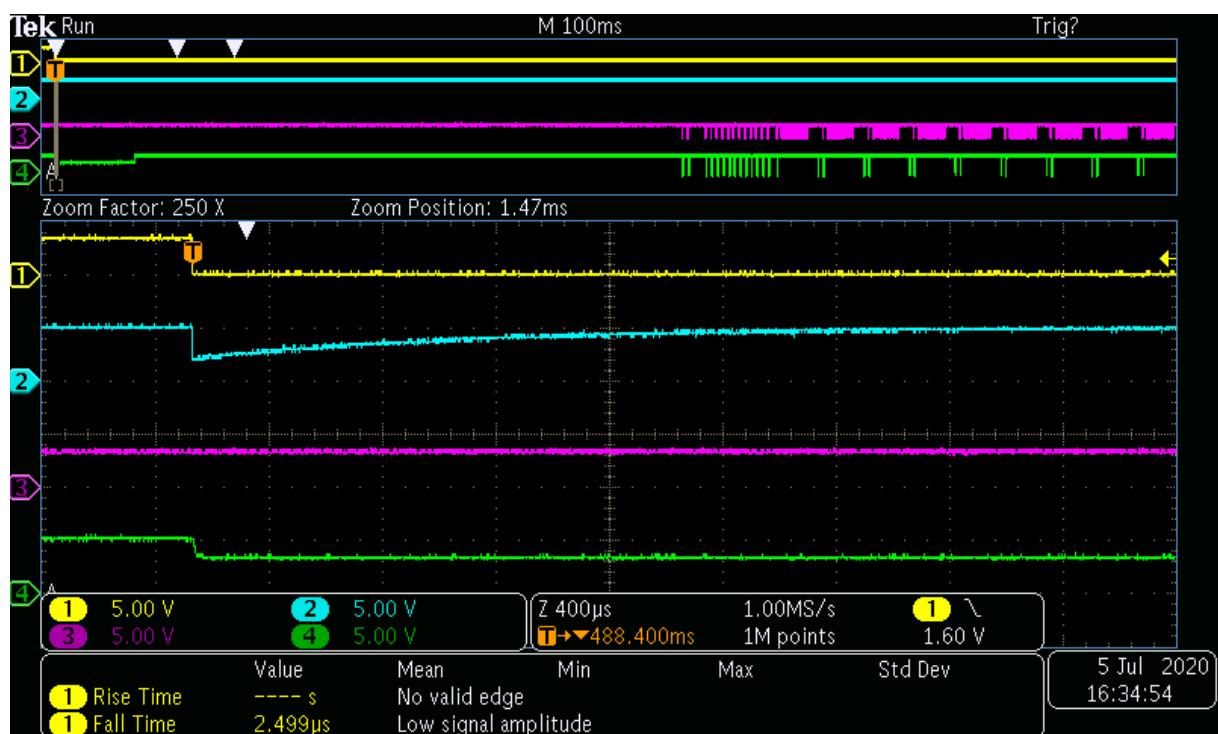
**Tabelle A.1:** Verbindung zwischen USB und Mikrocontroller.

### A.2.2 Handshake



**Abbildung A.2:** Hochladen des Programmcodes auf den ATMega2560.

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



**Abbildung A.3:** Handshake zum Hochladen des Programmcodes auf den ATMega2560. Zoom auf den Moment des Resets.

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

## A.3 ESP32

### A.3.1 Wiring

ESP			USB-Flash-Device
RX	<==	direkt	==== TX
TX	====	direkt	==> RX
EN	<==	über Transistor	==== RTS
IO_0	<==	über Transistor	==== DTR
IO_13	<==	über Widerstand	==== RTS
IO_15	<==	über Widerstand	==== CTS

Tabelle A.2: Verbindung zwischen USB und ESP.

### A.3.2 Handshake

Im Getting Started Guide vom ESP32 steht geschrieben, dass wenn während die beiden Pins IO0 und EN GND gezogen werden, das Modul in den Download-Modus versetzt wird, die Firmware über den UART-Port herunterlädt und diese in den Flash-Speicher schreibt. Um dies über die DTR- und RTS-Leitung machen zu können, braucht es eine zusätzliche Programmierlogik.

Mit der Schaltung, wie sie in Abbildung 4.15 ersichtlich ist, kann das ESP32-Modul mit einer bestimmten Signalabfolge an den Leitungen DTR und RTS in diesen Download-Boot-Modus gesetzt werden. Die Auswirkung der Schaltung auf IO0 und den EN-Pin kann in Tabelle A.3 entnommen werden.

Auto Program Circuit			
DTR	RTS	EN	IO0
1	1	1	1
0	0	1	1
1	0	0	1
0	1	1	0

Tabelle A.3: Strapping pins des ESP32 und deren Auswirkung auf den Boot- und Download-Modus beim aufstarten.

Die Programmierlogik wird aus dem ProgrammierTool angesteuert. Es befindet sich in der ESP-Bibliothek, welche verwendet wird, um Code aus der Arduino-Umgebung hochzuladen. Aus dem Python-Skript kann entnommen werden, wie die Pins angesteuert werden. Der Code ist in Abbildung A.4 dargestellt.

Bei der Interpretation des Codes ist darauf zu achten, dass die beiden Pins DTR und RTS Active-Low sind. (True = 0V = 0 = LOW und False = VCC = 1 = HIGH). Der Bezug zwischen dem Upload des Programms, den Pins des ESP32 und dessen Download-Boot-Modus sowie den Leitungen DTR und RST wird in Tabelle A.4 aufgezeigt.

Referenz  
einfügen:  
<http://loboris.com>

```

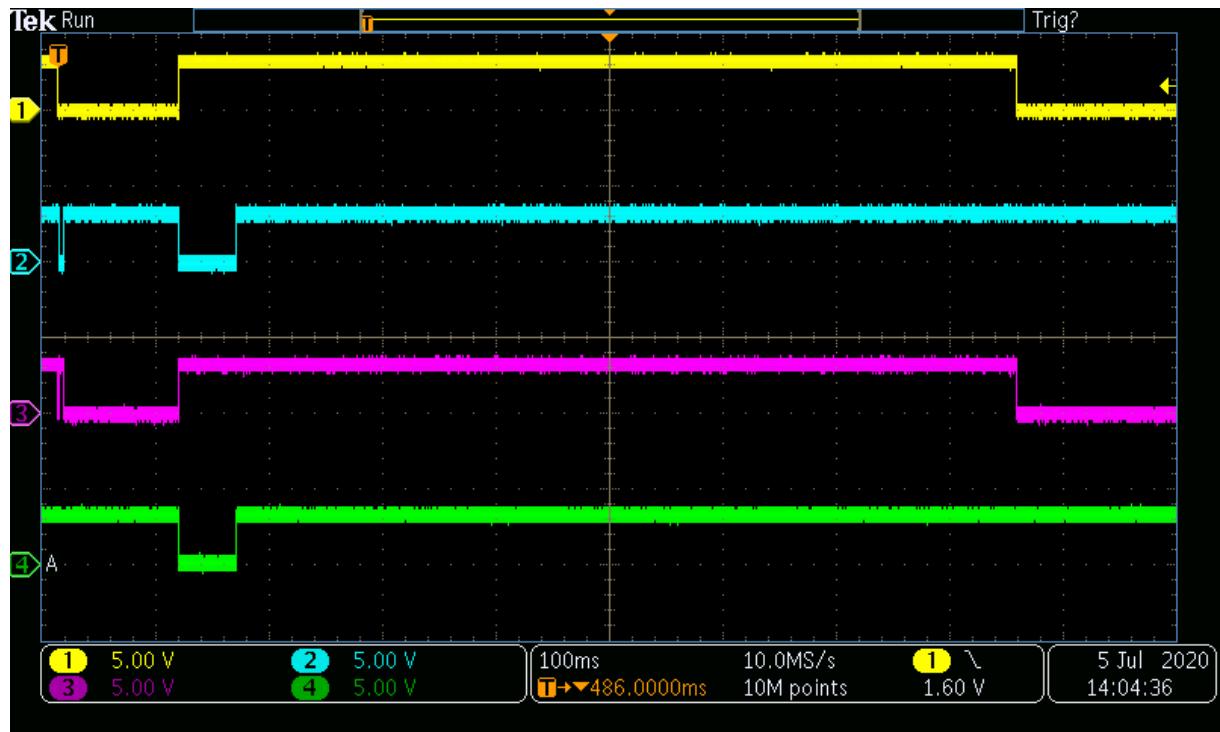
# issue reset-to-bootloader:
# RTS = either CH_PD/EN or nRESET (both active low = chip in reset
# DTR = GPIO0 (active low = boot to flasher)
#
# DTR & RTS are active low signals,
# ie True = pin @ 0V, False = pin @ VCC.
if mode != 'no_reset':
    self._setDTR(False) # IO0=HIGH
    self._setRTS(True) # EN=LOW, chip in reset
    time.sleep(0.1)
    if esp32r0_delay:
        # Some chips are more likely to trigger the esp32r0
        # watchdog reset silicon bug if they're held with EN=LOW
        # for a longer period
        time.sleep(1.2)
    self._setDTR(True) # IO0=LOW
    self._setRTS(False) # EN=HIGH, chip out of reset
    if esp32r0_delay:
        # Sleep longer after reset.
        # This workaround only works on revision 0 ESP32 chips,
        # it exploits a silicon bug spurious watchdog reset.
        time.sleep(0.4) # allow watchdog reset to occur
    time.sleep(0.05)
    self._setDTR(False) # IO0=HIGH, done

```

**Abbildung A.4:** Codeausschnitt aus dem Programmier-Tool. Auf diese Weise werden der DTR- und der RTS-Pin angesteuert während dem Programmervorgang.

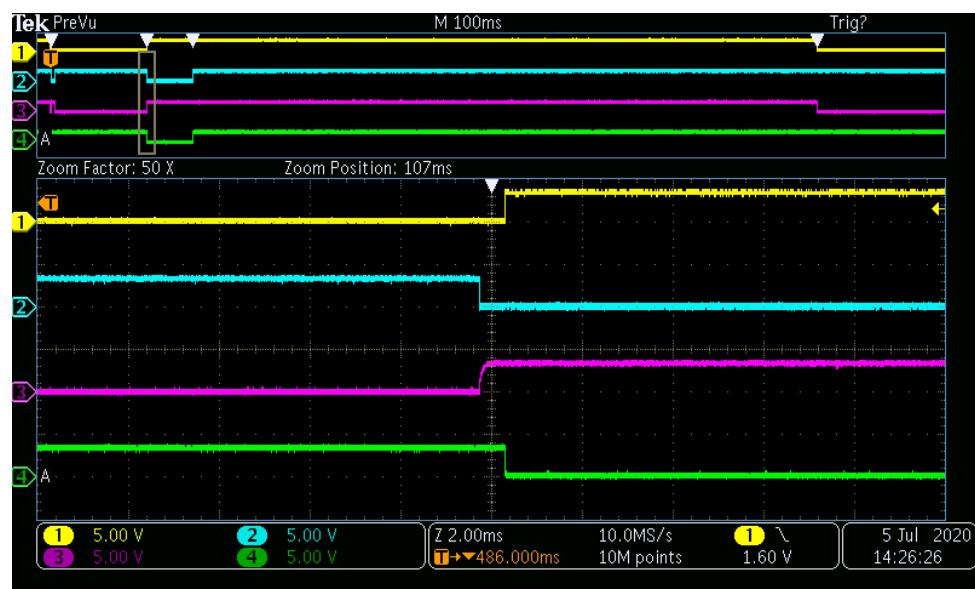
Schritt	Beschreibung	DTR	RTS	EN	IO0
1	Im ersten Schritt wird der Chip mit $EN = 0$ und $IO0 = 1$ in den Reset-Modus geschaltet und im Programm 0.1ms gewartet.	1	0	0	1
2	Im zweiten Schritt werden die Zustände geändert, jetzt ist $EN = 1$ und $IO0 = 0$ . Aufgrund der Kapazität am EN-Pin wird die Spannung für einen kurzen Moment tief gehalten. Dies hat Schritt 3 zur Folge.	0	1	1	0
3	Die durch den Kondensator C7 tief gehaltene Spannung bewirkt, dass sich die Zustände über den Pins wie folgt verhalten: $EN = 0$ und $IO0 = 0$ . Im Programm wird 0.05ms gewartet. Danach ist das ESP32 im Download-Boot-Modus und das zu speichernde Programm kann hochgeladen werden.	0	1	0	0
4	Nachdem das Programm hochgeladen wurde, werden beide Zustände auf 1 gesetzt. Das ESP32 startet nun den neu hochgeladenen Programmcode.	1	1	1	1

**Tabelle A.4:** Abfolge der Schritte während dem Aufrufen des Download-Boot-Modus.



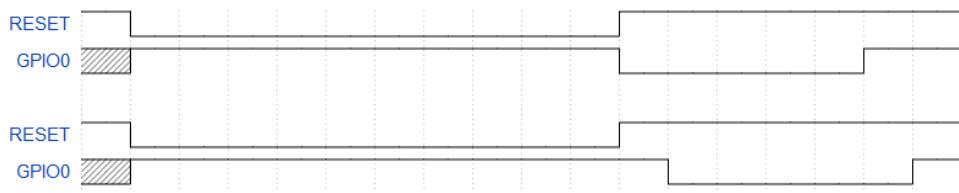
**Abbildung A.5:** Handshake zum Hochladen des Programmcodes auf das ESP32.

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



**Abbildung A.6:** Handshake zum Hochladen des Programmcodes auf das ESP32. Zoom auf gleichzeitiger Flankenwechsel RTS und DTR.

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



**Abbildung A.7:** Handshake zum Hochladen des Programmcodes auf das ESP32. Zoom auf gleichzeitiger Flankenwechsel RTS und DTR.



**Abbildung A.8:** Handshake zum Hochladen des Programmcodes auf das ESP32. Zoom auf gleichzeitiger Flankenwechsel RTS und DTR.

In einem Forum<sup>7</sup> wurde diese Auffälligkeit, welche in Kapitel ?? erwähnt wurde, auch schon besprochen. Die Diskussion führte zum selben Ergebnis wie bei 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.7 gezeigte Darstellung gefunden. Ein Kommentar weist darauf hin, das mit dem esptool.py der EN-Pin des ESP32 direkt auf RTS gehängt werden kann. So lassen sich die Zeitpunkte, zu der die Pins auf 0 sind, näher zusammenschieben.

Die Messung nach dem Einlöten der Brücke bestätigt dies. Abbildung A.8 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.

<sup>7</sup><https://forum.micropython.org/viewtopic.php?t=4607>

## B SD-Karte

### B.1 FAT32

#### B.1.1 Beschreibung Aufbau

Im **Master Boot Record** ist ein Startprogramm vorhanden, welches für BIOS baseierte Computer geschrieben wurde. Darin enthalten sind der Boot-Code in Assembler, Fehlermeldungen je nach Landessprache, Darstellungen der Fehlermeldungen, die Disk Signature, für die Organisation des Laufwerks wichtige Partitionstabelle und die Signature ID oder auch Magic Number 0x55 0xAA. Hier wird die aktive Partition ausgewählt und der Boot Sektor geladen. Da nicht über die SD-Karte gebootet wird, besteht dieser Teil nur aus Gründen der Kompatibilität.

Im **Volume Boot Record** befinden sich 3 Sektoren. Der erste Sektor enthält einen Sprungbefehl und einen "tu nichts" Befehl ("nop"), den Systemname (OEM ID), den BIOS Parameter Block, den Boot Code, der Bootloader File Name, Fehlermeldungen, Darstellung der Fehlermeldungen und ebenfalls die Magic Number 0x55 0xAA. Der zweite Sektor enthält den Anfang des erweiterten Volume Boot Records eines FAT32 Systems, den Anfang der Daten und das nächste verfügbare Cluster sowie die Magic Number. Hier kann kalkuliert werden, wieviel freier Speicher noch verfügbar ist. Der dritte Sektor ist eine Erweiterung des ersten Sektors, falls der Speicher für den Bootcode nicht ausreicht. Unter FAT32 wird für die drei Sektoren eine Sicherheitskopie abgelegt, welche zur Wiederherstellung des Volume Boot Records dient.

Im **File Allocation Table** wird organisiert, welche Cluster auf der Partition belegt oder frei sind und wo die Folgecluster eines Files sind. Die Clustergrösse kann selbst bestimmt werden und beträgt bei SD-Karten mit kleinen Dateien optimalerweise 512Bytes. Dies ist die minimale Grösse, da ein Cluster aus mehreren Sektoren besteht und ein Sektor 512 Bytes gross ist. Für die Cocktailmachine weden pro File ca. 100 Bytes verwendet, wodurch pro File ca. 412 Bytes unnutzbar werden. Der Nachteil ist, dass die FAT so grösser wird und grosse Dateien mehr zerstückelt werden. Die Cluster sind fortlaufend nummeriert, wobei sich die Nummerierung in der FAT immer auf den Ort innerhalb der FAT bezieht, nicht auf den realen Ort auf der Partition. Ist ein File grösser als 512 Bytes, so ergibt sich eine Clusterkette, welche einen Verweis auf das nächste Cluster enthält und ein Verweis, falls die Kette zu ende ist. Jeder Eintrag der FAT ist 32 Bit lang ( $4 \times 8 \text{ Bytes} = 4 \text{ Doppelwörter}$ ). Ein Cluster kann z.B den Wert 0x0E 00 00 00 haben, wobei der Verweis 0E auf das nächste Cluster zeigt. Ein Endcluster ist mit dem Doppelwort 0xFF FF FF 0F gekennzeichnet. Ist eine Datei Fragmentiert, ist die Nummerierung nicht fortlaufend, sonder spring hin und her. Konklusion: Die Werte innerhalb des FAT zeigen an, "wo sich relativ gesehen die Daten eines Files auf der Partition finden lassen. Dabei sind die Werte entweder das Zeichen für ein Clusterende oder sie zeigen auf den nächsten Cluster der Clusterkette innerhalb der FAT". Um jedoch das Anfangscluster auf der Partition finden zu können, muss das Directory Table angeschaut werden.

cite:  
<https://www.sales.com/dat>

cite:  
<https://docpla/Aufbau-des-fat32-dateisystems>

cite:  
<https://docpla/Aufbau-des-fat32-dateisystems>

Im **Directory Table** sind alle Files und alle Ordner der Partition beschrieben. Der Startsektor errechnet sich aus der letzten FAT und der Grösse der FAT (Addition). In diesem Bereich befindet sich das Root Directory, oder auch Wurzelverzeichnis genannt. Es hat eine hierarchische Baumstruktur, was bedeutet, dass "das Wurzelverzeichnis das oberste mögliche Verzeichnis ist unter dem sich weitere Verzeichnisse befinden können". Bis auf das Wurzelverzeichnis haben alle anderen Verzeichnisse ein einziges Elternverzeichnis, können aber mehrere Unterverzeichnisse haben.

Der Name des Files ist in der Regel 8 Bytes lang, worauf 3 Bytes folgen für die Bezeichnung der Fileart (z.B .txt). Ein Byte steht für die Volume ID, welche Attribute das File oder der Ordner hat. Hier befinden sich der Zeitstempel des Files, das Datum des letzten Zugriffs und das Datum der letzten Änderung des Files. Vier Bytes enthalten das Startcluster des Files oder des Ordners, vier weitere Bytes enthalten die Grösse des Files. Die restlichen Bytes sind mit 0x00 beschrieben. Um den tatsächlichen Startsektor auf der SD-Karte zu finden, wird eine Umrechnungsformel benötigt. Diese und ein Beispiel wie die File-Organisation aussieht, kann dem File "Aufbau des FAT32 Dateisystems" entnommen werden.

Die Ordner sind auf die selbe Weise organisiert, bis auf die Ausnahme, dass in FAT32 keine Größenangaben für Ordner gemacht wird. Die Bytes werden deswegen mit dem Wert 0x00 beschrieben.

cite:  
<https://docplata.com/Aufbau-des-fat32-dateisystems.html>

cite:  
<http://www.informatik.uni-ulm.de/ni/Lehre/SS15/Dateisysteme/FAT32.pdf>

cite:  
<https://docplata.com/Aufbau-des-fat32-dateisystems.html>

## C ESP32

### C.1 Tabellarischer Vergleich zwischen ESP32 und ESP8266

MCU	Xtensa Single-Core 32-bit L106 (ESP8266)	Xtensa Dual-Core 32-bit LX6 (ESP32)
802.11 b/g/n	HT20	HT40
Bluetooth	No	Bluetooth 4.2 and BLE
Arbeitsfrequenz	80 MHz	160 MHz
SRAM	No	Yes
Flash	No	Yes
GPIO	17	36
SPI/I2C/I2S/UART	2/1/2/2	4/2/2/2
ADC	10-bit	12-bit
Ethernet Interface	No	Yes
Touchsensor	No	Yes
Temperatursensor	No	Yes
Hall-Sensor	No	Yes
Arbeitstemperatur	-40°C to 125°C	-40°C to 125°C
Price	\$ (3\$ - \$6)	\$\$ (\$6 - \$12)

Tabelle C.1: Vergleich ESP8266 zu ESP32.

### C.2 Strapping-Pins

Ausgangsspannung internen Spannungsregler (VDD_SDIO)				
RS232	ESP	default	3.3V	1.8V
RTS	IO12	Pull-down	θ <sup>8</sup>	+
Boot-Modus				
RS232	ESP	default	SPI-flash Boot	Download Boot
DTR	IO0	Pull-up	1	0
-	IO2	Pull-down	Egal	0
Debugging Log Print über U0TXD während Booten				
RS232	ESP	default	U0TXD Active	U0TXD Silent
RTS	IO12	Pull-down	1	0
Timing <sup>9</sup> des SDIO				
RS232	ESP	default	FF Sampling FF Output	FF Sampling SF Output
CTS	IO15	Pull-up	0	0
-	GPIO5	Pull-up	0	1
				0
				1

Tabelle C.2: Tabelle Pinkonfiguration für Strapping-Pins.

### Spannung des internen Spannungsgeglers (VDD\_SDIO)

Das ESP32 hat einen eingebauten host controller für SD/SDIO/MMC-Speichergeräte. Dieser kann mit 3.3V (IO12 = 0) oder 1.8V (IO12 = 1) betrieben werden. Wird der Pin auf 1 gesetzt, könnte ein Brown-out der IO-Versorgungsspannung VCC\_IO ausgelöst werden.

### Booting Mode

Der Pin IO0 gibt vor, ob das ESP32 vom internen SPI-flash bootet (IO0 = 1) oder ob neuer Code in den flash Speicher geschrieben wird (IO0 = 0). Soll das ESP vom Flash-Speicher booten, so hat der Pin IO2 kein Einfluss, um jedoch den Code speichern zu können, muss der Pin auf 0 sein.

### De-/Aktivieren vom Debug-Log über U0TXD während dem Bootvorgang

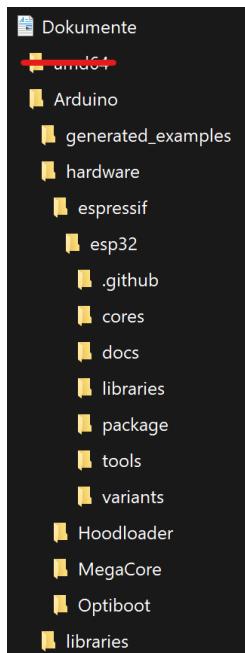
Über den Pin IO12 kann konfiguriert werden, ob während dem Booten ein Debug-Log über die Serielleschnittstelle gesendet wird (IO12 = 1) oder nicht (IO12 = 0).

### Timing der Kommunikation mit dem SDIO Slave:

Über die Pins IO15 und IO5 kann das Übertragungsprotokoll des SDIO-Slaves festgelegt werden. Dabei kommt es darauf an, ob das Sampling auf eine fallende Flanke (IO15 = 0) oder steigende Flanke (IO15 = 1) geschehen soll, und ob der Output auf eine fallende Flanke (IO5 = 0) oder steigende Flanke (IO5 = 1) geschehen soll.

## C.3 Inbetriebnahme

### C.3.1 Arduino IDE



**Abbildung C.1:** Ordnerstruktur Boards Arduino IDE.

<sup>18</sup>Da das ESP32-WROOM-32U einen 3.3V SPI flash integriert hat, kann der MTDI nicht auf 1 gesetzt werden, wenn die Module aufgestartet sind.

<sup>9</sup>FF = Fallende Flanke, SF = Steigende Flanke

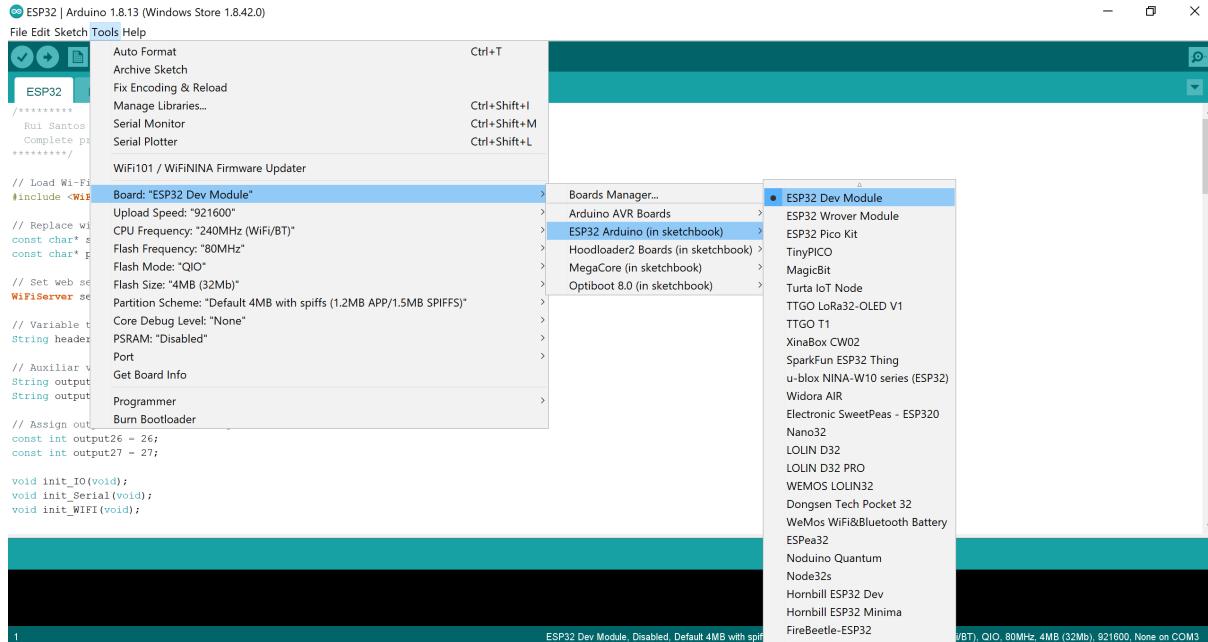


Abbildung C.2: ESP32 auswählen in der Arduino IDEs.

Verbundene Geräte:	1 von 8	
Gerätename	IP-Adresse	Physische Adresse (MAC)
espressif	192.168.137.22	3c:71:bf:fe:42:70

Abbildung C.3: ESP32 verbindet sich nach Hochladen des Codes mit dem WiFi.

```

COM17
11!22v!AP!!Iq?Connecting to Master-Laptop
.
WiFi connected.
IP address:
192.168.137.22
New Client.
GET / HTTP/1.1
Host: 192.168.137.22
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0) Gecko/20100101 Firefox/80.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

New Client.

```

At the bottom of the window, there are checkboxes for 'Autoscroll' and 'Show timestamp', and a dropdown for '9600 baud' with a 'Clear output' button.

Abbildung C.4: ESP32 erkennt Netzwerk und Client (Browser).

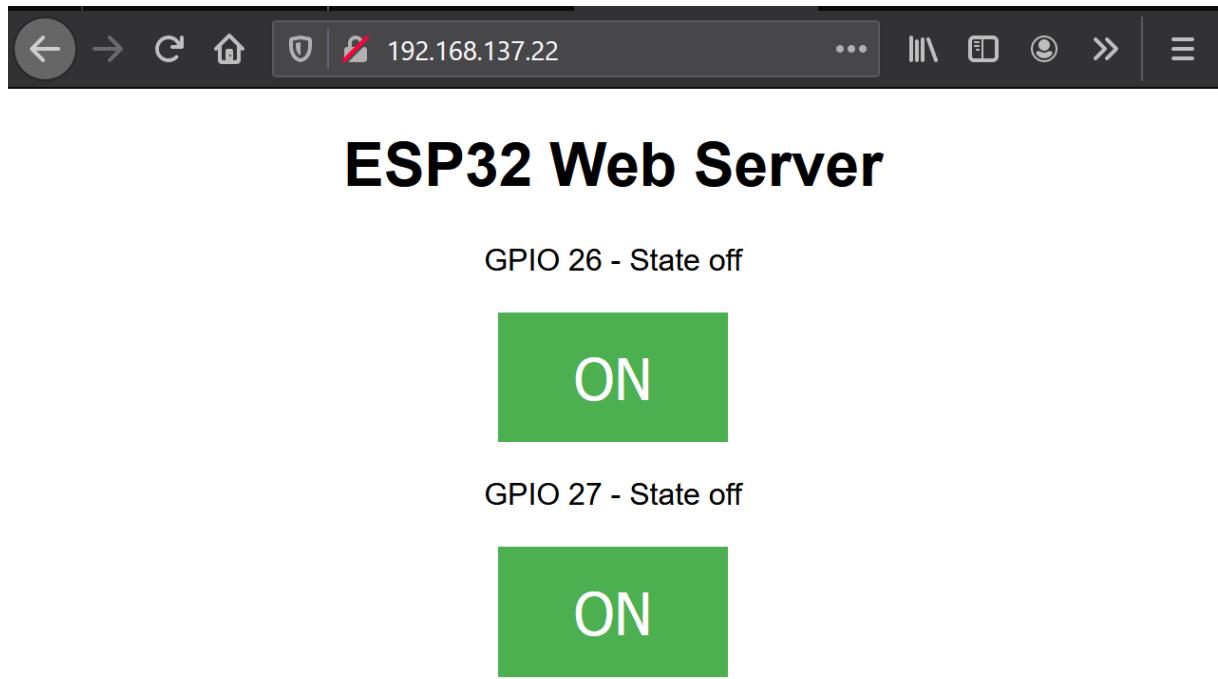


Abbildung C.5: ESP32 als Host.

## D Mikrocontroller

### D.1 Inbetriebnahme

#### D.1.1 Fuse-Bits

Da während dem Entwickeln 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 (anders als stk500v2 Erklärung). 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 µC verwenden wir einen 16MHz Full-Swing-Crystal-Oszillator, 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 TMC4671 noch aktiv ist, wird der Motor weiter gefahren. Dieses Risiko soll eingeschränkt werden, indem dieser Modus ausgeschaltet wird.

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

Abbildung D.1: Tabelle Brown-out-Detection.

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

Abbildung D.2: Tabelle Frequenzbereich Crystal Oszillator.

Oscillator Source / Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V <sub>CC</sub> = 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	16K CK	14CK + 65ms	1	11

Abbildung D.3: Tabelle Aufstartzeit.

cite :  
<https://www-user.tu-chemnitz.de/ha/viewchm.pdf>

cite: Datenblatt Atmega 2560, Seite 361

cite: Datenblatt Atmega 2560, Seite 43

cite: Datenblatt Atmega 2560, Seite 43

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Appli-cation Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (Start Boot Loader Section)
1	1	512 words	4	0x0000 - 0x7FFF	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	4096 words	32	0x0000 - 0x6FFF	0x7000 - 0x7FFF	0x6FFF	0x7000

Abbildung D.4: Tabelle Bootloader Speicherplatz.

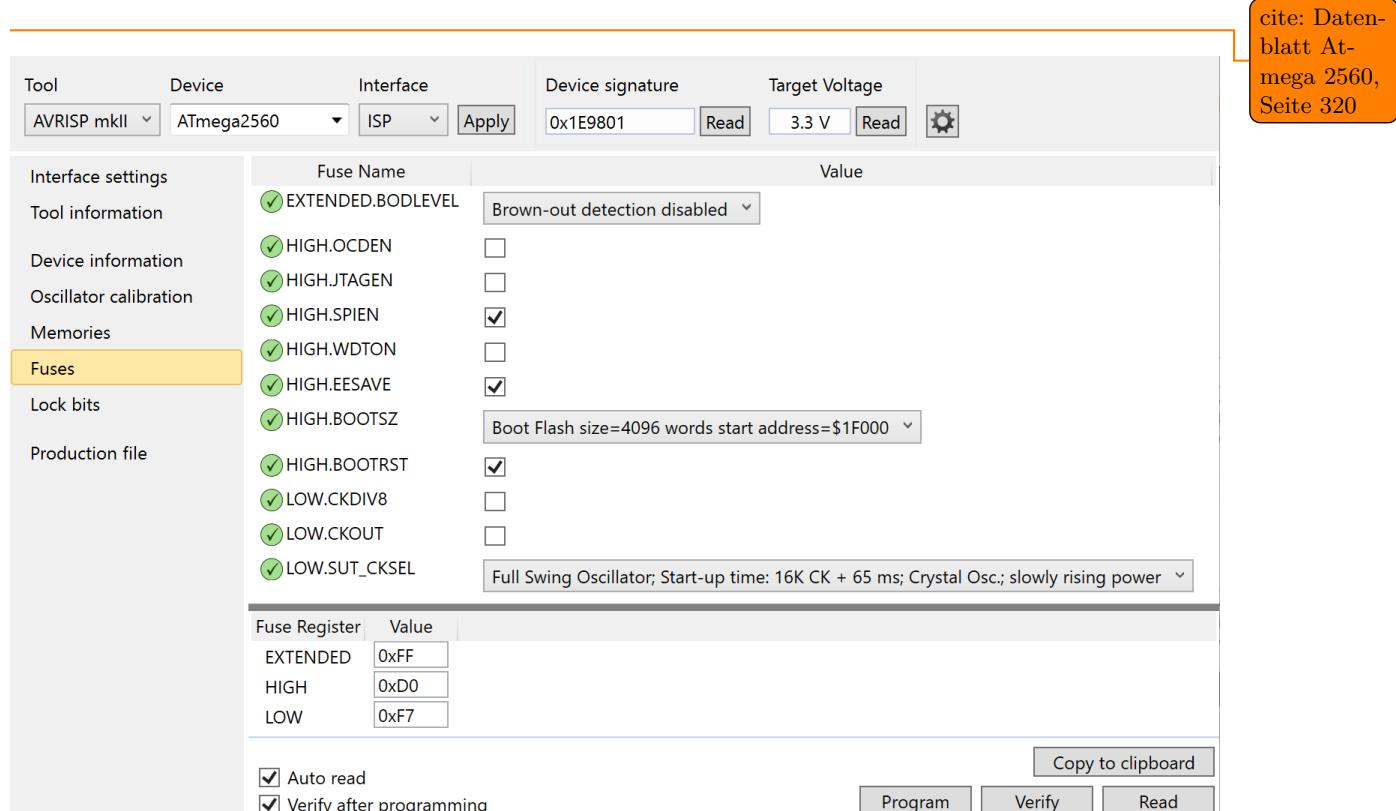


Abbildung D.5: Fuse-Bits Atmega2560.

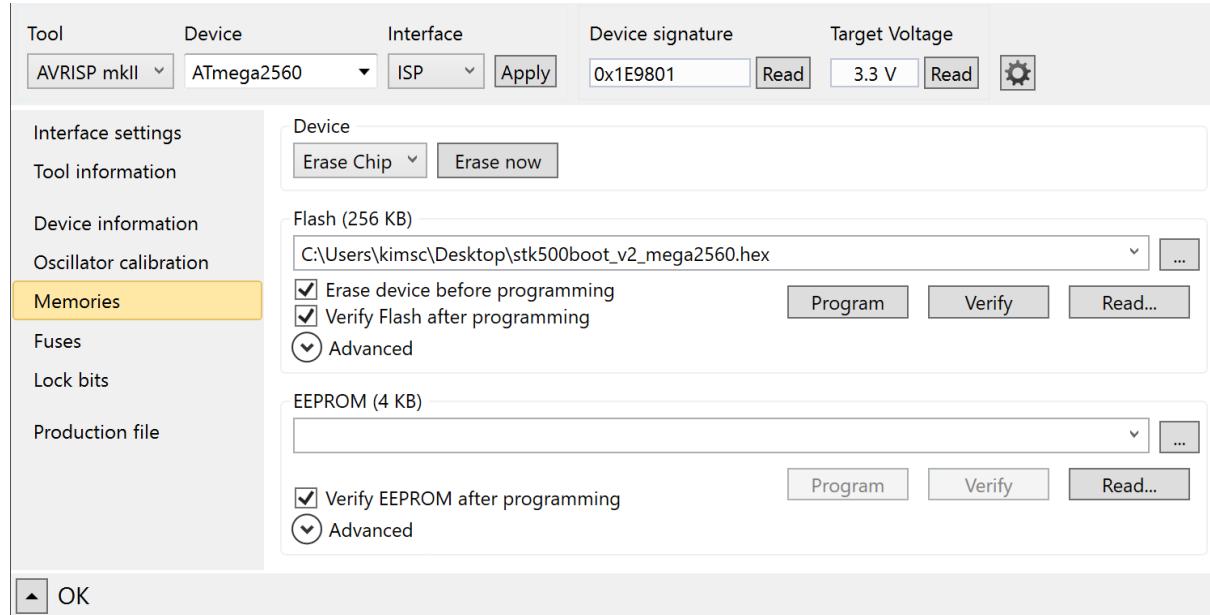
### D.1.2 Bootloader

Ein Bootloader (BL) ermöglicht seitens µC den Programmierzorgang über USB und stellt sicher, dass der frisch kompilierte Code in den Programmspeicher des µC gelangt. Im Grunde ist der BL ein Code am Beginn des Programmablaufs, welcher entsprechend nach einem Reset aufgerufen wird. Innerhalb des BL-Codes wird für zwei Sekunden auf eingehende Daten der UART0-Schnittstelle gewartet. Wird innerhalb dieser zwei Sekunden ein Anwenderprogramm in Form eines HEX-Files an den Mikrocontroller gesendet, wird es gemäss stk500v2-Protokoll in den Flash-Speicher geladen. Aufgrund des BL geht es nach einem Reset des µCs zwei Sekunden, bis das eigentliche Programm startet.

Für den µC des Cocktaillmixers wird ein stk500v2-BL verwendet. Dieser kann über Github<sup>10</sup> heruntergeladen werden. Im File stk500v2.c befinden sich einige Anweisungen, welche Fuse- und Lock-Bits wann gesetzt werden müssen. Achtung, die Angaben haben im Falle des Cocktaillmixers nur mit Abweichungen funktioniert (4096 words anstelle 1024). Durch Anpassen des

<sup>10</sup><https://github.com/arduino/Arduino-stk500v2-bootloader/blob/master/stk500boot.c>

start-Vektors auf den Programmspeicher und Setzen der korrekten Bootloader-Fuses wäre es möglich, den BL-Speicherplatz zu verkleinern um den Programmspeicher zu vergrößern.



**Abbildung D.6:** Bootloader brennen.

### D.1.3 Lock-Bits

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.“

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 D.7:** Tabelle Memory Lock.

cit: Datenblatt Atmega 2560, Seite 326

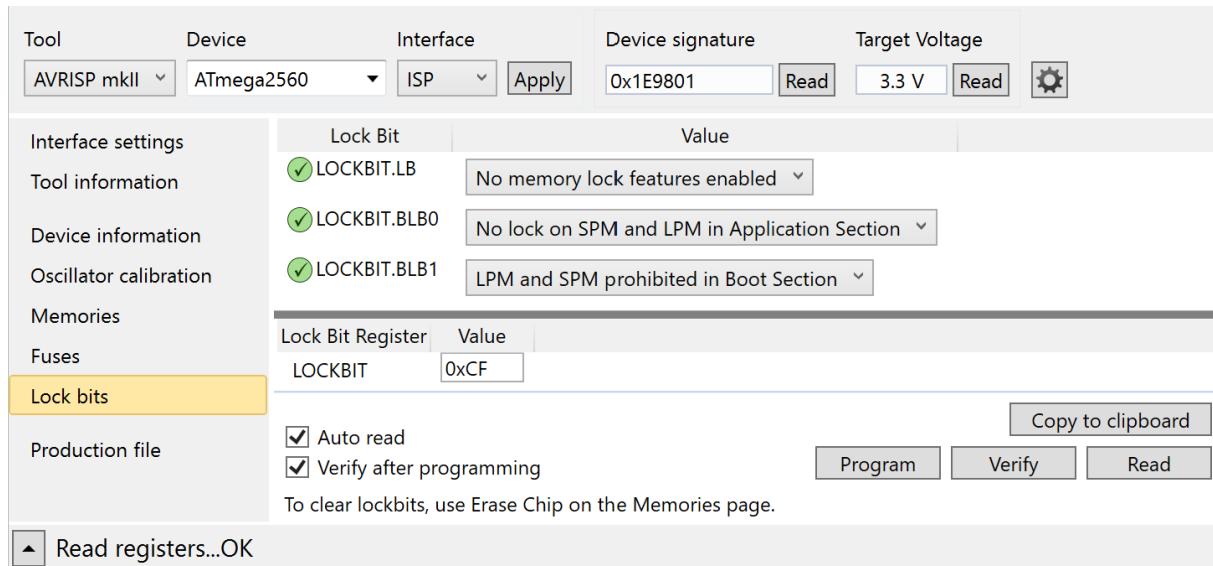


Abbildung D.8: Lock-Bits Atmega2560.

#### D.1.4 Einbinden AVRdude und stk500v2 (wiring)

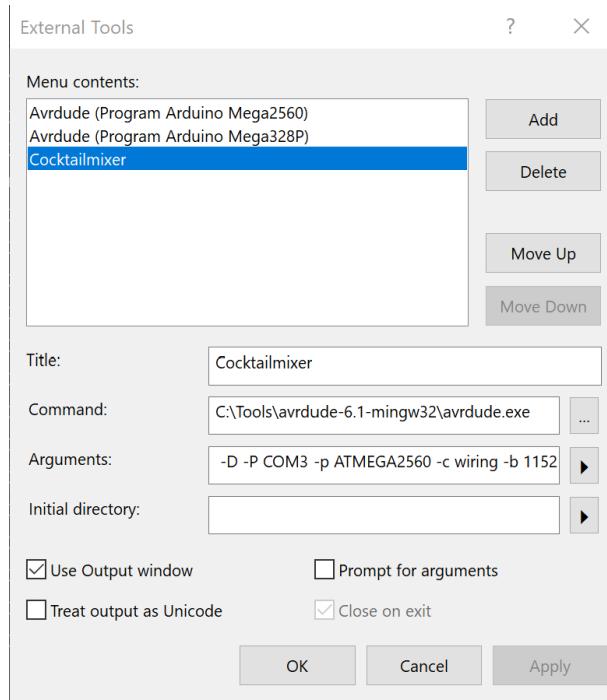


Abbildung D.9: External Tools Atmega2560.

### D.1.5 Schwingung 16MHz Quarz

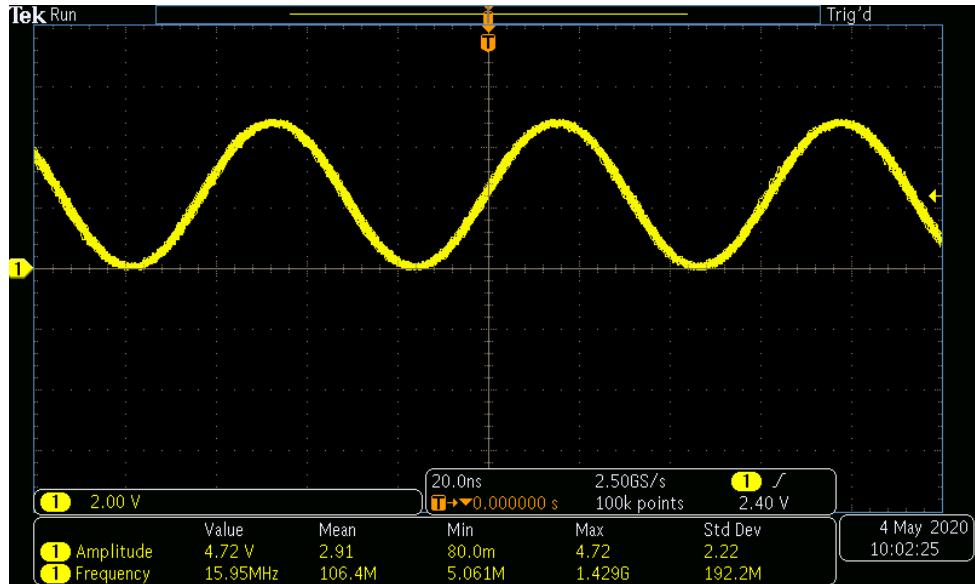


Abbildung D.10: Schwingung des Oszillators

## D.2 Speicherorganisation

### D.2.1 Doubly Dynamic Linked List

In Abbildung D.11 wird nur der start-Pointer (head) initialisiert, ohne auf ein Element zu zeigen (NULL). Sobald der benötigte Speicherplatz des Elementes alloziert wurde, wird der Zeiger auf die angelegte Struktur (Element) gelegt. Dem Head-Zeiger wurde nun ein Element zugewiesen und der Beginn der Liste wurde definiert.

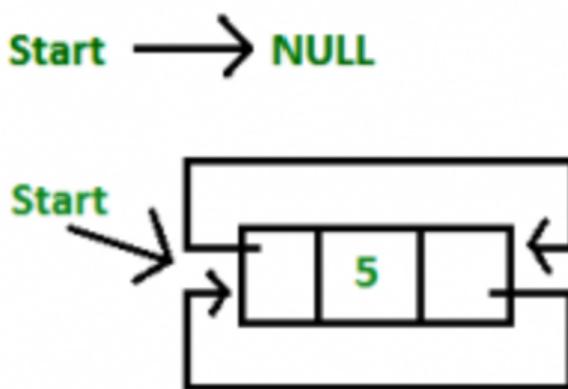


Abbildung D.11: Doppelt verkettete Liste mit zwei Elementen.

Abbildung D.12 zeigt eine bestehende Liste mit zwei Elementen. Ein drittes Element soll am Schluss eingefügt werden. Dazu muss zuerst der Speicherplatz für das neue Element alloziert werden (Data, next, prev). Danach können die Zeiger der bestehenden Elemente (in diesem Falle noch head und tail) umgelegt werden und die Zeiger des neuen Elementes auf das vorhergehende und nächste Element gelegt werden. Auch der tail-Zeiger muss nun auf das neue Element gelegt werden.

cite  
Bild:[https://www.tutorialspoint.com/circular\\_linked\\_list/circular\\_linked\\_list\\_set\\_1.htm](https://www.tutorialspoint.com/circular_linked_list/circular_linked_list_set_1.htm)

cite  
Bild:[https://www.tutorialspoint.com/circular\\_linked\\_list/circular\\_linked\\_list\\_set\\_1.htm](https://www.tutorialspoint.com/circular_linked_list/circular_linked_list_set_1.htm)

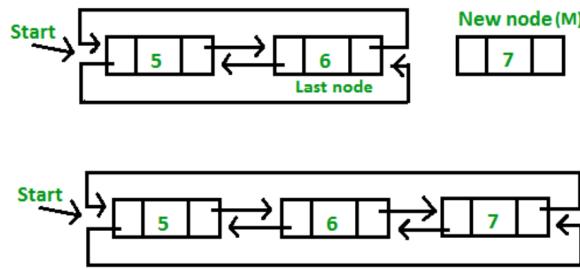


Abbildung D.12: Doppelt verkettete Liste mit zwei Elementen.

Abbildung D.13 zeigt ebenfalls eine bestehende Liste mit zwei Elementen. Ein drittes Element soll am Beginn eingefügt werden. Dazu muss zuerst der Speicherplatz für das neue Element alloziert werden (Data, next, prev). Danach können die Zeiger der bestehenden Elemente (in diesem Falle noch head und tail) umgelegt werden und die Zeiger des neuen Elementes auf das vorhergehende und nächste Element gelegt werden. Auch der head-Zeiger muss nun auf das neue Element gelegt werden.

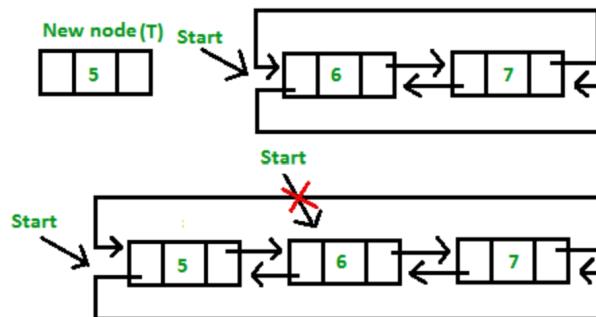


Abbildung D.13: Doppelt verkettete Liste mit zwei Elementen.

Abbildung D.14 zeigt eine bestehende Liste mit vier Elementen. Ein fünftes Element soll in der Mitte eingefügt werden. Dazu muss zuerst der Speicherplatz für das neue Element alloziert werden (Data, next, prev). Danach können die Zeiger der bestehenden Elemente umgelegt werden und die Zeiger des neuen Elementes auf das vorhergehende und nächste Element gelegt werden.

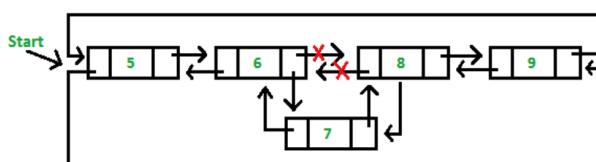


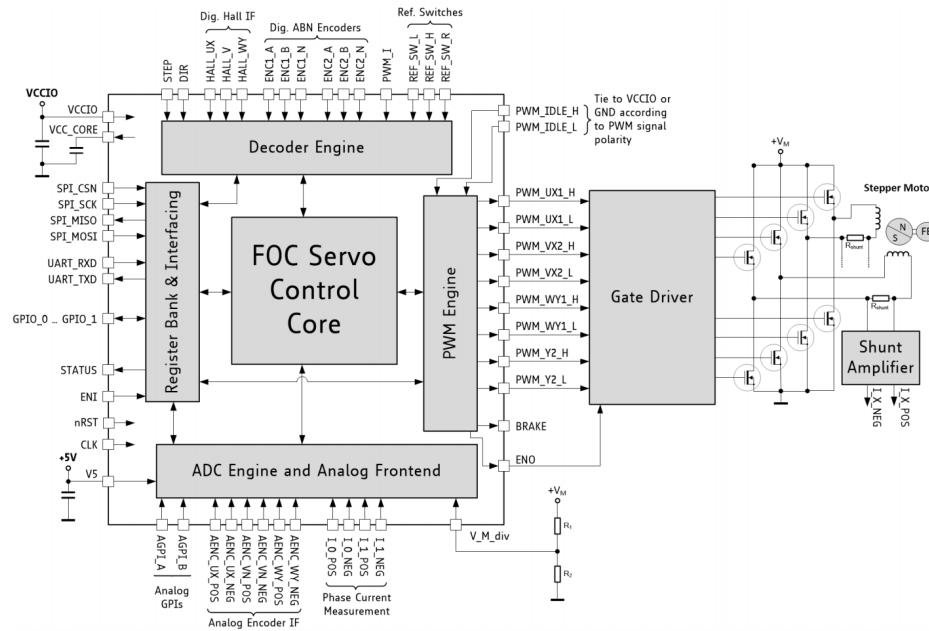
Abbildung D.14: Doppelt verkettete Liste mit zwei Elementen.

cite  
Bild:[https://www.tutorialspoint.com/circular\\_linked\\_list\\_set\\_1/introduction\\_and\\_insertion/](https://www.tutorialspoint.com/circular_linked_list_set_1/introduction_and_insertion/)

cite  
Bild:[https://www.tutorialspoint.com/circular\\_linked\\_list\\_set\\_1/introduction\\_and\\_insertion/](https://www.tutorialspoint.com/circular_linked_list_set_1/introduction_and_insertion/)

## E TMC4671

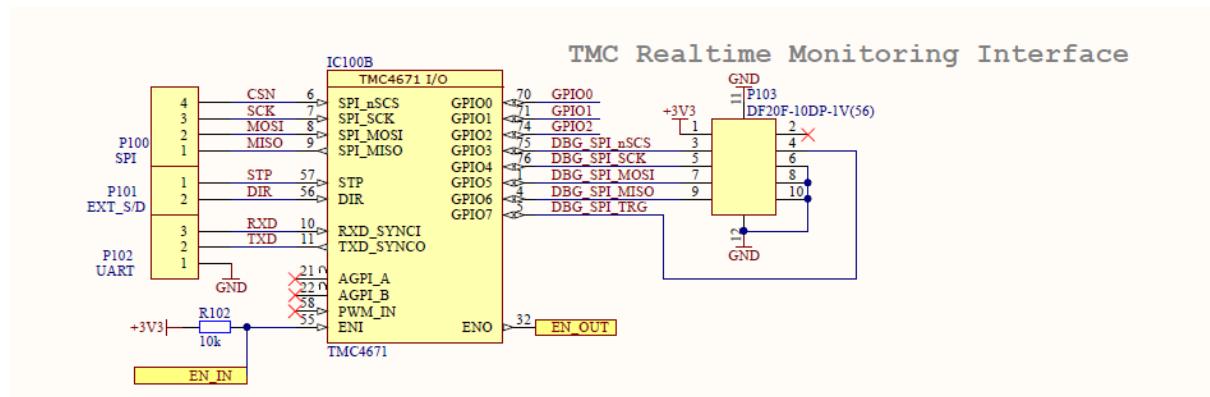
## E.1 Standard-Schaltkreis



**Abbildung E.1:** Standard-Anwendungs-Schaltung. [3, S.138]

citeTMC467  
Datenblatt

## E.2 BOB



**Abbildung E.2:** SPI-Input des TMC4671-BOB. [4, S.2]

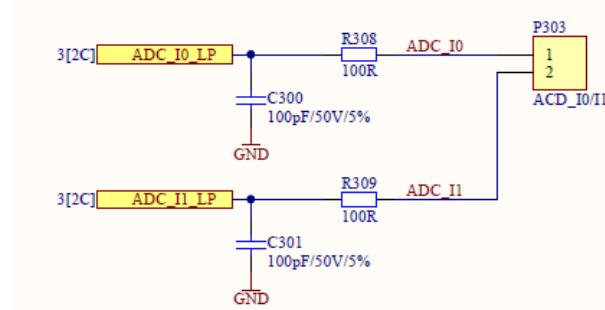


Abbildung E.3: Phasenstroeme-Input des TMC4671-BOB. [4, S.4]

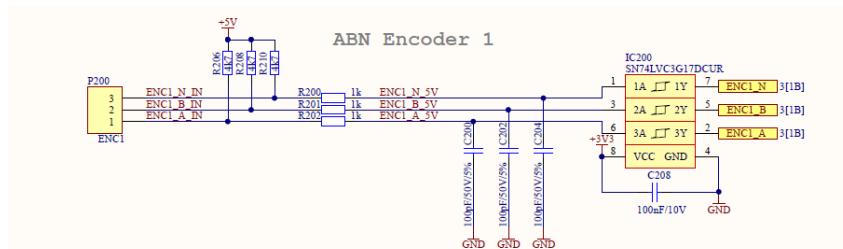


Abbildung E.4: ABN-Encoder-Input des TMC4671-BOB. [4, S.3]

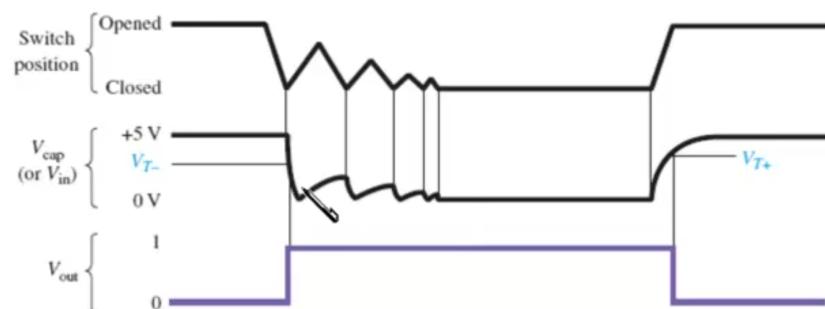
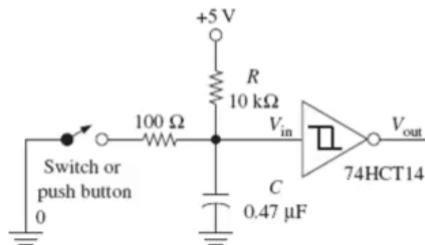


Abbildung E.5: Schmitt-Trigger Debounce-Schaltung. [23, 3:00]

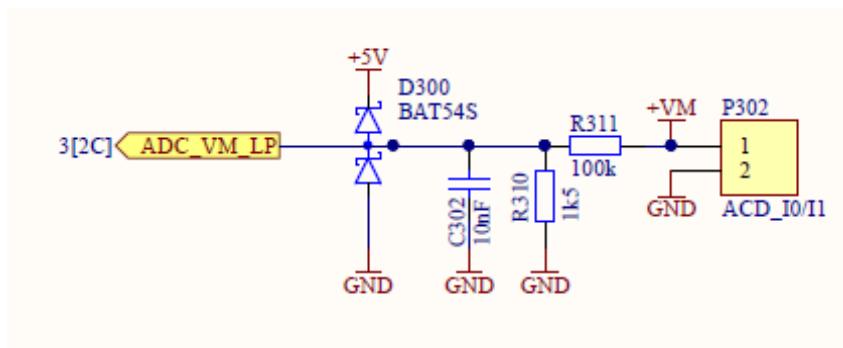


Abbildung E.6: Motorspannung-Input des TMC4671-BOB. [4, S.4]

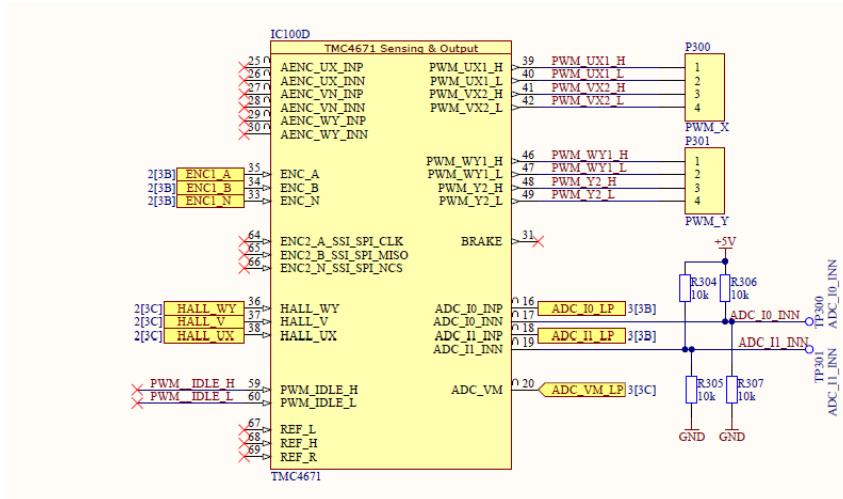


Abbildung E.7: PWM-Output des TMC4671-BOB. [4, S.3]

## E.3 Inbetriebnahme

### E.3.1 Setup

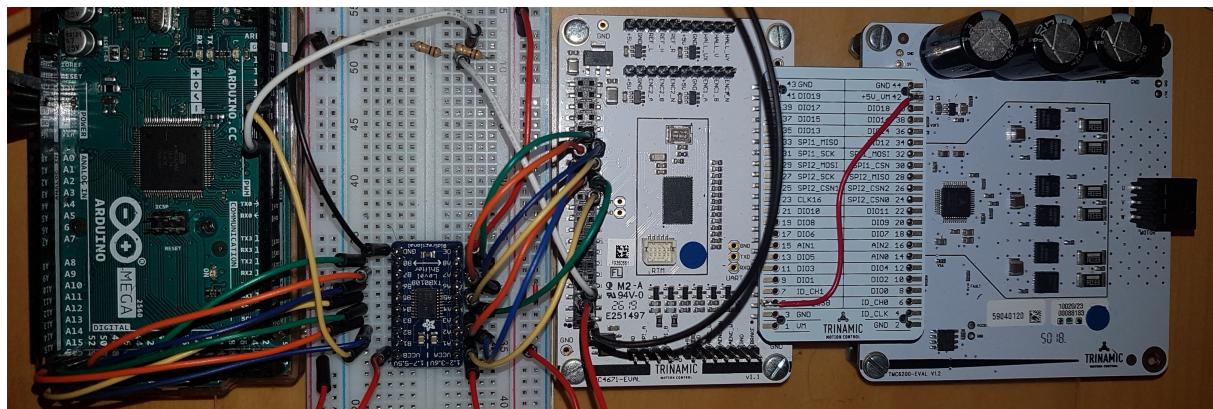


Abbildung E.8: Gesamtansicht Setup.

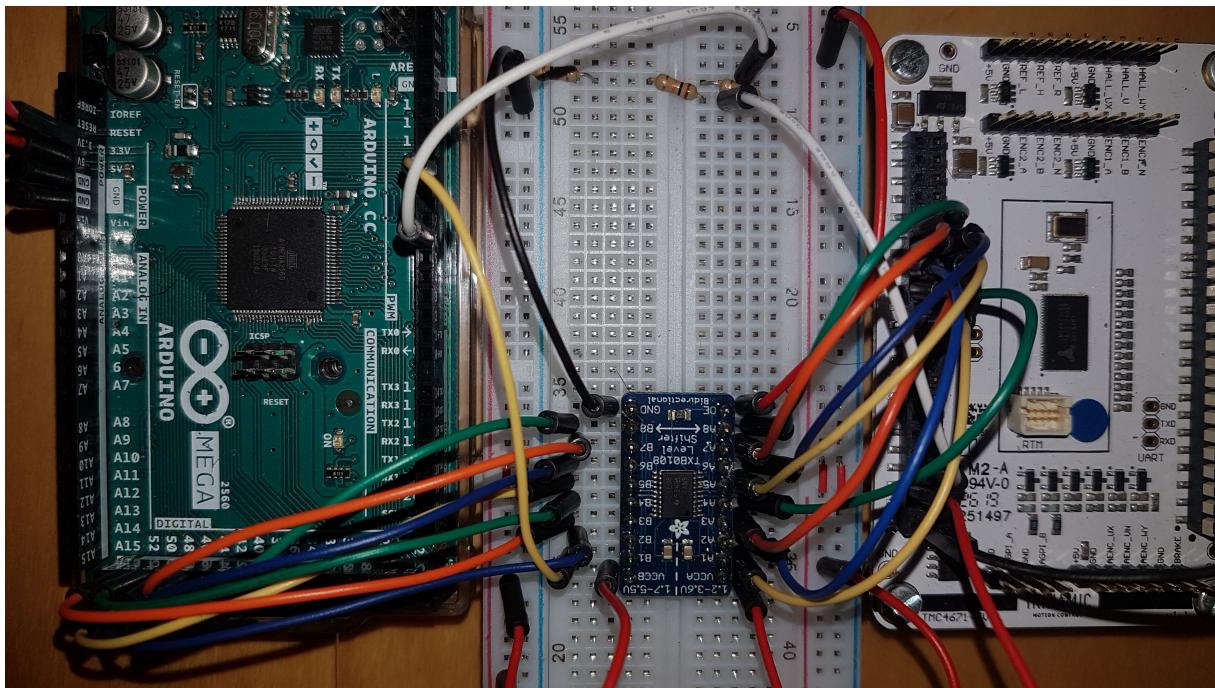


Abbildung E.9: Gesamtansicht Setup Wiring.

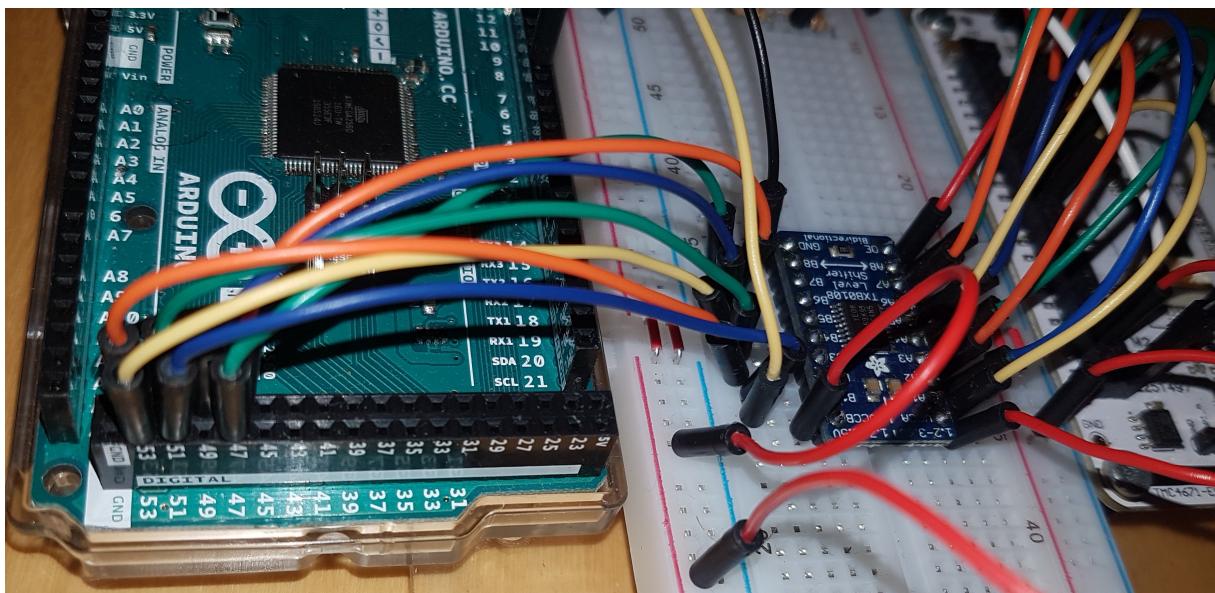


Abbildung E.10: Setup mit Fokus auf Arduino.

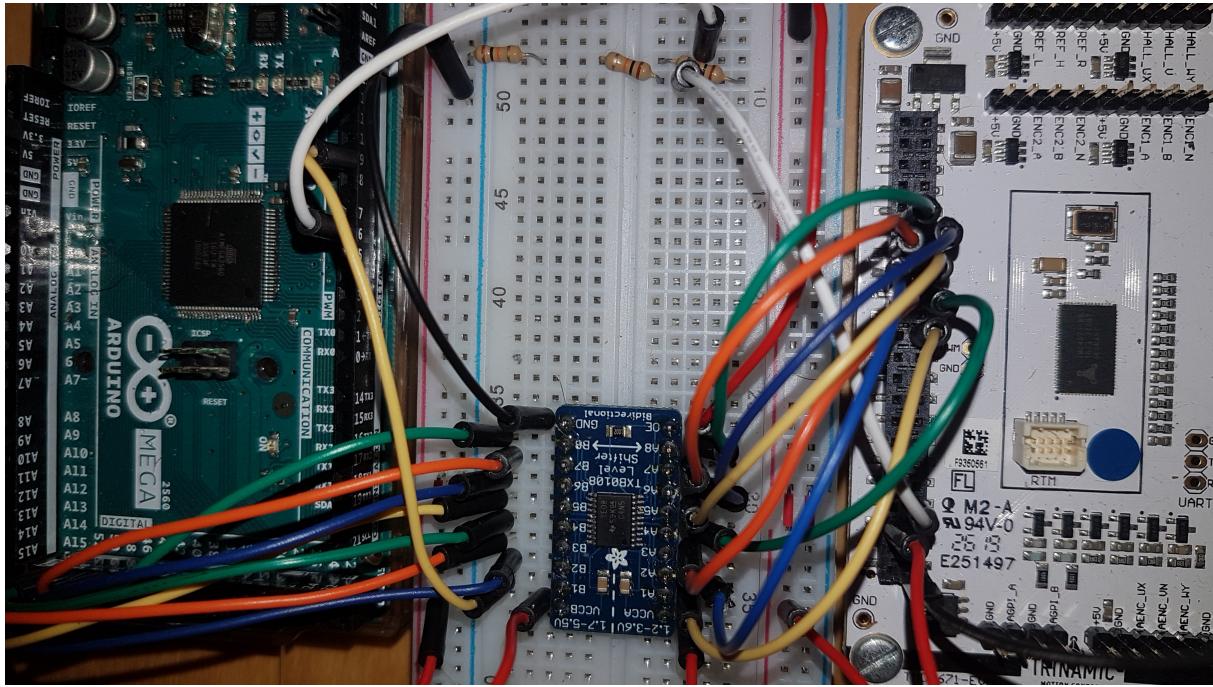


Abbildung E.11: Setup mit Fokus auf TMC4674-EVAL.

### E.3.2 Inbetriebnahme SPI-Kommunikation

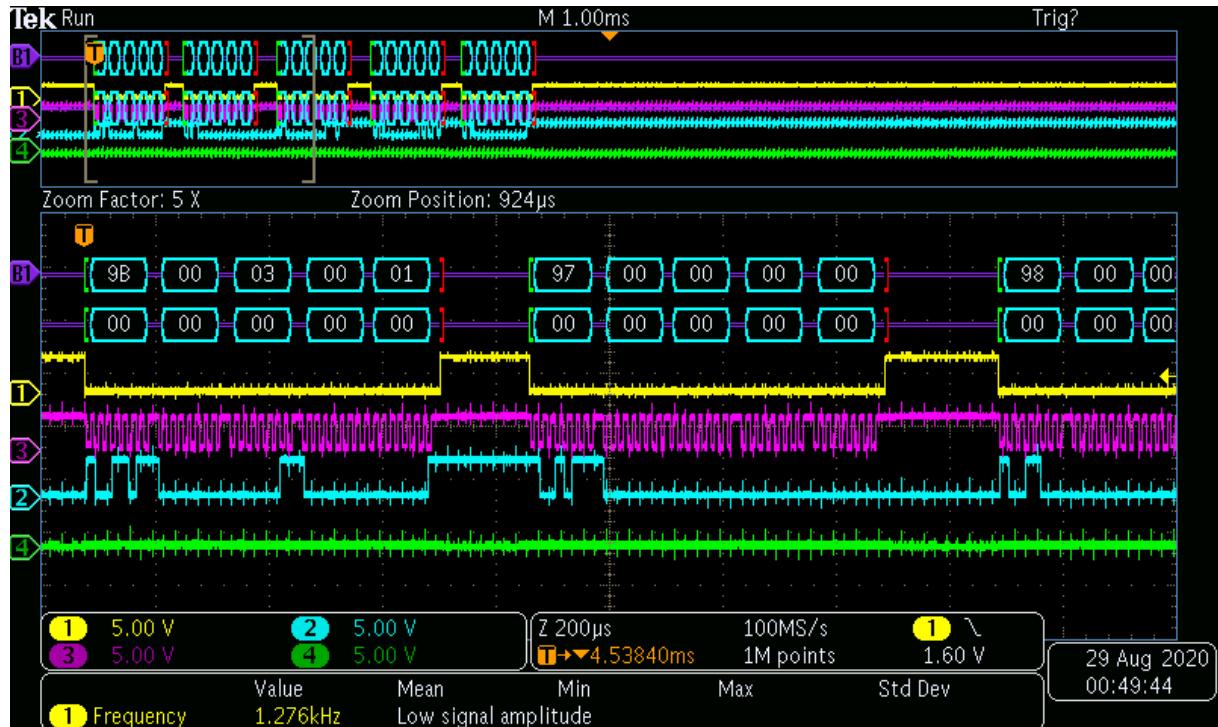


Abbildung E.12: SPI-Übertragung Write (Hier Motor Pole Pairs).

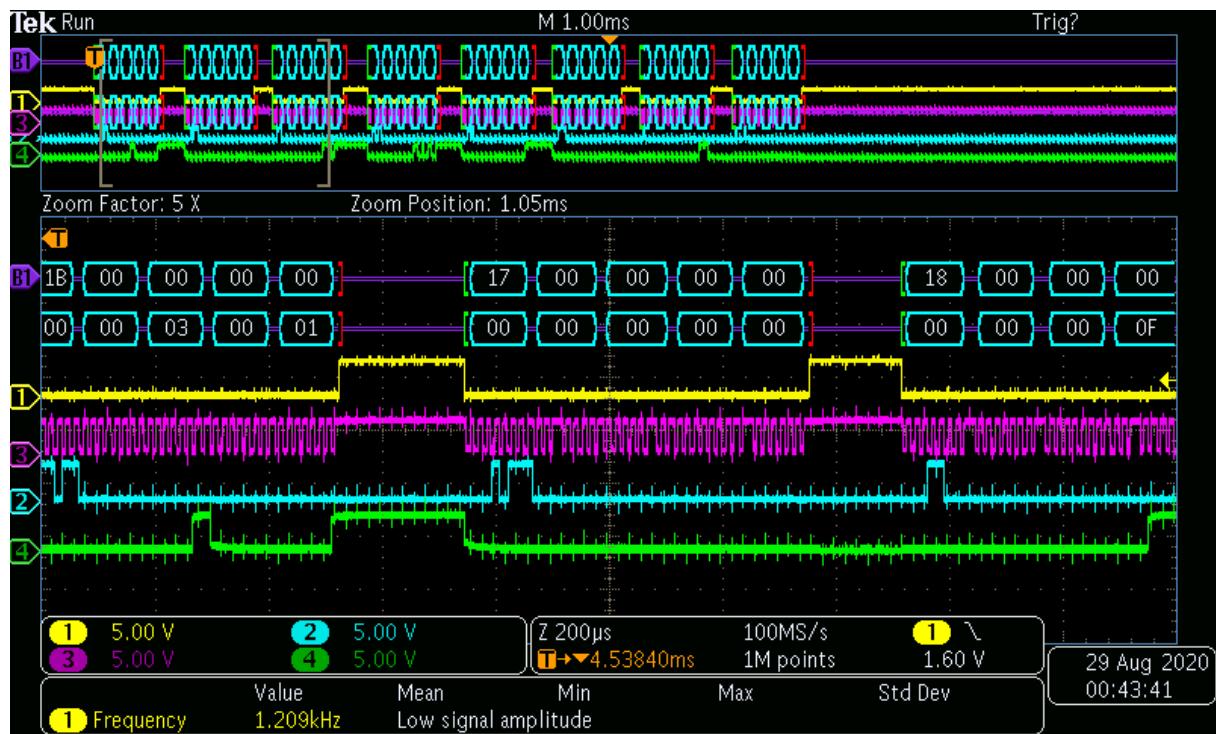


Abbildung E.13: SPI-Übertragung Read (Hier Motor Pole Pairs).

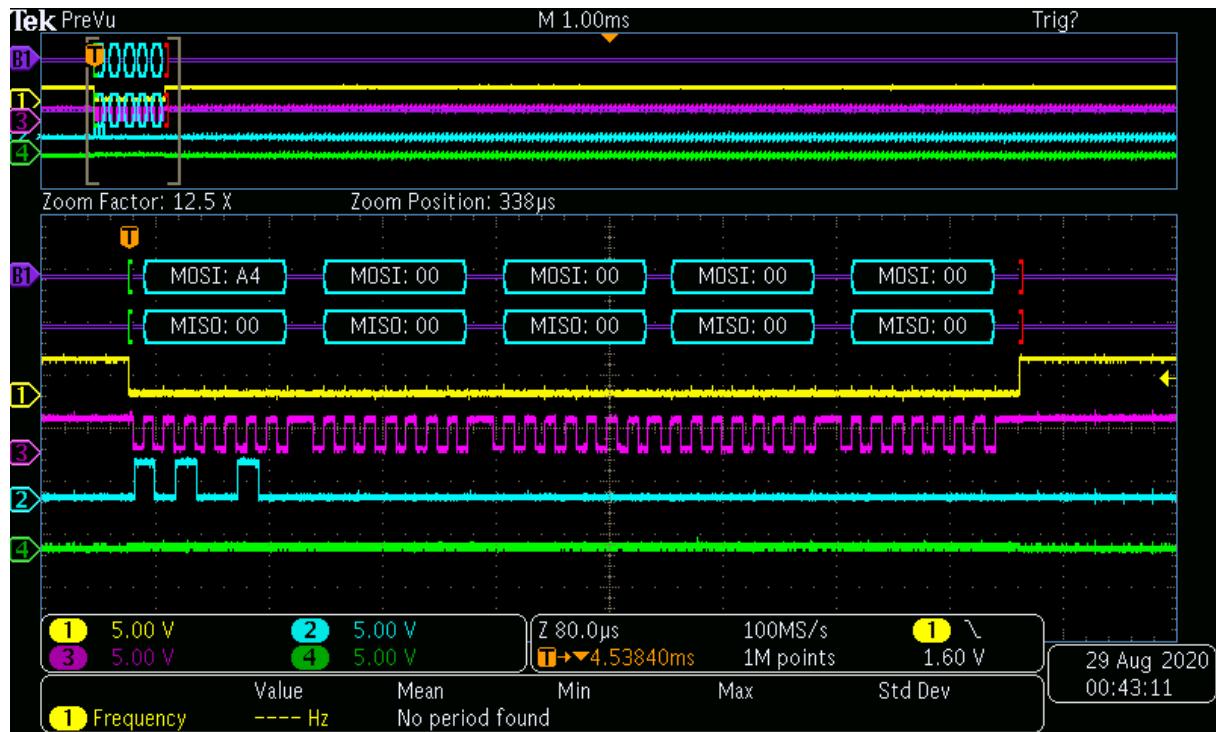


Abbildung E.14: Übertragung mit Zoom (Testdrive).



Abbildung E.15: Event-Table Inbetriebnahme TMC4671.



Abbildung E.16: Event-Table Inbetriebnahme TMC4671.

### E.3.3 Inbetriebnahme Gate-Ctrl

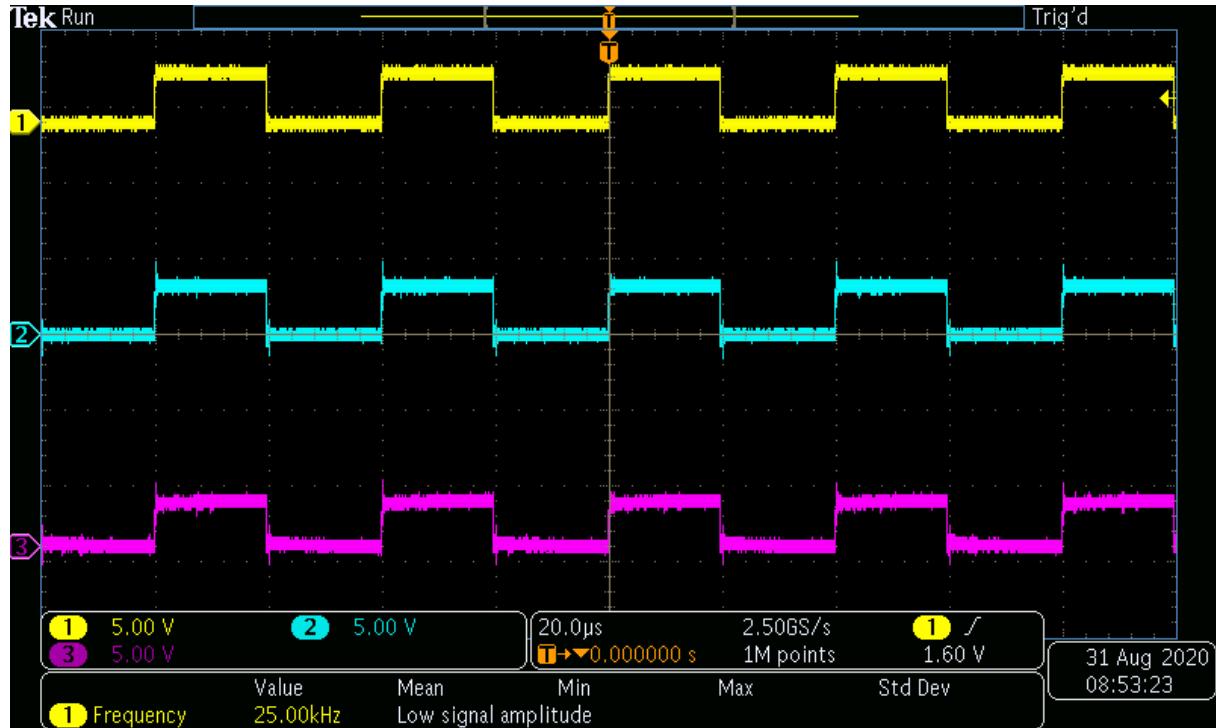


Abbildung E.17: Steuersignale PWM 48V. Gelb = U, Blau = V, Magenta = W

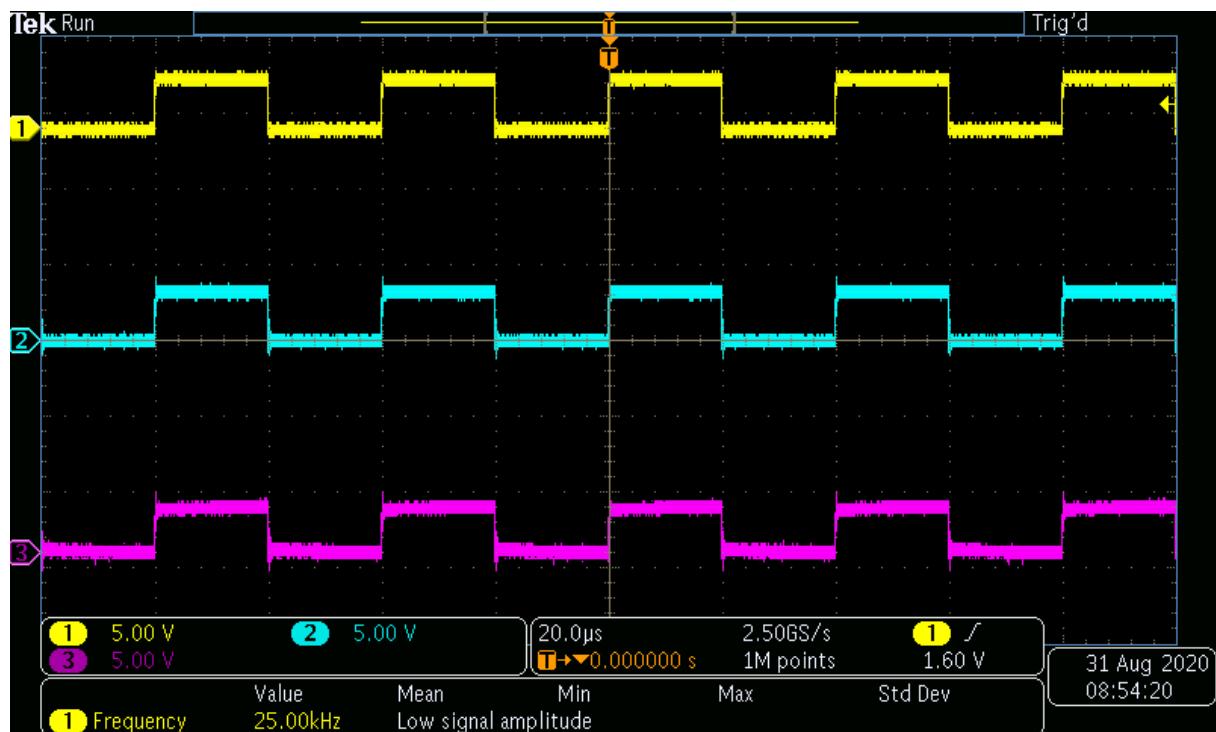


Abbildung E.18: Steuersignale PWM 0V. Gelb = U, Blau = V, Magenta = W

## F TMC6200

### F.1 Standard-Schaltkreis

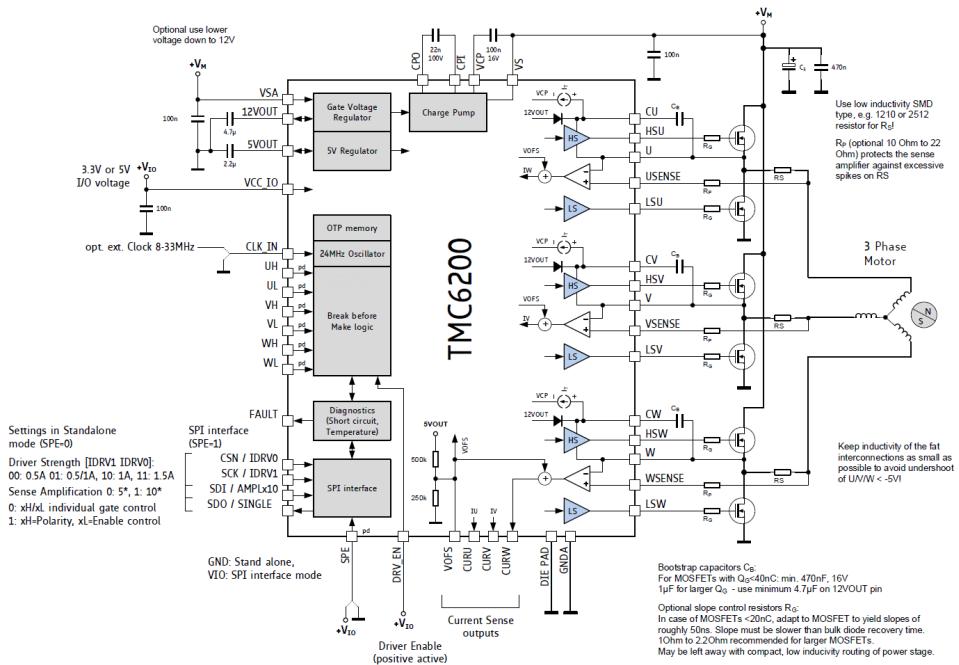


Abbildung F.1: Standard-Anwendungs-Schaltung TMC6200.

### F.2 Blockdiagramm

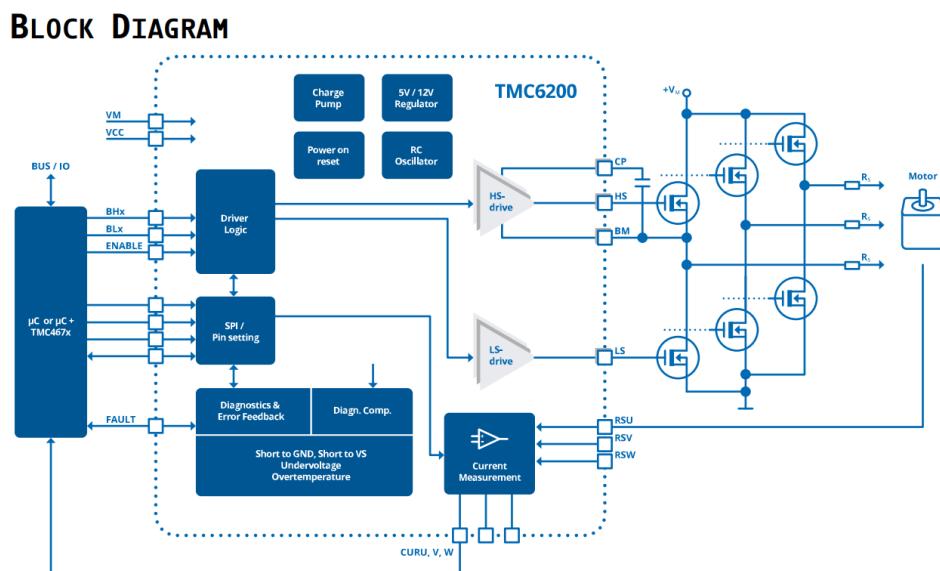


Abbildung F.2: Blockdiagramm TMC6200.

### F.3 Externe Gate-Spannungsversorgung

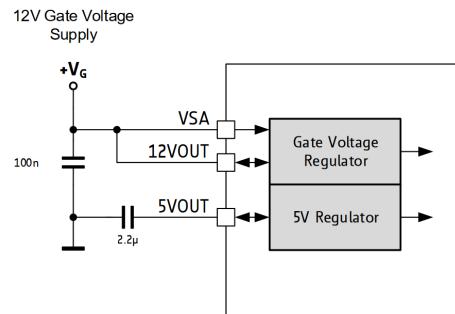


Abbildung F.3: Schema externe Gate-Spannungsversorgung.

### F.4 Inbetriebnahme

#### F.4.1 Inbetriebnahme Setup

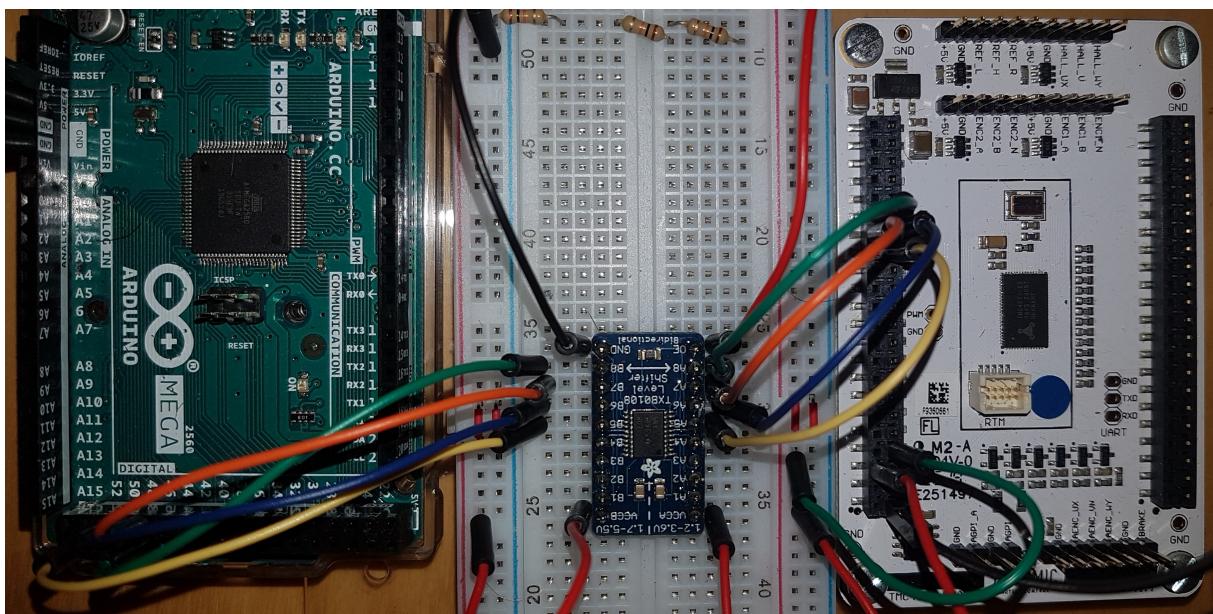


Abbildung F.4: Gesamtansicht Setup.

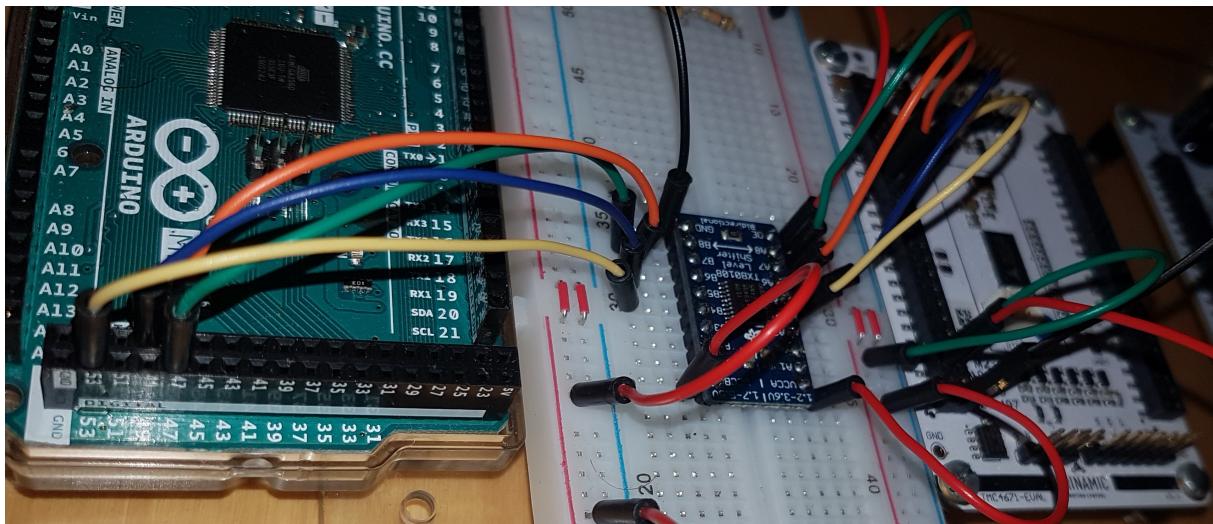


Abbildung F.5: Setup mit Fokus auf Arduino.

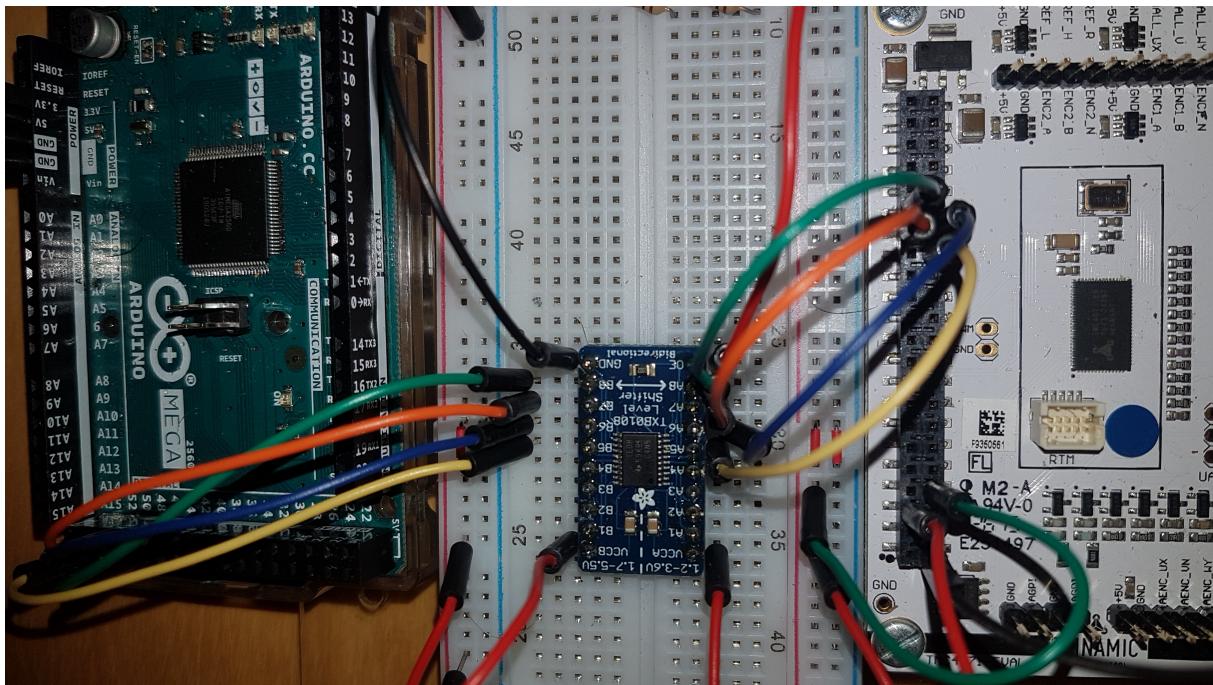


Abbildung F.6: Setup mit Fokus auf TMC4674-EVAL.

#### F.4.2 Inbetriebnahme Parameter

#### F.4.3 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.7:** Tabelle zur Bestimmung des Strommesswiderstandes aus dem Datenblatt von Trinamic.

#### F.4.4 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.8:** Tabelle zur Bestimmung der Gatewiderstände aus dem Datenblatt von Trinamic.

#### F.4.5 Inbetriebnahme SPI-Kommunikation

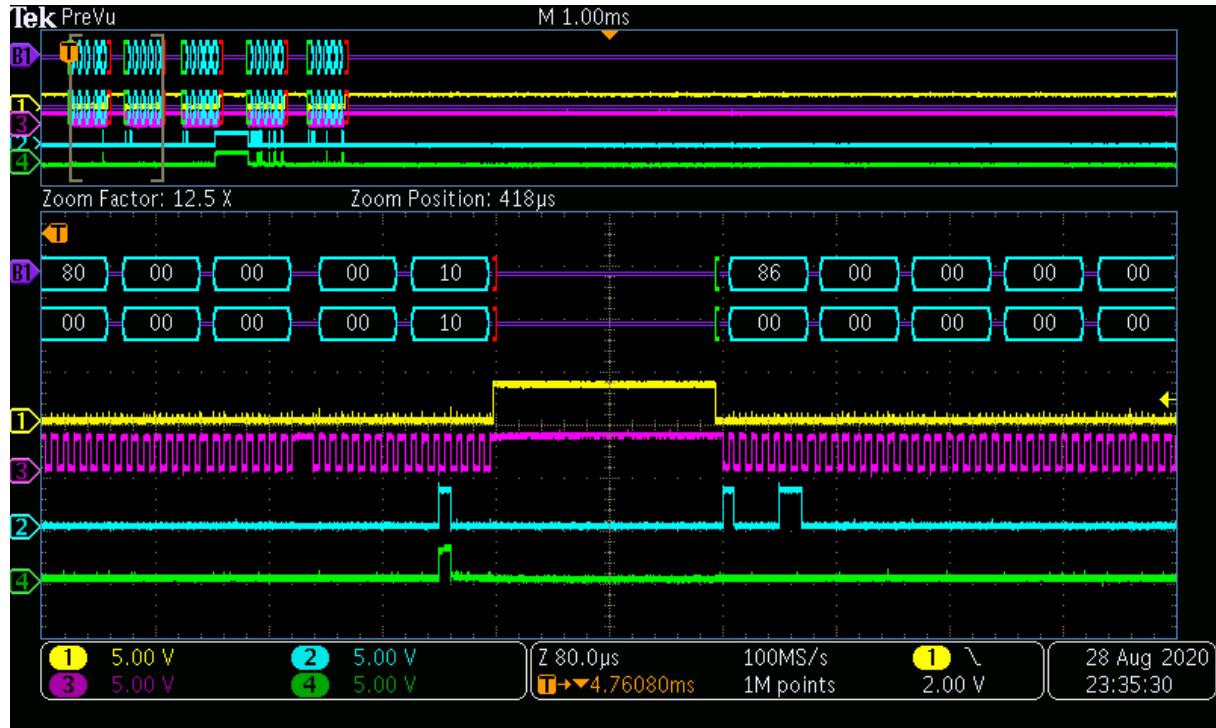


Abbildung F.9: SPI-Übertragung Write 1.

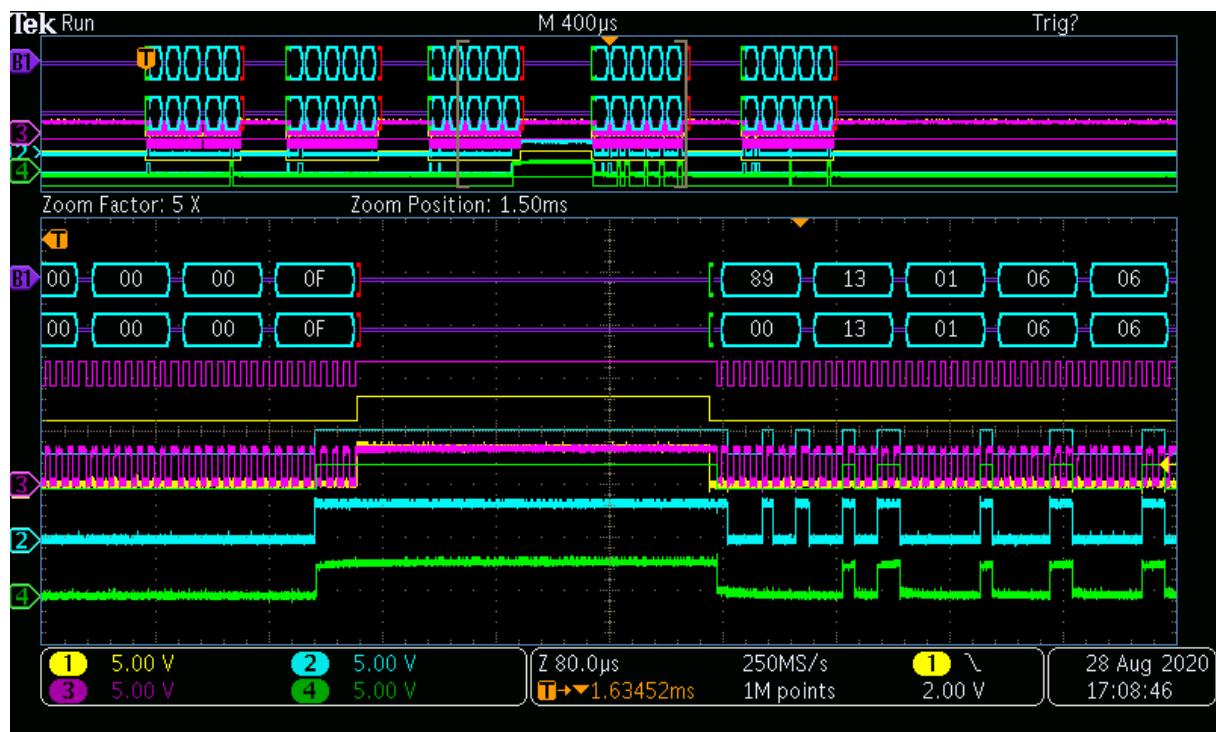


Abbildung F.10: SPI-Übertragung Write 2.

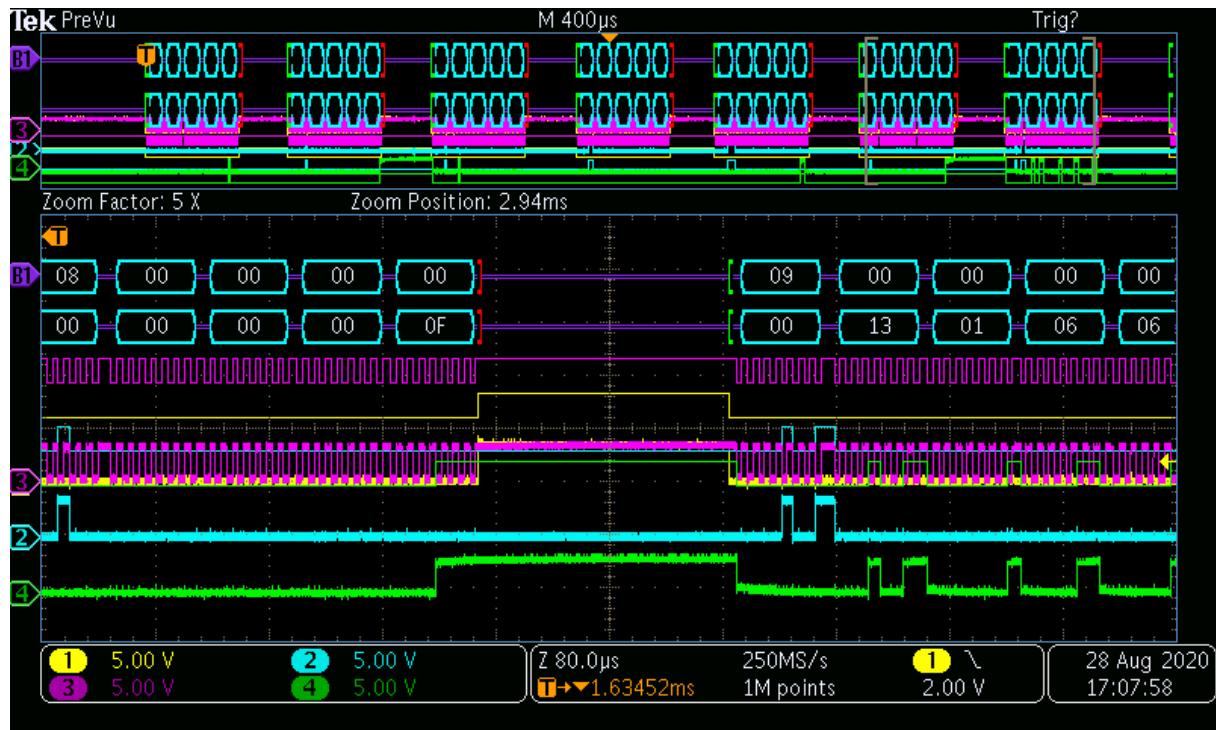


Abbildung F.11: SPI-Übertragung Read.



Abbildung F.12: Event-Table Inbetriebnahme TMC6200.



Abbildung F.13: Event-Table Inbetriebnahme TMC6200.

#### F.4.6 Inbetriebnahme Gate-Ctrl

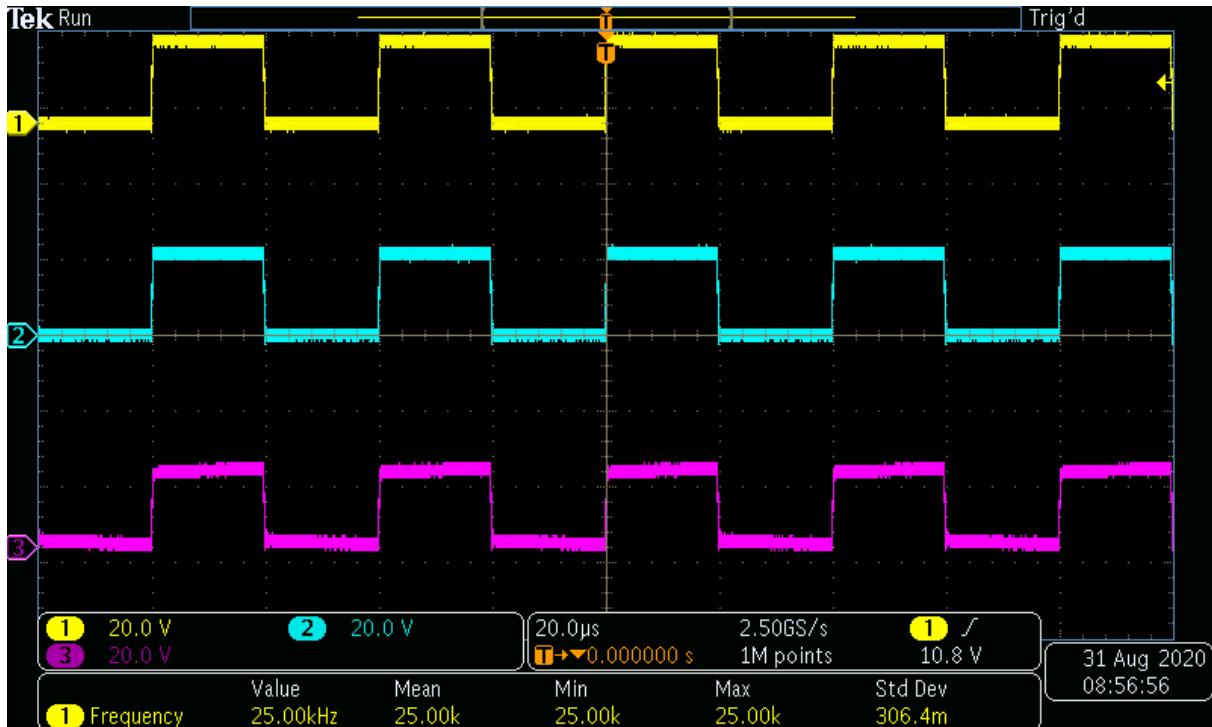


Abbildung F.14: Steuersignale PWM 48V von TMC6200 auf H-Brücke. Gelb = U, Blau = V, Magenta = W

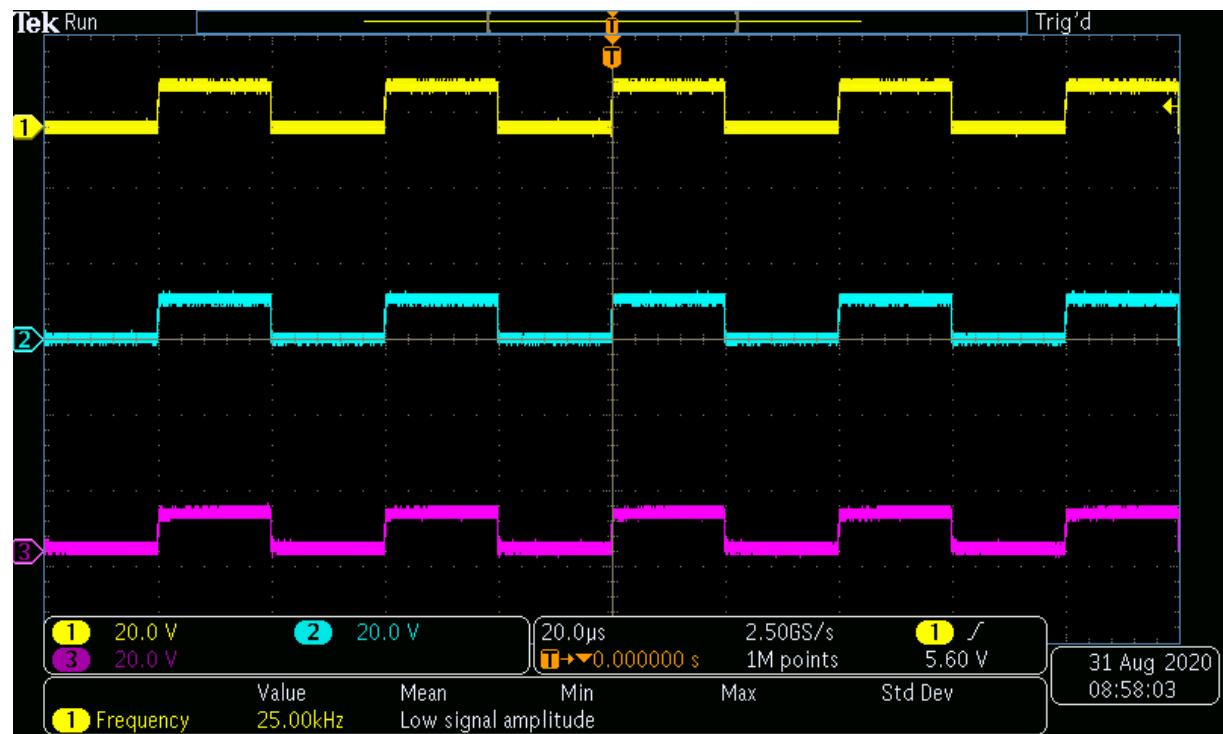


Abbildung F.15: Steuersignale PWM 0V von TMC6200 auf H-Brücke. Gelb = U, Blau = V, Magenta = W

## G H-Brücke

### G.1 Referenzschema

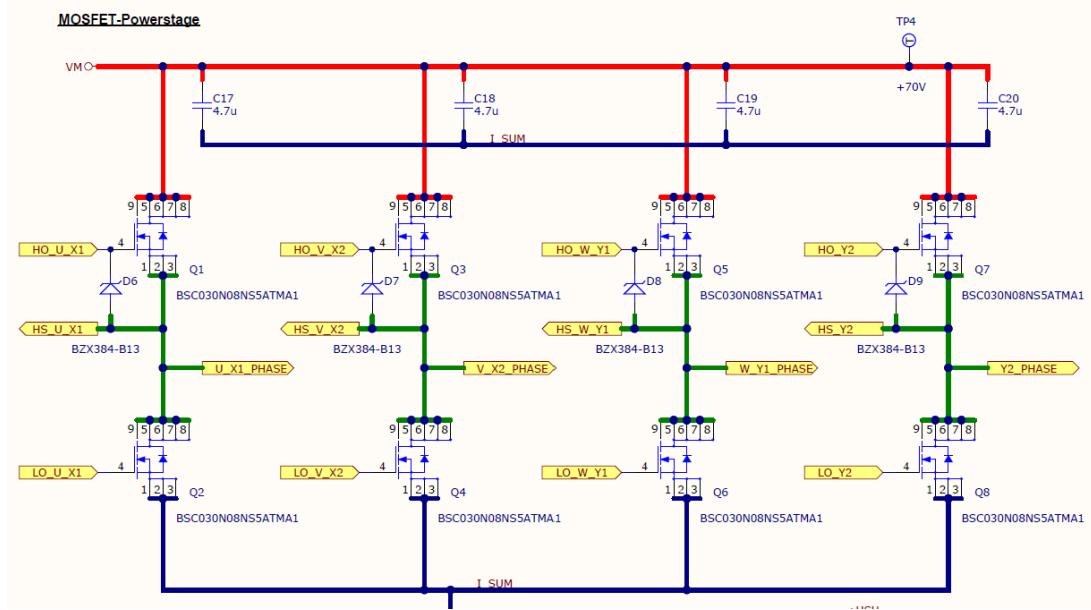


Abbildung G.1: H-Brücke. [5, S.3]

### G.2 Inbetriebnahme

#### G.2.1 Inbetriebnahme Setup

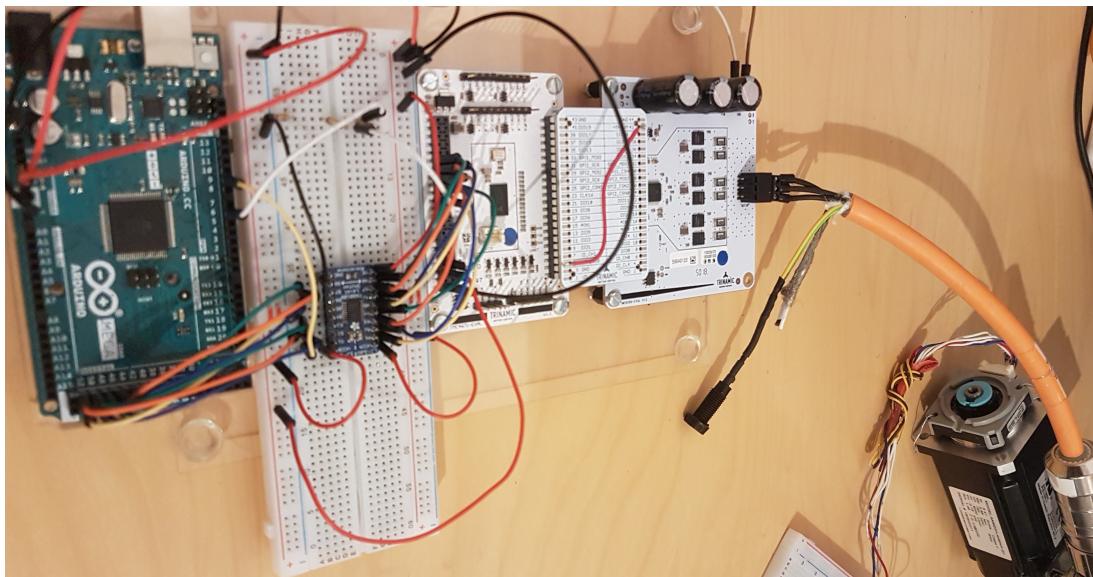
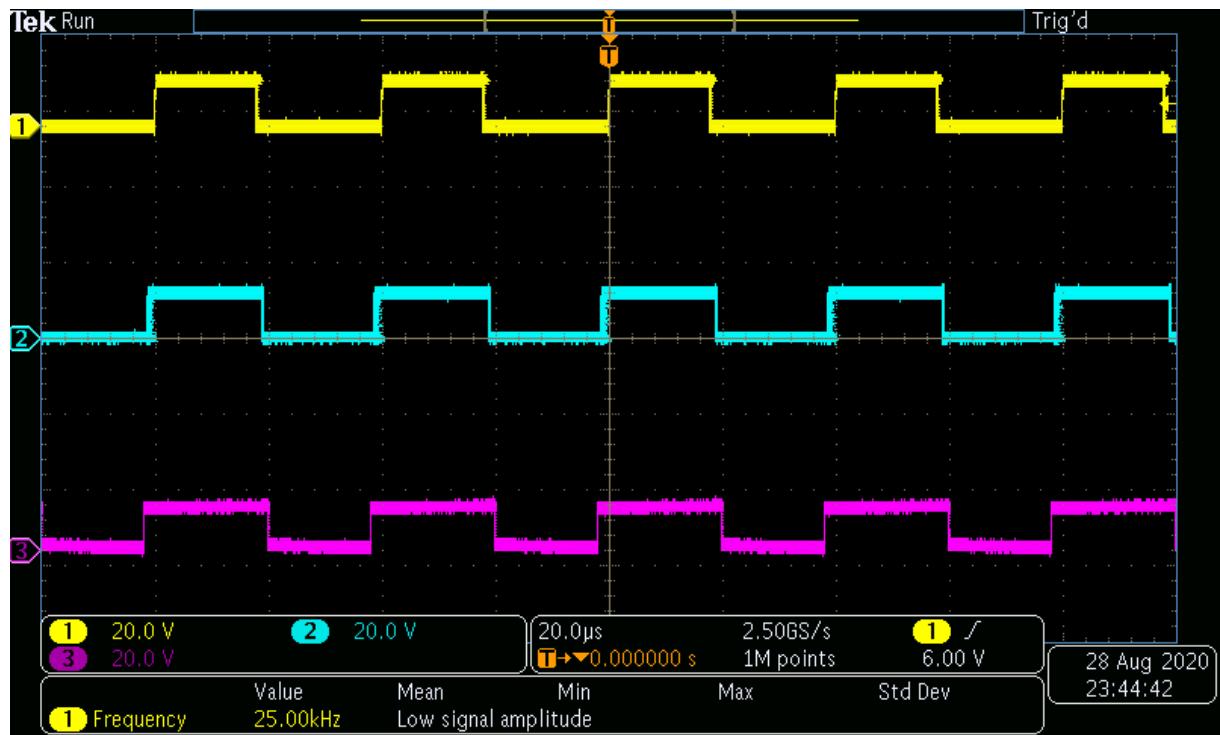


Abbildung G.2: Setup Inbetriebnahme H-Brücke und Motor.

### G.2.2 Inbetriebnahme Schaltsignale



**Abbildung G.3:** Schaltsignale während dem Openloop Testdrive. Gelb = U, Blau = V, Magenta = W

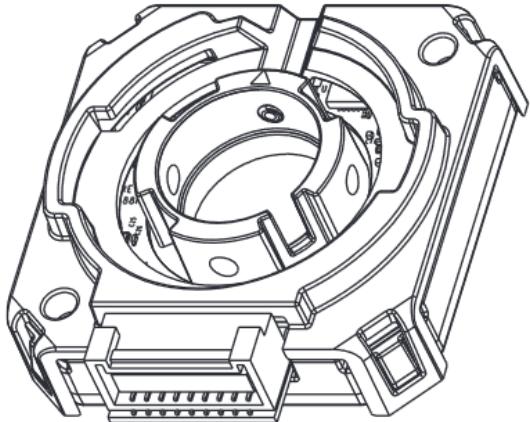
## H CUI AMT332S-V ABN-Encoder

### H.1 Pinout and Interface

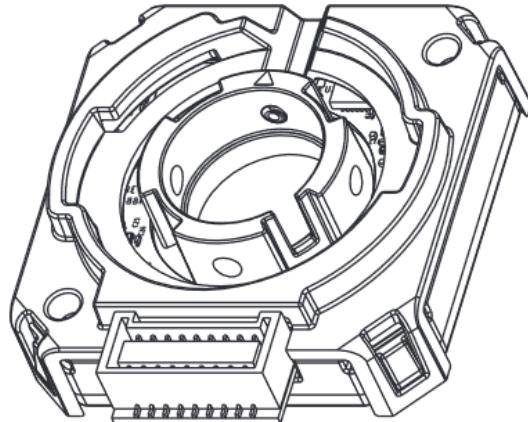
CONNECTOR PINOUT				
#	Function			
#	AMT332S	AMT333S	AMT332D	AMT333D
1	RX_ENC+	RX_ENC+	RX_ENC+	RX_ENC+
2	TX_ENC+	TX_ENC+	TX_ENC+	TX_ENC+
3	U+	U+	U+	U+
4	GND	GND	GND	GND
5	W+	W+	W+	W+
6	+5 V	+5 V	+5 V	+5 V
7	V+	V+	V+	V+
8	A+	A+	A+	A+
9	N/A	N/A	A-	A-
10	B+	B+	B+	B+
11	N/A	N/A	B-	B-
12	Z+	Z+	Z+	Z+
13	N/A	N/A	Z-	Z-
14	MCLR	MCLR	MCLR	MCLR
15	N/A	N/A	W-	W-
16	N/A	N/A	V-	V-
17	N/A	N/A	U-	U-
18*	NOISE GND	NOISE GND	NOISE GND	NOISE GND

\*Pin 18 is not connected internally for standard encoders. Contact CUI for support with high noise applications.

**AMT332**



**AMT333**



Mating Connector:  
JST ZPDR-18V-S

Abbildung H.1: Encoder-Interface Pinout-Table.

## H.2 Inbetriebnahme

### H.2.1 Inbetriebnahme Setup



Abbildung H.2: Encoder-Interface Pinout-Table.

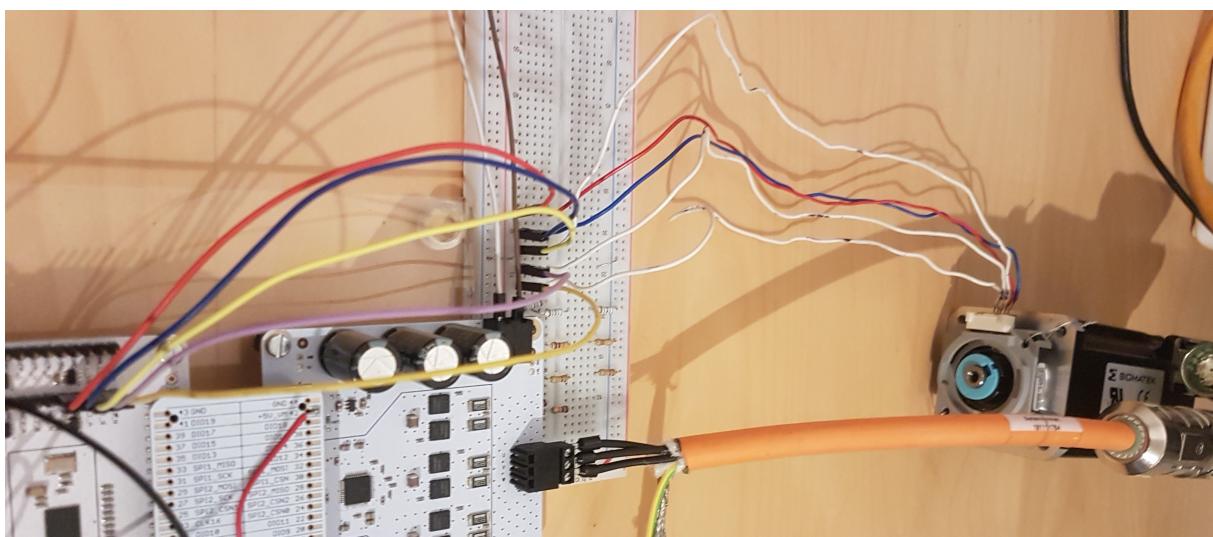


Abbildung H.3: Encoder-Interface Pinout-Table.

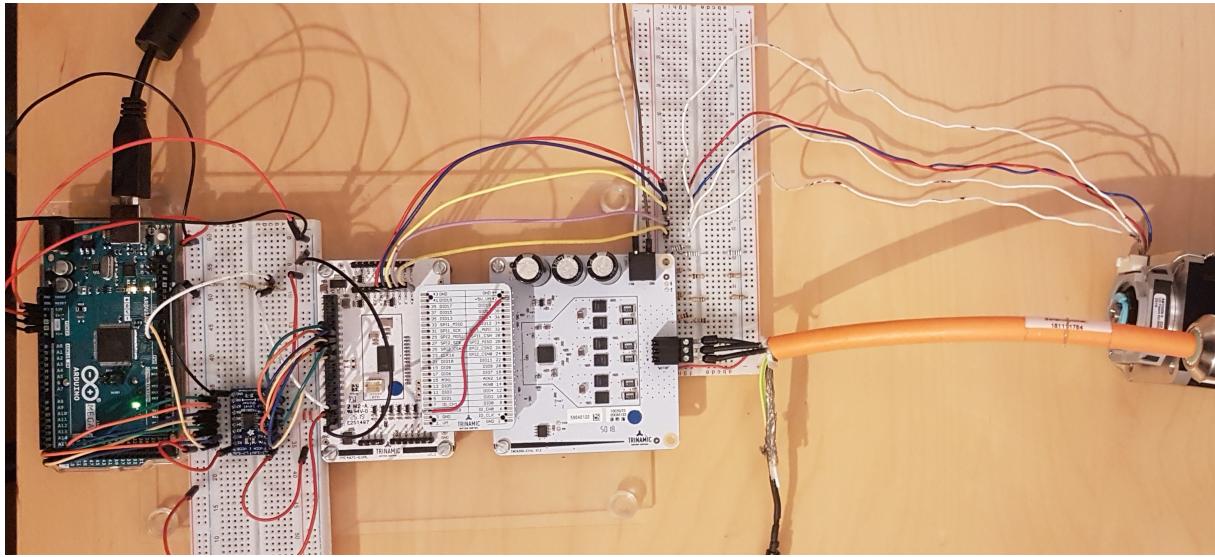


Abbildung H.4: Encoder-Interface Pinout-Table.

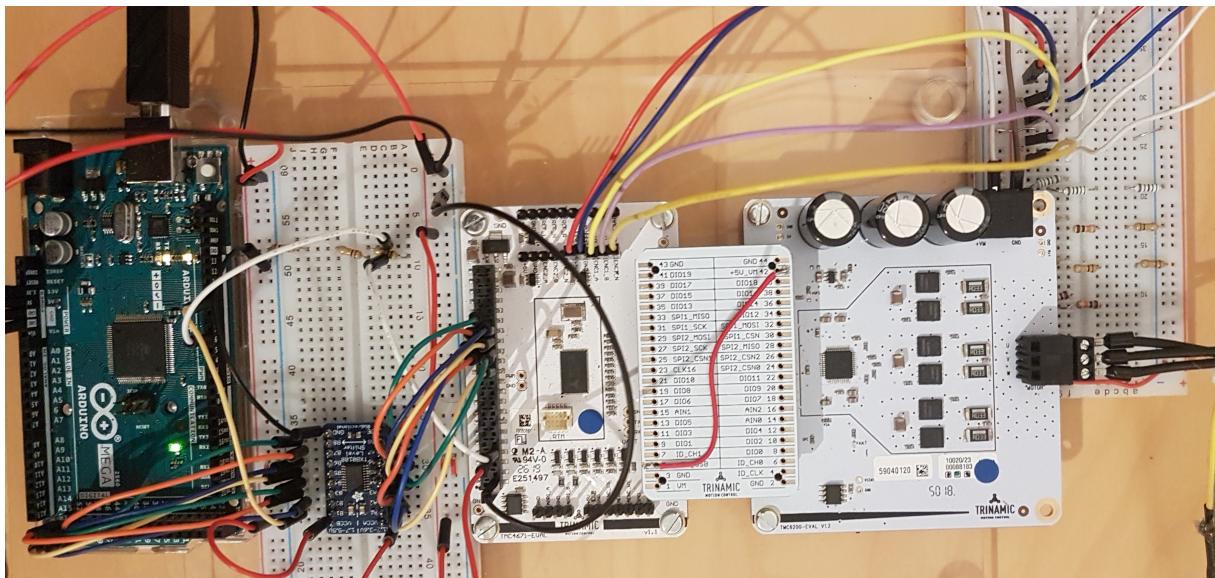


Abbildung H.5: Encoder-Interface Pinout-Table.

### H.2.2 Encoder-Signale

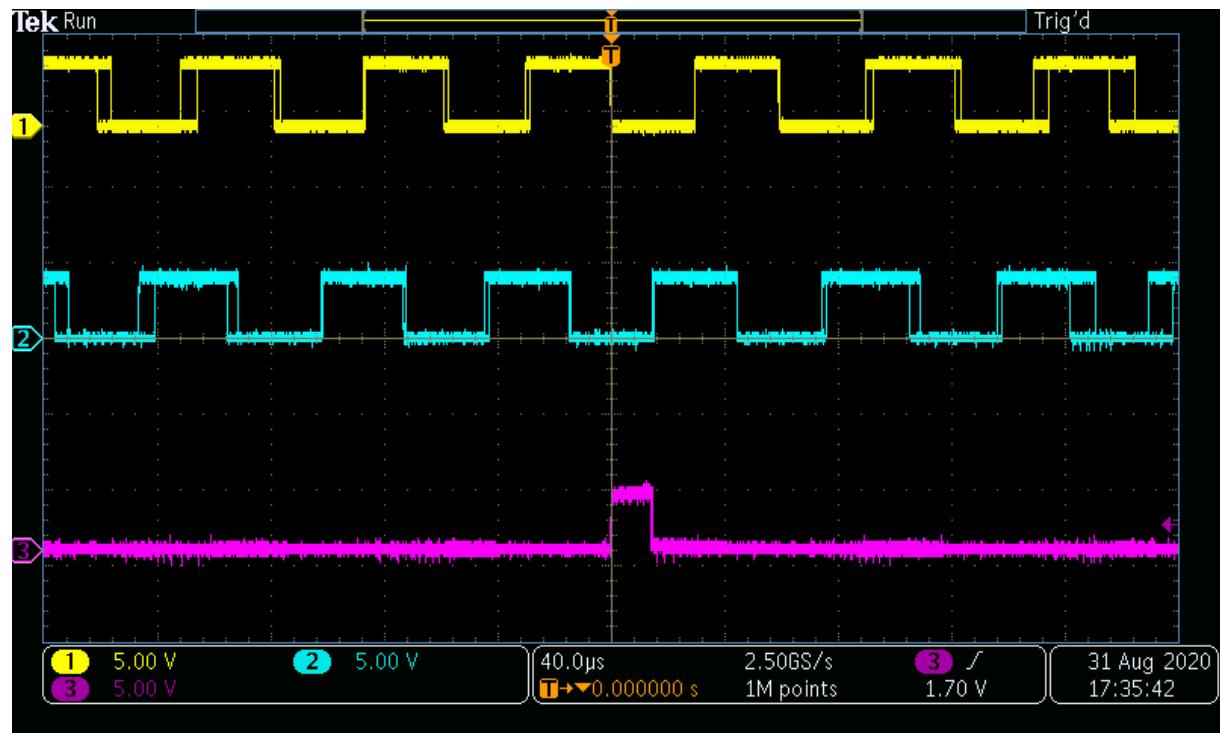
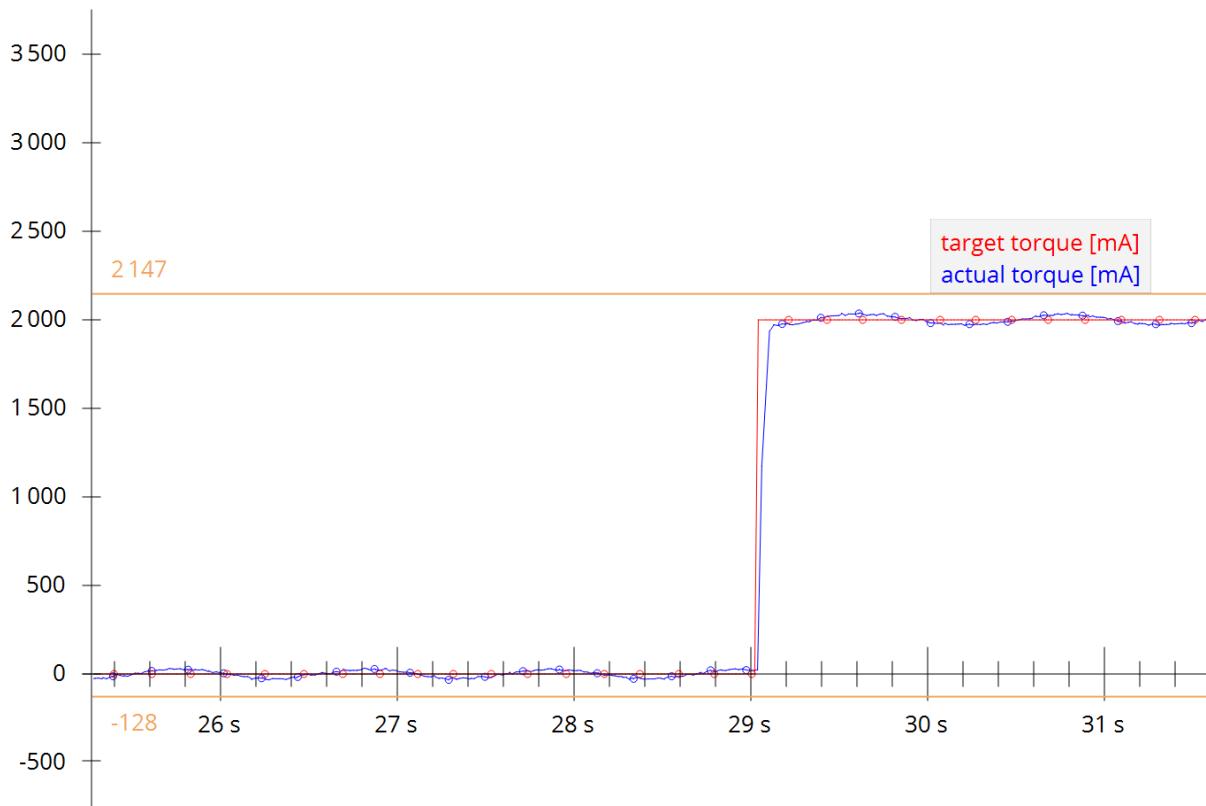


Abbildung H.6: Encoder-Interface Pinout-Table. Gelb = A, Blau = B, N = Magenta

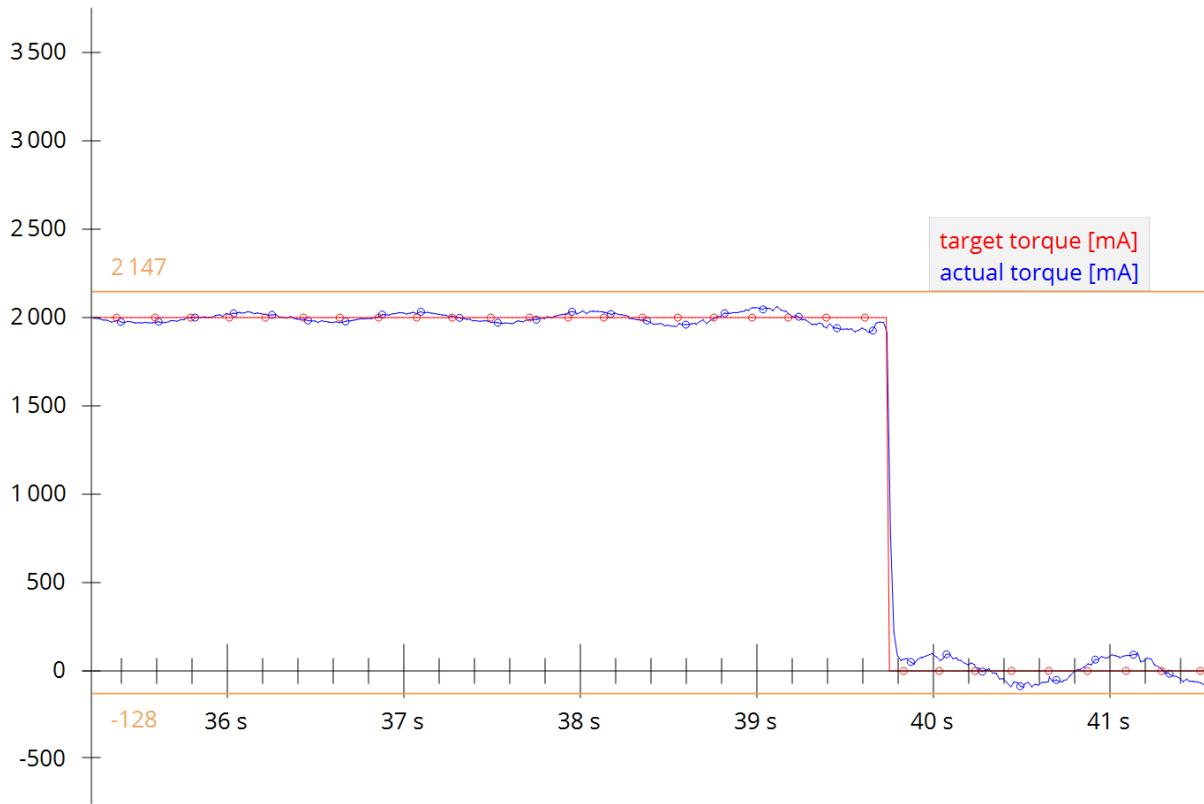
## I PI-Regler

### I.1 TMCL-IDE

#### I.1.1 Drehmoment

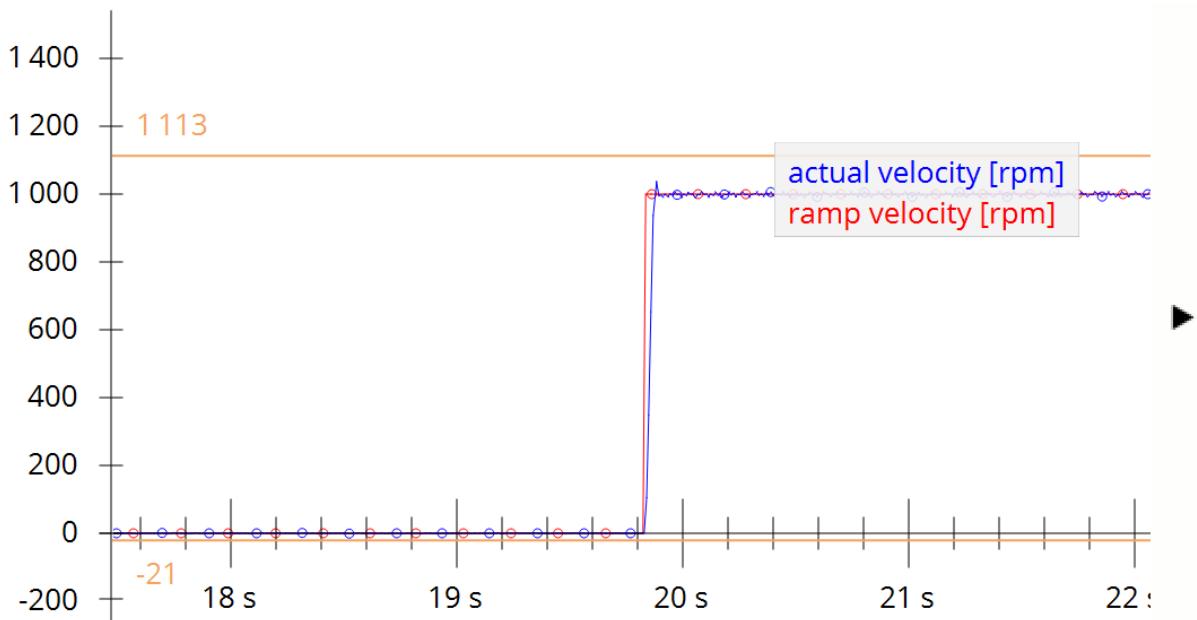


**Abbildung I.1:** Geräte-Manager mit den aufgelisteten USB-UART-Converter (Mikrocontroller und WiFi-Modul).

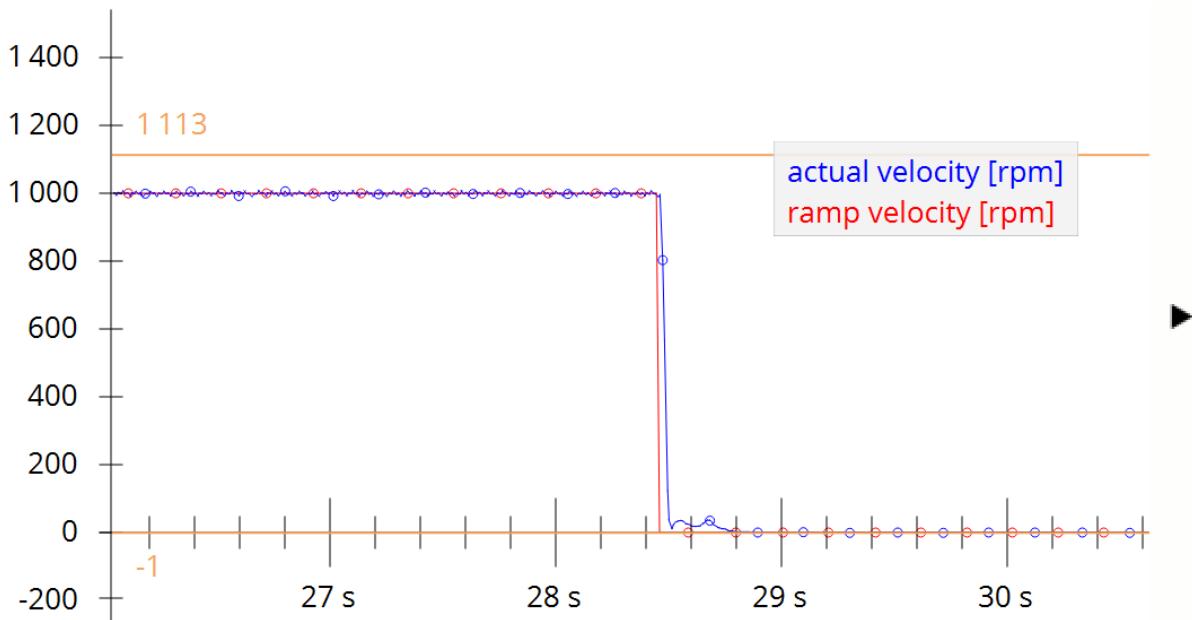


**Abbildung I.2:** Geräte-Manager mit den aufgelisteten USB-UART-Converter (Mikrocontroller und WiFi-Modul).

### I.1.2 Geschwindigkeit

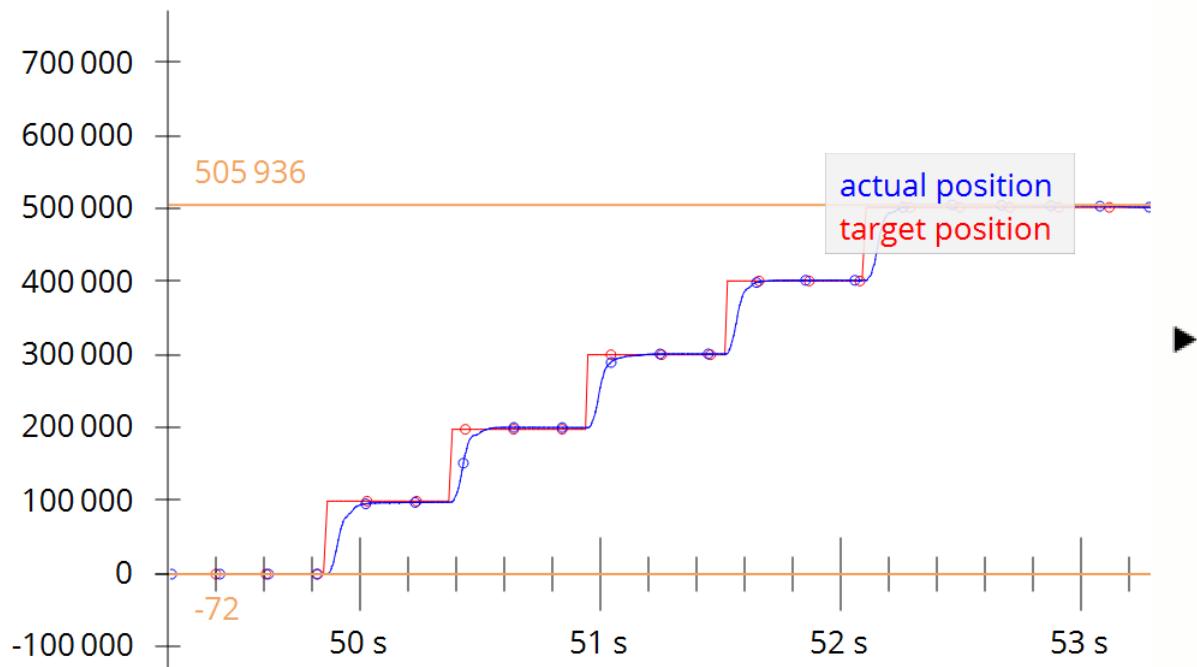


**Abbildung I.3:** Geräte-Manager mit den aufgelisteten USB-UART-Converter (Mikrocontroller und WiFi-Modul).



**Abbildung I.4:** Geräte-Manager mit den aufgelisteten USB-UART-Converter (Mikrocontroller und WiFi-Modul).

### I.1.3 Position



**Abbildung I.5:** Geräte-Manager mit den aufgelisteten USB-UART-Converter (Mikrocontroller und WiFi-Modul).