

# Fachbericht

PROJEKT 6 COCKTAILMASCHINE - TEAM SCHENK & AEBI  
4. Mai 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 . . . . .	2
2.2 Komponentenauswahl . . . . .	2
<b>3 Neue Hardware</b>	<b>3</b>
3.1 Wirelessmodul . . . . .	3
3.2 USB-B . . . . .	4
3.3 RFID . . . . .	6
3.4 Beleuchtung . . . . .	7
<b>4 Printaufbau</b>	<b>8</b>
<b>5 Teilsysteme</b>	<b>8</b>
5.1 Speisungen . . . . .	8
5.1.1 48V Speisung . . . . .	9
5.1.2 12V Speisung . . . . .	9
5.1.3 5V Speisung . . . . .	11
5.1.4 3.3V Speisung . . . . .	12
5.2 Motor . . . . .	14
5.2.1 BLDC . . . . .	15
5.2.2 H-Brücke . . . . .	16
5.2.3 ABN-Encoder . . . . .	17
5.2.4 FOC-Treiber TMC4671 . . . . .	18
5.2.5 Gate-Treiber . . . . .	22
5.3 Flüssigkeitsbeförderung . . . . .	24
5.3.1 Pumpen . . . . .	24
5.3.2 Durchflussmessgeräte . . . . .	24
5.4 Benutzerschnittstellen . . . . .	25
5.4.1 Display . . . . .	25
5.4.2 ESP . . . . .	25
5.4.3 USB-B . . . . .	26
5.4.4 RFID . . . . .	26
5.4.5 Beleuchtung . . . . .	27
5.5 Mikrocontroller . . . . .	28

<b>6 Inbetriebnahme</b>	<b>28</b>
6.1 Speisungen . . . . .	28
6.1.1 12V Speisung . . . . .	28
6.1.2 5V Speisung . . . . .	28
6.1.3 3.3V Speisung . . . . .	28
6.2 Mikrocontroller . . . . .	29
6.3 Benutzerschnittstellen . . . . .	32
6.3.1 Touch-Display . . . . .	32
6.3.2 ESP . . . . .	32
6.3.3 USB-C . . . . .	33
6.3.4 RFID . . . . .	33
6.4 Flüssigkeitsbeförderung . . . . .	33
6.4.1 Pumpen . . . . .	33
6.4.2 Durchflussmessgeräte . . . . .	33
6.5 Beleuchtung . . . . .	33
6.6 Motor . . . . .	33
6.6.1 BLDC und H-Brücke . . . . .	33
6.6.2 ABN-Encoder . . . . .	33
6.6.3 Treiber . . . . .	33
<b>7 Software</b>	<b>34</b>
7.1 Strukturplan . . . . .	35
7.2 Programmflussdiagramm . . . . .	35
<b>8 Evaluation</b>	<b>35</b>
<b>9 Fazit</b>	<b>35</b>
9.1 Zielerreichung . . . . .	35
9.2 Kosten . . . . .	35
<b>10 Schlusswort</b>	<b>35</b>
<b>11 Ehrlichkeitserklärung</b>	<b>35</b>
<b>A TMC4671</b>	<b>36</b>
A.1 Standard-Schaltkreis TMC4671 . . . . .	36
A.2 Blockdiagramm TMC4671 . . . . .	36

<b>B TMC6200</b>	<b>37</b>
B.1 Standard-Schaltkreis TMC6200 . . . . .	37
B.2 Blockdiagramm TMC6200 . . . . .	37
B.3 Dimensionierungstabelle Verstärkungsfaktor Strommessung und Strommesswiderstand . . . . .	38
B.3.1 Dimensionierungstabelle Register und Gate-Vorwiderstand . . . . .	38
<b>C H-Brücke</b>	<b>39</b>
C.1 Referenzschema . . . . .	39
<b>D Mikrocontroller</b>	<b>39</b>
D.1 Brown-out-Detection . . . . .	39
D.2 Full Swing Crystal Oscillator . . . . .	39
D.3 Bootloader-Speicherplatz . . . . .	40
D.4 Memory-Lock Bootloader . . . . .	40
<b>E Atmel Studio</b>	<b>41</b>
E.1 Fuse Bits . . . . .	41
E.2 Lock Bits . . . . .	41
E.3 Einbinden AVRdude und stk500v2 (wiring) . . . . .	42
E.4 Bootloader "Brennen" . . . . .	42

## 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. Andererseits könnte man als Gastgeber selber anbieten Cocktails zu mixen und so den Getränkenachschub zu gewährleisten. Da gibt es jedoch ein grosses Problem. Denn wären wir die Gastgeber, so würden wir nicht den ganzen Abend hinter der Bar stehen wollen, sondern lieber bedenkenlos mitfeiern. Damit genau dies möglich ist haben wir uns in diesem und dem nächsten Projekt (5&6) 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 aussehen soll und aus welchen Teilsystemen diese bestehen wird. Ausserdem werden die einzelnen Teilsysteme genauer unter die Lupe genommen und in einem systemspezifischen Testverfahren evaluiert. Dieses Projekt bietet demnach die Basis des Projekt 6 und soll dieses so gut wie möglich vorbereiten.

## 2 Ausgangslage

### 2.1 Blockschaltbild

### 2.2 Komponentenauswahl

### 3 Neue Hardware

Im folgenden Kapitel werden die Hardware-Teile beschrieben, welche im Projekt 5 noch nicht erarbeitet wurden. Zuerst werden grundlegende Anforderungen und Inhalte beschrieben, welche zur Auswahl der Bauteile geführt hat.

#### 3.1 Wirelessmodul

Über einen Web-Host soll der User die Möglichkeit haben, Getränke auszuwählen und seinem RFID-Chip zuzuordnen, sowie diverse kleinere Einstellungen an der Maschine vorzunehmen. Dazu wird ein WiFi-Modul benötigt.

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 einfacher Leistungsstärker ist. Die genauen Datenvergleiche sind in Tabelle 3.1 ersichtlich.

MCU	Xtensa Single-core 32-bit L106	Xtensa Dual-Core 32-bit LX6
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 3.1:** Vergleich ESP8266 zu ESP32.

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. Dieses ist in Abbildung 3.1 als Wroom dargestellt und in Abbildung 3.2 als Development Kit.

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.



Abbildung 3.1: ESP32-32U Wroom.

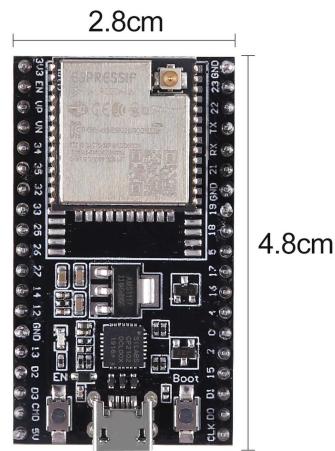


Abbildung 3.2: ESP32-32U DevKit.

## 3.2 USB-B

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 mit einer USB-B-Schnittstelle realisiert wird.

Die USB-B-Schnittstelle benötigt nur zwei Kommunikationsleitungen (D+ und D-), die zu programmierenden Geräte benötigen mehr Anschlüsse um in einen Programmiermodus zu kommen oder mit den Geräten zu kommunizieren. Deswegen benötigt es einen USB-UART-Converter. Hier wurde auf das Flash-Interface von Arduino zurückgegriffen **arduino\_cc\_arduino\_2017**, um die Schaltung für den Mikrocontroller zu planen und auf das Flash-Interface des ESP32 **espressif\_systems\_esp32\_2016**, um die Schaltung für den ESP32 zu planen. Darin ist ersichtlich, dass für den Mikrocontroller und ESP folgende Leitungen nötig sind. Dies wird in Tabelle ?? dargestellt. Die USB-B-seitige Schnittstelle wird an den Computer angeschlossen und muss nicht höher betrachtet werden.

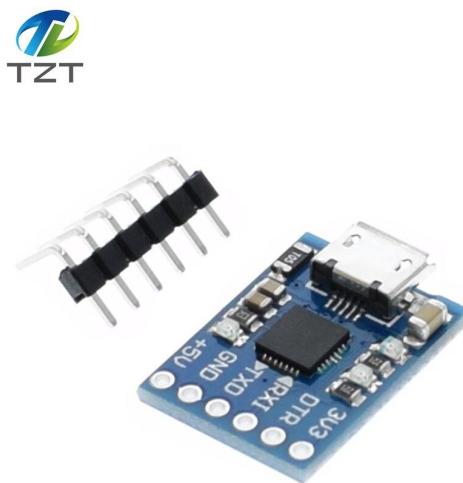
Mikrocontroller				USB-Flash-Device
RX	<==	direkt	====	TX
TX	====	direkt	==>	RX
Reset	<==	Kondensator	====	DTR

Tabelle 3.2: Verbindung zwischen USB und Mikrocontroller.

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

**Tabelle 3.3:** Verbindung zwischen USB und ESP.

Auch hier wurde vom Arduino Uno-Board abgekipft und der selbe Chip verwendet, um die USB-Signale in UART-Signale zu konvertieren. Das vorkommende Bauteil ist der CP2102N. Da dieser Chip ein TQFP-28-Gehäuse hat, könnte es auch schwierigkeiten geben beim Löten. Deswegen wird auch ein Breakout-Board (BOB) mitgeplant, damit es eine Ausweichmöglichkeit gibt falls die Bauteile zu klein sind.

**Abbildung 3.3:** CP2102N-BOB für uC.

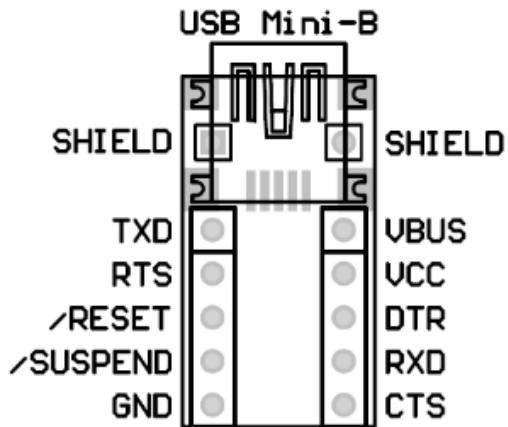


Abbildung 3.4: CP2102N-BOB für ESP.

### 3.3 RFID

Eine vorgesehene Funktion der Maschine ist, dass der User ohne Suchen sein Lieblingsgetränk zubereiten lassen kann. Dafür wird auf ein System zurückgegriffen, was öfters für Zutrittskontrollen oder Ähnliches verwendet wird. Nämlich RFID<sup>1</sup>.

Um schnell ein funktionierendes Teilsystem testen zu können, soll ein Breakout-Board verwendet werden. Eine Recherche hat ergeben, dass es nicht allzu viele Produkte gibt. Ein Eingenzungskriterium war, dass es es für den Ausgewählten IC eine vorgeschriebene Library gibt, welche sich mit Arduino bewährt hat. Der Chip, welcher diese Anforderungen erfüllt, ist der **Mifare MFRC522**. Er ist in Abbildung 3.5 abgebildet. Dieses Brakeout-Board kann ausserdem an der Verschalung der Maschine angeschraubt werden. Somit kann er sich an einer Stelle befinden, welche weiter weg von der Hauptsplatine ist. Das Modul arbeitet auf 13.56MHz und gilt somit als kurzwelliges System (HF).

Der MFRC522 ist der Reader im RFID-System. Er kann Daten lesen und ggf. auch schreiben. Er erzeugt ein hochfrequentes, elektromagnetisches Wechselsignal mit einer Frequenz von 13.56MHz. Der RFID-Tag wird von der Energie des Wechselsignals gespiesen. Durch Kurzschliessen der Tag-Antenne wird ein Teil der Energie des vom Reader ausgehenden Wechselfeldes verbraucht. Diese Energiedifferenz kann ein Reader detektieren. Die Reichweite für ein solches System beträgt weniger als 10cm.

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.

---

<sup>1</sup>Radio Frequency IDentification



Abbildung 3.5: MFRC522.

### 3.4 Beleuchtung

Damit die Maschine optisch etwas hergibt, wurde entschieden die ganze Maschinerie mit LED-Streifen zu verzieren. Dazu ist einerseits das LED-Band von nötigen 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 3.6 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 auf dem Bild.

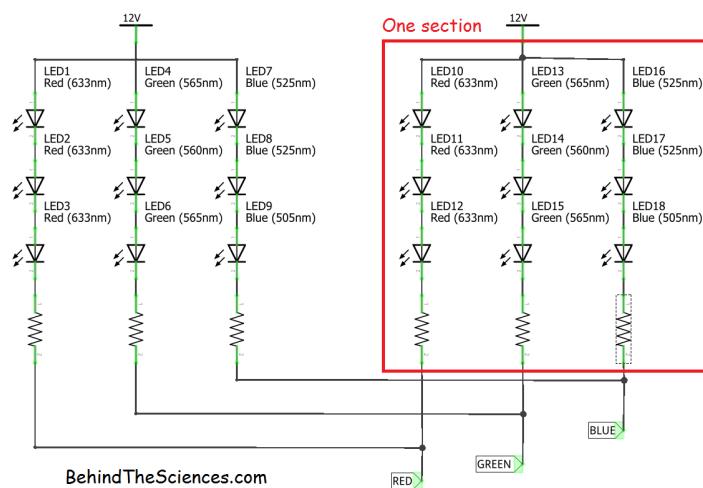


Abbildung 3.6: LED Beispiel.

## 4 Printaufbau

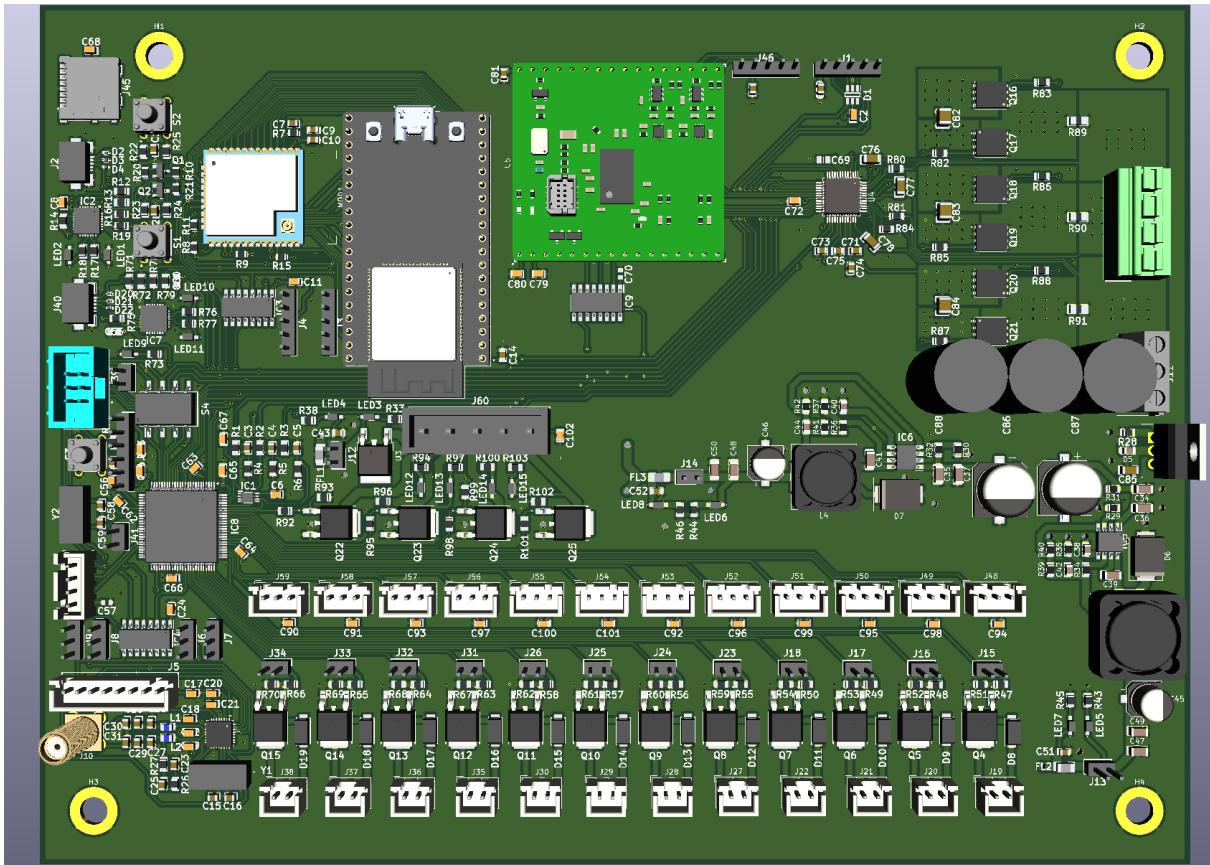


Abbildung 4.1: Print 3D

## 5 Teilsysteme

Da der Partymixer aus vielen kleineren und grösseren Teilsystemen besteht, werden diese in diesem Kapitel einzeln aufgelistet und im Detail angeschaut. Es wird dabei bei jedem Teilsystem auf drei Punkte eingegangen, die Problemstellung (Was ist der Zweck der Teilschaltung und weshalb wird sie benötigt?), das Schema und der Funktionsbeschrieb der Schaltung. Das komplette Schema kann in **Anhang Schema** begutachtet werden.

### 5.1 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.

### 5.1.1 48V Speisung

Der Motor wird mit einer Spannung von 48V betrieben. Dies ist zugleich auch die höchste verwendete Speisespannung. Um diese Speisung gewährleisten zu können, wird ein fertiges Netzteil 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 5.1 begutachtet werden.



**Abbildung 5.1:** 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.

### 5.1.2 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 5.2 betrachtet werden.

### Schema

Das Schema in Abbildung 5.2 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 Överbetriebsschutzimplementiert. Vor dem Ausgang der Schaltung ist dann erneut eine Filterstufe implementiert, welche das Ausgangssignal glättet.

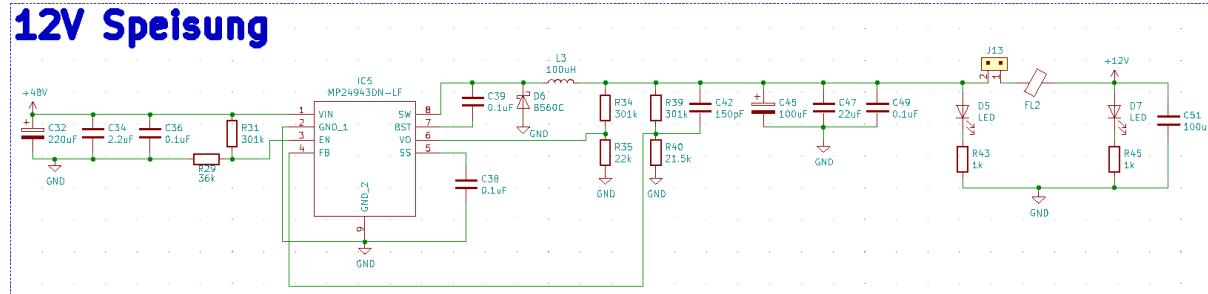


Abbildung 5.2: Schema der 12V Speisung

### Funktionsbeschrieb der Schaltung

Um den MP24943DN-LF auf aktiv zu setzen, wird eine minimale Spannung von 1.8V vorausgesetzt. Fällt diese unter 0.4V, so wird dieser 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 Eingangsspegel 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 5.1.

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

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

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

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

Bei einem Widerstandsverhältnis von  $R_{34}=301\text{k}\Omega$  &  $R_{35}=22\text{k}\Omega$  entspricht dies einer Überspannungsschutzwelle von 13.21V.

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

$$L_3 = \frac{V_{out} \cdot (V_{in} - V_{out})}{V_{in} \cdot \Delta I_L \cdot f_{osc}} \quad (5.3)$$

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.

### 5.1.3 5V Speisung

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

#### Schema

Das Schema in Abbildung 5.3 kann wie bei der 12V Speisung gemäss Kapitel 5.1.2 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 Overvoltage-Protection eingestellt. Vor dem Ausgang der Schaltung ist dann erneut eine Filterstufe implementiert, welche das Ausgangssignal glättet.

#### Funktionsbeschrieb der Schaltung

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

Das Widerstandsverhältnis von R41 & R42, welches die Ausgangsspannung definiert, wurde gemäss Formel 5.1 berechnet. Somit ergeben sich für  $R_{41}=301\text{k}\Omega$  und für  $R_{42}=57.6\text{k}\Omega$ , Was einer Ausgangsspannung von 4.98V entspricht.

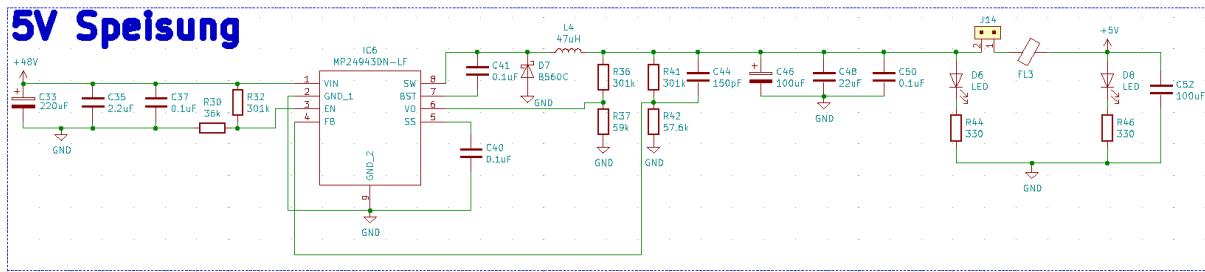


Abbildung 5.3: Schema der 5V Speisung

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  $R_{36}=301\text{k}\Omega$  und  $R_{37}=53\text{k}\Omega$  gewählt. Gemäss Formel 5.2 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 5.3 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.

#### 5.1.4 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 Schema kann in Abbildung 5.4 begutachtet werden.

#### 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.

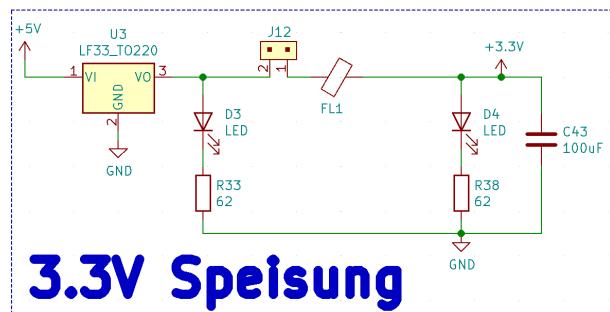
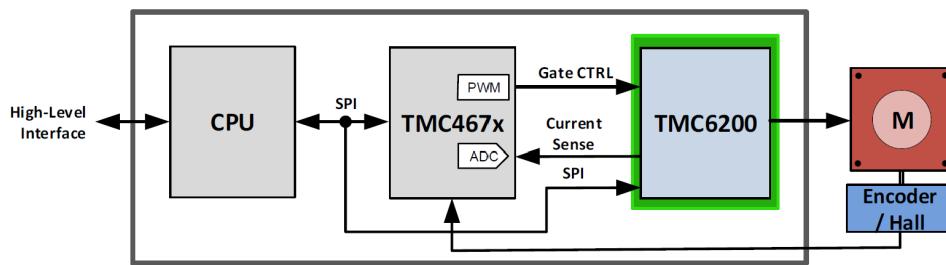


Abbildung 5.4: Schema der 3.3V Speisung

Wie bei der 12V Speisung in Kapitel 5.1.2 und der 5V Speisung in 5.1.3 wurde ein Jumper, sowie zwei LED's zu Testzwecken implementiert und auch hier Filtert ein Ferrit FL1 letzte Störungen heraus.

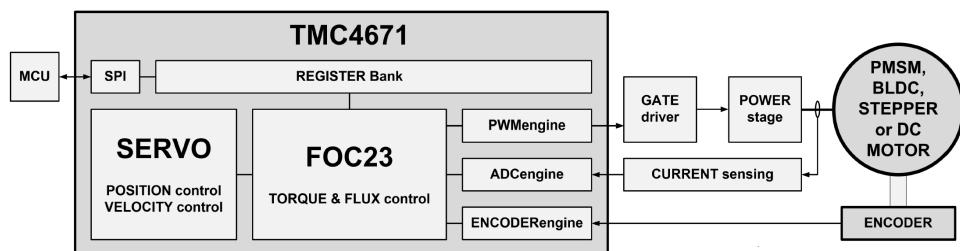
## 5.2 Motor

Um ein Glas während der Zubereitung hin- und her zu bewegen, wird eine Antriebsgruppe benötigt. Die Auswahl der Motorengruppe wurde mehr oder weniger vom Dozenten vorgegeben. 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. Das benötigte Feedback wird vom ABN-Encoder AMT33 von CUI devices verwendet. Auf die erwähnten Komponenten wird im Folgenden eingegangen. Abbildung 5.5 zeigt, wie die ICs zusammenhängen.



**Abbildung 5.5:** Blockschaltbild Konfiguration IC's mit BLDC und Encoder

Es wurde darauf geachtet, dass der Aufbau des Prints so gut wie möglich dem Testaufbau entspricht. Im Folgenden wird deshalb ein detaillierteres Blockschaltbild gezeigt, welches den Aufbau eher nach Funktionen beschreibt, siehe dazu Abbildung 5.6. Darin eingezeichnet sind die verwendeten Referenceboards.



**Abbildung 5.6:** Blockschaltbild Konfiguration Motorengruppe nach Funktionen.

TMC4671	= Trinamic TMC4671	FOC-Treiber
GATE driver	= Trinamic TMC6200	Gate-Treiber
POWER stage	= Trinamic UPS 10A70V	H-Brücke
Current sensing	= UPS 10A70V ==> TMC6200	Messung Phasenströme
PMSM/BLDC	= Sigmatec AKM22h	BLDC
ENCODER	= CUI devices ATS33	ABN-Encoder

### 5.2.1 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.

Ein Nachteil von BLDC-Motoren ist, dass bei Drehgeschwindigkeiten über Nenndrehzahl nur mit einer geeigneten Regelung der Statorwicklungen in Feldschwächung gefahren werden kann, anstelle dass der Erregerstrom verringert wird. Ansonsten zeichnet sich ein BLDC durch ein besseres Leistungs-/Gewichtsverhältnis aus als herkömmliche Motoren.

#### 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.

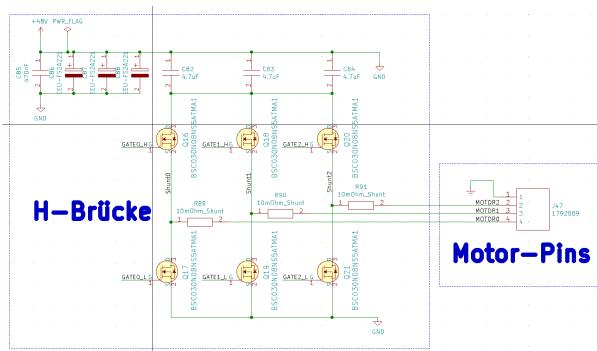


Abbildung 5.7: H-Brücke und Motor-Pins für BLDC

**Funktionsbeschrieb** Hier gibt es nicht viel zu beschreiben, der Anschluss des Motors ist selbsterklärend. Lediglich die Phasenfolge ist zu beachten.

Der Motor an sich lässt sich wie folgt beschreiben:

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]

### 5.2.2 H-Brücke

Das Bindeglied zwischen Kraft (Motor) und Steuersignalen wird von der H-Brücke gebildet. Durch Laden-/Entladen der MOSFET-Gates wird eine Spannung an einer Spule angelegt oder weggenommen. Wie sich das Ein- und Ausschalten der einzelnen Phasenbrücken auf den Motor auswirkt, wird zu einem späteren Zeitpunkt erklärt.

#### Schaltungsaufbau

Für den Aufbau und die Dimensionierung wurde das Referenzschema des Evaluationsboard verwendet, mit dem getestet wurde (Trinamic UPS 10A70V). Es ist aufgezeigt im Anhang Kapitel C Abbildung C.1. Die Dimensionierung der Shunts wird in Kapitel 5.2.5 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 Stütz- und Filterkondensatoren bestückt, um eine saubere Netzspannung zu gewährleisten.

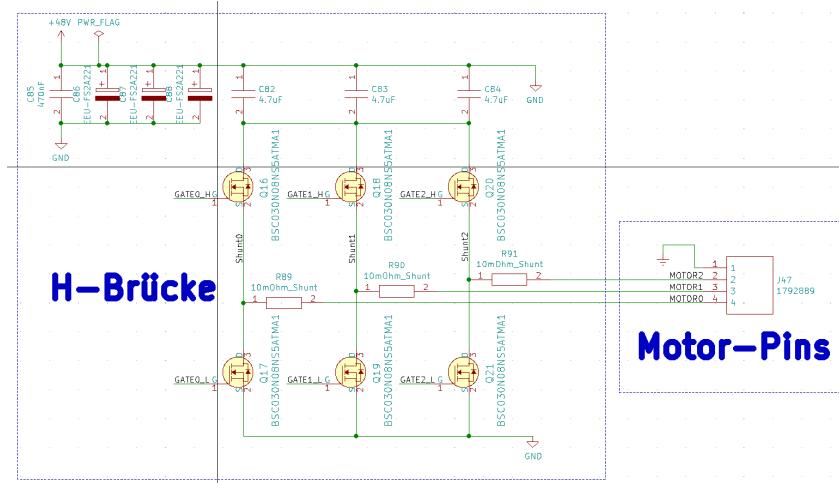


Abbildung 5.8: H-Brücke.

Referenzierung  
auf mög-  
liches  
Kapitel  
FOC-  
Steuerung  
mit BLDC-  
Motor /  
Regleraus-  
legung

### Funktionsbeschrieb

Wie erwähnt werden die Eingänge gefiltert. Dies geschieht mit den Kondensatoren C85-C88, um die Eingangsspannung konstant zu halten und hochfrequente nicht ins Netz gespiesen werden. Bei C86-C88 handelt sich um Low-ESR Stützkondensatoren, jeweils ein Kondensator pro Phase. Sie befinden sich nahe am Spannungseingang und der H-Brücke.

Die Kondensatoren C82-C84 sind Kondensatoren, welche direkt zwischen Ein- und Ausgang eines H-Brücken-Strangs platziert wurden. Sie dienen auch zu Filterzwecken.

Die MOSFETS Q16-Q21 bilden die eigentliche H-Brücke. Sie Schalten den Leistungsfluss gemäss den Ansteuersignalen. Sie können 100A Nennstrom schalten und 100V standhalten. Die Gate-Source-Spannung beträgt  $\pm 20\text{V}$ .

#### 5.2.3 ABN-Encoder

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

**Schaltungsaufbau** Der Schaltungsaufbau gestaltet sich an den Pins eher einfach. Nebst der 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 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.

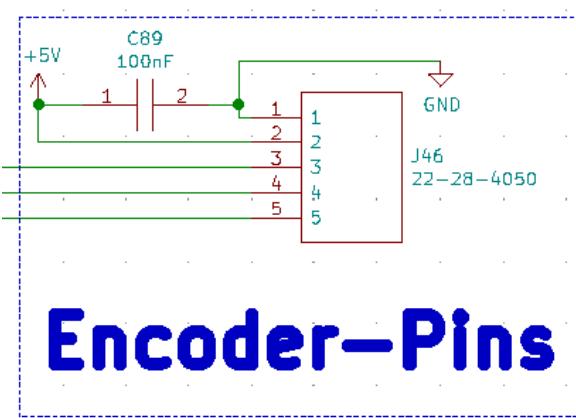


Abbildung 5.9: Schema ABN-Encoder.

### 5.2.4 FOC-Treiber TMC4671

Wie dem Grobkonzept in Kapitel 2.1 entnommen werden kann, wird für die Ansteuerung des BLDC-Motors eine Steuerlogik (Motor Control Unit) benötigt. Dabei handelt es sich um den 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. Die vom FOC-Treiber ausgehenden PWM-Leitungen gehen direkt auf den Gate-Treiber, welcher dann wiederum die MOSFETs ansteuert. Auf den FOC-Treiber und weitere für diesen Schaltkreis benötigte Komponenten wird im folgenden eingegangen. Da in diesem Projekt die Gewichtung auf dem Aufbau der Cocktailmachine liegt, wird ein Breakout-Board (BOB) verwendet, welche die Aufgabe der Kommuntierung übernimmt. Auf dem Board sind die Eingänge des FOC-Treibers schon geschützt mit Filtern und Überspannungsdioden. Somit entfällt eine Dimensionierung der Filter und Schutzschaltungen. Es wird jedoch die jeweilige Funktion der Schaltung beschrieben.

#### Schaltungsaufbau

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

Im Anhang A zeigt Abbildung A.2 anhand eines Blockdiagramms, welche Funktionen im Treiber vorhanden sind und wie diese zusammenhängen. Im wesentlichen kann daraus entnommen werden, dass der TMC4671 aus einer FOC-Logik<sup>2</sup>, einer Servo-Logik, einem SPI-Interface und diversen Engines (PWM, ADC, Encoder) besteht. In Abbildung A.1 wird eine Standard-Anwendungsschaltung gezeigt, worin erkennbar ist, dass der Treiber sehr universell aufgebaut ist und für mehrere Motorentypen verwendet werden kann.

In Abbildung 5.10 ist erkennbar, welche Pins für den Cocktailmixer verwendet werden und welche Verbindungsleitungen zum Treiber führen. 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 PWM-Signal (PWM). Bei den SPI-Leitungen ist zu sehen, dass die Leitungen zuerst über einen Level-Shifter müssen, da der Treiber nur 3.3V verarbeiten kann ohne dabei kaputt zu gehen. Zu erkennen an der zusätzlichen Leitungsbezeichnung LV (Lower Voltage). Die Leitung SPI1 geht vom Treiber zum Mikrocontroller und muss deshalb nicht geshiftet werden.

---

<sup>2</sup>FOC= Field Oriented Control

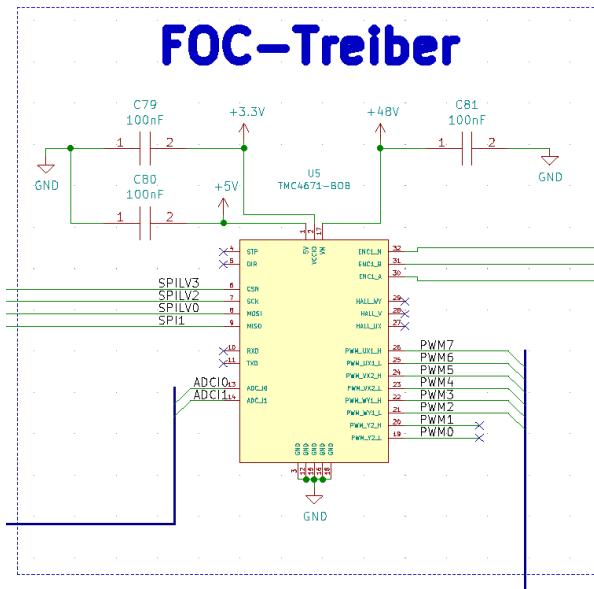


Abbildung 5.10: Schema FOC-Treiber.

### Funktionsbeschrieb

Im Folgenden werden kurz die Teilsysteme auf dem TMC4671-BOB beschrieben, welche für den Cocktailmixer gebraucht werden. Dazu wird auf das Datenblatt von Trinamic TMC4671 zurückgegriffen.

**Kommunikation Input (SPI)** Der Kommunikationseingang ist nicht speziell geschützt. Hier ist für das eigene Layout darauf zu achten, dass die Eingänge nicht mit mehr als 3.3V belastet werden. Dies wird mit einem Level-Shifter zwischen Mikrocontroller und TMC4672 gewährleistet.

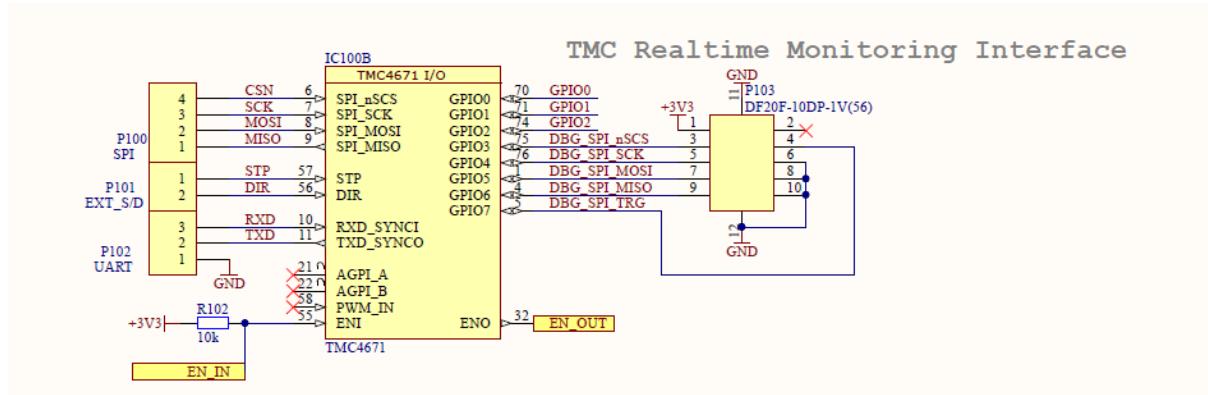


Abbildung 5.11: SPI-Input des TMC4671-BOB.

**Ströme Input (ADC)** Der Eingang zur Strommessung ist mit einem Tiefpass geschützt. Dieser hat die Zeitkonstante:

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

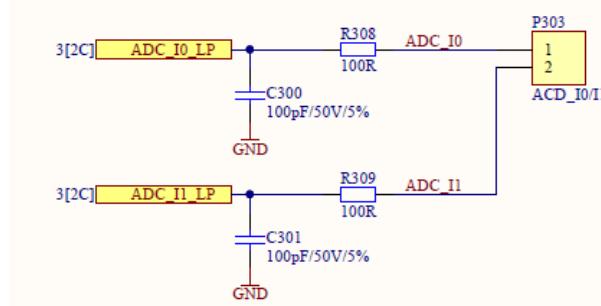


Abbildung 5.12: Phasenstroeme-Input des TMC4671-BOB.

**Encoder Input (ENC)** Der Encoder-Input ist ziemlich clever aufgebaut. Um Überspannungen zu verhindern, wird ein Schmitt-Trigger mit eingebautem Level-Shifter verwendet. Der Level-Shifter verhindert dabei ein Ansteigen der Eingangsspannung über dessen Eingangsspannung. 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:

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

Fällt die Spannung über dem Kondensator während dem Entladen unter einen bestimmten Wert, so wird auch der Schmitt-Trigger aktiv und springt auf 0V. Sobald der Eingang nicht mehr vom Encoder auf 0V gezogen wird, so lädt sich der Kondensator über die Widerstände R200-R202 sowie R206,R208,R210. Somit ergibt sich eine längere Zeitkonstante:

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

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).

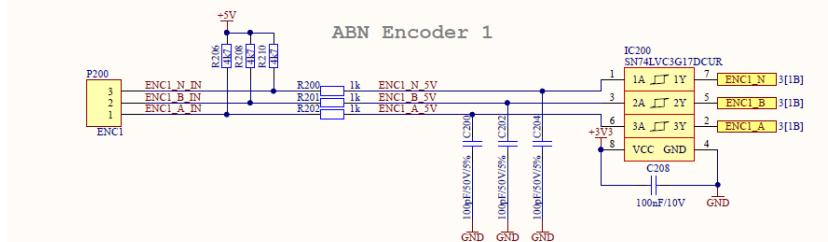


Abbildung 5.13: ABN-Encoder-Input des TMC4671-BOB.

**Motorspannung Input (48V)** Die Motorspannung ist wichtig, um einen Kommuntierungs-vorgang zu berechnen. Wichtiger als die Zeitkonstante ist hier ein Spannungsteiler, welcher die Motorspannung auf unter 3.3V bringt. Dazu wird folgende Formel angewendet:

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

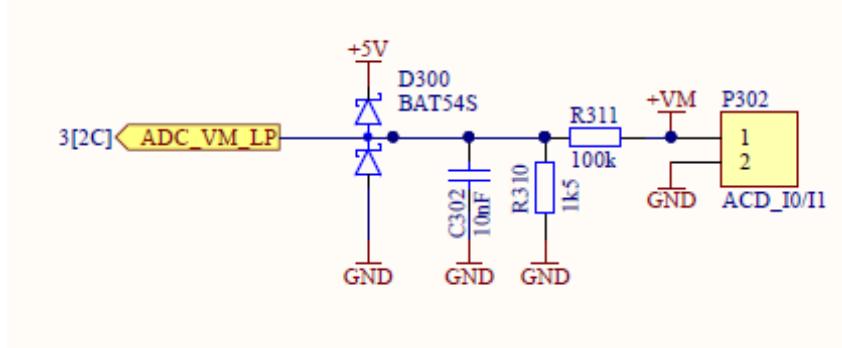


Abbildung 5.14: Motorspannung-Input des TMC4671-BOB.

**Steuersignale Output (PWM)** Die Ausgangssignale für den Gate-Treiber gehen direkt auf die Header-Pins des MC4671-BOB. Sie werden nicht geschützt.

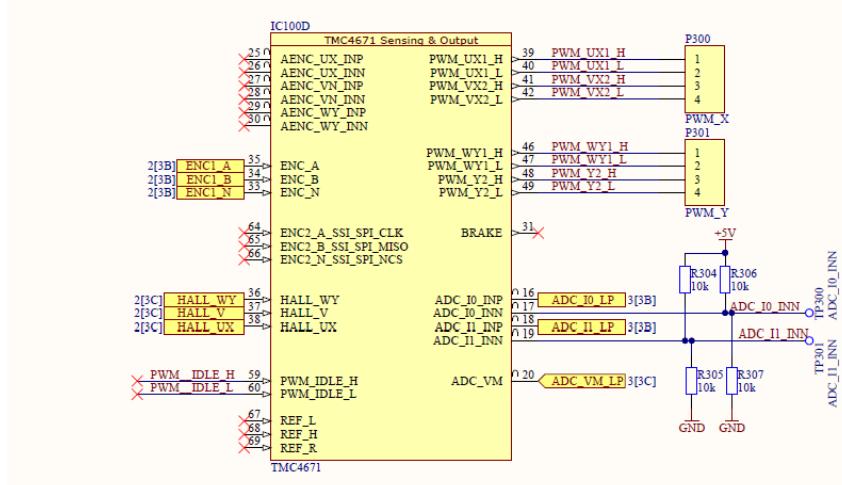


Abbildung 5.15: PWM-Output des TMC4671-BOB.

### 5.2.5 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. Damit die Umschaltverluste und daraus folgende Abwärme verhindert werden kann, ist es vorteilhaft die Gates so schnell wie möglich laden und entladen zu können. Da kommt der Gate-Treiber ins Spiel. Dieser ladet und entlädt das Gate schnell genug und stellt die dazu benötigte Energie für das Gate zur Verfügung.

#### Schaltungsaufbau

Damit ein komplizierter Aufbau vermieden werden kann und einige Zusatzfunktionen wie Strommessung, Messverstärkung etc. angeboten 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 B Abbildung B.1 und Abbildung B.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.

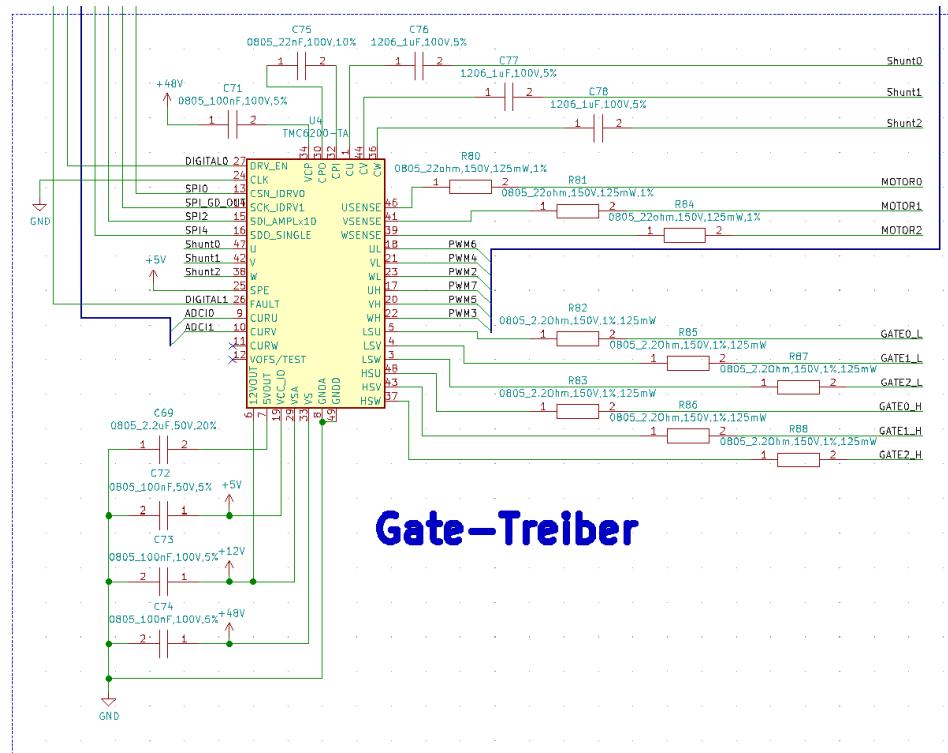


Abbildung 5.16: Schema Gate-Treiber.

Kondensatoren  
im Text  
erwähnen

**Shunt-Widerstände  $R_{Sense}$**  Die Shunt-Widerstände ermöglichen einen geschlossenen Loop für die Stromsteuerung, so wie es für eine feldorientierte Steuerung (FOC) benötigt wird. Dazu wird solch ein Shunt in Serie mit der Spule des BLDC-Motors geschaltet. 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 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. Aus Qualitätsgründen wird jedoch von 10A ausgegangen, was die Zuverlässigkeit und Lebensdauer erhöhen soll, zudem wird so die Verlustleistung des Shunts kleiner gehalten als bei einer Grösse ober-/unterhalb. Eine von Trinamic erstellte Tabelle (siehe Anhang B Tabelle B.3) ergibt folgende Grösse für die Shunt-Widerstände: [trinamic\\_tmc6200\\_datasheet\\_2013](#)

$$R_{Sense} = 10\text{m}\Omega$$

$$\text{Verstärkungsfaktor} = 10x$$

**Gate Vorwiderstände  $R_{Gate}$**  Da das Gate ein kapazitives Verhalten zeigt, ist der Strom, welcher zu Beginn ins Gate fliest, sehr hoch. Der Gate-Vorwiderstand begrenzt diesen, um das Gate und den Pin des TMC6200 vor Überströmen zu schützen.

Die Dimensionierung der Gate-Widerständen sollte grundsätzlich an die MOSFET Gate-Drain-Ladung (Miller charge) angelehnt werden, um an angemessene Anstiegszeiten während des Schaltens zu erreichen. Auch in diesem Falle wurden von Trinamic schon einige Parameter ermittelt und im Anhang B in Tabelle B.4 dargestellt. Die Gate-Ladung des ausgewählten MOSFET's beträgt 61nC. Mit dem programmierbaren Register DRV\_STRENGTH wird der Strom ins Gate angepasst. [trinamic\\_tmc6200\\_datasheet\\_2013](#)

$$R_{Gate} = R_{Gate} \leq 2.5\Omega$$

$$\text{DRV\_STRENGTH} = 1 \text{ bis } 3$$

**Schutzwiderstände Messeingang  $R_{Protect}$**  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 Dimensionierung dieses Widerstands wird im Datenblatt mit einem Widerstandswert zwischen  $10\Omega$  und  $22\Omega$  angegeben. [trinamic\\_tmc6200\\_datasheet\\_2013](#)

$$R_P = 10\Omega \text{ bis } 22\Omega$$

**Bootstrap Kondensatoren  $C_{Bootstrap}$**  Die Bootstrap-Kondensator werden benutzt, wenn eine Änderung des Potentials innert kurzer Zeit benötigt wird.

Die Dimensionierung dieses Widerstands wird im Datenblatt mit einem Kapazitätswert zwischen  $470\text{nF}$  und  $1\mu\text{F}$  angegeben, bei einer Nennspannung von  $16\text{V}$  oder  $25\text{V}$ . Weiter gilt gemäss Datenblatt, dass bei MOSFET's mit einem  $Q_G \geq 40\text{nC}$  die Gatekapazität  $1\mu\text{F}$  sein soll. Da die Kapazität  $61\text{nF}$  beträgt, ist dies der Fall. [trinamic\\_tmc6200\\_datasheet\\_2013](#)

$$C_{Bootstrap} = 1\mu\text{F (25V)}$$

### 5.3 Flüssigkeitsbeförderung

#### 5.3.1 Pumpen

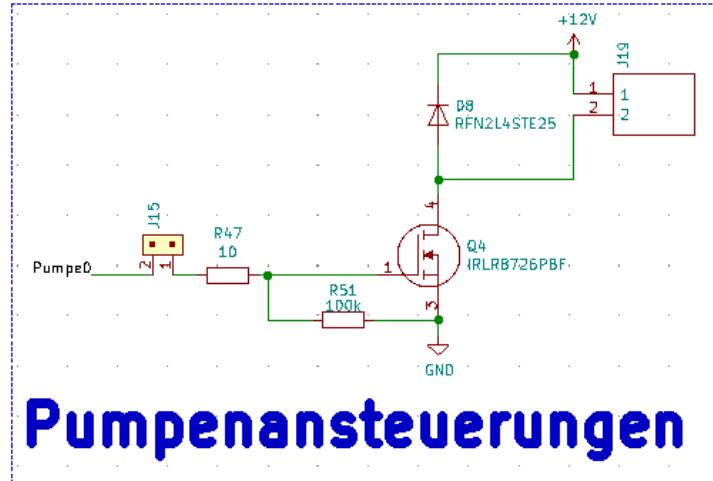


Abbildung 5.17: Schema der Pumpenansteuerung.

#### 5.3.2 Durchflussmessgeräte

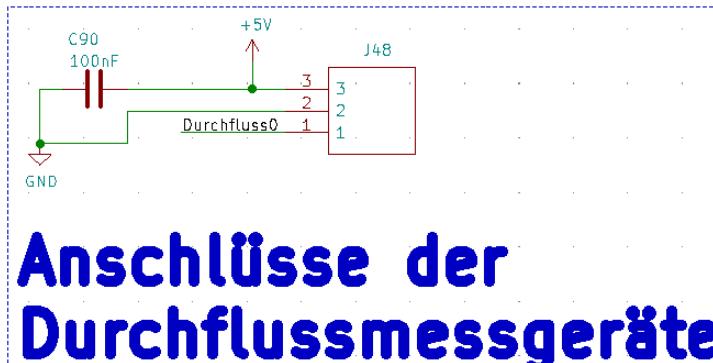


Abbildung 5.18: Schema der Durchflussmessgeräte.

## 5.4 Benutzerschnittstellen

### 5.4.1 Display

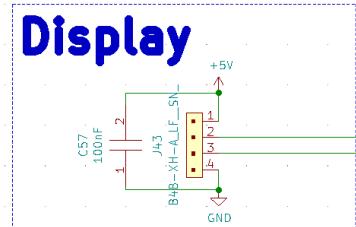


Abbildung 5.19: Schema Display.

### 5.4.2 ESP

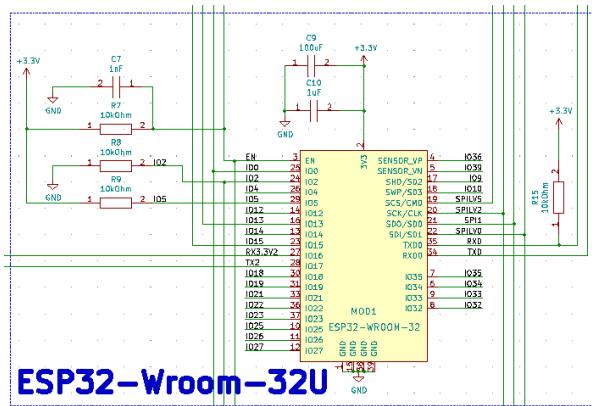


Abbildung 5.20: Schema ESP32-Wroom-32U.

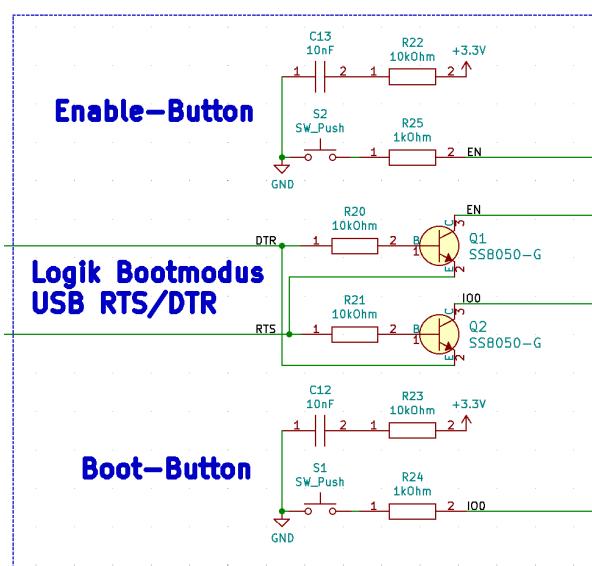


Abbildung 5.21: Schema ESP32-Wroom-32U.

### 5.4.3 USB-B

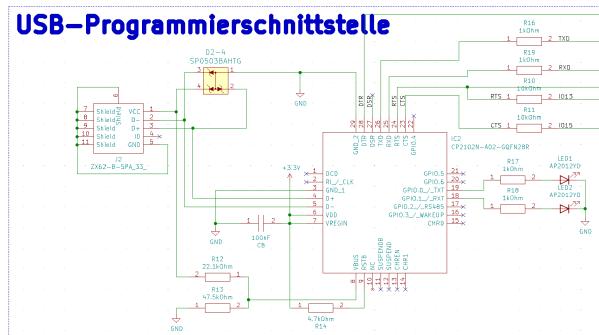


Abbildung 5.22: Schema USB-B.

### 5.4.4 RFID

Ein Bestandteil der Cocktailmaschine ist, mittels einem persönlichen kontaktlosen Chip oder Badge das Lieblingsgetränk auswählen zu können. Dazu wird auf die RFID<sup>3</sup>-Technologie gesetzt. Sie ermöglicht es mittels RFID-Tags oder auch mit NFC<sup>4</sup> des Handys Daten zwischen Tag und Maschine auszutauschen. So kann der User identifiziert werden.

Damit die Datenübertragung stattfinden kann, wird an der Antenne ein Wechselfeld angelegt, welches im Folgenden Energiesignal genannt wird. Dieses induziert beim Empfänger eine Spannung, welche als Energieversorgung dient. Der Sender moduliert ein Datensignal über das Energiesignal. 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.

### Schaltungsaufbau

Bild 5.23 zeigt den Schaltungsaufbau des RFID-Moduls. Darin ersichtlich sind folgende Teilbereiche:

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

### 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. Das Anpassnetzwerk sollte die gleiche Impedanz haben wie die Antenne. Wie die Bauteile dimensioniert werden, kann in einer Anleitung nachgesehen werden. Sie dazu Anleitung: **nxp\_bv\_2010\_antenna\_2010**.

<sup>3</sup>Radio-Frequency Identification

<sup>4</sup>Near-Field-Communication

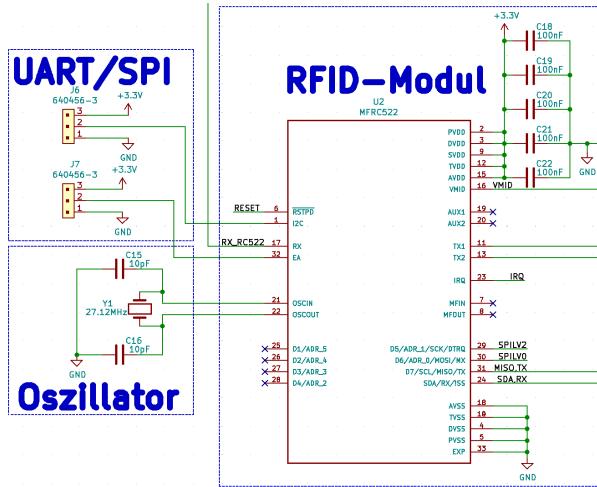


Abbildung 5.23: Schema des RFID Sender/Empfänger

Die Anschlüsse sind so gestaltet, dass falls eine einens gelayoutete Antenne verwendet wird, die Kommunikation zwischen UART und SPI gewählt werden kann. UART lässt sich einfacher in die Software einbinden, so die bisherige Erfahrung. Aus zeitgründen wurde jedoch für unser System ein Breakout-Board verwendet, welches nur einen SPI-Anschluss hat. In erster Linie damit keine Zeit gebraucht wird, sich mit einem Antennendesign auseinandersetzen zu müssen. Aber auch, weil der IC in ein TQFP28-Gehäuse gepackt ist, und schwierigkeiten bereiten könnte beim Löten.

#### 5.4.5 Beleuchtung

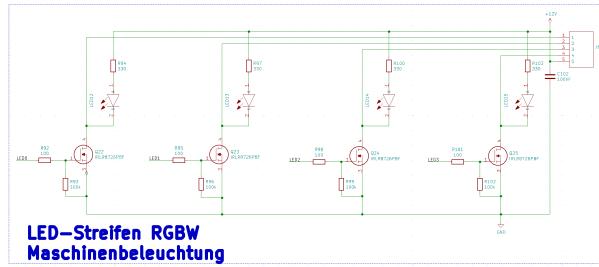
Damit die Maschine ein richtiger Hingucker wird, müssen die Blicke auf sie gezogen werden. Dazu eignet sich ein LED-Band hervorragend, am besten wenn es noch verschiedenfarbig ist. Dazu wird eine geeignete Ansteuerung benötigt. Es wird davon ausgegangen, dass das LED-Band die benötigten Widerstände schon eingebaut hat und die Anschlüsse somit direkt an die FET's gehangen werden können.

#### Schaltungsausbau

Abbildung 5.24 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.

#### Funktionsbeschrieb der Schaltung

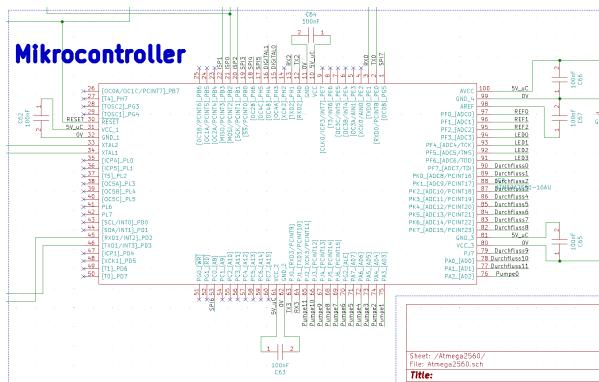
Licht sind bekanntlich elektromagnetische Wellen im sichtbaren Wellenlängen-Bereich. Dabei gibt es drei oder vier Hauptfarben, Rot, Grün, Blau und je nach dem Weiss. Zwar kann Weiss auch aus einer Kombination aller drei Farben erstellt werden, kommt aber besser mit einer



**Abbildung 5.24:** Schema der LED-Ansteuerung

separaten Diode. Das resultierende Licht des Bandes ist eine Überlagerung der Wellenlnge n. Diese Überlagerung kann vom Mikrocontroller uber den Duty-Cycle eines PWM-Signals gesteuert werden. So lassen sich mit dem Licht praktisch aus den Grundfarben praktisch alle Farben mischen.

## 5.5 Mikrocontroller



**Abbildung 5.25:** Schema Mikrocontroller

## 6 Inbetriebnahme

## 6.1 Speisungen

### 6.1.1 12V Speisung

## 6.1.2 5V Speisung

### 6.1.3 3.3V Speisung

## 6.2 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.

### stk500v2 Bootloader

Ein Bootloader (BL) ermöglicht den Programmervorgang über USB und stellt sicher, dass der frisch kompilierte Code in den Programmspeicher des  $\mu$ C 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 dieses Programm nach stk500v2-Protokoll in den Flash-Speicher geladen. Aufgrund des BL geht es nach einem Reset des  $\mu$ Cs zwei Sekunden, bis das eigentliche Programm startet.

Für den  $\mu$ C des Cocktailmixers wird ein stk500v2-BL verwendet. Dieser kann über Github<sup>5</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 Cocktailmixers nur mit Abweichungen funktioniert (4096 words anstelle 1024). Durch Anpassen des start-Vektors auf den Programmspeicher und Setzen der korrekten Bootloader-Fuses wäre es möglich, den BL-Speicherplatz zu verkleinern um dem Programmspeicher mehr Platz zu geben.

cite:  
<http://www1.m...>

### 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 aktiviert werden. Der Speicherplatz für den BL wird auf 4096 words gesetzt (anders als stk500v2 Erklärung). Das EEPROM soll beim Löschen des  $\mu$ C 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  $\mu$ C verwenden wir einen 16MHz Full-Swing-Crystal-Oszillatator, 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.

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."

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

Daraus folgt für die Fuse- und Lock-Bits die Einstellungen gemäss Tabelle 6.1. Siehe Anhang D und E für Details.

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

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

<sup>5</sup><https://github.com/arduino/Arduino-stk500v2-bootloader/blob/master/stk500boot.c>

## AVRdude in Atmel Studio einbinden

AVRdude ist eine Software, mit der Atmel AVR Controller programmiert werden können. Sie schreibt den bereits kompilierten HEX-Code der Programmierumgebung über den Bootloader in den Flash-Speicher des Controllers. Hier gäbe es auch eine Methode, die Fuse- und Lock-Bits des µC zu setzen.

<https://www...>

Über einen Link<sup>6</sup> kann eine Datei heruntergeladen werden in Form von [avrduude-6.3-mingw32.zip](#). Der gleichnamige Ordner wird im Ordner [C:\Tools](#) gespeichert. Danach wird in AtmelStudio der Reiter ["Tools→External Tools"](#) ausgewählt und ein neues Tool hinzugefügt. Im Falle des Atmega2560 geben wir die Commands gemäss Tabelle 6.2 ein:

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

**Tabelle 6.2:** AVRdude Commands

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

cite Herr  
Meier  
Skript mc1

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 angefügt.

## Inbetriebnahme

1. Als Erstes wurden die Fuse-Bits gesetzt. Dies geschah über den Reiter:  
[AtmelStudio → Tools → Device programming → Fuses](#) (siehe Abbildung E.1)  
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 E.4)  
Hier wird ein stk500v2-BL verwendet, kann aber auch abweichen. (Entsprechende Anpassungen nötig, nicht Teil dieses Projektes.)
3. Als Drittes wurden die Lock-Bits gesetzt unter:  
[AtmelStudio → Tools → Device programming → Lock-Bits](#) (siehe Abbildung E.3)  
Diese sollten nicht mehr geändert werden. Bei jedem Brennen des BL wieder zu setzen.
4. Ggf. USB-Firmware installieren auf dem USB-UART-Converter. (Nicht Teil dieses Projektes.)
5. Mikrocontroller mit der kompilierten Software (Cocktailmixer.HEX) programmieren.  
[AtmelStudio → Tools → Cocktailmixer](#)  
Alternativ direkt mit ISP-Programmer wie in Schritt 2 (ohne Bootloader):  
**z.B C:\Users\DaU\Software\Cocktailmixer\Cocktailmixer\Debug\Cocktailmixer.HEX**

<sup>6</sup><http://download.savannah.gnu.org/releases/avrduude/>

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<sup>7</sup>.

Für die Inbetriebnahme des Mikrocontrollers wurde der Alternativweg gewählt. Die Ergebnisse können sich zeigen lassen. Der Mikrocontroller ist programmierbar und erste Tests mit der Software waren erfolgreich.

## 6.3 Benutzerschnittstellen

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

### 6.3.1 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.

### 6.3.2 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 können. Es wird getestet, ob das ESP sich im vorgegebenen Netz anmeldet und als Webhost dient, ob bei Klicken auf ein Button im Explorer ein Event auslöst, ob die Kommunikation zwischen µC und Mikrokontroller funktioniert.

Folgende Schritte wurden befolgt:

1. Benötigte Daten von Github<sup>8</sup> herunterladen.
2. Dateien Entpacken und speichern unter:  
`C:\Users\Benutzer\Documents\Arduino\hardware\espressif\esp32`  
Damit die Arduino IDE die Files findet.
3. Arduino IDE starten und folgenden Reiter öffnen:  
`Arduino IDE → Werkzeuge → Board`  
ESP32 Dev Module auswählen.

---

<sup>7</sup><https://www.instructables.com/id/Turn-Ardudos-Serial-Converter-Into-AVRISP-MkII-Cl/>

<sup>8</sup><https://github.com/espressif/arduino-esp32>

4. 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.

5. Testprogramm herunterladen<sup>9</sup>, leicht modifizieren und hochladen.

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

Für die Inbetriebnahme wurde das Testprogramm so modifiziert, dass das ESP32 gleich wie das Touch-Display getestet werden kann, einfach über einen anderen Port des µC. Es ist bei den anderen Firmwares auf dem USB-Stick zu finden. Die "Debug Kommunikation" wird über den ersten Seriellen Port des ESP32 stattfinden. Diese kann gleich in der Arduino IDE geöffnet werden unter:

[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. 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.

### 6.3.3 USB-C

### 6.3.4 RFID

## 6.4 Flüssigkeitsbeförderung

### 6.4.1 Pumpen

### 6.4.2 Durchflussmessgeräte

## 6.5 Beleuchtung

## 6.6 Motor

### 6.6.1 BLDC und H-Brücke

### 6.6.2 ABN-Encoder

### 6.6.3 Treiber

---

<sup>9</sup><https://randomnerdtutorials.com/esp32-web-server-arduino-ide/>

## 7 Software

**7.1 Strukturplan****7.2 Programmflussdiagramm****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:

---

---

## A TMC4671

### A.1 Standard-Schaltkreis TMC4671

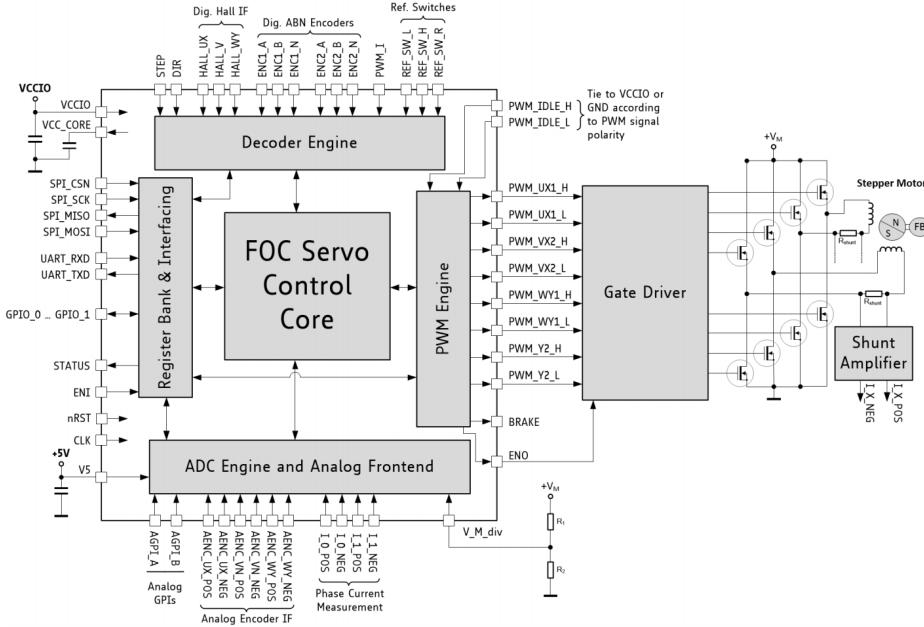


Abbildung A.1: Standard-Anwendungs-Schaltung.

citeTMC4671  
Datenblatt

### A.2 Blockdiagramm TMC4671

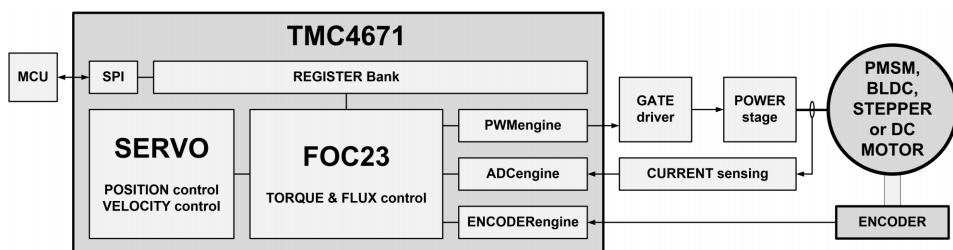


Abbildung A.2: Blockdiagramm TMC4671.

citeTMC4671  
Datenblatt

## B TMC6200

### B.1 Standard-Schaltkreis TMC6200

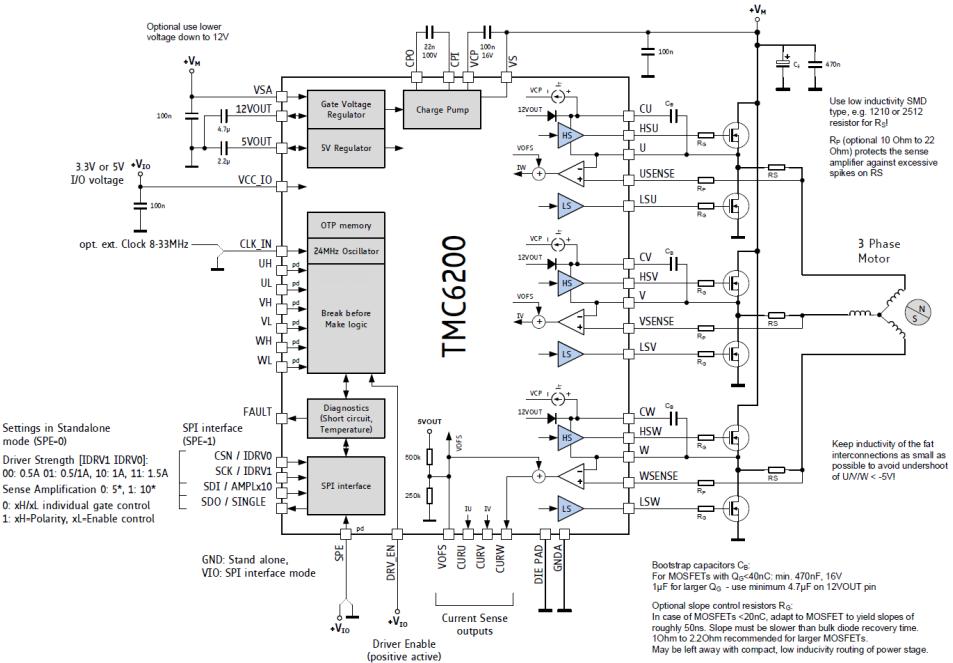


Abbildung B.1: Standard-Anwendungs-Schaltung TMC6200.

### B.2 Blockdiagramm TMC6200

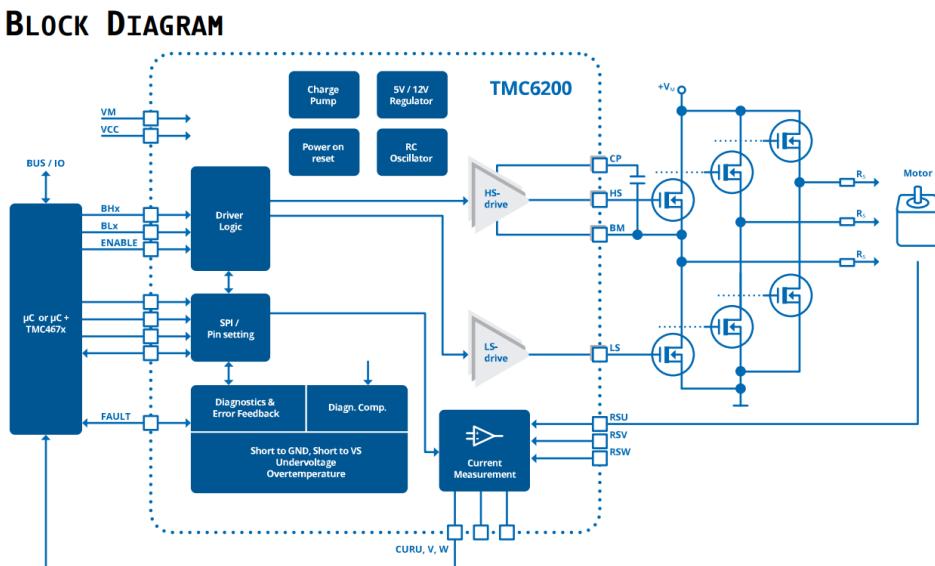


Abbildung B.2: Blockdiagramm TMC6200.

### B.3 Dimensionierungstabelle Verstärkungsfaktor Strommessung und Strommesswiderstand

CHOICE OF $R_{SENSE}$ AND AMPLIFICATION DEPENDING ON MAX. COIL CURRENT				
$R_{SENSE}$ [mΩ]	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 B.3:** Tabelle zur Bestimmung des Strommesswiderstandes aus dem Datenblatt von Trinamic.

#### B.3.1 Dimensionierungstabelle Register und Gate-Vorwiderstand

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

**Abbildung B.4:** Tabelle zur Bestimmung der Gatewiderstände aus dem Datenblatt von Trinamic.

## C H-Brücke

### C.1 Referenzschema

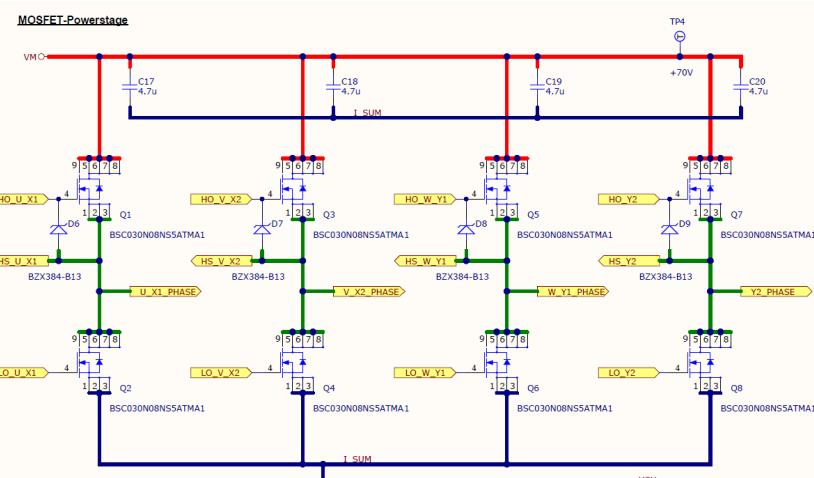


Abbildung C.1: H-Brücke.

cite: Datenblatt UPS 10A70V Schema

## D Mikrocontroller

### D.1 Brown-out-Detection

BODLEVEL 2:0 Fuses	Min. $V_{BOT}$	Typ. $V_{BOT}$	Max. $V_{BOT}$	Units	
BOD Disabled					
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.

cite: Datenblatt Atmega 2560, Seite 361

### D.2 Full Swing Crystal Oscillator

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.

cite: Datenblatt Atmega 2560, Seite 43

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

Abbildung D.3: Tabelle Aufstartzeit.

### D.3 Bootloader-Speicherplatz

cite: Datenblatt Atmega 2560, Seite 43

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

Abbildung D.4: Tabelle Bootloader Speicherplatz.

### D.4 Memory-Lock Bootloader

cite: Datenblatt Atmega 2560, Seite 320

Memory Lock Bits			Protection Type
BLB1 Mode	BLB12	BLB11	
1	1	1	No restrictions for SPM or (E)LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
<u>3</u>	<u>0</u>	<u>0</u>	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.5: Tabelle Memory Lock.

cite: Datenblatt Atmega 2560, Seite 326

## E Atmel Studio

### E.1 Fuse Bits

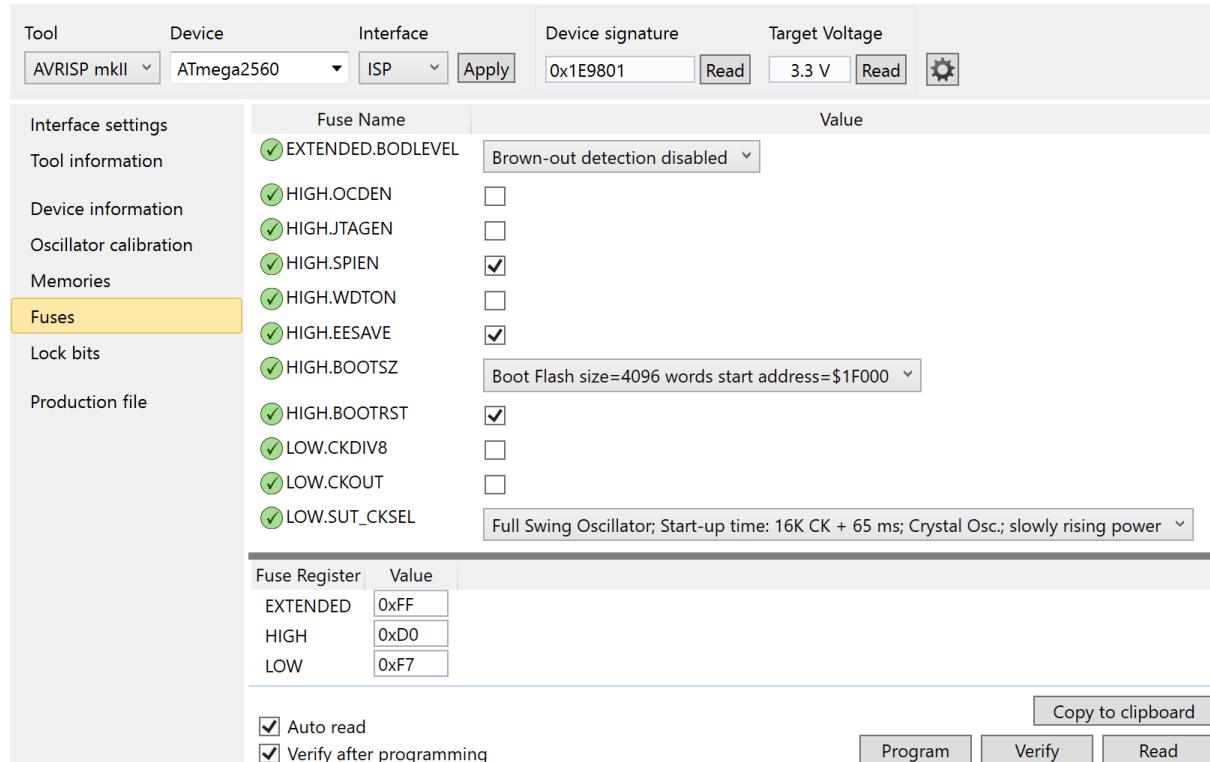


Abbildung E.1: Fuse-Bits Atmega2560.

### E.2 Lock Bits

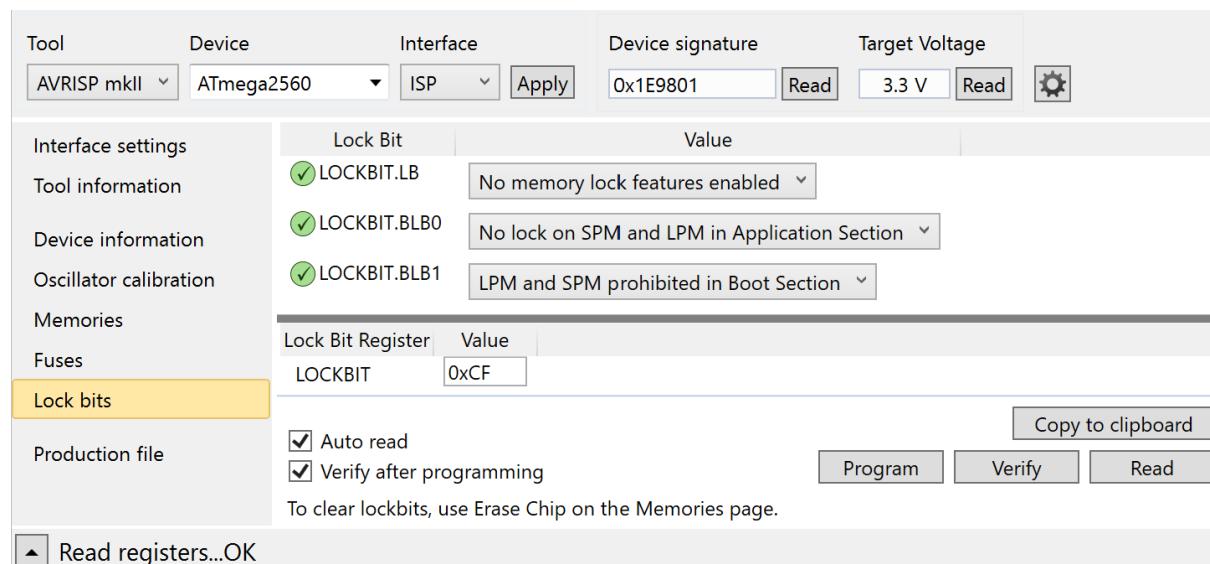
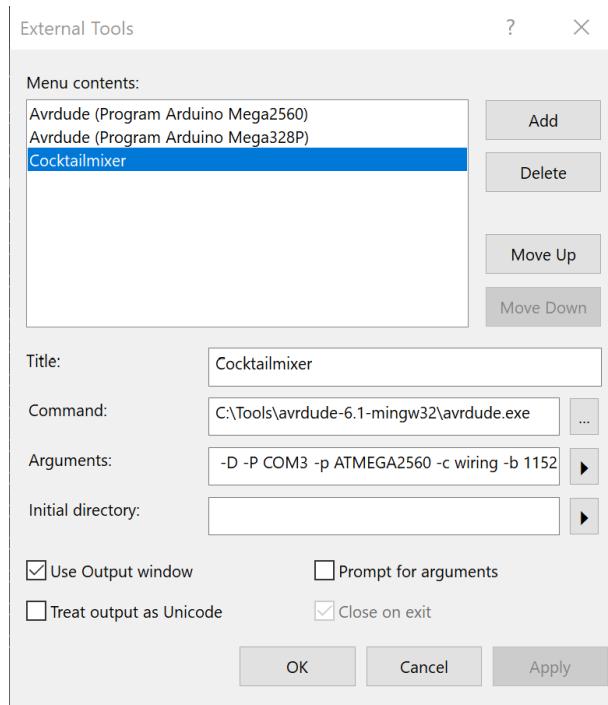


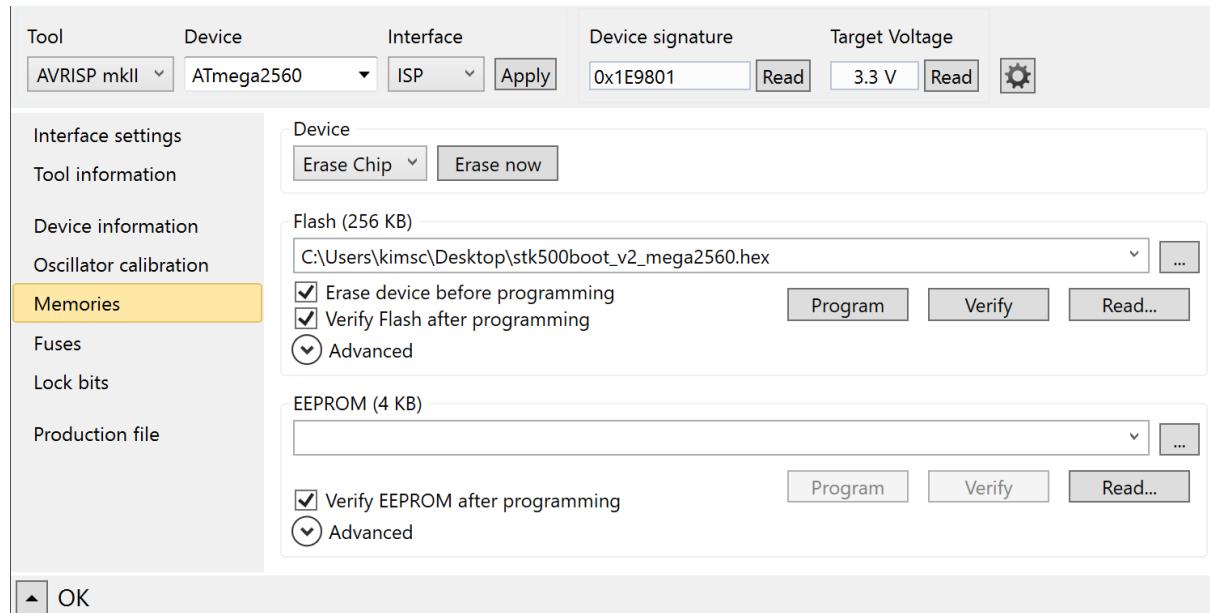
Abbildung E.2: Lock-Bits Atmega2560.

### E.3 Einbinden AVRdude und stk500v2 (wiring)



**Abbildung E.3:** External Tools Atmega2560.

### E.4 Bootloader "Brennen"



**Abbildung E.4:** Bootloader brennen.