Inverses Pendel II

Elektronische Produktentwicklung II (MECH-B-5-AED-EPE2-ILV)

Bachelor: Mechatronik Design und Innovation

5. Semester

Lehrveranstaltungsleiter: Matthias Gfall

Gruppe: BA-MECH-23

Verfasser: Stefan Hörtnagl

3. Januar 2026

Einleitung

Das Projekt „Inverses Pendel“ wurde im Rahmen der Vorlesungen „Elektronische Produktentwicklung I & II“ entwickelt, aufgebaut und in Betrieb genommen. Der folgende Bericht behandelt den Entwicklungsprozess von der digitalen Konzeptionierung bis hin zum Zusammenbau der einzelnen Komponenten und der Inbetriebnahme.

Dabei werden alle nötigen Informationen bereitgestellt, damit das gesamte Projekt nach Ende der Vorlesung an eine neue Gruppe für die nachfolgende Vorlesung übergeben werden kann. Dazu gehören die Funktions- und Vorgehensweisen bei der Entwicklung sowie die Hardware. Zudem werden Empfehlungen für weitere Verbesserungen und Vereinfachungen gegeben, um den Komplexitätsgrad zu reduzieren und das Projekt für den finalen Einsatz als Demonstrator in der Vorlesung „Regelungstechnik" zu optimieren.

Als Grundlage für den Bericht ist die Dokumentation zur Entwicklung des PCB aus der vorhergegangenen Vorlesung zu berücksichtigen [1]. Dort wird der Schaltplan des PCB detailliert dokumentiert und in den folgenden Kapiteln weiter vervollständigt. Die elektronischen Komponenten der Peripherie blieben unverändert und es mussten für die Inbetriebnahme keine größeren Änderungen am System vorgenommen werden.

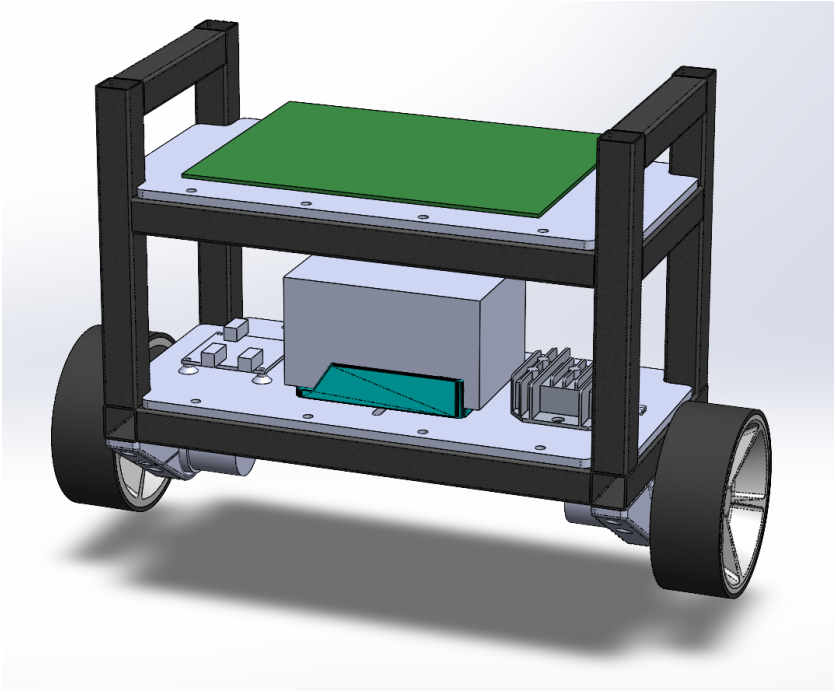


Abbildung 0.1: Finale Version der Hardware

Zur Unterstützung der weiteren Entwicklung wurden Verbesserungsvorschläge und Anreize für Optimierungen genannt, um der nachfolgenden Gruppe eine Richtung zu geben, in die das Produkt weiter verbessert werden kann. Abbildung 0.1 zeigt die endgültige Version der mechanischen Entwicklung.

Um den Laborbericht zu übersetzen und den Sprachstil zu verbessern, ohne den Inhalt zu verändern, wurden Übersetzungstools verwendet [2].

Die Gesamten projektdaten sind hierbei im folgenden Github Repository verfügbar [3].

Inhaltsverzeichnis

1 Projektziel und Projektstand

1.1 PROJEKTZIEL

Ziel des Projekts ist die Entwicklung eines Demonstrators für die Vorlesung „Regelungstechnik". Dieser soll es ermöglichen, Regelungsalgorithmen live über eine USB-Verbindung aus MATLAB zu programmieren und zu testen.

Die in MATLAB entwickelten Regelungen werden dabei vom Mikrocontroller übernommen, verarbeitet und in Echtzeit ausgeführt, um das Eigengewicht des Pendels aktiv zu stabilisieren. Die hierfür benötigten Messdaten werden über einen Gyrosensor erfasst.

Darüber hinaus ist vorgesehen, relevante System- und Sensordaten während des Betriebs kabellos über WLAN zu übertragen, sodass Live-Messwerte für Analyse und Visualisierung zur Verfügung stehen.

1.2 PROJEKTSTAND

Die mechanische Entwicklung des Projekts ist vollständig abgeschlossen. Das Gestell inklusive aller vorgesehenen Anbauteile wurde aufgebaut und geprüft. Ebenso ist die Leiterplatte vollständig bestückt und montiert.

Der Mikrocontroller lässt sich zuverlässig über die USB-Schnittstelle ansteuern und programmieren. Alle notwendigen Spannungsversorgungen wurden überprüft und arbeiten stabil innerhalb der vorgesehenen Toleranzen.

Damit sind die mechanischen und hardwareseitigen Voraussetzungen erfüllt. Das System befindet sich nun in einem funktionsfähigen Grundzustand und ist bereit für die weitere Entwicklung und Implementierung der Software.

Ergänzend wurde die Hardware so ausgelegt, dass spätere Erweiterungen und Anpassungen der Regelungsalgorithmen möglich sind. Die Parametrierung der Regelung, die Implementierung der WLAN-Datenübertragung und die Auswertung der Sensordaten sind Teil der noch ausstehenden Softwareentwicklung.

Das Projekt befindet sich somit in einem definierten Übergabestand: Die Hardware ist verifiziert und betriebsbereit, sodass sich die weitere Arbeit vollständig auf die Software- und Regelungsentwicklung konzentrieren kann. Nach Bedarf kann das Projekt somit ohne Veränderung der Hardware fortgeführt und die Software implementiert werden.

Das ist eine von zwei Möglichkeiten, mit dem Projekt fortzufahren. Die andere Möglichkeit ergibt sich aus der Optimierung der Platine und einem Redesign, um eine Reihe von Verbesserungen vorzunehmen. Dazu später mehr.

2 Zusammenbau der Platine

Zu Beginn der Bestückung wurden alle Bauteile kontrolliert und beschriftet, um bei der Platzierung Zeit zu sparen. Der Zusammenbau der Platine wurde in zwei Schritte unterteilt. Im ersten Schritt wurde das erste Drittel der Platine mithilfe einer Schablone mit Lötpaste benetzt und alle Bauteile wurden platziert.

Da diese nicht für das Backen im Ofen geeignet sind, wurden alle THT-Bauteile in einem dritten Zusammenbauschritt von Hand verlötet.

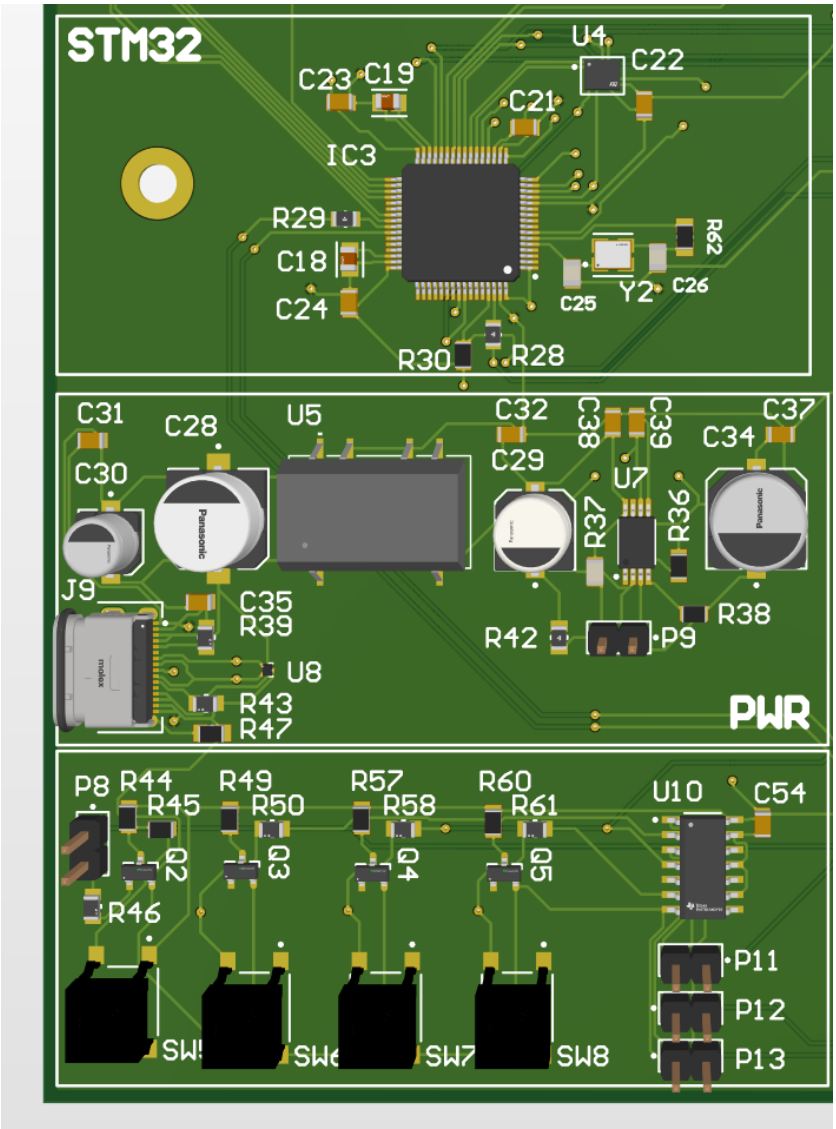


Abbildung 2.1: Erster Teil der Bestückung

Die in Abbildung 2.1 zu sehenden Bauteile wurden im ersten Schritt im Ofen gebacken und verlötet.
Die Platzierung des Mikrocontrollers und anderer kleinerer Bauteile erfolgte mithilfe eines Bestückungsgerätes.

Im zweiten Schritt wurde die restliche Platine wie in Abbildung 2.2 zu sehen ist, bestückt und gebacken. Hierzu wurde das Schablone in zwei Teile geschnitten, da sich vom ersten Schritt bereits Bauteile auf der Platine befanden.

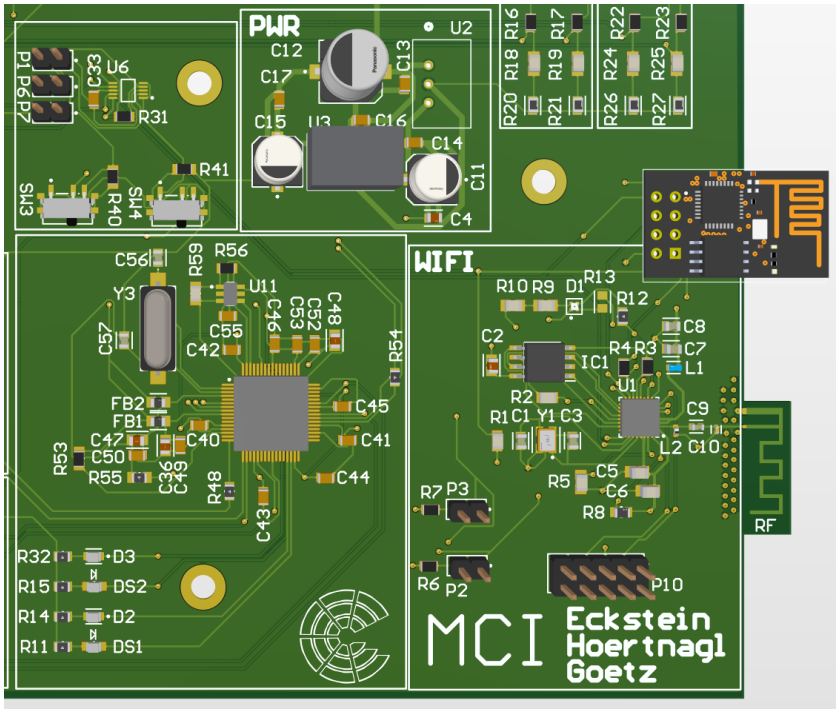


Abbildung 2.2: Zweiter Teil der Bestückung

Im dritten Schritt wurden alle THT-Bauteile von Hand verlötet. Das betraf hauptsächlich die Breakout-Pins und Kabelklemmen, wie in Abbildung 2.3 zu sehen ist.

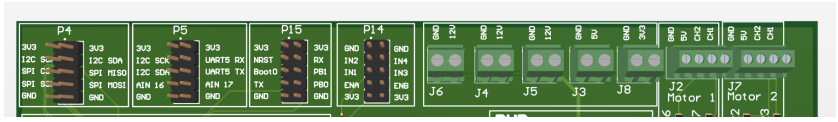


Abbildung 2.3: Dritter Teil der Bestückung

3 Dokumentation Inbetriebnahme

Während der Inbetriebnahme der Platine traten mehrere Herausforderungen auf, die jedoch erfolgreich behoben werden konnten. Die identifizierten Probleme, das angewandte Vorgehen und die erarbeiteten Lösungen sind in diesem Kapitel strukturiert dokumentiert.

3.1 PROBLEME

3.1.1 Spannungsversorgung

Die Platine verfügt über drei wesentliche Spannungsebenen, die für den Betrieb der verschiedenen Funktionsblöcke notwendig sind:

- 12 V Versorgung der Hardware aus dem Akku.
- 5 V Versorgung über die USB-C-Schnittstelle.
- 3,3 V Versorgung der Platinenkomponenten.

Ein zentrales Problem bestand darin, dass alle drei Spannungsebenen korrekt und stabil anliegen mussten, jedoch nicht gleichzeitig initialisiert werden sollten. Für die spätere Verwendung ist vorgesehen, dass die Akku- und die USB-C-Spannung parallel angeschlossen werden können, beispielsweise während der Programmierung.

Bei der Inbetriebnahme wurden die folgenden Fehler festgestellt:

- Falsch angeschlossener Common-Pin bei den Schaltern.
- Widerstand mit falschem Widerstandswert bestückt.
- FTDI-Chip verdreht verlötet.

Diese Fehler führten zu Kurzschlüssen auf der Platine, wodurch die Spannungsversorgung einbrach.

3.2 VORGEHEN

Die erste Inbetriebnahme erfolgte über ein Labornetzteil in Kombination mit dem vorhandenen USB-C-Anschluss. Dabei wurde der Strom auf 50 mA begrenzt. Diese Begrenzung ist ausreichend, um die Spannungsniveaus zu überprüfen und mögliche Kurzschlüsse zu identifizieren.

Da bei den ersten Tests kein stabiles Spannungsniveau erreicht wurde, wurde das Problem schrittweise eingegrenzt. Dazu wurden folgende Maßnahmen durchgeführt:

1. Überprüfung aller Widerstände und Kondensatoren.
2. Entfernen des ESP8266-Chips.
3. Entfernen des FTDI-Chips.

Bei der Überprüfung der Widerstände wurde ein falscher Widerstand gefunden und ersetzt. Der verwendete Widerstand wies 10 Ω auf, obwohl 10 kΩ benötigt wurden. Der viel zu kleine Widerstand führte zu einem Kurzschluss, der behoben wurde. Dabei handelte es sich um den Widerstand R31, wie in Abbildung 3.1 dargestellt.

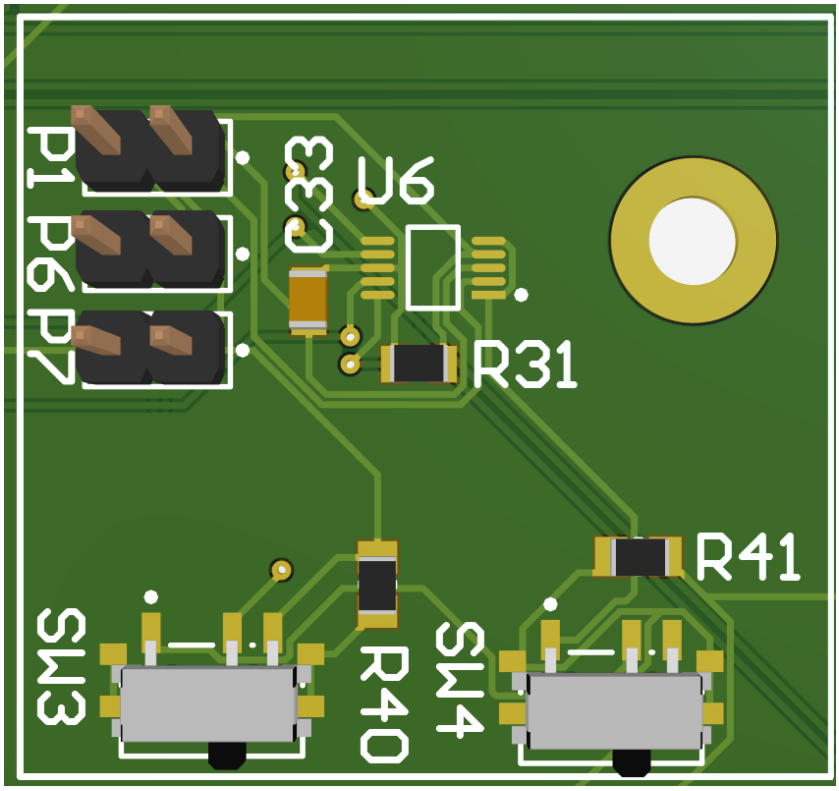


Abbildung 3.1: Falscher Widerstand

In weiteren Schritten wurden der ESP8266 und ein neuer FTDI-Chip schrittweise wieder eingesetzt. Die weitere Inbetriebnahme der Spannungsversorgung erfolgte gemäß den wieder platzierten Komponenten anschließend gestuft. Zunächst wurde die Platine ausschließlich über die 5 V USB-C-Versorgung betrieben. Dabei wurden alle relevanten Spannungen mit dem Multimeter überprüft, insbesondere die korrekte Erzeugung der 3,3 V-Versorgung. Zusätzlich wurde auf Kurzschlüsse und erhöhte Stromaufnahme kontrolliert. Bei der weiteren Kontrolle brach die Spannung erneut ein, insbesondere beim Betätigen der Schalter 3.2 zur Signalumschaltung.

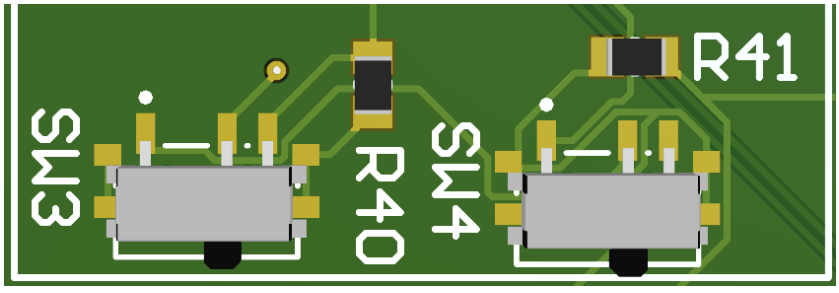


Abbildung 3.2: Signalumschaltung Schalter

Dies ist darauf zurückzuführen, dass der Common-Pin nicht, wie im Schaltplansymbol 3.3 angenommen, auf dem ersten Pin liegt, sondern, wie dem Datenblatt [4] zu entnehmen ist, auf dem zweiten Pin liegt.

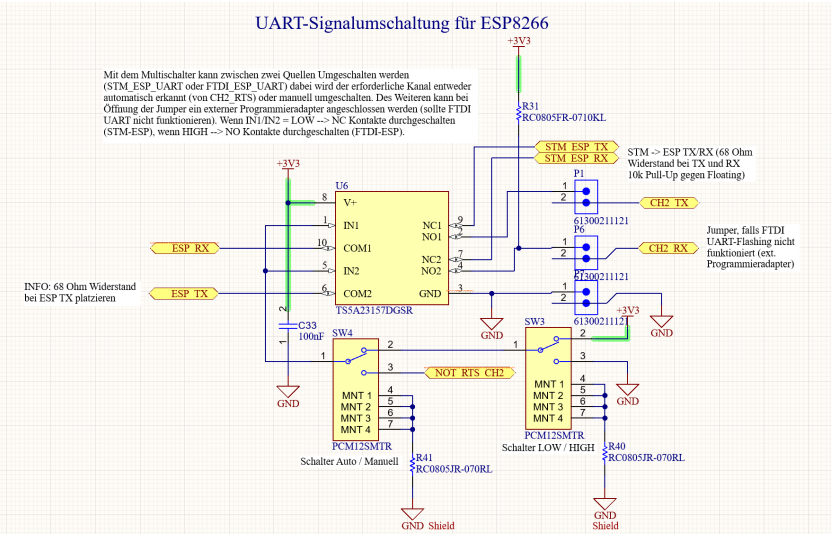


Abbildung 3.3: Schaltplan switshes

Dies führte zu einem Kurzschluss bei der Umschaltung des Schalters, da das Spannungsniveau direkt von +3,3 V auf Ground gezogen wurde. Dies konnte für die weitere Inbetriebnahme zunächst ignoriert werden, da die erste Position des Schalters die Kommunikation zwischen dem Mikrocontroller und dem FTDI-Chip herstellt, was für die Inbetriebnahme des Mikrocontrollers ausreichte. Bei der Umschaltung des Schalters wird die Verbindung zum ESP8266 hergestellt, um die benötigte Firmware zu flashen.

Dank des schrittweisen und strukturierten Vorgehens konnte jede Spannungsebene unabhängig geprüft und validiert werden. Dadurch konnten Fehler in der Spannungsversorgung erkannt werden, ohne dass andere Baugruppen gefährdet wurden.

Nach Abschluss der beschriebenen Maßnahmen stand eine stabile Spannungsversorgung zur Verfügung. Anschließend konnte die Platine zuverlässig betrieben werden und war für weitere Funktionstests bereit.

4 Test der Kommunikationsschnittstelle

Nach dem erfolgreichen Aufbau der Platine wird zunächst die USB-C-Seriell-Kommunikation des FT2232H mit dem Mikrocontroller geprüft. Das Ziel besteht darin, sicherzustellen, dass beide Kanäle korrekt erkannt werden, dass Daten zuverlässig gesendet und empfangen werden können und dass das angeschlossene Zielsystem antwortet.

Dabei wurde festgestellt, dass der Reset-Pin des Mikrocontrollers einen Floating-Zustand aufweist. Dies ist auf den über den SN74LVC14AD angesteuerten Open-Drain-MOSFET (Q2) zurückzuführen, der den Reset-Pin bei undefiniertem Eingangszustand nicht eindeutig auf High oder Low zieht. Nach dem Entfernen von Q2 4.1 war der Reset-Pin wieder klar über den Pull-Up definiert und stabil.

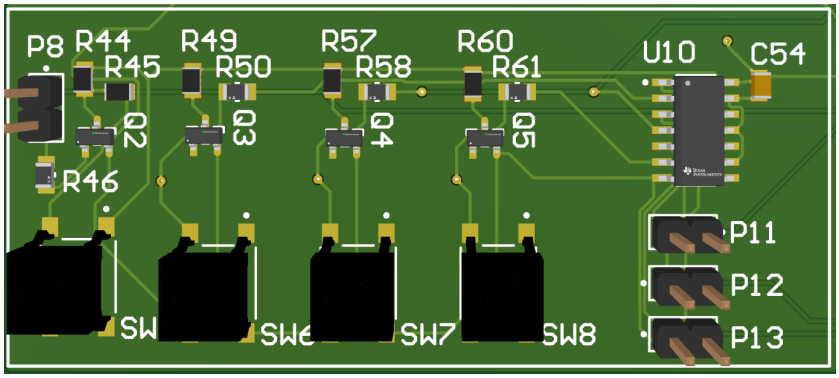


Abbildung 4.1: Q2

Dadurch war der Reset-Pin des Mikrocontrollers auf einem definierten High-Level, sodass ein normaler Start bei angeschlossener Stromversorgung möglich ist. Zunächst wurden die Ports des FTDI-Chips nicht vom Computer erkannt. Abhilfe schafft hier das manuelle Reset des Chips, das später im Kapitel „Bedienung" 6 beschrieben wird.

Hinweis: MCU manuell zurücksetzen, nachdem die USB-Verbindung hergestellt wurde.

4.1 BENÖTIGTE SOFTWARE

Für die Tests werden folgende Programme verwendet und benötigt:

- Treiber des FTDI-Chips.
- Python und das Paket „pyserial" dienen der einfachen Überprüfung der Kommunikation.

4.2 GRUNDLEGENDER KOMMUNIKATIONSTEST ÜBER EIN TERMINAL

1. STM32 Cube Programmer öffnen.
2. Passenden FT2232H-COM-Port auswählen (A oder B).

3. Verbindung herstellen.
4. Zeichen senden und prüfen, ob eine Antwort bzw. ein Echo erscheint.

Ein sichtbares Echo bzw. eine Antwort bestätigte die grundlegende Funktionsfähigkeit der USB-Verbindung und des FT2232H. Dabei wird die aufgespielte Version des Bootloaders des Mikrocontrollers zurückgegeben. Dies ist auch das Zeichen, dass die Kommunikation mit dem Mikrocontroller funktioniert und er bereit ist, um Software zu flashen.

4.3 SCHNELLSTEST MIT PYTHON

Nach Installation des dazugehörigen Packetes kann ein systematischer Funktionstest mit folgendem Python-Skript durchgeführt werden:

```
pip install pyserial

import serial

ser = serial.Serial("COM20", 115200, timeout=1)

while True:
    print(ser.readline().decode(errors="ignore"))
```

Es erscheinen periodisch Antworten bzw. Echos, was bedeutet, dass der Mikrocontroller arbeitet und die Schnittstelle zuverlässig funktioniert.

4.4 VISUELLE BESTÄTIGUNG DER KOMMUNIKATION

Sobald die RX- und TX-Leitungen aktiv Signale senden oder empfangen, leuchten entsprechend dem Schaltplan [5] die LEDs. Wie in Abbildung 4.2 dargestellt, leuchten die beiden LEDs DS1 (grün) und D2 (rot) bei einer aktiven Verbindung.

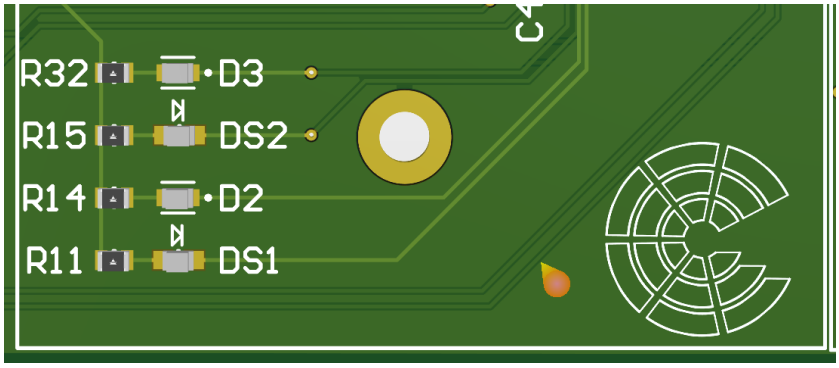


Abbildung 4.2: LED leuchten

Die Kommunikation wurde bis zu diesem Stand in Betrieb genommen und überprüft. In den zukünftigen Schritten müssen alle weiteren Kommunikationswege überprüft werden.

5 Anschlussplan

Um die Hardware zu vervollständigen, muss die gesamte Verkabelung des Demonstrators noch installiert werden. Dazu gehören im Wesentlichen die folgenden Komponenten:

- Akku [6]
- DCDC-Wandler [7]
- Motor Driver [8]
- PCB [5]
- Motoren [9]
- Aufnahme Akku [10]

5.1 VERBINDUNG AKKU

Zunächst werden die Verbindungen zwischen Halterung, DCDC-Wandler und PCB hergestellt. Am Ende der gesamten Verkabelung wird der Akku in die dafür vorgesehene Halterung eingesteckt. Die Verkabelung soll dazu wie folgt ausgeführt werden:

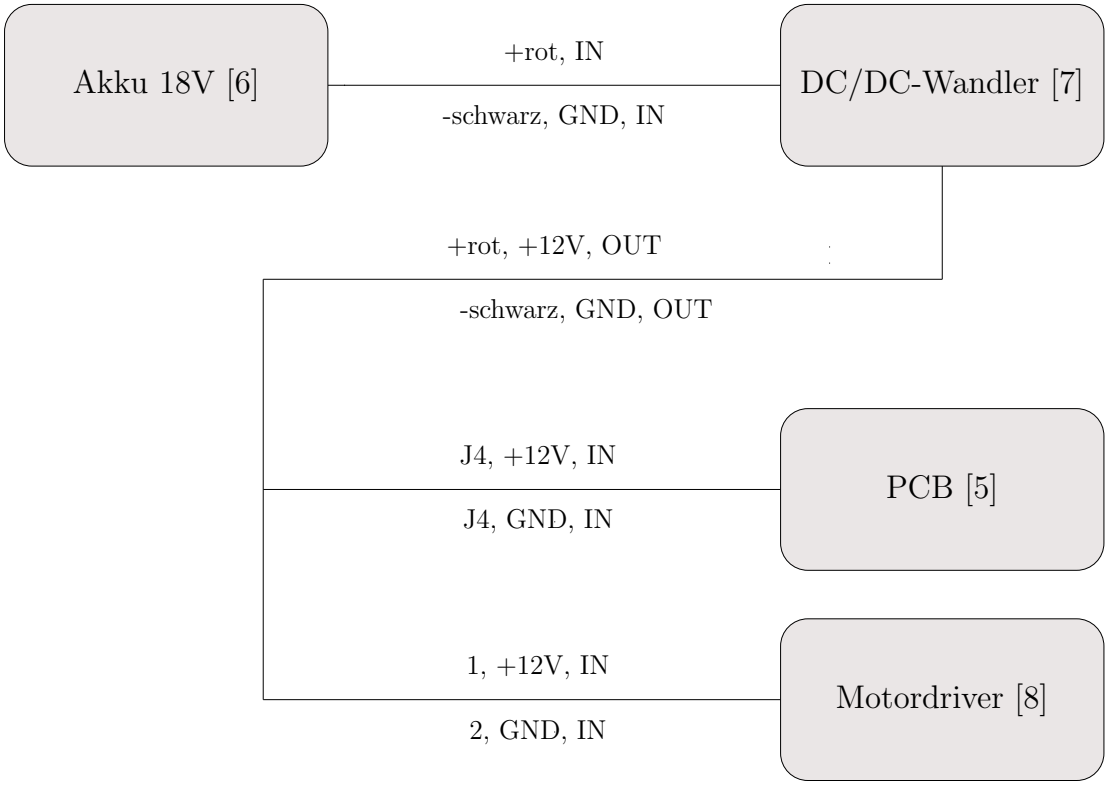


Abbildung 5.1: Akku zu PCB

Nach der Verkabelung kann der Akku eingesteckt werden, alternativ ist es ebenso möglich die Versorung über ein Labor Netzteil bereitzustellen und die Stromaufnahme zu begrenzen. Anschließend sind die folgenden Spannungswerte zu überprüfen:

- 12 V Ausgang am DCDC Wandler
- 3,3 V PCB Versorgung
- 5 V Ausgänge PCB

Der DCDC-Wandler [7] muss auf die entsprechenden 12 V eingestellt werden. Dies ist zu überprüfen. Anschließend den Akku für die weitere Verkabelung wieder entfernen.

5.2 MOTOREN

Für den Anschluss der Motoren wird die Abbildung 5.2 herangezogen. Diese ist sowohl für den ersten als auch für den zweiten Motor mit den entsprechenden Pins durchzuführen. Zusätzlich ist sicherzustellen, dass die Zuordnung der Steuersignale (IN, EN) zum jeweiligen Motorkanal korrekt erfolgt, damit die Ansteuerung eindeutig dem gewünschten Motor zugeordnet ist. Vor der Inbetriebnahme sollten alle Verbindung überprüft werden, um Fehlfunktionen oder Schäden an den Komponenten zu vermeiden.

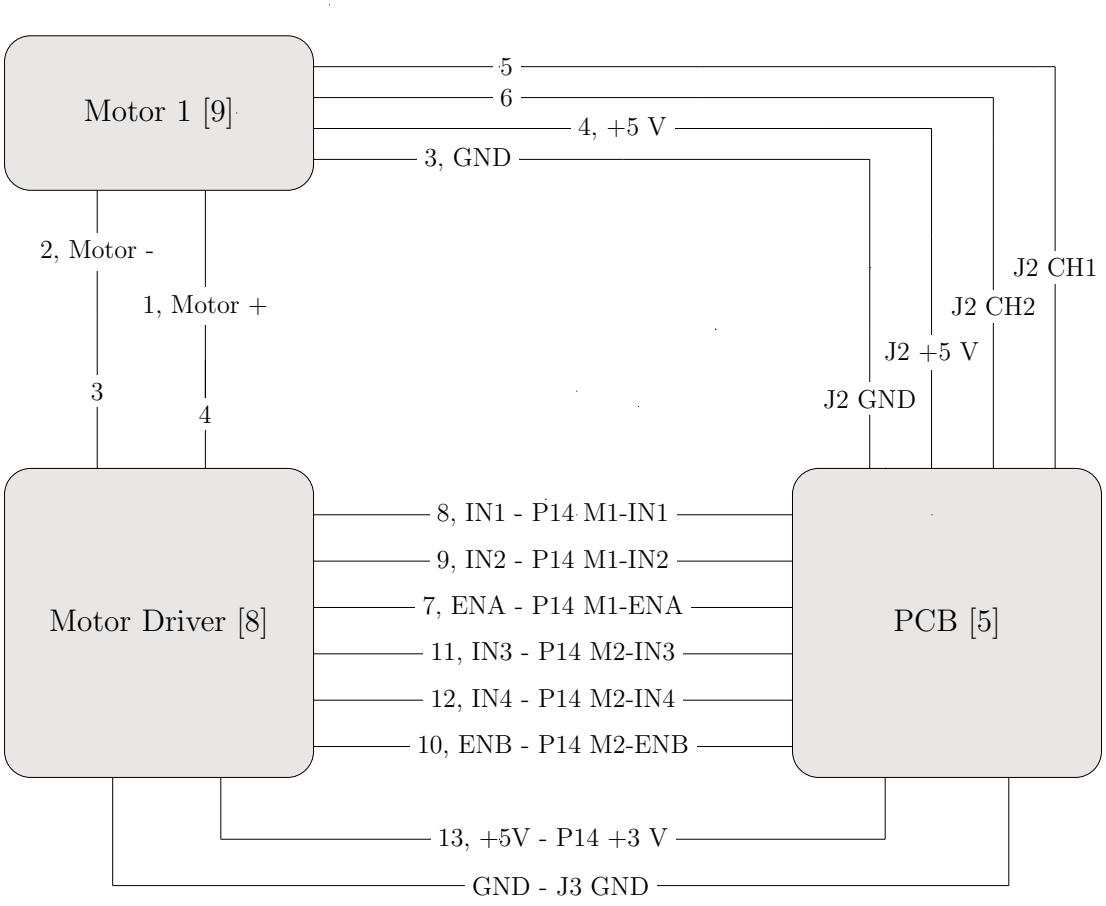


Abbildung 5.2: Motoren

5.2.1 Wichtige Hinweise zur Verdrahtung

Bei der Verdrahtung ist darauf zu achten, dass alle Komponenten eine gemeinsame Masse (GND) besitzen, da sonst die Steuersignale zwischen Platine und Motortreiber nicht korrekt erkannt werden. Die Versorgung des Motortreibers erfolgt über die 12 V-Schiene, während die Logik-Referenz des Treibers mit der 3,3 V-Logikspannung der Platine verbunden werden muss. Die Encoder dürfen ausschließlich mit 5 V versorgt werden und niemals mit 12 V. Die Motorleitungen sind von den Encoder- und Signalleitungen möglichst getrennt zu führen, um Störungen zu vermeiden. Dreht ein Motor in die falsche Richtung, können einfach die beiden Motoranschlüsse am Treiber vertauscht werden.

Die benötigten Anschlüsse sind in der folgenden Abbildung 5.3 markiert. Die genaue Verbindung ist den jeweiligen Diagrammen für die Spannungsversorgung 5.1 sowie der Abbildung für die Motoren 5.2 zu entnehmen. Alle vorgeschlagenen Verkabelungen sind den Datenblättern 5 zu entnehmen und zu überprüfen.

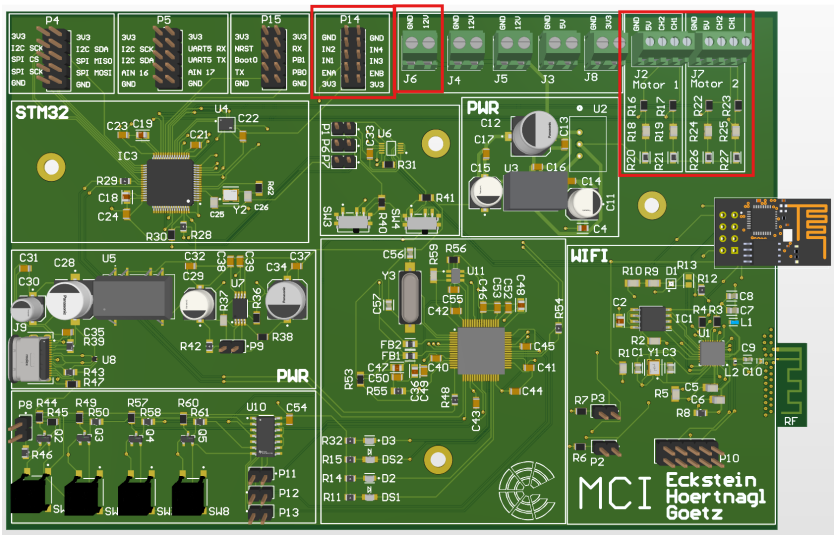


Abbildung 5.3: PCB Motoren

6 Bedienung

Die Platine kann sowohl über Software als auch über mehrere Hardware-Taster bedient werden. Über diese können die Reset- und Boot-Modi der integrierten Mikrocontroller ausgewählt werden. Auf der Platine befinden sich ein ESP8266 und ein STM32, die jeweils unabhängig voneinander gesteuert und programmiert werden können.

6.1 SOFTWARE

Je nach Mikrocontroller wird die Software über unterschiedliche Schnittstellen aufgespielt. Der STM32 kann entweder über den internen Bootloader (UART) oder über einen Debugger programmiert werden. Dazu wurden im initialen Zusammenbau die benötigten Pins für die SWM hinausgeführt. Der ESP8266 wird typischerweise über die serielle UART-Schnittstelle geflasht. Die Auswahl des jeweiligen Flash-Modus erfolgt hardwareseitig über Taster und UART-Umschalter oder später über die Software.

6.2 MANUELLE BEDIENUNG

Zur Hardware-Bedienung stehen mehrere Taster zur Verfügung, mit denen sich Reset- und Boot-Zustände der Mikrocontroller einstellen lassen. Die folgenden Abschnitte beschreiben die Funktion dieser Taster und deren Auswirkungen auf den jeweiligen Modus.

6.2.1 ESP8266 Taster SW5 (GPIO0) & SW6 (EN/Reset)

Der ESP8266 nutzt den Zustand des Pins GPIO0 in Kombination mit Enable, um zwischen Normalbetrieb und Bootloader-Modus zu unterscheiden.

Tabelle 6.1: Boot- und Reset-Modi des ESP8266

SW5 (GPIO0)	**SW6 (EN)**	**Ergebnis**	**Beschreibung**
nicht drücken	nicht drücken	Normaler Start	ESP8266 bootet Firmware aus Flash-Speicher.
nicht drücken	kurz drücken	Neustart	Reset des ESP8266, dann normaler Boot aus Flash.
halten	kurz drücken	Flash-Modus	GPIO0 liegt auf Low während des Resets, der ESP wechselt in den UART-Bootloader und kann programmiert werden.
halten	nicht drücken	Keine Änderung	-

6.2.2 STM32 Taster SW7 (BOOT0) & SW8 (NRST)

Der STM nutzt den Zustand des Pins BOOT0 in Kombination mit einem Reset, um zwischen Normalbetrieb und Bootloader-Modus zu unterscheiden.

Tabelle 6.2: Boot- und Reset-Modi des STM32

SW7 (BOOT0)	**SW8 (NRST)**	**Ergebnis**	**Beschreibung**
nicht drücken	nicht drücken	Normaler Start	Der STM32 startet das im Flash gespeicherte Programm.
nicht drücken	kurz drücken	Normaler Reset	Neustart des Mikrocontrollers, Start aus dem Flash.
halten	kurz drücken	Bootloader-Modus	BOOT0 ist High während des Resets, der STM32 startet den Bootloader.
halten	nicht drücken	Keine Änderung	-

6.2.3 Standardzustand ohne Tastendruck

Wenn beim Einschalten oder während des Betriebs keine Tasten betätigt werden, befindet sich die Platine im Normalzustand. Beide Mikrocontroller starten die jeweils installierte Firmware aus dem internen Flash-Speicher.

Tabelle 6.3: Systemzustand ohne Benutzereingriff

Bauteil	**Zustand**	**Ergebnis**
ESP8266	GPIO0 High, EN High	Normaler Boot aus Flash
STM32	BOOT0 Low, NRST High	Normaler Boot aus Flash
UART-SW3/SW4	Mechanisch	entsprechend Schalterposition

6.2.4 UART-Umschalter (SW3 & SW4)

Die UART-Umschalter SW3 und SW4 legen die Verschaltung des UART des ESP fest. Unabhängig davon ist der STM32 fest über den USB-C-Anschluss mit dem FTDI-Interface verbunden.

Tabelle 6.4: UART-Schaltkonfigurationen (ESP-UART)

SW3	**SW4**	**UART-Verbindung**	**Verwendung**
Auto	Low	STM32 ↔ ESP	Standardbetrieb: Kommunikation zwischen STM32 und ESP.
Auto	High	FTDI/USB ↔ ESP	Flashen des ESP über USB.
Low	Low	erzw. STM32 ↔ ESP	Manuell festgelegte Verbindung zwischen STM32 und ESP.
High	High	erzw. FTDI ↔ ESP	Manuelle Verbindung zwischen FTDI und ESP.

Hinweis: Der STM32 ist dauerhaft über USB-C mit dem FTDI verbunden und kann unabhängig von SW3 und SW4 programmiert und debuggt werden.

7 Jumper

7.1 UART VERBINDUNG ESP

Die Jumper P1 und P6 dienen der Umschaltung der UART-Kommunikationsleitungen des ESP. P1 ist der TX-Jumper und beeinflusst die Sendeleitung, während P6 die RX-Leitung steuert. Gemeinsam ermöglichen sie den Betrieb des ESP entweder über die integrierte USB-UART-Schnittstelle oder über einen externen UART-Programmieradapter.

Tabelle 7.1: Jumper P1 und P6 – UART-Signalumschaltung des ESP

Jumper	Signal	Funktion und Wirkung
P1	ESP_TX	P1 schaltet die TX-Leitung des ESP zwischen der internen USB-UART-Schnittstelle (FTDI) und einer externen Signalquelle um.
P6	ESP_RX	P6 schaltet die RX-Leitung des ESP zwischen der internen USB-UART-Schnittstelle (FTDI) und einer alternativen bzw. externen Signalquelle um.

Hinweis: Im Normalbetrieb muss der Jumper geschlossen sein.

7.2 BOOTLOADER-AKTIVIERUNG: AUTOMATISCHE UND MANUELLE OPTIONEN

Für den Automatikbetrieb müssen die Jumper P11, P12 und P13 über die Software geschlossen sein. Der Jumper P8 wird direkt über den FTDI-Chip gesteuert und bedarf keiner Invertierung. Er muss für den Automatikbetrieb ebenfalls geschlossen sein.

Tabelle 7.2: Funktion und Zustand der Jumper P8, P11–P13

Jumper	Signal	geschlossen	geöffnet
P8	BOOT0 STM	Boot-Pegel kann über die Software gesteuert werden.	Start in nomral Betrieb, Boot Pin wird manuel angesteuert über den Taster SW5
P11	NRST STM	Enable- bzw. Reset-Signal wird über die Software gesteuert. STM32 werden zuverlässig freigegeben bzw. zurückgesetzt.	Start in nomral Betrieb, Reset Pin wird manuel gesteuert über den Taster SW6
P12	GPIO0 ESP	Boot-Pegel wird über die Software gesteuert. ESP: Flash-Modus möglich (GPIO0 = Low beim Reset).	Start in nomral Betrieb, Pin wird manuel gesteuert über den Taster SW7
P13	EN ESP	Der Reset Pin kann über die Software gesteuert werden.	Rese Pin wird manuel gesteuert über den Taster SW8

Die Jumper P8 sowie P11 bis P13 bilden gemeinsam die zentrale Reset- und Boot-Steuerlogik der Platine. Über sie wird festgelegt, ob die Reset- und Boot-Signale für ESP und STM32 automatisch

durch die Onboard-Logik erzeugt werden oder ob sie davon getrennt sind und manuell betätigt werden müssen.

Hinweis: Im geschlossenen Zustand ermöglichen die Jumper ein automatisches Flashen und Zurücksetzen der Mikrocontroller.

7.3 ONBOARD-LOGIK FÜR RESET UND BOOT (ESP & STM32)

Die Platine verfügt über eine Onboard-Logik 7.1 zur automatischen Steuerung der Reset- und Boot-Signale von ESP und STM32. Das Ziel besteht darin, das Flashen über USB ohne manuelle Tastersequenzen zu ermöglichen und reproduzierbare Startzustände sicherzustellen.

Die Logik wertet die UART-Steuersignale der USB-UART-Schnittstelle (DTR/RTS) aus. Da diese Signale in ihrer Polarität und elektrischen Ausprägung nicht direkt für die Mikrocontroller geeignet sind, werden sie durch einen Logikbaustein invertiert, entkoppelt und zeitlich korrekt aufbereitet.

Die Ansteuerung der Steuersignale erfolgt softwareseitig. Die dafür notwendige Software zur gezielten Steuerung von DTR und RTS muss jedoch noch implementiert werden. Anschließend übernimmt die Onboard-Logik die Umsetzung in gültige Reset- und Boot-Signale für die Mikrocontroller.

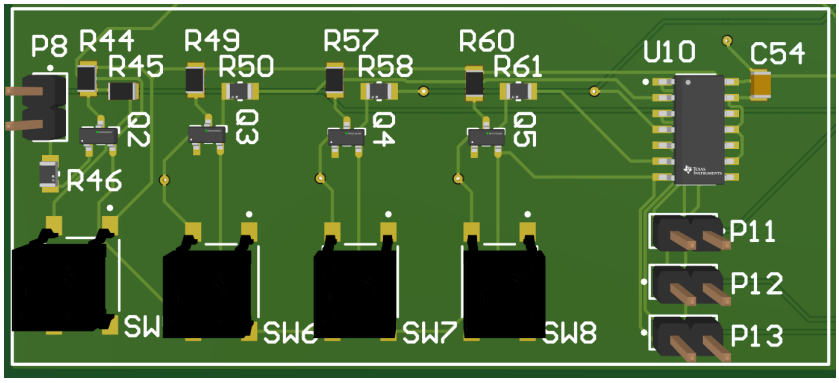


Abbildung 7.1: Onboard logik

Hinweis: Der FTDI-Chip wird von der Software gesteuert, die Onboard-Logik bereitet die Signale auf und die Mikrocontroller wechseln den Betriebsmodus.

8 Konstruktion und Montage

8.1 GEHÄUSE

Zur Vereinfachung des Projekts wurde die zweite Montageposition des Akkus entfernt. Durch die Verwendung der Aluminiumprofile mit den Abmessungen 20 x 20 mm kann diese jedoch jederzeit wieder ergänzt werden.

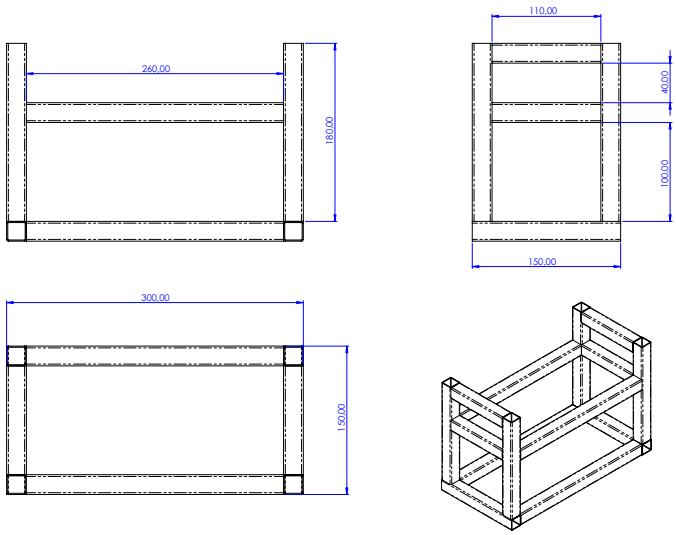


Abbildung 8.1: Frame

Die einzelnen zugeschnittenen Profile wurden mit dem systemeigenen Verbindungssystem miteinander verbunden. Dadurch wurden die Verbindungsschrauben sauber und versteckt. Zusätzlich lässt das Aluminiumprofil noch Platz für weitere Verbesserungen sowie eine saubere Möglichkeit zur Kabelverlegung entlang der Profile.

8.2 HALTERUNG MOTOREN

Die Halterung der Motoren wurde mittels 3D-Druckverfahren hergestellt. Die additive Fertigung ermöglichte eine zielgerichtete Konstruktion, die perfekt auf die mechanischen Bedürfnisse abgestimmt ist.

Die Halterung umschließt die Motoren und berücksichtigt dabei den geometrischen Zapfen der Motorwelle, der zugleich als Verdrehsicherung wirkt. Die Umfassung wurde geschlitzt ausgeführt, um den Motor mit einer M4x20-Zylinderkopfschraube zu klemmen.

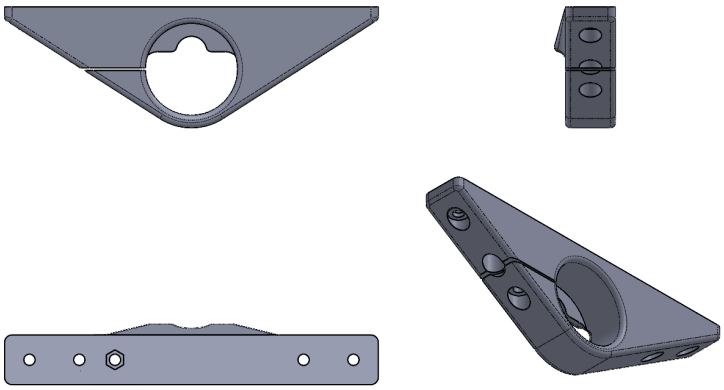


Abbildung 8.2: Motor halterungen

Die Halterung selbst wird mit vier M4-Schrauben mit dem Rahmen verbunden. Dazu werden die systemspezifischen Nutensteine des Profilherstellers als Muttern verwendet. Das garantiert eine belastbare Verbindung im Falle eines Aufpralls am Boden des Demonstrators.

8.3 INSTALLATIONSEBENE ELEKTRONIK

Die Halterung der elektronischen Bauteile wurde mittels 3D-Druckverfahren hergestellt. Dadurch war eine individuelle Positionierung der Bauteile sehr einfach möglich. Die Unterseite der Ebene wurde mit Rippen verstärkt.

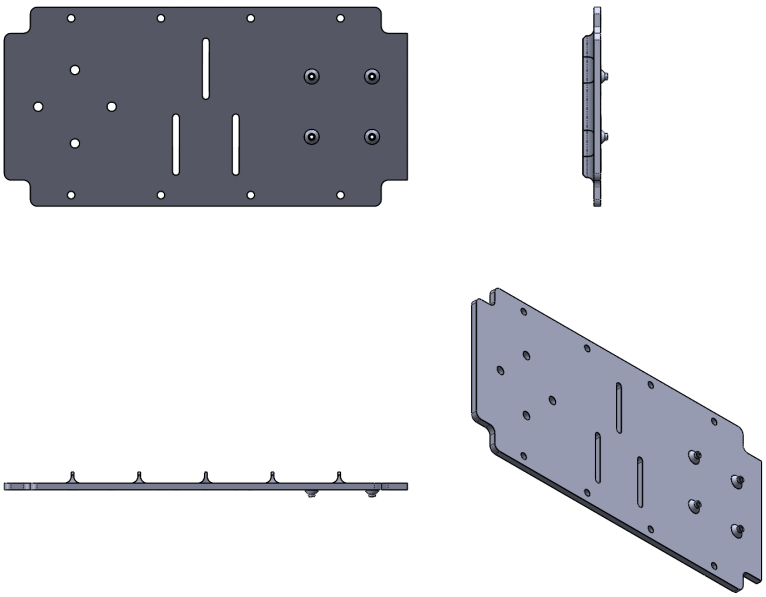


Abbildung 8.3: Ebene Elektronik

Die Aufnahme des Akkus wurde mit Langlöchern versehen. Dies dient dazu, den Massenschwerpunkt des Akkus anzupassen und ihn mittig zum Rahmen auszurichten. Durch diese variable Positionierung können Ungleichgewichte in der gesamten Konstruktion ausgeglichen werden, indem die Position des Akkus korrigiert wird.

Die vier Durchgangsbohrungen werden für die Befestigung des DCDC-Wandlers verwendet. Für die Montage werden zwei Bohrungen benötigt. Die nicht genutzten Befestigungsbohrungen können für eine alternative Position des Wandlers verwendet werden. Somit kann der Wandler bei Bedarf gedreht werden, beispielsweise wenn die Verkabelung dies erfordert.

8.4 INSTALLATIONSEBENE PLATINE

Die Halterung wurde ähnlich wie in der elektrischen Installationsebene ausgeführt. Dabei spiegelten die vier Dome die Anschraubpunkte des PCB wider. Durch das Entfernen der alternativen Akku-Position ist somit ebenfalls ein einfacher Zugriff auf das PCB gegeben.

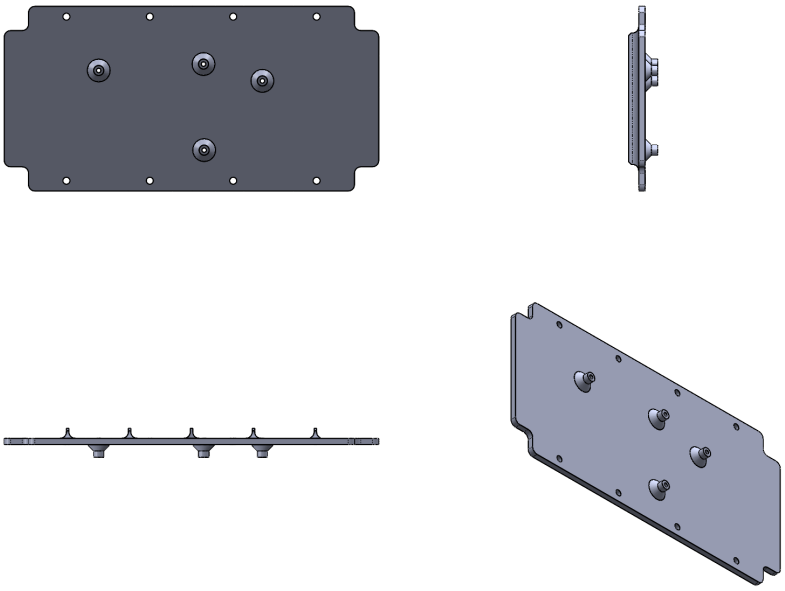


Abbildung 8.4: Ebene Platine

Auch die Unterseite der Halterung wurde mit Verstärkungsrippen ausgeführt. Die Materialstärken und das verwendete Material (PETG) sind für die auftretenden mechanischen Belastungen ausreichend.

8.5 REIFEN

Die Reifen wurden, ebenso wie die Installationsebenen, mittels additiver Fertigung hergestellt. Die Aufnahme der Reifen wurde passend zur Motorwelle designt. Die Toleranzen wurden so gewählt, dass Reifen und Motorwelle über eine Presspassung miteinander verbunden sind.

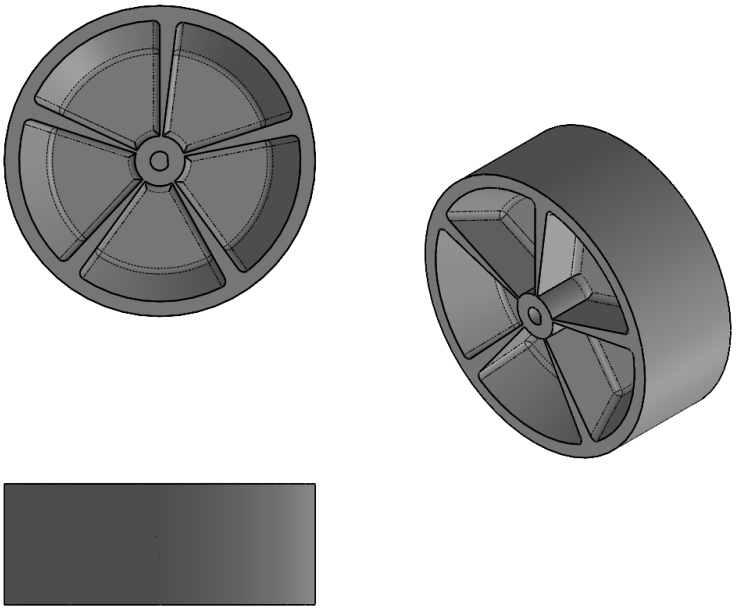


Abbildung 8.5: Reifen

Sowohl die mechanische Verbindung zwischen Reifen und Motorwelle als auch die mechanische Belastung sind für den Betrieb ausreichend. Für die Reifen wurde das Material PLA gewählt, da es über ausreichende mechanische Belastbarkeitsgrenzen für diesen Anwendungsfall verfügt.

9 Weiteres Vorgehen und offene Punkte

Im Rahmen der bisherigen Arbeiten konnte ein funktionsfähiger Demonstrator aufgebaut werden, der jedoch noch über keine Software verfügt. Zudem wurden noch nicht alle Funktionen genau getestet. Für eine stabile Nutzung und Weiterentwicklung des Systems bestehen jedoch sowohl auf mechanischer als auch auf elektronischer Ebene weitere Optimierungsmöglichkeiten.

9.1 HARDWARE-VERBESSERUNGEN

Die mechanische Ausführung des Aufbaus ist zwar grundsätzlich funktionsfähig, bietet jedoch Potenzial zur Verbesserung in Bezug auf Stabilität, Wartbarkeit und Kabelführung.

1. Umsetzung der Verkabelung.
2. Zur Verbesserung der Haftung und Reduktion von Schlupf wurden Gummielemente an den Rädern ergänzt.
3. Das Gestell kann um einen zweiten Stock erweitert werden, um Elektronik und Mechanik räumlich besser zu trennen und eine alternative Akku-Position zu schaffen (optional).
4. Es werden Dämpfungselemente integriert, um ein Umfallen des Systems bei instabilen Regelzuständen zu verhindern bzw. zu dämpfen.
5. Kabelhalterungen für die ITEM-Profile werden nachgerüstet, um eine strukturierte Kabelführung zu ermöglichen.

9.2 PCB-VERBESSERUNGEN

Auch beim Leiterplattendesign gibt es mehrere Ansatzpunkte, um die Inbetriebnahme zu vereinfachen und die Robustheit zu erhöhen.

1. Umstellung auf ein fertiges Modul für Wifi, wie zum Beispiel der ESP8684-MINI-1-H4 [11], um eine modernere, kompaktere und besser unterstützte WLAN-Plattform zu verwenden.
2. Entfernen der bestehenden Signalumschaltung für den ESP und Ersatz durch einen dedizierten Programmieradapter mit nach außen geführten Pins.
3. Ergänzung von Testpunkten für relevante Signale (UART, Reset, Versorgung), um Debugging und Fehlersuche zu erleichtern.
4. Den fehlenden Pin für die SWD verbindung des ST-Link hinausführen und auf einen eigenen header legen.
5. Vereinfachung der Spannungswandlungen sind möglich wie der Pololu 3.3V, 1A Step-Down Voltage Regulator D24V10F3 [12]
6. Integration von Lötbrücken, Jumpern, 0 Ω Widerstände um die Platine abschnittsweise in Betrieb zu nehmen.
7. Spannungsteiler von den Encoders zum STM32 korrigieren. Die 3.2V liegen nicht am STM an, sondern nur 1.8. Hier müssen andere Widerstände eingelötet werden

Die genannten Punkte stellen sinnvolle nächste Schritte dar, um den Demonstrator sowohl für die Vorlesungen als auch für eine Weiterentwicklung durch nachfolgende Projektgruppen robuster, wartungsfreundlicher und besser erweiterbar zu gestalten.

Wie in der Auflistung beschrieben, kann der Abschnitt von 5 V auf 3,3 V durch den Einsatz von Modulen deutlich vereinfacht werden.

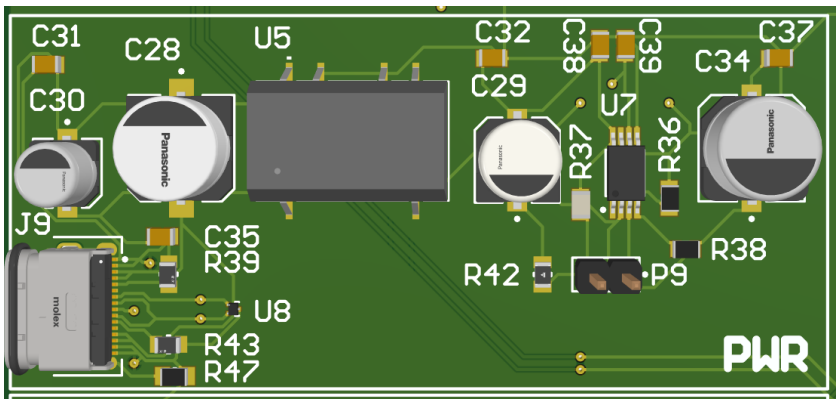


Abbildung 9.1: 5 V auf 3,3 V

Das gleiche Modul ist auch im Abschnitt 12 V auf 3,3 V einsetzbar und würde den Fokus auf die übrige Platine legen.

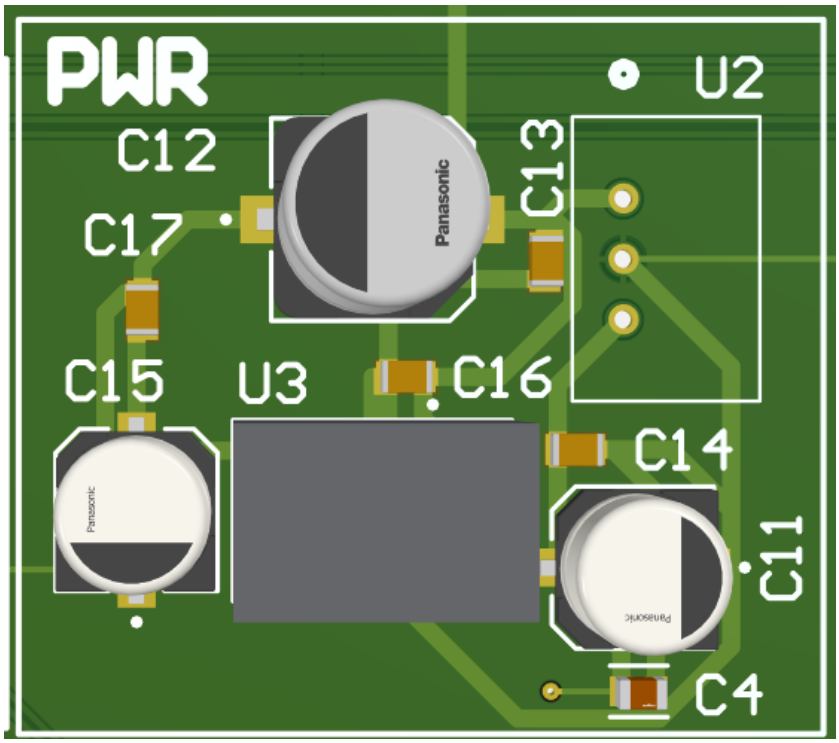


Abbildung 9.2: 12V auf 3,3 V

Die UART-Umschaltung zwischen den ESP-, STM- und FDTI-Chips kann entfernt werden, indem

die nötigen Pins für die externe Programmierung des ESP in einem eigenen Header herausgeführt werden. Das WLAN-Modul sollte in der Regel nur bei den Funktionstests programmiert werden und benötigt nach der richtigen Konfiguration der Software nicht mehr die Möglichkeit, die Kommunikationsverbindung schalten zu können.

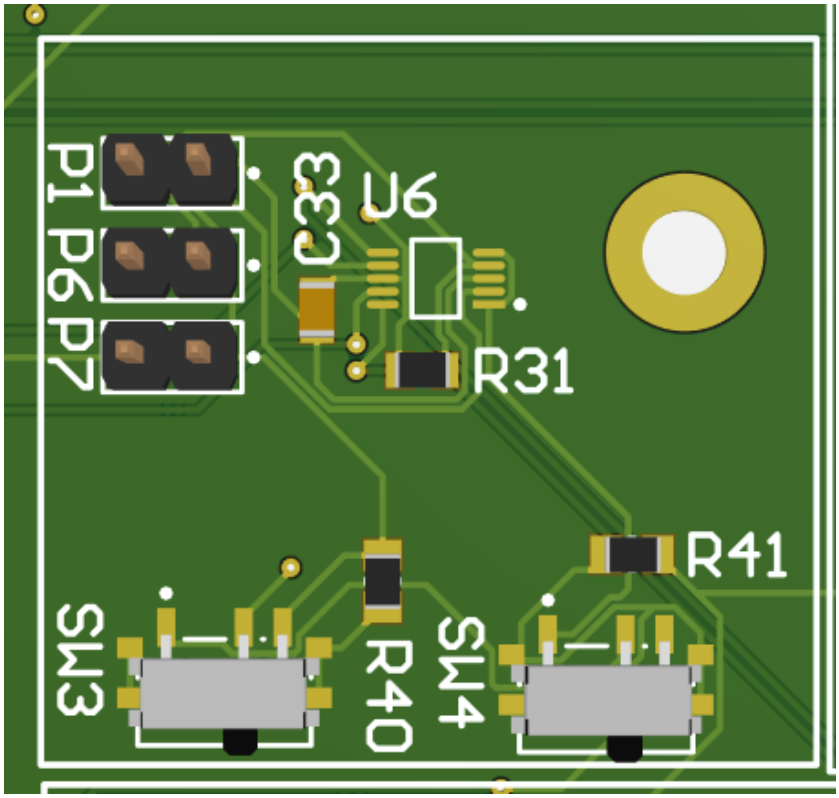


Abbildung 9.3: UART Umschaltung

Das gesamte WLAN-Modul inklusive PCB-Antenne sollte deutlich vereinfacht werden. Dafür können fertige Module wie das ESP8684-MINI-1-H4 [11] verwendet werden. Dies wird vom Hersteller ebenso empfohlen und bietet zudem die Erweiterung um Bluetooth.

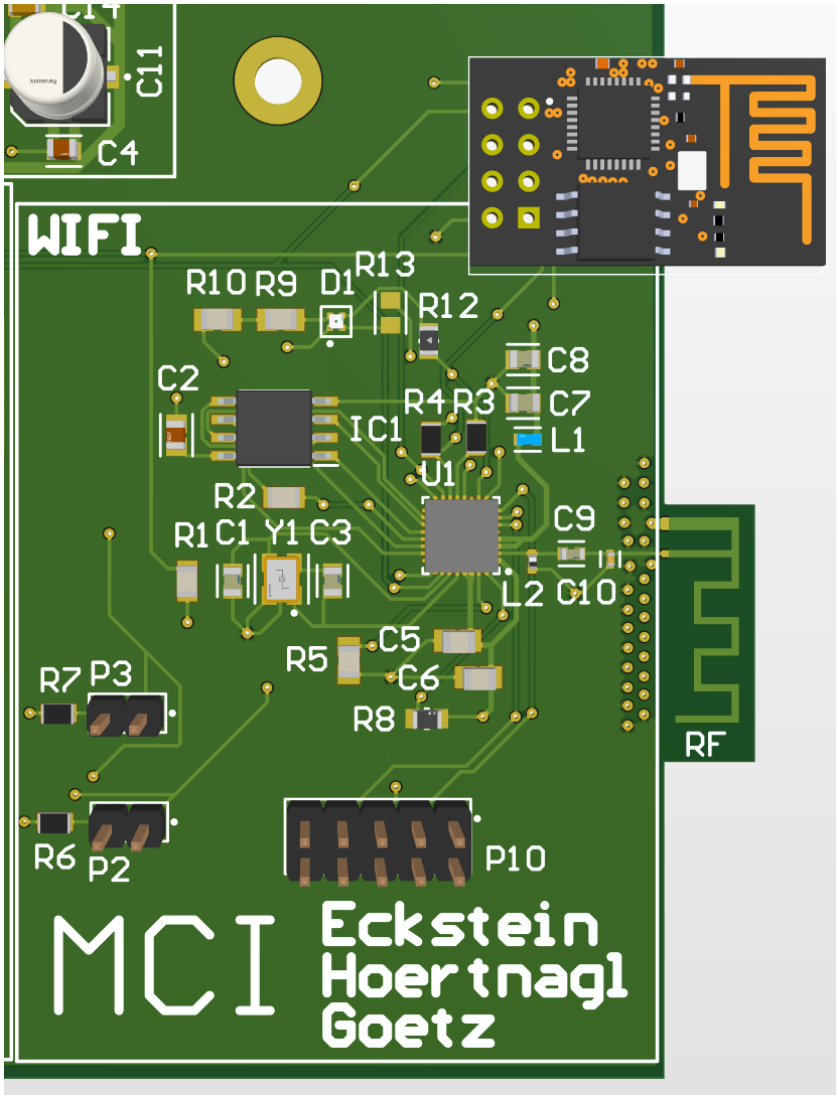


Abbildung 9.4: Wifi Abschnitt

9.3 WEITERES VORGEHEN BEI DER SOFTWARE

Unabhängig von den hardwaretechnischen Anpassungen ist die Entwicklung der Gesamtsoftware ein zentraler Schritt, um den Demonstrator vollständig einsatzfähig zu machen. Die Software bildet die Schnittstelle zwischen Regelungsalgorithmus, Sensorik und Aktorik und ist somit für den funktionalen Betrieb des Systems von entscheidender Bedeutung.

Zunächst muss eine stabile Firmware für die Mikrocontroller implementiert werden, die die grundlegenden Aufgaben wie die Initialisierung der Hardware, die Sensorabfrage und die Aktoransteuerung übernimmt. Darauf aufbauend ist eine Kommunikationsschnittstelle zu realisieren, über die der Demonstrator mit einem externen Rechner verbunden werden kann.

Ein wesentliches Ziel ist die Anbindung an MATLAB, um Regelungsparameter oder komplette Regelungsstrukturen vom PC aus zu übertragen und während des Betriebs anzupassen. Zusätzlich sollen Messdaten, wie beispielsweise Lage- oder Geschwindigkeitsinformationen, kontinuierlich an MATLAB zurückgesendet werden, um eine Live-Analyse und -Visualisierung zu ermöglichen.

Abschließend sind Tests und Validierungen erforderlich, um die Zuverlässigkeit der Software sowie die Interaktion von Hard- und Software unter realen Betriebsbedingungen zu gewährleisten.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] L. G. Hoertnagl Stefan, Maximilian Eckstein, "Elektronische produktentwicklung," Dokumentation, Abschlussbericht, Bericht Inverses Pendel, 2025, begleitdokumentation.

[2] DeepL SE. (2025) Deepl translator. DeepL SE. Online translation service, accessed on 3. Januar 2026. [Online]. Available: https://www.deepl.com/translator

[3] S. Mci, "Inverted pendulum – electronic product development," https://github.com/Stefmci/INV_Pendel_Elektronische_Produktentwicklung, 2024, gitHub repository.

[4] CK Switshes, *PCM Series Ultraminiature Surface Mount Slide Switches*, CK Switshes, 2022, zugriff: 27.12.2025. [Online]. Available: https://datasheet.octopart.com/PCM12SMTR-C%26K-Components-datasheet-175508494.pdf?src-supplier=Newark

[5] L. G. Hoertnagl Stefan, Maximilian Eckstein, *Schaltplan Inverses Pendel*, 2025, eigene Schaltungsentwicklung, interne Dokumentation.

[6] Makita, *BL1840B Li-Ion Akku 18V 4.0Ah*, Makita, 2025, lXT Lithium-Ionen Akku. [Online]. Available: https://www.makita.at/product/bl1840b.html

[7] Unbekannt, *DC-DC Step-Down Wandler 18V auf 5V/3.3V*, 2023, schaltregler zur Spannungsreduktion für Mikrocontroller und Peripherie. [Online]. Available: https://bauer-united.com/products/dc-dc-18v-36v-zu-12v-10a-spannungswandler

[8] DFRobot, *7A Dual DC Motor Driver*, DFRobot, 2016, sKU DRI0041. [Online]. Available: https://wiki.dfrobot.com/7A_Dual_DC_Motor_Driver_SKU__DRI0041

[9] ——, *Metal DC Geared Motor with Encoder 12V 122RPM*, DFRobot, 2024, gB37Y3530-12V-90EN. [Online]. Available: https://www.dfrobot.com/product-1210.html

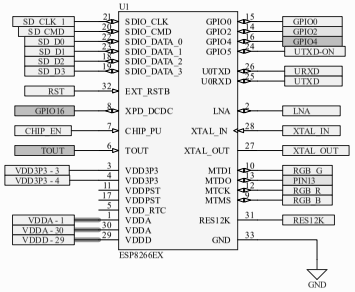
[10] QUPERR, *Akku-Adapter fuer 18V Makita LXT Akkus*, QUPERR, 2023, akku-Aufnahme mit Sicherung und Anschlusskabeln für Makita LXT Akkus. [Online]. Available: https://www.amazon.de/QUPERR-Sicherung-Akku-Adapter-Lithium-Akku-Power-Converter/dp/B0BPRYRGVF

[11] Digi-Key Electronics, "Esp8684-mini-1-h4 wlan/bluetooth modul," https://www.digikey.at/de/products/detail/espressif-systems/ESP8684-MINI-1-H4/21572303, 2025, zugriff am 29.12.2025.

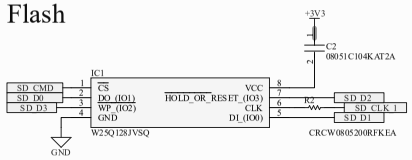
[12] Pololu Corporation, "3.3v, 500ma step-down spannungsregler d24v5f3," 2025, zugriff am 29.12.2025. [Online]. Available: https://www.pololu.com/product/2830

A Schaltpläne und Platine

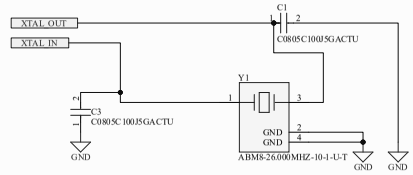
ESP8266



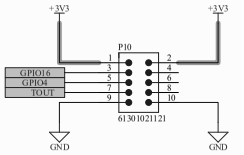
Flash



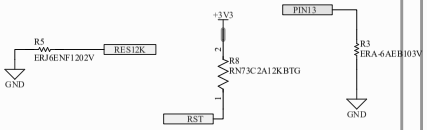
Oszi



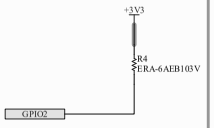
Breakout



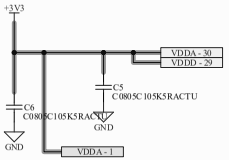
Kleinteile



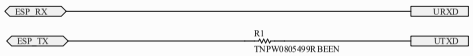
GPIO2 High bei Boot



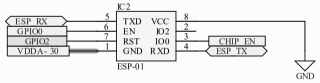
Power



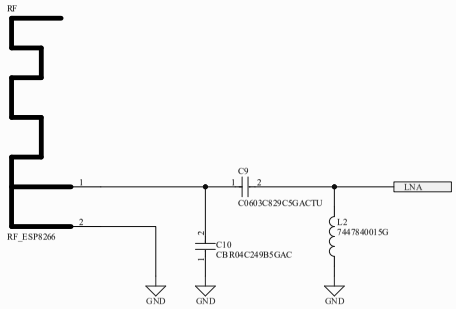
UART



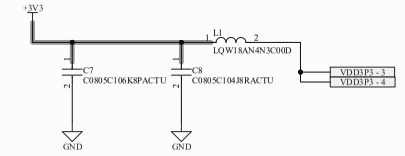
Plan B



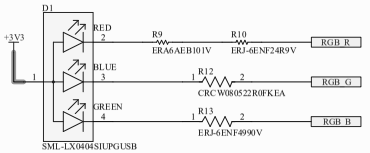
RF



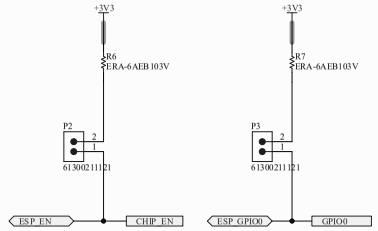
Power PA&LNA



LED



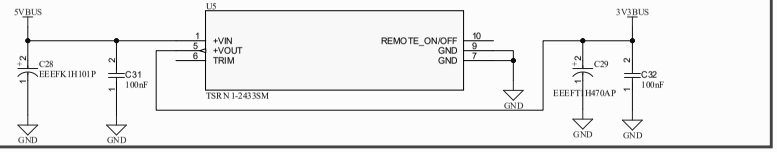
Jumper



Title		
Size: A2	Number	Revision
Date: 12.10.2025		Sheet of
File: C:\Users\...\ESP8266_final.SchDoc		Drawn By:

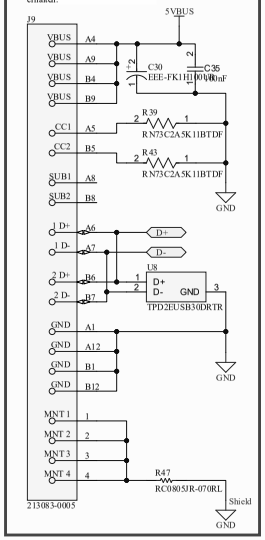
DC-DC Konverter 5V USB-Spannung auf 3V3

Es muss ein sehr leistungsstarker DC/DC Wandler verwendet werden, damit auch mit den 5V Eingangsspannung 1A am Ausgang ausgegeben werden kann.



USB-C Schnittstelle

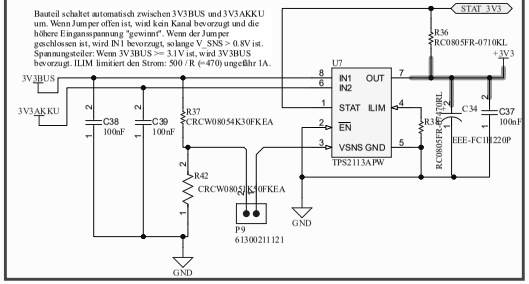
USB-C-Schnittstelle wurde anstelle der Mini-USB-Schnittstelle verwendet, um 5V @1.5A zu erhalten.



Auto-switching Power MUX

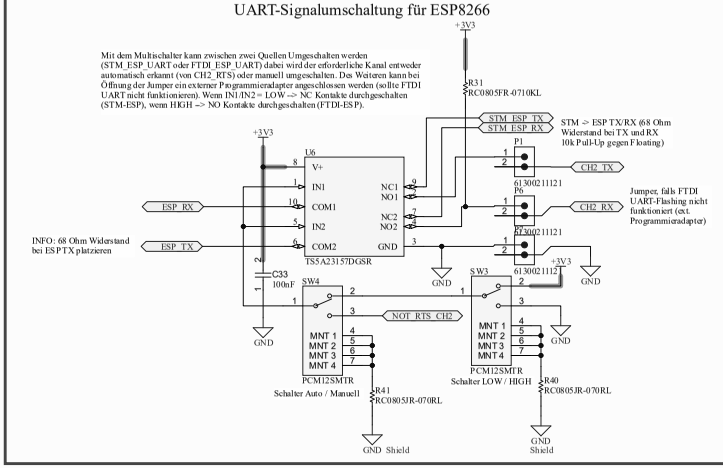
Bauteil schaltet automatisch zwischen 3V3BUS und 3V3AKKU um. Wenn Jumper offen ist, wird kein Kanal bevorzugt und die höhere Eingangsspannung "gewinnt". Wenn der Jumper geschlossen ist, wird IN1 bevorzugt, solange V_SNS > 0.8V ist. Spannungsteiler: Wenn 3V3BUS >= 3.1V ist, wird 3V3BUS bevorzugt. ILIM limitiert den Strom: 500 / R (=470) ungefähr 1A.

STAT_3V3 ist HIGH, wenn IN2 (Akku) Kanal aktiv ist.



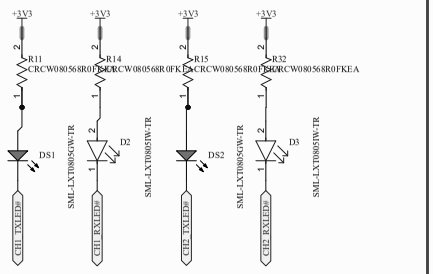
UART-Signalumschaltung für ESP8266

Mit dem Multischalter kann zwischen zwei Quellen Umgeschalten werden (STM_ESP_UART oder FTDI_ESP_UART) dabei wird der erforderliche Kanal entweder automatisch erkannt (von CH2_RTS) oder manuell umgeschalten. Das Weiteren kann bei Öffnung der Jumper ein externer Programmieradapter angeschlossen werden (sollte FTDI UART nicht funktionieren). Wenn IN1/IN2 = LOW --> NC Kontakte durchgeschalten (STM-ESP), wenn HIGH --> NO Kontakte durchgeschalten (FTDI-ESP).

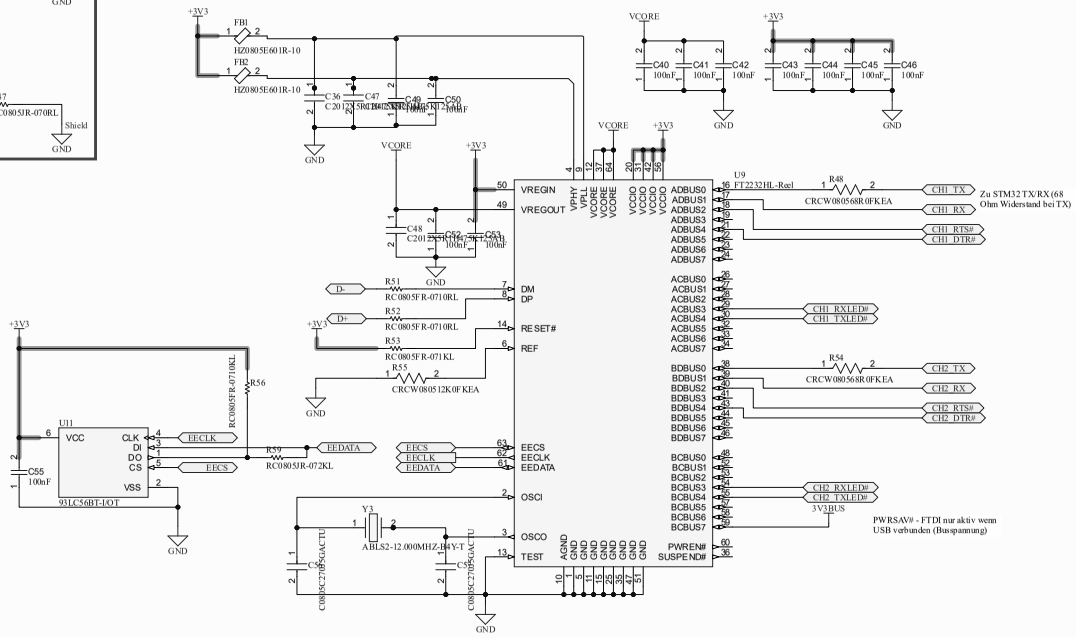


Status LEDs (UART Signal und 3V3 Akku IN2)

LEDs blinken bei UART Kommunikation (TX = Green, RX = Red)
Wenn 3V3 von Akku stammen (STAT_3V3 auf High gezogen) --> Rote LED an zur Statusanzeige



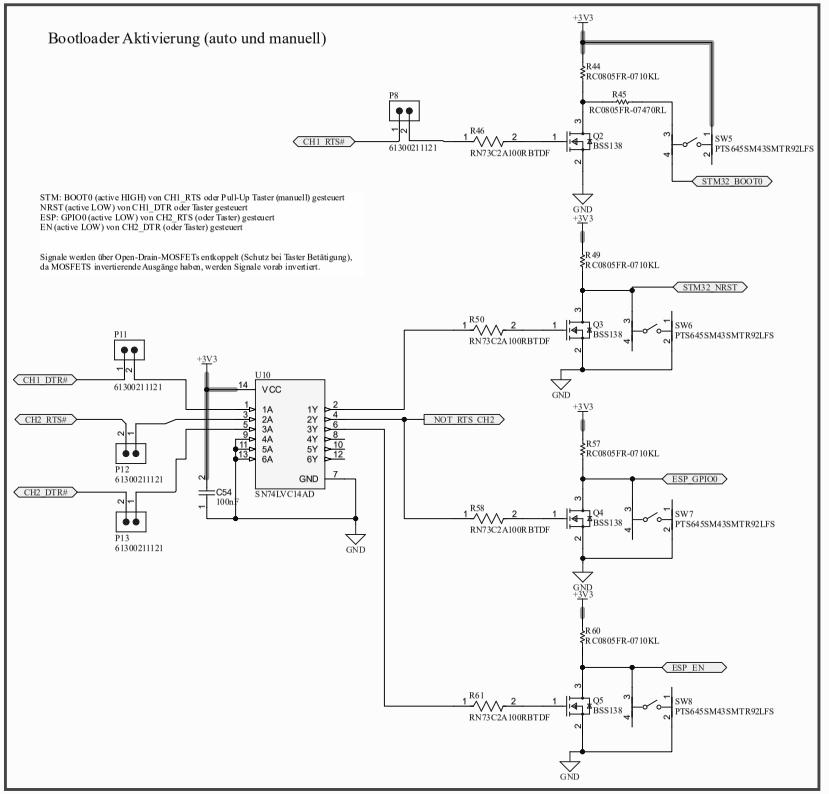
FTDI: USB-TTL/UART mit EEPROM

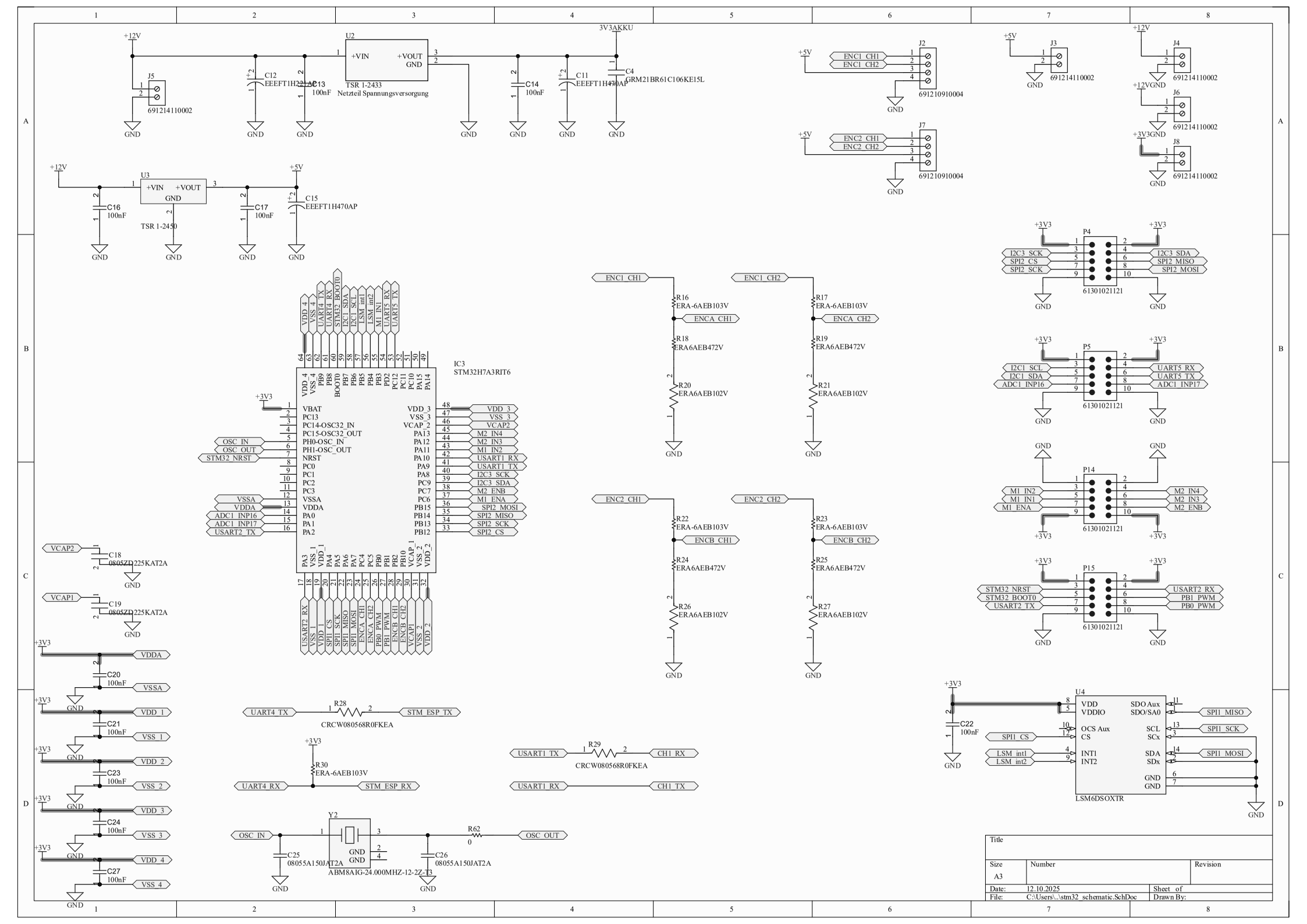


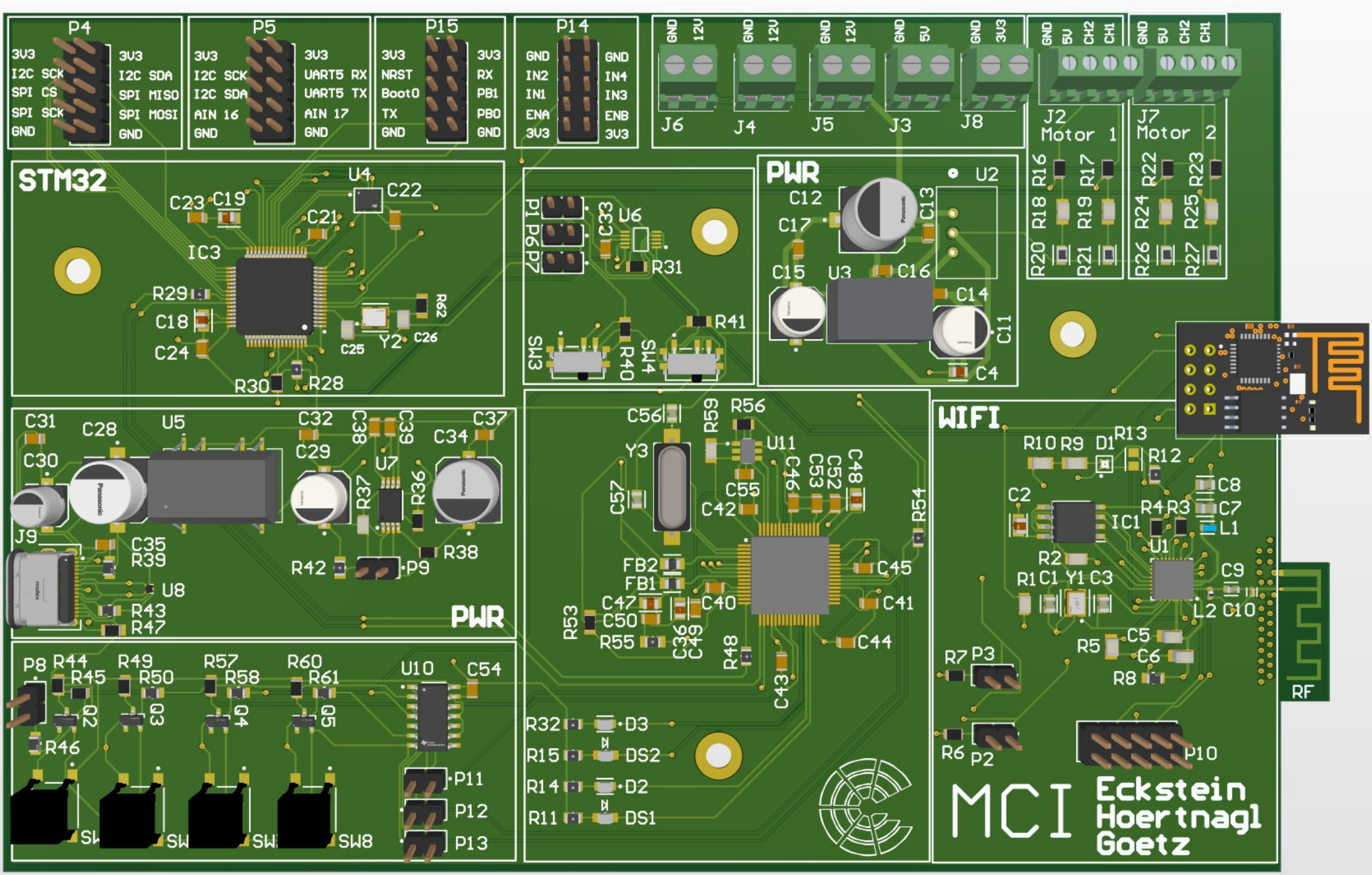
Bootloader Aktivierung (auto und manuell)

STM: BOOT0 (active HIGH) von CH1_RTS oder Pull-Up Taster (manuell) gesteuert
NRST (active LOW) von CH1_DTR oder Taster gesteuert
ESP_GPIO0 (active LOW) von CH2_RTS (oder Taster) gesteuert
EN (active LOW) von CH2_DTR (oder Taster) gesteuert

Signale werden über Open-Drain-MOSFETs entkoppelt (Schutz bei Taster Betätigung), da MOSFETS invertierende Ausgänge haben, werden Signale vorab invertiert.




+12V
U2
+VIN
+VOUT
GND
TSR 1-2433
Netzteil Spannungsversorgung
3V3AKKU
J5
691214110002
C12
EEEFT1H221AP
C13
100nF
C14
100nF
C11
EEEFT1H470AP
C4
GRM21BR61C106KE15L
GND
+5V
J2
ENC1_CH1
ENC1_CH2
691210910004
J3
691214110002
J4
J6
J7
ENC2_CH1
ENC2_CH2
J8
+3V3GND
U3
TSR 1-2450
C16
100nF
C17
100nF
C15
EEEFT1H470AP
IC3
STM32H7A3RIT6
VBAT
PC13
PC14-OSC32_IN
PC15-OSC32_OUT
PH0-OSC_IN
PH1-OSC_OUT
NRST
PC0
PC1
PC2
PC3
VSSA
VDDA
PA0
PA1
PA2
OSC_IN
OSC_OUT
STM32_NRST
ADC1_INP16
ADC1_INP17
USART2_TX
VDD_3
VSS_3
VCAP2
PA13
PA12
PA11
PA10
PA9
PA8
PC9
PC7
PC6
PB15
PB14
PB13
PB12
M2_IN4
M2_IN3
M1_IN2
USART1_RX
USART1_TX
I2C3_SCK
I2C3_SDA
M2_ENB
M1_ENA
SPI2_MOSI
SPI2_MISO
SPI2_SCK
SPI2_CS
P4
P5
P14
P15
61301021121
I2C3_SDA
I2C1_SCL
I2C1_SDA
UART5_RX
UART5_TX
ADC1_INP17
M1_IN2
M1_IN1
M1_ENA
M2_IN4
M2_IN3
M2_ENB
STM32_NRST
STM32_BOOT0
USART2_TX
PB1_PWM
PB0_PWM
ENC1_CH1
ENC1_CH2
R16
ERA-6AEB103V
R17
ERA-6AEB103V
ENCA_CH1
ENCA_CH2
R18
ERA6AEB472V
R19
ERA6AEB472V
R20
ERA6AEB102V
R21
ERA6AEB102V
ENC2_CH1
ENC2_CH2
R22
ERA-6AEB103V
R23
ERA-6AEB103V
ENCB_CH1
ENCB_CH2
R24
ERA6AEB472V
R25
ERA6AEB472V
R26
ERA6AEB102V
R27
ERA6AEB102V
VCAP2
C18
0805ZD225KAT2A
VCAP1
C19
0805ZD225KAT2A
VDDA
C20
100nF
VSSA
VDD_1
C21
100nF
VSS_1
VDD_2
C23
100nF
VSS_2
VDD_3
C24
100nF
VSS_3
VDD_4
C27
100nF
VSS_4
UART4_TX
R28
CRCW080568R0FKEA
STM_ESP_TX
R30
ERA-6AEB103V
UART4_RX
STM_ESP_RX
USART1_TX
R29
CRCW080568R0FKEA
CH1_RX
USART1_RX
CH1_TX
Y2
ABM8AIG-24.000MHZ-12-2Z-T3
C25
08055A150JAT2A
C26
08055A150JAT2A
R62
0
OSC_IN
OSC_OUT
U4
LSM6DSOXTR
C22
100nF
VDD
VDDIO
GND
OCS Aux
CS
SCL
SCx
SDA
SDx
INT1
INT2
SDO Aux
SDO/SA0
SPI1_MISO
SPI1_SCK
SPI1_CS
SPI1_MOSI
LSM_int1
LSM_int2
Title
Size
A3
Number
Revision
Date:
12.10.2025
Sheet of
File:
C:\Users\..\stm32_schematic.SchDoc
Drawn By:


P4
3V3
I2C SCK
SPI CS
SPI SCK
GND
3V3
I2C SDA
SPI MISO
SPI MOSI
GND
P5
3V3
I2C SCK
I2C SDA
AIN 16
GND
3V3
UART5 RX
UART5 TX
AIN 17
GND
P15
3V3
NRST
Boot0
TX
GND
3V3
RX
PB1
PB0
GND
P14
GND
IN2
IN1
ENA
3V3
GND
IN4
IN3
ENB
3V3
GND 12V
J6
GND 12V
J4
GND 12V
J5
GND 5V
J3
GND 3V3
J8
GND 5V CH2 CH1
J2
Motor 1
GND 5V CH2 CH1
J7
Motor 2
STM32
IC3
U4
C22
C23
C19
C21
R29
C18
C24
R30
R28
C25
Y2
C26
R62
P1
P6
P2
C33
U6
R31
SW3
R40
SW4
R41
PWR
U2
C12
C17
C13
C15
U3
C16
C14
C11
C4
R16
R17
R18
R19
R20
R21
R22
R23
R24
R25
R26
R27
C31
C28
U5
C32
C38
C39
C37
C34
C30
C29
U7
R37
R36
J9
C35
R39
U8
R43
R47
R42
P9
R38
PWR
C56
R59
R56
U11
Y3
C57
C55
C46
C53
C52
C48
C42
R54
FB2
FB1
C47
C50
R55
R53
C36
C49
C40
R48
C45
C41
C44
C43
R32
D3
R15
DS2
R14
D2
R11
DS1
WIFI
R10
R9
D1
R13
R12
C2
IC1
R4
R3
C8
C7
L1
U1
R2
R1
C1
Y1
C3
C9
L2
C10
R5
C5
C6
R8
R7
P3
R6
P2
P10
RF
P8
R44
R45
R49
R50
R57
R58
R60
R61
U10
C54
Q2
Q3
Q4
Q5
R46
P11
P12
P13
SW
SW
SW
SW8
MCI
Eckstein
Hoertnagl
Goetz