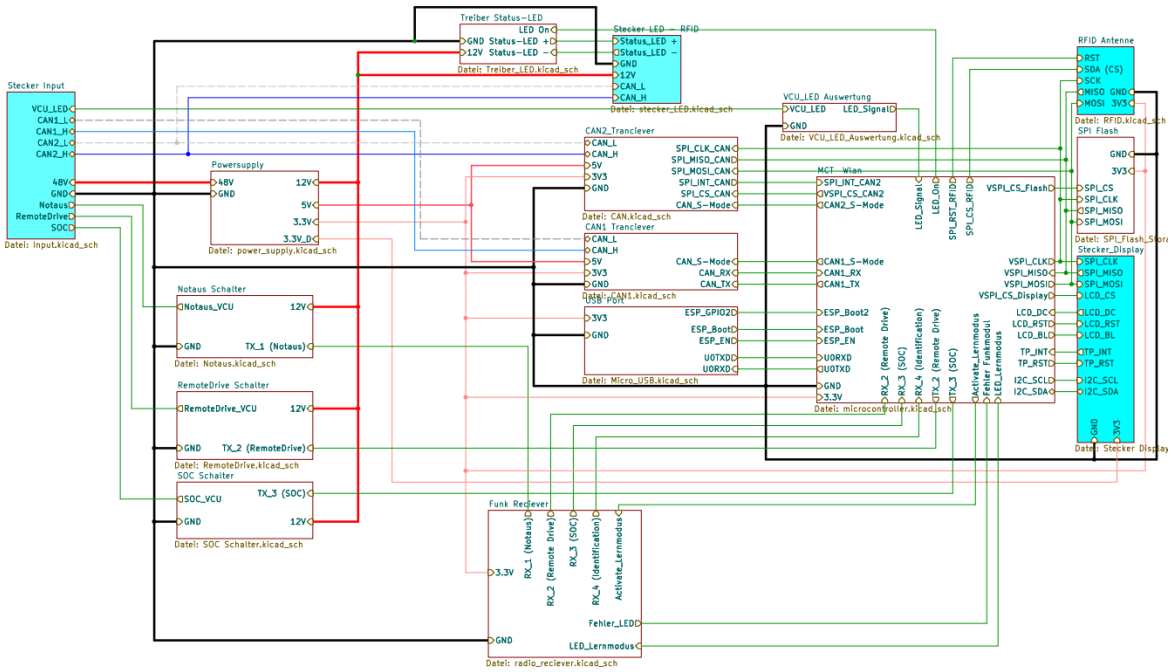
5. Hardwareentwicklung

5.1. Schaltungsaufbau

Der erste Entwicklungsschritt für das neue Erweiterungssystem ist der Aufbau geeigneter Schaltungen, welche alle Funktionalitäten abbilden. In diesem Kapitel soll es daher um die Schaltungen, deren Aufbau und Funktionsweiße gehen. Der Schaltungsaufbau ist dabei so organisiert, dass jede Einzelschaltung in einem eigenen Schaltplan realisiert ist und ein Masterblatt diese Systeme alle miteinander verbindet. Dabei wird jede Schaltung einzeln betrachtet und erläutert. Zur Übersicht ist das Gesamtsystem folgend dargestellt.



Schaltplan 1: Gesamtübersicht der Schaltung

Mikrocontroller

Zuallererst soll hierbei der Mikrocontroller als Zentrales Element der Schaltung in Schaltplan 2 betrachtet werden. Dabei kommt ein ESP32-Microcontroller zum Einsatz. Dieser übernimmt sämtliche Rechenoperationen und Kommunikationsaufgaben für das System. Der Mikrocontroller wird dabei mit 3.3V versorgt. Dadurch, dass der 3.3V auch Signalpegel für viele Signale ist, ist es sehr wichtig, ein stabiles und rauschfreies Signal am Microcontroller zu erhalten. Dafür sind die Kondensatoren C801 und C802 zwischen VCC und Ground vorgesehen [13, S.37]. Bei ihnen handelt es sich um Abblockkondensatoren, welche Hochfrequente Störungen gegen Ground ableiten und kurzfristige, minimale Spannungseinbrüche abpuffern sollen. Außerdem ist es wichtig sicherzustellen, dass die Stromversorgung beim Einschalten des ESP32 stabil ist, um einen sicheren Boot-Vorgang zu gewährleisten ohne Fehler zu erhalten. Das Einschalten passiert dabei über den EN-Pin des Mikrocontrollers. Hierfür wird ein RC-Glied vor den

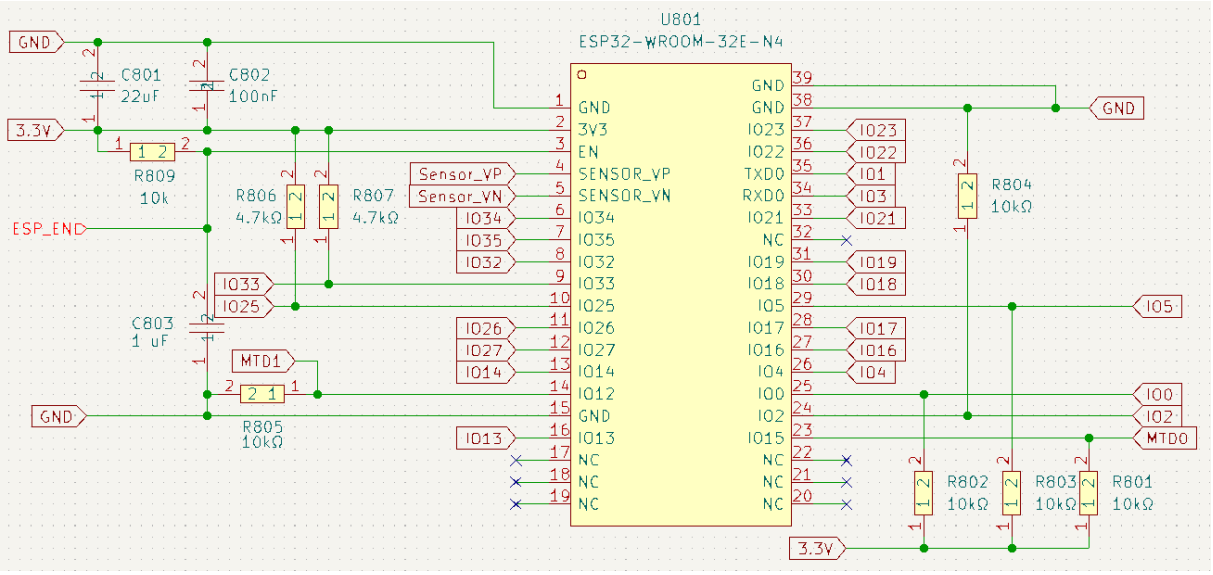
Enable-Pin geschalten. Das Datenblatt empfiehlt dabei mindestens $t_{STBL} = 50\mu s$ als Wartezeit, bis die Spannungsversorgung am ESP32 stabil anliegt, bevor der Chip über EN-Pin aktiviert wird [20, S. 19]. Die Verzögerung des RC-Glied lässt sich dabei folgendermaßen berechnen

$$\tau = RC$$

$f\ddot{u}r\ R = 10\ k\Omega$: $$C > 50\mu s/10k\Omega$$

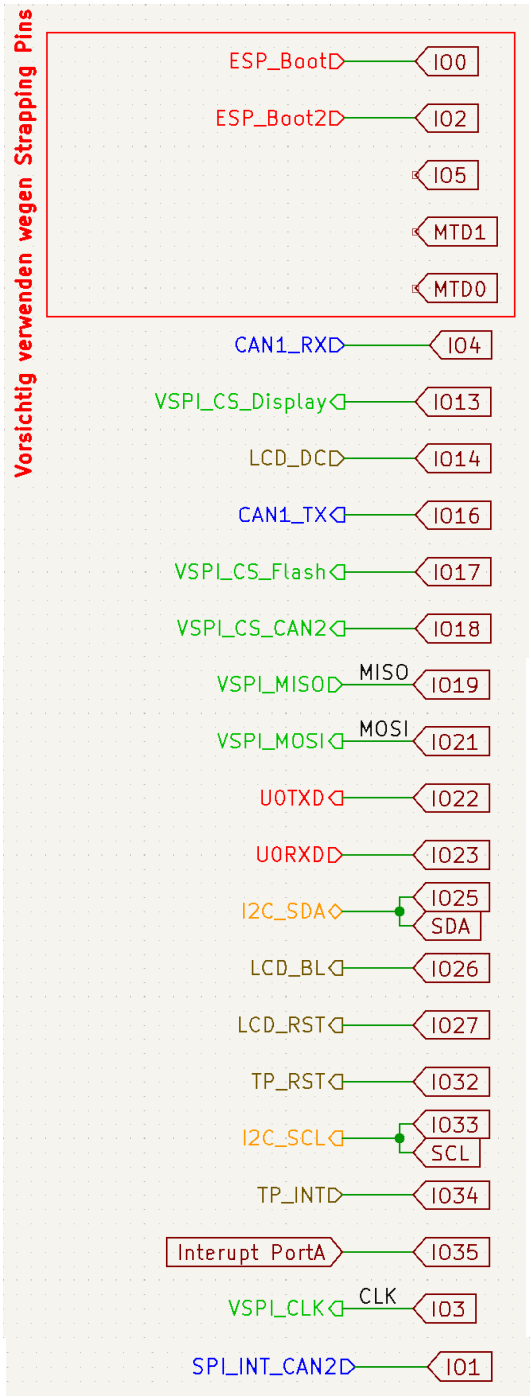
$$C > 5nF$$

Das Datenblatt empfiehlt für $C = 1\mu F$ [20, S.19] , das entspricht einer Verzögerung von $\tau = 10\ k\Omega * 1\mu F = 10ms$.



Schaltplan 2: Beschaltung des Mikrocontrollers

Zusätzlich beim Bootvorgang zu beachten sind die Strapping Pins des Mikrocontrollers. Bei Strapping-Pins handelt es sich um die Pins, welche den Ablauf des Bootvorgangs definieren. So lässt sich beispielsweise der Boot Mode definieren, ob die Software, welche bereits auf dem Gerät gespeichert ist für den Bootvorgang verwendet wird oder ob extern neue Software aufgespielt werden soll. Die Strapping Pins des ESP32 sind dabei GPIO 0, GPIO 2, GPIO 5, GPIO 12 (MTD1), und GPIO 15 (MTD 0). Für diese Pins ist im Datenblatt eine Standardbelegung definiert, bei welcher der normale Bootvorgang abläuft, wobei der Chip aus seinem eigenen Speicher bootet [13, S. 13]. Die Pegel werden dabei über Pull-Up und Pull-Down Widerstände erzielt, das sind die Widerstände R801 bis R805. Um diesen Bootvorgang nicht zu beeinflussen, wurde daher bewusst darauf verzichtet, die Strapping Pins für weitere Funktionalitäten zu nutzen. Lediglich die Pins GPIO 0 und 2, welche den Bootmodus definieren, werden noch zusätzlich vom USB-Controller gesteuert, um ein Flashen per USB zu ermöglichen [13, S. 13f].



Schaltplan 3: Pinbelegung des Mikrocontrollers

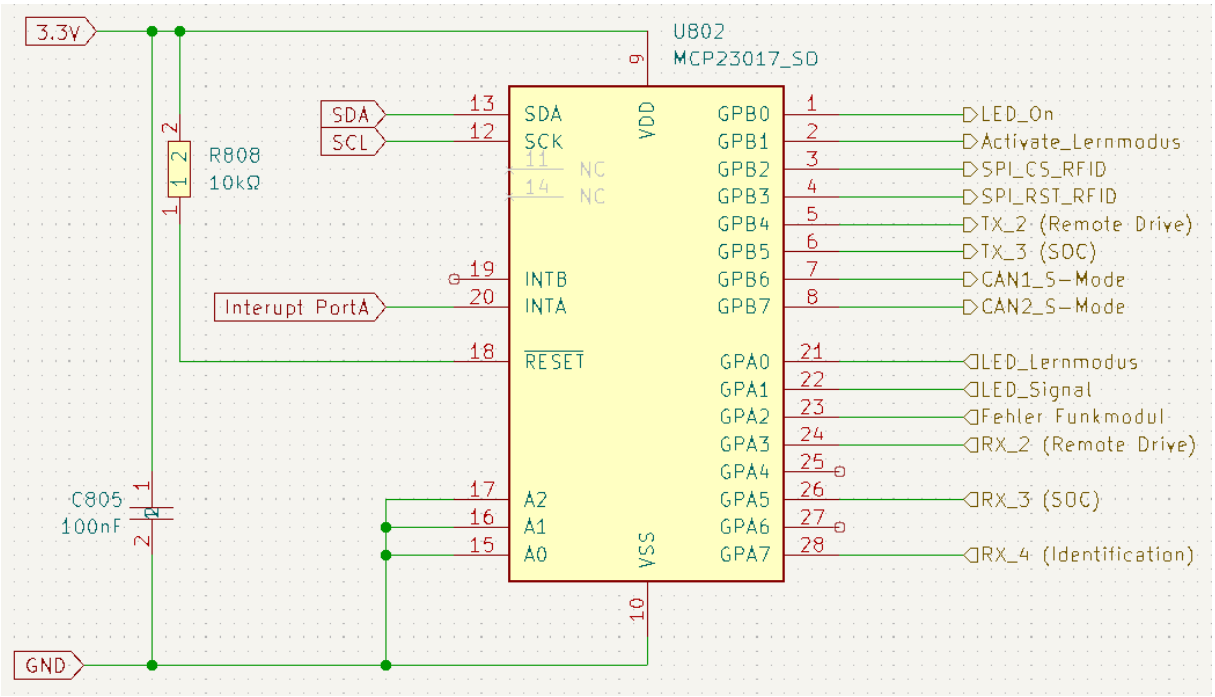
Ein großer Vorteil des ESP32 ist, dass beinahe alle Pins für alle Funktionen verwendet werden können. Lediglich die Pins GPIO 34, 35, 36 und 39 sind nur als Input verwendbar und die Pins GPIO 6 bis 11 werden intern verwendet und dürfen nicht verwendet werden [13, S. 11f]. Somit bietet der Chip sehr viele Möglichkeiten für Signale oder Kommunikation. Die Belegung der Pins kann im Folgenden dem Schaltplan 3 entnommen werden.

Ein verwendetes Kommunikationsprotokoll ist I^2C. Es wird sowohl zur Ansteuerung des Displays als auch für die Datenübertragung an eine GPIO Port Erweiterung verwendet. I^2C ist ein Protokoll, welches mit zwei Signalleitungen auskommt. Für Datenübertragung (SDA) ist hierbei Port GPIO 25 vorgesehen, für das Clock-Signal wird Pin GPIO 33 verwendet. Die Auswahl der Pins beruht dabei auf praktischen Gesichtspunkten, so sollen die Ports physisch möglichst nebeneinander liegen, um das spätere Layout zu vereinfachen. Für I^2C müssen zudem Pull-Up Widerstände vorgesehen werden, da es sich bei I^2C um Open Drain-System handelt, weswegen die Signale auf einen Stabilen Pegel gezogen werden müssen. Bei Übertragung eines Signals wird die Leitung dann auf Ground Potential gezogen. Bei I^2C haben sich 4.7 kΩ als Standardwert für die I^2C Kommunikation etabliert. Ein weiterer sehr verbreiteter Standard zur Kommunikation mit Peripheriegeräten ist SPI. Bei dem aktuellen System werden damit das Display, ein Flashspeicher, sowie der Controller zur Kommunikation auf CAN-Bus 2 gesteuert. Anders als I^2C benötigt SPI mindestens 4 Leitungen. Dazu gehört auch wieder ein Clock-Signal, welches jeden Teilnehmer synchronisiert. Dazu kommen anders als bei I^2C aber zwei Datenleitungen. Hier wird unterschieden zwischen gesendeten und empfangenen Nachrichten. Jeder Teilnehmer besitzt ein Master In, Slave Out-Signal (MISO) und ein Master Out, Slave In-Signal (MOSI) über welches die Teilnehmer miteinander kommunizieren können. Hinzu kommt ein ChipSelect Signal für jeden Teilnehmer, der auf dem Bus Kommunizieren möchte. Über dieses Signal muss immer genau ein Teilnehmer vom Master freigegeben werden, welcher dann kommunizieren kann. Auch hier werden

die Pins so ausgewählt, dass das Layout möglichst vereinfacht werden kann. Eine weitere verbreitete Kommunikationsart, vor allem zur Kommunikation unter Steuergeräten, ist der CAN-Bus. Der Chip besitzt intern einen Controller für Two-Wire Automotive Interface (TWAI), welcher in der Lage ist eine CAN-Kommunikation aufzubauen [20, S. 43]. Dabei werden 2 physische Leitungen für CAN-High und CAN-Low benötigt. Diese werden nicht am Mikrocontroller angeschlossen, weil zur Kommunikation noch ein CAN-Transceiver benötigt wird. Dieser erhält über UART seine Daten und Befehle vom TWAI-Controller, dabei kommunizieren die Teilnehmer über zwei Leitungen für erhaltene und gesendet Nachrichten. Daher werden auch zwei Pins am Chip dafür verwendet. Da es sich hierbei um ein asynchrones Protokoll handelt, ist hier kein Clocksignal wie bei SPI oder I^2C nötig. Im vorliegenden System kommuniziert darüber sowohl der RFID-Reader als auch die Datenübertragung mit dem Steuergerät findet über CAN-Kommunikation statt. Da der Mikrocontroller selbst leider keine zwei CAN-Busse unterstützt muss für den zweiten Bus ein externer CAN-Controller vorgesehen werden, dieser überträgt seine Daten via SPI. Der Controller für Bus 2 bietet allerdings eine Interrupt-Funktionalität. Bei Empfang einer Nachricht wird über diesen Pin ein Signal ausgegeben, welches am ESP32 als Interrupt detektiert werden kann.
Die Signale U0TXD und U0RXD gehören zu einer UART-Schnittstelle. Diese wird im vorliegenden Beispiel lediglich zur Kommunikation mit einem extern angeschlossenen USB-Gerät verwendet und soll als Schnittstelle zum flashen eines neuen Programms dienen. Auch hier wird mit zwei Leitungen kommuniziert, wobei jeweils zwischen Receive und Transmit unterschieden wird, also eine Signalleitung, zum Empfangen von Nachrichten und eine zum Senden. Auch hier werden die nebeneinanderliegenden Pins GPIO 22 und 23 genutzt. Die Restlichen Pins werden für die Steuerung des Displays genutzt. Zum einen gibt es einen Pin für die Hintergrundbeleuchtung des Displays. Dafür wird der Pin als Ausgang eines PWM-Generators definiert. Über die Frequenz des Pulsweiten Modulierten Signals wird somit die Helligkeit der Hintergrundbeleuchtung bestimmt. Ein weiterer Pin ist der DC-Pin des Displays. Dieser ist ein standardmäßiger Ausgang für ein Digitalsignal. Dieses Signal wird dafür genutzt, um dem Display mitzuteilen, ob ein Befehl übertragen wird oder ob es gerade Daten zum Darstellen erhält. Die Pins GPIO 26 und 27 werden als Digitaler Ausgang genutzt. Hier werden Signale ausgegeben, welche genutzt werden können, um das Display und den Touch Controller zu resetten. GPIO 34 wird ähnlich wie bei CAN genutzt, um einen Interrupt durch den Touch Controller des Displays auszulösen. Der Pin wird auf High gezogen, wenn der Touch Controller eine Berührung erkennt, und der Pin kann im Chip als Interrupt definiert werden, sodass jede Berührung direkt verarbeitet werden kann. Hierfür wird einer der Pins genutzt, welche nur als Eingang fungieren, da es sich nur um einen Einseitigen Interrupt handelt. Lediglich der Touch Controller darf diesen Pin setzen, niemals der Mikrocontroller, dieser wertet das Signal lediglich aus. Dasselbe gilt auch für den Pin GPIO 35. Über diesen ist der Interrupt-Pin des GPIO-Expanders an den Chip angeschlossen. Dieser erzeugt ein Signal an dem Pin, welcher dann als Interrupt im ESP32 definiert werden kann, sobald sich der Wert an einem der als Input definierten Pins

ändert. Somit können auch dort ausgelesene Daten zeitnah vom ESP32 ausgelesen und verarbeitet werden.

Die bereits angesprochene GPIO Port Erweiterung hat dabei die Funktion alle übrigen Funktionalitäten und Pins zur Verfügung zu stellen, welche am Mikrocontroller keinen Platz mehr gefunden haben. Die Port Erweiterung stellt allerdings lediglich GPIO-Funktionalität zur Verfügung und kann keine Sonderfunktionalitäten wie SPI oder I^2C bereitstellen. Daher sind die Pins am Chip für die Kommunikation verwendet und die meisten GPIO-Funktionalitäten werden über die Erweiterung erzielt. Zudem wurde darauf geachtet keine Zeitkritischen Signale über die Erweiterung zu realisieren, da hier jedes Mal per I^2C erst Daten abgefragt werden müssen, bevor diese am ESP32 verarbeitet werden können, wodurch die Daten deutlich später zur Verfügung stehen, als wären sie direkt am Chip angeschlossen. Der GPIO-Expander besitzt dabei 2 Register mit jeweils acht Eingängen [21, S. 11]. Um es übersichtlich zu halten, wurde Register A nur für Eingangssignal und Register B nur für Ausgangssignale verwendet. Das hat außerdem den Vorteil, dass das Signal für Interrupts von Port B ignoriert werden kann, da dort keine Eingehenden Signale erwartet werden, wodurch ein Port mehr für andere Funktionalität direkt am Chip erhalten bleibt. So übernimmt der GPIO-Expander zum Beispiel das Auslesen der vom Funkempfänger gesendeten Signale.



Schaltplan 4: Beschaltung der GPIO-Porterweiterung

Die Angeschlossenen Signale sind in Schaltplan 4 dargestellt und zu entnehmen. Dazu gehören neben dem Fehlersignal und der Rückgabe des Lernmodus auch die Empfangenen Signale auf Kanal 2, 3 und 4. Durch die Interrupt Funktionalität des Bauteils wird der Mikrocontroller benachrichtigt, sobald sich an diesen Werten etwas ändert. Neben den Signalen liest die Porterweiterung auch das LED-Signal aus, welches von der VCU gesendet wird. Da dieses nur noch in seltenen Fällen verwendet werden wird, nämlich wenn ein altes Kart mit einer neuen Platine aufgerüstet werden soll, hat dieses

Signal keine hohe Priorität, weshalb die höhere Bearbeitungszeit, bis es am Mikrocontroller ankommt, vernachlässigbar ist. Gleichzeitig wird über die Erweiterung auch die LED gesteuert. LED_ON ist dabei das Signal, welches den Transistor für die LED-Schaltet. Auch hier fallen minimal längere Verarbeitungszeiten nicht ins Gewicht. Dasselbe gilt für das Signal Activate_Lernmodus. Hierrüber wird der Lernmodus des Funkempfängers aktiviert, mit welchem man neue Fernbedienungen einem Funkmodul zuweisen kann. Dieses Signal besteht aus mehreren Impulsen, welche mit Verzögerung gesendet werden, sodass hier genügend Zeit zwischen den Signalen besteht, um einen neuen Impuls über I^2C an die Erweiterung zu übermitteln. Zwei weitere Signale, welche über die Port Erweiterung realisiert, werden sind CAN1_S-Mode und CAN2_S-Mode. Über diese Signale wird der Silent Mode für die CAN-Transceiver für CAN1 und CAN2 aktiviert. Dieses Signal eignet sich sehr gut, um über die langsamere Expansion realisiert zu werden, da dieses Signal nur sehr selten verändert wird. Bei den Signale TX_2 und TX_3 handelt es sich wieder um Signale, welche nur zur Kompatibilität mit alten Systemen vorgesehen sind. TX_2 realisiert dabei die Aktivierung des Fahrmodus, wenn ein Signal per Funk empfangen wurde, TX_3 sendet eine SOC-Anfrage zur VCU. Diese Signale kommen aber nur bei Karts alter Generationen zum Einsatz, die keinen CAN-Bus zur Kommunikation zur Option 1 führen, sondern bei denen das Signal noch über eine eigene Signalleitung an die VCU übermittelt wird. Die restlichen beiden Signale sind Steuersignale für einen über SPI gesteuerten RFID-Reader. Diese sind lediglich für testzwecke vorgesehen, um den sehr teuren RFID-Reader, welcher aktuell direkt per CAN angeschlossen ist, zukünftig ersetzen zu können durch ein deutlich günstigeres Bauteil. Deshalb wird hier auch ein Signal wie ein SPI ChipSelect und der Reset für den Controller über den langsamen GPIO-Expander realisiert, obwohl es sich bei Signalen für die SPI-Kommunikation um zeitkritischere Signale handelt. Allerdings werden diese Signale im normalen Betrieb keinerlei Rolle spielen. Um überhaupt Daten übermitteln zu können muss die Adresse des Geräts definiert werden. Das passiert über eine 7 Bit lange Adresse, wobei die letzten Drei Bit über die Eingänge A0 bis A2 definiert werden [21, S. 15]. Das ist vor allem dann relevant, wenn in einer Schaltung mehrere GPIO-Port Erweiterungen verbaut werden. Im vorliegenden System gibt es keine Überschneidungsgefahr bei den Adressen, weshalb die letzten Adressbits auf 0 0 0 festgelegt werden, definiert durch das Ground-Potenzial. Die Spannungsversorgung erfolgt auch hier mit 3.3V. Auch hier ist ein Abblockkondensator vorgesehen, um eine stabilere Spannungs-versorgung zu gewährleisten. Der Widerstand vor dem Reset Pin sorgt dafür, dass das lowaktive Signal auf einem definierten Pegel liegt und die Port Erweiterung dauerhaft aktiv ist, ohne ungewollt zurückgesetzt zu werden [21, S. 12].

Funkempfänger

Das Funkmodul, welches die Signale für Aktivierung des Notaus, Aktivierung des Fahrmodus oder Abfrage des State of Charge empfangen wird, wird als Aufsteckplatine realisiert. Dafür wird das Empfangs – und Auswertmodul CX-12 R von SVS verwendet.

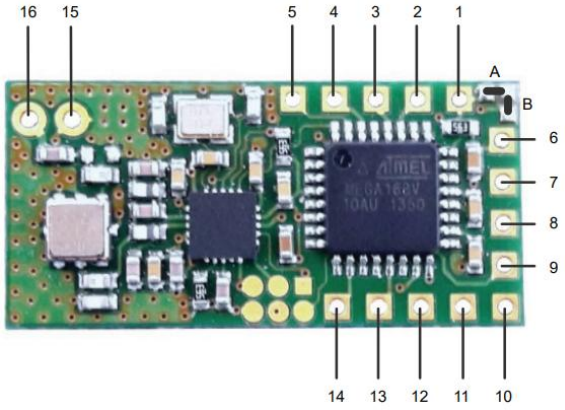


Abbildung 19: Pinbezeichnungen des Funkmoduls [22]

Dieses ist mit den bisher verwendeten Funkfernbedienungen kompatibel und bietet alle Funktionalitäten, welche für das System benötigt werden. Das Modul besitzt 16 Pins, welche im Folgenden mit den Bezeichnungen aus Abbildung 19 genauer betrachtet werden sollen, sowie die Beschaltung, welche näher beleuchtet und erklärt wird. Der erste Pin ist für die Aktivierung eines Sleep Modus vorgesehen. Um den Stromverbrauch zu senken, wird nach einiger Zeit eine Taktung aktiv. Diese Taktung tritt 10 Minuten nach Einschalten des Moduls oder dem letzten Datenempfang in Kraft. Ist diese Taktung aktiv, treten allerdings Verzögerungen beim Empfang und dem Verarbeiten der Funksignale auf. Diese Latenz kann bis zu einer Sekunde betragen. Über den Pin lassen sich dafür verschiedene Modi aktivieren. Dabei wird unterschieden, ob die Lötbrücken A und B auf der Platine gesetzt sind:

	Pin 1 Pegel	
	GND	VCC / unverbunden
Lötbrücke B gesetzt	Taktung nach 10 Minuten seit Datenempfang	Dauerhaft in Taktung
Lötbrücke B nicht gesetzt	Dauerbetrieb ohne Taktung	Taktung nach 10 Minuten seit Datenempfang

Tabelle 1: Pin 1 Funktionalität des Funkmoduls [22, S. 2]

Die Lötbrücke A bewirkt, das Pin 1 dauerhaft auf Ground gezogen wird. Die Platine wird standardmäßig ohne gesetzte Lötbrücken geliefert. Um den Arbeitsaufwand zu verringern, soll es nicht nötig sein, jede Platine noch einmal händisch bearbeiten zu müssen, um Lötbrücken zu setzen. Daher soll der Modus nur über den Pegel an Pin 1 bestimmt werden und beide Lötbrücken bleiben ungesetzt [22, S. 2]. Für das aktuelle System soll der Empfänger dauerhaft aktiv bleiben, somit also Pin 1 auf Groundpotential gezogen werden ohne gesetzte Lötbrücke. Das hängt damit zusammen, dass durch den Notaus auch Sicherheitsrelevante Funktionalitäten über den Funkempfänger realisiert werden, für den 1 Sekunde Latenz zu lange als Verarbeitungszeit sind.

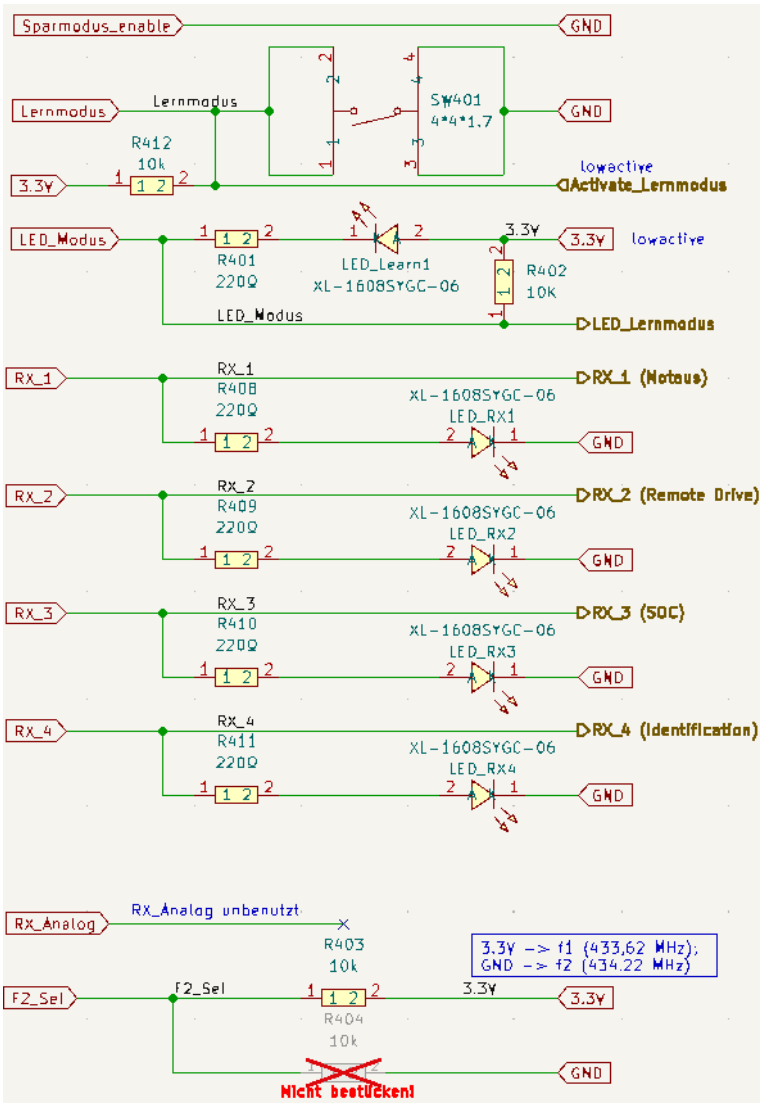
Um Signale empfangen zu können, muss es zuerst möglich sein Fernbedienungen mit dem Empfänger zu verbinden. Dafür bietet das Funkmodul mehrere Modi, welche sich

über Pin 2 aktivieren lassen. Die Lernmodi werden dabei durch unterschiedliche Impulse auf Ground Potential aktiviert:

Modus	Aktivierungsmethode
Lernmodus I	GND, 1x kurz (<1s)
Lernmodus II	GND, 2x kurz (<1s)
Lernmodus III	GND, 3x kurz (<1s)
Lernmodus IV	GND, 4x kurz (<1s)
Löschmodus I	GND, 1x lang (>3s)
Löschmodus II	GND, 2x lang (>3s)

Tabelle 2: Aktivierungsablauf der Lern- und Löschmodi des Funkmoduls [22, S. 2]

Wenn ein neuer Sender mit dem Empfängermodul verbunden wird, wird dessen Adresse im Funkempfänger gespeichert. Dieser besitzt bis zu 60 Speicherplätze für Senderadressen. Bei Lernmodus I und III wird eine neue Fernbedienung mit dem Modul verbunden. Hierbei können nun über alle Tasten die unterschiedlichen Funktionalitäten von Kanal 1 bis 4 genutzt werden. Der Unterschied zwischen Modus I und III liegt darin, dass bei Modus I eine Bestätigung an den Sender übermittelt wird und der Nutzer somit an der Fernbedienung eine Rückmeldung über ein eventuelles Fehlschlagen der Verbindung erhält [22, S. 2]. Mit Modus II und IV lässt sich eine einzelne Taste anlernen. Hier wird nicht die gesamte Fernbedienung mit dem Empfangsmodul gekoppelt, sondern lediglich die jeweilige taste. Auch hier unterscheiden sich die beiden Modi nur in der Rückmeldung an den Sender [22, S. 2]. Die verbleibenden beiden Modi werden genutzt, um bereits angelernte Fernbedienungen wieder zu lösen. Dabei entfernt Modus I einen Eintrag aus der Liste. Dabei handelt es sich um die Adresse des Senders, von welchem das Modul ein Signal empfängt, solange der



Schaltplan 5: Beschaltung des Funkmoduls

Löschmodus aktiv ist. Löschmodus II entfernt hingegen alle gespeicherten Einträge aus der Adressen Speicherliste [22, S. 2]. Auf dem bisherigen Gesamtmodul wurde diese Funktionalität über einen Knopf auf der Platine realisiert. Das hatte zur Folge, dass eine neue Fernbedienung nur im Ausgebauten Zustand verbunden werden konnte. Mit dem neuen System soll das auch über den Mikrocontroller gesteuert werden können und damit eine Aktivierung über die Netzwerkverbindung oder das Display möglich sein. Nichtsdestotrotz ist weiterhin ein Knopf vorgesehen, um das Funkmodul ohne den Chip testen zu können. Bei dem Knopf handelt sich dabei um einen Drucktaster, welcher den Pin auf Ground zieht. Der Widerstand R412 dient dabei als Pull-Up Widerstand, um bei offenem Schalten einen definierten Spannungspegel am Pin zu garantieren. Der Pin ist zusätzlich mit der GPIO Port Erweiterung am ESP32 verbunden, über welcher nun durch Schalten des GPIO-Ports gegen Ground ebenfalls der Lernmodus aktiviert werden kann.

Rückmeldung über den Aktivierten Lernmodus gibt Pin 3. Bei dem Pin handelt es sich um einen LED-Ausgang, welcher durch unterschiedliche Frequenz unterschiedliche Modi anzeigt.

Modus	**Anzeige**
Normaler Betrieb	1x blinken pro empfangenes Signal
Lernmodus I	1x Blinken pro 2 Sekunden
Lernmodus II	2x Blinken pro 2 Sekunden
Lernmodus III	3x blinken pro 2 Sekunden
Lernmodus IV	4x blinken pro 2 Sekunden
Löschmodus I	Blinkt dauerhaft

Tabelle 3: Rückmeldungen der einzelnen Lern- und Löschmodi des Funkmoduls [22, S. 2]

Diese Rückmeldung soll für Test- und Debugging Zwecke auch visuell über eine LED dargestellt werden, soll aber auch vom Mikrocontroller ausgewertet werden können, um Rückmeldung per Display oder Netzwerk geben zu können, sobald die Modi über den ESP32 aktiviert wurden. Da es sich bei dem Ausgang um ein Digitalsignal bis maximal 3.3V handelt, kann das Signal einfach an einen der Pins der Porterweiterung des Mikrocontrollers angeschlossen werden. Um einen Stabilen High-Pegel für das Lowaktive Signal zu erzeugen, ist R402 als Pull-Up Widerstand dazwischen geschalten. Die LED wird zwischen den Ausgang des Funkmoduls und ein 3.3V Signal geschalten, da es sich bei dem Signal um ein lowaktives handelt. Als Vorwiderstand dient hier R401. Der Widerstandswert ergibt sich dabei aus dem gewünschten Strom, der durch die LED fließen soll:

$$U_{LED} = 2.2V$$
$$U = 3.3V$$
$$I = 5\ mA$$
$$R_V = \frac{U - U_{LED}}{I} = \frac{3.3V - 2.2V}{5mA} = 220\Omega$$

Durch die LED fließt damit ein Strom von 5 mA.

Bei Pin 4 handelt es sich um den ersten digitalen Ausgang des Funkempfängers. Dieser Ausgang schaltet auf High, wenn ein Funksignal empfangen wird, ausgelöst durch den Druck der Taste 4 auf einer der verbundenen Fernbedienungen. Der Ausgang bleibt danach so lange auf 3.3V Spannungspegel bis die Taste losgelassen wird, mindestens aber 1.5 Sekunden [22, S. 2]. Der Pin wird zur Auswertung direkt mit dem Mikrocontroller verbunden. Zusätzlich wird noch eine LED hinzugefügt, welche den Pegel des Ausgangs anzeigt, um das Debugging zu vereinfachen und die Fehlersuche zu erleichtern. Da es sich hier um ein High-Aktives Signal handelt wird die LED gegen Ground geschalten. Auch hier sollen 5 mA Strom durch die LED fließen, weshalb auch hier wieder 220Ω als Vorwiderstand genutzt werden.
Um digitale Ausgänge handelt es sich auch bei den Pins 5, 8 und 9 für die Kanäle 3, 2 und 1. Diese verhalten sich genau wie der Digitale Ausgang für Kanal 4 mit der Erweiterung, dass die 3 restlichen Kanäle zusätzlich zum ESP32 auch noch einen Transistor schalten, welcher das Signal zusätzlich bei alten Systemen an die VCU übermitteln soll [22, S. 2]. Bei dem Signal von Kanal 1 fällt zusätzlich noch die Verbindung zum Mikrocontroller weg, da es sich hier um ein Sicherheitsrelevantes Signal handelt, welches ausfallsicher realisiert werden muss und deshalb nicht softwareseitig verarbeitet werden soll.

Ein weiterer Ausgang für empfangene Signale ist ein Analogausgang auf Pin 6. An diesem Pin wird ein Analogwert zwischen 3.3V und 0V mit 10 Bit Auflösung ausgegeben [22, S. 2]. Für unser System wird dieser Ausgang jedoch nicht benötigt, weshalb er unverbunden bleibt.

Der nächste Pin wird zur Konfiguration des Moduls benötigt. Das Modul erwartet Signale auf einer Frequenz von 433,62 MHz. Sollte diese Frequenz nicht nutzbar oder gestört sein besteht die Möglichkeit auf eine Alternativfrequenz von 434,22 MHz auszuweichen [22, S. 2]. Die Auswahl dieser Frequenz wird mit dem Pegel an Pin 7 definiert. Liegt der Pin auf GND, arbeitet das Modul auf Frequenz 2, liegt der PEGEL BEI 3.3V oder wird der Pin nicht beschalten, arbeitet der Empfänger mit Frequenz 1. Da bereits ein ähnliches Modul im Einsatz war, welches mit denselben Frequenzen arbeitete, und dort kaum Probleme auftraten, wird die Frequenz einmalig festgelegt. Zu testzwecken soll trotzdem die Möglichkeit bestehen im fehlerfall auf alternative Frequenzen auszuweichen, weshalb ein Widerstand als Pull-Up auf 3.3V und ein Pull-Down Widerstand gegen Ground vorgesehen sind, wobei der Pull-Down Widerstand standardmäßig unbestückt bleibt.

Die folgenden Pin 10 und 11 definieren die Versorgung und logischen Pegel für den Funkempfänger. Pin 10 ist die Versorgungsspannung, welche auch gleichzeitig den logischen High-Pegel definiert, und zwischen 2.0 und 3.6V liegen muss. Als Versorgungsspannung wird daher 3.3V verwendet. Pin 11 definiert das Groundpotential [22, S. 3].

Neben den Eingängen, über welche das Modul Signale empfängt, besitzt das Modul auch mit Pin 12 einen Analogeingang. Dieser wird standardmäßig dafür verwendet, um Batteriespannungswerte zu übermitteln, wenn das Modul nicht mit einer konstanten

Spannungsquelle verwendet wird. Das Modul sendet entsprechend des anliegenden Spannungspegels ein Signal, welches von der Funkfernbedienung empfangen wird und mittels einer LED auf der Fernbedienung visualisiert wird [22, S. 3].

Dabei gelten folgende Grenzwerte:

Spannung U	**Rückmeldung**	**Bedeutung**
U > 1.2V	LED aus	Ausreichende Versorgungsspannung
1.2V > U > 1.1V	LED blinkt	Abnehmende Versorgungsspannung
1.1V > U	LED blinkt schnell	Zu geringe Versorgungsspannung

Tabelle 4: Funktionen des Analogeingangs des Funkmoduls [22, S. 3]

Für das vorliegende System ist die Ausgabe eines Analogen Spannungswerts nicht relevant, weshalb der Pin Dauerhaft auf 3.3V gelegt wird, sodass ein durchgehend ausreichendes Spannungssignal erkannt wird und keine Rückmeldung an die Fernbedienung geschieht.

Zusätzlich zu den LEDs als visuelle Rückgabe gibt es auch die Möglichkeit auditive Rückmeldung vom Modul zu erhalten. Dafür ist am Empfänger der Ausgang für eine Piezoscheibe vorgesehen. Dieser Ausgang gibt vordefinierte Signale bei bestimmten Ereignissen aus, welchem dem Nutzer Rückmeldung über den Status des Systems geben [22, S. 3]. So gibt die Piezo-Scheibe Tonfolgen aus, wenn die Versorgungsspannung unter einen definierten Schwellwert sinkt. Auch die Aktivierung und Deaktivierung des Lernmodus wird mit je einer individuellen Tonfolge pro Modus ausgegeben. Auch das Einlernen und Löschen eines Senders wird mit je einer Tonfolge quittiert, genauso wie das Löschen der gesamten Speicherliste [22, S. 3]. Da diese Informationen bereits durch Auswertung der LED-Pins durch den Mikroprozessor erhalten werden und dieser die Informationen auf Display oder Netzwerk visuell zur Verfügung stellt, besteht für dieses System kein Bedarf diese Informationen zusätzlich noch über Tonfolgen auszugeben. Daher ist der Ausgang unbeschalten.

Ein weiterer LED-Ausgang, welcher auch vom Mikrocontroller ausgewertet wird, ist Pin 14. Hier handelt sich um eine visuelle Ausgabe von aufgetretenen Fehlern. Diese Fehler werden auch hier durch unterschiedlich viele Lichtimpulse zurückgegeben. Dabei wird unterschieden zwischen normalem Empfangsbetrieb und Aktiviertem Lösch-/Lernmodus:

Normaler Empfangs-betrieb	**Fehler**	**Darstellung**
	Senderversorgungsspannung gering	Blinkt 1x
	Senderversorgungsspannung kritisch	Blinkt 3x
Lern- / Lösch-modus	**Fehler**	**Darstellung**
	Eintrag konnte nicht gelöscht werden	Blinkt 2x
	Speicherliste ist voll	Blinkt 3x
	Sender wurde bereits eingelernt	Blinkt 2x

Tabelle 5: Fehlerauswertung des Funkmoduls [22, S. 3]

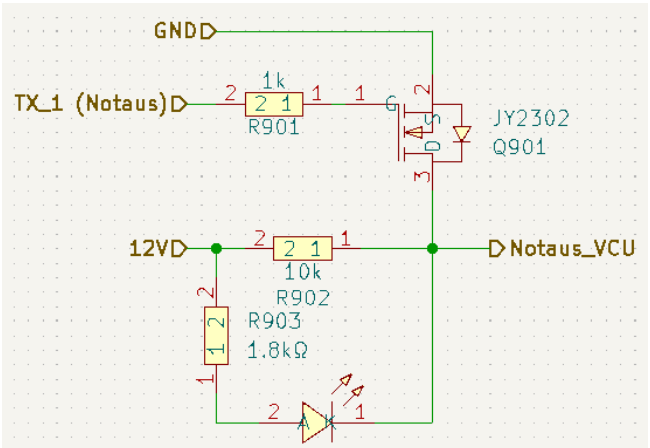
Zu Test und Debugging Zwecken ist auch hier weiterhin eine LED zur visuellen Rückmeldung vorgesehen, um die Fehler auch ohne Mikrocontrollerauswertung schnell und einfach identifizieren zu können. Auch hier wird ein gewünschter Strom von 5 mA vorgegeben, sodass sich auch hierwieder ein Vorwieder stand von $R_V = 220\Omega$ berechnen lässt. Auch hier handelt es sich wieder um ein Lowaktives Signal, sodass die LED zwischen 3.3V und Pin geschalten werden muss, um bei Aktivierung des Signalpins zu leuchten. Zusätzlich wird der Pin noch mit einem der Eingänge an der GPIO-Porterweiterung des Mikroprozessors verbunden, um die Fehler am ESP32 auswerten und für den Kunden darstellen zu können. Dafür ist auch der Widerstand R407 als Pull-Up vorgesehen, um bei dem Lowaktiven Signal einen definierten High-Pegel zu erzeugen.

Die Pins 15 und 16 sind nun noch als Anschlüsse für eine Antenne vorgesehen. Standardmäßig wird das Modul mit einer einfachen Drahtantenne ausgestattet. Dafür ist Pin 16 vorgesehen. Für die Verwendung einer abgesetzten Antenne lässt sich aber auch ein Koaxialkabel mit einer Impedanz von 50Ω anschließen, in diesem Fall wird Pin 15 für den Anschluss der Schirmung und Pin 16 für den Anschluss des Innenleiters verwendet. Für die Verwendung einer Drahtantenne gibt das Datenblatt eine Länge von 173mm vor [22, S. 3]. Diese berechnet sich für eine Monopolantenne nach folgender Formel:

$$l = \frac{\lambda}{4} \; mit \; Wellenlänge \; \lambda = \frac{c}{f} \qquad aus \; [12, S. 286]$$

$$l = \frac{\frac{299792.458 \frac{km}{s}}{433620kHz}}{4} = \frac{0{,}6913 \; m}{4} = 0{,}1729 \; m = 173 \; mm$$

VCU-Signale



Schaltplan 6: Erzeugung des Notaussignal für die VCU

Wie bereits weiter oben beschrieben sollen die ersten drei Kanäle des Funkmoduls auch ein Signal an die VCU senden. Zu diesem Zweck gibt es drei Transistorschaltungen auf der Platine, welche jeweils ein Signal erzeugen, welches anschließend über eine eigene Signalleitung an die VCU übertragen wird. Diese Signale sind allerdings nur noch aus Kompatibilitätsgründen für ältere Systeme vorgesehen, welche noch keine CAN-Verbindung zur Option1

vorgesehen haben. Einzige Ausnahme bildet der Notaus, welcher über die Funkfernbedienung aktiviert werden kann. Dieser muss aus Sicherheitsgründen Ausfallsicher realisiert werden, weshalb er nicht über den Mikrocontroller geführt werden darf. Dieses Signal ist weiterhin als eigene Leitung von der Option 1 zur VCU vorgesehen. Die Schaltung ist dafür ebenfalls drahtbruchsicher realisiert, wie man Schaltplan 6 entnehmen kann. Das bedeutet, dass der Notaus sowohl bei einem erkannten Funksignal als auch einer defekten Leitung als aktiviert erkannt wird, sodass ein Schaden am Kabel nicht zu einem Ausfall des Sicherheitssystems führt. Das wird erreicht, indem als Zustand „Notaus aktiviert“ der Stromlose zustand definiert wird. Im Idle Zustand fließt somit dauerhaft ein geringer Strom. Ein Abfall dieses Stromflusses führt zur Erkennung eines aktivierten Notaus. In der Schaltung wird dies durch einen Transistor als Low-Side Schalter realisiert. Das Signal Notaus_VCU, welches durch über den Stecker zur VCU geführt wird, wird durch den Widerstand R902 dauerhaft auf 12V gezogen. Somit wird im Zustand „Notaus nicht aktiv“ dauerhaft ein 12V Signal erkannt. Liegt nun am Eingang TX_1, welcher direkt mit dem Pin am Funkempfänger Modul verbunden ist, ein logischer High-Pegel an, Schaltet der Transistor und zieht den Ausgang gegen Ground. Am Steuergerät wird nun Kein Pegel mehr erkannt, was zum Auslösen der Notausprozedur führt. Das ist dasselbe Verhalten, als würde das Kabel beschädigt und den Kontakt öffnen, obwohl der Transistor nicht durchgeschalten ist. Zur Einfacheren Fehlersuche und zu Debugging Zwecken wurde zusätzlich noch eine LED hinzugefügt. Die LED soll leuchten sobald der Notaus durch den Transistor aktiviert wurde, der Ausgang also auf Groundpotential liegt. Die LED wird deshalb zwischen 12V und Ausgang geschalten. Zusätzlich wird ein Vorwiderstand hinzugefügt, um den Strom durch die LED auf 5mA zu begrenzen.

$$R_V = \frac{U - U_{LED}}{I} = \frac{12V - 2.2V}{5mA} = \frac{9.8V}{5mA} = 1.96k\Omega$$

Durch Verwendung und Verfügbarkeit stehen nur Widerstände der E12 Reihe zur Auswahl. Dort ist der nächste Widerstandswert 1.8 kΩ. Damit ergibt sich für den Strom

$$I = \frac{U - U_{LED}}{R_V} = \frac{12V - 2.2V}{1.8k\Omega} = \frac{9.8V}{1.8k\Omega} = 5{,}44mA$$

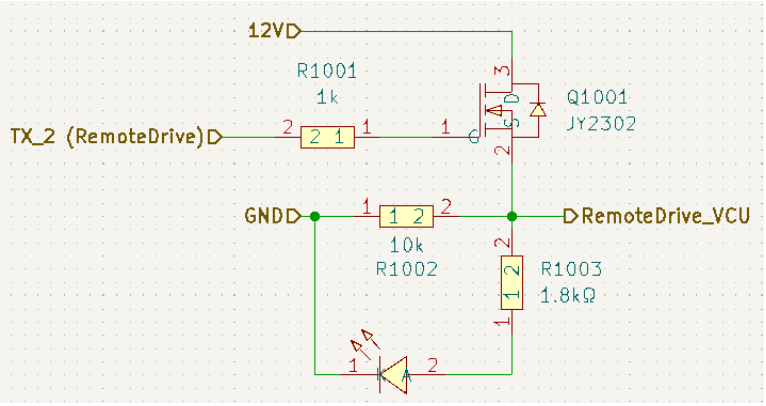
Bei dem Transistor handelt es sich um einen N-kanal MosFET, welcher als Schalter eingesetzt wird. Der Transistor schaltet, wenn $U_{GS} > U_{Th}$, wobei die Threshold Spannung bei 1V liegt. Dadurch, dass Source dauerhaft auf Ground liegt, gilt immer $U_G = 0V$, damit gilt immer

$$U_{GS} = U_G - U_S = U_G - 0 = U_G$$

somit schaltet der Transistor, wenn gilt

$$U_{GS} > U_{Th} = U_G > 1V$$

Diese Bedingung ist mit TX_1 auf 3.3V erfüllt, sodass der Ausgang auf Ground gezogen wird, sobald ein Signal auf Kanal 1 vom Funkempfänger erkannt wird. Der Gatewiderstand R901 ist dabei vorhanden, um die Gateströme zu begrenzen und das Funkmodul zu schützen.



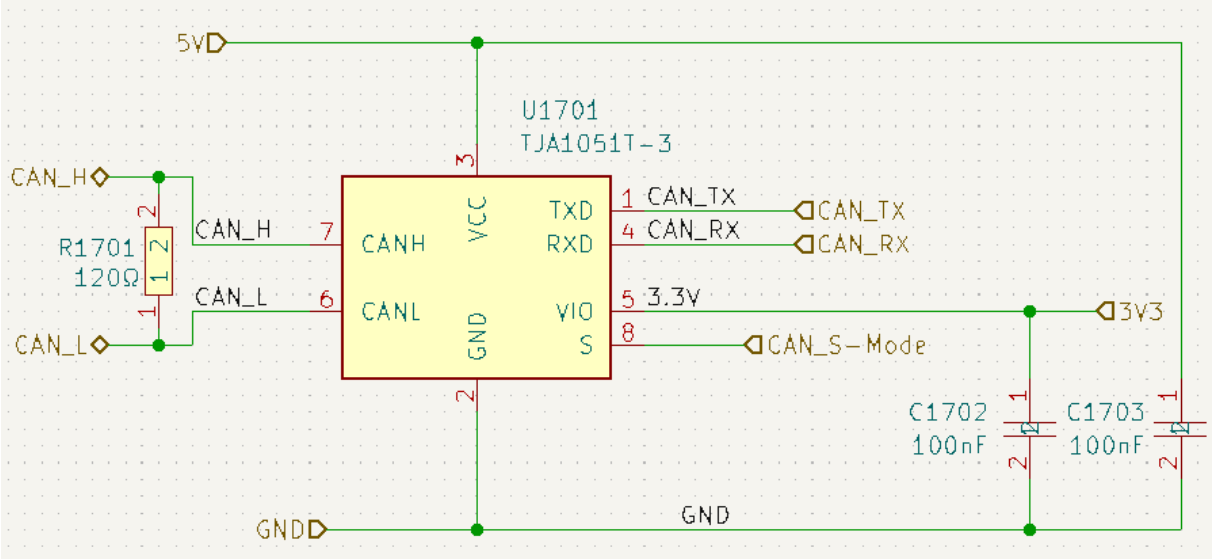
Schaltplan 7: Signalerzeugung des RemoteDrive Signals für die VCU

Sehr ähnlich zu der Transistorschaltung für das Notaussignal sind auch die Schaltungen für das RemoteDrive-Signal und die SOC-Abfrage, was man sehr gut anhand der Schaltpläne 6 und 7 entnehmen kann. Anders als das Notaussignal müssen diese Signale aber nicht drahtbruchsicher realisiert werden. Stattdessen liegt der Ausgang RemoteDrive_VCU standardmäßig auf Ground und wird beim Schalten des MosFET auf 12V gezogen. Der Widerstand R1002 dient hier also als Pull-Down Widerstand. Am Schalten des MosFET hat sich im Vergleich zur Schaltung für das Notaus Signal nichts verändert. Dadurch das der Transistor gedreht ist und Source weiterhin auf Ground liegt, schaltet der Transistor, sobald die 3.3V als High-Pegel am Gate des Transistors anliegen. Auch hier soll eine LED hinzugefügt werden, welche optisch den Pegel des Ausgangs anzeigen soll. Diese wurde hier ebenfalls so eingebaut, das sie leuchtet, sobald der Transistor geschalten hat. In diesem Fall liegt der Ausgang auf 12V, sodass die LED zwischen Ground und Ausgang platziert werden muss. Auch wurden als Vorwiderstand 1.8kΩ gewählt, da sich an den Bauteilen zur Notaus-Schaltung nichts verändert hat. Unterschiedlich zur Notaus-Schaltung ist jedoch, dass das Eingangssignal TX_2 nicht direkt von dem Funkempfänger kommt, sondern vorher vom Mikrocontroller ausgelesen wurde, welche jetzt den Transistor steuert. Das hat den Zweck, dass über Software steuerbar ist, ob die Signale am Ausgang gesetzt werden sollen, da diese nur als Backup für alte Systeme vorgesehen sind. Für die Schaltung zur Erzeugung des State of Charge Signals gelten dieselben Aussagen, da die Schaltungen zur Erzeugung von RemoteDrive-Signal und SOC-Abfrage identisch sind.

CAN-Transceiver

Statt die Signale jeweils über ein eines Datenkabel an die VCU zu übermitteln, sollen sie zukünftig als CAN-Signal übermittelt werden. Dafür benötigt der Mikroprozessor jedoch Zusatzbeschaltungen. Zwar besitzt der ESP32 intern einen TWAI-Controller, für die Kommunikation auf dem CAN-Bus ist jedoch zusätzlich ein CAN-Transceiver notwendig, welcher als Interface zwischen dem CAN-Controller und dem physikalischen CAN-Bus

fungiert [23, S. 1]. Dafür wird hier der TJA1051T-3 als Bauteil eingesetzt. Die Beschaltung des Bauteils wird unten gezeigt. Der CAN-Transceiver übernimmt dabei die Umsetzung der vom Controller erzeugten CAN-Frames auf physikalische Spannungspegel auf dem Bus. Andersherum wandelt er genauso erhaltene Buspegel in empfangene Nachrichten um, sodass die Frames schließlich vom CAN-Controller ausgewertet werden können. Damit das Funktioniert muss der Transceiver jedoch auch richtig beschalten werden. Das Bauteil erwartet eine Versorgungsspannung zwischen 4.5V und 5.5V, wodurch sich ein Spannungspegel von 5V hervorragend anbietet [23, S. 2]. Da das restliche System jedoch auf einem Logischen Spannungslevel von 3.3V arbeitet wird hier die Bauteilvariante „T-3“ eingesetzt. Diese besitzt einen weiteren Spannungseingang V_{IO}, mit welchem der die Versorgungsspannung für den I/O Level Adapter festgelegt wird, also die Definition des Logikpegels für die Eingangssignale. Dieser wird auf 3.3V gelegt und definiert für den CAN-Transceiver somit denselben Logikpegel wie für die restliche Schaltung [23, S. 6].



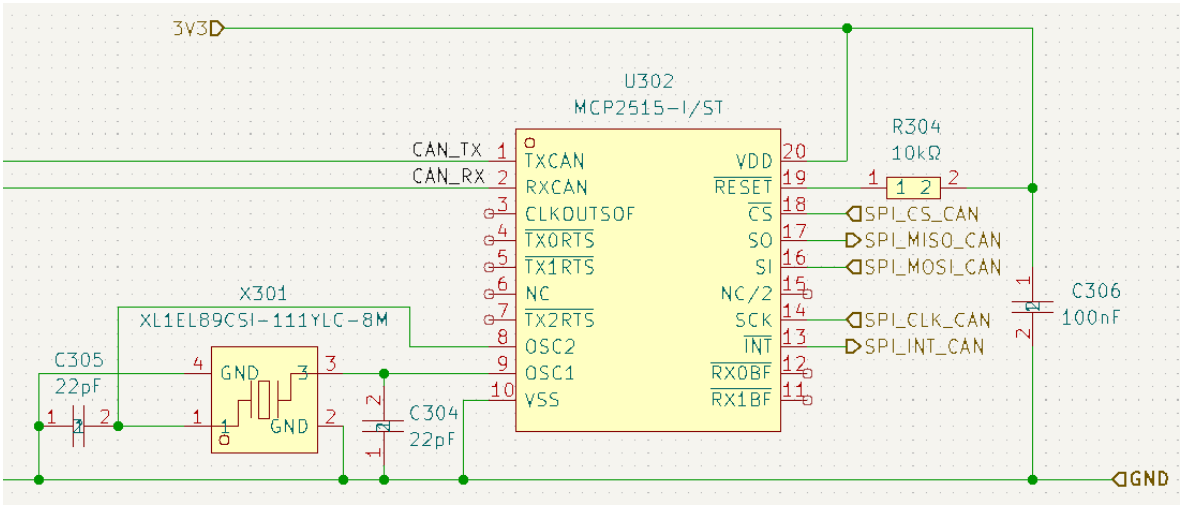
Schaltplan 8: Beschaltung des CAN-Transceivers

Die beiden Kondensatoren C1702 und C1703 sind hierbei wieder als Abblockkondensatoren vorgesehen. Sie sollen Spannungsspitzen abblocken und so das Bauteil schützen und gleichzeitig kurzfristige Spannungseinbrüche abdämpfen [23, S. 13]. Ein weiter Kontrolleingang ist der Pin S. Über diesen kann der Silent Mode für den CAN-Transceiver aktiviert werden. Im Silent-Mode ist der Sender deaktiviert, die Bus-Pins gehen über in einen rezessiven Zustand. Alle anderen IC-Funktionen, einschließlich des Empfängers, arbeiten weiterhin wie im Normal-Mode. Der Silent-Mode kann verwendet werden, um zu verhindern, dass ein fehlerhafter CAN-Controller die gesamte Netzwerkkommunikation stört [23, S. 5].

Die restlichen 4 Pins dienen der tatsächlichen Datenübertragung. Bei den Pins TXD und RXD handelt es sich um die serielle Kommunikation mit dem CAN-Controller, was sowohl Schaltplan 8 als auch 9 entnehmbar ist. Von diesem erhält der Transceiver fertige CAN-Frames und sendet seinerseits wieder empfangene Bussignale, welche er in einen CAN-Frame umgewandelt hat. Die tatsächliche Buskommunikation findet über die Pins CANH und CANL statt. Diese bilden die physikalischen CAN High und CAN Low Leitungen ab.

Der Transceiver wandelt die Nachrichten um und sendet diese als dominante und rezessive Pegel auf den Bus. Zwischen CANH und CANL findet sich noch ein Widerstand. Dieser dient als Terminierungswiderstand für den Bus. Er ist an beiden Enden des Busses notwendig, um Störungen zu minimieren und Reflexionen zu vermeiden. Im CAN-Protokoll ist die Impedanz des Busses auf 120Ω definiert, dadurch ergibt sich auch der Widerstandswert. Durch die Verwendung von 2 CAN-Bussen für Daten und den RFID-Reader findet sich diese Schaltung zweimal auf der Platine. Der ESP32 besitzt jedoch nur einen CAN-Controller, weshalb die Kommunikation auf dem zweiten CAN-Bus durch einen Externen Controller realisiert werden muss. Dafür wird die oben gezeigte und erklärte Schaltung um einen CAN-Controller erweitert. Als CAN-Controller kommt hierbei der MCP2515-I/ST zum Einsatz. Dieser bietet dieselbe Funktionalität wie der interne TWAI-Controller, mit dem Unterschied, dass die Kommunikation über SPI stattfindet. Die Daten, aus denen der Controller einen CAN-Frame bauen soll, werden dabei per SPI übermittelt und der Controller übernimmt dieselben Aufgaben wie der TWAI-Controller [24, S. 1]. Dazu gehören das Erstellen der CAN-Frames aus den übermittelten Daten, ID und den berechneten Kontrollfeldern wie CRC oder DLC und die relevanten Bits wie Acknowledge oder End of Frame. Zusätzlich übernimmt der Controller Aufgaben wie Arbitrierung und die damit verbundenen Steuerung, wann die Daten tatsächlich auf den Bus gesendet werden können. Auch die Auswertung der empfangenen Frames übernimmt der Controller. Diese Daten werden wiederum an den Microcontroller übermittelt, welcher die Abarbeitung der Daten im Programmcode durchführen kann [24, S. 1]. Die SPI-Kommunikation findet dabei über die Pins SO, SI und SCK statt. Bei SCK handelt es sich um das Clocksignal, welches grundlegend für die Synchrone Kommunikation zwischen Master und Slave ist. SO ist das „Slave Out“ Signal, welches im restlichen System als MISO verwendet wird. Über dieses Signal kommuniziert der Slave und sendet seine Nachrichten und Daten an den ESP32. Über den Pin SI kommuniziert der ESP32 mit dem CAN-Controller. Hier empfängt der Slave seine Daten, welche ihm vom Master über die MOSI-Signalleitung übermittelt werden. Damit die Kommunikation zustande kommt muss vorher der CAN-Controller über den ChipSelect ausgewählt werden. Das passiert über den CS-Pin am CAN-Controller. Wird dieser gegen Ground gezogen, wird der Controller informiert, dass er nun mit dem Master kommunizieren soll [24, S. 4]. Zusätzlich bietet der SPI-Controller im CAN-Controller eine Interruptfunktionalität über den Pin $\overline{INT}$. Über diesen Pin wird ein Signal an den ESP32 gesendet, sobald eine Nachricht empfangen wurde oder ein Fehler aufgetreten ist [24, S. 51]. Dieser Pin kann im Mikrocontroller als Auslöser für einen Interrupt definiert werden, sodass eingehende Nachrichten Zeitnah bearbeitet werden können. Versorgt wird der Controller mit 3.3V. Auch hier wird mit C306 ein Abblockkondensator zum Schutz des Bauteils und der Sicherstellung der Spannungsversorgung hinzugefügt. Der Reset-Pin

wird mit einem Pull-Up Widerstand auf High gezogen, sodass er bei jedem Neustart automatisch aktiviert und bei Shutdown resettet wird [24, S. 57].

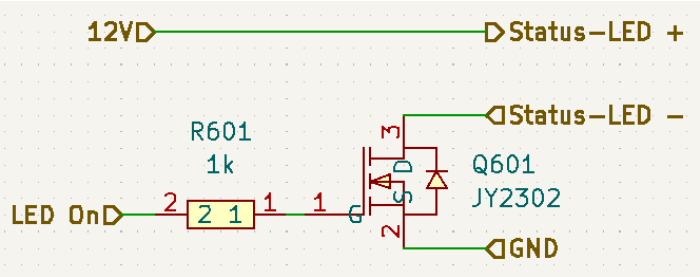


Schaltplan 9: Beschaltung des CAN-Controllers

Neben dem Interrupt Pin vom SPI-Controller gibt es auch Interrupt Pins, welche direkt vom CAN-Controller selbst gesteuert werden. So indizieren die Pins RX0BF und RX1BF eine empfangene Nachricht im Receive Buffer RXB0 oder RXB1. Auch diese können am ESP32 als Interruptquelle definiert werden. Da bereits die Interruptfunktionalität über SPI genutzt wird, wird die direkte Interrupt Möglichkeit aus dem Buffer des Controllers nicht benötigt und die Pins bleiben unbeschalten [24, S. 24]. Dasselbe gilt für die Pins TX0RTS, TX1RTS und TX2RTS. Diese Signale bieten die Möglichkeit ein Request To Send Signal direkt an den Buffer im Controller zu senden, woraufhin der Controller seine im Buffer gespeicherten Frames an den Transceiver übermitteln würde. Diese Funktionalität wird ebenfalls über Befehle via SPI realisiert, weswegen die Pins nicht benötigt werden [24, S. 16]. Als nächstes sollen die Funktionen der Pins OSC1 und OSC2 betrachtet werden. Das Datenblatt empfiehlt für eine Stabile Kommunikation einen externen Oszillator als System Clock für den CAN-Controller. OSC1 ist dabei der Input für den Timer, OSC2 der Output, welcher zum Oszillator zurückgeführt wird. Der ausgewählte Oszillator arbeitet dabei auf einer Frequenz von 8 MHz. Das Datenblatt gibt dabei für den Oszillator die Beschaltung vor mit welcher das Signal möglichst stabil wird und die Start-Up Zeit, welche vom Taktgeber abhängt, möglichst gering bleibt. Die empfohlenen Werte für einen 8MHz Oszillator sind dabei mit 22pF angeben, woraus sich auch die Kondensatoren in der Schaltung ergeben [24, S. 55f]. Das Clock Signal kann zusätzlich über den Ausgang CLKOUTSOF abgegriffen werden. Dieser Ausgang stellt ein Clocksignal zur Verfügung, welches mittels eines Prescalers verändert werden kann, um als Software Timer für genutzt werden zu können. Die Pins TX und RX übernehmen bei dem Controller nun die tatsächliche Kommunikation mit dem Transceiver. Hier werden die tatsächlichen Frames an den CAN-transceiver übermittelt, welcher anschließend die Kommunikation auf dem

CAN-Bus übernimmt. TX dient dabei dazu die Frames an den Transceiver zu übermitteln, RX empfängt die Daten des Transceivers.

LED-Treiber

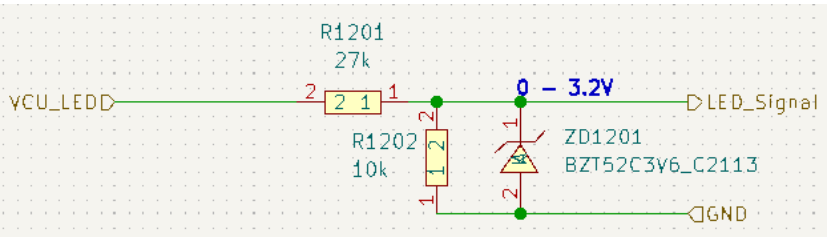


Schaltplan 10: Treiberschaltung zur Ansteuerung der Status-LED

Da die Pins des Mikrocontrollers nicht genug Strom liefern und das LED-Modul einen Spannungspegel von 12 V benötigt, wird eine Transistorschaltung als LED-Treiber notwendig, welche in Schaltplan 10 dargestellt ist. Der Treiber wird hier als Low-Side Switch realisiert, welcher den negativen Anschluss der LEDs gegen Ground verbindet. Als Schalter wird hier ein MosFET eingesetzt, welcher vom ESP32 angesteuert wird, wenn die LED leuchten soll. Die positive Seite der LEDs liegt dabei dauerhaft auf 12V, die Negative Seite der LEDs wird auf Ground gezogen, sobald der Transistor durchschaltet. Der Gatewiderstand ist vorhanden, um den Pin am Mikrocontroller vor Stromspitzen zu schützen. Auch hier gilt dieselbe Rechnung wie bei den Transistorschaltungen für die VCU-Signale. Dadurch, dass Source dauerhaft auf Ground liegt, schaltet der Transistor sobald $U_G > 1V$ gilt. Das ist der Fall, sobald der Pin am ESP32 das Gate auf 3.3V zieht.

LED-Signal Auswertung

Neben den Signalen, welche für veraltete Systeme noch per Signalleitung an die VCU übermittelt werden, übermitteln diese Karts auch die LED-Signale von der VCU noch mittels einer eigenen Signalleitung. Diese Signale müssen ausgewertet werden, um an die LED-Treiber Schaltung weitergegeben zu werden. Da es sich um ein digitales 12V Signal handelt, reicht es das Signal auf maximal 3.3V zu normieren, sodass das Signal direkt an einen der Pins des ESP32 angeschlossen werden kann. Diese Schaltung wird in Schaltplan 11 gezeigt. Das Signal vom Steuergerät kommt über den Eingang VCU_LED direkt vom Stecker. Anschließend wird das Signal von 12V mittels eines Spannungsteilers auf maximal 3.3V am Pin normiert. Die Spannung am Ausgang LED_Signal, welcher mit dem Pin der GPIO-Erweiterung vom Mikrocontroller verbunden ist, lässt sich folgendermaßen berechnen:



Schaltplan 11: Schaltung zur Auswertung des LED-Signals von der VCU

$$U_{ESP} = U * \frac{R_2}{R_1 + R_2}$$

Daraus ergeben sich die Formel für R_2

$$R_1 = R_2 * \left(\frac{U}{U_{ESP}} - 1\right) = R_2 * \left(\frac{12V}{3.3V} - 1\right) = R_2 * 2.63$$

Für $R_2 = 10k\Omega$ ergibt sich damit für R_1

$$R_1 = 2.63 * 10k\Omega = 26.3k\Omega$$

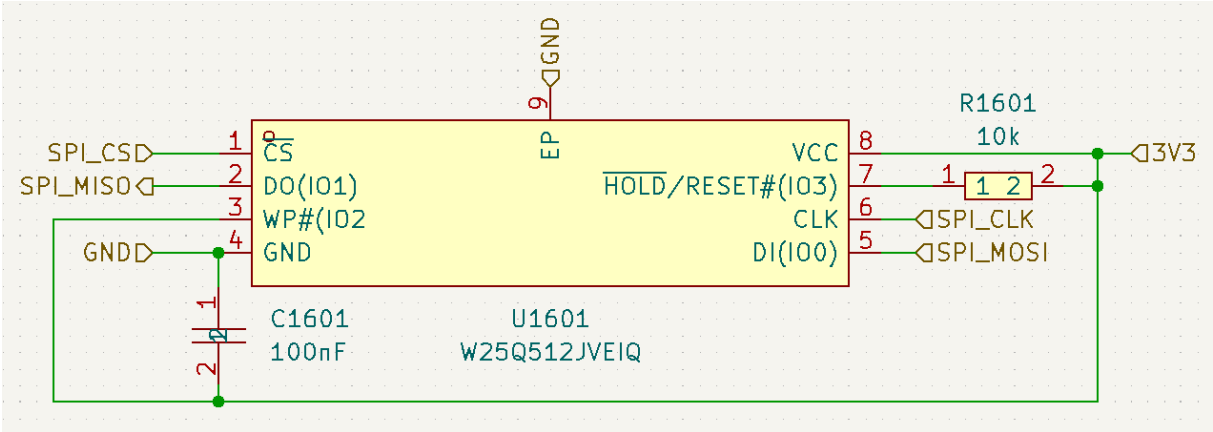
Durch die E-Reihe ergibt sich damit der nächste Wert als 27kΩ. Mit 27kΩ beträgt die maximale Spannung am ESP32-Pin

$$U_{ESP} = 12V * \frac{R_2}{R_1 + R_2} = 12V * \frac{10k\Omega}{27k\Omega + 10k\Omega} = 12V * 0.27 = 3.24V$$

Damit erkennt der ESP32 an seinem Pin 0V, wenn kein Signal gesendet wird und 3.24V als High Pegel.
Zum Schutz des Mikrocontrollers ist zusätzlich noch eine Zenerdiode zwischen den Pin und Ground platziert. Die Zenerdiode hat eine Durchbruchspannung bei 3.3V. Das bedeutet, sobald die Spannung am Mikrocontroller 3.3V überschreitet, wird die Diode leitend und hält den Pin bei 3.3V und schützt den Controller so vor Schäden durch Überspannung.

SPI-Flash

Um die Funktionalität eines persistenten Speichers für Logdaten und Einstellungen zu erreichen, ist auf der Platine ein Flash Speicher vorgesehen. Dieser Speicher ist per SPI-Kommunikation adressierbar und beschreibbar. Der Speicher ist dabei persistent, das bedeutet das er seinen Inhalt behält, selbst wenn er nicht mehr bestromt wird. Die Daten sind auch beim nächsten Start-up erhalten. Die Größe des Speichers beträgt 4 MB, für Speicherung von Textdateien und einzelnen Integerwerten ist das völlig ausreichend. Wie in Schaltplan 12 zu sehen, wird der Speicher mit 3.3V versorgt. Wie bei allen anderen Bauteilen ist auch hier der Kondensator C1601 als Abblockkondensator vorgesehen. Zusätzlich zum Groundpin GND besitzt der SPI-Flash noch ein großes Lötpad EP unter dem Bauteil, welches ebenfalls auf Ground gezogen wird [25, S. 7].



Schaltplan 12: Beschaltung des SPI-Flashs

Die SPI-Kommunikation findet bei dem Speicher über die Pins DO und DI statt. DO steht dabei für „Data Out“ und wird daher mit dem MISO-Signal verbunden, dass der Speicher darüber Daten an den Mikroprozessor senden kann. Der Daten empfang über die MOSI-Signalleitung findet am „Data In“ (DI) Pin statt. Hierüber empfängt der Flash befehle und Daten, welche er persistent speichern soll. Als Taktgeber für die SPI-Kommunikation dient der CLK-Pin. Hier erhält der Chip das Clock-Signal vom SPI-Bus. Das letzte SPI-Signal ist der Chip Select für den Speicher. Über diesen wird dem Speicher mitgeteilt, dass nun mit ihm kommuniziert werden soll [25, S. 7]. Zusätzlich zu den Pins für die SPI-Kommunikation besitzt der Chip noch zwei weitere externe Steuersignale. Eines davon ist Pin 7 für die HOLD-Funktionalität. Der HOLD-Pin ermöglicht es, das Gerät anzuhalten, während es aktiv ausgewählt ist. Wenn HOLD auf Low gesetzt wird, während Chip Select auf Low ist, dass Gerät also gezielt angesprochen wird, wird der Data Out Pin deaktiviert und Signale an den DI- und CLK-Pins werden ignoriert. Sobald der Pegel am HOLD-Pin wieder auf High gesetzt wird, geht der Chip wieder in normalen Betriebsmodus über und nimmt seine Kommunikation wieder auf [25, S. 10]. Das ist vor allem dann nützlich, wenn sich mehrere Geräte dieselben SPI-Signale teilen, zum Beispiel bei mehreren parallelen Speicherchips. Da dies hier nicht der Fall ist, wird diese Funktionalität nicht benötigt. Weil es sich bei dem Signal um ein lowaktives handelt, wird der Pin mittels eine s Pull-Up Widerstands dauerhaft auf 3.3V gezogen, um ungewollte Störungen durch einen undefinierten Pegel am Pin zu vermeiden. Ein weiteres Steuersignal ist WriteProtect (WP) an Pin 3. Über dieses Signal lässt sich das Schreiben in das Statusregister des Chips verhindern. Zusammen mit Steuerbits im Chip lässt sich damit der gesamte Speicher oder einzelne Bereiche gezielt vor Überschreiben schützen. Da es für das System keine besonders geschützten Speicherbereiche benötigt, wird auch diese Funktionalität nicht genutzt. Der Pin wird dafür dauerhaft auf High gezogen, da es sich auch hier um ein lowaktives Signal handelt [25, S. 10].

Power Supply

Wichtig für das System ist die Erzeugung der verschiedenen Spannungslevel. Die Versorgung der Platine findet mit den 48V direkt aus der Batterie des Karts statt. Mit diesem Pegel kann jedoch kein einziges Bauteil auf der Platine arbeiten, dies benötigen 12V, 5V und hauptsächlich 3.3V. Alle diese Spannungspegel müssen daher auf der Platine selbst erzeugt werden. Besonderer Fokus liegt dabei darauf möglichst stabile Versorgungspannungen zu erzeugen, um vor allem Probleme in der Kommunikation zwischen den Komponenten zu verhindern. Auch die Auswahl der Spannungswandler ist wichtig, da diese nur einen bestimmten Strom liefern können. Der Stromverbrauch ist aber bereits durch die Bauteile definiert. Zudem muss darauf geachtet werden Spannungswandler nicht unnötig zu kaskadieren. Durch Kaskadierungen sinkt der Wirkungsgrad der Schaltungen enorm, wodurch die gleichzeitig die Verlustleistung zunimmt. Das hat deutlich mehr Abwärme zur Folge welche Probleme auf der Platine schaffen kann. Als erstes soll daher der Strom bestimmt werden, den die jeweiligen Spannungswandler treiben können müssen. Dazu werden die Maximalströme der Bauteile aus dem Datenblatt betrachtet, um auch den Worst Case abdecken zu können.

Bauteil	**Versorgungsspannung**	**Stromverbrauch**	
CAN-Controller	3.3V	10 mA	[24, S. 72]
Funkreceiver	3.3V	18.2 mA	[22, S. 3]
GPIO Port Erweiterung	3.3V	125 mA	[21, S. 4]
Mikrocontroller	3.3V	250 mA	[20, S. 52]
USB to Serial Wandler	3.3V	12 mA	[26, S. 8]
Display	3.3V	33 mA	[27, S. 183]
SPI-Flash	3.3V	40 mA	[25, S. 82]
RFID-Reader	3.3V	26 mA	[18, S. 2]
Gesamt		**515 mA**	

Tabelle 6: Stromverbrauch der mit 3.3V versorgten Bauteile

Alle Bauteile, welche mit 3.3V versorgt werden, benötigen im Worst Case 515 mA. Um auf bereits vorhandene Lagerbestände zurückzugreifen und nicht viele verschiedene Bauteile beschaffen zu müssen, sollen möglichst Bauteile eingesetzt werden, welche bereits in anderen Systemen in der Firma zum Einsatz kamen. Für einen Spannungswandler zu 3.3V gibt es daher zwei mögliche Optionen:

1. **Bauteil P7803-500:**
   Bei dem Bauteil handelt es sich um einen DCDC-Wandler, welcher direkt aus dem 48V Eingangssignal ein 3.3V Signal erzeugt. Das hätte den Vorteil, dass unnötige Kaskadierungen vermieden werden, wodurch der Wirkungsgrad steigt. Außerdem ist das Spannungssignal stabiler, da nicht die Störeinflüsse mehrerer Spannungswandler weitergegeben werden können. Jedoch ist dieses Bauteil nicht

in der Lage den benötigten Strom zu liefern. Der Ausgangsstrom beträgt maximal 500mA, sodass das Bauteil den Maximalstrom nicht liefern kann, was auf Dauer zur Zerstörung des Moduls oder der Schaltung führt [28, S. 1].

2. **Bauteil AN_SY8089A:**
   bei dem Bauteil handelt es sich um einen Spannungswandler, welcher die 3.3V aus einem 5V Eingangssignal generiert. Dieses Bauteil hat den Vorteil, dass die Ausgangsspannung meist stabiler und weniger Störanfällig ist, je kleiner der Spannungsunterschied zwischen Eingang und Ausgang ist. Nachteile sind die durch die Kaskadierung von 48->5V und von 5V -> 3.3V bedingten höheren Leistungsverluste und Wärmeabgabe. Jedoch ist dieses Bauteil in der Lage bis zu 2A kontinuierlich an Strom zu liefern. Deshalb wird dieses Bauteil in der Schaltung verwendet [29, S. 1].

Bauteil	**Versorgungsspannung**	**Stromverbrauch**
CAN-Transceiver 1	5V	50mA [23, S. 8]
CAN-Transceiver 2	5V	50mA [23, S. 8]
3.3V Spannungswandler	5V	365mA [29, S. 5]
Gesamt		**465mA**

Tabelle 7: Stromverbrauch der mit 5V versorgten Bauteile

Alle Bauteile, die mit 5V versorgt werden, benötigen im Worst Case 470 mA. Der Stromverbrauch des 3.3V Spannungswandlers berechnet sich dabei aus dem Worst Case Stromverbrauch der 3.3V Versorgungsspannung.

$$P_{Out} = U_{Out} * I_{out} = 3.3V * 0.515A = 1.7W$$
$$P_{In} = \frac{P_{out}}{\eta} = \frac{1.7W}{93\%} = 1.83W$$
$$I_{In} = \frac{P_{In}}{U_{In}} = \frac{1.83W}{5V} = 365mA$$

Der Wirkungsgrad η ist dem Datenblatt bei einem Laststrom von 515mA entnommen. Damit ergibt sich ein maximaler Eingangsstrom von 365 mA für den maximalen Stromverbrauch aller 3.3V.
Auch für die Spannungserzeugung des 5V Signals gibt es wieder unterschiedliche Bauteile, welche zur Auswahl stehen.

1. **N7805-500**
   Bei diesem Bauteil handelt es sich um einen DCDC, welcher aus 12V ein 5V Signal erzeugt. Die Verwendung dieses Bauteils würde bedeuten, dass zuerst aus 48V ein 12V Signal generiert werden muss, aus welchem dann wiederum ein 5V Signal erzeugt wird. Dieses würde wiederum Grundlage zur Generierung des 3.3V Signals sein. Diese Kaskadierung birgt viele Nachteile und sollte deshalb vermieden werden [30].

2. **P7805-500**
   Dieses Bauteil erzeugt die 5V direkt aus dem 48V Eingangssignal. Dadurch wird die Kaskadierung aufgebrochen. Zudem ist der Ausgangsstrom mit 500mA ausreichend für die Versorgung aller Bauteile mit 5V Spannungspegel. Die Wahl fällt daher auf dieses Bauteil [28].

Bauteil	**Versorgungsspannung**	**Stromverbrauch**	
RFID-Reader	12V	120 mA	[18, S. 2]
Status-LED	12V	100 mA	[gemessen]
Gesamt		**220 mA**	

Tabelle 8: Stromverbrauch der mit 12V versorgten Bauteile

Alle Bauteile, die mit 5V versorgt werden, benötigen im Worst Case 470 mA. Für den 48V zu 12V DCDC steht nur ein Bauteil zur Verfügung, welches in den Lagerbeständen vorrätig ist. Dieses liefert einen Ausgangsstrom von $I_{Out} = 500mA$. Damit ist die Stromstärke deutlich ausreichend, um alle Bauteile zu versorgen, welche mit der 12V Spannungsversorgung gespeist werden. Hier kann man Zusätzlich noch erkennen, warum die Wahl für den 5V DCDC auf das Bauteil fällt, welches direkt 48V als Eingangsspannung nutzt. Würde zusätzlich noch der 12V->5V DCDC versorgt werden müssen, betrüge die mindestens benötigt Stromstärke

$$I = I_{Ges} + I_{DCDC} = 220\ mA + 360\ mA = 580\ mA$$

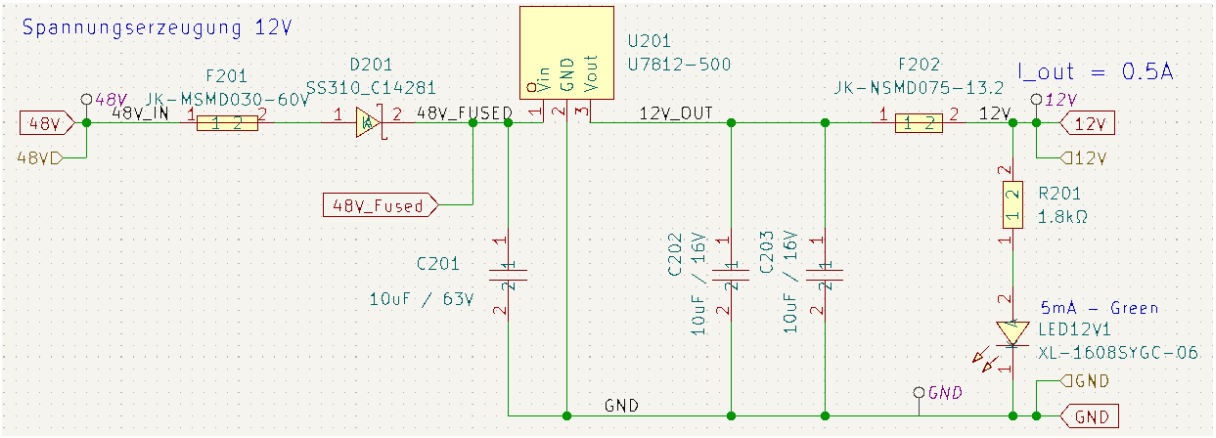
Das würde den Ausgangsstrom des DCDC um 80 mA überschreiten. Die daraus hervorgehende Schaltung ist in Schaltplan 13 dargestellt.
Als zusätzlichen Schutz für die Schaltung ist zusätzlich noch eine Sicherung hinter dem 12V Pfad eingebaut. Diese Sicherung bleibt leitend bis 750 mA und löst bei 1.5A garantiert aus [31, S. 2]. Diese Sicherung ist dafür da im Falle eines Bauteilschadens die folgenden Bauteile für dem hohen Kurzschlussstrom des DCDC zu schützen. Für den Eingang wird ebenfalls eine Sicherung vorgesehen, zur Auslegung dieser Sicherung muss jedoch der Eingangsstrom bekannt sein. Dieser setzt sich zusammen aus dem Eingangsstrom der beiden DCDC-bauteile für 12V und 5V.

$$I_{5V} = \frac{P_{In}}{U_{In}} = \frac{\frac{P_{out}}{\eta}}{U_{In}} = \frac{\frac{2.5W}{82\%}}{48V} = 63.5mA$$

$$I_{12V} = \frac{P_{In}}{U_{In}} = \frac{\frac{P_{out}}{\eta}}{U_{In}} = \frac{\frac{6W}{87\%}}{48V} = 143.6mA$$

$$I_{Ges} = I_{5V} + I_{12V} = 63.5mA + 143.6mA = 207.1mA$$

Die ausgewählte Sicherung bleibt leitend bis zu einem Strom von 300mA. Garantiert auslösen wird sie lt. Datenblatt ab einem Strom von 650 mA [32, S. 2]. Die Sicherung eignet sich damit sehr gut, um als Schutz für die Schaltung bei Kurzschlüssen zu fungieren.

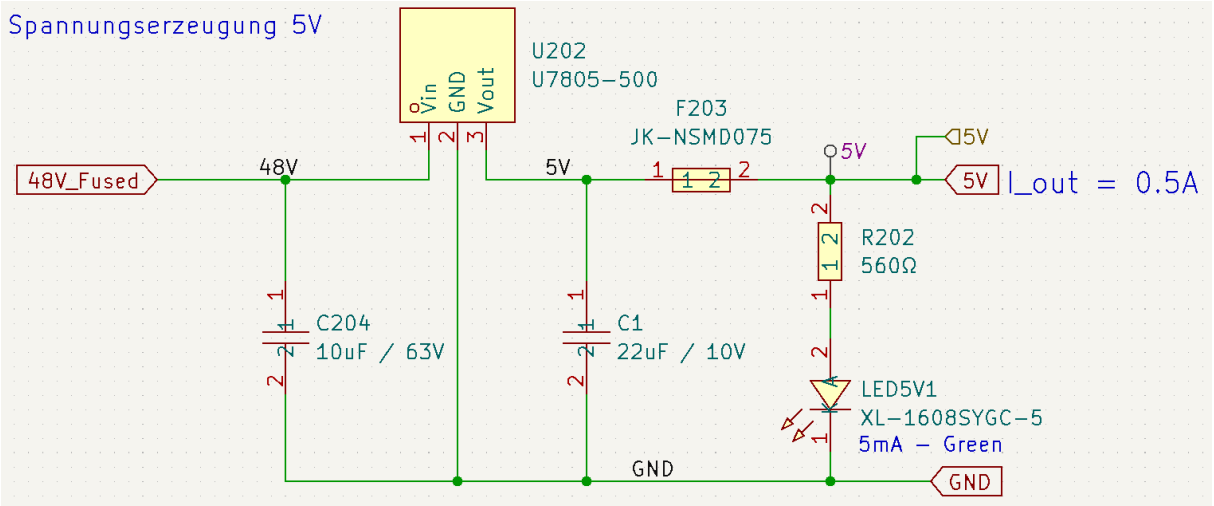


Schaltplan 13: Erzeugung der 12V Versorgungsspannung

Zusätzlich wird als Schutzbauteil noch eine Schottky Diode hinzugefügt. Diese hat die Aufgabe die Schaltung bei Verpolung zu schützen. Wird die Schottky Diode in Sperrrichtung betrieben, wird sie zerstört, wodurch keine Spannung mehr anliegt, welche die restliche Schaltung beschädigen kann. Die Kondensatoren, welche an den Ein und Ausgängen der Spannungswandler platziert sind, dienen der Glättung der Spannungspegel und sollen Spannungsspitzen dämpfen, um Schäden am Bauteil zu verhindern. Die Werte der Kondensatoren sind durch das Datenblatt festgelegt [28, S. 4]. Zur einfacheren Fehlersuche wird dem Ausgang noch eine LED hinzugefügt. Diese soll leuchten, wenn der Spannungswandler einen Spannungspegel an seinem Ausgang bereitstellt. Der Vorwiderstand berechnet sich dabei entsprechend dieser Formel:

$$R_V = \frac{U - U_{LED}}{I_{LED}} = \frac{12V - 2.2V}{5mA} = \frac{9.8V}{5mA} = 1.96k\Omega$$

Gemäß der verwendeten e-12 Widerstandsreihe ist der nächstmögliche Wert 1.8kΩ. Die Schaltung für den Spannungswandler von 48V zu 5V verhält sich genau wie die vorherige Schaltung, mit der Ausnahme, dass sich die Werte des einen Kondensators ändern, welche im Datenblatt spezifiziert werden [28, S. 4].

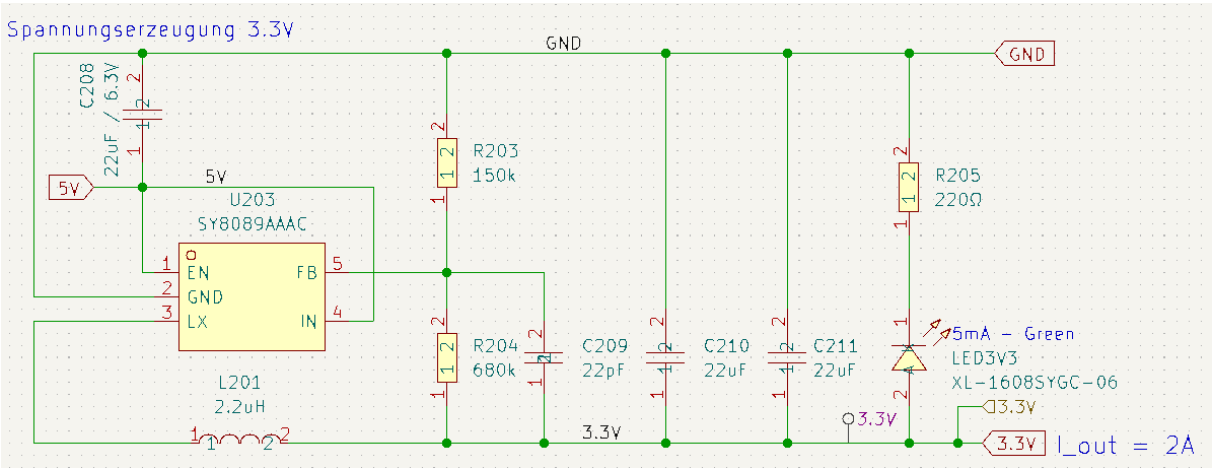


Schaltplan 14: Erzeugung der 5V Spannungsversorgung

Damit ist die Schaltung auf Schaltplan 14 gemäß dem Datenblatt aufgebaut. Auch die Parameter der Sicherung bleiben dieselben, nur das dieses Bauteil für den Einsatz bis 6V zugelassen ist, während die vorherige Sicherung im Spannungsausgang bis 13.6V eingesetzt werden durfte. Die Ströme, bei welchen die Sicherung auslöst, bleiben identisch. Zusätzlich ändert sich der Wert des Vorwiderstands der LED.

$$R_V = \frac{U - U_{LED}}{I_{LED}} = \frac{5V - 2.2V}{5mA} = \frac{2.8V}{5mA} = 560\Omega$$

Als letzte Schaltung muss nun noch der Spannungsregler für das 3.3V Signal betrachtet werden. Dieser erzeugt aus dem 5V Eingangssignal ein 3.3V Ausgangssignal. Dieser Spannungsregler eignet sich für mehrere Ausgangsspannungen im Bereich von 1.2V bis 3.3V. Deshalb muss der Chip korrekt beschalten werden, um den korrekten Output zu erzeugen. Die Pins IN und En sind dabei die, welche das Bauteil versorgen. In ist der Spannungseingang, in unserem Fall, dass 5V Signal. Dieses wird auch mit dem EN-Pin verbunden. Dieser aktiviert das Bauteil. Die Festlegung der Ausgangsspannung wird über die Pins FB und LX erreicht. Über FB wird eine Spannung ausgegeben, wobei über den Spannungsteiler die gewünschte Ausgangsspannung erzeugt wird. Über eine Spule wird dieser Spannungswert wieder zum Bauteil zurückgeführt, welches nun auf diese Spannung programmiert wird [29, S. 1].



Schaltplan 15: Erzeugung der 3.3V Spannungsversorgung

Die Ausgangsspannung lässt sich dabei mit der Formel

$$V_{Out} = 0.6 * \left(1 + \frac{R_1}{R_2}\right) \qquad [29, S. 7]$$

berechnen. Damit gilt für R_1:

$$R_1 = \left(\frac{V_{out}}{0.6} - 1\right) * R_2$$

$$R_1 = \left(\frac{3.3V}{0.6} - 1\right) * R_2 = 4.5 * R_2$$

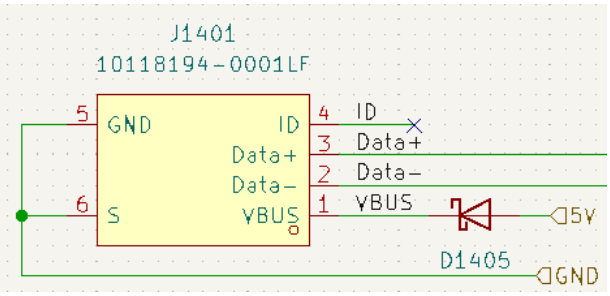
Mit den Werten $R_1 = 680k\Omega$ und $R_2 = 150k\Omega$ ergibt sich eine Abweichung von 0.7% für die Spannung.

Sowohl die Bauteilwerte der Spule als auch die der Kondensatoren sind wiederum vom Datenblatt vorgegeben. Auch hier soll wieder eine LED vorgesehen werden, welche einen Spannungspegel am Ausgang visuell anzeigt. Für den Vorwiderstand lässt sich folgender Wert berechnen:

$$R_V = \frac{U - U_{LED}}{I_{LED}} = \frac{3.3V - 2.2V}{5mA} = \frac{1.1V}{5mA} = 220\Omega$$

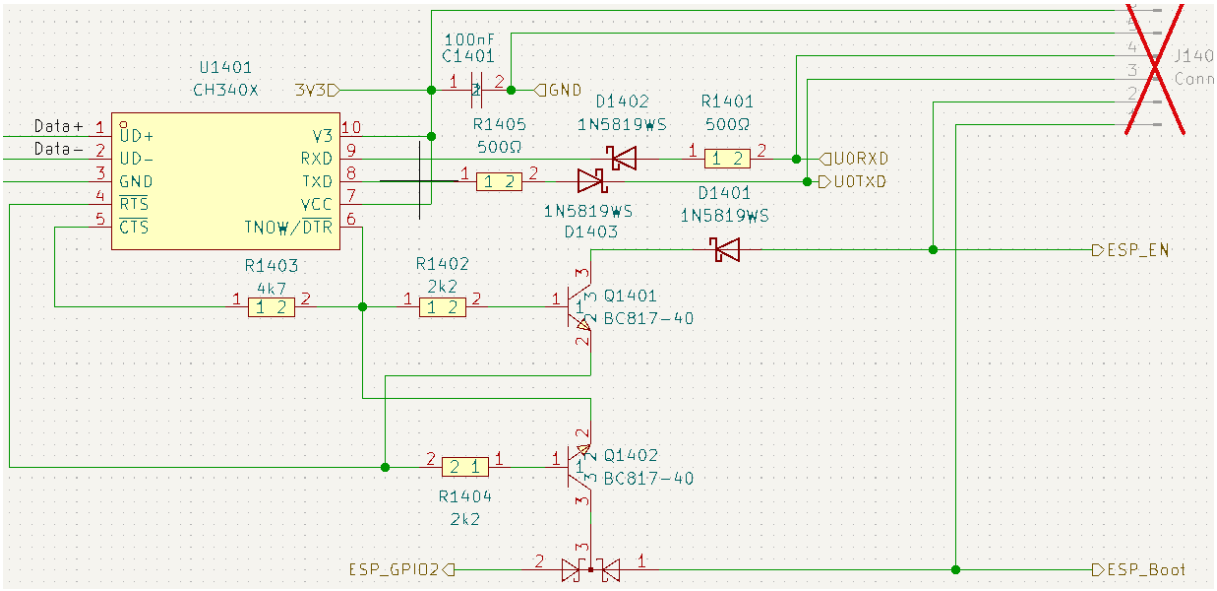
USB-Schaltung

Um diese ganzen Funktionalitäten nutzen zu können bedarf es verschiedener Schnittstellen. Eine dieser Schnittstellen eine Micro-USB Buchse, über welche es möglich sein soll, den Chip mit neuer Software zu bespielen. Da der ESP32 mit USB nicht direkt arbeiten kann, wird noch ein Wandler benötigt, welcher die USB-Kommunikation in eine UART-Kommunikation umwandelt, über welche der Mikrocontroller kommunizieren kann. Die in Schaltplan 16 gezeigte USB-Buchse besitzt 6 Pins. Diese Pins übertragen alle Signale der Physischen USB-Kommunikation. Die Pins GND und S liegen dabei auf Groundpotential. Bei Pin S handelt es sich um den Schirm das Kabels, welches die Signale vor Störungen schützen soll. Hier ist es wichtig, den Schirm auf dasselbe Potenzial wie den Systemground zu legen, da sonst Störungen unter Umständen sogar verstärkt werden können. Über VBUS lässt die Busspannung abgreifen. Diese beträgt bei USB 5V, da der Host immer die Spannungsversorgung für alle Devices übernimmt. Die Spannung wird hier abgegriffen und auf den 5V Versorgungsspannungspfad geführt, um über USB die Platine bestromen zu können. Normale USB-Ports eines PCs können bis zu maximal 500mA an Strom liefern, das ist vollkommen ausreichend, um die CAN-Transceiver und den Spannungswandler auf 3.3V zu betreiben, da der Spannungswandler von 48V auf 5V auch nicht mehr als 500mA liefern kann. So muss die Platine für Testzwecke nicht immer an eine Externe Spannungsquelle angeschlossen sein, welche die 48V Eingangsspannung liefern kann, sondern kann auch mit einem normalen USB-Netzteil betrieben werden oder beim Flashen direkt vom PC aus. Zwar fallen dabei die 12V Funktionalitäten aus, da diese nicht mit Spannung versorgt werden können, dazu gehören jedoch nur der RFID-Reader, die LED und die VCU-Signale, welche für einen grundlegenden Systemtest vernachlässigbar sind, da es auf der Platine dafür optische Indikation über LEDs gibt, ob die Schaltungen funktionieren. Die Schaltung wird dabei von einer TVS-Diode geschützt. Diese Verhindert Spannungsspitzen, welche zum Beispiel ESD-bedingt auftreten können und schützt so die gesamte Schaltung vor Zerstörung. Die tatsächliche Datenübertragung der Bits findet über die Pins Data+ und Data- statt. Die Bits werden dabei über Zwei Signalpegel übertragen, wobei D+ und D- immer 0V oder 3.3V



Schaltplan 16: Pinbelegung der USB-Buchse

führen. Für den Ruhezustand liegt D+ auf 3.3V, D- auf 0V. Wird nun eine logische 1 übertragen, bleiben die Pegel unverändert bei einer logischen 0 tauschen die beiden Leitungen ihren Zustand. Dieser Wechsel findet bei jeder übertragenen logischen 0 statt. Die Auswertung der Pegelveränderungen übernimmt dabei der USB-To-Serial Wandler, wie er in Schalplan 17 dargestellt ist. Im vorliegenden System übernimmt das der CH340X [26]. Der USB-Wandler hat dabei 2 Pins für Spannungsversorgung. Über VCC wird der Chip mit Spannung versorgt und über V3 wird die Spannung definiert, mit welcher der Chip versorgt wird. Im Vorliegenden System soll der Chip mit 3.3V versorgt werden, da auch alle restlichen Komponenten auf 3.3V arbeiten. Damit das Bauteil korrekt arbeitet muss dafür der Pin V3 ebenfalls auf 3.3V gezogen werden. Würde der Chip mit 5V versorgt, würde der Pin V3 über einen Abblockkondensator gegen Ground verbunden werden [26, S. 5f]. Die beiden Pins DU+ und DU- sind die beiden Pins, welche mit der USB-Buchse verbunden sind. Über diese erhält der Wandler die Bits, welche über die USB-Leitung übermittelt werden und anschließend umgewandelt werden, um über eine UART-Schnittstelle an den ESP32 gesendet zu werden. Die Übermittlung an den ESP32 geschieht über die Pins RXD und TXD. Über diese beiden Pins findet die asynchrone UART-Kommunikation statt [26, S. 4]. Die beiden Widerstände sind dabei zum Schutz der Bauteile vorgesehen. So sollen diese im Falle eines Kurzschlusses den Strom begrenzen und so den ESP32 und den USB-Wandler schützen. Gleichzeitig dienen die Widerstände zur Dämpfung von Signalflanken, wodurch Störungen vermindert werden soll. Das ist vor allem bei langen Leitungen relevant. Bei den beiden Dioden handelt es sich um TVS-Dioden. Diese Schützt sowohl den Chip als auch den Mikrocontroller vor Überspannung, indem die Diode bei Überspannung durchbricht und die Spannungen auf einen Maximalwert begrenzt. Die TVS-Diode hat dabei gegenüber einer Zenerdiode den Vorteil, dass sie deutlich schneller arbeitet und damit kürzere und höhere Spannungsspitzen abfangen kann. Diese treten vor allem durch ESD-Probleme auf, welche über das Kabel auftreten können.



Schaltplan 17: Beschaltung des USB-To-Serial Wandlers

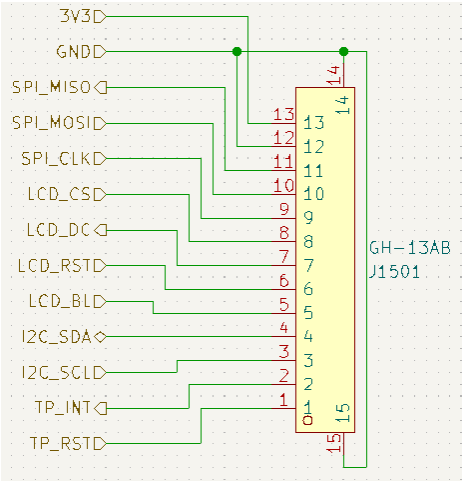
Damit die Software, welche über die USB-Schnittstelle übermittelt wird, auch als neue Software für den Mikrocontroller genutzt wird, muss dieser in den Entsprechenden Bootmodus gebracht werden. Dafür müssen GPIO 0 und 2 auf den korrekten Pegel gezogen werden. Die dafür vorgegebenen Pegel sind Beide Pins gegen Ground zu ziehen. Das wird über die Pins RTS und DTR erreicht. Standardmäßig erfüllt der Pin 6 jedoch nicht die Funktionalität von DTR, sondern TNOW [26, S. 5]. Dieses Signal ist ein Indikator für eine aktuell ausgeführte Übertragung von Daten über die USB-Schnittstelle. Über einen externen Widerstand lässt sich die Funktion des Pins ändern, sodass er nun die Funktion eines DTR-Pins erfüllt. Dieser Pin zeigt den Zustand „Data Terminal Ready“ an und ist lowaktiv. Dafür muss der TNOW-Pin mittels eines 4.7kΩ zu CTS verbunden werden. CTS ist dabei das Signal für “Clear To Send”. Das weitere Signal, welches zur korrekten Aktivierung des Boot-Modus benötigt wird, ist das Signal RTS. Dieses Signal ist ein Ausgang, welcher als Lowaktives Signal ein „Request To Send“ anzeigt [26, S. 5]. Um einen korrekten Bootvorgang zu gewährleisten sind verschiedene Phasen nötig, damit die Software auch als neues Programm auf den Mikrocontroller aufgespielt wird. Zuerst muss der Enable-Pin am ESP32 auf Ground gezogen werden, um den Reset einzuleiten. Beim darauffolgenden Start-Up müssen die GPIO-Pins 0 und 2 auf den korrekten Pegeln liegen, damit der korrekte Bootmodus ausgeführt wird [13, S. 15]. Dieser Ablauf wird über die Schaltung mit den beiden Transistoren automatisch erzeugt, ohne dass ein Button auf der Platine benötigt wird, mit welchem händisch ein Bootmodus aktiviert werden muss, wie bei einigen Entwicklungsboards der Fall ist. Die Transistorschaltung ist dabei einem ESP Devkit V4 entnommen, welches mit demselben Prinzip arbeitet [33]. Durch diese Schaltung werden abhängig DTR und RTS unterschiedliche Pegel an den Pins GPIO 0 und EN erzeugt:

DTR	**RTS**	**EN**	**GPIO 0 / GPIO 2**
1	1	1	1 / 0
0	0	1	1 / 0
1	0	0	1 / 0
0	1	1	0 / 0

Die beiden Transistoren schalten jeweils, wenn die Spannung zwischen Basis und Emitter $U_{BE} > 0.7V$ ist. Das ist der Fall, wenn DTR und RTS unterschiedliche Signalpegel aufweisen. Sind die beiden Pegel an DTR und RTS identisch, dann schalten die Transistoren nicht. In diesem Fall haben die Pins GPIO 0 / 2 und EN die Pegel, welche über die Widerstände am Mikrocontroller definiert sind. Das ist für GPIO 0 High wie für Enable, GPIO 2 wird über einen Pull-Down Widerstand standardmäßig auf Ground gezogen. Wenn DTR auf High liegt und RTS auf Low schaltet der Transistor Q1401. Er zieht damit den Enable-Pin am Mikrocontroller auf den Pegel von RTS, also gegen Ground und löst damit den Reset des Microcontrollers aus. GPIO 0 liegt weiterhin auf High durch den Pull-Up Widerstand am Mikrocontroller. Sind die Pegel vertauscht, liegt also DTR auf LOW und RTS auf High schaltet der Transistor Q1402. Dadurch wird GPIO 0 auf den Pegel von DTR gezogen, welcher zu diesem Zeitpunkt auf Ground liegt. Der Enable-Pin des

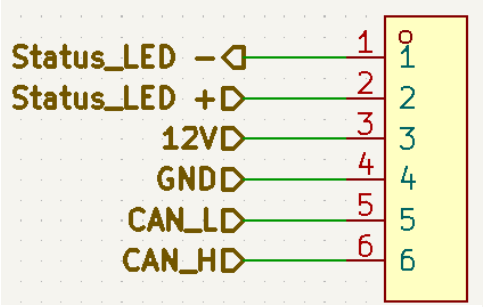
Mikrocontrollers bleibt dabei auf High durch den Pull-Up Widerstand am Mikrocontroller. Das ist damit der Bootmodus, indem der ESP32 mit den per USB übermittelten Daten startet. Beim Empfang von Daten via USB erzeugt der USB To Serial-Wandler somit automatisch einen Reset am Controller, aktiviert daraufhin den Bootmodus und kehrt anschließend wieder in den normalen Boot- und Betriebsmodus zurück. Auch hier wurden zum Schutz der Bauteile TVS Dioden vor den Pins des Mikrocontrollers vorgesehen, um Zerstörung durch ESD bedingte Spannungsspitzen zu verhindern. Zusätzlich wurde noch ein Programming Pinheader vorgesehen. Über diesen können sowohl Daten übertragen als auch die Pins in den korrekten Bootmodus versetzt werden. Dieser hat den Vorteil, dass im Falle einer defekten oder Fehlerhaften Schaltung trotzdem noch ein Flashen über die Verwendung eines Devkits oder eines ähnlichen Hilfsmittels möglich ist.

Stecker



Schaltplan 18: Pinbelegung des Display-Steckers

Um alle Funktionalitäten des Systems und alle Peripheriegeräte nutzen zu können, benötigt die Platine einige Stecker. Über den Stecker in Schaltplan 18 wird das Display an die Erweiterungsplatine angeschlossen. Das Display benötigt dabei 13 verschiedene Signale. Ein Großteil dieser Signale fallen auf Signalpins für die I^2C und die SPI-Kommunikation. Zusätzlich müssen noch Spannungsversorgung und Ground, sowie das PWM-Signal für die Hintergrundbeleuchtung des Displays vorgesehen werden. Sowohl der Stecker als Bauteil als auch die Pinbelegung werden dabei vom Display selbst vorgegeben [34]. Für die Verbindung zwischen Display und Platine soll ein fertiges Flachbandkabel verwendet werden, sodass die Reihenfolge der Pinbelegung am Platinenstecker der des Steckers am Display entsprechen muss.

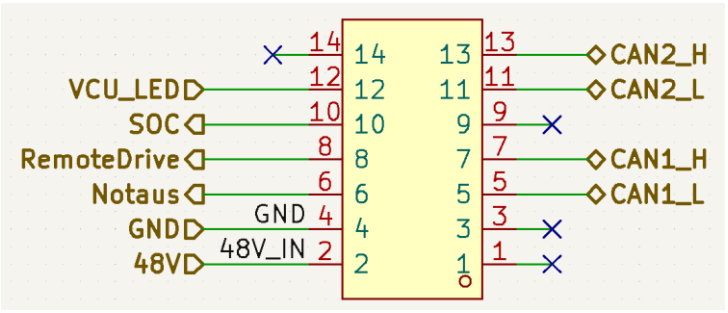


Schaltplan 19: Pinbelegung des Ausgangssignal Steckers

Für Ein- und Ausgangssignale werden Zwei Stecker vorgesehen. Der Stecker J_RFID in Schaltplan 19 verbindet den Kabelbaum mit der Platine, am anderen Stecker werden die beiden Zukaufteile RFID-Reader und Status-LED angeschlossen. Diese Trennung bringt den Vorteil, dass der gesamte Kabelbaum mit dem passenden Stecker vorbereitet werden kann, ohne nachträglich beim Einbau noch weitere Pins hinzufügen zu müssen.

Bei dem Stecker für die beiden letzten Bauteile handelt es sich nicht um eine zweiteilige Steckverbindung mit Buchse und Stecker, sondern die Litzen der Kabel werden über eine Quetschverbindung direkt im Stecker befestigt, welcher fest auf der Platine sitzt. So können einzelne externe Bauteile ohne großen Aufwand entfernt und weggelassen werden, sollte der Kunde diese Erweiterung nicht haben wollen. Der RFID-Reader benötigt dabei 4Pins. Bei diesen handelt es sich um die 12V Spannungsversorgung und Ground, sowie die beiden CAN-Leitungen. Für die Status-LED werden die beiden Signal des LED-Treibers nach außen geführt. Für Testzwecke wird zusätzlich noch ein Pinheader vorgesehen, über welchen ein RFID-Reader angeschlossen werden kann, welcher per SPI kommuniziert. Ziel ist es zukünftig den teureren RFID-Reader, welcher aktuell im Einsatz ist, durch eine kleinere und günstigere Variante zu ersetzen. Hierfür wird jedoch kein eigener Stecker vorgesehen, sondern lediglich ein Pinheader, da diese Schnittstelle für den Einsatz im Kart keine Rolle spielt und auf den Platinen weggelassen werden wird.

Der letzte Stecker auf Schaltplan 20 bildet die Schnittstelle zum Kabelbaum und damit zu allen anderen Komponenten des Karts. Über diesen Stecker findet sowohl die Spannungsversorgung von der Batterie für die Erweiterungsplatine statt als auch die Kommunikation mit dem Steuergerät, sowohl über die beiden CAN-Busse als auch über die Signalleitungen. Die Pinbelegung ist dabei hauptsächlich auf möglichst einfaches Layout ausgelegt. Vor allem die Leiterbahnen für die CAN-Busse sollen so kurz wie möglich sein, weswegen die Pins dafür entsprechend belegt wurden.



Schaltplan 20: Pinbelegung des Eingangssignal Steckers

5.2. PCB-Layout

Nachdem im vorangegangenen Kapitel der Schaltplan des entwickelten Systems detailliert beschrieben wurde, folgt im nächsten Schritt die Übertragung des Schaltungsentwurfs in ein Leiterplattendesign. Dieser Prozess umfasst nicht nur die reine Umsetzung elektrischer Verbindungen, sondern erfordert auch die Berücksichtigung mechanischer und Elektrischer Vorgaben. Dabei spielen insbesondere die äußeren Abmessungen der Leiterplatte sowie die Positionierung von Befestigungspunkten und Schnittstellen eine wesentliche Rolle, um eine zuverlässige Integration in das Gesamtsystem zu gewährleisten. Aufbauend darauf ist die Platzierung der elektronischen Bauteile entscheidend für die Signalqualität, die thermische Stabilität und die spätere Bestückbarkeit der Platine. Ebenso ist der gewählte Layer-Aufbau maßgeblich für die Leitungsführung, die Abschirmung empfindlicher Signale und die Sicherstellung einer stabilen Versorgungsspannung. Schließlich sind auch hochfrequente Signale gesondert zu betrachten, da deren Führung besondere Anforderungen an Impedanzkontrolle, Längenanpassung und EMV-gerechtes Design stellt. Im Folgenden werden diese Aspekte des PCB-Designs systematisch analysiert und die jeweiligen Designentscheidungen begründet.

Die Mechanischen Vorgaben werden vollständig vom bereits vorhandenen Gehäuse definiert. Dieses wird in Abbildung 20 detailreich gezeigt. Durch die Geometrie des Gehäuses werden die Abmaße sowie die Form der Platine bestimmt. Die Abmaße des Gehäuses betragen dabei 85mm x 98mm, wobei in den Ecken des Gehäuses jeweils Aussparungen von 7mm x 7mm vorgesehen werden müssen, da hier Einschmelzgewinde zur Befestigung des Gehäusedeckels eingebracht werden. Im Bild sind diese Bereiche blau markiert. Der vordere Teil des Gehäuses ist zudem für die Kabel vorgesehen, welche seitlich in das Gehäuse eingeführt werden. Somit muss der Bereich zwischen den beiden unteren Anschraubpunkten freigehalten werden. Als Auflage-fläche für die Platine dienen dabei die rot markierten Auflageflächen, welche zusätzlich auch die Anschraubpunkte zur Befestigung der Platine bieten. Für die Platine ergibt sich so eine Größe von 84.50mm x 76.50mm. Differenzen von 0.5mm sind dabei als Toleranz vorgesehen, sodass die Platine ohne Spannungen im

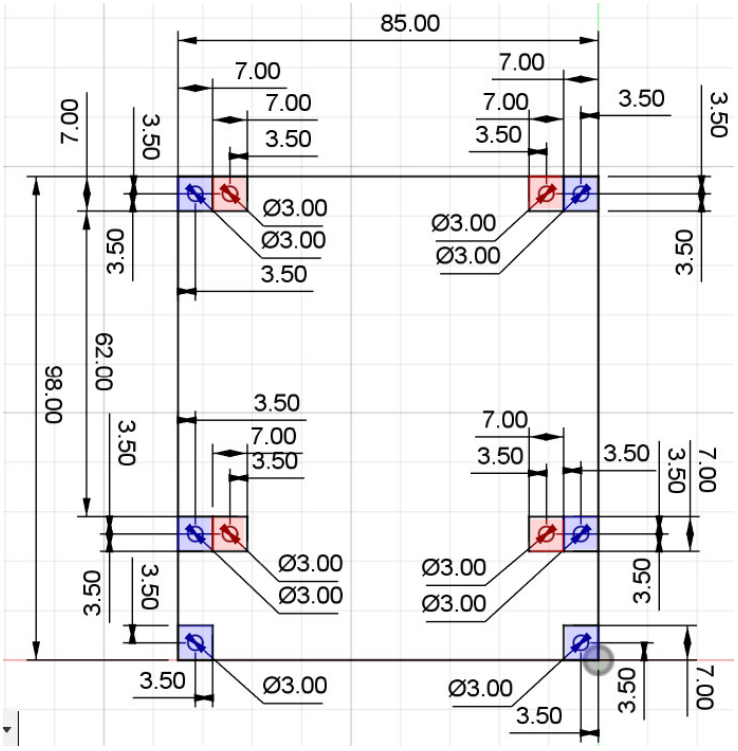


Abbildung 20: Bemaßung des bereits bestehenden Gehäuses

Gehäuse sitzt. Von den gegebenen Abmessungen müssen jedoch in jeder Ecke noch die Ausschnitte für die Anschraubpunkte des Deckels entfernt werden. Neben den Abmaßen und der Form der Platine werden auch die Anschraubpunkte bereits durch das bestehende Gehäuse definiert, zu sehen in Abbildung 21. Diese sind mittig in die Auflageflächen eingebracht. Dabei sind Schrauben der Größe M3 zur Befestigung vorgesehen. Zur Durchführung werden daher Löcher mit mindestens 3.6mm Durchmesser benötigt. Wichtig ist aber nicht nur die Löcher für die Schraube vorzusehen, sondern auch den Platz für den Kopf der Schraube, welche nicht mit den Bauteilen kollidieren darf. Dessen Größe beträgt bei einer Zylinderschraube einen Durchmesser von 5.5mm. Dieser Platz ist auf der Platine als Sperrfläche vorgesehen (grün), um versehentliches Platzieren von Bauteilen zu verhindern. Auch Leiterbahnen sollten in diesem Bereich nicht verlaufen, um Störungen zu verhindern. Ebenfalls als Sperrflächen vorgesehen sind die gesamten Auflageflächen auf der Unterseite der Platine (blau), um dort zu verhindern, dass Bauteile mit Durchsteckkontakten vorgesehen werden, sodass die Platine auf den Kontakten aufliegen würde.

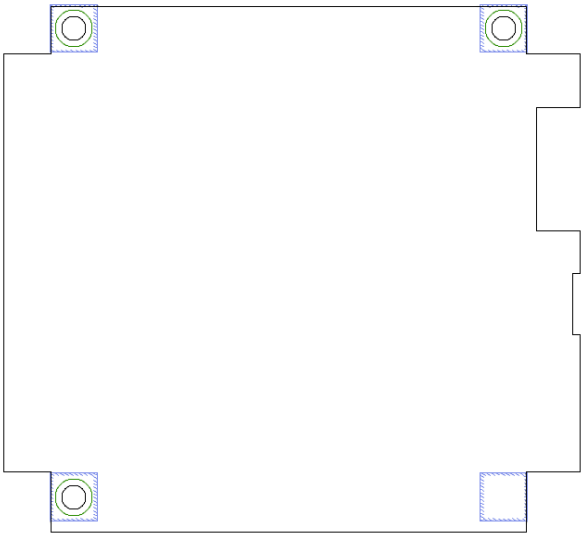
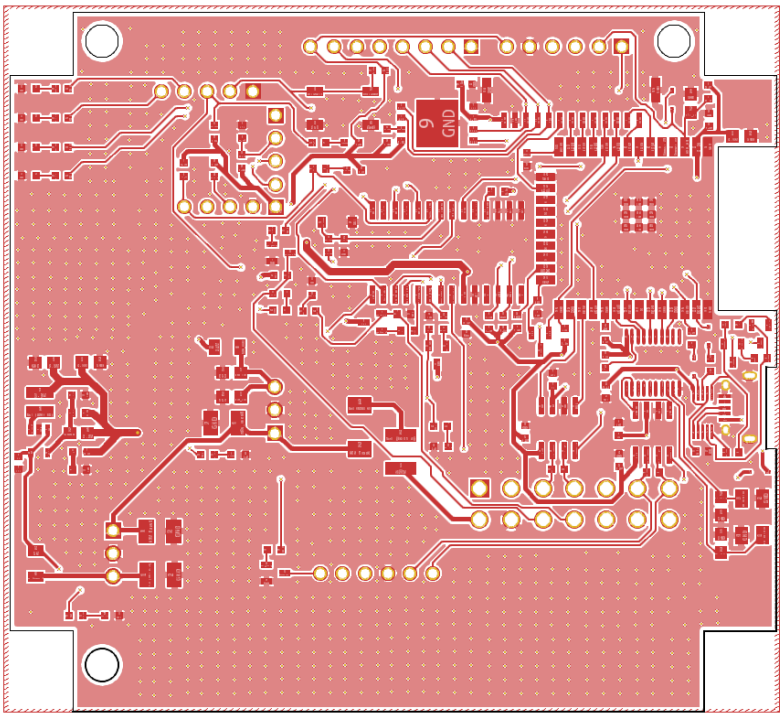


Abbildung 21: Darstellung der Anschraubpunkte und Sperrflächen

Zu den mechanischen Anforderungen einer Platine zählt auch die Wärmeableitung. Dafür wird bei dieser Platine eine durchgehende Massefläche, zu sehen in Platinenlayout 1, vorgesehen. Dabei werden zwei große Kupferflächen auf Top- und Bottomlayer hinzugefügt, welche nur die Bereiche von anderen Signalpfaden aussparen. Durch die großen Kupferflächen kann Wärme von heißen Bauteilen sehr gut abgeführt werden und über die gesamte Platine verteilt werden. Dadurch werden Thermal Hotspots vermieden, wodurch Platine und Bauteile geschützt werden. Auch die Verwendung und Anbindung an Kühlkörper wird so vereinfacht, auch wenn das für dieses Projekt eine eher untergeordnete Rolle spielt, da hier weder aktive Kühlelemente wie Lüfter vorgesehen sind, noch passive Kühlung über Kühlkörper oder Ableitung an das Gehäuse einen großen Effekt erzielen würden durch die Verwendung eines Kunststoffgehäuses mit schlechter Wärmeleitfähigkeit. Die Verwendung einer großen Massefläche bietet allerdings nicht nur mechanische, sondern vor allem viele Elektrische Vorteile. So birgt eine Verwendung von großen Masseflächen den Vorteil, dass diese Massefläche eine sehr niedrige Impedanz im Vergleich zu einzelnen Leiterbahnen bietet und damit wenig parasitäre Induktivitäten besitzt. Der Strom hat hier für jedes Signal die Möglichkeit, sich seinen Rückstrompfad selbst zu suchen, wodurch dieser sehr viel effektiver wird. Auch so werden hohe Temperaturen durch hohen Stromfluss an einzelnen Stellen der Platine vermieden. Zudem bieten die beiden Masseflächen auf Bottom und Top Layer einen guten Schutz

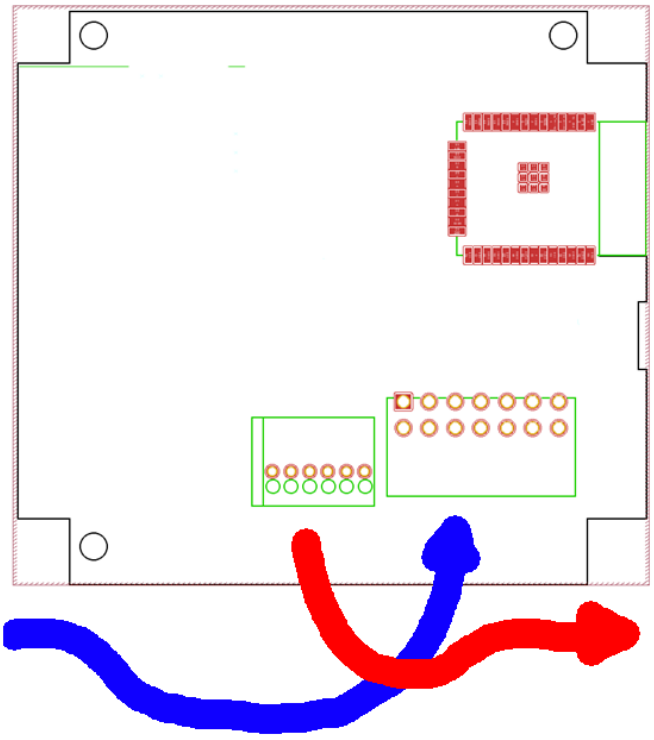
gegen Elektromagnetische Störungen. Zum einen wirken sie wie ein Schirm schützend für hochfrequente Signalleitungen durch Einflüsse von außen, zum anderen wird durch die breite Masseflächе die Erzeugung von EMV-Störungen durch die Rückstrompfad über eventuelle Groundleitungen stark verringert. Diese Störungen können vor allem auch dann auftreten, wenn die Leiterbahnen für den Rückstrom nicht sternförmig geführt werden. Diese Fehlerquelle wird durch die Verwendung von Masse-flächen über die gesamte Platine eliminiert. Um ähnliche Effekte zu erzielen, wird zusätzlich Via Stitching zwischen den beiden Masseflächen eingesetzt. Dabei handelt es sich um Durchkontaktierungen zwischen oberer und unterer Kupferschicht der Platine, welche engmaschig über die gesamte Platine verteilt werden, um die beiden Masseflächen auf Top und Bottom-Layer thermisch, mechanisch und elektrisch miteinander zu verbinden. Die Vias schaffen dabei einen entstörenden Bereich um hochfrequente Signale, wie zum Beispiel CAN, SPI oder I^2C, und verhindern so, dass Störungen von der Platine ausgesendet werden oder Signale auf der Platine gestört werden. Durch die Durchkontaktierung zwischen beiden Masseflächе wird zudem eine niederimpedante Verbindung zwischen allen Masseflächen geschaffen und die Bildung von Isolierten Kupferinseln, welche von Leiterbahnen umschlossen sind, verhindert. Der Rückstrompfad ist somit für alle Signale bestmöglich kurz, wodurch Störungen und Reflexionen vermieden werden. Zusätzlich werden die Stabilität und die Signalqualität verbessert durch den Einsatz von Vias, da hierüber Stromschleifen und somit Störspannungen vermieden werden, welche durch Schlechtes Layout von Leiterbahnen bei Verwendung eines Sternpunkt entstehen können. Das kommt vor allem differenziellen Signalen, wie sie auf dieser Platine mit CAN und UART auftreten, zugute. Durch die Vias wird neben den elektrischen Vorteilen auch thermische Eigenschaften verbessert. Durch die Vias ist es möglich, die Wärme über die gesamte Fläche der Platinen auf beide Seiten zu verteilen, ohne dabei vereinzelte Wärme Hotspots zu schaffen. Dadurch wird der Schutz hitzekritischer Bauteile nochmals verbessert und die Langlebigkeit der Platine sichergestellt. Das ist für dieses Projekt nicht unrelevant, da die Platine oft in wärmeren Umgebungen während Rennevents eingesetzt wird und die großen Spannungswandler auf der Platine viel Verlustleitung in Form von



Platinenlayout 1: Darstellung der Masseflächе und Leiterbahnen

Wärme erzeugen. Zusätzlich verbessern die Vias die mechanische Stabilität der Platine. Durch die Vias wird eine stabile Verbindung zwischen den Kupferschichten geschaffen, welche sie durch das Dielektrikum hindurch verbindet und somit Delamination verhindern kann.

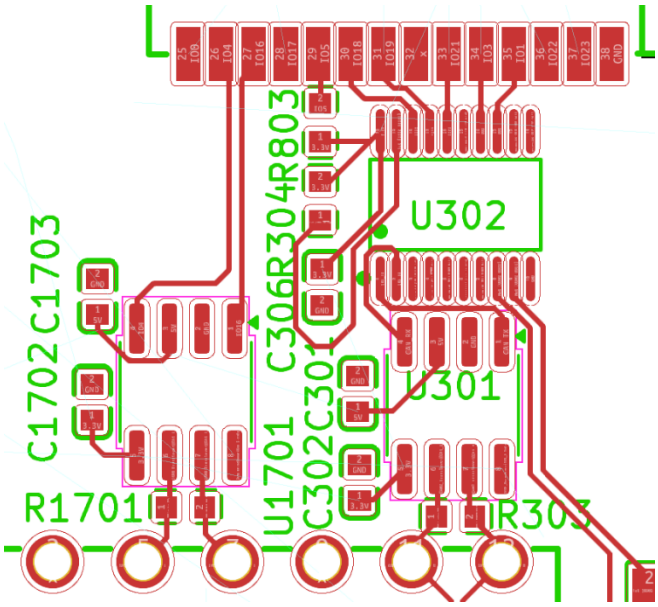
Nach dem Aufbau der Platine soll nun das tatsächliche Layout der Schaltung betrachtet werden. Für die Platzierung der Stecker spielen auch hier wieder mechanische Gegebenheiten eine Rolle. Die Platzierung der Stecker ist dabei so gewählt die Biegeradien der Kabel möglichst gering zu halten. Die Durchführung für die Kabelbaumseite befindet sich dabei auf der linken Seite des Gehäuses, weshalb die Platzierung des Steckers möglichst weit rechts auf der Platine platziert wird. Dadurch ist genug Platz vorhanden, um Entlastung für die Kabel und Biegeradien zu schaffen, wie in Platinenlayout 2 mit dem Blauen Pfeil dargestellt. Aus diesem Grund ist der Stecker auch auf die Platine eingerückt und nicht direkt an deren kante platziert. Zum einen ergeben sich so Flächen zur Darstellung der Pinbelegung auf der Platine. Zum anderen muss zusätzlich der Platz bedacht werden, welchen der kabelbaumseitige Stecker der Steckverbindung benötigt. Mit dieser Platzierung schließt die gesamte Steckverbindung mit der kante der Platine ab. Dasselbe gilt für den Stecker, welcher als Verbindung für RFID-Reader und LED dient, dargestellt durch den Roten Pfeil. Auch hier sollen Biegeradien möglichst geringgehalten werden. Dadurch, dass die Kabel in dem Stecker in einem 45° Winkel herausgeführt werden, ergeben sich auch keine Probleme durch die Überkreuzung von Eingehenden und Ausgehenden Kabeln, da die Leitungen problemlos übereinander geführt werden können. Die Platzierung des Mikrocontrollers ergibt sich durch die Platzierung des Kabelbaumsteckers und den Anforderungen, welche das Bauteil stellt. Durch die SMD WLAN-Antenne, welche Teil des Chips ist, muss der Mikrocontroller an den Rand der Platine platziert werden. Das ergibt sich dadurch, dass sich unter der WLAN-Antenne keine Kupferschichten befinden dürfen, da diese ansonsten die Signale stören und eine Verbindung unmöglich machen können. Die einfachste Methode, dies sicherzustellen, ist, am Rand der Platine einen Ausschnitt hinzuzufügen und den ESP32 darüber zu platzieren. Wichtig ist hierbei zu beachten, dass die Antenne nicht über die ursprüngliche Platinenkante hinüberragt, um Beschädigungen der Antenne zu vermeiden. Zudem wurde hier darauf geachtet den Mikrocontroller so zu



Platinenlayout 2: Platzierung der Stecker mit verlauf der Kabel

platzieren, dass das Layout der eingehenden CAN-Leitungen möglichst kurz und einfach gehalten werden kann. Deshalb wurde sich hier dafür entschieden, den Chip an die Rechte Seite der Platine zu platzieren.

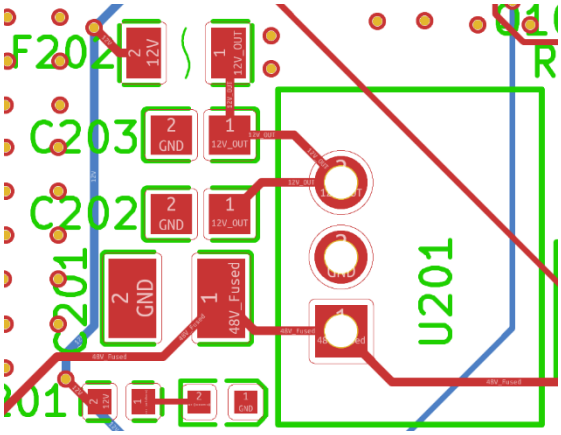
Für das Layout von CAN ist Störsicherheit das wichtigste Kriterium, welches es zu beachten gilt. Das soll anhand Platinenlayout 3 dargestellt werden. Zu diesem Zweck wurde darauf geachtet, dass die Signale möglichst optimiert geroutet sind. Dazu gehört, dass die Leitungen als Differenzielles Leitungspaar geführt sind. Das bedeutet, dass die Leitungen eng beieinander geführt werden und gleich lang sind. Ziel ist es, dass etwaige Störungen in gleichem Maße auf beide Signalleitungen einwirken, wodurch die Störung durch die Auswertung des Differenziellen CAN-Signals keinen Einfluss auf die Signalqualität hat. Die Reihenfolge der Bauteile, die Ausrichtung des Mikrocontrollers und die Pinbelegung des Steckers sind darauf ausgelegt, die Leitungen so einfach und Störungssicher wie möglich Routen zu können. Zudem sollte auf Knicke und Abzweigungen vermieden werden. Im aktuellen System betragen die Unterschiede der Leitungslängen zwischen Stecker und Transceiver 0.05mm und 0.02mm. Auch wurde darauf geachtet die Abschlusswiderstände möglichst nah am Stecker zu platzieren. Auch das Trägt dazu bei Reflexionen und Störungen zu verringern, da es durch die kurzen Abstände kaum Bereiche gibt, an denen eine Störung in die Schaltung eingreifen kann. Weitere Maßnahmen, um die Störungen zu verringern, sind undurchbrochene Masseflächen unter den differenziellen Signalen und deren Bauteile. So wurde bei CAN bewusst darauf geachtet, keine anderen Signale im selben Bereich zu routen. Vor allem Kreuzende Signalleitungen würden ein sehr hohes Störungspotenzial bietet, was unter allen Umständen vermieden werden soll. Neben anderen Signalen wurde im unmittelbaren Bereich der CAN-Schaltungen auch auf Masse-Vias verzichtet, um hier keine Störungen durch andere Signale in den CAN-Aufbau einzubringen. Auch die CAN-Signale selbst wurden durchgehend auf einem Layer geroutet, um die Leitungsimpedanzen so gering wie möglich zu halten. Des Weiteren wird auch darauf geachtet, alle Kondensatoren, welche hier als Abblockkondensatoren wirken sollen, so nah wie möglich an den jeweiligen Pins zu platzieren. Bei CAN2 muss zusätzlich noch der CAN-Controller platziert werden. Auch



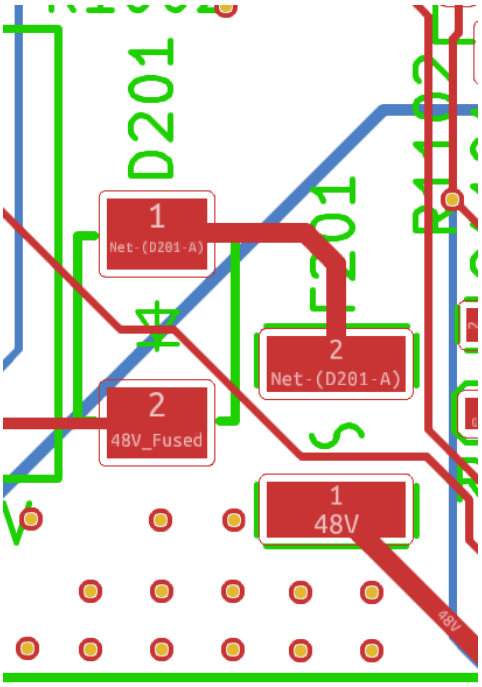
Platinenlayout 3: Layout der CAN-Beschaltung

hier wurde darauf geachtet, die Leitungsbahnen für die SPI-Kommunikation so kurz wie möglich zu halten und keine Leitungen zu kreuzen.

Viele der eben genannten Maßnahmen lassen sich dabei auf die gesamte Platine anwenden. So sollten alle Hochfrequenten, zusammengehörigen Leitungspaare, welche meist für Kommunikation genutzt werden, möglichst nahe beieinander geführt werden, um Störungen durch die Leitungen zu minimieren. Zudem wird durch eng beieinander liegende Leitungen eine in etwa gleiche Leitungslänge erreicht, welche vor allem bei Synchronen Kommunikationsprotokollen relevant ist. Außerdem bleibt es so leichter andere Signale zu routen, da bei hochfrequenten Signalen darauf geachtet werden sollte möglichst keine kreuzenden Leitungen über oder unter den Signalen entlangzuführen. Auch sollten die Leitungslängen so kurz wie möglich gehalten werden. Leider hat das beim vorliegenden System nicht immer funktioniert und die Signalleitungen für SPI und I^2C kreuzen sich untereinander und mit anderen Signalleitungen. In solchen Fällen sollte darauf geachtet werden, den Winkel zwischen den kreuzenden Leitungsbahnen möglichst nah an 90° zu bringen. Damit ist der Bereich, in dem die Leitungen sich überschneiden, minimal und ein Übersprechen der Signale am unwahrscheinlichsten. Auch für alle Kondensatoren, welche als Filterschaltung oder zur Verbesserung des Signalqualität dienen, wie in Platinenlayout 4 für den 12V DCDC dargestellt, gelten für alle Schaltungen dieselben Regeln, wie sie bereits bei CAN genannt wurden. Es ist wichtig die Leiterbahnen möglichst kurz zu halten und die Kondensatoren möglichst nah an dem Bauteil zu platzieren, welches geschützt werden soll oder dessen Spannung gefiltert werden soll. Das schützt davor, dass das bereits gefilterte Signal erneut gestört wird über die Leiterbahn zwischen Kondensator und Bauteil. Am wichtigsten ist die Platzierung der Abblock-kondensatoren als Filter hochfrequenter Spannungsspitzen direkt an den Ausgängen der Spannungswandler. Alle hier erzeugten Störungen breiten sich über die Versorgungsspannung über die gesamte Platine aus. Genauso wichtig ist es sich mit der korrekten Platzierung von Schutzbauteilen zu Beschäftigen. Dabei handelt es sich zum einen um die Sicherungen nach den Spannungswandlern auf 12V und 5V, aber auch um die Schutzdioden, welche vor allem bei der USB-Schaltung zum Einsatz kommen. Die Sicherungen sollen die Schaltungen vor Zerstörung durch hohe Ströme, zum Beispiel bei einem Kurzschluss in der Schaltung, schützen. Die Gefahr besteht dabei darin, das durch zu hohen Eingangsstrom Bauteile zerstört werden oder Leiterbahnen sich enorm erhitzen. Es ist daher besonders darauf zu achten, dass Sicherungen besonders nah der Quelle der Spannung platziert wird, um als erstes auszulösen, bevor sich bereits andere Bauteile oder Leiterbahnen erhitzen

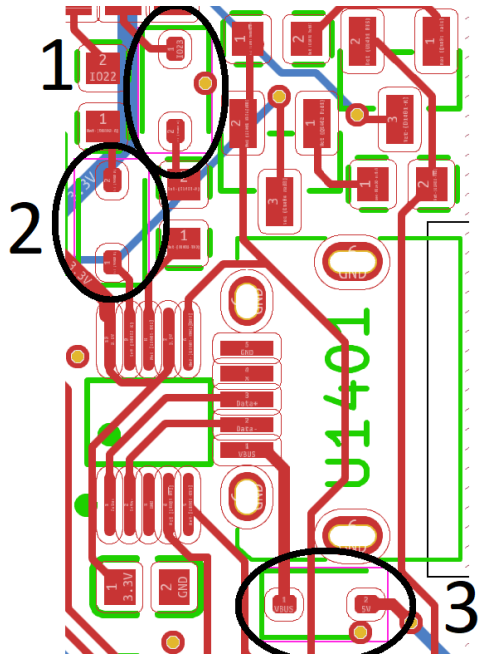


Platinenlayout 4: Platzierung von Abblockkondensatoren anhand des 12V Spannungswandlers



Platinenlayout 5: Platzierung von Sicherungen anhand der Eingangssicherung

konnten. In der Vorliegenden Schaltung bedeutet das die Sicherungen möglichst nah an den Ausgangspins der Spannungswandler zu platzieren, oder im Falle der Eingangssicherung für die 48V Versorgungsspannung, möglichst nah am Eingangsstecker. Außerdem müssen die thermischen Eigenschaften der Sicherung beachtet werden, da sich diese im Betrieb stark erhitzen können. Auf der Platine sind deshalb rund um die Sicherungen große Abstände und unter den Sicherungen durchgehende Kupferflächen vorgesehen, um die Wärme besser ableiten zu können und keine hitzeempfindlichen Bauteile zu beschädigen. All das kann man sehr gut anhand der Eingangssicherung betrachten auf Platinenlayout 5. Ähnliches gilt für die Schutzdioden. Die Dioden sollen im Falle eines Spannungsimpulses, zum Beispiel bedingt durch ESD-Impulse, die Überspannung ableiten und so die Schaltung schützen. Dafür ist es wichtig, dass die Dioden so nah wie möglich an der Quelle der Störung, also meist einem Stecker, über welchen die Störung von außen eingebracht wird, sitzen. Die Diode muss dabei sehr schnell reagieren, das kann sie nur, wenn der Pfad zu ihr so kurz wie möglich ist, wie man



Platinenlayout 6: Platzierung von Schutzdioden anhand der USB-Beschaltung

in Platinenlayout 6 sehr gut erkennen kann. Außerdem wird so die Leitung verkürzt, von welcher eine Störung in die restliche Schaltung ausstrahlen kann. An den beiden Schutzdioden 1 und 2 kann man die beiden genannten Prinzipien sehr gut verdeutlichen. Diode 2 soll den Spannungsspitzen am Ausgang des USB-Wandlers abfangen und ist deshalb möglichst nah am IC platziert. Diode 1 ist in den Signalpfad eingebracht mit Signalen, die vom USB-Wandler an den ESP32 geschickt werden und ist daher nahe am Ausgangspin des ESP32 platziert. Diode 3 ist die Schutzdiode, welche das 5V Signal vom USB-Host zur Versorgung der Schaltung abgreift, ist das kritischste Signal, da hier die Spannungsspitzen den größten Schaden anrichten können. Auch hier wurde darauf geachtet die Diode nah am Stecker zu positionieren, um Störungen frühzeitig zu filtern.

Ein weiterer Wichtiger Aspekt zum Schutz der Schaltung ist die korrekte Auslegung der Leiterbahnen für die Versorgungsspannung und die berechneten Ströme. Werden die Leiterbahnen zu dünn ausgelegt, ist die Erhitzung bei den fliesenden Strömen unter Umständen zu hoch und die Leiterbahnen und die Bauteile können Schaden nehmen.

Die mindestens benötigte Leiterbahnbreite berechnet sich dabei gemäß IPC-2221 [35, S. 50] folgendermaßen:

$$I = K * dt^{0.44} * (W * H)^{0.725}$$

Mit:

$$I = Strom\ [A]$$
$$K = 0.048\ für\ Außenschichtbahnen$$
$$\Delta t = Temparaturanstieg\ [°C]$$
$$W = Leiterbahnbreite\ [mil]$$
$$H = Leiterbahndicke\ [mil]$$

Daraus ergibt sich für W:

$$W = \frac{\sqrt[0.725]{\frac{I}{K * \Delta t^{0.44}}}}{H}$$

Bei den zu bestellenden Platinen beträgt die Leiterbahndicke $H = 35\mu m = 1.38\,mil$
Die Temperaturerhöhung wird auf $\Delta t = 20°C$ festgelegt. Dieser Wert wird recht hoch gewählt, da das Kart vor allem im Hochsommer auf Außenplätzen verwendet wird, wo das Kart durchgehend der Sonne ausgesetzt ist.
Für die 12V und 5V führenden Spannungspfade wird ein Strom von $I\_\max = 0.75A$ festgelegt, da das der Wert ist, ab welchem die Sicherung auslöst, sodass dadurch keine höheren Ströme langfristig auf der leiterbahn fließen können. Damit ergibt sich für 5V und 12V eine Leiterbahnbreite von

$$W = \frac{\sqrt[0.725]{\frac{0.75A}{0.048 * 20°C^{0.44}}}}{1.38mil} = 5.21mil = 0.13mm$$

Als Sicherheitsfaktor wird die Leiterbahn auf 0.2mm gewählt, um die leiterbahnen nicht am Limit zu betreiben.
Für den Spannungspfad der 3.3V werden 2A als Maximalstrom angesetzt. Das ist der Wert, welchen der Spannungswandler kontinuierlich liefern kann. Damit ergibt sich folgenden Leiterbahnbreite.

$$W = \frac{\sqrt[0.725]{\frac{2A}{0.048 * 20°C^{0.44}}}}{1.38mil} = 20.17mil = 0.51mm$$