Masterarbeit

Entwicklung, Konstruktion und Programmierung eines Fußballroboters der RoboCup Small Size League

ausgeführt am
Institut für Elektronik
Technische Universität Graz
Leiter: Univ.-Prof. Dipl.-Ing. Dr. techn. Wolfgang Pribyl

von
066/0330985 Walter Craffonara

Begutachter: Ass. Prof. Dipl.-Ing. Dr. Gunter Winkler

Graz, 22. November 2010

Abstract

This master's thesis discribes the design and development of a hardware platform for a football robot of the RoboCup Small Size Leage.
The deliberatly oversized platform proves the oppertunity to easily add other features without having to pay attention to the performance ability of the central processor.
Based on robotics a new architecture of a robot is developed. The robots' hardware is divided into several moduls whose functionality and dimensionings are discribed in detail. Besides the results of the simulations of a number of circuitry are illustrated and discussed.
Furthermore, the firmware of the robot is elaborated on: the operating mode is visualised and explained through process charts.
Finally the outcomes of the master's thesis are discussed and potential extensions of the designed hardware platform are reviewed.

Kurzfassung

Im Rahmen dieser Masterarbeit wird die Entwicklung einer Hardwareplattform eines Fußballroboters der RoboCup Small Size League beschrieben. Die bewusst überdimensionierte Plattform bietet die Möglichkeit, weitere Features problemlos hinzuzufügen ohne auf die Leistungsfähigkeit des Hauptprozessors achten zu müssen.
Ausgehend von den Grundlagen der Robotik wird eine neue Architektur eines Roboters ausgearbeitet. Die Hardware des Roboters wird in mehrere Module unterteilt und deren Funktionalität und Dimensionierung im Detail beschrieben. Außerdem werden Simulationsergebnisse einiger Schaltungen gezeigt und diskutiert.
Im weiteren Verlauf wird auf die Firmware des Roboters näher eingegangen; deren Funktionsweise wird mittels Flußdiagrammen visualisiert und erklärt.
Abschließend werden die Ergebnisse dieser Masterarbeit diskutiert und auf mögliche Erweiterungen der entworfenen Hardwareplattform eingegangen.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Eidesstattliche Erklärung

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

Ort	Datum	Unterschrift

Danksagung

Mein Dank gilt Herrn Ass. Prof. Dipl.-Ing. Dr. Gunter Winkler, Betreuer meiner Masterarbeit, für die exzellente Betreuung während der Erstellung dieser Arbeit.
Auch möchte ich allen Mitgliedern des RoboCup-Teams *Austrian Cubes* unter der Leitung von Herrn FH-Prof. DI (FH) Alexander Hofmann für die gute Zusammenarbeit danken.
Weiters möchte ich mich bei allen Kollegen und Freunden, die mich während des Studiums immer wieder ermutigt und geholfen haben, bedanken.
Für das gewisshafte Lektorat der Arbeit richtet sich mein Dank an Herrn Dipl.-Ing. Martin Hausner.
Mein größter Dank gilt allerdings meinen Eltern Giovanni und Hilda Craffonara, die mir meine Ausbildung überhaupt ermöglicht und mich immer in meinen Entscheidungen bestärkt haben, *dêr bel iolan de döt*!

Graz, 22. November 2010 Walter Craffonara

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1. Warum sollen Roboter Fußball spielen?

Warum sollen Roboter Fußball spielen? - Dies ist wohl die häufigste Frage, die während der Arbeit gestellt worden ist. Geht es hier nur um Spielerei, oder steckt tatsächlich eine sinnvolle wissenschaftliche Arbeit dahinter? Wie im echten Fußball so muss auch im Roboterfußball die Mannschaft versuchen, eine Aufgabe gemeinsam zu lösen. In diesem Fall spricht man von einem *Multi-Agenten-System* (MAS). In einem MAS geht es darum, möglichst koordiniert, in Teilaufgaben aufgegliedert ein gemeinsames Ziel zu verfolgen[36].
Im Roboterfußball sind jedoch die Rahmenbedingungen denkbar ungüstig. Genauso wie in vergleichbaren MAS der Robotik (z.B. autonomer Staubsauger) müssen Hindernisse überwunden (Wände, Mobiliar, Personen, usw.) und Objekte erkannt werden mit dem Unterschied, dass beim Roboterfußball diese Objekte aktiv dagegen wirken. Man hat somit mit einem Extremfall von autonomen MAS zu tun.
Neben den MAS werden im Roboterfußball mehrere Wissenschaften vereint: angefangen bei der Software zur Bildverarbeitung, welche komplexe Mustererkennungsalgorithmen verlangt, über die Trajektorienberechnungen des Roboters, die hochpräzise und effiziente Regelungen erfordert, bis hin zur Elektronik und Mechanik des Roboters. Fußballroboter bieten weiterhin einen spielerischen Zugang zur Wissenschaft und werden deshalb oft in Hochschulen zu Bildungszwecken eingesetzt.

1.2. Aufgabenstellung

Das Ziel dieser Masterarbeit war, eine bestehende Hardware-Platform eines Fußballroboters der RoboCup Small Size League (SSL) auf den neuesten Stand der Technik zu bringen und neue Features hinzuzufügen. Die neue Hardware des Roboters sollte weiterhin erweiterbar sein (z.B. Kameramodul am Roboter) und modular aufgebaut sein: das Funkmodul und die Schussvorrichtung sollten austauschbar und durch zukünftige Weiterentwicklungen ersetzbar sein. Weiters sollte die Funktionalität der gesamten

Hardware softwaremäßig getestet werden. Als Rahmenbedingungen dienen die Richtlinien der RoboCup Small Size League[1].

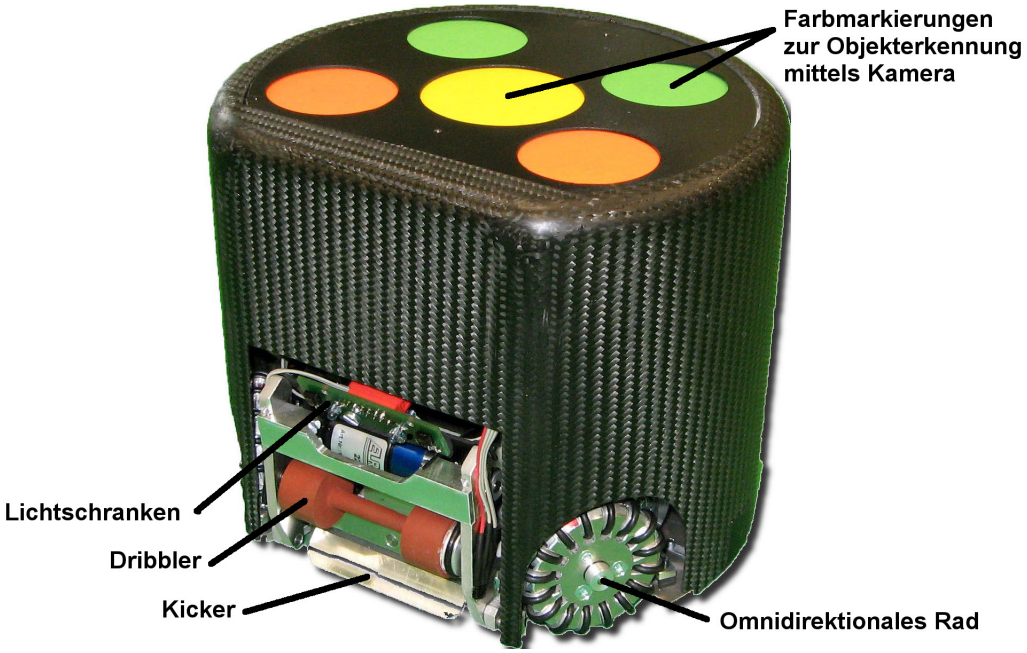


Abbildung 1.1.: Neu entwickelte Roboter-Platform

[1] Richtlinien der RoboCup Small Size League - small-size.informatik.uni-bremen.de/rules:main

2 Grundlagen

2.1. Gleichtrommotoren

Wie im Kapitel 3.6 und 3.8 beschrieben wird, kommen sowohl Motoren mit mechanischem Kommutator (bürstenbehaftete DC-Motoren) als auch Motoren mit elektrischem Kommutator (brushless DC-Motoren, BLDC-Motoren) zum Einsatz. Die folgenden Seiten erläutern die Unterschiede zwischen den beiden Motortypen und deren Ansteuerung. Es wird vor allem auf die heutzutage immer beliebter werdenden BLDC-Motoren eingegangen, da sich deren Ansteuerung etwas komplizierter gestaltet im Gegensatz zu den traditionellen bürstenbehafteten DC-Motoren.

2.1.1. Aufbau und Funktion von bürstenbehafteten DC-Motoren

Ein DC-Motor mit mechanischer Kommutierung besteht aus folgenden Komponenten (siehe Abbildung 2.1):

- Statorgehäuse mit Gleitlager und Welle (**1**)
- Rotor, bestehend aus einem Blechpaket, das die Wicklungen trägt (**2**)
- Wicklungen, die über den Kommutator vom Steuerstrom durchflossen werden (**3**)
- Kommutator (Polwender), der die abwechselnde Polung des Rotors erzeugt (**4a** und **4b**)
- Bürsten mit verschleißarmen Kontaktierungen (**5**)
- Lagerdeckel (**6**)
- Stromzufuhr (**7**)
- Dauermagnete (**8**)

Der feststehende Stator besteht in den meisten Fällen aus einem Gehäuse mit zwei Permanentmagnets die ein statisches Magnetfeld erzeugen. Dieser umschließt den beweglichen Teil des Motors (Rotor), bestehend aus einem Elektromagnet, der meistens aus mehreren Spulen mit Eisenkern aufgebaut ist. Der Rotor wird über die Bürsten und einen segmentierten Kommutator mit Strom versorgt. Fließt nun ein Strom durch den Elektromagnet, so entsteht ein Magnetfeld im Rotor, das in Wechselwirkung mit dem

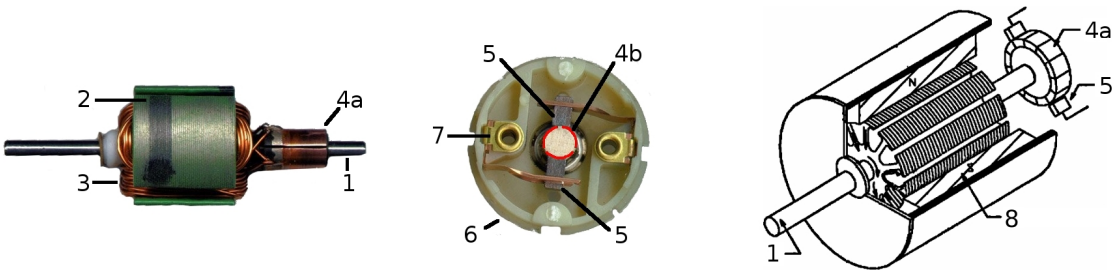


Abbildung 2.1.: Komponenten eines DC-Motors[28]

Magnetfeld des Stators tritt. Der Rotor richtet sich aus und mit ihm der segmentierte Kommutator, der immer die passenden Wicklungen des Rotors in den Stromkreis schaltet. Wäre der Kommutator nicht segmentiert, so würde sich der Rotor nur solange drehen, bis dieser am Magnetfeld des Stators ausgerichtet ist. In Abbildung 2.2, wird die Funktion des mechanischen Kommutators dargestellt.

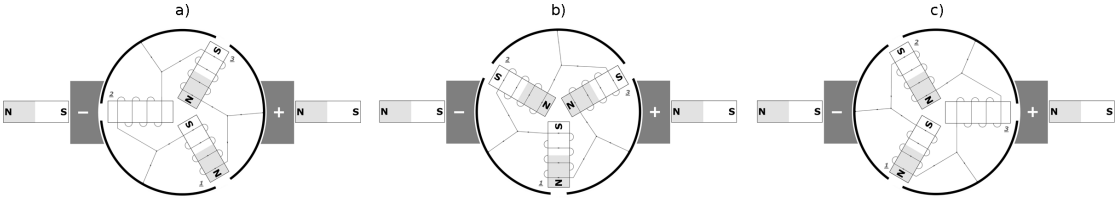


Abbildung 2.2.: Funktion des Kommutators[1]

Dieser Motor besitzt 3 Elektromagnete und 3 Segmente am Kommutator. Die beiden Bürsten des Motors müssen so breit sein, dass sie gleichzeitig zwei Segmente des Kommutators kurzschließen. In Abbildung 2.2 a) tritt dieser Fall ein: Die beiden Enden des Elektromagnets 2 werden an der gleichen Bürste angeschlossen. Aus diesem Grund fließt kein Strom durch diesen Elektromagnet und ist somit inaktiv. Jetzt wird der Elektromagnet 3 vom Nordpol angezogen, bzw. der Elektromagnet 1 vom Nordpol abgestoßen und der Rotor dreht sich im Uhrzeigersinn. Nach $\frac{1}{12}$ Umdrehung sind alle Elektromagnete von Strom durchfloßen (Abbildung 2.2 b)). Nach einer weiteren $\frac{1}{12}$ Umdrehung wird der Elektromagnet 3 ausgeschalten und der Vorgang beginnt von Neuem.

2.1.2. Aufbau und Funktion von BLDC-Motoren

Im Gegensatz zu Motoren mit mechanischer Kommutierung, arbeiten BLDC-Motoren mit elektronischer Kommutierung: die Ankerwicklungen befinden sich im Stator und der Rotor besteht aus einem Permanentmagnet (siehe Abbildung 2.3).

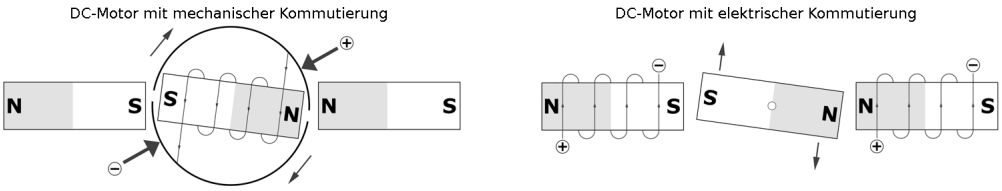


Abbildung 2.3.: Arten der Kommutierung dei DC-Motoren[1]

An Stelle der Permanentmagnete im Stator, kommen beim BLDC-Motor Elektromagnete im Einsatz, die in Abhängigkeit von der Rotorstellung intelligent angesteuert und über leistungselektronische Schalter geschaltet werden müssen[16].
Weiters gibt es, genauso wie bei Gleichstrommotoren mit Bürsten, zwei Typen von BLDC-Motoren und zwar Motoren mit Innen- und Außenläufer. Beim Innenläufer befindet sich der Rotor im Inneren des Motors, während sich dieser beim Außenläufer an der Außenseite befindet. In Tabelle 2.1 sind die Merkmale dieser beiden Typen gegenübergestellt.

	Innenläufer	**Außenläufer**
Wirkungsgrad	> 80%	65 – 85%
Drehzahl	hoch	niedrig
Drehmoment	mittel-hoch	hoch
Kühlung	problematisch	gut
Durchmesser	klein	groß
Herstellung	aufwändig	einfach
Preis	teuer	günstig

Tabelle 2.1.: Unterschiede zwischen Innen- und Außenläufer bei BLDC-Motoren[25]

Beide Ausführungen bestehen aus den gleichen Komponenten (siehe Abbildung 2.4): Stator(**1**), Kupferwicklungen (**2**), Oberflächenmagnete (**3**), Rotorblechpaket (**4**) und Rotorwelle (**5**). Die meisten BLDC-Motoren bestehen aus 3 Phasen, die je nach Lage des Rotors vom Kommutator mit Strom versorgt werden oder auch nicht. Damit sich ein BLDC-Motor drehen kann, muss der Regler die Winkelposition des Rotors kennen um

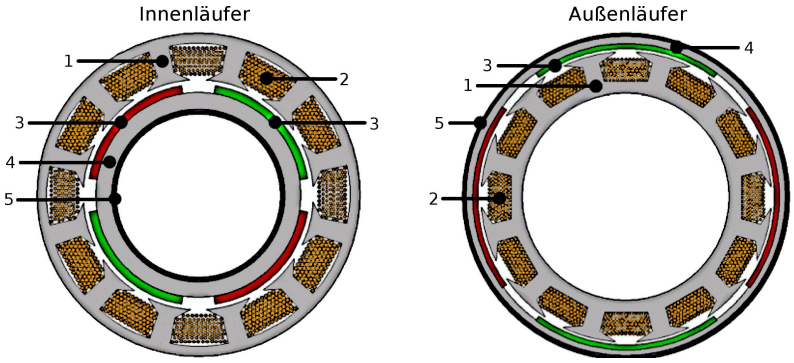


Abbildung 2.4.: Aufbau eines Innen- und Außenläufers[25]

die Reihenfolge und den Zeitpunkt der Kommutierung berechnen zu können. Wie die elektronische Kommutierung und die Bestimmung der Winkelposition des Rotors beim BLDC-Motor funktioniert, wird in Kapitel 2.1.3 erörtert.
Zusammenfassend werden nochmals in Tabelle 2.2 die Unterschiede zwischen BLDC-Motoren und bürstenbehaftete DC-Motoren aufgelistet.

2.1.3. Ansteuerung von BLDC-Motoren

Es gibt grundsätzlich zwei Arten der Kommutierung um die 3 Phasen des BLDC-Motors mit Strom zu versorgen:

Blockkommutierung (Trapezansteuerung) : es werden immer nur 2 Phasen mit Strom versorgt, während die 3. Phase offen bleibt. Daraus ergeben sich 6 möglichen Schaltkombinationen.

Sinuskommutierung : alle 3 Phasen werden mit sinusförmigen Phasenströme, die 120° phasenverschoben sind, versorgt.

Das Ziel jeder Kommutierungsart ist, die Ströme an den Phasen so anzulegen, damit das größtmögliche Drehmoment entsteht. Dies erreicht man durch die senkrechte Orientierung der Magnetfelder von Permanentmagnet und Wicklung. Da wir die Orientierung des Magnetfeldes der Wicklung (Stator) selbst erzeugen, brauchen wir nur noch die Winkelposition des Permanentmagnets (Rotor) um ein möglichst großes Drehmoment erzeugen zu können.

	BLDC Motor	**Bürstenbehaftete DC Motor**
Kommutierung	elektronisch	mechanisch, mittels Bürsten
Wartung	gering, aufgrund der fehlenden Bürsten	periodische Wartung erforderlich
Lebensdauer	lang	kürzer
$\frac{\text{Geschwindigkeit}}{\text{Drehmoment}}$-Charakteristik	gleichmäßig über den gesamten Geschwindigkeitsbereich	gleichmäßig bei niedrigeren Geschwindigkeiten
Effizienz	hoch, da kein Spannungsabfall an Bürsten	mittelmäßig
$\frac{\text{Leistung}}{\text{Raum}}$	hoch, kleine Baugröße aufgrund der guten Wärmeableitung: Die Windungen befinden sich am Stator, der am Gehäuse befestigt ist und somit die erzeugte Wärme leichter ableiten kann	mittelmäßig/schlecht, die erzeugte Wärme kann nur über den Luftspalt zwischen Rotor und Stator abgegeben werden
Trägheitsmoment	gering, da der Rotor aus Permanentmagneten besteht, was die dynamische Antwort verbessert	hoch, was die dynamische Charakteristik des Motors beeinträchtigt
Geschwindigkeitsgrenze	hoch, da mechanischer Kommutator nicht vorhanden	niedrig, mechanisch begrenzt aufgrund der Bürsten
Störungen	gering	hoch, aufgrund des Bürstenfeuers
Baukosten	hoch	gering
Steuerung	komplex und teuer	einfach und billig
Steuerungsanforderung	um den Motor zum Laufen zu bringen ist immer eine Steuerung notwendig	eine Steuerung ist nur dann notwendig wenn die Geschwindigkeit variieren soll

Tabelle 2.2.: Unterschiede zwischen bürstenbehafteten DC- und BLDC-Motoren[48]

Die Winkelposition des Rotors kann anhand von Hall-Sensoren, Encoder, Resolver oder sensorlos bestimmt werden.

2.1.3.1. Blockkommutierung mit Hall-Sensoren

Wie man in Abbildung 2.5 zeigt, werden die Phasenströme immer nach 60° (oder besser gesagt nach $\frac{60°}{\#\text{Polpaare}}$) abrupt geschaltet. Dieses abrupte Umschalten führt zu einem nicht konstanten Drehmoment innerhalb des Kommutierungsintervalls. Dies manifestiert sich als Vibrationen im höhrbaren Bereich und bei niedrigen Drehzahlen zu ungleichmäßigen Bewegungen.

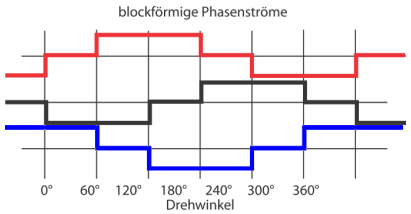


Abbildung 2.5.: Phasenströme bei Blockkommutierung (einpoliger BLDC-Motor)[30]

Wie in Kapitel 2.1.3 beschrieben, ist die Kenntnis über die Winkelposition des Rotors eine Voraussetzung um BLDC-Motoren ansteuern zu können. Eine Möglichkeit um die absolute Position des Rotors zu messen ist der Einsatz von Hall-Sensoren. In den meisten Fällen sind solche Sensoren in den Motoren bereits eingebaut, können aber auch nachträglich extern angebracht werden. Bei den hier verwendeten BLDC-Motoren *EC 45 flat* sind 3 solcher Sensoren im Abstand von 120° auf einer Printplatte im hinteren Bereich des Motors montiert. Diese Sensoren können die Richtung des auf der Welle angebrachten Steuermagnets detektieren und erzeugen 5V am Ausgang falls ein Nordpol in der Nähe ist. Ein Südpol erzeugt hingegen 0V am Ausgang. Die 3 Hall-Sensoren erzeugen somit gemeinsam alle $\frac{60°}{\#\text{Polpaare}}$ eine neue Kombination der Ausgangszustände. Mit der gleichen Auflösung werden bei der Blockkommutierung auch die Phasenströme des BLDC-Motors umgeschaltet (siehe Blockkommutierung in Abbildung 2.5 und 2.6).

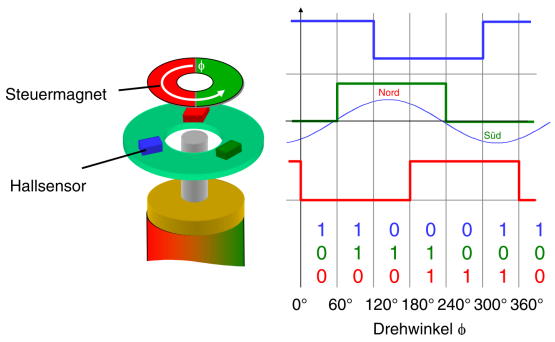


Abbildung 2.6.: Signalverlauf der Hall-Sensoren bei einpoliger BLDC-Motor[**?**]

Je höher die Anzahl der Polpaare eines Motors ist, desto höher muss die Kommutierungsfrequenz sein um die gleiche Drehzahl zu erreichen.

Das höchst mögliche Drehmoment wird erzeugt, indem die Phasenströme $\frac{30°}{\#Polpaare}$ vor jedem Signalwechsel der Hall-Sensoren umgeschaltet werden. Dadurch wird verhindert, dass sich der Rotor nach dem Feld des Stators ausrichten kann. Die Felder des Rotors und des Stators stehen somit immer senkrecht aufeinander und üben somit die maximale Kraft aus (maximales Drehmoment) [**?**].

Eigenschaften der Blockkommutierung[30]

- relativ einfache und kostengünstige Elektronik
- kontrollierter Anlauf
- Drehmomentrippel von typisch 15%
- hohe Anlaufmomente und Beschleunigungen möglich

2.1.3.2. Sensorlose Blockkommutierung

Viele BLDC-Motoren werden ohne jegliche Art von Sensoren ausgeliefert und bieten daher nicht die Möglichkeit die Rotorposition, wie im Falle der Hall-Sensoren, direkt messen zu können. Diese sensorlosen Motoren haben lediglich 3 Anschlüsse für die Wicklungen. Um in diesem Fall die Position des Rotors ermitteln zu können, behilft man sich der sogenannten Gegen-Elektromotorischen Kraft (Gegen-EMK, engl. Back-EMF). Darunter versteht man jene Spannung, die in den Wicklungen des Motors induziert wird, sobald sich der Rotor im Magnetfeld des Stators dreht. Die Gegen-Elektromotorische Kraft ist drehzahlabhängig; je höher die Drehzahl, desto höher ist die induzierte Spannung[2].
Die Erzeugung der Blockkommutierung mit Hilfe der Gegen-EMK kann am Besten anhand eines einpoligen BLDC-Motors mit Wicklung in Sternschaltung erläutert werden (siehe Abbildung 2.7).

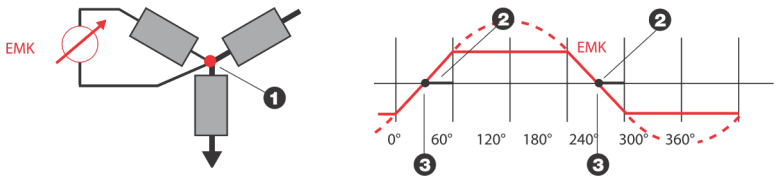


Abbildung 2.7.: Sensorlose Kommutierung[30]

Wie bereits in Kapitel 2.1.3.1 erläutert, fließt bei der Blockkommutierung immer nur

ein Strom durch zwei Phasen, während die dritte nicht bestromt wird. In dieser Phase wird jedoch vom rotierenden Permanentmagnet eine sinusförmige Spannung, die Gegen-EMK, induziert. Diese induzierte Spannung hat genau in der Mitte des 60° Kommutierungsintervalls einen Nulldurchgang, der ermittelt werden kann, wenn der Sternpunkt (1 in Abbildung 2.7) der Wicklungen zugänglich ist.
Wird ein Nulldurchgang detektiert (2 in Abbildung 2.7), so muss sich der Rotor noch weitere 30° drehen, bis der nächste Schaltvorgang der Blockkommutierung ausgelöst werden kann (3 in Abbildung 2.7). Durch Messung der Zeitdifferenzen zwischen aufeinanderfolgenden Nulldurchgängen kann die Drehzahl ermittelt werden und daraus die Zeit, die man zwischen einem Nulldurchgang und dem nächsten Schaltvorgang der Blockkommutierung warten muss. Im darauf folgenden Kommutierungsintervalls wird wiederum jene Phase betrachtet, die nicht bestromt ist.
Diese Art der Regelung kann eingesetzt werden, solange die Drehzahl des Motors hoch genug ist, dass die Nulldurchgänge der Gegen-EMK detektiert werden können; bei niedrigen Drehzahlen ist die sinusförmige Induktionspannung der Gegen-EMK zu niedrig um die Nulldurchgänge präzise zu ermitteln. Noch schlimmer, im Stillstand verschwindet die Gegen-EMK vollständig.
Aus diesem Grund verwendet man bei tiefen Drehzahlen eine spezielle Anlaufprozedur um die Motoren auf touren zu bringen: die 3 Phasen des Motors werden der Reihe nach, gemäß der Reihenfolge der Blockkommutierung bestromt, ohne auf die Gegen-EMK zu achten. Die Kommutierungsfrequenz wird dabei ständig erhöht und der Rotor beschleunigt. Sobald die Mindestdrehzahl erreicht ist, kann die eigentliche sensorlose Kommutierung eingeschaltet werden.
Die Gegen-EMK kann als abfallende Spannung zwischen einem Phasenanschluss und dem Sternpunkt der Wicklungen gemessen werden. Der Sternpunkt ist jedoch bei den meisten Motoren von außen nicht zugänglich oder gar nicht vorhanden (z.B. bei Motoren mit Dreieckschaltung).
Es gibt jedoch eine Möglichkeit einen virtuellen Sternpunkt zu schaffen, indem drei Widerstände in Sternschaltung parallel zur Motorwicklung geschaltet werden (siehe Abbildung 2.8).

Eigenschaften der sensorlosen Blockkommutierung[30]

- kein definierter Anlauf
- nicht geeignet für kleine Drehzahlen
- nicht geeignet für dynamische Anwendungen
- Drehmomentrippel von typisch 15%

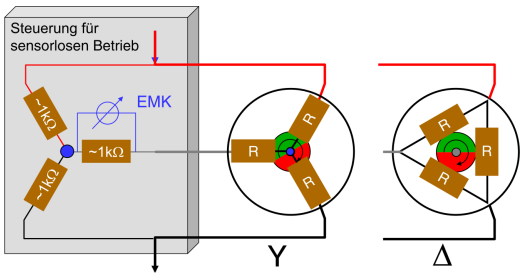


Abbildung 2.8.: Virtueller Nullpunkt bei BLDC-Motoren[?]

2.1.3.3. Sinuskommutierung

Ein Nachteil der Blockkommutierung ist, dass durch das abrupte Umschalten der Phasenströme, ein Drehmomentrippel entsteht, der sich besonders bei tiefen Drehzahlen durch einen sehr ungleichmäßigen Rotationsverlauf bemerkbar macht. Dies kann verhindert werden, indem die Phasenströme graduell angeglichen werden in Form eines sinusförmigen Stromverlaufs (siehe Abbildung 2.9). Dadurch bleibt das Drehmoment konstant und man erreicht einen sehr weichen und präzisen Lauf des Motors.

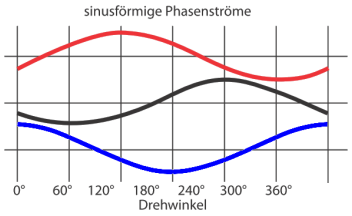


Abbildung 2.9.: Phasenströme bei Sinuskommutierung[30]

Da bei der Sinuskommutierung die Phasenströme viel häufiger angepasst werden müssen als im Falle der Blockkommutierung, muss auch die Positionsauflösung des Rotors entsprechend hoch sein. Deswegen werden für die Positionsbestimmung des Rotors meist Encoder oder Resolver mit hoher Auflösung verwendet[?].

Eigenschaften der Sinuskommutierung[30]

- aufwändigere Elektronik
- kein Drehmomentrippel

- sehr gute Gleichlaufeigenschaften auch bei kleinsten Drehzahlen
- ca. 5 % höheres Dauerdrehmoment als bei Blockkommuntierung

2.2. Kinematik des omnidirektionalen Antriebes

In diesem Kapitel wird die Berechnung der Winkelgeschwindigkeit der einzelnen omnidirektionalen Räder (siehe Kapitel 3.8) erklärt. Die am PC laufende Künstliche Intelligenz berechnet ungefähr 15 Mal in der Sekunde die Soll-Geschwindigkeiten in x- (ϑ_x) und y-Richtung (ϑ_y) und der Soll-Drehwinkel (ω) aller Roboter am Spielfeld. Diese Daten werden an die Roboter gesendet, welche daraus die Drehzahlen der einzelnen omnidirektionalen Räder berechnen. Die Künstliche Intelligenz berechnet somit zu jedem Zeitpunkt die Beschleunigungs-, bzw. Bremswege aller Roboter.
Basierend auf den hergeleiteten Matrizen zur Berechnung der Drehzahlen für einen asymetrischen 4-Rad-Antrieb mit Omniwheels in [34] (siehe Kapitel *Control of a four wheeled asymmetrical robot*) kann die Transformationsmatrix P bei gegebenen ϑ_x, ϑ_y und ω abgeleitet werden:

$$\left(\vartheta'_a, \vartheta'_b, \vartheta'_c, \vartheta'_d\right)^T = \begin{pmatrix} \frac{-sin(\varphi_1)}{d} & \frac{cos(\varphi_1)}{d} & \frac{R}{r} \\ \frac{-sin(\varphi_2)}{d} & \frac{cos(\varphi_2)}{d} & \frac{R}{r} \\ \frac{-sin(\varphi_3)}{d} & \frac{cos(\varphi_3)}{d} & \frac{R}{r} \\ \frac{-sin(\varphi_4)}{d} & \frac{cos(\varphi_4)}{d} & \frac{R}{r} \end{pmatrix} \left(\vartheta_x, \vartheta_y, \omega\right)^T \tag{2.1}$$

mit
- $\vartheta'_{a,b,c,d}$ ⋯ Winkelgeschwindigkeit der Motoren bei direkter Übersetzung
- ϑ_x ⋯ Geschwindigkeit in x-Richtung
- ϑ_y ⋯ Geschwindigkeit in y-Richtung
- ω ⋯ Drehwinkel
- $\varphi_{1,2,3,4}$ ⋯ Winkel zwischen Motor a, b, c, d und Vorderseite des Roboters
- R ⋯ Radius des Roboters
- r ⋯ Radius des Omniwheels
- d ⋯ Durchmesser des Omniwheels

Die so berechneten Drehzahlen gelten nur für eine direkte Übersetzung ohne Getriebe. Bei dem hier beschriebenen Roboter beträgt das Übersetzungsverhältnis jedoch $A =$

$\frac{1}{2.8}$. Um die tatsächlichen Drehzahlen $\vartheta_{a,b,c,d}$ zu berechnet, müssen $\vartheta'_{a,b,c,d}$ durch das Übersetzungsverhältnis des Getriebes dividiert werden.

$$\begin{aligned} \vartheta_{a,b,c,d} &= \frac{\vartheta_{a,b,c,d}}{A} \\ &= 2,8 * \vartheta_{a,b,c,d} \end{aligned} \tag{2.2}$$

Die entsprechenden Winkeln $\varphi_{1,2,3,4}$ können aus Abbildung 2.10 entnommen werden und betragen $\varphi_1 = 54°$, $\varphi_2 = 135°$, $\varphi_3 = 225°$ und $\varphi_4 = 306°$. Würden die Räder in 90° und 180° aufeinander stehen, wäre der Rückstoß beim Schießen zu stark und der Roboter würde sich nach hinten verschieben.

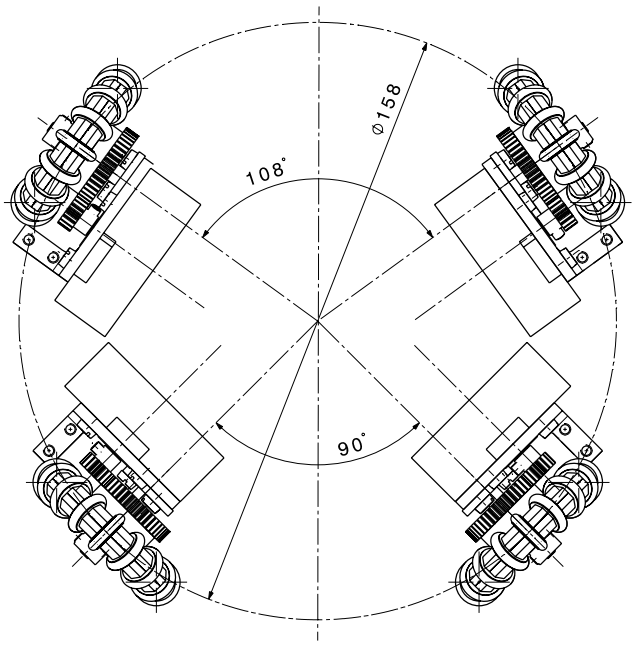


Abbildung 2.10.: Fahrwerk des Roboters

3 Hardware

Die Elektronik ist in mehrere Module unterteilt, die jeweils in eigenen Unterkapiteln genau beschrieben werden. Manche Schaltungen wurden mit dem kostenlosen Simulationsprogramm *LTSpice IV*[1] von *Linear Technology* simuliert und optimiert und die Ergebnisse dieser Simulationen sind ebenfalls in den entsprechenden Abschnitten dokumentiert.
Die Elektronik des Roboters wurde komplett neu entwickelt, d.h. es wurden keinerlei Module des alten Roboters übernommen. In Abbildung 3.1 sehen Sie eine CAD-Zeichnng der elektronischen Komponenten des Roboters.

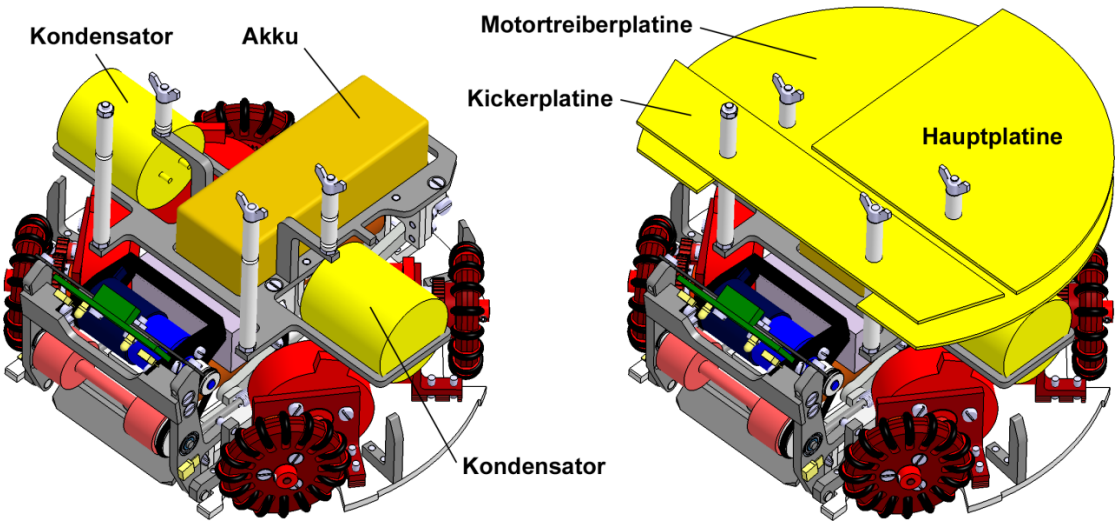


Abbildung 3.1.: CAD Modell des Roboters

[1] *LTSpice IV* - http://www.linear.com/designtools/software/

3.1. Stromversorgung

Der Roboter wird von einem Lithium-Polymer-Akku mit insgesamt 3 Zellen und einer Kapazität von 1800mAh betrieben. Die Spannung einer jeden Zelle darf $3,1V$ nicht unterschreiten, da sie ansonsten einen Schaden davon trägt oder sogar zerstört wird. Die Schaltung zur Spannungsversorgung muss die Betriebsspannungen $5V$ und $3,3V$ bereitstellen und eine Tiefentladung der einzelnen Zellen verhindern.

3.1.1. Schaltungsbeschreibung

Die Schaltung besteht aus 2 Teilen:

LiPo-Monitor zur Überwachung des Akkus

Spannungsregler zur Erzeugung der Betriebsspannungen $5V$ und $3,3V$

Im Folgenden werden die beiden Module genauer beschrieben.

3.1.1.1. LiPo-Monitor

Der LiPo-Monitor funktioniert folgendermaßen: die Spannung jeder einzelnen Zelle wird überprüft und bevor eine Tiefentladung am Akku stattfindet wird der Hauptprozessor aufgefordert den Roboter ordnungsgemäß herunterzufahren. Gleichzeitig wird ein Timer gestartet, der nach ungefähr $5s$ die Stromversorgung des Roboters unterbricht um einen Schaden am LiPo-Pack zu verhindern.
Die Zellenspannungen werden am Balancer-Stecker des LiPo-Packs mit Hilfe des Differenzverstärkers *INA2132*[20] (IC_1) mit Verstärkung $A_{IC_1} = 1$ gemessen (siehe Abbildung 3.2). Der Unterspannungsdetektor *STM1061N31*[38] (IC_2, IC_3 und IC_6) überwacht die gemessenen Zellenspannungen und schaltet den OD-Ausgang $/Out$ auf Low im Falle einer Tiefentladung.
Ab diesem Zeitpunkt wird der Timer *TLC555*[24] (IC_4) mit Strom versorgt und beginnt seine Abarbeitung: C_4 wird über R_1 aufgeladen und erzeugt somit ein einmaliges Triggerereignis an TR (solange $TR < \frac{1}{3}V_+$ ist $Q = V_+$). Nun wird C_7 über R_2 aufgeladen und sobald die Spannung $U_{Thr} > \frac{2}{3}V_+$ ist, wird C_8 über Q entladen. Die fallende Flanke am */OFF*-Eingang des High-Side-MOSFET-Treibers *MAX1614*[32] (IC_5) führt dazu, dass die Stromversorgung des Roboters unterbrochen wird, indem T_1 geöffnet wird. Wenn aber während des Aufladens von C_7 alle Zellenspannungen wieder über

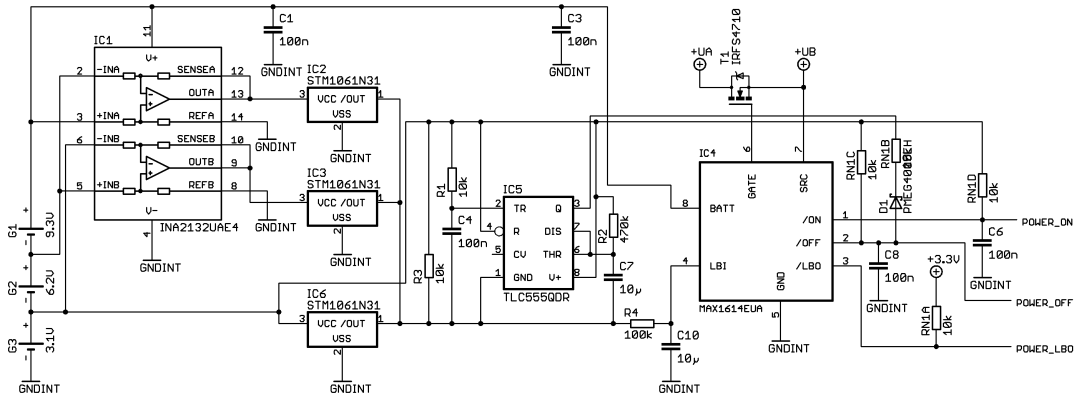


Abbildung 3.2.: Schaltung zum LiPo-Monitor

$3,1V$ gestiegen sind, dann wird IC_4 abgeschaltet und es folgt keine fallende Flanke an */OFF*. Somit ist sichergestellt, dass der Roboter aufgrund kurzzeitiger Überlastungen, bei denen die Zellenspannungen sinken können, nicht abgeschaltet wird. Damit die Stromversorgung unterbrochen wird, muss eine der Zellenspannungen für mindestens

$$\begin{aligned} t_{<3,1V} &= -R_2 C_7 \ln\left(1-\frac{2}{3}\right) \\ &= -470k\Omega * 10\mu F * \ln\left(1-\frac{2}{3}\right) = 5,16s \end{aligned} \tag{3.1}$$

unter $3,1V$ fallen.
Bevor die Stromversorgung des Roboters endgültig abgeschaltet wird, signalisiert eine fallende Flanke an *POWER_LBO* dem Hauptprozessor, dass die Abschaltung des Systems bevorsteht und dass eventuelle Einstellungen sofort abgespeichert werden müssen. Ungefähr $1s$ nachdem eine Tiefentladung detektiert wurde ist die Spannung an *LBI* unter $1,2V$ und */LBO* geht auf *Low*. Das Signal *POWER_LBO* ist mit dem *NMI*-Eingang (nicht maskierbare Interrupt-Quelle) des Hauptprozessors verbunden. Bei einer fallenden Flanke am *NMI*-Eingang erfolgt ein Interrupt mit höchster Priorität, was das Abspeichern von wichtigen Informationen am Prozessor zur Folge hat.
Um den Roboter einzuschalten wird mit Hilfe eines Push-Buttons eine fallende Flanke am Eingang */ON* erzeugt.

3.1.1.2. Spannungsregler

Als Spannungsregler zur Erzeugung der Betriebsspannungen $5V$ und $3,3V$ wurde ein *LT3500*[44] (IC_1) gewählt (siehe Abbildung 3.3). Dieser integriert sowohl einen Schalt- als auch einen Linearregler und bietet somit die beste Lösung auf engstem Raum.

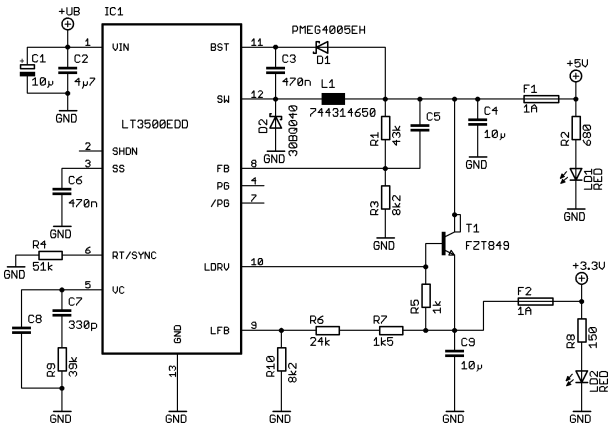


Abbildung 3.3.: Schaltung zum Spannungsregler

Die Schaltung zum Spannungsregler wurde aus dem Datenblatt des *LT3500* entnommen und die verwendete Speicherdrossel L_1 mit Hilfe vom *WE Inductor Selector 1.0*[2] dimensioniert (siehe Abbildung 3.4).
Zum Schutz gegen Überlastungen wurden die Sicherungen F_1 und F_2 an den Versorgungsleitungen $5V$ und $3,3V$ eingebaut. Die Funktion der Sicherungen lässt sich anhand der LEDs LD_1 und LD_2 überprüfen.

3.1.2. Schaltungssimulation

Die Schaltungen zur Spannungsversorgung wurden mit LTSpice simuliert und optimiert (siehe Abbildung 3.5).
Bei der Simulation des LiPo-Monitors wird bei $t = 4s$ eine Tiefentladung detektiert und $U_{/Out}$ geht auf Masse (siehe Abbildung 3.6). Daraufhin wird der Timer gestartet und C_{10} wird über R_4 entladen. Ungefähr $1s$ später erreicht U_{LBI} die Schwelle von $1,2V$.

[2] *WE Inductor Selector 1.0* - http://www.we-online.com/

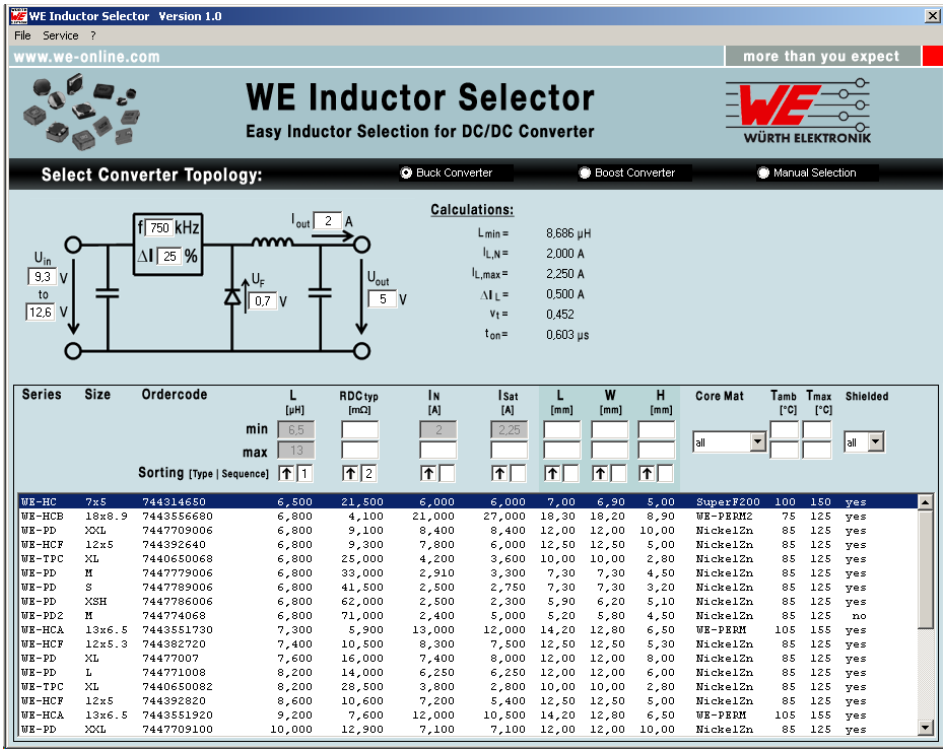


Abbildung 3.4.: Dimensionierung der Speicherspule für die Spannungsversorgung

Die Abschaltung des Systems durch U_q (*OUT*-Ausgang des Timers) erfolgt ungefähr 5*s* nach Einschalten des Timers.

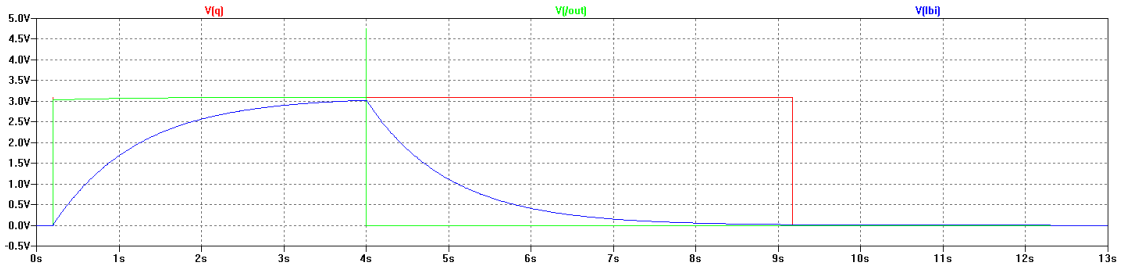


Abbildung 3.6.: Simulationsergebnis für $U_{/Out}$, U_q und U_{LBI}

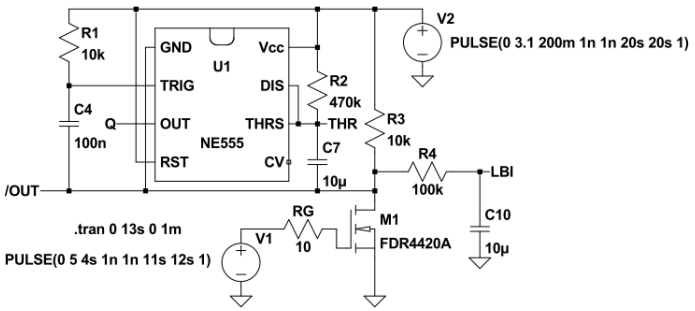


Abbildung 3.5.: Simulationsschaltung des LiPo-Monitors in *LTSpice IV*

Die Simulation des Spannungsreglers *LT3500* ergab wie erwartet folgende Ergebnisse (siehe Abbildungen 3.7 und 3.8):

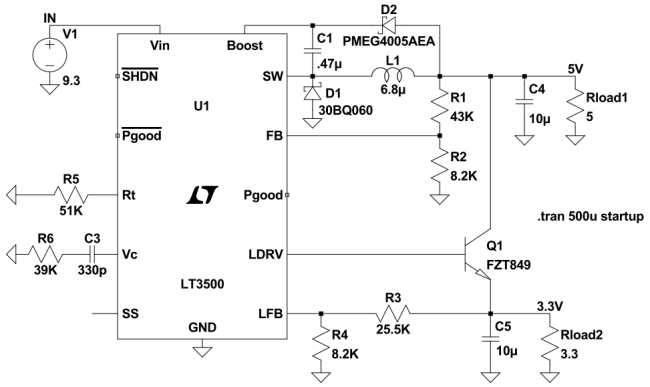


Abbildung 3.7.: Simulation der Spannungsversorgung in *LTSpice IV*

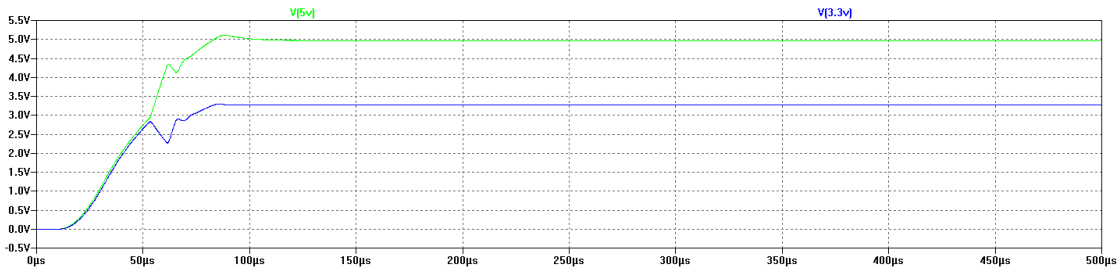


Abbildung 3.8.: Simulationsergebnis für U_{5V}, $U_{3,3V}$

3.2. Schussvorrichtung

Im Gegensatz zu den Robotern der RoboCup Middle Size League (MSL), bei denen meistens pneumatische Schussmechanismen im Einsatz kommen, werden in der RoboCup SSL ausschließlich elektronische Schussvorrichtungen verwendet. Diese bestehen grundsätzlich aus 3 Komponenten:

DC/DC-Konverter zur Erzeugung einer Hochspannung

Kapazität zur Zwischenspeicherung der erzeugten Hochspannung

Hubmagnet zur Umwandlung der in der Kapazität gespeicherten elektrischen Energie in kinetischer Energie

Aufgrund des kompakten Aufbaues und des geringen Gewichtes, bietet der DC/DC-Konverter eine gute Lösung. Die verwendeten Hubmagnete wurden vom vorhergehenden Roboter übernommen und die Schaltung darauf dimensioniert.
Die hier entwickelte Schussvorrichtung bietet die Möglichkeit 2 Hubmagnete anzusteuern und somit einen horizontalen und einen Chip-Kicker (Flanken) zu realisieren.

3.2.1. Schaltungsbeschreibung

Die Schaltung der Schussvorrichtung besteht aus 3 Teilen (siehe Abbildung 3.9):

Aufwärtswandler zur Erzeugung und Speicherung der Energie mit der geschossen wird

Schaltstufe zum Kurzschließen der gespeicherten Energie über das Hubmagnet

Entschärfer zur Vernichtung der gespeicherten Energie beim Ausschalten

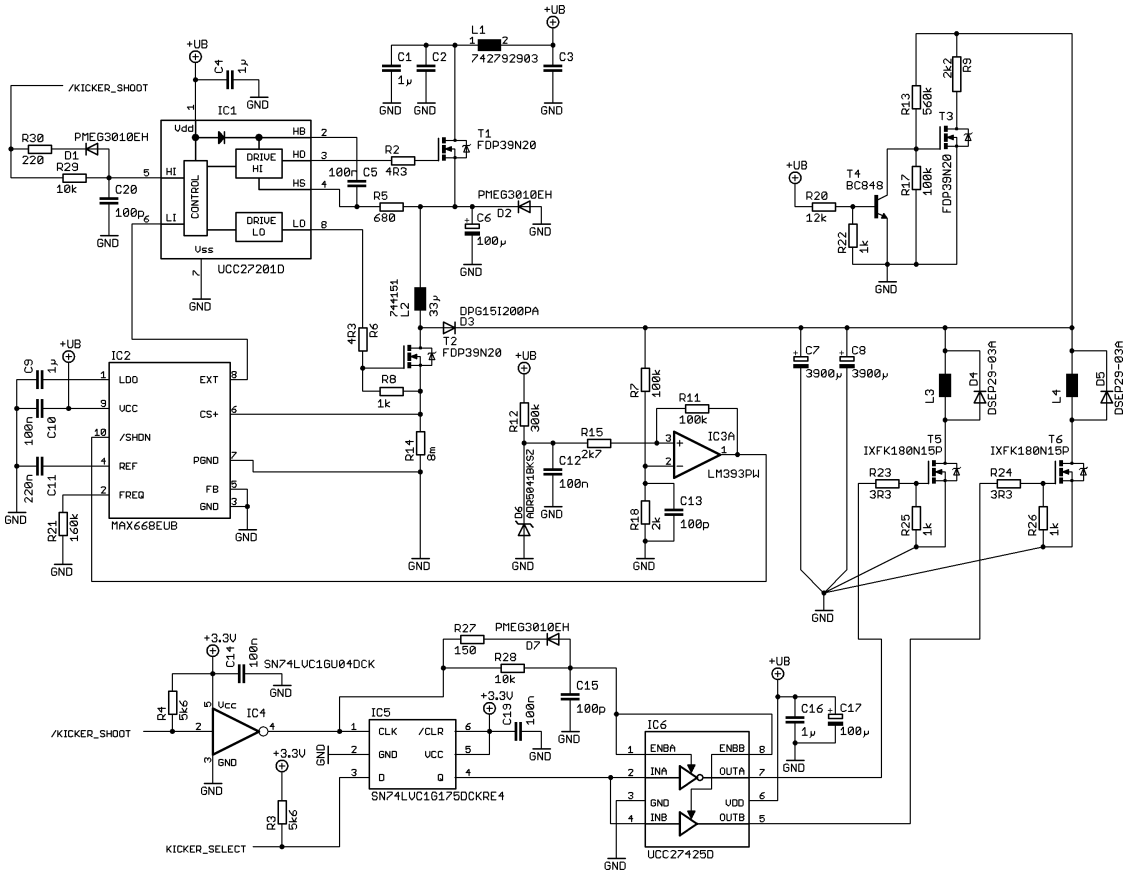


Abbildung 3.9.: Schaltung der Schussvorrichtung

Im Folgenden werden die Teilschaltungen genau beschrieben.

3.2.1.1. Aufwärtswandler

Der Aufwärtswandler wird mit einer Versorgungsspannung von $9,3V$ bis $12,6V$ betrieben $(+UB)$und erzeugt eine Ausgangsspannung von $140V$. Als PWM-Kontroller wird ein *MAX668*[33] (IC_2) verwendet. Die konstante Schaltfrequenz des Aufwärtswandlers lässt sich anhand des Widerstandes R_{21} einstellen und beträgt:

$$\begin{aligned} f_S &= \frac{5 * 10^{10}}{R_{21}} \\ &= \frac{5 * 10^{10}}{160k\Omega} = 312,5kHz \end{aligned} \tag{3.2}$$

Wird der *MAX668* im Boost-Modus betrieben, so wird die Ausgangsspannung U_{Out} des Aufwärtswandlers über einen Spannungsteiler am Feedback-Eingang *FB* eingestellt. Würde man den Spannungsteiler auf eine Ausgangsspannung von $U_{Out} = 140V$ dimensionieren, so würde der *MAX668* die Kondensatoren C_7 und C_8 zwar genau auf $140V$ aufladen, doch die Aufladezeit würde sich von $8s$ auf ungefähr $25s$ erhöhen. Der Grund dafür ist, dass der Duty Cycle des PWM-Signals zur Ansteuerung des MOSFETs T_2 sich mit zunehmender Ausgangsspannung U_{Out} verkleinert, um keine höhere Ausgangsspannung als die eingestellten $140V$ zu riskieren. Um den Ladevorgang zu beschleinigen, d.h. den Duty-Cicle des PWM-Signals maximal zu halten (90%), wird *FB* mit Masse verbunden. Man erreicht somit, dass U_{Out} die PWM-Generierung nicht beeinflusst (siehe 3.10).

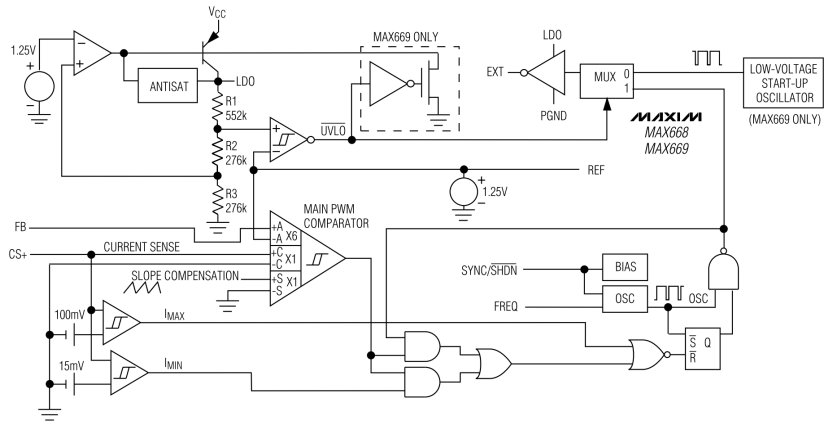


Abbildung 3.10.: Blockschaltbild des *MAX668*[33]

Die Aufgabe von *FB* erledigt stattdessen eine Komparatorschaltung mit Hystherese. Diese hat die Aufgabe, den *MAX668* dann abzuschalten, wenn $U_{Out} \approx 140V$ ist, indem */SHDN=Low* gesetzt wird. Durch die Mitkopplung am R_{11} wird eine Hystherese erzeugt, die dafür sorgt, den *MAX668* nur dann wieder einzuschalten, wenn die Ausgangsspannung um einige Volt gefallen ist. Damit wird Energie gespart, da der *MAX668* die Ausgangsspannung nicht andauernd stabilisieren muss. Als Referenzspannung für die Komparatorschaltung wurde ein *ADR5051*[6] (D_6) mit $U_{Ref} = 2,5V$ gewählt und daraus ergeben sich dann folgende Ein- und Ausschaltspannungen:

$$\begin{aligned} U_{S_{Aus}} &= 0V && \text{Ausgangsspannung des } IC_{3A} \text{ bei } /SHDN = Low \\ U_{S_{Ein_{Max}}} &= 12,6V && \text{max. Ausgangsspannung des } IC_{3A} \text{ bei } /SHDN = High \\ U_{S_{Ein_{Min}}} &= 9,3V && \text{min. Ausgangsspannung des } IC_{3A} \text{ bei } /SHDN = High \end{aligned}$$

$$\begin{aligned} U_{Ein} &= U_{S_{Aus}} \frac{R_{15}}{R_{15}+R_{11}} + U_{Ref} * \frac{R_{11}}{R_{15}+R_{11}} \\ &= U_{Ref} \frac{R_{11}}{R_{15}+R_{11}} \qquad (3.3) \\ U_{Aus_{Max}} &= U_{S_{Ein_{Max}}} \frac{R_{15}}{R_{15}+R_{11}} + U_{Ein} \qquad (3.4) \\ U_{Aus_{Min}} &= U_{S_{Ein_{Min}}} \frac{R_{15}}{R_{15}+R_{11}} + U_{Ein} \qquad (3.5) \end{aligned}$$

Um U_{Ein} und U_{Aus} nicht zu weit ausseinander zu halten, muss $R_{11} \gg R_{15}$ sein. Mit $R_{11} = 100k\Omega$ und $R_{15} = 2,7k\Omega$ ergeben sich folgende Ein- und Ausschaltspannungen:

$$\begin{aligned} U_{Ein} &= 2,5V \frac{100k\Omega}{2,7k\Omega + 100k\Omega} = 2,43V \\ U_{Aus_{Max}} &= 12,6 \frac{2,7k\Omega}{2,7k\Omega + 100k\Omega} + 2,43V = 2,77V \\ U_{Aus_{Min}} &= 9,3 \frac{2,7k\Omega}{2,7k\Omega + 100k\Omega} + 2,43V = 2,68V \end{aligned}$$

Sobald am R_{18} eine Spannung von $2,77V$ anliegt, soll der Komparator */SHDN* auf *Low* setzen. Daraus ergeben sich folgende Widerstandswerte für R_7 und R_{18}:

$$\begin{aligned} \frac{U_{Aus_{Max}}}{U_{Out}} &= \frac{R_{18}}{R_{18}+R_7} \qquad (3.6) \\ \frac{2,77V}{140V} &= \frac{R_{18}}{R_{18}+R_7} \end{aligned}$$

Es wird $R_7 = 100k\Omega$ gewählt und daraus ergibt sich

$$R_{18} = \frac{\frac{2,77V}{140V}100k\Omega}{1-\frac{2,77V}{140V}} = 2018\Omega \Rightarrow 2k\Omega$$

Aus diesen errechneten Werten ergeben sich folgende Ausgangsspannungen U_{Out}:

$$\begin{aligned} & U_{Out_{Ein}} = 123,93V \\ \text{für } U_B = 12,6V: \quad & U_{Out_{Aus}} = 141,27V \\ \text{für } U_B = 9,3V: \quad & U_{Out_{Aus}} = 136,68V \end{aligned}$$

An U_{Out} sind Störungen im Bereich der Schatfrequenz f_S zu erwarten. Diese werden durch den Tiefpassfilter R_7C_{13} mit der Grenzfrequenz $f_{G_{R_7C_{13}}} = 15915Hz$ gefiltert.
Um den Sense-Widerstand R_{14} und die Speicherdrossel L_2 zu dimensionieren wurde angenommen, dass der Aufwärtswandler die gewünschte Ausgangsspannung innerhalb von $8s$ erzeugen muss. Bei einer Kapazität von insgesamt $7,8mF$ und einer Ausgangsspannung von $140V$ ergibt sich somit einen Ausgangsstrom von ungefähr $650mA$.
Das, von der Firma Würth Elektronik, bereitgestellte Programm *WE Inductor Selector 1.0* zur Dimensionierung von Speicherdrosseln für Auf- und Abwärtswandler wurde dann dazu benutzt, die richtige Spule aus dem Würth-Elektronik-Sortiment zu finden (siehe Abbildung 3.11).
Aus den errechneten Speicherdrosseln wurde dann die WE-HCB *7443556680* ausgewählt. Diese SMD-Hochstrominduktivität wurde bzgl. Kernverlusten optimiert und die Wicklung der Spule erfolgt mit Flachdraht, was folgende Vorteile mit sich bringt[13]:

- **große Drahtoberfläche** dadurch geringe Hochfrequenzverluste (Skineffekt)
- **kleine Wicklungskapazität** somit höhere Eigenresonanzfrequenz
- **kleiner Gleichstromwiderstand** dadurch geringe Eigenerwärmung bei hohen Dauerströmen
- **hohe Packungsdichte** somit kleinere Baugröße als vergleichbare Drosseln mit Runddraht
- **hohe Betriebstemperatur** bis max. +150°C

Das Programm liefert weiters auch den maximalen Strom durch die Spule $I_{L_{2_{Max}}}$, der zur Berechnung von R_{14} benötigt wird:

$$\begin{aligned} R_{14} &= \frac{85mV}{I_{L_{2_{Max}}}} \\ &= \frac{85mV}{11,325A} = 7,5m\Omega \Rightarrow 8m\Omega \end{aligned} \tag{3.7}$$

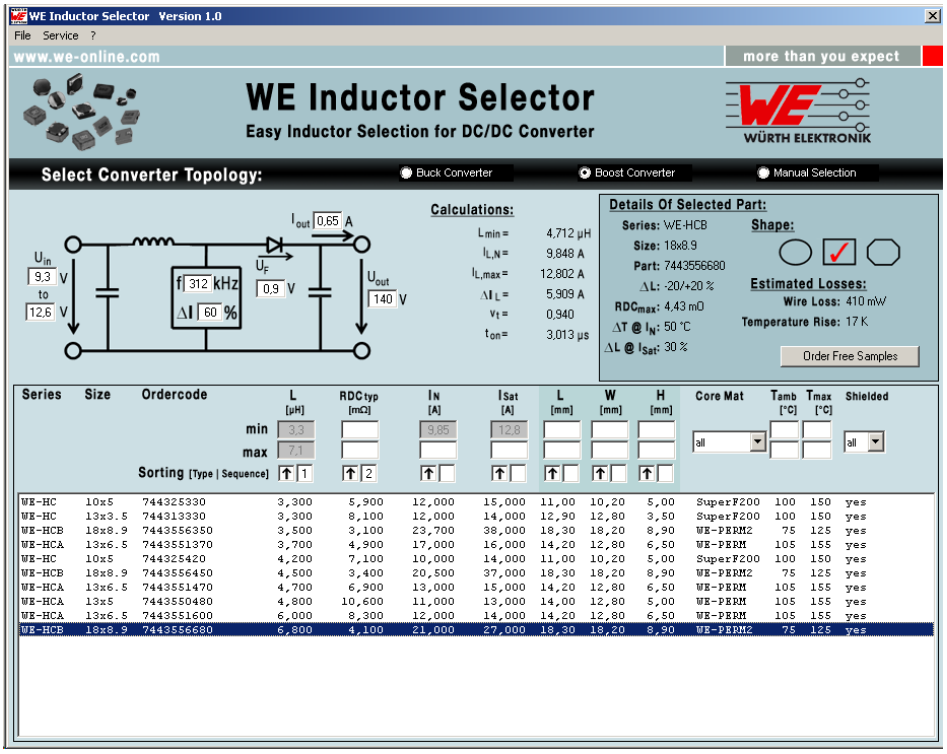


Abbildung 3.11.: Dimensionierung der Speicherdrossel für die Schussvorrichtung

3.2.1.2. Schaltstufe

Die Schussvorrichtung kann anhand zweier Signale gesteuert werden:

/KICKER_SHOOT wenn *Low* dann werden C_7 und C_8 über den ausgewählten Hubmagnet entladen. Wenn *High* dann werden diese aufgeladen.

KICKER_SELECT wenn *Low* dann werden C_7 und C_8 beim Schießen über L_3 entladen. Wenn *High* dann werden sie über L_4 entladen

Bevor einen der Hubmagnete aktiviert wird, d.h. */KICKER_SHOOT=Low*, wird die Versorgungsleitung für den Aufwärtswandler am High-Side NMOS-Transistor T_1 unterbrochen. Dies soll verhindern, dass ein hoher Querstrom durch L_2, D_3 und den ausgewählten Hubmagnet fließt, wenn $U_{Out} < U_B - U_{D_3}$ wird. Dieser hohe Strom würde die LiPo-Zellen zu stark belasten und U_B würde zusammenbrechen.

Um während des Schießens nicht zwischen den Hubmagneten wechseln zu können, wird der Zustand von *KICKER_SELECT* mittels eines D-Flip-Flops *SN74LVC1G175* [23] (IC_5) abgespeichert. Der Ausgang von IC_5 wird mit dem MOSFET-Treiber IC_6 verknüpft, der weiters die NMOS-Transistoren T_5 und T_6 ansteuert. Der RC-Tiefpass $R_{28}C_{15}$ bildet gemeinsam mit den Schmitt-Trigger-Eingänge $ENB_{A/B}$ von IC_6 ein Verzögerungsglied, das sicherstellen soll, dass T_1 geöffnet ist, bevor T_5 oder T_6 geschlossen wird. Die genaue Verzögerungszeit zwischen fallender Flanke an */KICKER_SHOOT* und steigender Flanke an $OUT_{A/B}$ kann folgendermaßen berechnet werden:

$t_{PD_{IC_4}} = 1,1ns$ Verzögerungszeit des Inverters IC_4

$t_{PD_{IC_6}} = 30ns$ Verzögerungszeit des MOSFET-Treibers IC_6

$t_{PD_{R_{28}C_{15}}}$ Aufladezeit von C_{15} von $0V$ bis zum *High*-Pegel an IC_6

$$t_{PD_{Shoot}} = t_{PD_{IC_4}} + t_{PD_{R_{28}C_{15}}} + t_{PD_{IC_6}} \tag{3.8}$$

Die Verzögerungszeit $t_{PD_{R_{28}C_{15}}}$ lässt sich anhand der Ladekurve für Kondensatoren berechnen:

$$\begin{aligned} U_{IN\_H_{Min_{IC_6}}} - U_{D_7} &= 3,3V\left(1 - e^{\left(-\frac{t_{PD_{R_{28}C_{15}}}}{R_{28} * C_{15}}\right)}\right) \\ \Rightarrow t_{PD_{R_{28}C_{15}}} &= -R_{28}C_{15}\ln\left(1 - \frac{U_{IN\_H_{Min_{IC_6}}} - U_{D_7}}{3,3V}\right) \\ &= -10k\Omega * 100pF * \ln\left(1 - \frac{1,7V - 0,7V}{3,3V}\right) = 361ns \end{aligned} \tag{3.9}$$

Somit beträgt $t_{PD_{Shoot}}$:

$$t_{PD_{Shoot}} = 1,1ns + 361ns + 30ns = 392,1ns$$

Die Zeit, die vergeht, bis IC_1 den High-Side-MOSFET T_1 öffnet beträgt ungefähr $105ns$. Ähnlich wie oben, lassen sich die Verzögerungszeiten der Schaltstufen bei steigender Flanke an */KICKER_SHOOT* berechnen. In diesem Fall vergehen ca. $115ns$ bis T_5 und T_6 geöffnet sind und $390ns$ bis T_1 wieder geschlossen wird. Somit ist sichergestellt, dass T_1 nie leitend ist, wenn entweder T_5 oder T_6 leitend sind.

3.2.1.3. Entschärfer

Aus sicherheitstechnischen Gründen, sollen C_7 und C_8 entladen werden, sobald der Roboter abgeschaltet wird. Um das zu bewerkstelligen, wird die Transistorschaltung

rund um T_3 und T_4 verwendet. Der NPN-Transistor T_4 lässt den NMOS-Transistor T_3 solange offen, bis die Betriebsspannung U_B abgeschaltet wird. Ab diesem Zeitpunkt liegt am Gate von T_3 eine maximale Spannung von $U_{GSS} = 21.2V$ an ($U_{GSS_{Max}} = \pm 30V$) und T_3 entlädt C_7 und C_8 über R_9 solange, bis $U_{GSS} = 3V$ beträgt. Die Ausgangsspannung nach dem Entschärfen (nach ungefähr $35s$, siehe Abbildung 3.12) beträgt noch $U_{Out_{Off}} = 19.8V$ und stellt für den Benutzer keine Gefahr mehr dar.

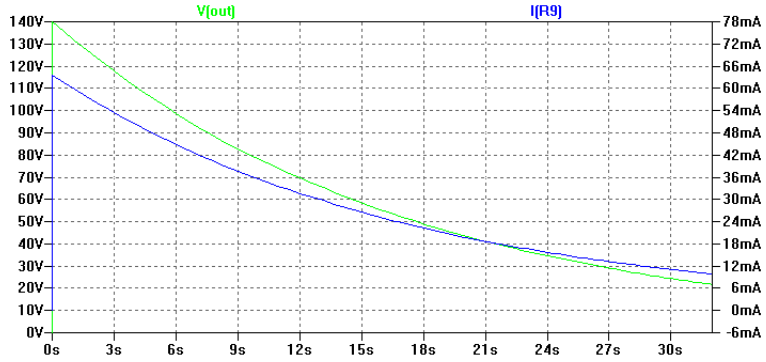


Abbildung 3.12.: Entschärfung der Schussvorrichtung

3.3. Hauptprozessor

Die zentrale Steuereinheit des Roboters besteht aus dem leistungsstarken Blackfin-Prozessor *BF527*[10] von Analog Devices. Dieser stellt die Kommunikation zwischen PC und Roboter über das Funkmodul her (siehe Kapitel 3.5), führt die notwendigen Algorithmen zur Berechnung der einzelnen Drehzahlen für die Motortreiber aus und steuert die einzelnen Module des Roboters, wie z.B. Dribbler, Schussmechanismus, Motortreiber, Lichtschranken, usw.
Da die Beschaltung des *BF527* sehr aufwendig ist (BGA-Gehäuse mit 208 Anschlüssen, Flash- und RAM-Speicher müssen extra angeschlossen werden), wurde das Core-Modul *CM-BF527*[42] der Firma *Bluetechnix*[3] verwendet (siehe Abbildung 3.13). Neben den $8MB$ Flash- und $32MB$ RAM-Speicher besitzt das *CM-BF527* weiters einen $10/100MBit$ Ethernet-PHY, eine 16-Bit PPI (Parallel Port Interface), eine *USB2.0*-Schnittstelle, einen On-Board Core-Spannungsregler und einen Reset-Baustein.

[3] Bluetechnix Mechatronische Systeme - www.bluetechnix.com

Abbildung 3.13.: Das Core-Modul *CM-BF527*[42] von *Bluetechnix*

Das Ganze ist auf einem Modul der Größe $31,5mm$x$36,5mm$ untergebracht[42].
Das fertige Core-Modul ist zusätzlich mit dem, von Bluetechnix entwickelten, *BLACKSheep*-Bootloader[4] geflasht mit dem eigene Applikation gebootet werden können.
In Abbildung 3.14 sehen Sie die Beschaltung des Core-Moduls *CM-BF527* am Roboter und in der darauffolgenden Tabelle 3.1 ist die Funktion der verwendeten Pins beschrieben.

Zwei IO-Expander *MCP23S17*[17] (IC_2 und IC_3) von Microchip Technology Inc.[5] wurden verwendet um 32 zusätzlichen GPIOs dem Core-Modul zur Verfügung zu stellen (siehe Abbildung 3.15). Die beiden IO-Expander werden über die *SPI*-Schnittstelle angesteuert und benötigen nur eine einzige *CS*-Leitung (SPI Chip-Select) des *CM-BF527*. Diese zusätzlichen GPIOs können als Ein- und Ausgänge konfiguriert werden. Die als Eingänge konfigurierten GPIOs können auf Flanken und Pegeln reagieren und Interrupts auslösen, die dann über die Leitungen *BF527_PH11/12/13/14* an den *BF527* für die Weiterverarbeitung geleitet werden.

[4] VDK BLACKSheep - www.bluetechnix.at
[5] Microchip Technology Inc. - www.microchip.com

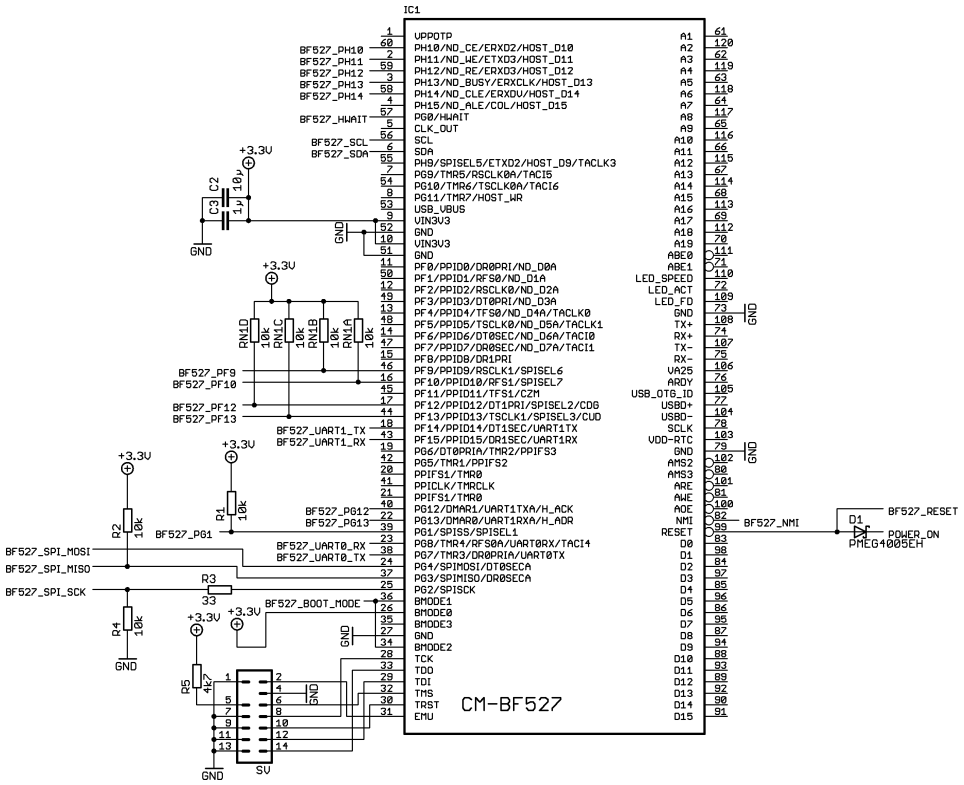


Abbildung 3.14.: Beschaltung des Core-Moduls *CM-BF527*

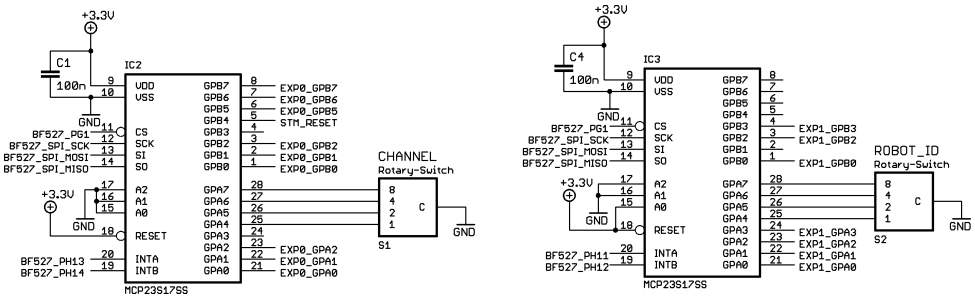


Abbildung 3.15.: Beschaltung der IO-Expander für den *CM-BF527*

An den Eingängen des *MCP23S17* können Pull-Up-Widerstände durch entsprechen-

Signal	Beschreibung
PB527_PH10	*CS*-Leitung des digitalen Potentiometers an den Lichtschranken.
PB527_PH11/12/13/14	verbunden mit den Interrupt-Ausgängen der IO-Expander
BF527_HWAIT	verbunden mit dem Funkmodul
BF527_SCL/SDA	I^2C-Schnittstelle für die Bewegungssensorik.
BF527_PF9/10/12/13	*CS*-Leitungen der Motortreiber
BF527_UART1_TX/RX	kann mit der seriellen Schnittstelle der Motortreiber verbunden werden.
BF527_PG12	verbunden mit *DR*-Ausgang des Funkmoduls
BF527_PG13	verbunden mit *RTS*-Leitung des Funkmoduls
BF527_PG1	*CS*-Leitung der IO-Expander
BF527_UART0_TX/RX	Serielle Schnittstelle zum Funkmodul
BF527_MOSI/MISO/SCK	SPI-Schnittstelle zu den Motortreibern
BMODE0/1/2/3	Boot Konfiguration
TCK/TDO/TDI/TMS/TRST/EMU	*JTAG*-Schnittstelle
BF527_RESET	Reset-Eingang des Moduls. Modul kann vom Funkmodul oder vom Reset-Taster resetiert werden.
BF527_NMI	Nicht maskierbarer Interrupt-Eingang. Verbunden mit *POWER_LBO* der Spannungsversorgung.

Tabelle 3.1.: Beschaltung des *CM-BF527*

des Konfigurieren dazugeschaltet werden. Dieses Feature wurde z.B. an den Eingängen *GPA4-7* eingesetzt um den Wert der Drehschalter S_1 (Funkkanal) und S_2 (Robot ID) zu ermitteln. In Tabelle 3.2 ist die Funktion der restlichen Expander-GPIOs erklärt.

3.4. Ballerkennung

Ausgehend von folgender (klassischen) Spielsituation:

> Roboter *A* ist im Ballbesitz. Roboter *B* steht frei und hat freie Sicht auf das gegnerische Tor. Roboter *A* soll den Ball relativ schnell an Roboter *B* passen. Roboter *B* soll sofort auf das Tor schießen, sobald er im Ballbesitz kommt.

Soll der Ball von der Kamera, die über dem Spielfeld positioniert ist durch die Bildverarbeitung am PC erkannt werden. Die Positionsbestimmung des Balls durch den Rechner und die Funkübertragung von Befehlen an den Roboter würde viel zu lange

Signal	Beschreibung
EXP0_GPA0/1/2	Ausgang der Lichtschranken. Erzeugt einen Interrupt bei Pegeländerung einer Lichtschranke.
EXP0_GPB0/1/2	schaltet die Lichtschranken ein/aus
EXP0_GPB5/6/7	Steuerleitungen für das Funkmodul
EXP1_GPA0	verbunden mit *ST*-Pin vom *ADXL322*
EXP1_GPA1/2	verbunden mit *ST1-*, bzw. *ST2*-Pin vom *EVAL-ADXRS610*
EXP1_GPA3	verbunden mit *ALERT/BUSY* vom *AD7994*
EXP1_GPB0	kann mit *STM_RESET* verbunden werden
EXP1_GPB2	kann mit *STM_UART_CHECK* verbunden werden
EXP1_GPA3	kann mit *STM_UART_SELECT* verbunden werden

Tabelle 3.2.: Beschaltung der IO-Expander

dauern, damit Roboter *B* bei erfolgtem Pass sofort auf das Tor schießen könnte. Um auf solchen Ereignissen schneller reagieren zu können ist es notwendig den Ball direkt am Roboter zu erkennen. Dazu werden Lichtschranken eingesetzt.

3.4.1. Schaltungsbeschreibung

Es wurde eine Schaltung entworfen, die die Reichweite von Lichtschranken einstellbar macht (siehe Abbildung 3.16). Die Lichtschranke besteht aus einem IR-Empfänger *IS471F* [14] (IC_1), einer Infrarot-LED *TSAL4400*[46] (LD_1) und einer einstellbaren Konstantstromquelle.
IC_1 erzeugt an $\overline{GL_{Out}}$ ein moduliertes Signal, welches zum Ansteuern einer IR-LED verwendet wird. Wird dieses modulierte Signal am Empfänger wieder erkannt, so wird V_O auf Masse gezogen. Aufgrund der geringen Pulsbreite an $\overline{GL_{Out}}$ von 6,3% kann durch LD_1 ein pulsierender Maximalstrom von $I_{LD_{1Max}} = 1,5A$ fließen. Da IC_1 an GL_{Out} lediglich 70mA liefern kann, wird die nachgeschaltete Verstärkerstufe verwendet um wesentlich höhere Ströme zu erzeugen.
Die Verstärkerstufe besteht aus einer MOSFET-Präzisionsstromquelle, die durch das digitale Potentiometer *AD5263*[4] (IC_{2A}) mit Nennwiderstand $R_{N_{IC_{2A}}} = 20k\Omega$ auf den gewünschten Ausgangsstrom eingestellt wird. Die Ausgangsspannung des OPVs *LT1631* [43] (IC_{3A}) stellt sich so ein, dass der Spannungsabfall durch den Widerstand R_B von IC_{2A} gleich dem Spannungsabfall an R_5 ist. Die Zener-Diode D_1 liefert eine Referenzspannung von $U_{Z_{D_1}} = 2,5V$ und daraus ergibt sich ein Maximalstrom durch R_5, bzw.

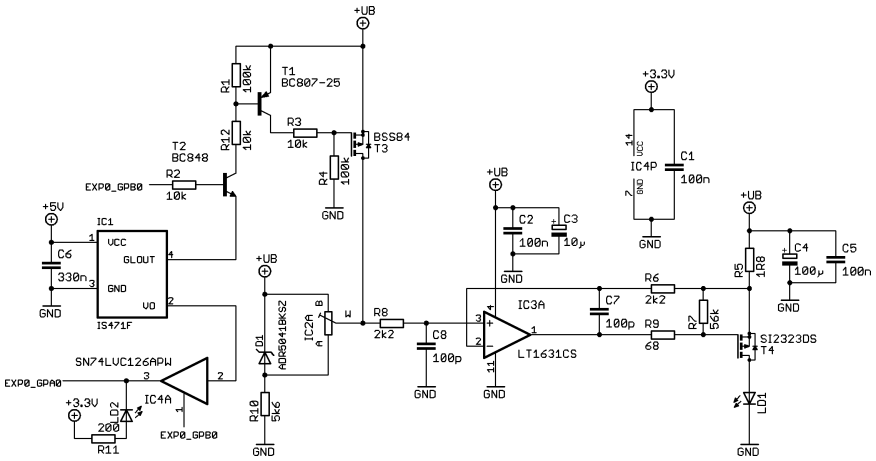


Abbildung 3.16.: Schaltung zur Ballerkennung

LD_1 von

$$I_{LD_{1_{Max}}} = \frac{U_{Z_{D_1}}}{R_5} = \frac{2,5V}{1,8\Omega} = 1,39A. \tag{3.10}$$

Es fließt nur dann ein Strom durch LD_1 wenn die Spannung am nichtinvertierenden Eingang von IC_{3A} kleiner als U_B ist und das ist nur dann der Fall, wenn T_3 offen ist[6]. Wenn IC_{2A} den kleinstmöglichen Wert für R_B eingestellt hat, d.h. $R_{B_{Min}} = 60\Omega$, dann beträgt der Spannungsabfall $U_{R_{B_{Min}}} = 7,4mV$ und der Ausgangsstrom beträgt

$$I_{LD_{1_{Min}}} = \frac{U_{R_{B_{Min}}}}{R_5} = \frac{7,4mV}{1,8\Omega} = 4,11mA. \tag{3.11}$$

Wenn T_2 oder der OC-Ausgang $\overline{GL_{Out}}$ geöffnet sind, dann ist T_3 geschlossen und es fließt kein Strom durch LD_1. Wie in Kapitel 4.2.2 beschrieben, werden die Signale $EXP_0\_GPB_{0/1/2}$ dazu verwendet, die Lichtschranken abwechselnd ein- und auszuschalten um somit eine gegenseitige Beeinflussung zu vermeiden.

[6]Man beachte, dass die Präzisionsstromquelle weitgehend unabhängig von U_B ist, d.h. I_{LD_1} ändert sich nicht wie im Falle eines Stromspiegels mit U_B

Am invertierenden Eingang von IC_{3A} wurde ein Filter bestehend aus R_6 und C_7 eingebaut, um die Stabilität der Stromquelle zu erhöhen. Da beim Messen von U_{R_5} auch ein Spannungsabfall an R_6 verursacht wird, muss U_{R_6} auch am nichtinvertierenden Eingang des IC_{3A} abfallen. Dies erreicht man durch $R_8 = R_6$. Würde man R_8 weglassen, so würde aufgrund des verfälschten U_{R_5} ein $I_{LD_1} \gg 0A$ fließen.

3.4.2. Schaltungssimulation

Die oben beschriebene Schaltung wurde in erster Linie mit *LTSpice IV* simuliert und dann durch Messungen an der Platine optimiert (siehe Abbildung 3.17). Überaschenderweise waren die Ergebnisse der Simulationen und der darauffolgenden Messungen an den Prototyp-Platinen weitgehend identisch, wodurch die Schaltungsentwicklung wesentlich vereinfachte wurde; die meisten Schaltungen konnten zuerst gründlich am Simulationsprogramm getestet werden, bevor an der Platine gearbeitet wurde. Bis auf

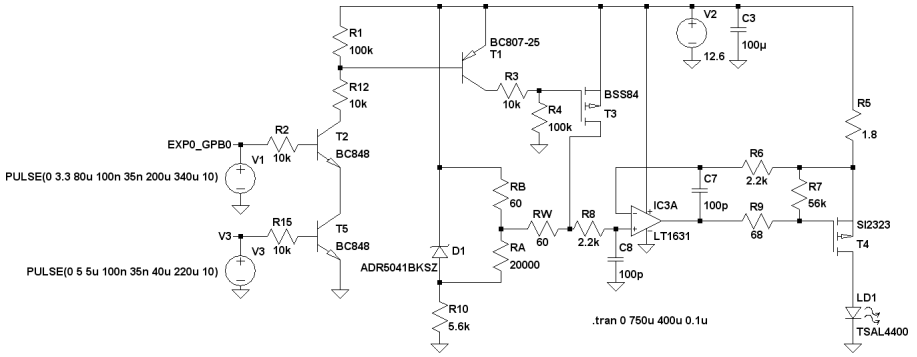


Abbildung 3.17.: Simulationsschaltung der Lichtschranken in *LTSpice IV*

dem OC-Ausgang $\overline{GL_{Out}}$, der mit den Komponenten V_3, R_{15} und T_5 modelliert und dem digitalen Potentiometer, das durch die Widerstände R_A, R_B und R_W ersetzt wurde, entspricht die simulierte Schaltung genau der tatsächlich gebauten Schaltung.
In Abbildung 3.18 sehen wir die Simulationsergebnisse für $I_{LD_{1_{Min}}}$.

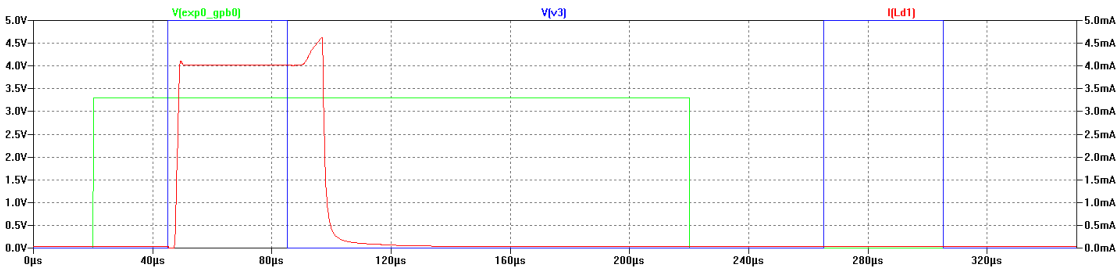


Abbildung 3.18.: Simulationsergebnis für $I_{LD_{1_{Min}}}$

Der eingestellte Strom I_{LD_1} fließt nur dann, wenn sowohl $EXP_0\_GPB_0$ als auch V_3 (entspricht $\overline{GL_{Out}}$) einen *High*-Pegel aufweisen. Wie man sieht, ist die Stromkurve I_{LD_1} sowohl beim Ein- als auch beim Ausschalten etwas verzögert. Der Grund liegt in der langsamen Ansteuerung des MOSFETs T_3[7]. Da aber dieser verzögerte und verlängerte Puls am *IS471F* trotzdem richtig erkannt wird, gibt es keinen guten Grund, die Schaltung durch aufwendigere Maßnahmen zu verbessern.
Im Folgenden sind noch die Simulationsergebnisse für $I_{LD_{1_{Max}}}$ und $I_{LD_1} = 700mA$ abgebildet (siehe Abbildungen 3.19 und 3.20).

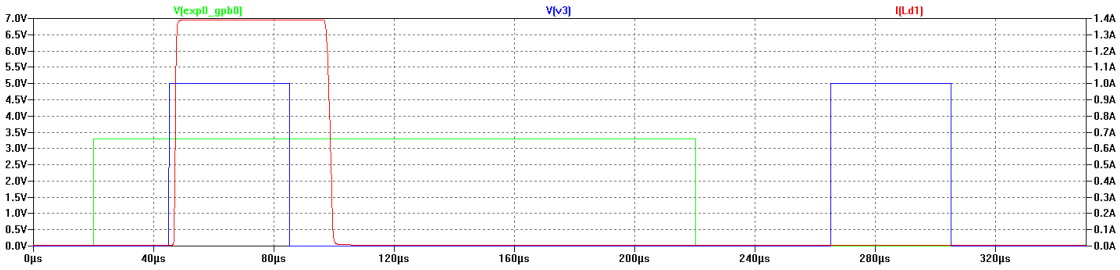


Abbildung 3.19.: Simulationsergebnis für $I_{LD_{1_{Max}}}$

[7] Man könnte die Schaltgeschwindigkeit des T_3 erhöhen, indem man R_3 und R_4 durch einen NPN-Transistor zum Einschalten ersetzt (MOSFET-Treiber)

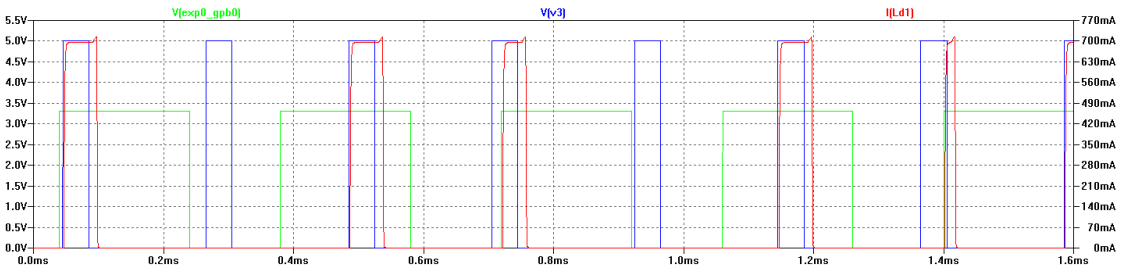


Abbildung 3.20.: Simulationsergebnis für $I_{LD_1} = 700mA$

3.4.3. Messungen und Erkenntnisse

Mit der verwendeten Schaltung kann die Reichweite der Lichtschranke zwischen $1cm$ und $1,5m$ eingestellt werden (siehe Abbildungen 3.21 und 3.25). Das Verhältnis zwischen Reichweite und Ausgangsstrom I_{LD_1} beträgt ungefähr $1\frac{\text{mm}}{\text{mA}}$. Somit wäre die maximale Reichweite der Lichtschranke ohne Verstärkung (also nur mit *IS471F*) ungefähr $7cm$. Durch die Verstärkung mittels Präzisionsstromquelle wurde die Reichweite somit um das 20-fache erhöht.

3.5. Funkmodul

Das Funkmodul stellt die Verbindung zwischen PC und Roboter her und wurde als aufsteckbare Platine gebaut um es durch zukünftigen Weiterentwicklungen austauschbar zu machen.
Folgende Regeln gelten bei der RoboCup Small Size League beim Einsatz von Funkmodulen:

- Das Komitee muss über die verwendete Funktechnik, Sendeleistung und Funkfrequenz informiert werden
- Es müssen mindestens 2 unterschiedliche Funkfrequenzen am Funkmodul einstellbar sein um eventuelle Interferenzen mit anderen Teams zu verhindern
- Die verwendete Funktechnik muss in dem Land, wo die Spiele stattfinden, dem vorgeschriebenen Normen entsprechen
- Bluetooth ist nicht erlaubt

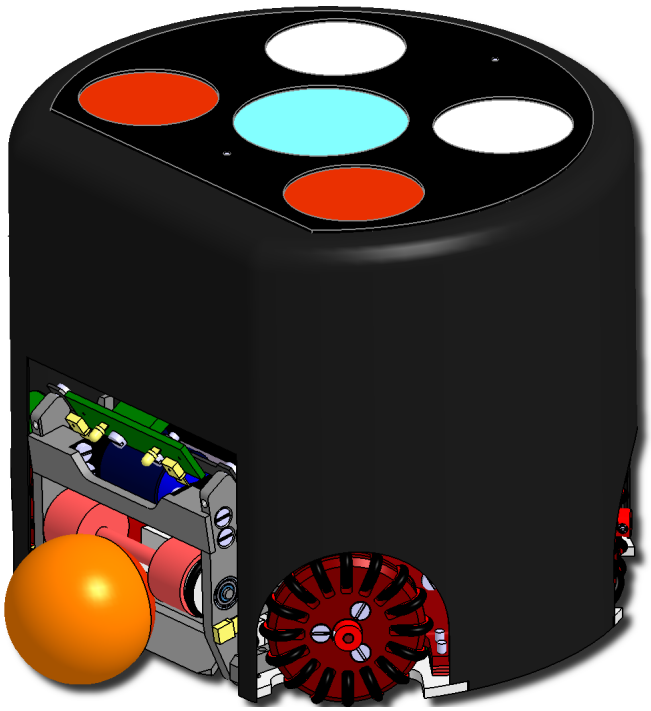


Abbildung 3.21.: Position der Lichtschranken am Roboter (grüne Platine mit gelben Komponenten)

Aufgrund des recht hohen Aufwandes, ein Funkmodul samt Funkprotokoll zu entwickeln, wurde ein fertiges Modul der Firma *Amber Wireless*[8] verwendet (siehe Abbildung 3.22).
Das verwendete ISM-Modul *AMB2520* [47] ist eine kompakte und kostengüstige Lösung zur drahtlosen Halbduplex-Kommunikation. Der integrierte Mikrokontroller *MSP430F1232* [21] steuert gemeinsam mit dem HF-Controller *CC2500* [18] die Datenkommunikation und übernimmt die Paketbildung, CRC-Prüfsummenbildung, Überwachung des Kanalzugriffs, Adressierung und das wiederholte Senden nicht quittierter Pakete. Eine Möglichkeit zur Bewertung der Qualität der Funkstrecke mittels gemessener Feldstärke (*RSSI*-Wert) ist ebenfalls vorhanden [3]. Das Funkmodul kann in folgenden Betriebsarten verwendet werden [47]:

Transparente, gepufferte Datenübertragung In diesem Modus werden Daten über die

[8] Amber Wireless - www.amber-wireless.de

serielle Schnittstelle empfangen und zwischengespeichert. Sobald eine entsprechende Bedingungen erfüllt ist, erfolgt die Bildung des HF-Telegramms mit Präambel, Prüfsumme und Adressinformationen (optional).

Kommandomodus Dieser Betriebsmodus dient primär der Konfiguration des Moduls, erlaubt aber auch die Übertragung von Nutzdaten über Funk.

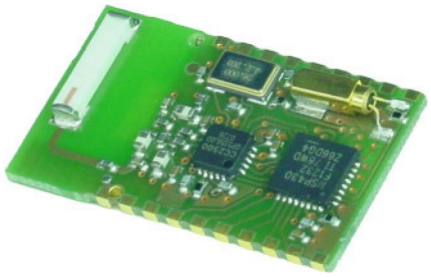


Abbildung 3.22.: Das Funkmodul *AMB2520*

Das Modul wird am Roboter im Kommandomodus betrieben da man damit die volle Kontrolle über das Funkmodul übernimmt und Parameter schnell und einfach gesetzt werden können. Das Funkmodul bietet weiters die Möglichkeit bis zu 165 unterschiedliche Funkkanälen einzustellen und Point-to-Point-, Point-to-Multipoint- und Peer-to-Peer-Verbindungen durch die Vergabe von IP-Adressen herzustellen. Die Anbindung des *AMB2520* an ein Host-System erfolgt über die UART-Schnittstelle mit Datenraten bis zu $115.2kBaud$.

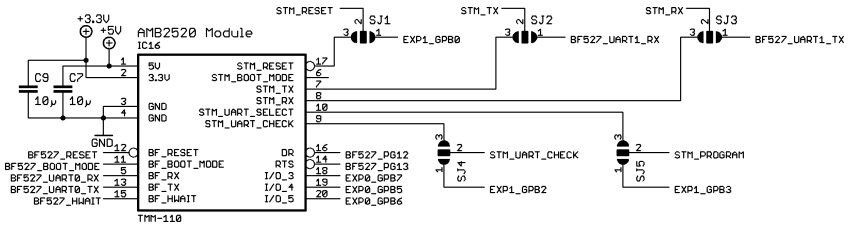


Abbildung 3.23.: Schnittstelle zum Funkmodul

Das *AMB2520*-Modul wurde direkt über die Kommunikationsschnittstelle (siehe Tabelle 3.3) mit dem Hauptprozessor verbunden. Weiters wurden zwei Low-Current LEDs LED_1 und LED_2 an den Leitungen $\overline{RTS}$ und $\overline{DATA\_INDICATE}$ hinzugefügt um das Senden und Empfangen der Daten zu visualisieren.

Die Kommunikationsschnittstelle wurde so festgelegt, dass sie neben einiger Steuerleitungen sowohl eine UART-Schnittstelle zu den Motortreibern als auch zum Hauptprozessor bietet. Diese UART-Schnittstellen bieten die Möglichkeit, die Firmware der Prozessoren mit Hilfe des Funkmoduls über Funk zu programmieren: indem man den vorprogrammierten Bootloader der entsprechenden Prozessoren startet kann man Daten über die UART-Schnittstelle ins interne Flash schreiben. Die Steuerleitungen und die UART-Schnittstelle der Motorentreiber lassen sich anhand der Jumper *SJ1* bis *SJ5* entweder mit dem Funkmodul oder mit dem Hauptprozessor (teilweise indirekt über einen I/O-Expander) verbinden. Somit können die Motortreiber sowohl vom Funkmodul als auch vom Hauptprozessor programmiert werden. Die Programmierung der Roboter über Funk wird zur Zeit des Erscheinens dieser Arbeit vom Herrn Andreas Mader am Institut für Elektronik an der TU Graz im Rahmen einer Bakkalaureatsarbeit fertig gestellt. Aus diesem Grund wird hier auf die Lösung des Problems nicht näher eingehen. Die Programmierung der Motortreiber wird im Kapitel 3.8.2.3 erläutert und die Schaltpläne des Funkmoduls finden Sie im Anhang A.6.

Pin-Name	**Beschreibung**
IO_1	General purpose I/O
IO_2	General purpose I/O
IO_3	General purpose I/O
IO_4	General purpose I/O
IO_5	General purpose I/O
STM_RESET	Motortreiber resetieren Aktiv *Low*
STM_PROGRAM	Motortreiber programmieren Aktiv *High*
STM_UART_CHECK	Motortreiber UART-Überprüfung [1]: falls UART *TX*-Pegel aller 4 Motortreiber gleich ist [0]: sonst
STM_TX	Motortreiber UART *TX*-Leitung
STM_RX	Motortreiber UART *RX*-Leitung
BF_RESET	Hauptprozessor Reset Aktiv *Low*
BF_HWAIT	Hauptprozessor *HWAIT*-Leitung
BF_BOOT_MODE	Hauptprozessor Boot-Modus [0]: Booten vom externen Flash-Speicher [1]: Booten vom UART0-Host
BF_RX	Hauptprozessor UART *RX*-Leitung
BF_TX	Hauptprozessor UART *TX*-Leitung

Tabelle 3.3.: Beschreibung der Kommunikationsschnittstelle

Am PC wird ebenfalls ein *AMB2520* verwendet, welches im transparenten Modus betrieben wird. Das Modul ist an einem USB-UART-Konverter *CP2102* [26] angeschlossen, der am Rechner über einen virtuellen COM-Port angesprochen wird (siehe Abbildung 3.24). Somit wird eine serielle Verbindung zum Roboter hergestellt.

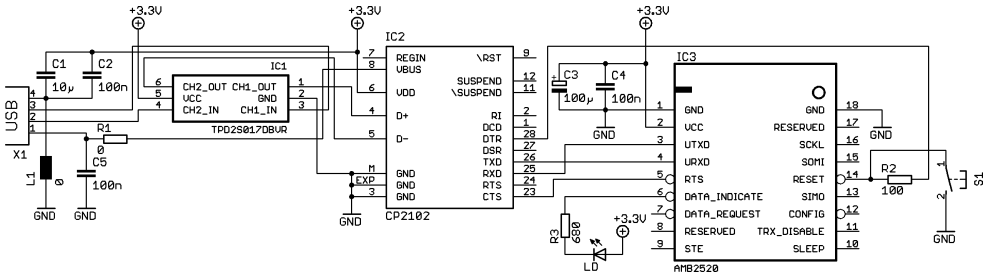


Abbildung 3.24.: Schaltung des Funkmoduls am PC

Das Funkmodul lässt sich mit dem Tool *ACC v2.6*[9] von *Amber Wireless* genau konfigurieren. Die *DTR*-Leitung der virtuellen COM-Schnittstelle wird dazu verwendet um das Funkmodul zu resetieren. Diese Leitung muss auf *Low*-Pegel gesetzt werden wenn mit dem Funkmodul kommuniziert werden soll. Ist dies von der Applikation her nicht möglich, so kann die Reset-Funktion durch Entfernen des Widerstandes R_2 abgeschalten werden. Das Modul lässt sich auch mit dem Taster S_1 resetieren.

3.6. Dribbler

Der Dribbler besteht aus einer Walze aus Gummi, die von einem DC-Motor *2224U012SR* [15] von der Firma *Faulhaber*[10] getrieben wird (siehe Abbildung 3.25). Dieser hat die Aufgabe, den Ball in Richtung des Roboters zu rollen und somit die Annahme und das Führen des Balls zu ermöglichen.
Der DC-Motor *2224U012SR* wurde zusätzlich von der Firma *ELRA*[11] mit einem Planetengetriebe mit Übersetzung 3,71 : 1 und einem Encoder mit 512 Impulse pro Umdrehung bestückt. Der Encoder wird dazu verwendet um die Drehzahl des Dribblers genau einstellen zu können.

[9] AMBER Config Center v2.6 - www.amber-wireless.de/files/acc.zip
[10] Faulhaber - www.faulhaber.com
[11] ELRA-Antriebstechnik Vertriebs Ges.m.b.H. - www.elra.at

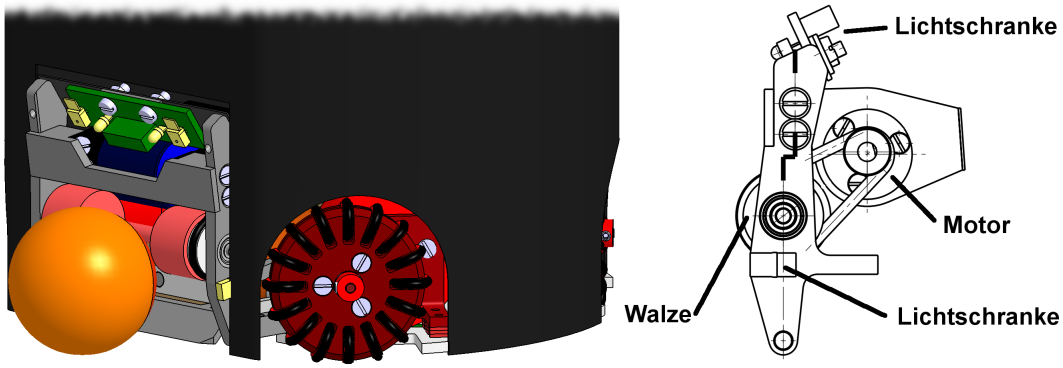


Abbildung 3.25.: Konstruktionszeichnung des Dribblers

Zum Ansteuern des *2224U012SR* wurde der Motoren-Treiber *DRV8800*[19] von Texas Instruments verwendet (siehe Abbildung 3.26). Dieser integriert eine volle H-Brücke, die einen maximalen Strom von $\pm 2,8A$ bei $36V$ liefern kann. Die Drehzahl des Motors wird anhand eines PWM-Signals an den *ENABLE*- und die Drehrichtung an den *PHASE*-Eingang des Treibers eingestellt.

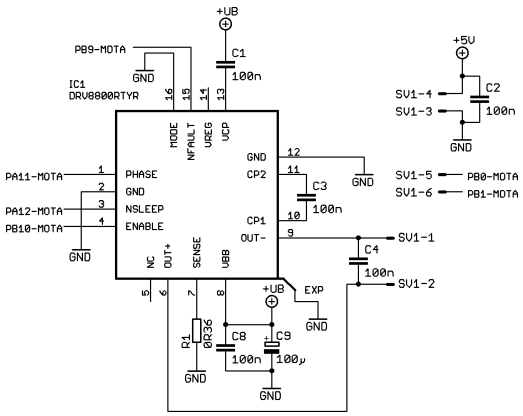


Abbildung 3.26.: Schaltung zum Dribbler

Der maximale Phasenstrom des Motors wird mit dem Widerstand R_1 eingestellt und

beträgt:

$$I_{Out_{Max}} = \frac{V_{Sense}}{R_1} = \frac{500mV}{360m\Omega} = 1,38A. \quad (3.12)$$

Da DC-Motoren aufgrund des Bürstenfeuers (siehe Tabelle 2.2) hochfrequente Störungen auf der Versorgungsleitung erzeugen, müssen diese mit Hilfe eines Kondensators C_4 an den Anschlüssen des Motors kurzgeschlossen werden. Eine bessere Entstörung kann man mit zusätzlichen Spulen an den Motoranschlüssen erziehlen, doch diese Entstörmaßnahme war in unserem Fall nicht notwendig.
Der *DRV8800* wird von einem *STM32*-Prozessor angesteuert, der zusätzlich die Regelung für einen BLDC-Motor übernimmt. Die Auswertung der Encoder-Signale vom DC-Motor wird ebenfalls vom *STM32*-Prozessor übernommen.

3.7. Bewegungssensorik

Die Bewegung des Roboters wird in erster Linie von der Kamera und der Bildverarbeitung registriert und analysiert. Aus den berechneten Trajektorien entscheidet dann die Künstliche Intelligenz wie schnell und wohin sich der Roboter bewegen soll. Die ermittelten Geschwindigkeiten in x- und y-Richtung und die Rotationsgeschwindigkeit werden über Funk an den Roboter geschickt, der die Aufgabe hat die gewünschte Geschwindigkeit und Richtung einzustellen.
Damit die gewünschte Trajektorie eingehalten werden kann, benötigt der Roboter zusätzliche Bewegungssensoren um einen Feedback zu bekommen, ob die eingestellten Drehzahlen auch zur vorgegebenen Trajektorie führen. Als primäre Quelle zur Geschwindigkeitsmessung dienen die, am Motor liegenden Hall-Sensoren, bzw. Encoder (siehe dazu Kapitel 3.8), die die Drehzahl der Motoren direkt messen. Aus den gemessen Drehzahlen aller 4 Motoren, kann die Richtung und Geschwindigkeit des Roboters berechnet werden. Doch diese berechneten Größen können auch völlig falsch sein, wenn die Räder z.B. durchdrehen.
Aus diesem Grund wurden auf dem Roboter ein Beschleunigungssensor und ein Gyro angebracht um solche Fehler zu kompensieren. Die Daten dieser Sensoren dienen dazu, die Drehzahlen der einzelnen Räder besser berechnen zu können.
Die Schaltung zur Bewegungssensorik besteht aus 3 Teilen (siehe Abbildung 3.27):

Gyro zur Messung der Drehgeschwindigkeit

Accelerometer zur Messung der Beschleunigung in x- und y-Richtung

A/D-Wandler zur Digitalisierung der analogen Ausgangsspannungen der Bewegungssensoren

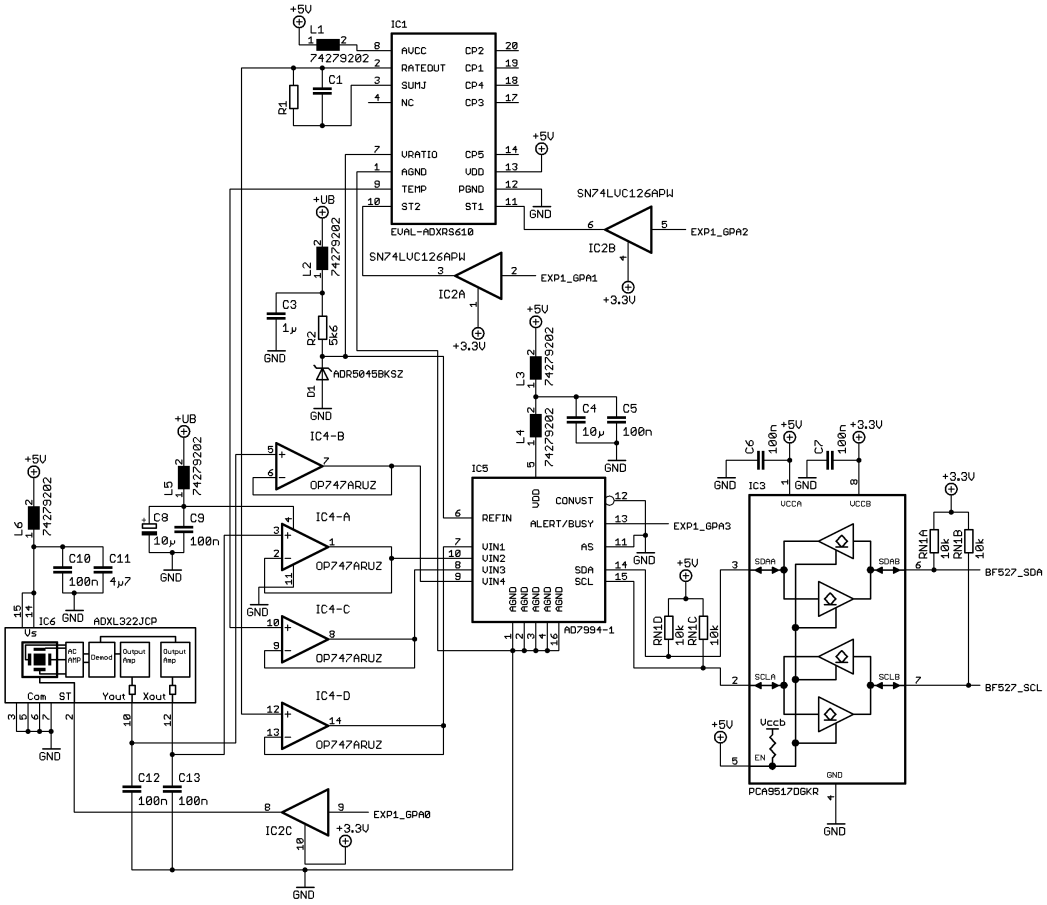


Abbildung 3.27.: Schaltung zur Bewegungssensorik

Im Folgenden werden die angeführten Teilschaltungen genau beschrieben.

3.7.1. Gyro

Als Gyro wurde ein *ADXRS610* [9] von *Analog Devices* verwendet. Da dieser Baustein nur im BGA-Gehäuse hergestellt wird und somit nicht einfach zu löten ist, wurde das Evaluierungsboard *EVAL-ADXRS610* [11] (IC_1) in DIL-20-Gehäuse eingesetzt (siehe Abbildung 3.28 und Tabelle 3.4). Am Evaluierungsboard ist die gesamte Außenbeschaltung des *ADXRS610* vorhanden und die Bandbreite, die durch R_1 und C_1 eventuell variiert werden kann, ist standardmäßig auf 9Hz eingestellt, d.h. es dürfen 9 Messungen pro Sekunde durchgeführt werden.

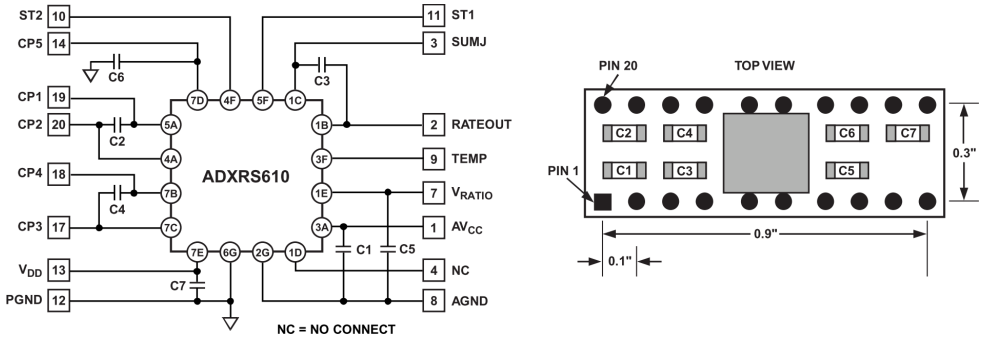


Abbildung 3.28.: Evaluierungsboard *EVAL-ADXRS610* [11]

Bauteil	Wert
C_1	100 nF
C_2	22 nF
C_3	100 nF
C_4	22 nF
C_5	100 nF
C_6	100 nF
C_7	100 nF

Tabelle 3.4.: Bauteileliste des *EVAL-ADXRS610* [11]

Der *EVAL-ADXRS610* benötigt eine Referenzspannung von $5V$, die von der hochpräzisen Spannungsreferenzquelle *ADR5045B* [7] (D_1) erzeugt wird. Damit die Referenzspannung an D_1 stabil bleibt muss der Durchlassstrom an D_1 zwischen $60\mu A$ und $15mA$

liegen. Durch die Wahl von $R_2 = 5,6k\Omega$ fließt somit ein Strom von

$$I_{D_{1_{Min}}} = \frac{U_{B_{Min}}}{R_2} \tag{3.13}$$

$$= \frac{9,3V}{5,6k\Omega} = 1,66mA \tag{3.14}$$

$$I_{D_{1_{Max}}} = \frac{U_{B_{Max}}}{R_2} \tag{3.15}$$

$$= \frac{12,6V}{5,6k\Omega} = 2,25mA \tag{3.16}$$

Die Versorgungsleitung zu D_1 wird zusätzlich anhand eines Tiefpassfilters bestehend aus dem Ferrit L_2 und dem Keramik-Kondensator C_3 gefiltert.
Die Bandbreite, bzw. die Grenzfrequenz des *EVAL-ADXRS610* wird mit Hilfe eines Tiepassfilters am Ausgang $RATEOUT$ eingestellt:

$$f_{-3dB_{IC_1}} = \frac{1}{2 * \pi * R_{Out} * C_{Out}} \tag{3.17}$$

$$= \frac{1}{2 * \pi * 180k\Omega * C_{Out}} \tag{3.18}$$

Der Ausgangswiderstand R_{Out} kann durch R_1 folgendermaßen variiert werden:

$$R_{Out} = \frac{180k\Omega * R_1}{180k\Omega + R_1} \tag{3.19}$$

C_1 liegt parallel zu C_{Out} und vergrößert somit die Ausgangskapazität.

Wie man aus dem Blockschaltbild in Abbildung 3.29 entnehmen kann, besitzt der *ADXRS610* einen internen Temperatursensor der für die Kalibrierung des Gyros verwendet werden kann. Da der Ausgang des Temperatursensors lastabhängig ist, muss dieser unbedingt mit einem Spannungsfolger weiterverarbeitet werden. Bei einer Temperatur von $25°C$ beträgt $U_{TEMP_{25°C}} = 2,5V$.
Die Funktion des Gyros lässt sich anhand der Steuersignale $ST1$ und $ST2$ überprüfen: Bei einem *High*-Pegel an $ST1$ verändert sich die Ausgangsspannung U_{RATE} um $-0,5V$ und bei einem *High*-Pegel an $ST2$ um $+0,5V$.
Der Gyro wurde genau in der Mitte des Roboter platziert um die Drehwinkelbeschleunigung bestmöglich messen zu können.

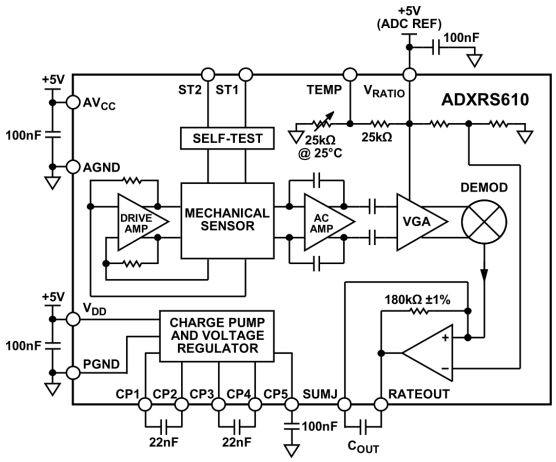


Abbildung 3.29.: Blockschaltbild zum *ADXRS610*[9]

3.7.2. Accelerometer

Als Beschleunigungssensor wurde der 2-Achsen Accelerometer *ADXL322* [8] (IC_6) von *Analog Devices* verwendet (siehe Abbildung 3.30). Der *ADXL322* kann Beschleunigungen bis $\pm 2g$ sowohl in x- als auch in y-Richtung messen. Die Bandbreite kann anhand der Kondensatoren C_{12} und C_{13} zwischen $0,5$Hz und $2,5$kHz eingestellt werden.

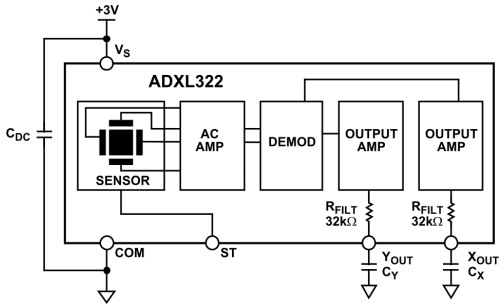


Abbildung 3.30.: Blockschaltbild des *ADXL322*[8]

Der Accelerometer liegt am Roboter in waagrechter Position, d.h. parallel zum Fußboden und somit können die Ausgänge X_{Out} und Y_{Out} Spannungen zwischen $0,57V$ und $2,42V$ liefern. In Ruhelage betragen die Ausgangsspannungen $X_{Out_{0g}} = Y_{Out_{0g}} = 1,5V$

(siehe Abbildung 3.31).
Das innovative Design des *ADXL322* ermöglicht den Betrieb über einen weiten Tempe-

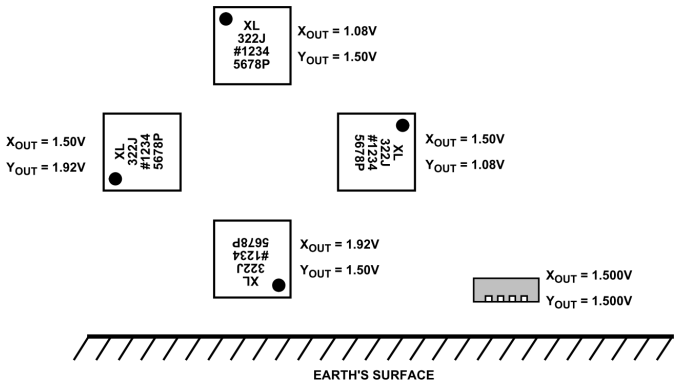


Abbildung 3.31.: Ausgangsspannungen vs. Orientierung des *ADXL322* [8]

raturbereich ohne Fehlerkorrekturen aufgrund von Temperaturschwankungen durchführen zu müssen (wie es z.B. beim *ADXRS610* der Fall ist). In Abbildung 3.32 ist der Verlauf der Ausgangsspannung von 8 verschiedenen Bauteilen bei $0g$ über einen Temperaturverlauf von $-20°C$ bis $+70°C$ aufgezeichnet.

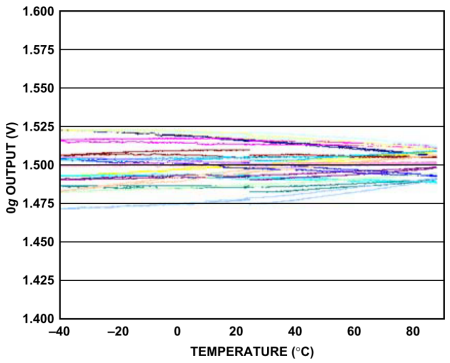


Abbildung 3.32.: Ausgangsspannung bei $0g$ vs. Temperatur des *ADXL322* [8]

Die Grenzfrequenz wird genauso wie im Falle des *ADXRS610* durch den Ausgangswiderstand $R_{Filt} = 32k\Omega$ des *ADXL322* und den Kondensatoren C_Y und C_X eingestellt

und beträgt

$$f_{-3dB_{IC_6}} = \frac{1}{2 * \pi * R_{Out} * C_{12,13}} \tag{3.20}$$

$$= \frac{1}{2 * \pi * 32k\Omega * 100nF} = 49,74Hz. \tag{3.21}$$

3.7.3. A/D-Wandler

Da der Hauptprozessor *BF527* keinen internen A/D-Wandler besitzt, muss dieser extern dazugeschaltet werden. Die Voraussetzungen für einen geeigneten A/D-Wandler sind folgende:

- Auflösung von mindestens $2mV$ (Auflösung des *ADXL322*) auf einer Skala von $0 - 5V$, d.h. mindestens 12-Bit Auflösung
- unipolare Referenzspannung von $5V$
- 4 Kanäle

Der *AD7994* [5] (IC_5) von *Analog Devices* erfüllt alle genannten Voraussetzungen und stellt somit die Schnittstelle zwischen Hauptprozessor und Bewegungssensorik über eine schnelle I^{2}C-Schnittstelle (bis max. 3,4MHz Clock-Frequenz) dar (siehe Abbildung 3.33).

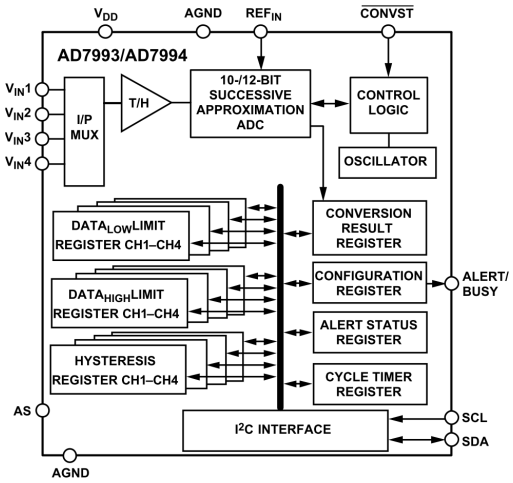


Abbildung 3.33.: Blockschaltbild des *AD7994* [5]

Alle analogen Eingangsspannungen werden von einem Spannungsfolger, bestehend aus einem *OP747* [12] (IC_4) von *Analog Devices*, entlastet. Falls die analogen Eingangsspannungen die am *AD7994* eingestellten Grenzen überschreiten, dann wird das am Pin *ALERT/BUSY* signalisiert. Weiters kann dieser Pin auch dazu benutzt werden um den aktuellen Konvertierungsstatus anzuzeigen.
Der *AD7994* muss über einen Pegelkonverter (*PCA9517* [31] (IC_3)) mit dem Hauptprozessor verbunden werden, da zwischen der $5V$- und $3,3V$-Domäne kommuniziert wird.

3.8. Antrieb

Als Antrieb werden 4 bürstenlose Gleichstrommotoren (BLDC-Motoren) verwendet an denen sogenannte omnidirektionale Räder (Omniwheels) montiert sind. Ein omnidirektionaler Antrieb ermöglicht die Bewegung in beliebiger Richtung (0° bis 360°) bei beliebiger Orientierung. Mit dieser Art von Antrieb kann sich ein Roboter um die eigene Achse drehen während er in einer bestimmten Richtung fährt. Die Positionierung des Roboters wird dadurch beschleunigt, da er sich nicht zuerst drehen muss um anschließend von A nach B zu gelangen. Ein omnidirektionaler Antrieb bietet weiters dem Tormann die Möglichkeit den Ball zu fixieren, während er sich in jeder x-beliebigen Richtung bewegt [27].
Ein Omniwheel besteht aus einem Rad, das aktiv von einem Motor getrieben wird und vielen kleinen Querräder, die sich passiv in orthogonaler Richtung zum Rad drehen können (siehe Abbildung 3.34).

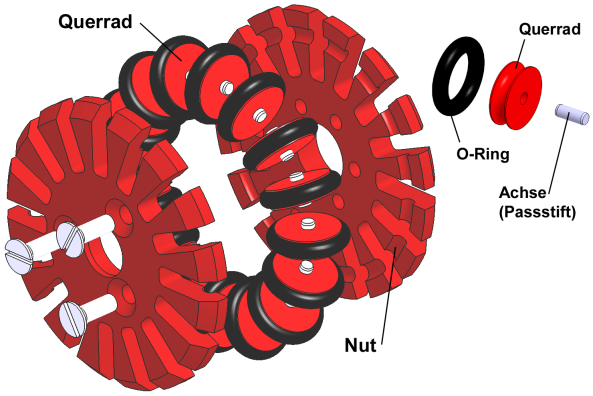


Abbildung 3.34.: Explosionszeichnung eines Omniwheels

Um die 3-phasige Kommutierung erzeugen und eine genaue Regelschleife für BLDC-Motoren implementieren zu können wurde der Cortex-M3-Prozessor *STM32F103C8T6* [40] von *STMicroelectronics* verwendet.

3.8.1. BLDC-Motor *EC 45 flat*

Aufgrund des Platzmangels am Roboter wurde der *EC 45 flat* [**?**] von *maxon motor*[12] gewählt (siehe Abbildungen 3.35 und 3.36). Der *EC 45 flat* zeichnet sich aufgrund seiner flachen Bauweise, hohen Leistung ($30W$ bei $12V$ Versorgungsspannung), hohes Drehmoment ($260mNm$) und eingebauten Hall-Sensoren aus und ist daher für diese Anwendung besonders gut geeignet.

[12] maxon motor - www.maxonmotor.com

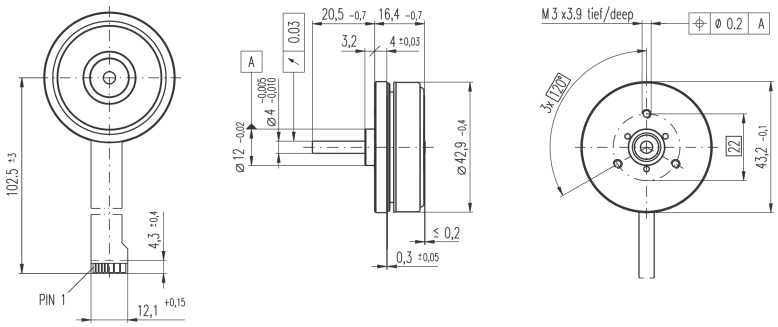


Abbildung 3.35.: Maße des *EC 45 flat* [**?**]

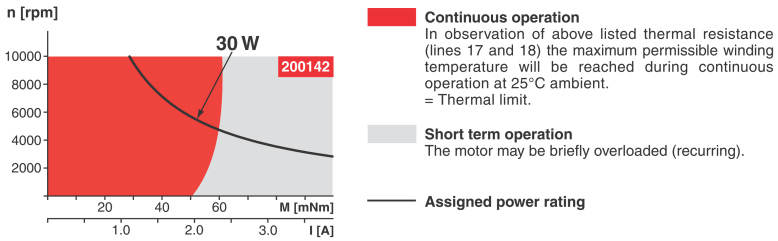


Abbildung 3.36.: Arbeitsbereich des *EC 45 flat* [**?**]

Die mehrpoligen *maxon motor*-Flachmotoren benötigen für eine Motorumdrehung eine höhere Anzahl Kommutierungsschritte (6 x Anzahl Polpaare). Sie weisen aufgrund der bewickelten Statorzähne eine höhere Anschlussinduktivität als Motoren mit eisenloser Wicklung auf (somit ist die Gegen-Elektromotorische Kraft höher)[30].

3.8.2. Schaltungsbeschreibung

Die Schaltung zur BLDC-Motorsteuerung ist etwas komplexer und deshalb werden jeweils Teilschaltungen betrachtet und sie im Detail analysiert.
Für die Signalleitungen des *STM32* gilt folgende Namenskonvention:

Signalname: Ppn-MOTm mit p...Port [A,B]
n...Pin-Nummer [0..15]
m...Motor [A,B,C,D]

3.8.2.1. Grundbeschaltung des *STM32*

Zunächst die Grundbeschaltung des *STM32F103C8T6* (siehe Abbildung 3.37):

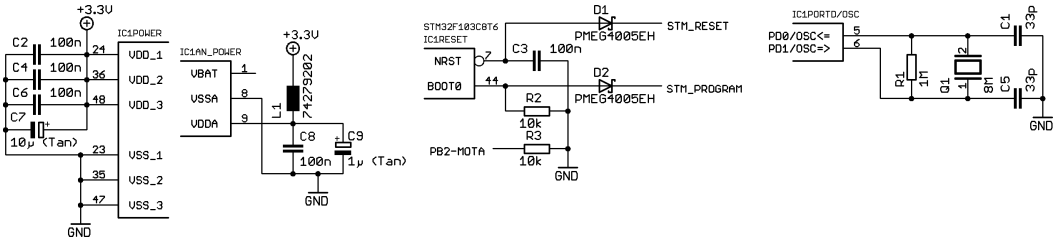


Abbildung 3.37.: Grundbeschaltung des *STM32*

Die Leitungen für die Spannungsversorgung des *STM32* werden durch Koppelkondensatoren entstört. Der Ferrit L_1 dient gemeinsam mit C_8 zur Filterung von hochfrequenten Störungen auf der Versorgungsleitung des A/D-Wandlers.
Wie in Kapitel 3.5 beschrieben, kann man dem *STM32* anhand des Signals *STM_PROGRAM* mitteilen, ob er vom Flash-Speicher oder vom Systemspeicher booten soll. Wenn der *STM32* vom Systemspeicher bootet, dann wird der vorprogrammierte Bootloader gestartet. Dieser kann Programmdaten über die Schnittstellen *USART1*, *USART2* (remapped), *CAN2* (remapped) oder *USB* entgegennehmen und sie in den internen Flash-Speicher schreiben (siehe [39]). Durch einen *Low*-Pegel an *STM_RESET* wird der *STM32* resetiert.
Die Taktfrequenz erhält der Mikrocontroller durch den externen Quarz Q_1. Die Frequenz des Quarzes (8MHz) wird dann durch die interne PLL-Schaltung des *STM32* auf 72MHz erhöht, mit der der Mikrocontroller getaktet wird.

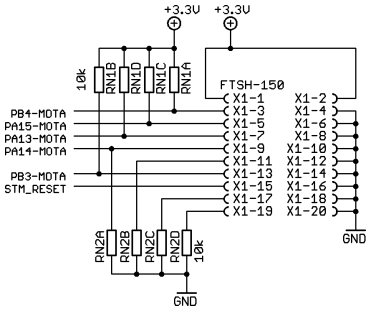


Abbildung 3.38.: JTAG-Schnittstelle am *STM32*

Für das Debuggen und Programmieren wurde eine JTAG-Schnittstelle hinzugefügt (siehe Abbildung 3.38). Als Programmiergerät wird ein J-Link [35] von Segger[13] verwendet. Aus Platzgründen wurde der JTAG-Stecker am Roboter kleiner gewählt, wodurch ein Adapter für das Programmiergerät notwendig wurde (siehe Anhang A.1).

3.8.2.2. Signalleitungen am *STM32*

In Tabelle 3.5 sind die Signalleitungen am *STM32* erklärt. Dazu wird die Beschaltung des Motortreibers für Motor *A* in Abbildung 3.39 beschrieben.

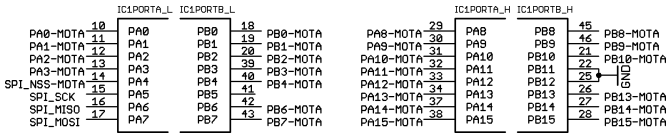


Abbildung 3.39.: Signalleitungen am *STM32*

Die Pins *PB11* und *PB12* dienen zur Konfiguration des *STM32*: je nach Beschaltung dieser Pins wird der Mikrocontroller so konfiguriert, dass er neben der Ansteuerung des BLDC-Motors noch eventuell den Dribbler oder den Kicker ansteuern kann. Der Mikrocontroller überprüft beim Starten die Pins *PB11* und *PB12* und initialisiert eventuell die Peripherie für den Kicker oder den Dribbler. Die Firmware aller Motortreiber bleibt somit identisch. Die Motortreiber und somit auch Dribbler und Kicker werden anhand eines selbst definierten Kommunikationsprotokoll über die SPI-Schnittstelle vom Hauptprozessor gesteuert.

[13] Segger - www.segger.com

Signal	Beschreibung
PA0-MOTm	Messung des Stromes in Phase *A*
PA1-MOTm	Messung des Stromes in Phase *B*
PA2-MOTm	Messung des Stromes in Phase *C*
PA3-MOTm	Messung der Motorspannung
SPI_NSS-MOTm	SPI Slave Select
SPI_CLK	SPI Clock
SPI_MISO	SPI Output
SPI_MOSI	SPI Input
PA8-MOTm	Ansteuerung des High-Side MOSFETs für Phase *A*
PA9-MOTm	Ansteuerung des High-Side MOSFETs für Phase *B*
PA10-MOTm	Ansteuerung des High-Side MOSFETs für Phase *C*
PA11-MOTm	wenn [PB12,PB11] = [0,0] → *PHASE*-Pin vom *DRV8800* wenn [PB12,PB11] = [1,0] → *KICKER_SELECT*-Pin vom Kicker ansonsten -
PA12-MOTm	wenn [PB12,PB11] = [0,0] → *nSLEEP*-Pin vom *DRV8800* ansonsten -
PA13-MOTm	JTAG-Schnittstelle
PA14-MOTm	JTAG-Schnittstelle
PA15-MOTm	JTAG-Schnittstelle
PB0-MOTm	wenn [PB12,PB11] = [0,0] → Kanal *B* vom Dribbler-Encoder ansonsten -
PB1-MOTm	wenn [PB12,PB11] = [0,0] → Kanal *A* vom Dribbler-Encoder ansonsten -
PB2-MOTm	BOOT1 Konfiguration
PB3-MOTm	JTAG-Schnittstelle
PB4-MOTm	JTAG-Schnittstelle
PB6-MOTm	Hall Sensor 1
PB7-MOTm	Hall Sensor 2
PB8-MOTm	Hall Sensor 3
PB9-MOTm	wenn [PB12,PB11] = [0,0] → *nFAULT*-Pin vom *DRV8800*
PB10-MOTm	wenn [PB12,PB11] = [0,0] → *ENABLE*-Pin vom *DRV8800* wenn [PB12,PB11] = [1,0] → *nKICKER_SHOOT*-Pin vom Kicker ansonsten -
PB12, PB11	*STM32* Konfiguration [0,0]… Zusätzliche Steuerung des Dribblers [0,1]… - [1,0]… Zusätzliche Steuerung des Kickers [1,1]… -
PB13-MOTm	Low-Side MOSFET in Phase *A*
PB14-MOTm	Low-Side MOSFET in Phase *B*
PB15-MOTm	Low-Side MOSFET in Phase *C*

Tabelle 3.5.: Beschreibung der Signalleitungen am *STM32*

3.8.2.3. Programmierung der Motortreiber über Funk

Im Folgenden werden die Mechanismen am Roboter erläutert, die die Programmierung über eine kabellose Verbindung überhaupt möglich machen.
Die Forderung war, die Firmware der 4 Motortreiber möglichst schnell zu aktualisieren und somit die zeitaufwändige JTAG-Programmierung jedes einzelnen Mikrocontrollers überflüssig zu machen. Dies ist nicht nur beim Entwickeln von Vorteil, sondern auch fürs fein-Tuning bei Wettbewerben.
Die Programmierung der Motortreiber über Funk kann auf zwei verschiedene Arten geschehen:

1. Die neue Firmware wird mit Hilfe des Tools *Flash loader demonstrator* [37][14] von *STMicroelectronics* über die virtuelle (kabellose) COM-Schnittstelle an die Motortreiber übertragen. Dabei werden die Programmdaten vom Funkmodul direkt an die 4 Motortreiber weitergeleitet und falls Programmierfehler auftreten, dann wird vom Funkmodul ein ungültiger Wert an dem *Flash loader demonstrator* zurückgeschickt, der seinerseits die Programmierung abbricht.

2. Die neue Firmware wird vom PC direkt zum Hauptprozessor übertragen. Dieser leitet die Programmdaten an den Motortreiber weiter und speichert sie zusätzlich noch in den eigenen SDRAM-Speicher. Falls Programmierfehler auftreten, dann kann der Hauptprozessor die Programmierung wiederholen, ohne dass die Firmware nochmals über Funk übertragen werden muss.

Die erste Variante wurde von Herrn Andreas Mader gewählt und erfolgreich umgesetzt. Die zweite Methode, den Hauptprozessor als Programmierer zu verwenden hat den Vorteil, dass das Funkmodul die Programmierung nicht überwachen muss. Das Funkmodul im zweiten Fall kann aus einem einzigen *AMB2520* bestehen.
Um die Programmierung der Motortreiber zu beschleunigen, wurden sie am Roboter so verbunden, dass sie alle 4 gleichzeitig geflasht werden können, ohne dass der Programmierer etwas davon mitbekommt. Wie man in Tabelle 3.3 zeigt, besteht die Programmierschnittstelle für die Motortreiber aus den Signalleitungen *STM_RESET*, *STM_PROGRAM*, *STM_UART_CHECK*, *STM_TX*, *STM_RX*. Das Programmieren funktioniert nun folgendermaßen:

1. Die 4 *STM32* werden in den Reset-Zustand gebracht indem *STM_RESET* auf *Low* gezogen wird

[14] Flash loader demonstrator - www.st.com/stonline/products/support/micro/files/um0462.zip

2. Nun wird *STM_PROGRAM* auf *High*-Pegel gesetzt. Dies bewirkt zuerst, dass die Signalleitungen *PA10-MOTm* an die *STM_RX*-Leitung angeschlossen werden (siehe Abbildung 3.40) damit Programmdaten über die *USART1*-Schnittstelle empfangt werden können. Da die Signalleitungen *PA10-MOTm* auch jeweils den High-Side MOSFET von Phase *C* ansteuern, muss garantiert werden, dass die Low-Side MOSFETs während der Programmierung nicht fälschlicherweise eingeschaltet werden, denn dann würde ein Kurzschlussstrom durch Phase *C* fließen. Um das zu vermeiden, wird *STM_PROGRAM* dazu verwendet, die Treiber der Low-Side-MOSFETs an Phase *C* (Tristate-Buffer *SN74ABT125* [22]) abzuschalten (siehe 3.40). Zum Schluss stellt noch *STM_PROGRAM* die Boot-Konfiguration des *STM32* auf das Booten vom Systemspeicher ein.

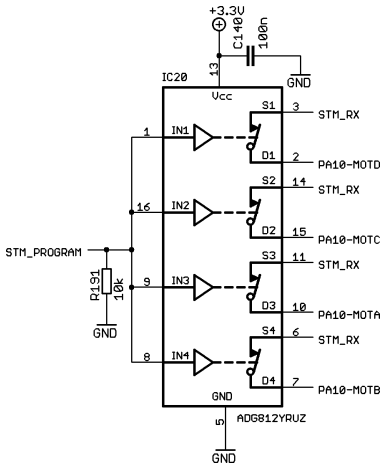


Abbildung 3.40.: Schalter für die Signalleitungen *PA10-MOTm*

3. Durch einen *High*-Pegel an *STM_RESET* wird nun der *STM32* gestartet und schreibt die über *STM_RX* empfangenen Programmdaten in den internen Flash-Speicher.

4. Nach jedem geschriebenen Programmblock, wird vom Mikrocontroller eine Check-Summe generiert, die vom Programmer (*Flash loader demonstrator* oder Hauptprozessor) auf Richtigkeit überprüft werden muss. Da aber 4 Mikrocontroller gleichzeitig programmiert werden, muss zuerst sichergestellt werden, dass alle Mikrocontroller den gleichen Rückgabewert generiert haben. Um das zu überprüfen wurde eine Logikschaltung entworfen, die *STM_UART_CHECK* auf *High*

setzt, falls die Rückgabewerte aller 4 Mikrocontroller gleich sind und auf *Low* zieht wenn die Rückgabewerte nicht alle gleich sind (siehe Abbildung 3.41 und Tabelle 3.6). Falls die Logikschaltung eine logische 1 liefert, dann wird einfach der Rückgabewert *PA9-MOTC* von Motor *C* an den Programmer zurückgeliefert.

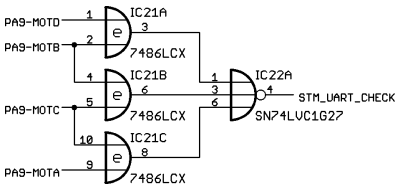


Abbildung 3.41.: Logikschaltung zum Vergleichen der *USART1-TX*-Leitungen

PA9-MOTA	PA9-MOTB	PA9-MOTB	PA9-MOTB	STM_UART_CHECK
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Tabelle 3.6.: Wahrheitstabelle zur Logikschaltung für das Vergleichen der *USART1-TX*-Leitungen

5. Die Programmdaten werden solange geliefert, bis die Firmware fertig geflasht worden ist oder ein Programmierfehler aufgetreten ist.
6. Nach erfolgter Programmierung wird *STM_RESET* wieder auf *Low* gezogen und STM_PROGRAM auf *Low* rückgesetzt.
7. Sobald *STM_RESET* wieder auf *High* geht werden die Motortreiber mit der neuen Firmware gestartet.

3.8.2.4. Sensoren-Eingang

Wie im Kapitel 2.1.3 erläutert wurde, kann eine genaue Regelung der Motoren nur mit Hilfe von Hall-Sensoren oder Encoder geschehen. Die Regelung basiert momentan auf den eingebauten Hall-Sensoren des *EC 45 flat*-Motors, doch sie lässt sich ganz einfach auf Encoder umrüsten.
In Abbildung 3.42 finden Sie die Beschaltung der Sensor-Eingänge (für die Beschreibung der Signalleitungen siehe Tabelle 3.5). Die Kondensatoren C_{21}, C_{22} und C_{23} dienen nur zur Filterung von eventuellen Störungen an den Hall-Sensoren und können beim Einsatz der Encoder-Platine weggelassen werden.

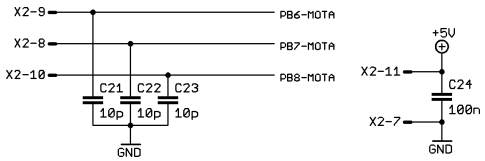


Abbildung 3.42.: Sensoren-Eingang am Motortreiber

3.8.2.5. Endstufe

Die Endstufe des Motortreibers besteht aus 6 MOSFETs mit den zugehörigen MOSFET-Treibern und 3 OPV-Schaltungen zum Messen der 3 Phasenströme des Motors (siehe Abbildung 3.43).
Die 6 MOSFETs zum Einschalten der Phasenströme bestehen aus 3 komplementäre Halbbrücken *SI4500* [45] (T_2, T_4 und T_6). Die Halbbrücken bestehen jeweils aus einem P- und einem N-Kanal-MOSFET, die getrennt steuerbar sind. Der P-Kanal-MOSFET wird von einem NPN-Transistor (T_1, T_3 und T_5) ein- und ausgeschaltet, während der N-Kanal-MOSFET von dem Logik-Buffer IC_1 mit $5V$ Ausgangsspannung angesteuert wird.

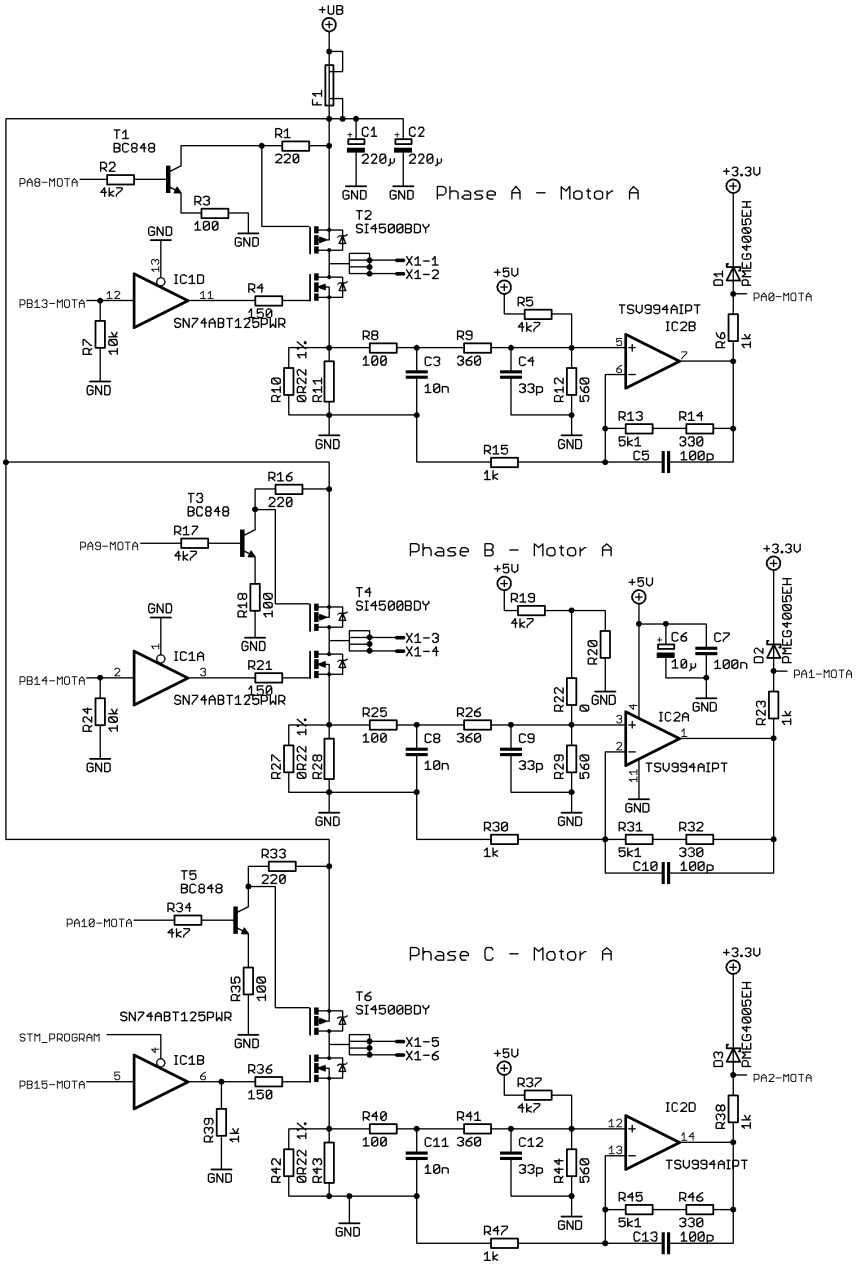


Abbildung 3.43.: Endstufe des Motortreibers

Wenn der *STM32* programmiert wird, dann schaltet dieser automatisch alle GPIOs auf Tri-State. Somit kann der High-Side-MOSFET nicht eingeschaltet werden und es können keine Kurzschlüsse an den Halbbrücken entstehen. In Kapitel 3.8.3 sieht man eine Simulation zur Ansteuerung der Halbbrücken. Die Simulation diente dazu die ungefähre Totzeit für die Ansteuerung der beiden MOSFETs zu ermitteln.
Die einzelnen Phasenströme des 3-phasigen BLDC-Motors werden mit Hilfe von Sense-Widerstanden (R_{10}, R_{27} und R_{42}) und nachgeschalteten OPV-Schaltungen gemessen. Da die abfallenden Spannungen an den Sense-Widerständen sowohl positiv als auch negativ sein können, muss sie mit Hilfe der OPV-Schaltungen in den positiven Spannungsbereich konvertiert werden, damit die A/D-Wandler des *STM32* die Spannung digitalisieren können. Um die OPV-Schaltung zu erklären wird Phase A betrachtet: Der Tiefpassfilter bestehend aus R_8 und C_3 dient zur Filterung hochfrequenter Störungen. Der maximale Phasenstrom des *EC 45 flat* bei laufendem Betrieb beträgt $I_{Mb_{max}} = 2,14A$. Dieser Maximalstrom führt an R_{10} zu einer maximalen Sense-Spannung von

$$\begin{aligned} U_{R_{10}} &= \pm R_{10} * I_{Mb_{max}} && (3.22)\\ &= \pm 220m\Omega * 2,14A \\ &= \pm 0,47V. && (3.23) \end{aligned}$$

Die OPV-Schaltung erzeugt bei einem Phasenstrom von $0A$ eine Spannung von $\frac{3,3V}{2}$, bei negativem Phasenstrom eine Ausgangsspannung zwischen $0V$ und $\frac{3,3V}{2}$ und bei positivem Phasenstrom zwischen $\frac{3,3V}{2}$ und $3,3V$. Falls der Phasenstrom größer wird als $I_{Mb_{max}}$, dann liegt eine Ausgangsspannung von $0V$, bzw $3,3V + U_{D_3}$ am A/D-Konverter des *STM32* an (siehe 3.8.3).
Der OPV *TSV994* [41] (IC_2) ist als nichtinvertierender Verstärker beschaltet und die Verstärkung beträgt:

$$\begin{aligned} A_{IC_2} &= \frac{R_{13} + R_{14} + R_{15}}{R_{15}} && (3.24)\\ &= \frac{5,1k\Omega + 330\Omega + 1k\Omega}{1k\Omega} \\ &= 6,43. && (3.25) \end{aligned}$$

Das Verhältnis zwischen $U_{R_{10}}$ und die Spannung $U_{IC_{2_+}}$ am nichtinvertierenden Eingang

kann anhand der Knoten- und Maschenregel aufgestellt werden:

$$U_{IC_{2+}} = \frac{5V}{k * R_5} + \frac{U_{R_{10}}}{k * (R_8 + R_9)} \tag{3.26}$$

$$k = \frac{1}{R_8 + R_9} + \frac{1}{R_5} + \frac{1}{R_{12}} \tag{3.27}$$

Es wurde eine Simulation dieser OPV-Schaltung mit *LTSpice* durchgeführt, die Ergebnisse werden in Kapitel 3.8.3 betrachtet.

3.8.3. Schaltungssimulation

Die Ansteuerung der Halbbrücke wurde zunächst mit *LTSpice* simuliert, damit eine ungefähre Totzeit ermittelt werden konnte, mit der man die einzelnen MOSFETs ein- und ausschalten kann. Da für die Implementierung der BLDC-Regelung eine Library von *STMicroelectronics* verwendet wurde, bei der die maximale Totzeit $3,5ms$ betragen kann, konnte man anhand der Simulation sehen, ob die dimensionierte Schaltung den Vorgaben entsprach (siehe Abbildungen 3.44 und 3.45). Die simulierte Schaltung ergab, dass die Ein- und Ausschaltzeit des High-Side-MOSFETs ungefähr $2ms$ beträgt und das ließ sich auch an der realen Schaltung nachweisen.

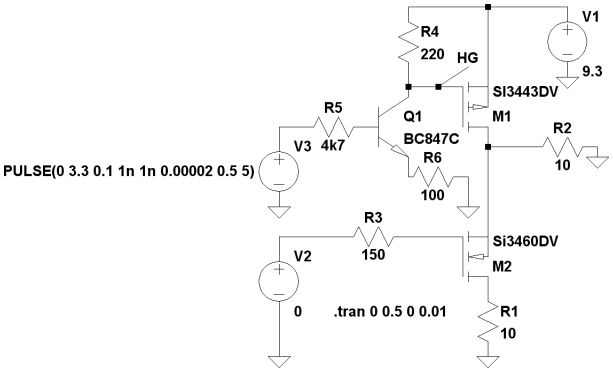


Abbildung 3.44.: Simulation zur Ansteuerung der Halbbrücke in *LTSpice IV*

Weiters wurde die OPV-Schaltung wiederum mit *LTSpice* simuliert. Daraus konnte man die Widerstandswerte der OPV-Schaltung optimieren (siehe Abbildungen 3.46 und 3.47).

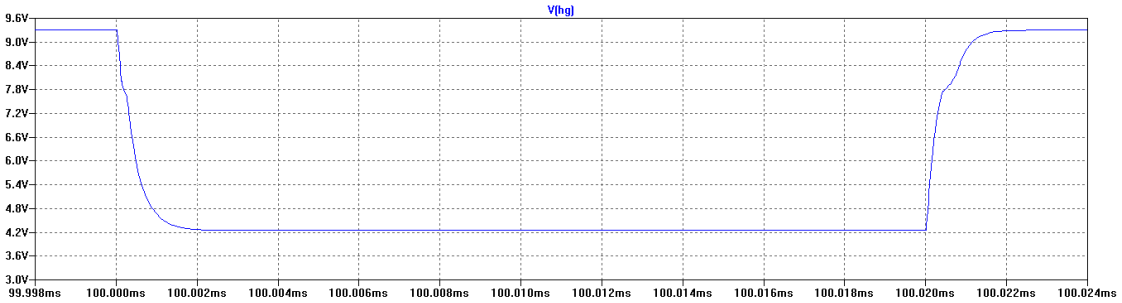


Abbildung 3.45.: Simulationsergebnis zur Ansteuerung der Halbbrücke

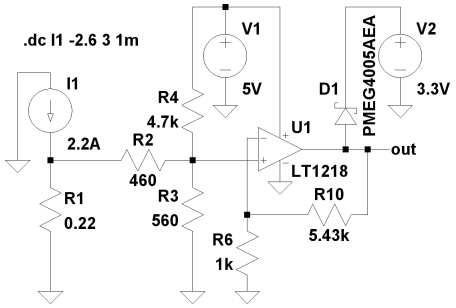


Abbildung 3.46.: Simulation der OPV-Schaltung zur Phasenstrommessung

3.8.4. Messungen und Erkenntnisse

Wie schon oben erwähnt hat sich herausgestellt, dass die Regelung anhand der Hall-Sensoren zu ungenau ist, da die Auflösung pro Umdrehung viel zu niedrig ist:

$$\frac{\#\text{Zustände}}{\text{Umdrehung}} = \#\text{Polpaare} * 2^{\#\text{Hall-Sensoren}} * \frac{1}{\text{Übersetzung}} \tag{3.28}$$

$$= 8 * 2^3 * \frac{1}{\frac{1}{2,8}}$$

$$= 179,2 \tag{3.29}$$

Mit dieser Art von Sensoren können Drehzahlen ab $70\frac{U}{min}$ geregelt werden. Um kleinere Drehzahlen zu regeln, müssen Encoder eingesetzt werden, die eine hohe Anzahl an Impulsen pro Umdrehung liefern. Die bereits verwendete Library von STMicroelectronics

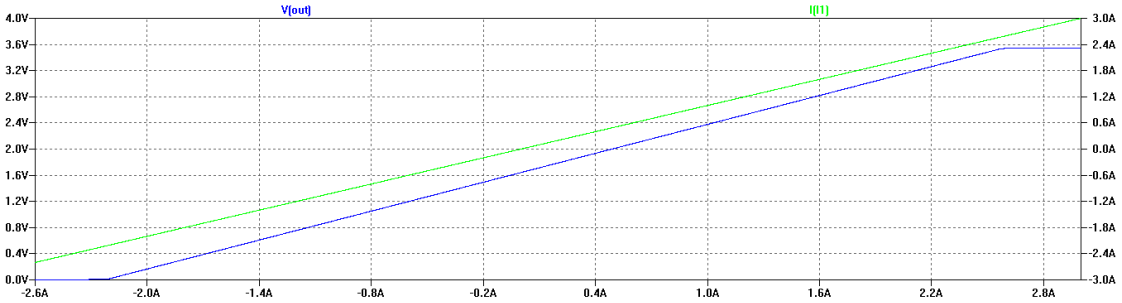


Abbildung 3.47.: Simulationsergebnis der OPV-Schaltung zur Phasenstrommessung

für die Regelung von BLDC-Motoren kann ohne viel Aufwand auf Encoder umkonfiguriert werden und die Schaltung entspricht der bereits verwendeten (es müssen nur die Encoder-Ausgänge an den Sensor-Eingängen angeschlossen werden).

4 Firmware

In diesem Kapitel wird die Firmware des Roboters beschrieben, wobei nicht auf die genaue Implementierung der einzelnen Module eingegangen wird, sondern nur ein grober Überblick der Struktur der Programme gegeben wird. Die Funktionsweise der Firmware wird anhand von Flussdiagrammen dargestellt, woraus die Interaktion der einzelnen Module zueinander erkennbar wird.
Im ersten Unterkapitel wird die Firmware der Motortreiber betrachtet. Diese baut auf die, von *STMicroelectronics* zur Verfügung stehenden Bibliothek, *STM32F103xx PMSM FOC software library V2.0* zur Ansteuerung von BLDC-Motoren auf.
Im zweiten Unterkapitel wird die Firmware des Hauptprozessors *BF527* von *Analog Devices* genauer betrachten.

4.1. Motortreiber

4.1.1. STM32F103xx PMSM FOC software library V2.0

Wie bereits erwähnt, wurde das Software-Paket *STM32F103xx PMSM FOC software library V2.0* (kurz *STM32MCLib V2.0*) von *STMicroelectronics* verwendet um die Regelung der BLDC-Motoren zu implementieren (siehe Abbildung 4.1). Diese Bibliothek wird auf CD gemeinsam mit dem Evaluierungsboard *STM3210B-MCKIT*[1] ausgeliefert oder kann nach Anfrage direkt vom STMicroelectronics-Support bezogen werden.
Diese Bibliothek implementiert verschiedene Regelalgorithmen für 3-phasige BLDC-Motoren auf Basis der *STM32F103xx*-Prozessoren. Diese 32-Bit, *ARM Cortex M3*-Prozessoren von *STMicroelectronics* sind mit spezieller Peripherie ausgestattet, die speziell für Motoransteuerungen konzipiert wurde, sei es für 3-Phasen-BLDC-Motoren mit Permanentmagneten als auch für 3-Phasen-AC-Induktionsmotoren. Die Bibliothek unterstützt lediglich die Erzeugung einer Sinus-Kommutierung (Trapez-Kommutierung ist nicht implementiert, mehr dazu in Kapitel 2.1.2), sowohl für die Drehmoment- als auch für die Geschwindigkeitsregelung.

[1] Evaluierungsboard *STM3210B-MCKIT* von STMicroelectronics - www.st.com/mcu/contentid-112-110-STM3210B_MCKIT.html

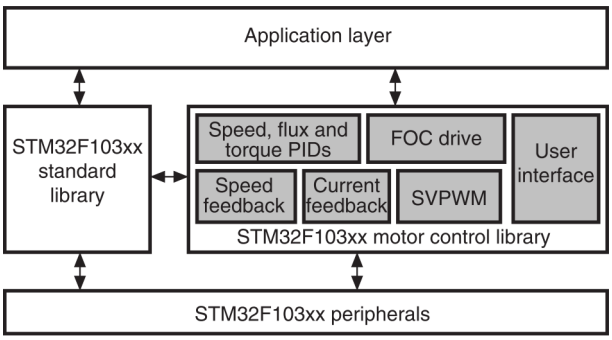


Abbildung 4.1.: Firmware-Architektur der Bibliothek *STM32MCLib V2.0*[**?**]

Folgende Features sind in der *STM32MCLib V2.0* implementiert[**?**]:

- Unterstützte Feedback-Quellen für die Geschwindigkeitsmessung:
  - Sensorlos
  - Hall-Sensoren in 60° oder 120° zueinander
  - Quadratur-Encoder
- Strommessverfahren:
  - 2 isolierte Strommesssensoren (ICS)
  - 1 gemeinsamer Shunt-Widerstand für 3 Phasen
  - 3 Shunt-Widerstände für 3 Phasen
- Optimierte IPMSM- (interior permanent magnet) und SM-PMSM-Ansteuerung (Surface-mounted permanent magnet)
- Feed-forward Stromregulierung
- Feldregelung
- Unterstützt Bremswiderstand
- Geschwindigkeitskontrolle für Geschwindigkeitsregelung
- Drehmomentkontrolle für Drehmomentregelung
- CPU-Auslastung unter 22% in 3-Shunt/sensorlose Konfiguration (bei 72MHz und 10kHz FOC Abtastrate)
- Code-Größe mit 3-Shunt/sensorlose Konfiguration bei ungefähr 12,5kB

Wie man sehen kann, kommen die meisten Features der *STM32MCLib V2.0* nur bei sensorlosen Ansteuerungen zum Einsatz. Aufgrund der geringen Serieninduktivität der BLDC-Motoren *EC 45* und der geringen Drehzahlen können die Motoren nicht sensorlos angesteuert werden. Die Geschwindigkeitsmessung muss anhand von Hall-Sensoren oder Encodern geschehen und daher kommt nur ein kleiner Teil der imple-

mentierten Regelalgorithmen der *STM32MCLib V2.0* zum Einsatz.
Die *STM32MCLib V2.0* lässt mit Hilfe des dazugehörigen Benutzerhandbuches *UM0493* [**?**] für jede beliebige Topologie leicht konfigurieren. Dabei werden die vordefinierten Makros in den Header-Dateien entsprechend gesetzt. Da es wenig Sinn macht die Konfiguration der gesamten Bibliothek zu diskutieren, sei auf den Quellcode zum Motortreiber auf der beiliegenden CD verwiesen.

4.1.2. Programmablauf am Motortreiber

Die Firmware am Motortreiber verwendet neben der bereits genannten Bibliothek *STM32MCLib V2.0* auch die Standard-Bibliothek *STM32F10xxx Firmware Library V2.0.3*[2], ebenfalls von STMicroelectronics. Letztere bietet Treiber für die Initialisierung der gesamten *STM32*-Peripherie und erleichtert somit die Softwareentwicklung am Mikrocontroller.
Wie man in Abbildung 4.2 sieht beginnt das Programm mit der Initialisierung der benötigten Peripherie-Modulen wobei auf die bereits implementierten Funktionen der *STM32F10xxx Firmware Library V2.0.3* zugegriffen wird.

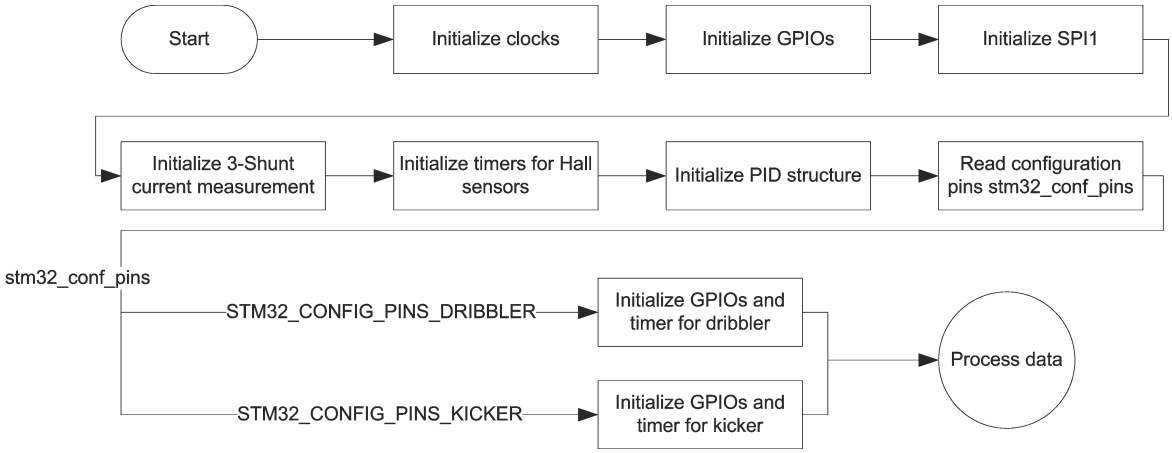


Abbildung 4.2.: Flussdiagramm zur Initialisierung der Peripherien am *STM32*

Zuerst werden die verschiedenen Clock-Frequenzen des Prozessors initialisiert, wobei

[2] STM32F10xxx Firmware Library V2.0.3 - www.st.com

die maximal zulässigen Werte eingestellt werden, um die bestmögliche Performance zu erziehlen. Im nächsten Schritt werden die GPIOs (General purpose input output) entsprechend konfiguriert und danach erfolgt die Initialisierung der SPI-Schnittstelle zum Hauptprozessor. Der *STM32* wird dabei als Slave definiert und wird daher vom Hauptprozessor gesteuert.
Ab diesem Zeitpunkt wird die Peripherie für die Ansteuerung der BLDC-Motoren initialisiert, angefangen mit dem internen A/D-Wandler, der für eine 3-Shunt-Strommessung vorbereitet wird. Daraufhin werden die an den Sensor-Eingängen intern verbundenen Timer für die Messung der Hall-Sensoren konfiguriert. Weiters werden die Strukturen des PID-Reglers der *STM32MCLib V2.0* aufgestellt und entsprechend resetiert.
Wie bereits in Kapitel 3.8.2 erklärt, werden 2 der 4 Motortreiber dazu verwendet, die Kicker-Platine und den Dribbler anzusteuern. Da die Firmware für alle Motortreiber gleich ist, wird die Konfiguration ermittelt und wenn nötig die entsprechenden GPIOs und Timers initialisiert.
Ab hier startet eine Endlosschleife, die kontinuierlich Daten vom Hauptprozessor empfängt und verarbeitet. Parallel dazu wird die Motorregelung der *STM32MCLib V2.0* Bibliothek gestartet und mit Daten versorgt (siehe Abbildung 4.3).

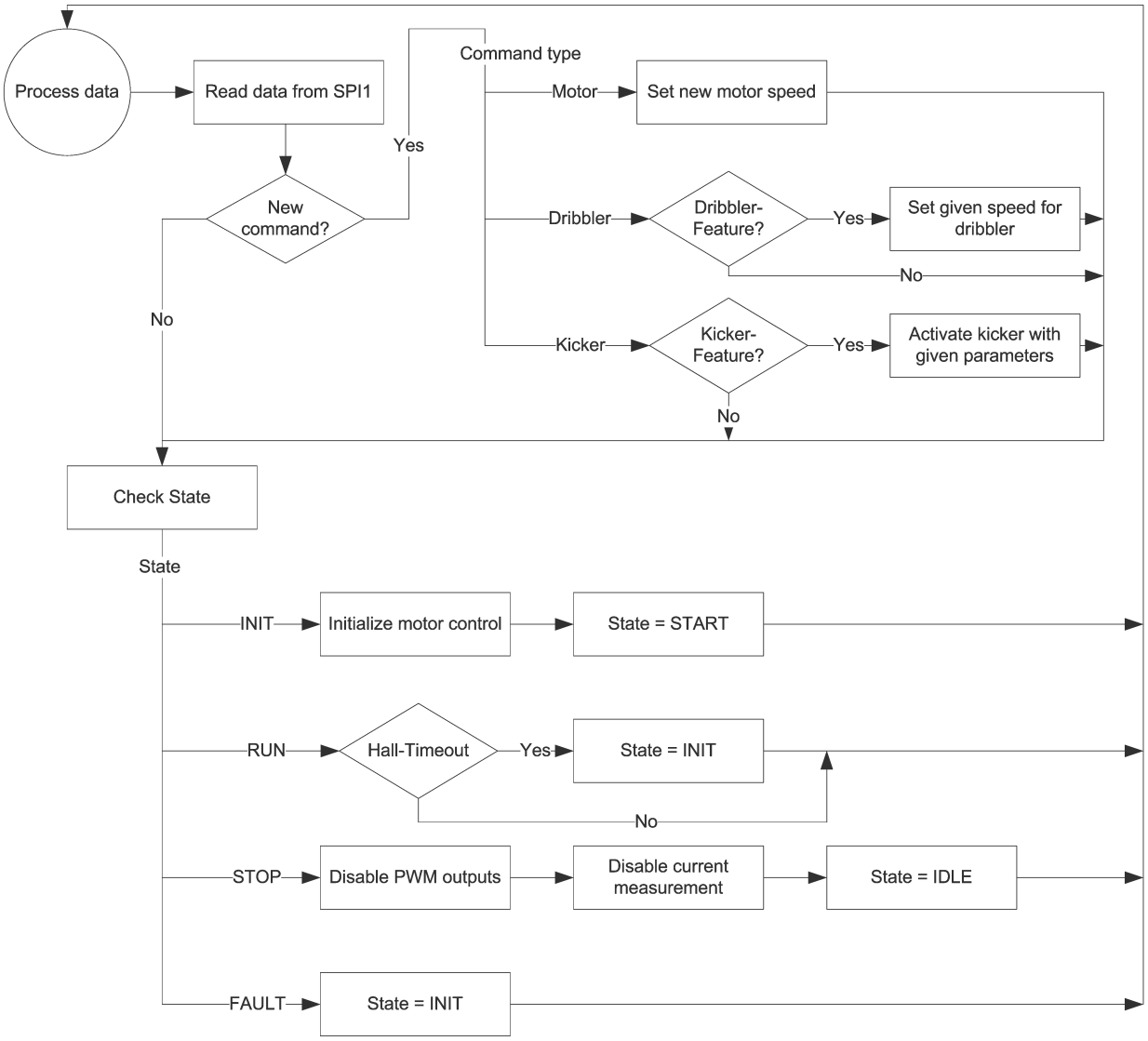


Abbildung 4.3.: Flussdiagramm zur Datenverarbeitung am *STM32*

Wie man erkennen kann, verlangt die *STM32MCLib V2.0* vom Benutzer lediglich das Umschalten zwischen den verschiedenen Reglerzuständen (State), während die Biblio-

thek den Rest erledigt.
Zu Beginn der Endlosschleife wird überprüft, ob ein neues Kommando über die SPI-Schnittstelle empfangen wurde. Wenn dies der Fall ist, wird überprüft um welchen Kommandotyp es sich handelt. Wurde ein Kommando für den Dribbler oder für den Schussmechanismus gesendet, so muss zuerst überprüft werden, ob der Motortreiber dieses Feature unterstützt (je nach Konfiguration-Pins). Wenn ja, dann wird das Kommando ausgeführt, wenn nicht, wird das Kommando ignoriert. Wird hingegen vom Hauptprozessor verlangt, das Drehmoment zu ändern, so wird das von jedem Motortreiber ausgeführt.
Im nächsten Schritt wird dann der aktuelle Status des Reglers abgefragt und entsprechend behandelt. Dieses Code-Segment wurde vom Demo-Programm der *STM32MCLib V2.0* übernommen und für diese Anwendung angepasst. Es werden demnach 4 verschiedene Zustände behandelt:

INIT Während der *INIT*-Phase wird der Regler initialisiert und wechselt in den Zustand *START*

RUN Der im Hintergrund laufende Regelalgorithmus welchselt automatisch vom Zustand *START* in den Zustand *RUN* sobald die Bewegungssensoren gültige Daten senden und man davon ausgehen kann, dass sich der Motor in Bewegung gesetzt hat. Wird während dieses Zustands ein Fehler an den Hall-Sensoren detektiert (wenn die Hall-Sensoren keine Bewegung melden, obwohl sich der Motor drehen müsste), so wird wiederum in den Zustand *INIT* gewechselt.

STOP In diesem Zustand werden sofort alle PWM-Ausgänge und die Strommessung ausgeschaltet. In diesem Zustand können sich die Motoren nur passiv drehen und es erfolgt keine Regelung.

FAULT Im Falle eines Fehlers wird (anders als im Demo-Programm) sofort wieder zur Initialisierungsphase (INIT) gewechselt.

Wie und welche Regelalgorithmen von der Library verwendet werden, kann in der Dokumentation zur *STM32MCLib V2.0*, die gemeinsam mit den Source-Codes mitgeliefert wird, nachgelesen werden.

4.2. Hauptprozessor

4.2.1. Blackfin-Entwicklungswerkzeug

Die Applikation für den Hauptprozessor wurde mit der Entwicklungsumgebung *VisualDSP++ 5.0*[3] von *Analog Devices* programmiert und als JTAG-Programmiergerät/ Debugger wurde ein *ADZS-HPUSB-ICE*[4] verwendet.
Die Firmware für den Hauptprozessor wurde in erster Linie mit dem Debugger direkt vom externen RAM-Baustein aus gestartet und getestet. Die endgültige Firmware wurde als bootfähiges Kompilat erzeugt (ladbare Datei kompiliert) und mit Hilfe des Flash-Tool-Treibers[5] von *Bluetechnix* ins externe Flash-Speicher geschrieben. Von dort werden dann im Boot-Vorgang die Programmdaten vom *BF527* geladen.
Eine Anleitung, wie das Core-Modul *CM-BF527* von *Bluetechnix* geflasht wird finden Sie unter support.bluetechnix.at/wiki/Flashing_the_Core_Module_(Blackfin).

4.2.2. Programmablauf am Hauptprozessor

Das Programm startet ähnlich wie beim Motortreiber mit der Initialisierung der benötigten Peripherie. Zunächst wird die Core-Clock-Frequenz des *BF527* auf 600 MHz und die System-Clock-Frequenz auf 133 MHz gesetzt. Diese Taktfrequenzen werden von dem angeschlossenen 25 MHz Oszillator anhand der internen PLL-Schaltung des *BF527* abgeleitet. Die Taktfrequenz und die interne Core-Spannung können zur Laufzeit verändert werden um den Stromverbrauch des Prozessors zu minimieren.
Im nächsten Schritt werden die verschiedenen GPIOs, die für die Ansteuerung der angeschlossenen ICs (z.B. Funkmodul *AMB2520*) verwendet werden, initialisiert. Daraufhin wird die Interrupt-Tabelle entsprechend konfiguriert und die Interrupt-Service-Routinen (ISRs) definiert. Es werden insgesamt 6 Interrupt-Quellen definiert:

DMA7 für die Ansteuerung der Motortreiber über die SPI-Schnittstelle mittels DMA

DMA8 für das Empfangen der Daten vom Funkmodul mittels DMA

DMA9 für das Senden der Daten zum Funkmodul mittels DMA

[3] Analog Devices VisualDSP++ 5.0 - www.analog.com/en/embedded-processing-dsp
[4] Analog Devices ADZS-HPUSB-ICE - www.analog.com/en/embedded-processing-dsp/blackfin/USB-EMULATOR
[5] Bluetechnix Flash-Tool-Treiber für CM-BF527 - www.bluetechnix.com

PH11/12/13/14 für die Behandlung der Interrupt-Anfragen der IO-Expander

PG13 zum Erkennen eines empfangenen Funktelegramms

TMR0 zum Umschalten der Lichtschranken

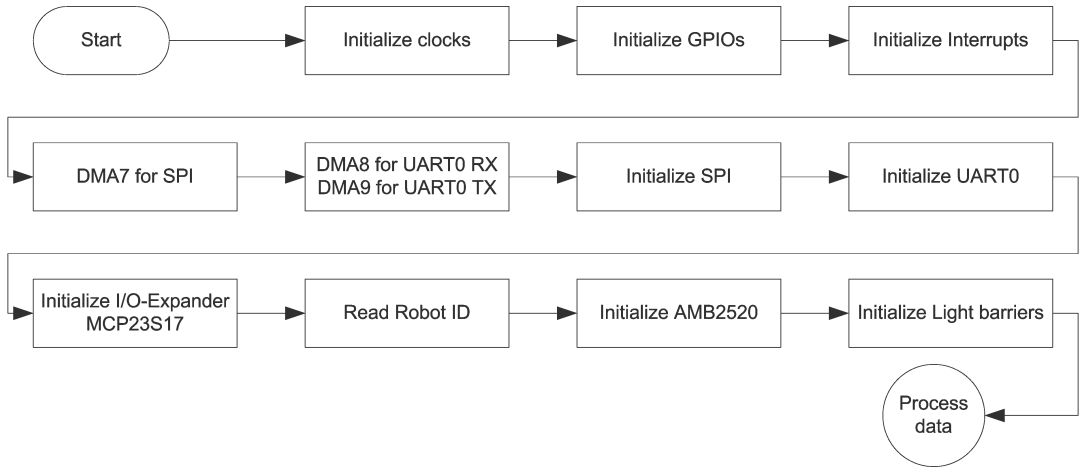


Abbildung 4.4.: Flussdiagramm zur Initialisierung der Peripherien am *CM-BF527*

Nachdem die Interrupt-Quellen definiert sind, kann die zugehörige Peripherie konfiguriert werden. Für die SPI- und UART-Schnittstelle werden die entsprechenden DMA-Kanäle mit einer Datenbreite von 8-Bit eingestellt.
Weiters wird der IO-Expander *MCP23S17* initialisiert und mit dessen Hilfe die Identifikationsnummer (ID) des Roboters eingelesen. Alle Befehle, die vom Roboter nach der Initialisierung des Funkmoduls *AMB2520* ausgeführt werden sollen, müssen diese eindeutig zugewiesene ID aufweisen, ansonsten werden sie einfach ignoriert.
Als nächstes wird das digitale Potentiometer für die Konfiguration der Lichtschranken initialisiert, mit dessen Hilfe man die Reichweite der einzelnen Lichtschranken einstellen kann. Weiters wird noch ein Timer benötigt, der nach jeder Periode die aktuelle Lichtschranke abschaltet und die Nächste in der Liste einschaltet. Somit wird sichergestellt, dass sich die Lichtschranken nicht gegenseitig stören.

Als Nächstes werden die Konstanten der Transformationsmatrix P berechnet (siehe Kapitel 2.2). Diese Konstanten werden später für die Berechnung der Drehzahlen aller 4 Motoren verwendet.
Ab hier startet eine Endlosschleife zur Verarbeitung der empfangenen Kommandos

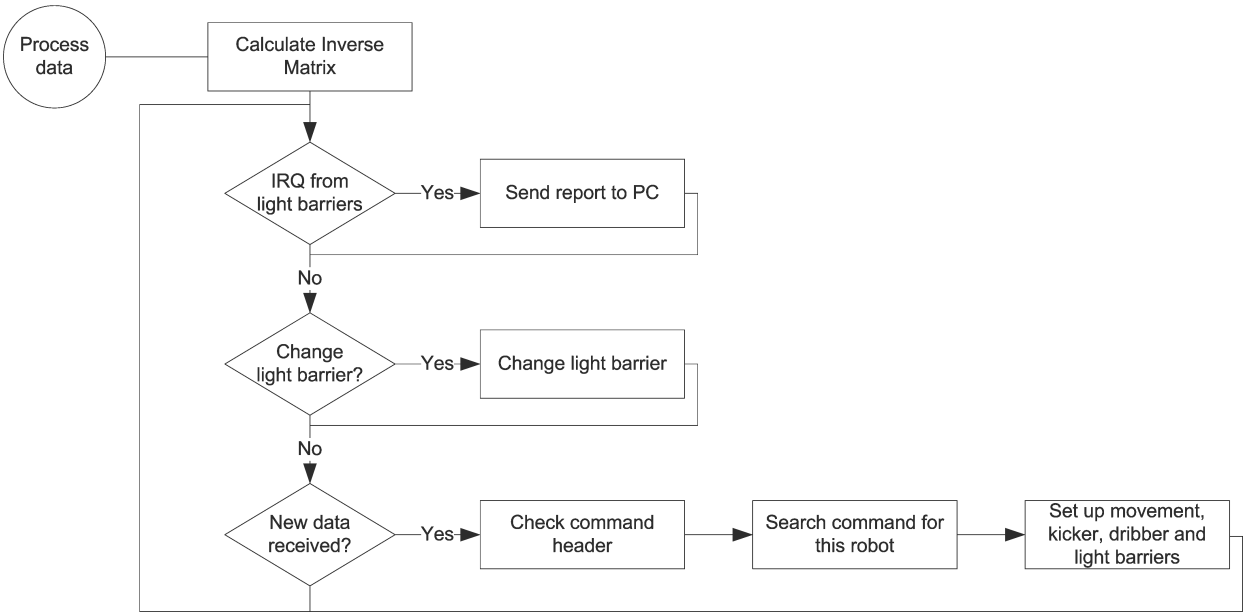


Abbildung 4.5.: Flussdiagramm zur Verarbeitung der Daten am *CM-BF527*

und Sensordaten. Zuerst wird überprüft ob ein Interrupt von einer Lichtschranke ausgelöst wurde. Wenn das der Fall ist, dann wird die Interruptquelle ermittelt. Ja nachdem welche Lichtschranke der Interrupt ausgelöst hat bedeutet das, dass der Ball entweder vor dem Roboter steht oder direkt am Dribbler anliegt. Diese Statusinformation wird sofort dem PC übermittelt und dort weiterverarbeitet.
Als Nächstes wird überprüft ob die Periode des Lichtschranken-Timers vorüber ist. Wenn ja, dann wird die momentan eingeschaltete Lichtschranke abgeschaltet und die Nächste in der Liste eingeschaltet.
Zum Schluss werden die empfangenen Funkdaten vom Funkmodul ausgelesen und nach einem Kommando mit der richtigen Robot ID gesucht. Wenn ein Kommando gefunden wurde, so werden die Drehzahlen für alle 4 Motoren neu berechnet und an die Motortreiber geschickt. Weiters werden eventuell die Module Kicker, Dribbler und Lichtschranken aktiviert/deaktiviert, je nach Kommando.

5 Ergebnisse und Erkenntnisse

5.1. Unterschiede zum Vorgängermodell

Wie bereits in der Einleitung erwähnt, gab es vom Roboter ein Vorgängermodell, das von den damaligen *Vienna Cubes* vom FH Technikum Wien für die RoboCup WM von 2005 und 2006 gebaut wurde. Da der Roboter im alten Design nicht mehr ausbaufähig war, wurde dieser von Grund auf neu entwickelt und mit neuen Features ausgestattet. In Tabelle 5.1 sind die Unterschiede zwischen altem und neuem Design aufgelistet:

	Altes Design aus 2006	**Neues Design**
Stromversorgung	• LiPo-Pack mit 4 Zellen (14.8V) • Getrennte Spannungsversorgung für Schussvorrichtung und Kontrollsystem • Lineare Spannungsregler • Tiefentladungsschutz für gesamtes LiPo-Pack	• LiPo-Pack mit 3 Zellen (9.3V) • Gemeinsame Spannungsversorgung für Schussvorrichtung und Kontrollsystem • Schalt- und lineare Spannungsregler • Tiefentladungsschutz für einzelne LiPo-Zellen
Schussvorrichtung	• Horizontaler Kicker • Spannung: 100V und 15.4mF • Schussgeschwindigkeit: $36\frac{km}{h}$ • Manuelle Entladung	• Horizontaler und Chip-Kicker • Spannung: 140V und 7.8mF • Schussgeschwindigkeit: $32\frac{km}{h}$ [29] • Automatische Entladung
Hauptprozessor	• 16-Bit *C167* • FPGA *XC2S200*	• 32/16-Bit Blackfin *BF527*
Ballerkennung	• 1 Lichtschranke	• 3 Lichtschranken • Position des Balls am Kicker kann grob ermittelt werden • Reichweite kann individuell angepasst werden
Funkmodul	• 2 Funkmodule • DECT-Modul *HW86010* • WLAN-Modul *Wi-ME*	• 1 Funkmodul • ISM-Modul *AMB2520* • 165 frei einstellbare Frequenzen

Dribbler	DC-Motor	DC-Motor
Bewegungssensorik	• Gyro *ADXRS300*	• Gyro *ADXRS610* • Accelerometer *ADXL322*
Antrieb	• 4 DC-Motoren Faulhaber *2642W 012 CR* • Omniwheels • Encoder • Treiber-Baustein für Ansteuerung • Große leistungsstarke Motoren	• 4 BLDC-Motoren Maxon *EC45 flat* • Omniwheels • Hall-Sensoren, lässt sich auf Encoder umrüsten • STM32-Prozessor für Ansteuerung • Kleine leistungsstarke Motoren
Camera-Unterstützung	Nein	Ja, Schnittstelle ist vorhanden
Update-Möglichkeit	• JTAG	• JTAG und serielle Schnittstelle • Wireless-Programmierung
Ausbaufähig	Nein	Ja

Tabelle 5.1.: Unterschiede zwischen altem und neuem Roboter-Design

5.2. Ergebnisse der RoboCup WM 2009 in Graz

Vom 29.06. bis 05.07.2009 fand in Graz die 13. RoboCup WM statt, bei der die Roboter zum ersten Mal zum Einsatz kamen. Für dieses Event wurden dank der tatkräftigen Hilfe der Sponsoren insgesamt 6 Roboter gebaut. Bei einem unserer Sponsoren gab es jedoch schwerwiegende interne Verständigungsprobleme und das beschehrte uns ernsthafte Probleme. Das Resultat: knappe 2 Wochen vor der WM wurden uns die Printplatten zugeschickt und erst dann konnten wir sie zum Bestücken weiterleiten. 3 Tage vor dem ersten Qualifikationsspiel waren dann die Printplatten vollständig bestückt und getestet.
Zu diesem Zeitpunkt hatte die Vorbereitungsphase der Weltmeisterschaft bereits begonnen und wir hatten noch keine Gelegenheit gehabt unsere Roboter mitsamt der PC-Software (Bildverarbeitung und Künstliche Intelligenz) zu testen. Die Roboter konnten lediglich über einen Joystick angesteuert werden, doch wie das Zusammenspiel zwischen PC-Software und Roboter funktionierte war noch völlig unklar.
Während die anderen Teams in der Vorbereitungsphase ihre Roboter kalibrieren, optimieren und testen konnten, nutzten wir die Zeit um unseren Robotern das Fahren bei-

zubringen. Die Kommunikation zwischen PC und Roboter funktionierte von Anfang an recht gut und somit konnten wir bald die Position der einzelnen Roboter am Spielfeld per Mausklick einstellen. Im Laufe der Vorbereitungsphase gelang es uns dann die Motorsteuerung soweit zu optimieren, dass die Roboter die gewünschte Position innerhalb kürzester Zeit einnahm. Doch ein Problem wurde sofort ersichtlich: aufgrund der geringen Auflösung der Hall-Sensoren, konnten sich die Roboter nicht genau positionieren da die Motorsteuerung keine langsamen Drehzahlen einstellen konnte. Das Resultat war, dass der Roboter während der Beschleunigungsphase und dem Standard-Fahrbetrieb zwar sehr schnell und präzise Richtung Ziel fuhr, doch während der Bremsphase und die darauffolgende Positionierungsphase konnten die niedrigen Drehzahlen nicht mehr eingestellt werden und der Roboter kam kurz vor dem Ziel zum Stillstand. Um das zu vermeiden wurde eine Mindestdrehzahl zur Berechnung der Trajektorie festgelegt. Der Roboter konnte somit die gewünschte Position während der Bremsphase zumindest ruckartig einnehmen.
Wie bereits gesagt, wurden die Roboter erst eine Woche vor der WM vollständig zusammengebaut und getestet. Die gesamte Elektronik bestehend aus 3 Platinen wurde in den vergangenen Monaten erfolgreich in Betrieb genommen. Erst als die Platinen am Roboter montiert wurden, fiel auf, dass der Hauptprozessor nicht starten konnte sondern ständig resertierte. Ein Blick auf die Spannungsversorgung des Roboters genügte um festzustellen, dass durch die Montage der Platinen am Robotergerüst hohe Störspannungen entstanden, die zum Ausfall des Systems führten. Daraufhin wurde die Kicker-Platine als Störquelle identifiziert: die Form dieser Platine und das daraus resultierende, EMV-technisch gesehen, schlechte Layout führt dazu, dass die Kabeln der Speicherkondensatoren eine Antenne bilden, und eine hohe Energie abstrahlen. In den benachbarten Platinen kommt es zu einer Strahlungskopplung und somit zum Systemausfall. Wenn man das Layout der Kickerplatine etwas verbessert indem man die Speicherkondensatoren direkt auf die Kickerplatine ohne Verbindungskabeln anlötet und somit die Leiterschleifen minimiert, entstehen keine Störspannungen. Die Störemission kann auch mit Hilfe von Klappferriten vermindert, aber nicht vollständig unterdrückt werden. Da wir keine Möglichkeit hatten, die Speicherkondensatoren direkt auf die Kicker-Platine anzulöten, musste diese entfernt werden. Uns blieb nichts Anderes übrig als den Dribbler als Schussmechanismus zu verwenden indem wir einfach beim Schießen die Drehrichtung der Walze umkehrten. In diesem Fall war wohl nicht mehr von *Schuss* die Rede , sondern eher von einem *kleinen Schubser*.
Trotz enormen Engagement des gesamten Entwicklerteams konnten wir uns nicht gegen die starke Konkurrenz durchsetzen und schieden deshalb bereits im Achtelfinale aus.

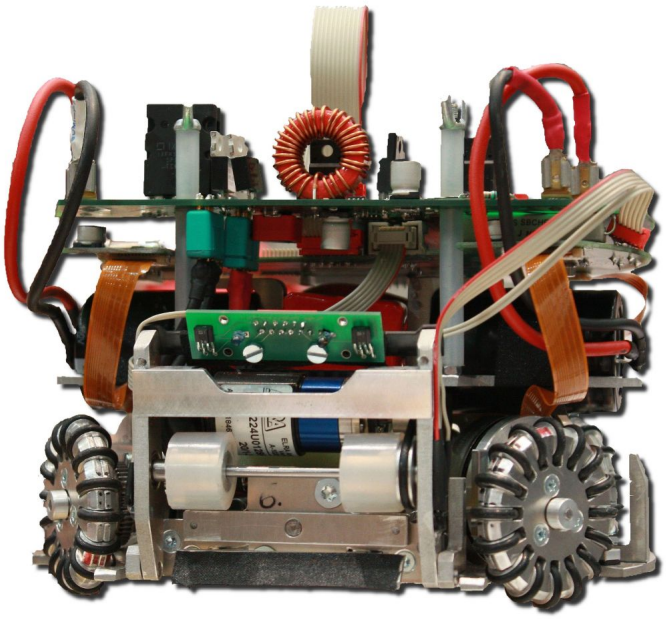


Abbildung 5.1.: Verkabelung der Kicker-Platine

5.3. Erweiterungsvorschläge für die nächste Generation

Die Teilnahme am RoboCup war in erster Linie hilfreich um die Schwächen des entworfenen Roboters zu erkennen und Verbesserungsvorschläge zu finden. Auch hat sich im Laufe dieser Masterarbeit die Produktpalette einiger Hersteller so erweitert, dass sich neue und bessere Möglichkeiten für den Roboter bieten.

Hohe Anzahl zu programmierende Einheiten : Am Roboter befinden sich insgesamt 5 Mikrokontroller, ein *CM-BF527* und 4 *STM32*-Prozessoren, die jeweils über eine eigene JTAG-Schnittstelle programmiert werden müssen. Neben der zeitaufwändigen Programmierung aller Prozessoren über JTAG, kommen noch die unterschiedlichen Entwicklungsumgebungen und die damit verbundenen Kompatibilitätsprobleme der C-Compiler hinzu. Eine saubere Lösung wäre die *STM32*-Prozessoren durch die, im 1. Quartal 2010 erschienenen, *BF50xF* BlackFin-

Prozessoren[1] zu ersetzen. Die BF50xF wurden extra für Motorsteuerungen konzipiert und bieten daher die Möglichkeit gleichzeitig 2 BLDC-Motoren anzusteuern. Weiters bieten die F-Varianten einen 4MB großen internen Flash-Speicher für Applikationen. Mit Hilfe der *BF50xF*-Prozessoren könnte man die Anzahl der Mikrokontroller von 5 auf 3 reduzieren und die Entwicklungsumgebung würde für alle Prozessoren die Gleiche sein.

Motorsteuerung mit Encoder und Sensorik : Bei der RoboCup WM 2009 in Graz hat sich herausgestellt, dass die Regelung der Motoren bei geringen Geschwindigkeiten durch die zu grobe Auflösung der Hall-Sensoren zu ungenau wird, bzw. gar nicht mehr möglich ist[2]. Die Regelung der BLDC-Motoren mit Hall-Sensoren funktioniert ab einer Geschwindigkeit von ungefähr $70\frac{U}{min}$ doch es kommt andauernd vor, dass die Motoren sehr viel langsamer fahren müssen (wenn sich die Roboter irgendwo genau positionieren müssen). Deswegen soll in der nächsten Roboter-Generation auf Encoder umgestiegen werden und zwar genauer gesagt auf die *AS5000*-Familie von austriamicrosystems[3].

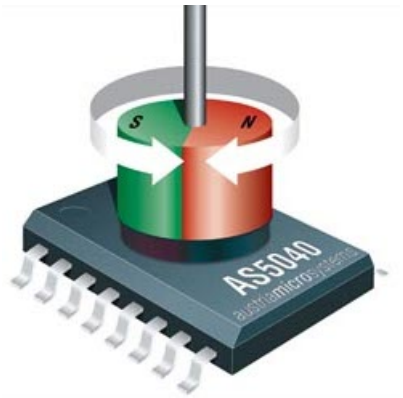


Abbildung 5.2.: On-Axis Encoder der *AS5000*-Familie von austriamicrosystems

Die *AS5000*-Familie ist eine sehr leistungsfähige Gruppe magnetischer Encoder

[1] BlackFin BF506F von Analog Devices - www.analog.com

[2] Es gäbe die Möglichkeit die Übersetzung des Getriebes höher zu wählen (momentan liegt sie bei $1 : 2,8$) und somit eine höhere Auflösung pro Umdrehung bei Verwendung von Hall-Sensoren zu erreichen. Eine höhere Übersetzung des Getriebes führt aber zu höheren Produktionskosten der mechanischen Teile. Mit höher werdenden Übersetzung verliert der Roboter an Maximalgeschwindigkeit und es müssen ggf. Planetengetriebe angefertigt werden, die zuviel Platz benötigen und teuer sind.

[3] austriamicrosystems - www.austriamicrosystems.com

für die Detektion von Linear- und Drehbewegungen. Diese Bausteine werden entweder über, sich auf Linear- bzw. auf Kreisbahnen bewegender Magnete (Off-Axis Encoder) oder über der Drehachse rotierender Magnete (On-Axis Encoder) positioniert (siehe Abbildung 5.2). Mit Hilfe von On-Axis Encoder lassen sich Rotationsvorgänge (Drehwinkel und Drehzahl) sehr genau messen und ersetzen somit die altbekannten mechanischen Encoder.
Die On-Axis Encoder verwenden 4 Hall-Elemente als magnetische Sensoren, die zueinander um 90° versetzt angeornet sind. Durch die Reihenschaltung der beiden jeweils gegenüberliegenden Sensoren erhält man das Sinus- und Cosinus-Signal des Drehwinkels eines über dem Baustein rotierenden Magnets. Diese Signale werden digital verarbeitet und von einer Recheneinheit CORDIC (Coordinate Rotation Digital Computer) in eine Amplituden- und Winkelinformation umgerechnet. Die Encoder der *AS5000*-Reihe erreichen Schrittweiten von 1,4° (8-Bit) bis 0,0879° (12-Bit). Durch die kurzen Leitungswege zwischen Sensor und Elektronik und der digitalen Verarbeitung der Sensorsignale bleiben Phasenfehler selbst bei hoher Drehzahl im vernachlässigbaren Bereich. Die Rotationsinformation kann über ein serielles Interface, über eine PWM-, Analog- oder Inkremental-Schnittstelle oder über Kombinationen, der zuvor genannten Möglichkeiten ausgegeben werden [**?**].

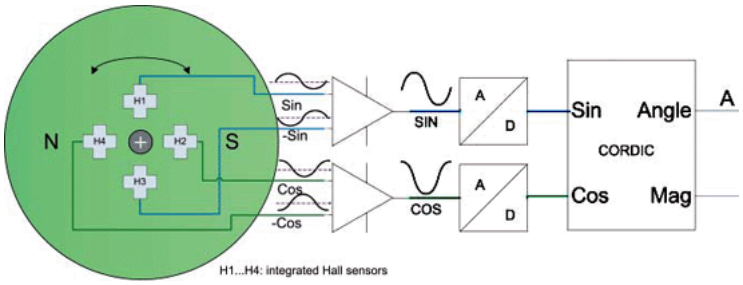


Abbildung 5.3.: Interner Aufbau der On-Axis-Encoder *AS5000*

Da die Motoren zum jetzigen Standpunkt mit Hall-Sensoren geregelt werden, wird auf den Einsatz von magnetischen Encoden nicht näher eingegangengehen. Im Anhang A.8 und A.9 finden Sie eine Adapterplatine, mit der sich die bestehende Hardware auf Encoder umrüsten lässt.
Geplant ist, eine runde Platine mit dem Durchmesser eines *EC 45 flat* zu bauen und diese an einem Motorgehäuse so zu fixieren, dass ein Magnet-Encoder

die Rotation des Außenläufers aufnehmen kann. Auf einer Seite der Platine wird ein Magnet-Encoder angebracht und auf der anderen Seite wird der *STM32* samt Endstufe befestigt. Somit wird ein kompakter BLDC-Motor samt Steuerung in einem Motorgehäuse untergebracht.
In der aktuellen Firmware-Version werden die Daten des Gyrometers und des Beschleunigungssensors nicht verarbeitet, da keine aufwendige Regelschleife implementiert wurde. In der nächsten Versionen sollen jedoch neue Algorithmen für die Berechnung der Radgeschwindigkeiten implementiert werden, wobei die Bewegungssensoren in der Regelschleife miteinbezogen werden müssen.

Betriebssystem auf BlackFin-Prozessoren : die aktuelle Firmware läuft in einem Single-Thread-System und dies führt zu extremen Performance-Verlusten. Die nächste Generation soll ein Multi-Threading-System, wie beispielsweise BLACKSheep von Bluetechnix[4], verwenden. Damit können sinnvolle Strukturen und Interaktionen zwischen den verschiedenen Threads implementiert werden.

EMV-gerechtes Layout für die Schussvorrichtung : Das Layout und die Form des Schussmechanismus muss EMV-gerecht konstruiert werden. Bevor aber eine neue Platine entworfen wird, soll die Form der Hubmagnete überarbeitet werden: die Form soll sehr flach und rechteckig sein mit einem rechteckigen Bolzen. Somit können die zwei Hubmagnete und die dazugehörige Platine samt Speicherkondensatoren übereinander liegen und eine sehr kompakte Form einnehmen.

[4]BLACKSheep OS von Bluetechnix - www.bluetechnix.com

A.1. J-Link Adapter ver.C

Menge	Device	Bauteil
1	SSM-110-L-DV-002	X1
1	FTSH-110-01-L-DV-K	X2
1	FFSD-10-D-10.00-01-N	

Tabelle A.1.: Materialliste von J-Link Adapter ver.C

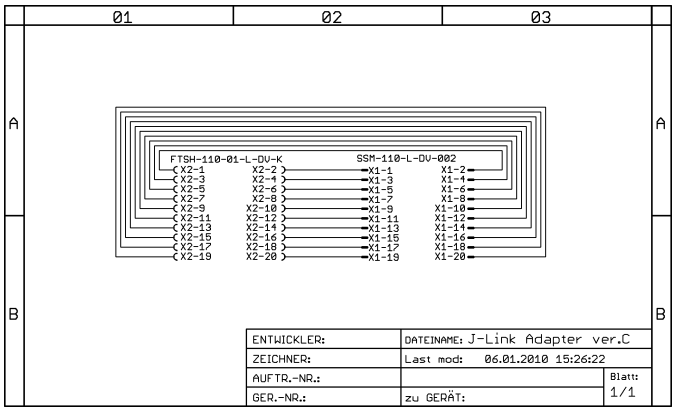


Abbildung A.1.: Schaltplan von J-Link Adapter ver.C

Abbildung A.2.: Bottom-Layer von J-Link Adapter ver.C

Abbildung A.3.: Top-Layer von J-Link Adapter ver.C

A.2. Kicker ver.D

Menge	Wert	Device	Bauteil
4	1µ	C-EUC1206	C1, C4, C8, C15
1	220n	C-EUC0805	C10
1	100p	C-EUC0603	C12
2		C-EUC0805	C2, C3
6	100n	C-EUC0603	C5, C9, C11, C13, C17, C18
2	100p	C-EUC0805	C6, C14
2	100µ	CPOL-EU153CLV-0810	C7, C16
3	PMEG3010EH	1N4148	D1, D2, D7
1	DPG15I200PA	BYT08P	D3
2	DSEP29-03A	BYT08P	D4, D5
1	ADR5041BKSZ	MAX6138	D6
1	UCC27201D	UCC27201D	IC1
1	MAX668EUB	MAX668EUB	IC2
1	LM393PW	LM393PW	IC3
1	SN74LVC1GU04DCK	SN74LVC1GU04DCK	IC4
1	SN74LVC1G175DCKRE4	SN74LVC1G175DCK	IC5
1	UCC27425D	UCC27425D	IC6
1	742792903	WE-PF	L1
1	33µ	WUERTH-744150	L2
1	GREEN	LEDCHIP-LED0805	LD1
1	2k2	RKH214-8	R1
4	1k	R-EU_R0805	R10, R13, R21, R22
1	300k	R-EU_R0805	R15
1	8m	R-EU_R2512	R16
1	2k7	R-EU_R0805	R17
2	3R3	R-EU_R1210	R18, R19
1	560k	R-EU_R0805	R2
1	2k	R-EU_R0805	R20
1	160k	R-EU_R0805	R23
1	150	R-EU_R0805	R24
3	5k6	R-EU_R0805	R3, R26, R27
1	220	R-EU_R0805	R4
2	4R3	R-EU_0204/7	R5, R11
2	10k	R-EU_R0805	R6, R25
3	100k	R-EU_R0805	R7, R12, R14
1	680	R-EU_R0805	R8
1	12k	R-EU_R0805	R9
4	FLACHSTECKER	FLACHSTECKER	SV1, SV2, SV3, SV4
3	FDP39N20	STP5NA50	T1, T2, T4
1	BC848	BC847	T3
2	IXFK180N15P	IXFK180N15P	T5, T6
1	MICROMATCH-12	MICROMATCH-12	X1
2	CON-MPX	CON-MPX	X2, X3

Tabelle A.2.: Materialliste von Kicker ver.D

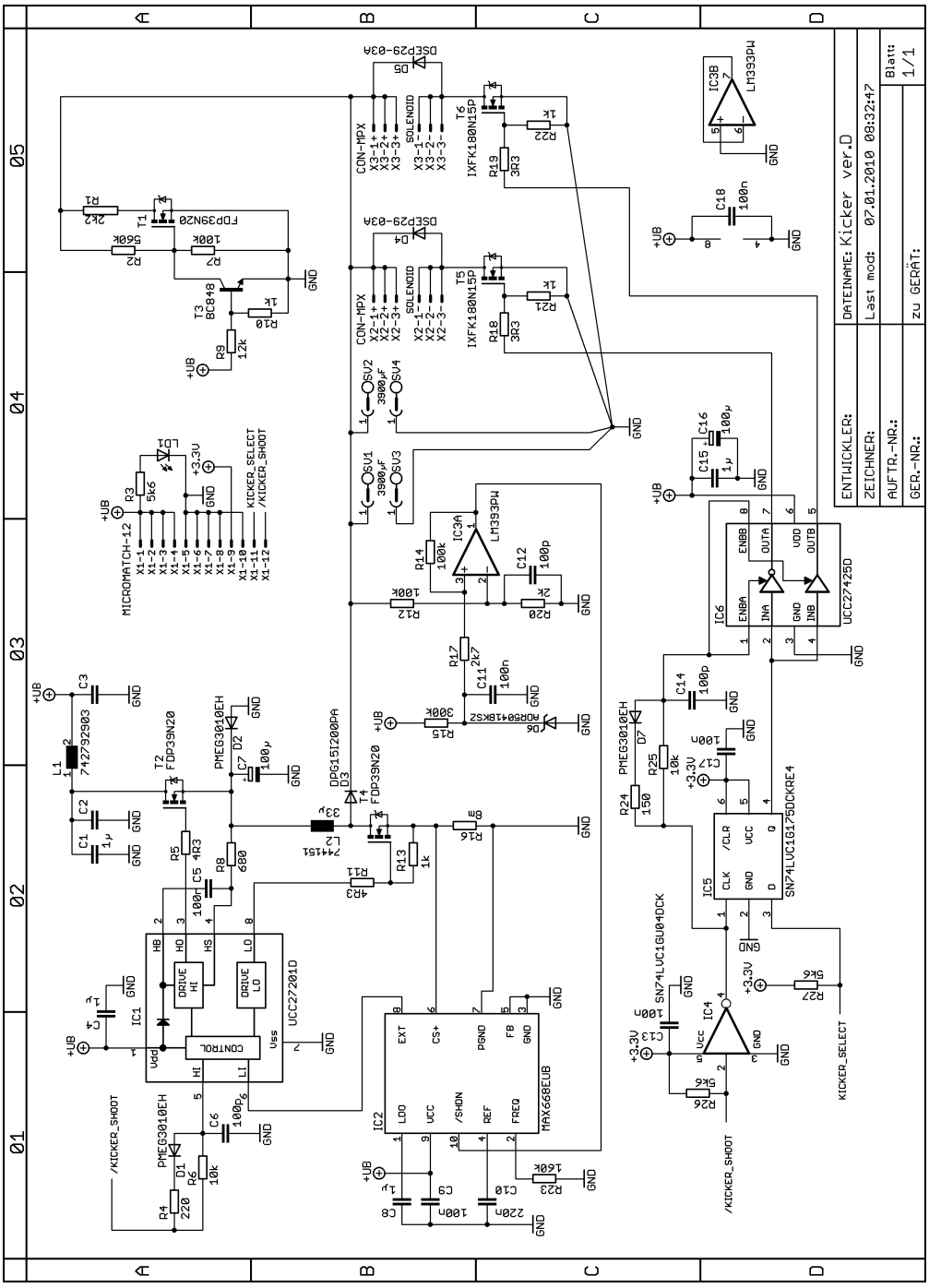


Abbildung A.4.: Schaltplan von Kicker ver.D

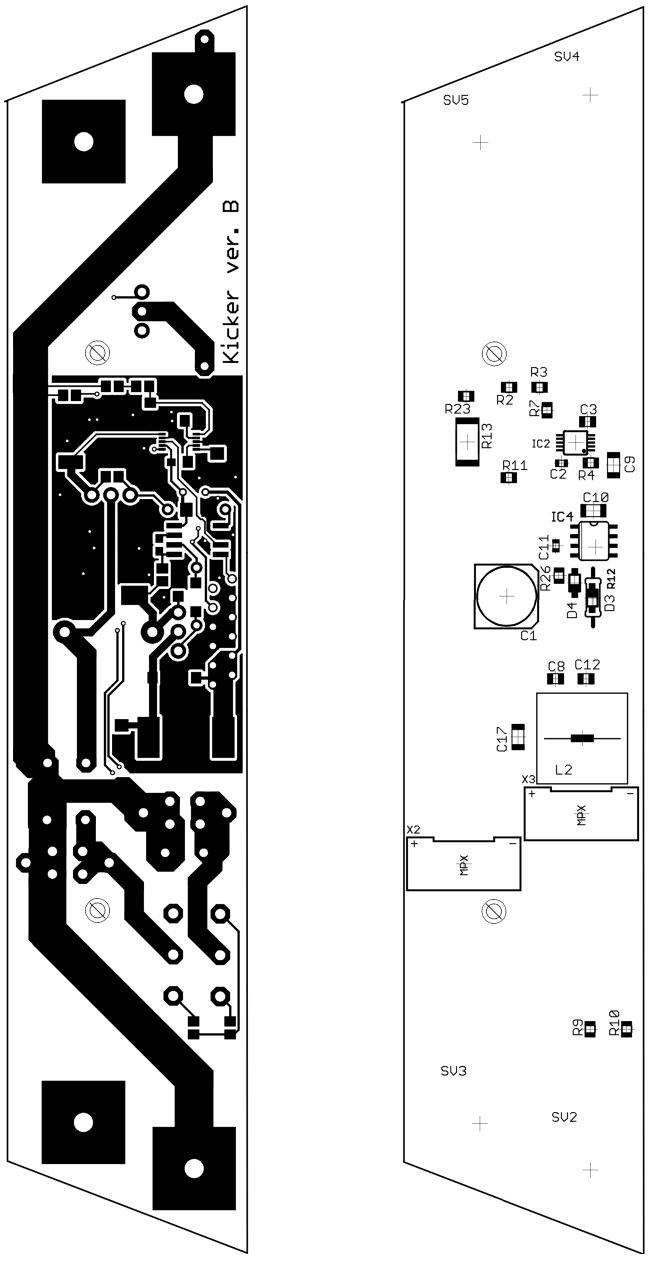


Abbildung A.5.: Bottom-Layer von Kicker ver.D

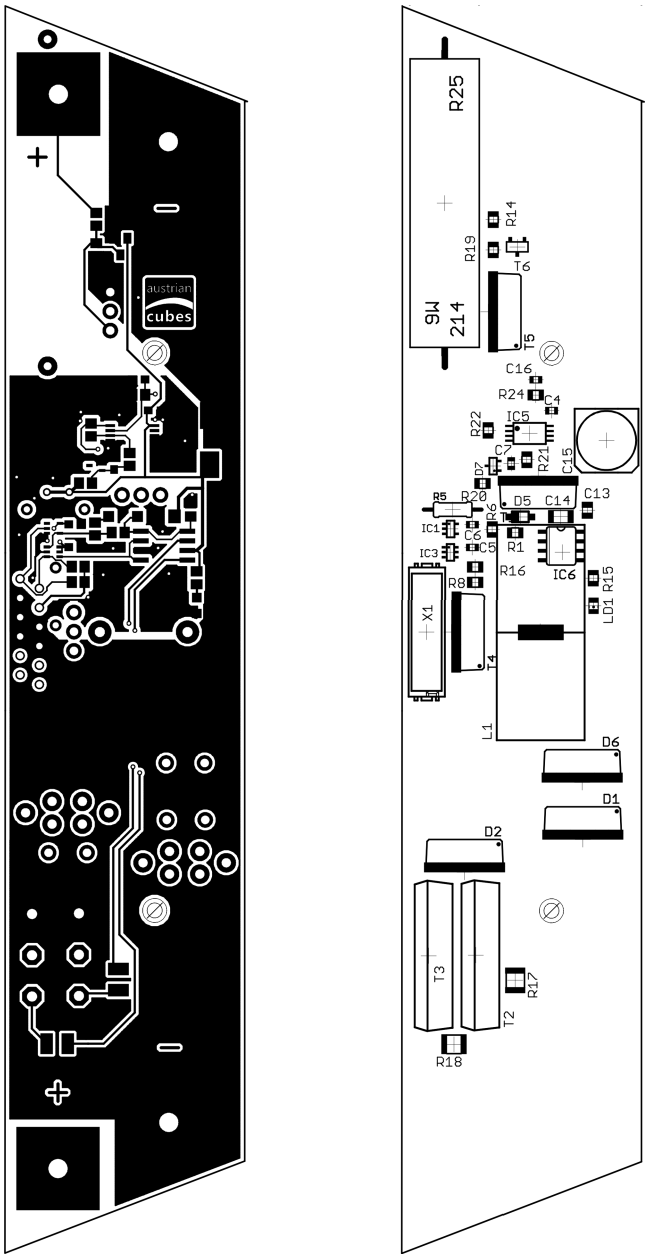


Abbildung A.6.: Top-Layer von Kicker ver.D

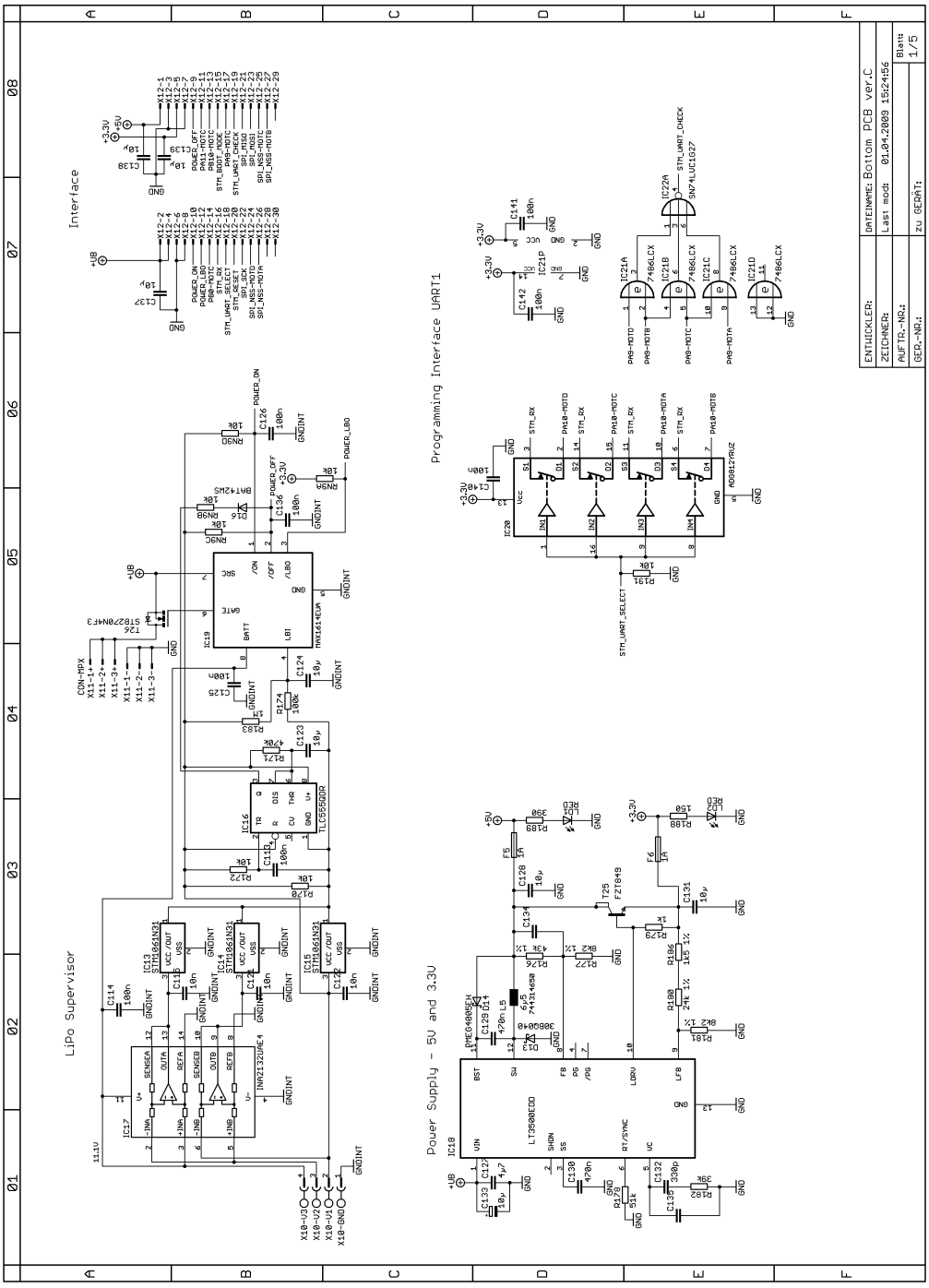
A.3. Bottom ver.D

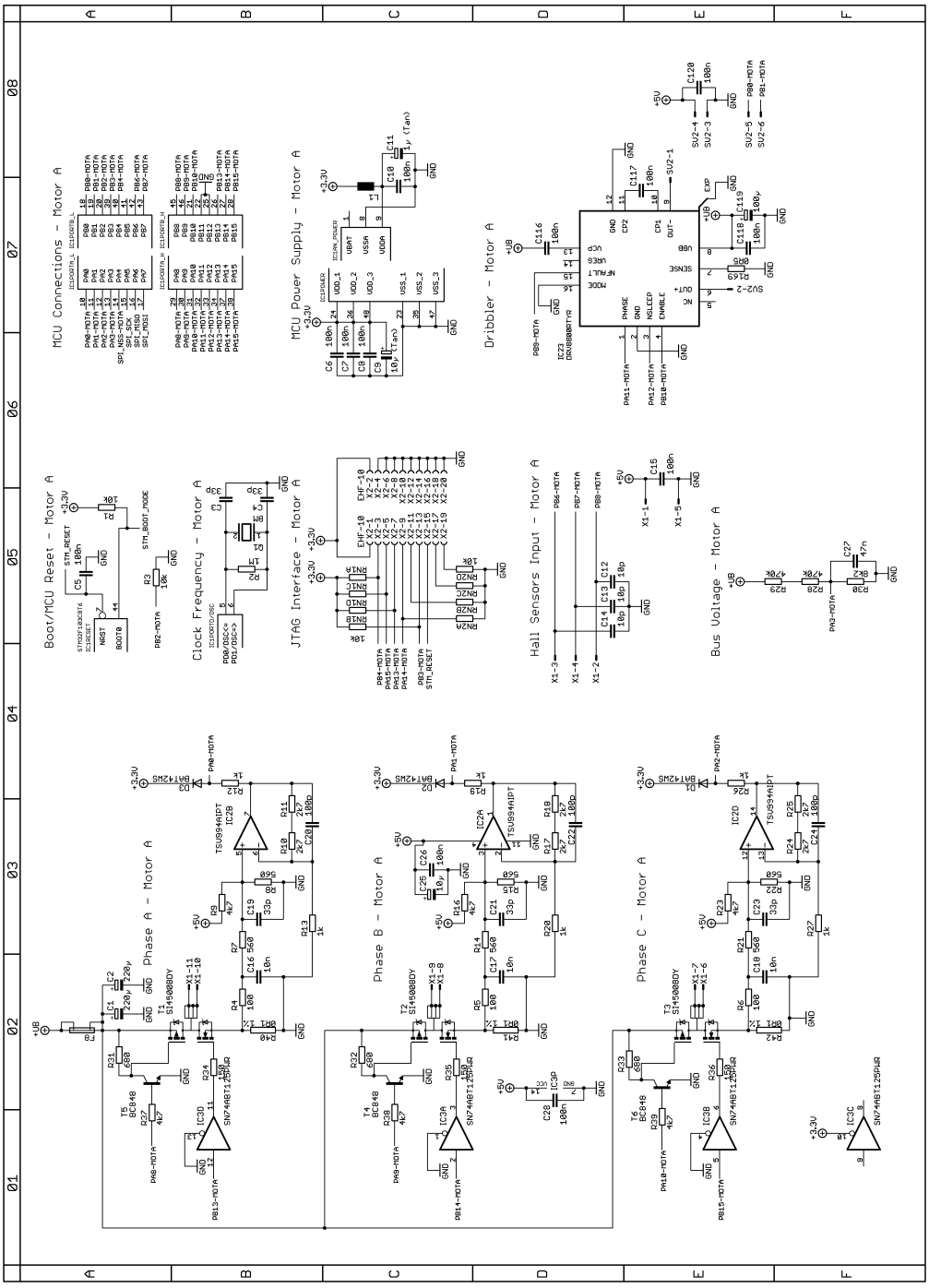
Menge	Wert	Device	Bauteil
8	220	CPOL-EU153CLV-0810	C1, C2, C55, C56, C83, C84, C111, C112
4	1µ(Tan)	CPOL-EUCT3216	C11, C37, C65, C93
1	100µ	CPOL-EU153CLV-0810	C119
12	10p	C-EUC0603	C12, C13, C14, C38, C39, C40, C66, C67, C68, C94, C95, C96
7	10µ	C-EUC1206	C123, C124, C128, C131, C137, C138, C139
1	4µ7	C-EUC1206	C127
2	470n	C-EUC0805	C129, C130
1	330p	C-EUC0805	C132
2		C-EUC0805	C134, C135
15	10n	C-EUC0805	C16, C17, C18, C42, C43, C44, C70, C71, C72, C98, C99, C100, C115, C121, C122
12	100p	C-EUC0805	C20, C22, C24, C46, C48, C50, C74, C76, C78, C102, C104, C106
5	10µ	CPOL-EU153CLV-0505	C25, C51, C79, C107, C133
4	47n	C-EUC0805	C27, C53, C81, C109
20	33p	C-EUC0805	C3, C4, C19, C21, C23, C29, C30, C45, C47, C49, C57, C58, C73, C75, C77, C85, C86, C101, C103, C105
45	100n	C-EUC0603	C5, C6, C7, C8, C10, C15, C26, C28, C31, C32, C33, C34, C36, C41, C52, C54, C59, C60, C61, C62, C64, C69, C80, C82, C87, C88, C89, C90, C92, C97, C108, C110, C113, C114, C116, C117, C118, C120, C125, C126, C136, C140, C141, C142, C143
4	10µ(Tan)	CPOL-EUCT3216	C9, C35, C63, C91
14	PMEG4005EH	SCHOTTKY-DIODESOD123	D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D14, D15

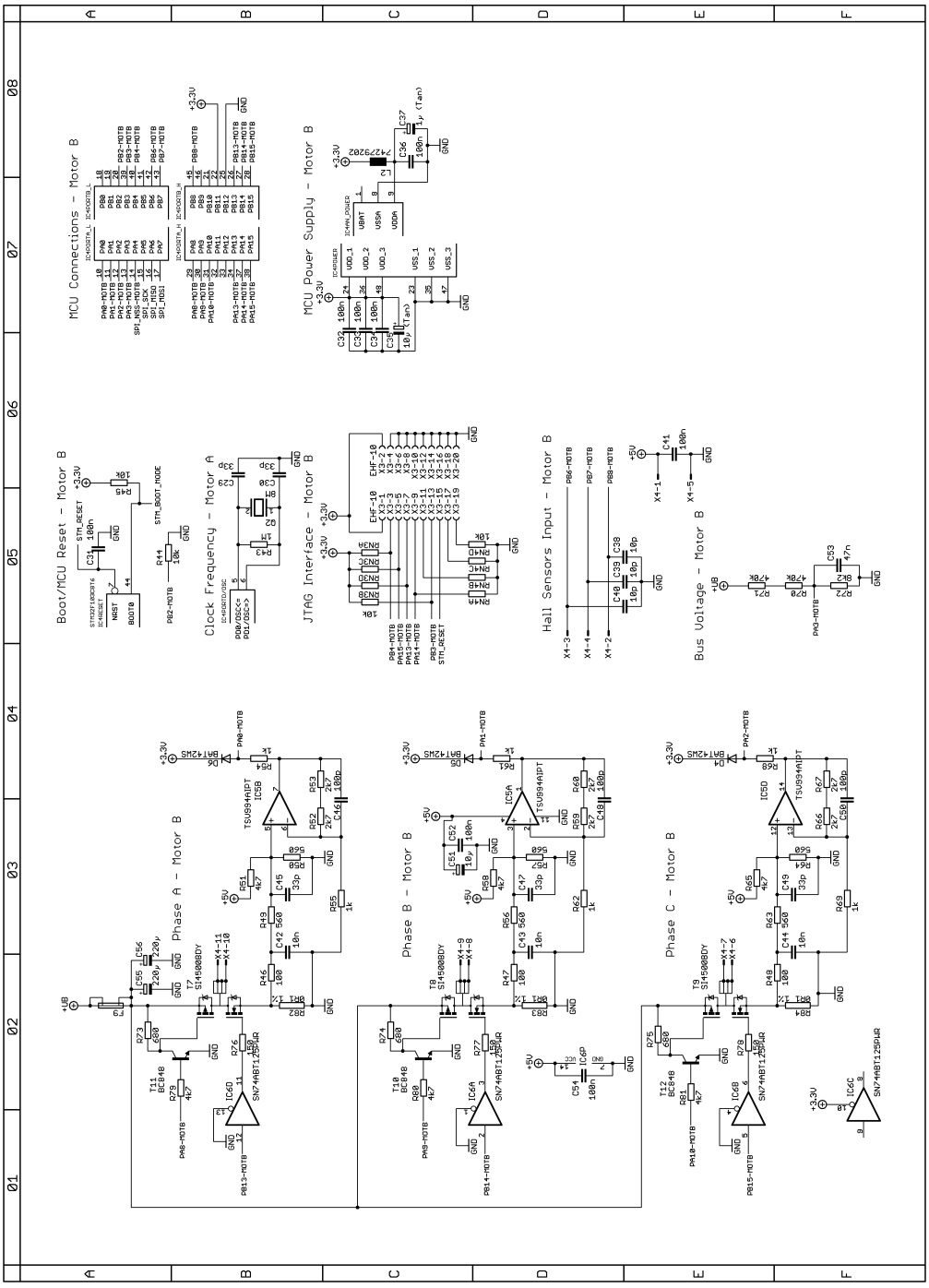
1	30BQ040	SCHOTTKY-DIODESMC	D13
4	2 x Sicherunghalter	SHK20QS	F1, F2, F8, F9
2	1A	FUSE_SMF	F5, F6
4	STM32F103C8T6	STM32F10XCXT6	IC1, IC4, IC7, IC10
3	STM1061N31	STM1061	IC13, IC14, IC15
1	TLC555QDR	SE555D	IC16
1	INA2132UAE4	INA2132U	IC17
1	LT3500EDD	LT3500EDD	IC18
1	MAX1614EUA	MAX1614EUA	IC19
4	TSV994AIPT	TSV994AIPT	IC2, IC5, IC8, IC11
1	ADG812YRUZ	ADG812YRUZ	IC20
1	7486LCX	7486LCX	IC21
1	SN74LVC1G27	SN74LVC1G27	IC22
1	DRV8800RTYR	DRV8800RTYR	IC23
4	74279202	WE-CBF_0805	L1, L2, L3, L4
1	744314650	WE-HC_744312...	L5
2	RED	LEDCHIP-LED0805	LD1, LD2
4	8M	XTAL/S	Q1, Q2, Q3, Q4
22	10k	R-EU_R0805	R1, R3, R10, R17, R24, R44, R52, R59, R66, R86, R87, R94, R101, R108, R128, R129, R136, R143, R150, R170, R172, R191
16	0	R-EU_R0805	R11, R18, R25, R53, R60, R67, R95, R102, R109, R137, R144, R151, R198, R200, R202, R204
25	1k	R-EU_R0805	R12, R13, R19, R20, R26, R27, R54, R55, R61, R62, R68, R69, R96, R97, R103, R104, R110, R111, R138, R139, R145, R146, R152, R153, R179
1	0R27	R-EU_0411/12	R169
12	0R1	R-EU_R2512	R173, R175, R184, R185, R187, R190, R192, R193, R194, R195, R196, R197
1	43k	R-EU_R0805	R176
1	51k	R-EU_R0805	R178
1	24k	R-EU_R0805	R180
1	39k	R-EU_R0805	R182
1	1k5	R-EU_R0805	R186
4		R-EU_R0805	R199, R201, R203, R205
5	1M	R-EU_R0805	R2, R43, R85, R127, R183
9	470k	R-EU_R0805	R28, R29, R70, R71, R112, R113, R154, R155, R171
6	8k2	R-EU_R0805	R30, R72, R114, R156, R177, R181

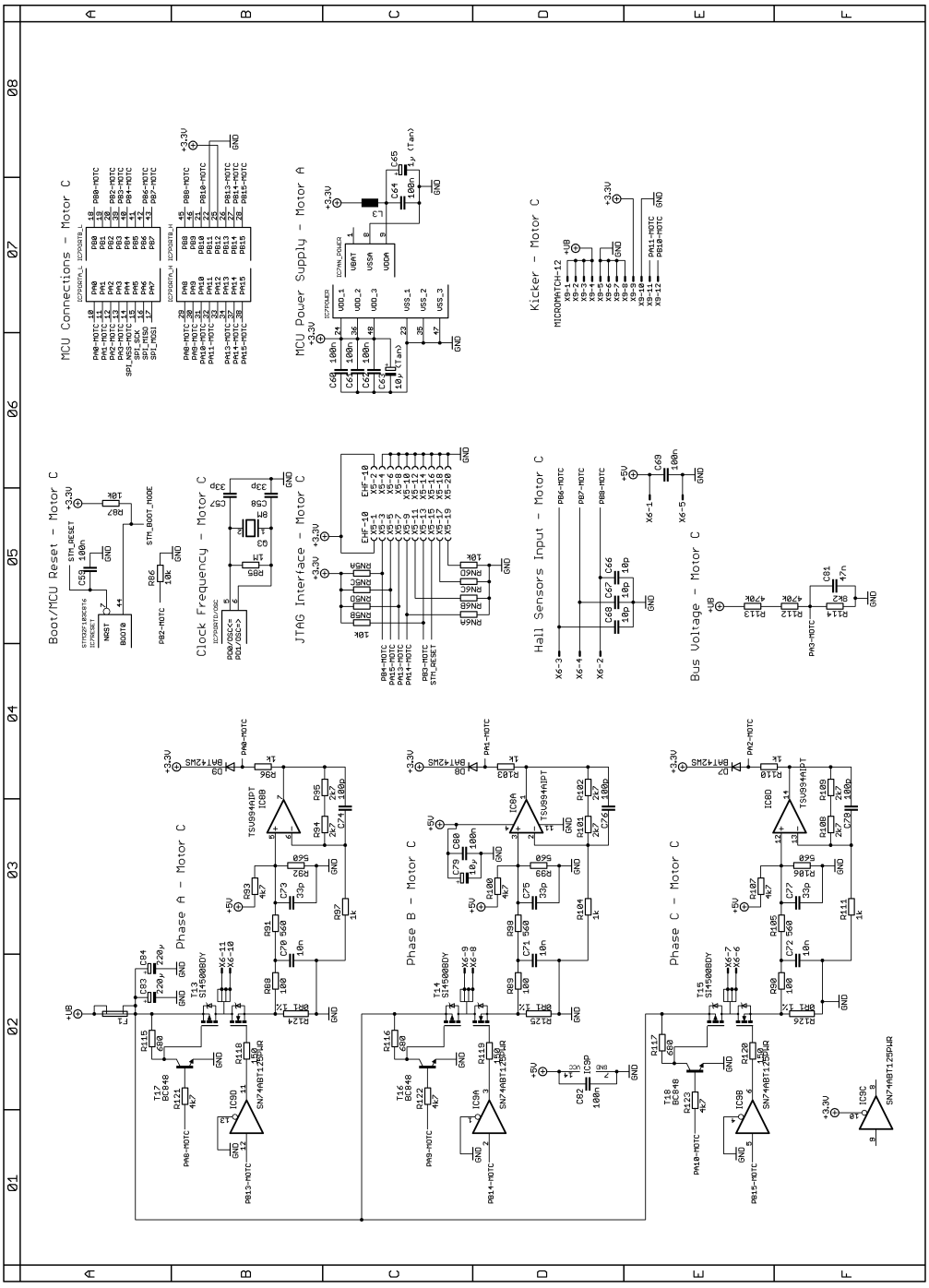
13	680	R-EU_R0805	R31, R32, R33, R73, R74, R75, R115, R116, R117, R157, R158, R159, R189
13	150	R-EU_R0805	R34, R35, R36, R76, R77, R78, R118, R119, R120, R160, R161, R162, R188
12	100	R-EU_R0805	R4, R5, R6, R46, R47, R48, R88, R89, R90, R130, R131, R132
12	0R1	SHUNT	R40, R41, R42, R82, R83, R84, R124, R125, R126, R166, R167, R168
2	100k	R-EU_R0805	R45, R174
12	200	R-EU_R0805	R7, R14, R21, R49, R56, R63, R91, R98, R105, R133, R140, R147
12	560	R-EU_R0805	R8, R15, R22, R50, R57, R64, R92, R99, R106, R134, R141, R148
24	4k7	R-EU_R0805	R9, R16, R23, R37, R38, R39, R51, R58, R65, R79, R80, R81, R93, R100, R107, R121, R122, R123, R135, R142, R149, R163, R164, R165
9	10k	RN-4P1206	RN1, RN2, RN3, RN4, RN5, RN6, RN7, RN8, RN9
1	IDC 90 Grad	ML6LE	SV2
12	SI4500BDY	SI4500BDY	T1, T2, T3, T7, T8, T9, T13, T14, T15, T19, T20, T21
1	FZT849	NPNSOT223	T25
1	IRFS4710	IRLX8743R	T26
12	BC848	BC848	T4, T5, T6, T10, T11, T12, T16, T17, T18, T22, T23, T24
4	0522071160	0522071160	X1, X4, X6, X8
1	S4B-EH	BALANCER-4	X10
1	CON-MPX	CON-MPX	X11
1	TYCO 5104652-3	CON_AMPMODU_REC_30	X12
4	FTSH-150	EHF-10	X2, X3, X5, X7
1	MICROMATCH-12	MICROMATCH-12	X9
4	SN74ABT125PWR		IC3, IC6, IC9, IC12

Tabelle A.3.: Materialliste von Bottom ver.D


Interface
Programming Interface UART1
Power Supply - 5V and 3.3V
LiPo Supervisor
ENTWICKLER:
ZEICHNER:
AUFTR.-NR.:
GER.-NR.:
DATEINAME: Bottom PCB ver.C
Last mod: 01.04.2009 15:24:56
ZU GERÄT:
Blatt
1/5


MCU Connections - Motor A
MCU Power Supply - Motor A
Dribbler - Motor A
Boot/MCU Reset - Motor A
Clock Frequency - Motor A
JTAG Interface - Motor A
Hall Sensors Input - Motor A
Bus Voltage - Motor A
Phase A - Motor A
Phase B - Motor A
Phase C - Motor A


MCU Connections - Motor B
Boot/MCU Reset - Motor B
Clock Frequency - Motor A
JTAG Interface - Motor B
MCU Power Supply - Motor B
Hall Sensors Input - Motor B
Bus Voltage - Motor B
Phase A - Motor B
Phase B - Motor B
Phase C - Motor B


MCU Connections - Motor C
Boot/MCU Reset - Motor C
Clock Frequency - Motor C
JTAG Interface - Motor C
MCU Power Supply - Motor A
Kicker - Motor C
Hall Sensors Input - Motor C
Bus Voltage - Motor C
Phase A - Motor C
Phase B - Motor C
Phase C - Motor C

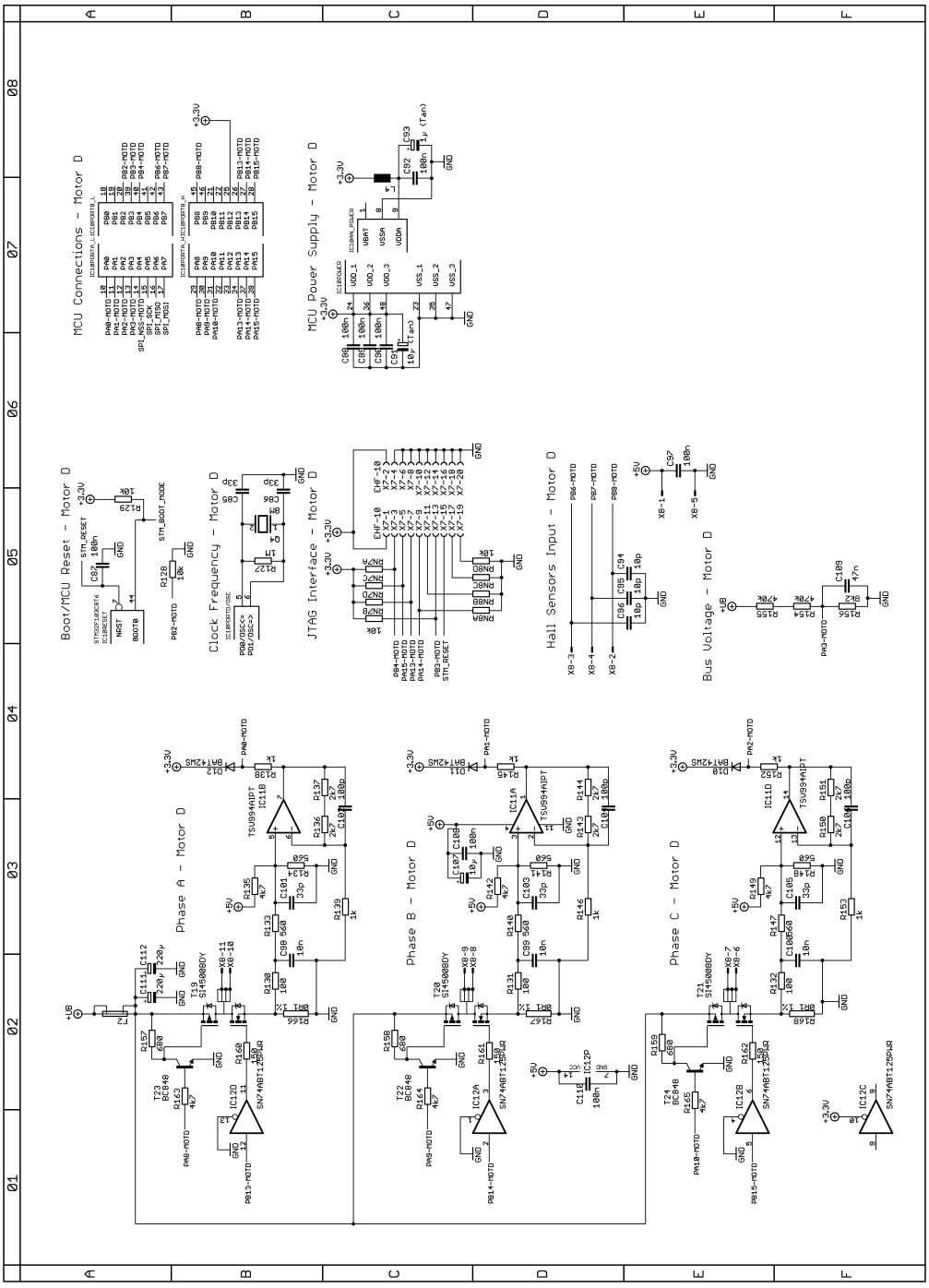


Abbildung A.7.: Schaltplan von Bottom ver.D

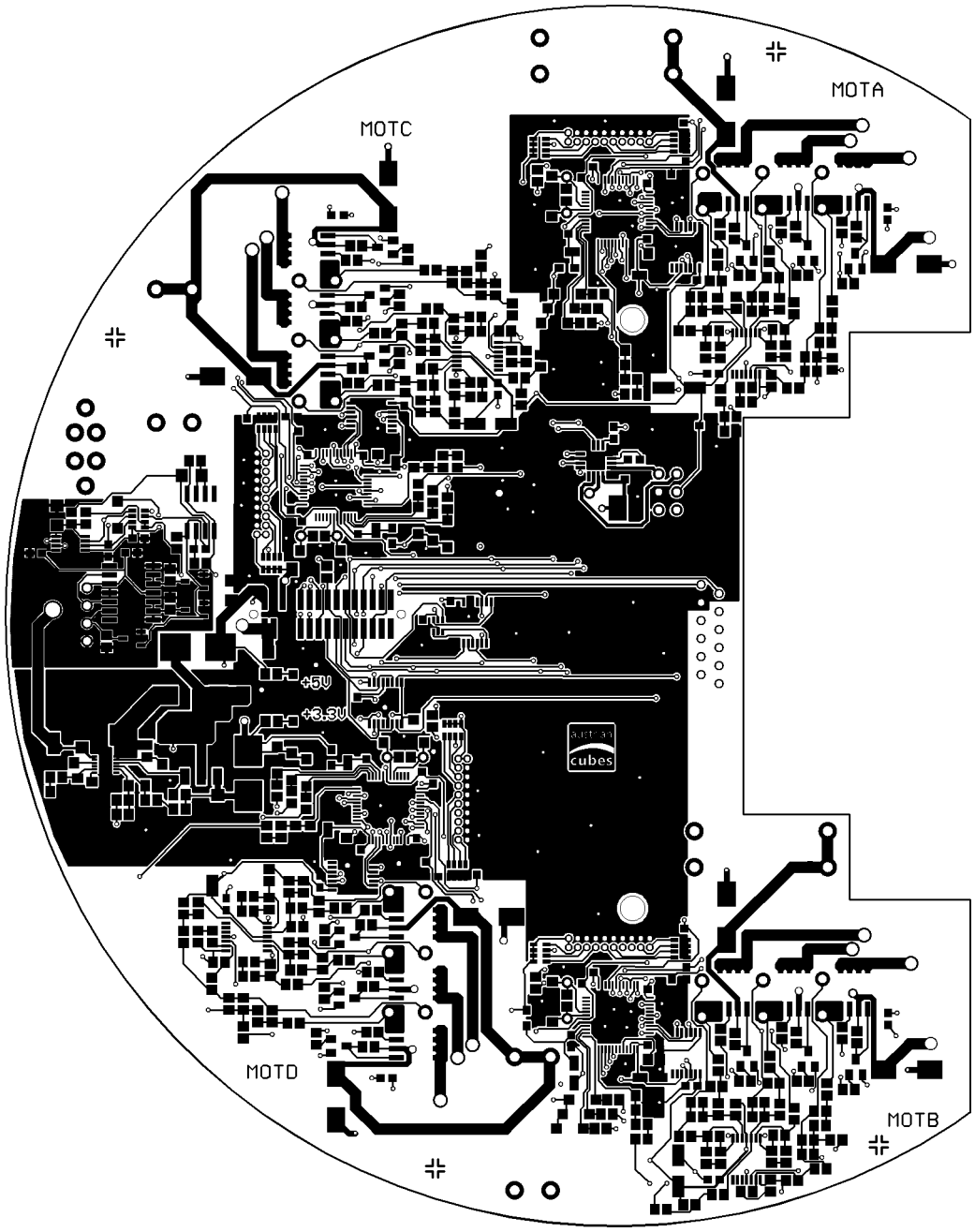


Abbildung A.8.: Top-Layer von Bottom ver.D

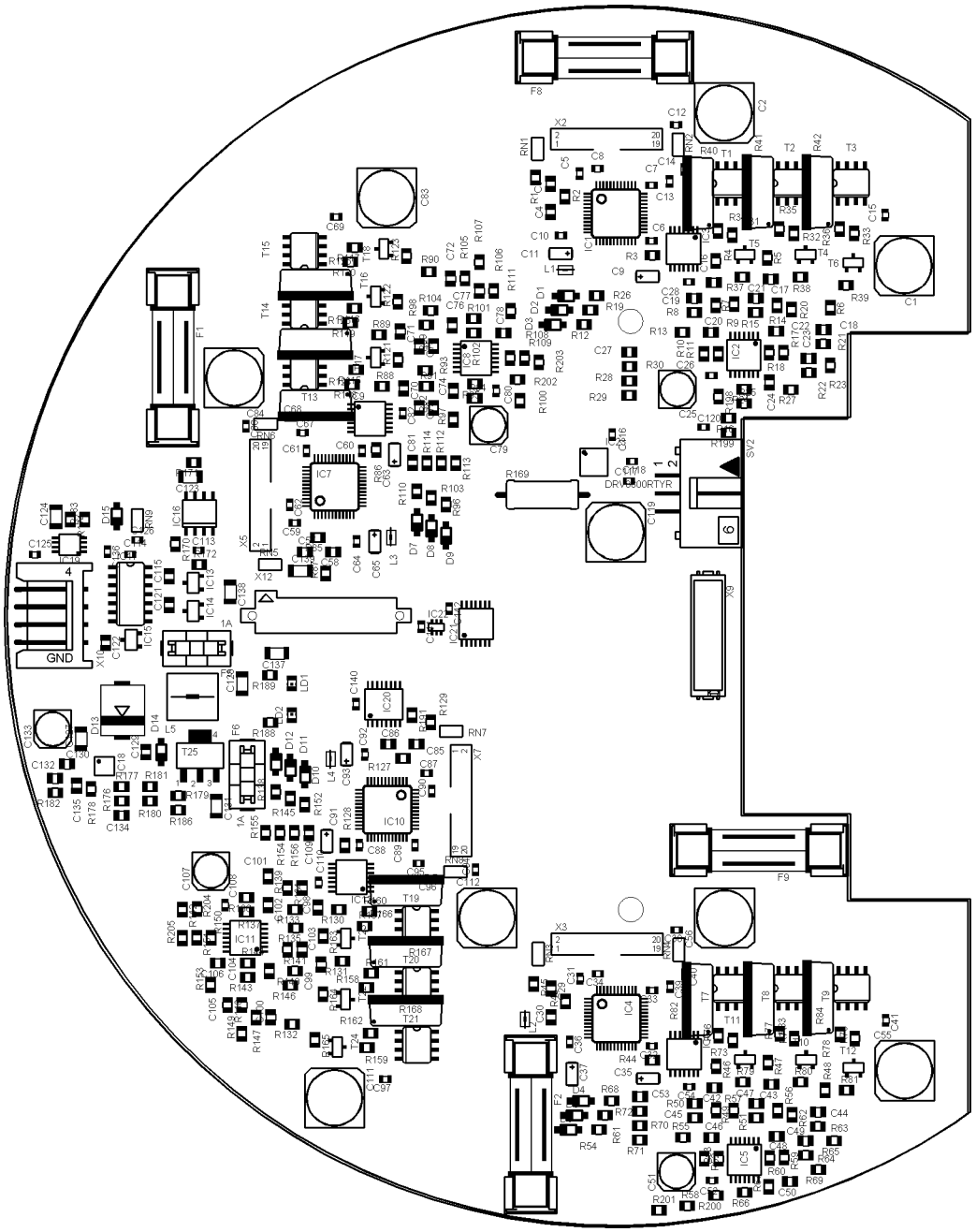


Abbildung A.9.: Top-Layer Bestückung von Bottom ver.D

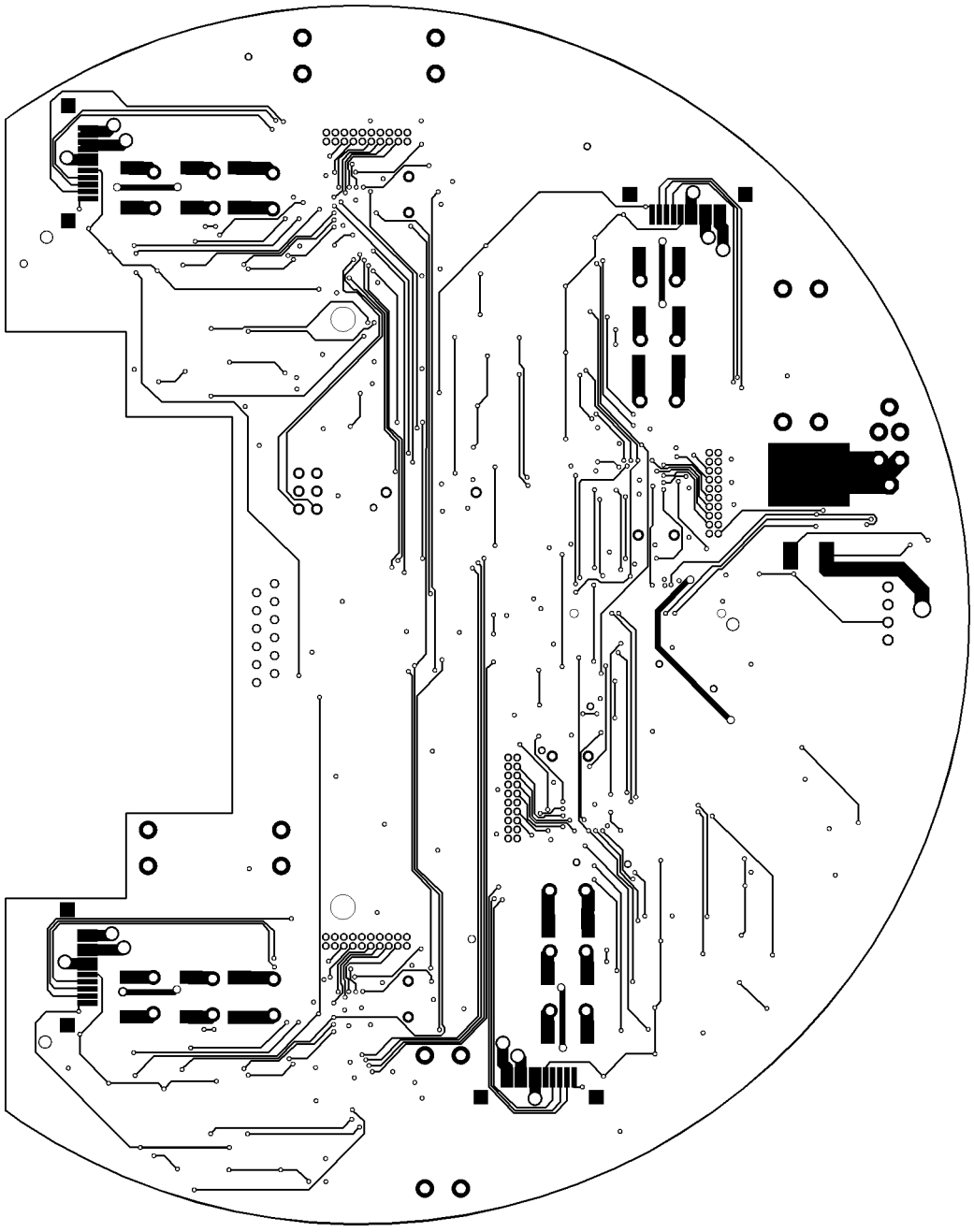


Abbildung A.10.: Bottom-Layer von Bottom ver.D

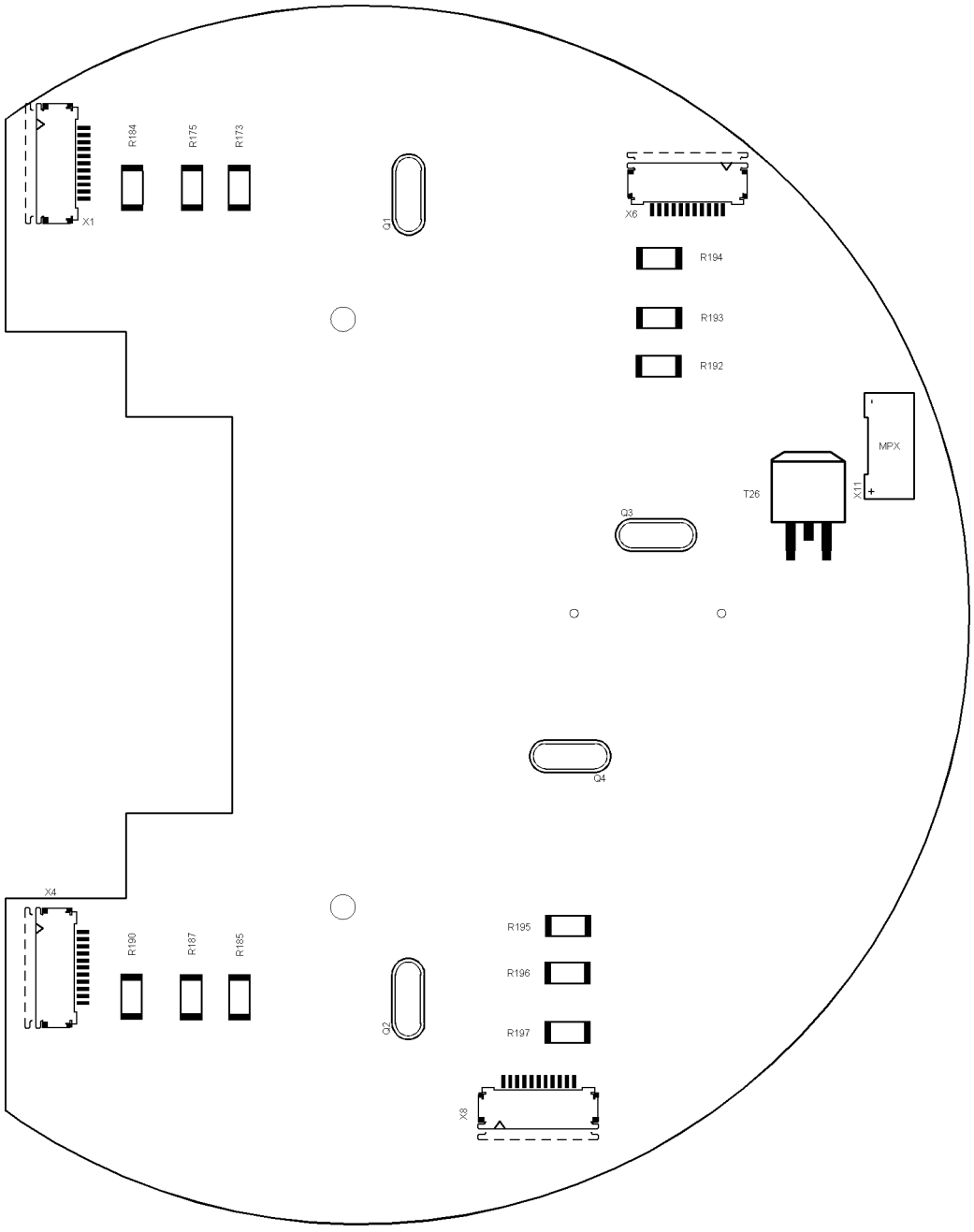


Abbildung A.11.: Bottom-Layer Bestückung von Bottom ver.D

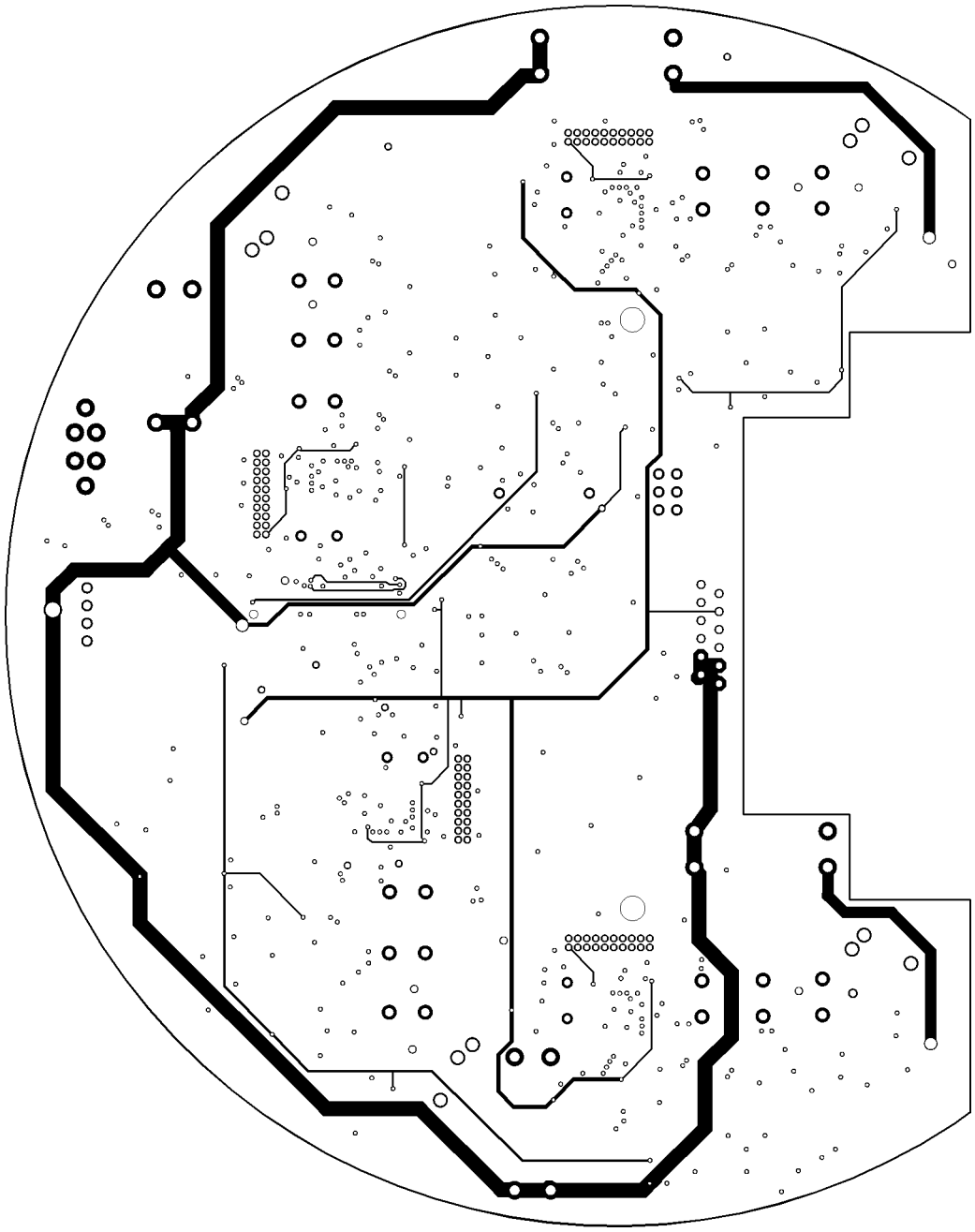


Abbildung A.12.: Power-Layer von Bottom ver.D

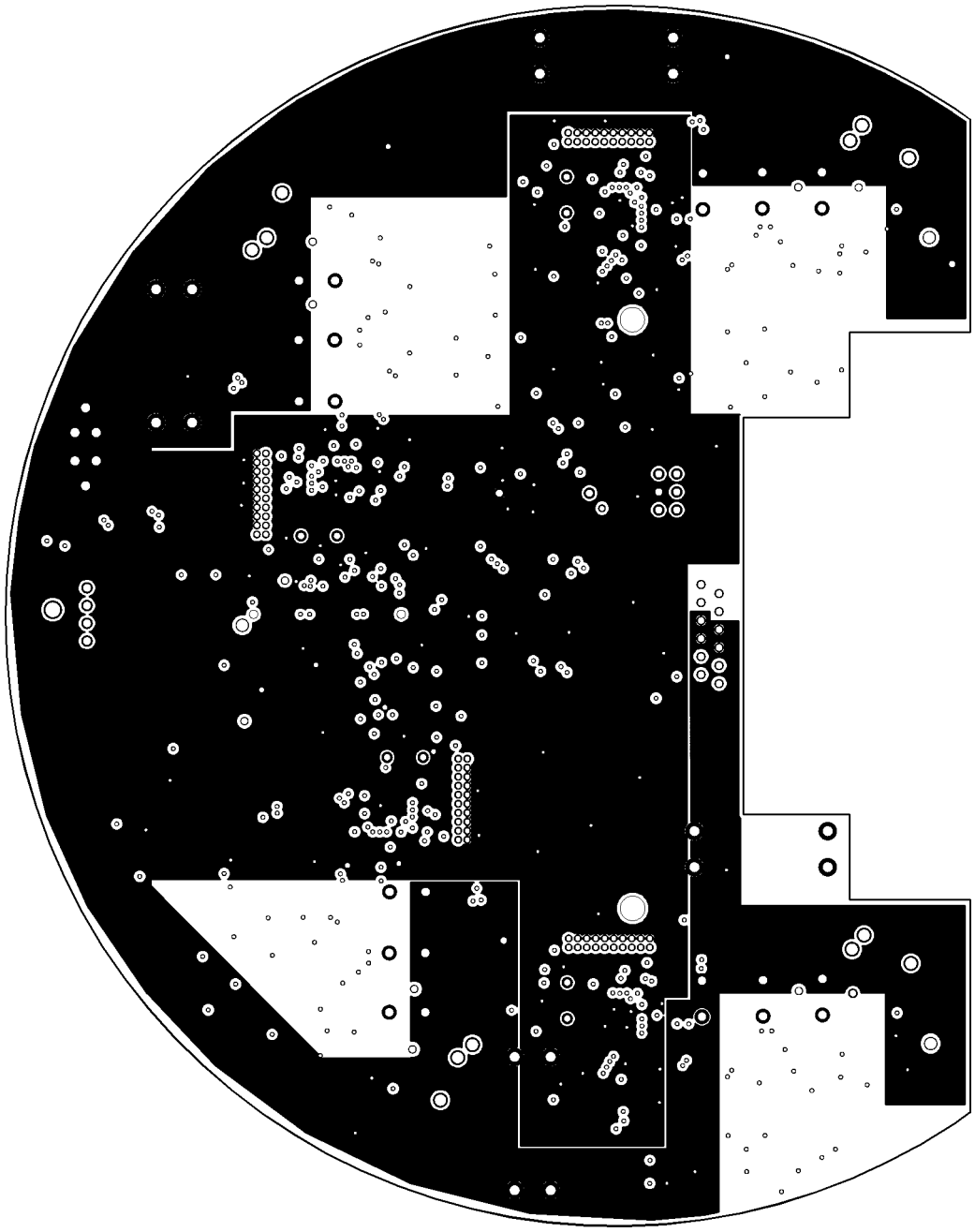


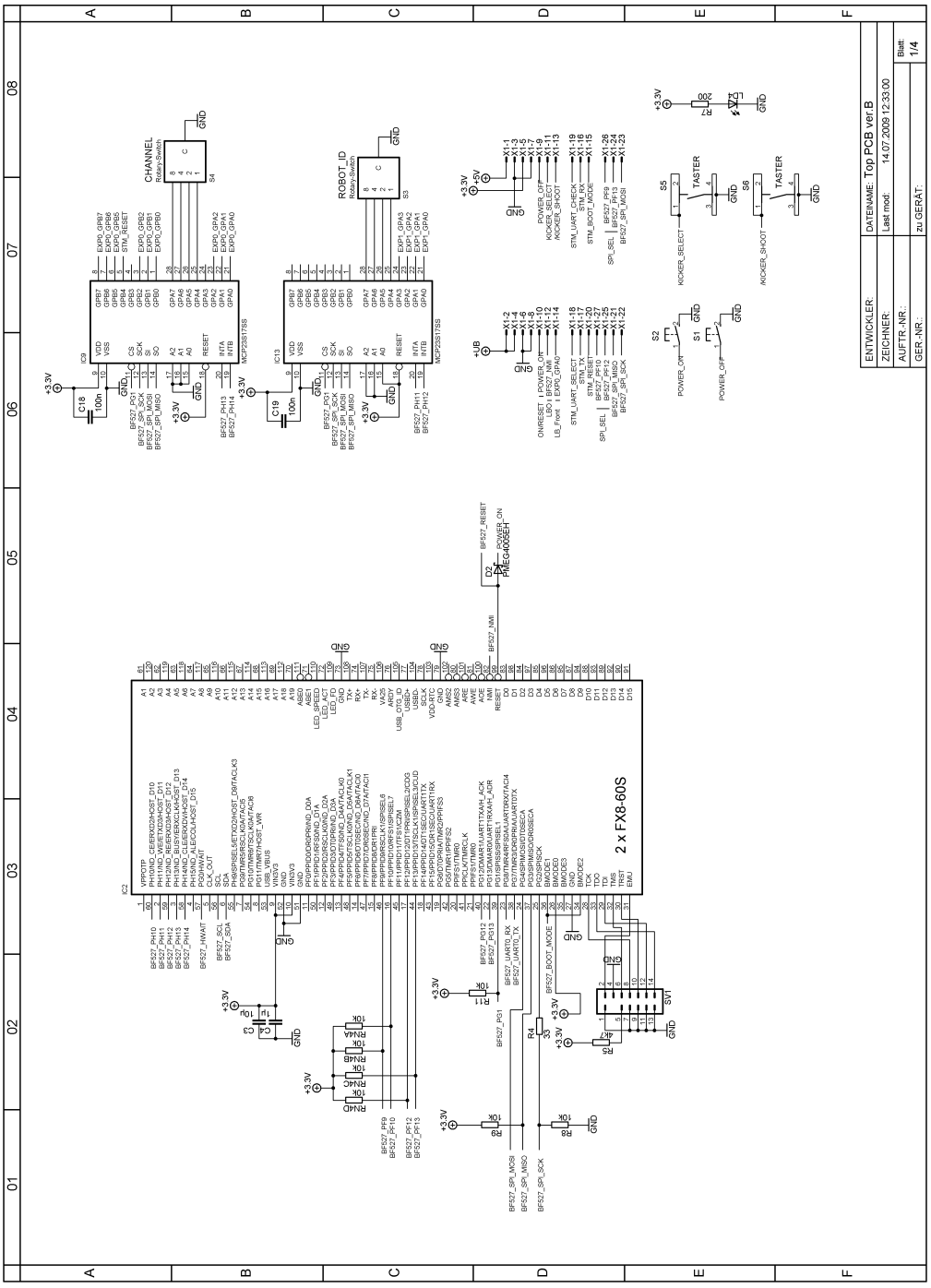
Abbildung A.13.: Ground-Layer von Bottom ver.D

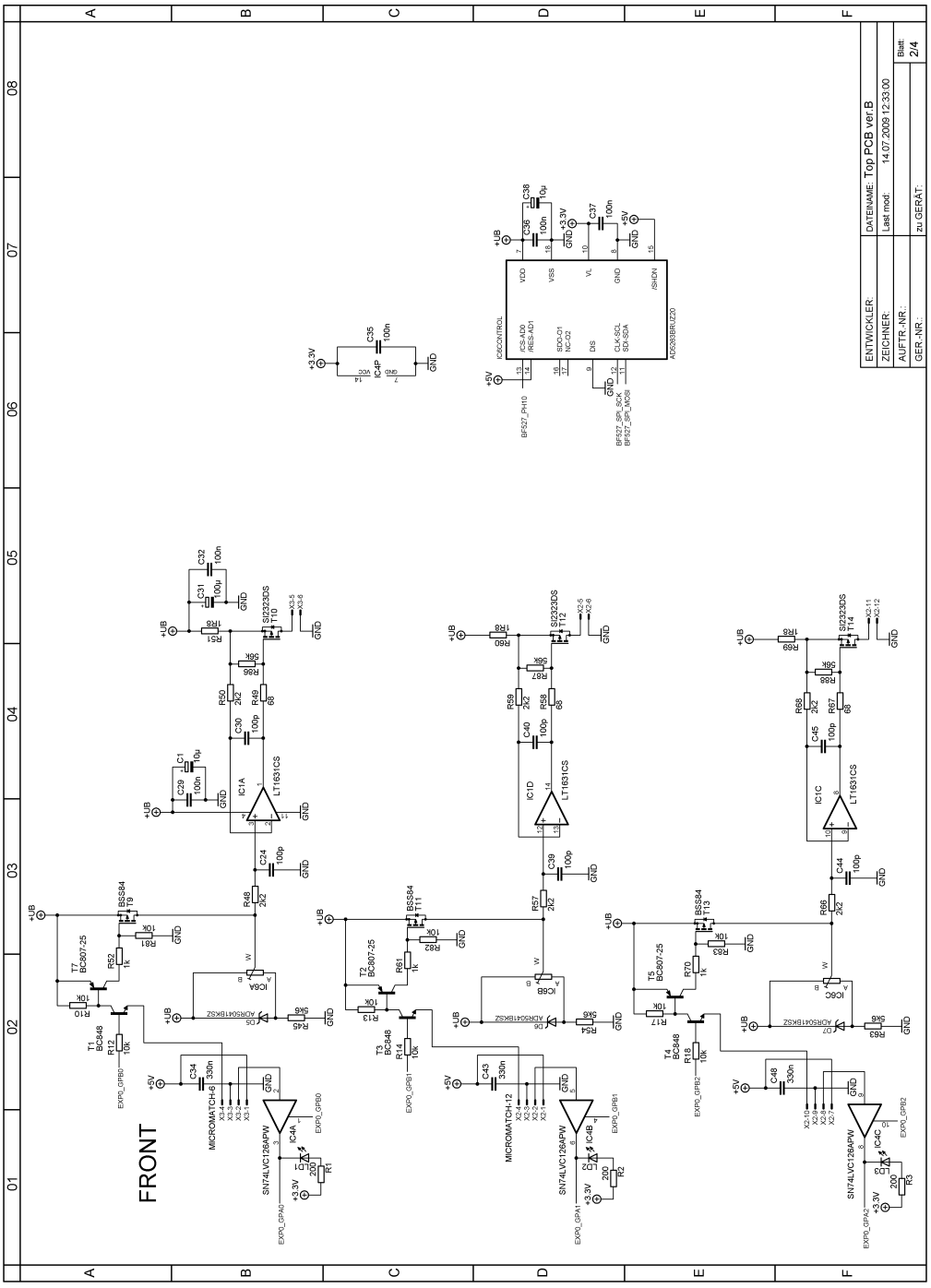
A.4. Top ver.B

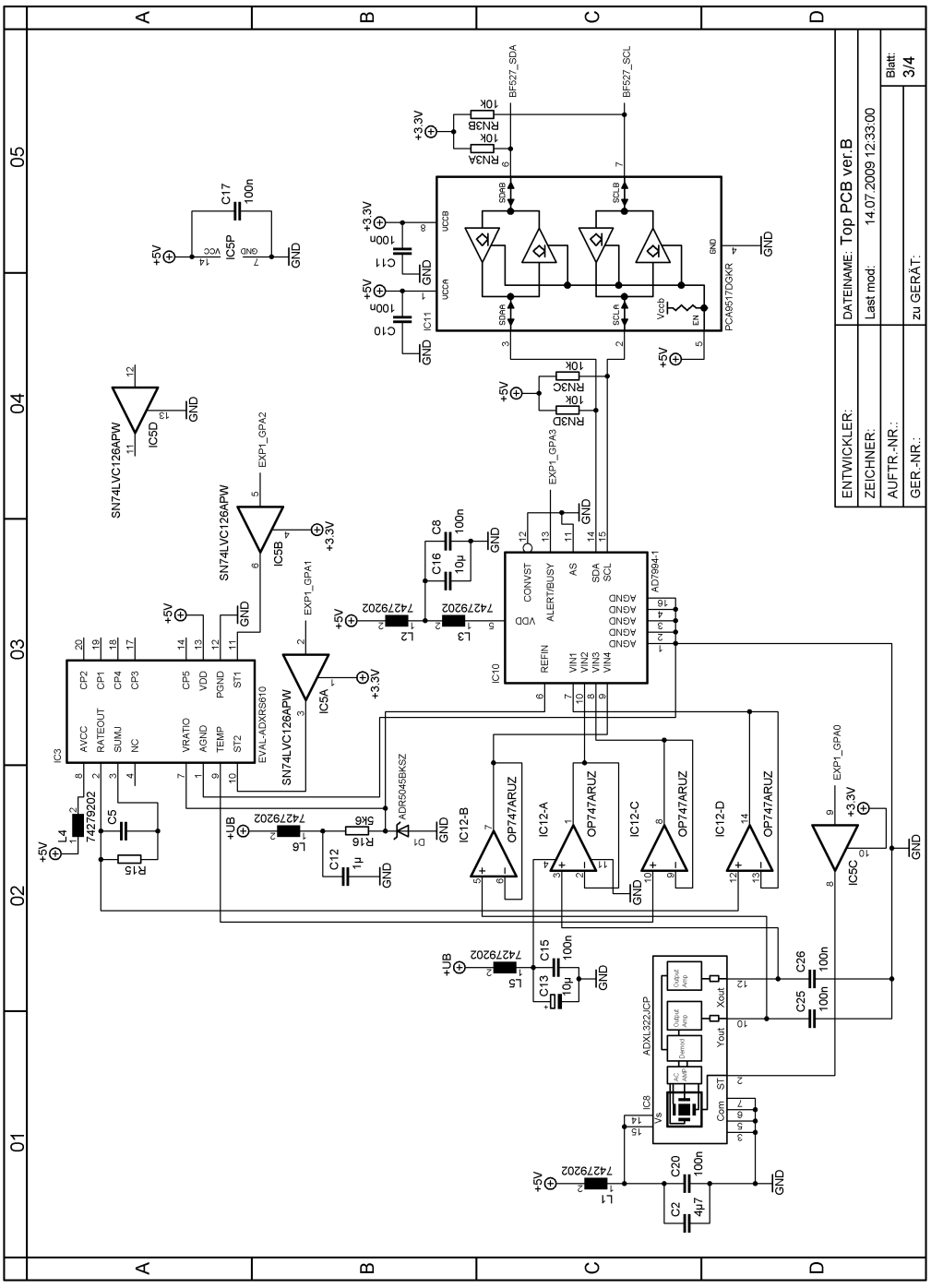
Menge	Wert	Device	Bauteil
3	10µ	CPOL-EU153CLV-0505	C1, C13, C38
1	1µ	C-EUC1206	C12
1	4µ7	C-EUC1206	C2
6	100p	C-EUC0805	C24, C30, C39, C40, C44, C45
4	10µ	C-EUC1206	C3, C7, C9, C16
1	100µ	CPOL-EU153CLV-0810	C31
3	330n	C-EUC0603	C34, C43, C48
1	1µ	C-EUC0805	C4
1		C-EUC0805	C5
15	100n	C-EUC0603	C8, C10, C11, C15, C17, C18, C19, C20, C25, C26, C29, C32, C35, C36, C37
1	ADR5045BKSZ	MAX6138	D1
1	PMEG4005EH	SCHOTTKY-DIODESOD123	D2
3	ADR5041BKSZ	MAX6138	D5, D6, D7
1	LT1631CS	LT1631CS	IC1
1	AD7994-1	AD7994-1	IC10
1	PCA9517DGKR	PCA9517DGKR	IC11
1	OP747ARUZ	OP747ARUZ	IC12
1	TMM-110	ROBOCUP_WIRELESS_MODULE-TMM	IC16
1	2 x FX8-60S	CM-BF527-V1.1-F	IC2
1	EVAL-ADXRS610	EVAL-ADXRS610	IC3
2	SN74LVC126APW	SN74LVC126APW	IC4, IC5
1	AD5263BRUZ20	AD5263BRUZ	IC6
1	ADXL322JCP	ADXL322JCP	IC8
2	MCP23S17SS	MCP23S17SS	IC9, IC13
6	74279202	WE-CBF_0805	L1, L2, L3, L4, L5, L6
4		LEDCHIP-LED0805	LD1, LD2, LD3, LD4
4	200	R-EU_R0805	R1, R2, R3, R7
1		R-EU_R0805	R15
4	5k6	R-EU_R0805	R16, R45, R54, R63
1	33	R-EU_R0805	R4
6	2k2	R-EU_R0805	R48, R50, R57, R59, R66, R68
3	68	R-EU_R0805	R49, R58, R67
1	4k7	R-EU_R0805	R5
3	1R8	R-EU_R1210	R51, R60, R69
3	1k	R-EU_R0805	R52, R61, R70

12	10k	R-EU_R0805	R8, R9, R10, R11, R12, R13, R14, R17, R18, R81, R82, R83
3	56k	R-EU_R0805	R86, R87, R88
2	10k	RN-4P1206	RN3, RN4
2	SWITCH-VERTICAL	SWITCH-VERTICAL	S1, S2
2	Rotary-Switch	ROTARY_SWITCH_16	S3, S4
2	TASTER	TASTER-9314_SMD	S5, S6
1		MA07-2-PIN3	SV1
3	BC848	BCX70SMD	T1, T3, T4
3	SI2323DS	BSS84	T10, T12, T14
3	BC807-25	BCX71SMD	T2, T5, T7
3	BSS84	BSS84	T9, T11, T13
1	TYCO 5104493-3	CON_AMPMODU_HEA_30	X1
1	MICROMATCH-12	MICROMATCH-12	X2
1	MICROMATCH-6	MICROMATCH-6	X3

Tabelle A.4.: Materialliste von Top ver.B


ENTWICKLER:
DATEINAME: Top PCB ver.B
ZEICHNER:
Last mod: 14.07.2009 12:33:00
AUFTR.-NR.:
Blatt: 1/4
GER.-NR.:
zu.GERÄT:
2 x FX8-60S
ATMEGA32U4
CHANNEL
ROBOT_ID
MCP23S17SS
+3.3V
+5V
+UB
GND


FRONT
ENTWICKLER:
ZEICHNER:
AUFTR.-NR.:
GER.-NR.:
DATEINAME: Top PCB ver.B
Last mod: 14.07.2009 12:33:00
zu.GERÄT:
Blatt: 2/4
ADXRS300BRGZ20
LT1631CS
SN74LVC126APW
MICROMATCH-8
MICROMATCH-12
BSS84
BC807-25
BC848
SI2323DS
AOR504TBKSZ
ADR504TBKSZ


ENTWICKLER:
ZEICHNER:
AUFTR.-NR.:
GER.-NR.:
DATEINAME: Top PCB ver.B
Last mod: 14.07.2009 12:33:00
zu GERÄT:
Blatt: 3/4
SN74LVC126APW
EVAL-ADXRS610
ADR5045BKSZ
AD7994-1
PCA9517DGKR
ADXL322-ICP
OP747ARUZ
74279202
BF527_SDA
BF527_SCL
EXP1_GPA0
EXP1_GPA1
EXP1_GPA2
EXP1_GPA3
+5V
+3.3V
+UB
GND

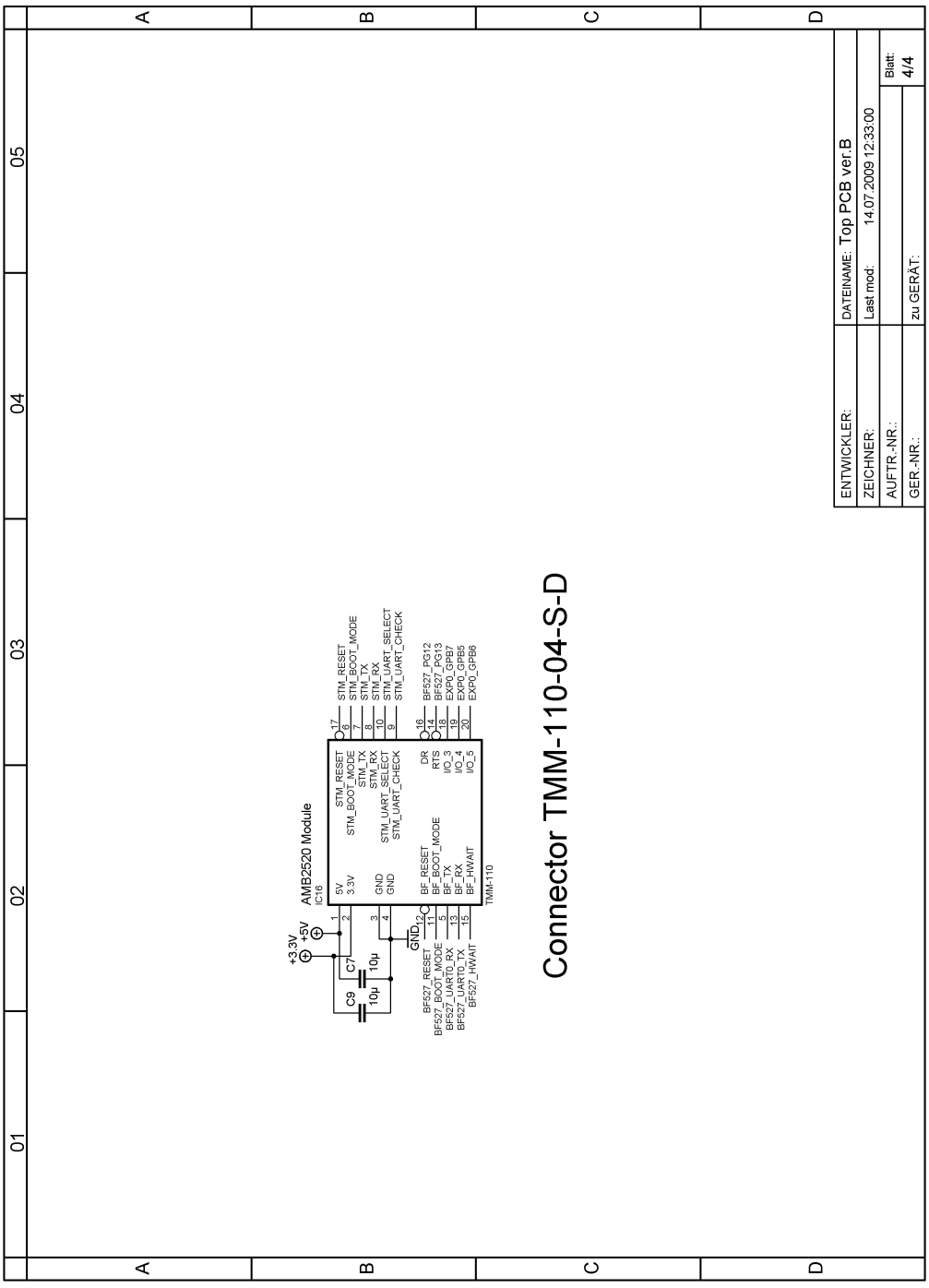


Abbildung A.14.: Schaltplan von Top ver.B

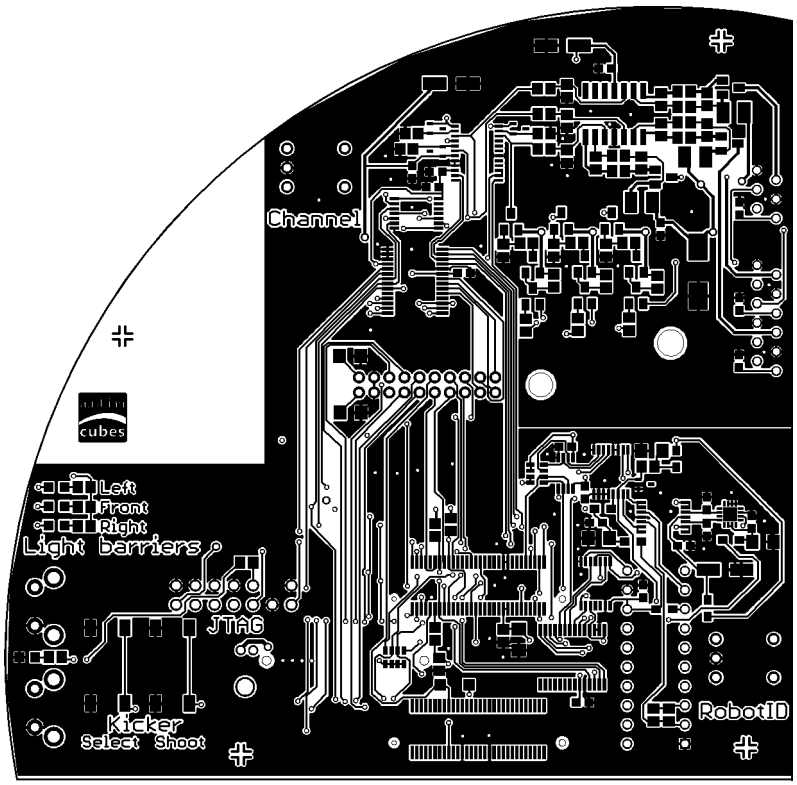


Abbildung A.15.: Top-Layer von Top ver.B

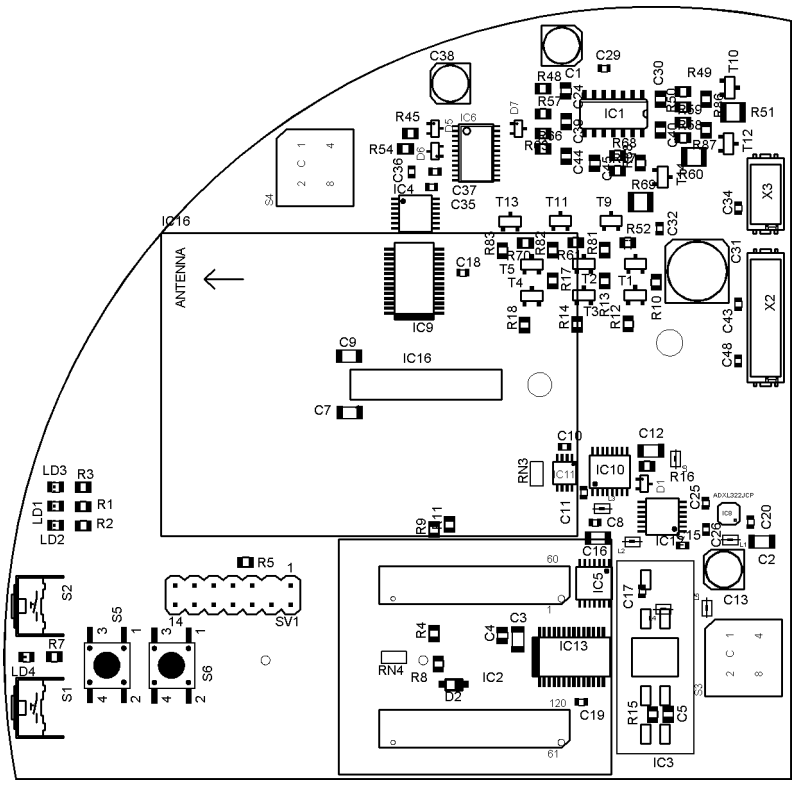


Abbildung A.16.: Top-Layer Bestückung von Top ver.B

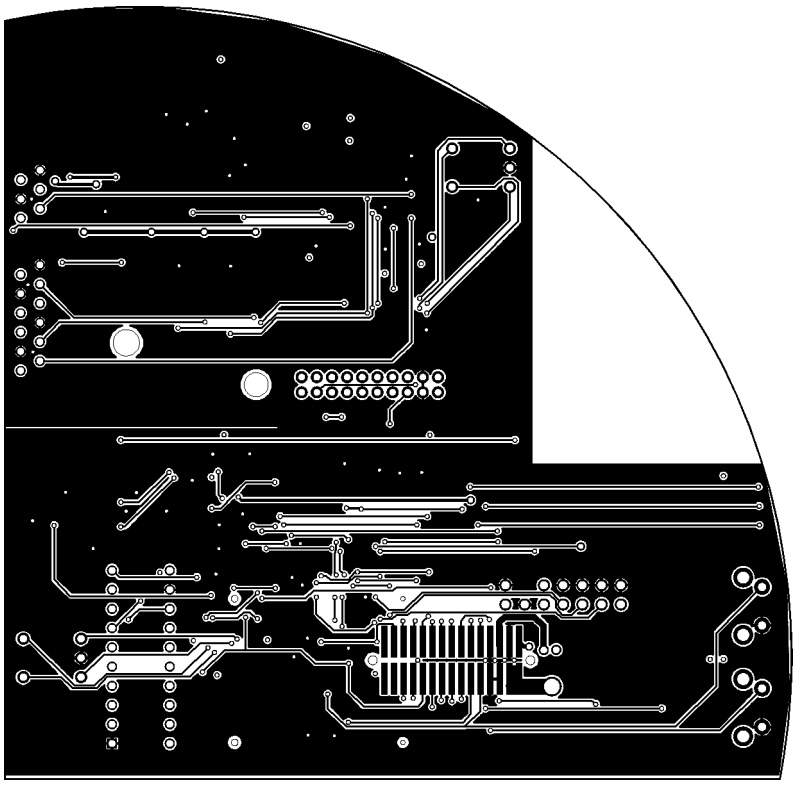


Abbildung A.17.: Bottom-Layer von Top ver.B

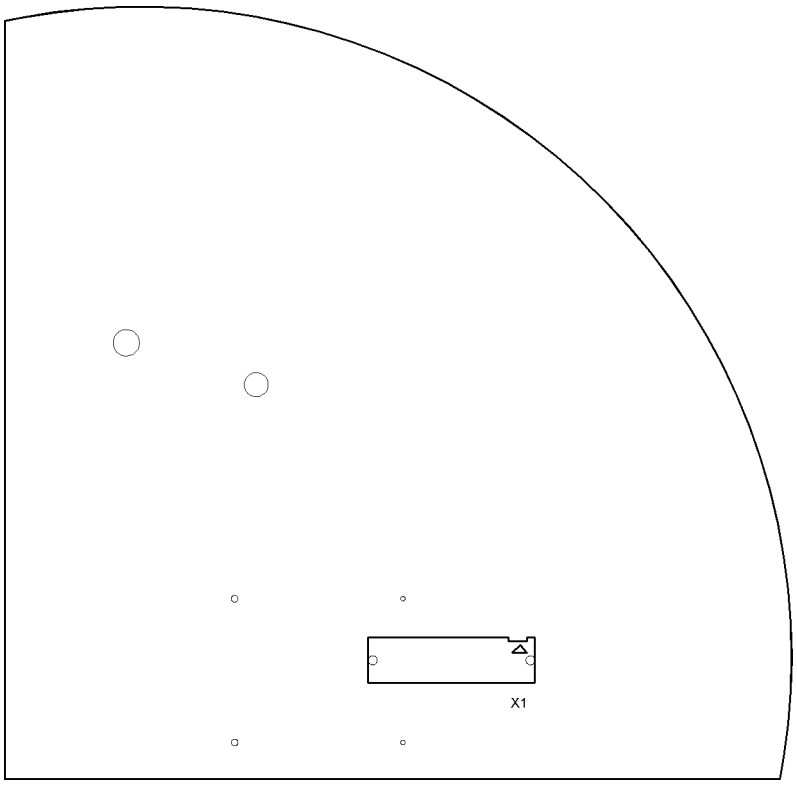


Abbildung A.18.: Bottom-Layer Bestückung von Top ver.B

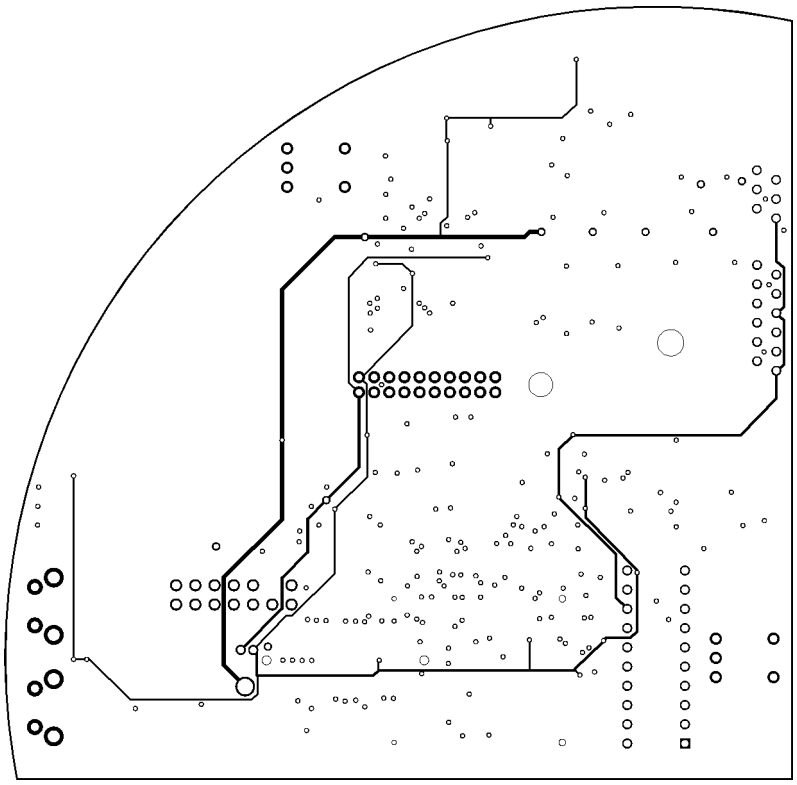


Abbildung A.19.: Power-Layer von Top ver.B

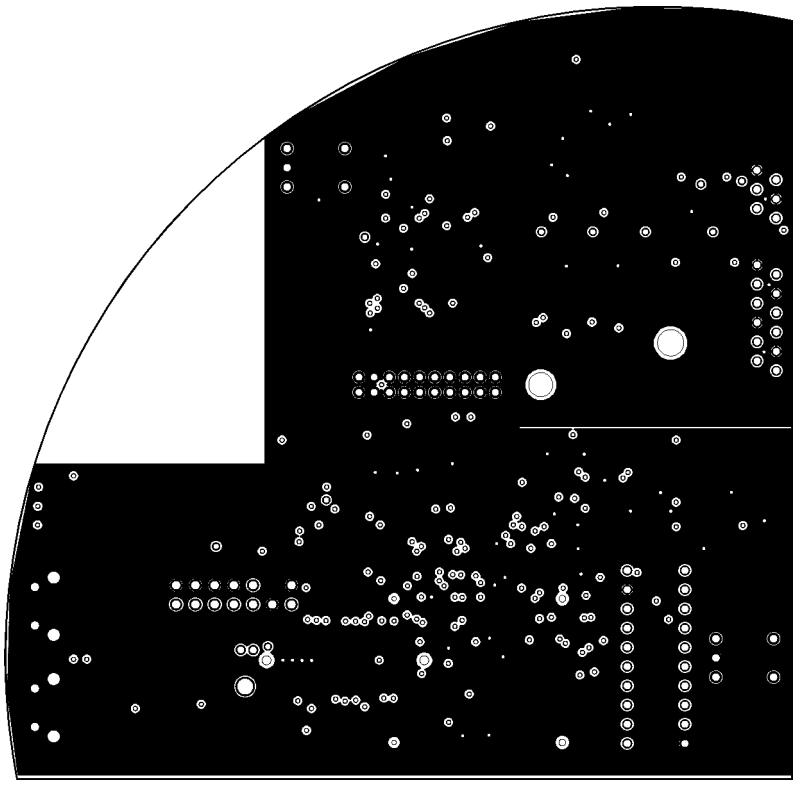


Abbildung A.20.: Ground-Layer von Top ver.B

A.5. Lichtschrankenhalter ver.A

Menge	Wert	Device	Bauteil
1	MICROMATCH-12	MICROMATCH-12	X1
2	IS471F	IS471F	IC1, IC2
2	TSAL4400	LED3MM	LD1, LD2

Tabelle A.5.: Materialliste von Lichtschrankenhalter ver.A

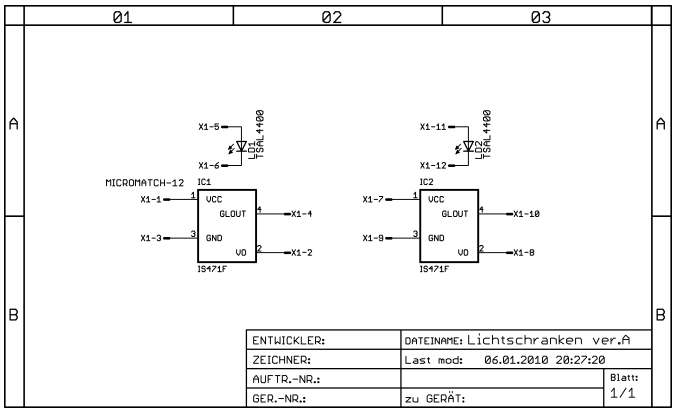


Abbildung A.21.: Schaltplan von Lichtschrankenhalter ver.A

Abbildung A.22.: Bottom-Layer von Lichtschrankenhalter ver.A

Abbildung A.23.: Top-Layer von Lichtschrankenhalter ver.A

A.6. Funkmodul ver.D

Menge	Wert	Device	Bauteil
2		LEDCHIPLED_0805	LED1, LED2
1	1M	R-EU_R0805	R5
1	8Mhz	XTAL/S	Q1
1	10µ(Tan)	CPOL-EUCT3216	C9
2	10k	R-EU_R0805	R3, R4
2	10k	RN-4P1206	RN1, RN2
2	33p	C-EUC0805	C2, C3
1	100µ	CPOL-EU153CLV-0807	C4
4	100n	C-EUC0603	C1, C6, C7, C8
1	100n	C-EUC0805	C5
2	150	R-EU_M0805	R1, R2
1	AMB2520	AMB2520	IC3
1	EHF-10	EHF-10	X1
1	WIRELESS_MODULE-SMM	WIRELESS_MODULE-SMM	IC2
1	STM32F10XCXT6	STM32F10XCXT6	IC1

Tabelle A.6.: Materialliste von Funkmodul ver.D

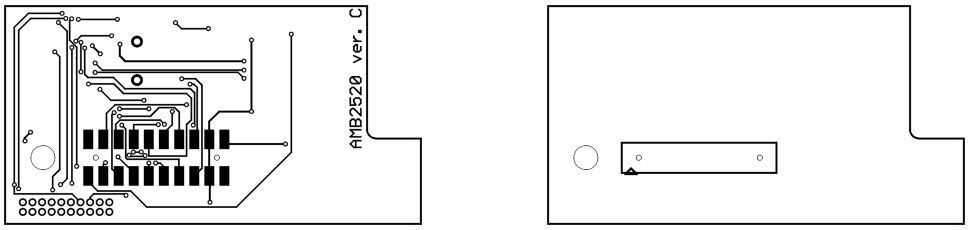


Abbildung A.24.: Bottom-Layer von Funkmodul ver.D

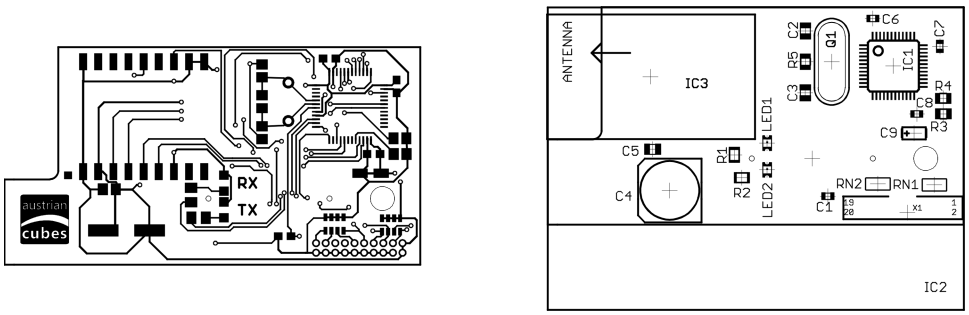


Abbildung A.25.: Top-Layer von Funkmodul ver.D

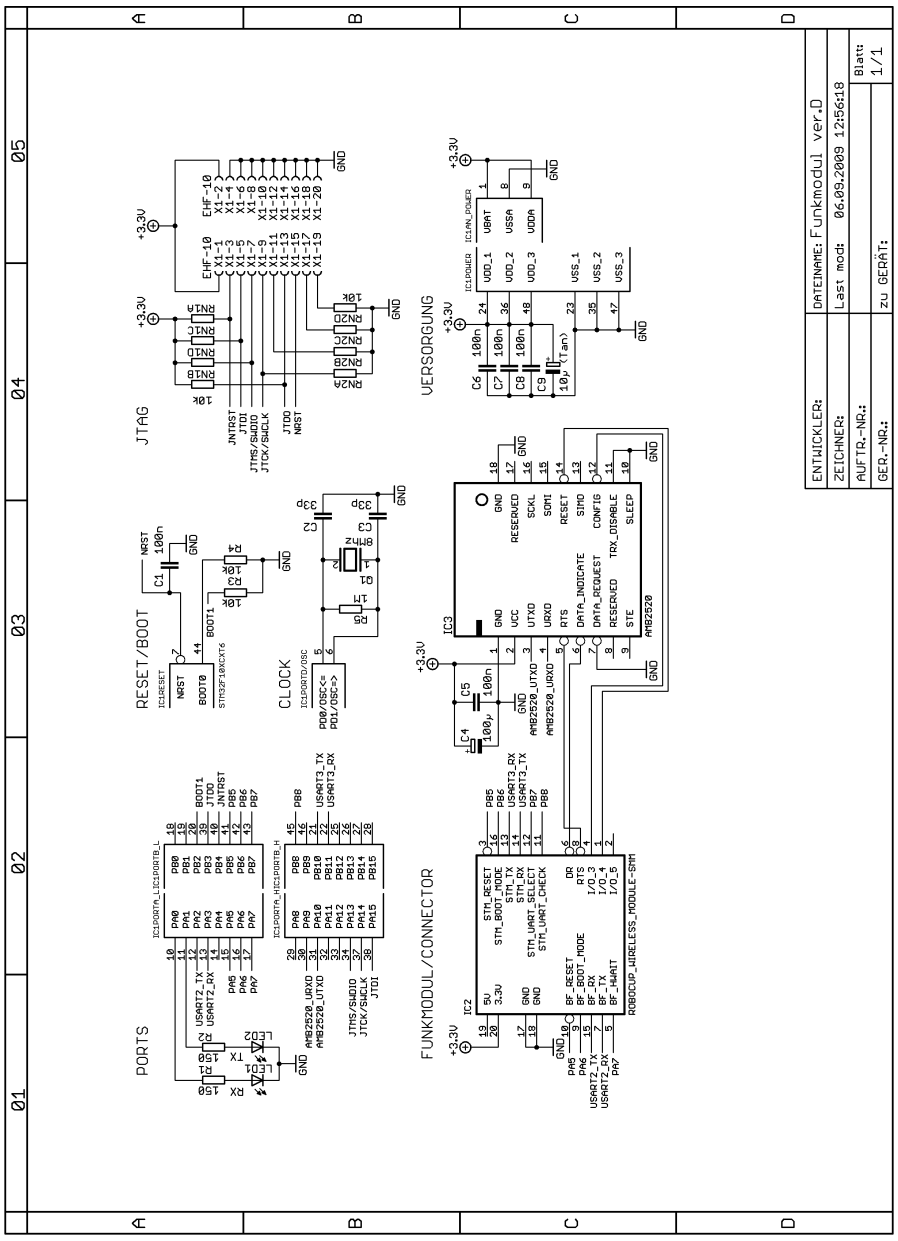


Abbildung A.26.: Schaltplan von Funkmodul ver.D

A.7. Funkmodul PC v1.1

Menge	Wert	Device	Bauteil
1	10µ	C-EUC1210	C1
3	100n	C-EUC0603	C2, C4, C5
1	100µ	CPOL-EU153CLV-0605	C3
1	TPD2S017DBVR	TPD2S017DBVR	IC1
1	CP2102	CP2102	IC2
1	AMB2520	AMB2520	IC3
1	0	WE-CBF_0805	L1
1		LEDCHIP-LED0805	LD
1	0	R-EU_R0805	R1
1	100	R-EU_R0805	R2
1	680	R-EU_R0805	R3
1		31-XX	S1
1		PN61729	X1

Tabelle A.7.: Materialliste von Funkmodul PC v1.1

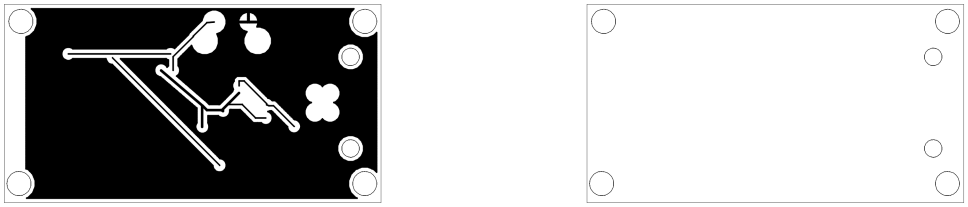


Abbildung A.27.: Bottom-Layer von Funkmodul PC v1.1

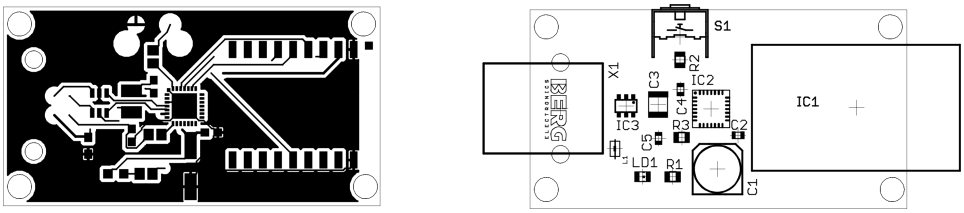


Abbildung A.28.: Top-Layer von Funkmodul PC v1.1

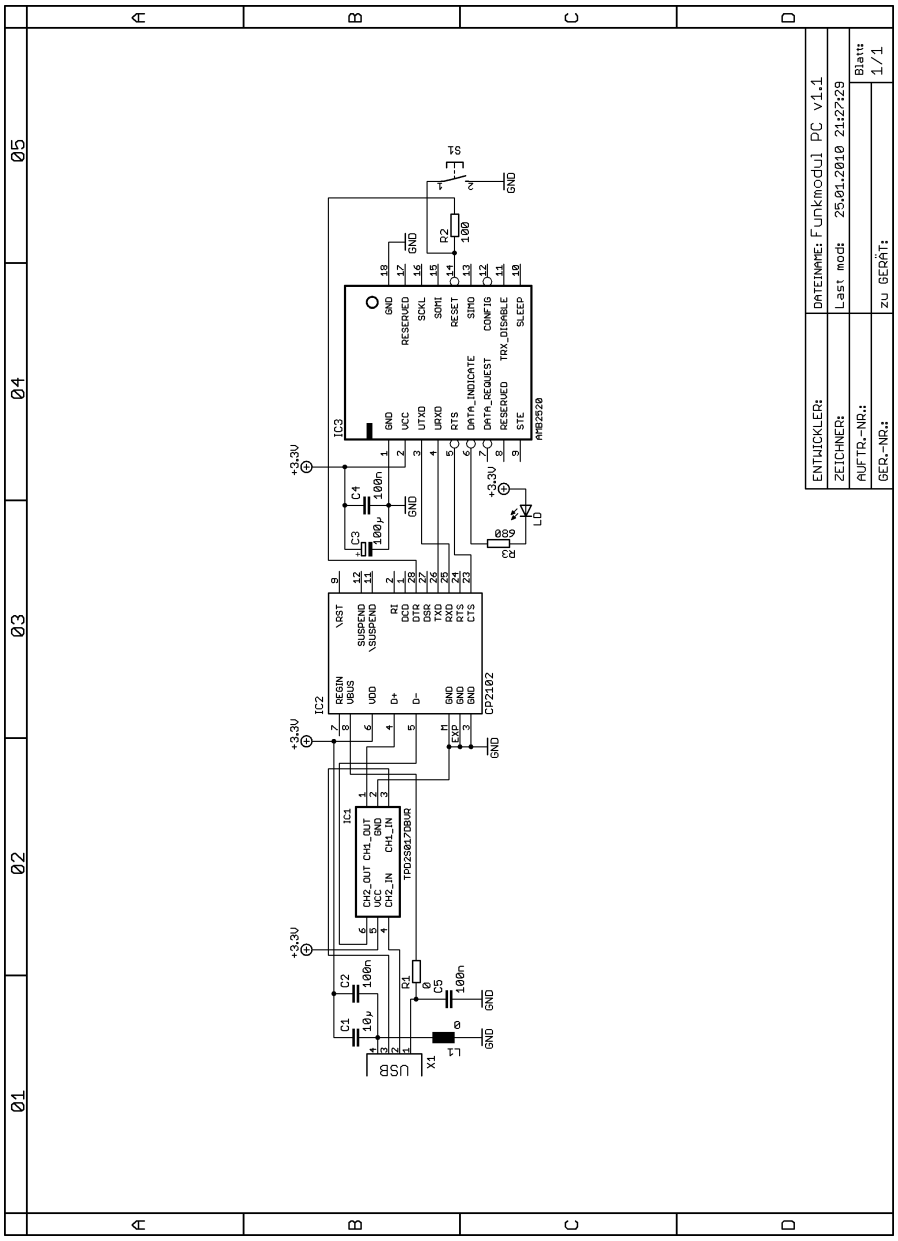


Abbildung A.29.: Schaltplan von Funkmodul PC v1.1

A.8. Encoder Adapter v1.0

Menge	Wert	Device	Bauteil
1	100n	C-EUC0603	C1
1	AS5134	AS5134	IC1
1	0R-JUMPA	0R-JUMPA	JMP1
1		JP1E	JP1
1	ZIF_6_1MM	ZIF_6_1MM	X1
1	MICROMATCH-12	MICROMATCH-12	X2

Tabelle A.8.: Materialliste von Encoder Adapter v1.0

Abbildung A.30.: Bottom-Layer von Encoder Adapter v1.0

Abbildung A.31.: Top-Layer von Encoder Adapter v1.0

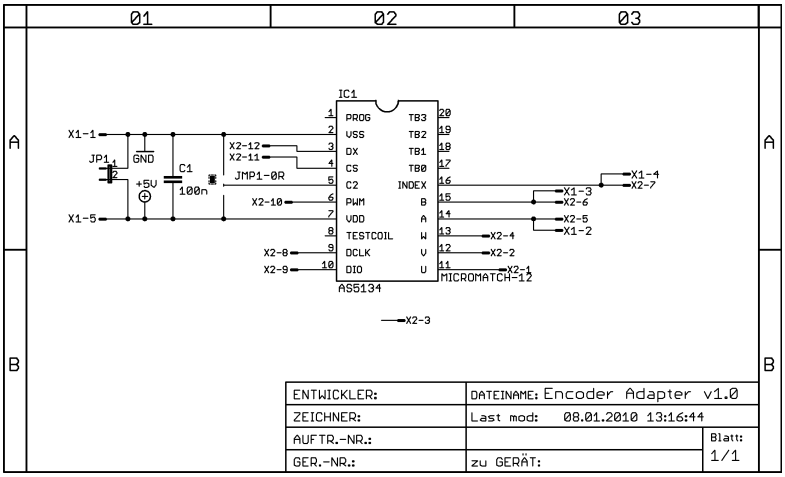


Abbildung A.32.: Schaltplan von Encoder Adapter v1.0

A.9. Interface Adapter v1.0

Menge	Wert	Device	Bauteil
1	0522071160	0522071160	X1
1	ZIF_6_1MM	ZIF_6_1MM	X2
1	FTMH-H11	FTMH-H11	X3

Tabelle A.9.: Materialliste von Interface Adapter v1.0

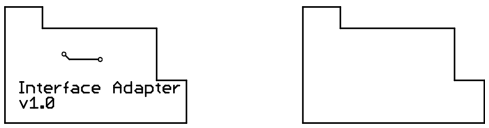


Abbildung A.33.: Bottom-Layer von Interface Adapter v1.0

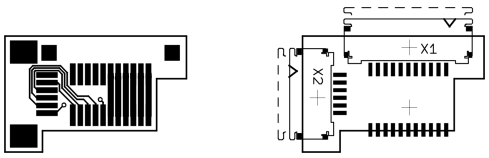


Abbildung A.34.: Top-Layer von Interface Adapter v1.0

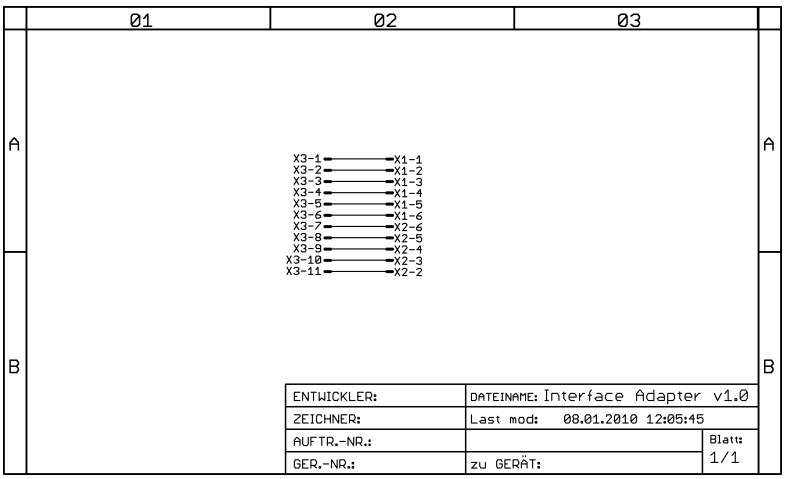


Abbildung A.35.: Schaltplan von Interface Adapter v1.0

B Literaturverzeichnis

[1] *How Electric Motors Work*.
www.stefanv.com/rcstuff/qf200212.html.

[2] *Wikipedia - Elektromotorische Kraft*.
de.wikipedia.org/wiki/Elektromotorische_Kraft.

[3] Amber Wireless. *Kompaktes Low-Cost 2,4 GHz Funkmodul AMB2520*. amber-wireless.de/42-0-AMB2520.html.

[4] Analog Devices. *AD5263*.
www.analog.com/static/imported-files/data_sheets/AD5263.pdf.

[5] Analog Devices. *AD7994*.
www.analog.com/static/imported-files/data_sheets/AD7993_7994.pdf
www.analog.com/static/imported-files/data_sheets/AD7993_7994ERRATA.pdf.

[6] Analog Devices. *ADR5041*.
www.analog.com/static/imported-files/data_sheets/ADR5040_5041_5043_5044_5045.pdf.

[7] Analog Devices. *ADR5045B*.
www.analog.com/static/imported-files/data_sheets/ADR5040_5041_5043_5044_5045.pdf.

[8] Analog Devices. *ADXL322*.
www.analog.com/static/imported-files/data_sheets/ADXL322.pdf.

[9] Analog Devices. *ADXRS610*.
www.analog.com/static/imported-files/data_sheets/ADXRS610.pdf.

[10] Analog Devices. *BF527*.
www.analog.com/static/imported-files/data_sheets/ADSP-BF522_BF523_BF524_BF525_BF526_BF527.pdf
www.analog.com/static/imported-files/processor_manuals/BF52xProcHWR031.pdf
Hardware Reference Manual Volume 2 of 2
www.analog.com/static/imported-files/ic_anom/ADSP-BF523_BF525_BF527(C)_anomaly_Rev.G_%20082509.pdf.

[11] Analog Devices. *EVAL-ADXRS610*.
www.analog.com/static/imported-files/eval_boards/EVAL-ADXRS610.pdf.

[12] Analog Devices. *OP747*.
www.analog.com/static/imported-files/data_sheets/OP777_727_747.pdf.

[13] Bernhard Rall Heinz Zenkner Dr. Thomas Brandner, Alexander Gerfer. *Trilogie der induktiven Bauelemente*. Number 978-3-89929-151-3. Würth Elektronik eiSos GmbH & Co. KG, 4. edition, November 2008.

[14] Sharp Electronics. *IS471F*.
pdfdata.datasheetsite.com/web/42749/IS471F.pdf.

[15] Faulhaber. *2224SR*.
www.faulhaber.com/uploadpk/DE_2224SR_DFF.pdf.

[16] PD Dr.-Ing. habil. W. Michalik. *Gleichstrommaschine mit elektronischem Kommutator*.
skrausz.net/daxue/aktorik/elektr.antr/K5a.pdf.

[17] Microchip Technology Inc. *MCP23S17*.
ww1.microchip.com/downloads/en/DeviceDoc/21952b.pdf.

[18] Texas Instruments. *CC2500*.
www.ti.com/lit/gpn/cc2500.

[19] Texas Instruments. *DRV8800*.
focus.ti.com/lit/an/slva320/slva320.pdf.

[20] Texas Instruments. *INA2132*.
www.ti.com/lit/gpn/ina2132.

[21] Texas Instruments. *MSP430F1232*.
www.ti.com/lit/gpn/msp430f1232.

[22] Texas Instruments. *SN74ABT125*.
www.ti.com/lit/gpn/sn74abt125.

[23] Texas Instruments. *SN74LVC1G175*.
www.ti.com/lit/gpn/sn74lvc1g175.

[24] Texas Instruments. *TLC555*.
www.ti.com/lit/gpn/tlc555.

[25] Wolfgang Korosec. *Elektroantriebe für Modellflugzeuge*.
www.mfv-arbon.ch/cms/fileadmin/PDFs/Elektrische_Antriebe_fuer_Modellflugzeuge.pdf.

[26] Silicon Labs. *CP2102*.
www.silabs.com/Support%20Documents/TechnicalDocs/cp2102.pdf.

[27] Justin Yance Jae Lew Robert L. Williams II Paolo Gallina Lance Wilson, Craig Williams. *Design and Modeling of a Redundant Omni-directional RoboCup Goalie*.
http://www.ent.ohiou.edu/~bobw/PDF/RoboCup01g.pdfwww.ent.ohiou.edu/ bobw/PDF/RoboCup01g.pdf.

[28] Marc Vila Mani. *A quick overview on rotatory Brush and Brushless DC Motors*.
www.ingenia-cat.com/reference/learn/TEC.PAP.7055681008.pdf.

[29] Christian Rabitsch Martin Mitterer. *Projekt Konstruktiv - Projekt Chipkick*. TU Graz - Konstruktionslehre Maschinenelemente.

[30] maxon motor. *EC Technik - Kurz und bündig*.
www.kwapil.com/downloads/technikkurzundbuendig.pdf.

[31] NXP. *PCA9517*.
www.nxp.com/documents/data_sheet/PCA9517.pdf.

[32] Maxim Integrated Products. *MAX1614*.
datasheets.maxim-ic.com/en/ds/MAX1614.pdf.

[33] Maxim Integrated Products. *MAX668*.
datasheets.maxim-ic.com/en/ds/MAX668-MAX669.pdf.

[34] Raul Rojas. *Omnidirectional Control*.
robocup.mi.fu-berlin.de/buch/omnidrive.pdf.

[35] Segger. *J-Link ARM Emulator for ARM and Cortex-M3 cores*.
www.segger.com/cms/admin/uploads/productDocs/UM08001_JLinkARM.pdf.

[36] Dipl.-Ing. Daniel Steinmair. *Diplomarbeit - Tinyphoon - Entwicklung einer Hardwareplattform für einen autonomen Fußballroboter*. TU Wien - Institut für Computertechnik, 2006.

[37] STMicroelectronics. *Flash loader demonstrator*.
www.st.com/stonline/products/literature/um/13916.pdf.

[38] STMicroelectronics. *STM1061N31*.
eu.st.com/stonline/books/pdf/docs/11595.pdf.

[39] STMicroelectronics. *STM32F101xx, STM32F102xx and STM32F103xx system memory boot mode*.
www.st.com/stonline/products/literature/an/14156.pdf.

[40] STMicroelectronics. *STM32F103C8T6*.
www.st.com/stonline/products/literature/ds/13587.pdf.

[41] STMicroelectronics. *TSV994*.
www.st.com/stonline/products/literature/ds/12833.pdf.

[42] Bluetechnix Mechatronische Systeme. *CM-BF527*.
www.bluetechnix.at/rainbow2006/site/blackfin_family/__core_modules/__cm-bf527/397/cm-bf527.aspx.

[43] Linear Technology. *LT1631*.
cds.linear.com/docs/Datasheet/16301fs.pdf.

[44] Linear Technology. *LT3500*.
cds.linear.com/docs/Datasheet/3500fb.pdf.

[45] Vishay. *SI4500*.
www.vishay.com/docs/70880/70880.pdf.

[46] Vishay. *TSAL4400*.
www.vishay.com/docs/81006/81006.pdf.

[47] Amber Wireless. *AMB2520*.
amber-wireless.de/files/amb8420_2520_hb.pdf.

[48] Padmaraja Yedamale. *AN885 - Brushless DC (BLDC) Motor Fundamentals*. Microchip Technology Inc., 2003.
ww1.microchip.com/downloads/en/appnotes/00885a.pdf.