Christoph Wiesmeier BSc

Modulares Universal Ladegerät mit konfigurierbaren Ladeverfahren

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium Telematik

eingereicht an der

Technischen Universität Graz

Betreuer

Ass.Prof. Dipl.-Ing. Dr.techn. Gunter Winkler

Institut für Elektronik

Graz, April 2016

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum

Unterschrift

Abstract

Currently available battery chargers are mostly designed for the consumer market. These devices often only offer a limited set of features, support usually only a small set of battery technologies and have no possibility to adapt the charging algorithm. For example, special tasks like changing the state of charge of a Li-Ion battery to an optimal value for storage are usually not supported.

Besides the basic charging of batteries, other interesting analytic features would be possible. Recording battery parameters on a cell level during charge/discharge cycles can provide detailed insights in the condition of a battery pack. Tracing of parameters over the lifetime of the pack can additionally provide information about the quality of the batteries and the charging algorithm.

As part of this thesis a new modular battery charger has been developed which makes the implementation of such analytic features possible. It consists of a controller module and several driver modules which are connected via a bus system. The controller module is based on a RaspberryPi with a touch screen. It provides a powerful user interface with many options including the visualization of the recorded data. Furthermore, it allows the user to define new parameterizable charging algorithms in Javascript. The driver modules can be optimized for specific use cases. For example, the currently implemented driver module has been designed for charging battery packs with arbitrary chemistry. It supports charging and discharging of battery packs with up to 30V and 10A. The power dissipation during discharging is limited to 100W. Additionally, a balancer with 12 channels is included.

Zusammenfassung

Momentan verfügbare Ladegeräte sind meist für den Durchschnittsverbraucher ausgelegt. Sie bieten nur einfache Ladefunktionen, unterstützen nur eine beschränkte Anzahl von Akkutechnologien und erlauben keinen Eingriff in das Ladeverfahren. Spezielle Anwendungen, wie das Anpassen des Ladezustands von Li-Ion Akkus für eine optimalen Lagerung, werden typischerweiseMomentan verfügbare Ladegeräte sind meist für den Durchschnittsverbraucher ausgelegt. Sie bieten nur einfache Ladefunktionen, unterstützen nur eine beschränkte Anzahl von Akkutechnologien und erlauben keinen Eingriff in das Ladeverfahren. Spezielle Anwendungen, wie das Anpassen des Ladezustands von Li-Ion Akkus für eine optimalen Lagerung, werden typischerweise nicht unterstützt.

Zusätzlich zu den elementaren Ladefunktionen gäbe es interessante Analysefunktionen. So gewährt das Aufzeichnen von Zellparametern während der Lade- und Entladevorgänge Einblicke in den Zustand von Akkupacks. Die Aufzeichnung der Parameter über den Lebenszyklus eines Akkupacks erlaubt des Weiteren Rückschlüsse auf die Qualität der Zellen und der Ladeverfahren.

Im Rahmen dieser Arbeit wurde ein neues modulares Ladegerät entwickelt welches das Umsetzten derartiger Analysen unterstützt. Es besteht aus einem Steuermodul und mehreren Ausgangsmodulen die über ein Bussystem miteinander verbunden sind. Das Steuermodul basiert auf einem RaspberryPi mit einem Touchscreen. Es bietet eine leistungsfähige Benutzeroberfläche mit vielen Optionen inklusive der Visualisierung der aufgezeichneten Daten. Des Weiteren erlaubt es die Definition und Konfiguration eigener Ladeverfahren in Javascript. Die Ausgangsmodule können für spezielle Anwendungen optimiert werden. Das aktuell implementierte Ausgangsmodul ist für das Laden von Akkupacks unterschiedlicher Technologien ausgelegt. Es ist in der Lage Akkus mit bis zu 30V und 10A zu laden bzw. zu entladen. Dabei können Entladeleistungen von 100W erreicht werden. Zusätzlich ist das Balancieren auf bis zu 12 Kanälen möglich.

Inhaltsverzeichnis

1 Motivation und Zielsetzung

Ziel dieser Arbeit ist die Entwicklung eines modularen und universell einsetzbaren Ladegerätes. Dabei sollen in erster Linie kleinere Akkupacks geladen werden, wie sie zum Beispiel im Modellbau oder in Elektrokleingeräten zum Einsatz kommen. Natürlich gibt es in diesem Bereich bereits eine große Vielfalt an Geräten, die in der Lage sind die gebräuchlichen Akkumulator-Typen zu laden. Leider sind diese Geräte immer nur für den Consumer-Markt optimiert. Sie erlauben keinen Eingriff in die bestehenden Ladeverfahren oder die Erstellung neuer Ladeverfahren. So ist es nicht möglich Eigenschaften, wie die Entladeschlussspannug beim Zyklisieren eines Ni-Mh-Akkumulators, zu verändern. Weiters erlauben die Geräte nur in eingeschränktem Maße Parameter der Batterie zu messen. Üblicherweise wird nur die Kapazität und in seltenen Fällen auch der Innenwiderstand bestimmt.

Um diese Einschränkungen zu umgehen, wird im Rahmen dieser Arbeit ein eigenes Ladegerät entwickelt. Dabei soll es für den Nutzer des Gerätes möglich sein die Ladeverfahren möglichst frei zu gestalten. Zu diesem Zweck soll das Erstellen eigener Ladeverfahren in einer Scriptsprache ermöglicht werden. Weiters soll das Ladeverfahren für den gegebenen Akkumulator parametriert werden um auf die Anforderungen unterschiedlicher Kapazität, Zellenanzahl, Zellspannung, Ladestrom und Ähnliches eingehen zu können. Als Steuerung soll ein Linux basiertes System dienen. Für die Bereitstellung einer leistungsfähigen Nutzeroberfläche dient ein Display und optional ein Webinterface. Die Leistungselektronik soll in der Lage sein bis zu 6 zellige Lithium Akkumulatoren zu laden und dabei Ladeströme bis $10A$ zu erreichen. Für eine adäquate Entladung ist eine leistungsfähige, elektronische Last mit entsprechender Kühlung für Entladeleistungen bis $100W$ vorgesehen.

Auf Basis dieser groben Zielsetzung wird eine genauere Spezifikation für die konkrete Entwicklung erstellt. Diese ist in Abschnitt 4.1 auf Seite 31 zu finden.

2 Akku Grundlagen

Dieses Kapitel gibt einen Überblick über gängige Batterie-Technologien. Es sollen sowohl die unterschiedlichen Eigenschaften und Anwendungsbereiche zeigen, als auch die Anforderungen an die Ladetechnik verdeutlichen. Weiters wird der Aufbau der Zellen diskutiert. Dies soll die Ursachen von Eigenschaften wie Alterung und Resistenz bezüglich Überladung verdeutlichen.

Dieses Kapitel beruht zu großen Teilen auf dem Buch "Moderne Akkumulatoren richtig einsetzen"[1].

2.1 Allgemeines

Akkumulatoren, wie auch Batterien, sind elektrochemische Energiespeicher. Sie erlauben die Bereitstellung elektrischer Energie durch Speicherung in chemischen Verbindungen. Als Akkumulatoren werden im Folgenden wiederaufladbare Speicher bezeichnet. Der Begriff Batterie bezieht sich auf einen nicht wiederaufladbaren Speicher (Primärzelle). Dabei wird im allgemeinen nicht zwischen einzelnen Zellen und mehreren miteinander vorschalteten Zellen (Pack) unterschieden.

Eine Batterie besteht im Prinzip aus zwei Komponenten. Einem Speicher, der die chemischen Verbindungen und damit die Energie enthält und einem Wandler, der die Umwandlung von chemischer in elektrische Energie vornimmt. Mit Ausnahme des in Entwicklung befindlichem Redox-Flow Verfahrens [13], besitzen alle handelsüblichen Akkumulatoren einen Internen Speicher, das heißt, Speicher und Wandler sind die selben Komponenten.

Akkumulatoren bestehen im Allgemeinen aus zwei Elektroden und einem Elektrolyten. Durch die unterschiedlichen chemischen Potentiale entsteht eine Spannung zwischen den Elektroden. Wird der Akku belastet, kommt es zu einem Stromfluss durch den Verbraucher. Im Akku schließt sich der Stromkreis durch den Elektrolyten. Dieses Prinzip wird in Abbildung 2.1 auf der nächsten Seite verdeutlicht.

Die verschiedenen Akkutechnologien unterschieden sich im Wesentlichen durch deren Werkstoffkombinationen. Dabei gibt die Auswahl von Anoden- und Kathodenmaterial die Spannung der Zelle vor.

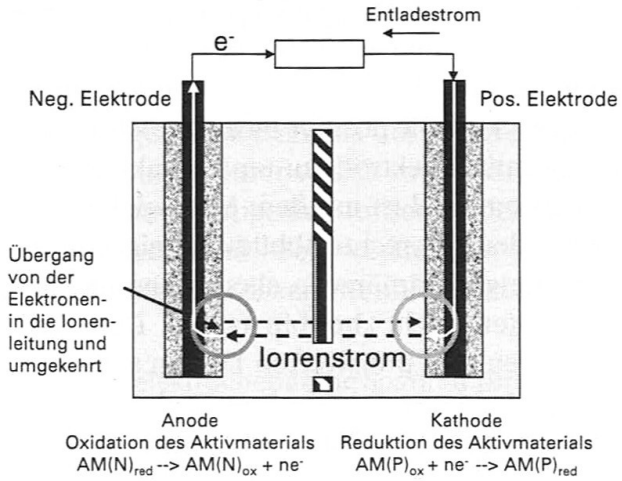


Abbildung 2.1: Grundprinzip eines Akkus. Bild aus [1]

2.2 Algemeiner Aufbau

Der reale Akkumulator besteht aus einer positiven und einer negativen Elektrode. Zwischen diesen befindet sich ein in Elektrolyt getauchter Separator. An den Außenseiten befinden sich zusätzliche Stromableiter.

Die positive und negative Elektrode bestehen aus einem Aktivmaterial. Dieser Teil ist für die Energiespeicherung verantwortlich. Da die Reaktionen meist an den Grenzflächen zum Elektrolyten stattfinden, werden die Elektroden porös ausgeführt. Dies erlaubt ein Eindringen des Elektrolyten und somit eine Maximierung der aktiven Oberfläche. Dabei sind Porengrößen von etwa $50\mu m$ üblich. Die poröse Struktur erhöht die Kontaktflächen mit dem Elektrolyten und somit die Leistungsfähigkeit des Akkus. Ein weiterer wesentlicher Punkt ist die Ausnutzung das Aktivmaterials, das an der Reaktion beteiligt ist. Bei vielen Technologien kann nicht das gesamte Elektrodenmaterial verwendet werden.

Aufgrund der geringen Leitfähigkeit einiger Aktivmaterialien der Elektroden werden diese meist dünn auf einem metallischen Stromableiter aufgebracht. Die wesentliche Anforderung ist hierbei eine gute elektrische Leitfähigkeit und die Beständigkeit gegenüber dem Elektrolyten bei den unterschiedlichen Spannungspotenzialen.

Als Elektrolyten kommen je nach verwendetem Aktivmaterial wässrige Lösungen wie Schwefelsäure, Kalilauge oder aber wasserfreie Elektrolyte zum Einsatz. Beim Verwenden von wässrigen Elektrolyten stellt vor allem die elektrolytische Zerlegung von H_2O eine Problem dar.

Als mechanische Trennung der beiden Elektroden kommt ein Separator zum Einsatz. Dabei handelt es sich meist um eine poröse Kunststofffolie. Diese verhindert das Berühren der Elektroden und somit einen Kurzschluss.

2.3 Batterien und Akkus

Bei den elektrochemischen Zellen wird im Wesentlichen zwischen drei Typen unterschieden.

2.3.1 Primärzellen

Primärzellen sind nicht wiederaufladbare elektrochemische Speicher und werden umgangssprachlich als Batterien bezeichnet. Momentan gebräuchliche Systeme sind die Alkali-Mangan-Zelle (Alkaline) und die kaum noch verwendete Zink-Kohle-Batterie, sowie Lithium-Mangan-Dioxid-Zellen, die meist als Knopfzelle zum Einsatz kommen.

Auch wenn diese Zellen theoretisch meist wiederaufgeladen werden könnten, stehen dem Wiederaufladen meist praktische Probleme gegenüber. So zerfällt oft die Struktur des Aktivmaterials bei der Entladung. Weiters kann bei einem Wiederaufladen die innere Geometrie nicht aufrecht erhalten werden. Außerdem kommt es beim Laden häufig zu Nebenreaktionen, z.b. der Entstehung von Wasserstoff. Ein damit einhergehender Druckanstieg in der Zelle führt schließlich zu einem Öffnen der Zelle und somit zu deren Zerstörung.

Ein Beispiel für den fließenden Übergang stellt die RAM-Zelle [10][12] dar. Dabei handelt es sich um eine spezielle Form der Alkali-Mangan-Zelle, die einige Male wiederaufgeladen werden kann.

2.3.2 Secondärzellen (Akkumulator)

Die im Rahmen dieser Arbeit interessante Gruppe der Sekundärzellen sind wiederaufladbare, elektrochemische Speicher und werden im Folgenden als Akkumulator kurz Akkus bezeichnet. Hier gibt es die Unterscheidung zwischen Zellen mit internem und mit externem Speicher. Beim internen Speicher erfolgt die Energiespeicherung sowie die Umwandlung von chemischer in elektrische Energie kombiniert im Aktivmaterial, Beim externen Speicher erfolgt die Umwandlung und die Speicherung getrennt.

Alle handelsüblichen Akkus verfügen über einen internen Speicher. Dabei ergibt sich ein starker Zusammenhang zwischen Energie- und Leistungs-Dichte.

Eines der wenigen Verfahren mit externem Speicher ist der Redox-Flow-Akku [13], der für die Langzeit-Energiespeicherung interessant ist.

2.3.3 Tertärzellen (Brenstoffzellen)

Eine Brennstoffzelle ist ein Wandler von chemischer in elektrische Energie mit externem Speicher. Im Gegensatz zum Akku erfolgt dabei die Umwandlung nur in eine Richtung. Klassischerweise wird dabei die Reaktion von Wasserstoff ($2H_2$) und Sauerstoff (O_2) zu Wasser ($2H_2O$) verwendet.

2.4 Alterung

Dieser Teil soll einige allgemeine Alterungsmechanismen beschrieben. Die konkreten Alterungsmechanismen werden dann bei den einzelnen Zelltypen noch einmal erwähnt.

Im Rahmen der chemischen Reaktionen kommt es meist zu einer Volumsänderung der Elektroden. Diese kann abhängig vom Elektrodenmaterial vom einstelligen Prozentbereich auf bis zu mehreren 100 Prozent (bei Primärzellen) betragen. Die Volumsänderung beim Laden und Entladen führt mit der Zeit zu einem Strukturverlust und somit zu einer Beschädigung der Elektroden. Daraus ergibt sich ein Anstieg des Innenwiderstandes.

Eine weitere Art der mechanischen Beschädigungen ist die Ausbildung von Dentriten. Diese Ablagerungen des Aktivmaterials entstehen meist durch eine hohe Löslichkeit des Aktivmaterials im Elektrolyten. Dabei lagert sich das Material im Seperator ab und stellt dadurch eine Verbindung zwischen der positiven und der negativen Elektrode dar. Es kommt zu einem inneren Kurzschluss, der sich meist in einer Erhöhung der Selbstentladung äußert.

Speziell bei wässrigen Elektrolyten stellt der Verlust des Elektrolyten einen wesentlichen Alterungsfaktor dar. Beim Überschreiten einer Zellenspannung von 1.24V kommt es unweigerlich zu elektrochemischen Zerlegung des Wassers. Einige Zelltypen versuchen die Gase zu rekombinieren. Erfolgt keine Rekombination kommt es zum Elekrolytverlust. Die Zerfallsreaktionen treten in erster Linie beim Überladen der Zellen auf. Das bei der Zerlegung des Wassers entstehende Gas führt zu einem Druckanstieg in der Zelle. Dies kann zum Öffnen des Überdruckventils und somit zu einem massiven Elektrolytverlust führen. Enthält die Zelle nicht mehr ausreichend Elektrolyten, kommt es zu einer Erhöhung des Innenwiderstandes.

Ein weiteres Problem ist auch die Korrosion des Stomableiters. Dies tritt zum Beispiel bei Bleibatterien, sowie beim Tiefentladen von Lithium-Zellen auf.

2.4.1 Ersatzschaltbild

Das gängige Ersatzschaltbild eines Akkus ist in Abbildung 2.2 dargestellt. Hier werden die wesentlichen Eingenschaften der Zelle modelliert.

Die innere Spannungsquelle U_0 repräsentiert das Spannungspotential, dass durch die chemische Reaktion im Inneren der Zelle hervorgerufen wird.

R_Ion beschreibt den ionischen Innenwiderstand. Dieser ist ein Resultat der begrenzten Ionenleitfähigkeit des Elektrolyten und der begrenzten Reaktionsgeschwindigkeit an der Grenzschicht zwischen Elektronylt und Elektrode.

Durch die unterschiedlichen Spannungspotentiale des Elektrolyten und der Elektroden entsteht an der Grenzschicht eine elektrisches Feld. Diese Doppelschichtkapazität kann erhebliche Energiemengen beinhalten. Sie kommt auch bei sogenannten Superkondensatoren (Doppelschichtkondensatoren) zum Einsatz. Die Modellierung erfolgt in Form von (C_D).

Der elektrische Widerstand der Elektroden sowie der Stromableiter wird in R_I zusammengefasst.

Schlussendlich wird die Selbstentladung noch durch R_G modelliert.

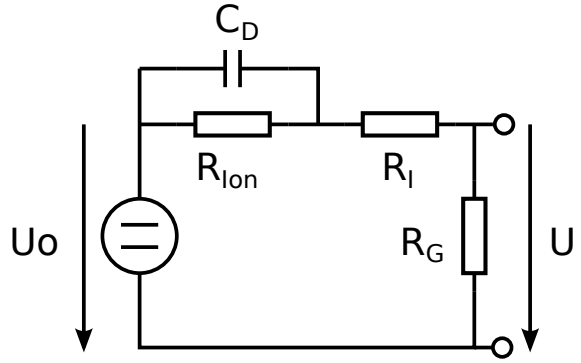


Abbildung 2.2: Batterie Ersatzschaltbild. U: Klemmenspannung, U_0: Chemisches Spannungspotential ,R_Ion: Ionischer Innenwiederstand, C_D: Doppelschichtkapazität, R_I: Ohmscher Innenwiderstand, R_G: Selbstentladung

Ein weiteres, aber langsameres Verhalten, das auf die gleiche Art wie R_Ion und C_D beschrieben werden kann ist die limitierte Diffusionsgeschwindigkeit. Es kommt dabei zu einer Reduktion der Ionendichte an den Grenzen zum Aktivmaterial. Dieser Effekt besitzt jedoch deutlich höhere Zeitkonstanten. Bei diesem Effekt ist zu beachten, dass die Diffusionsgeschwindigkeit sehr stark mit der Temperatur zunimmt.

Ein typische Antwort auf einen Lastanstieg ist in Abbildung 2.3 auf der nächsten Seite zu sehen. Dabei sind die Effekte der ohmschen und ionischen Widerstände sowie der Doppelschichtkapazität zu erkenne.

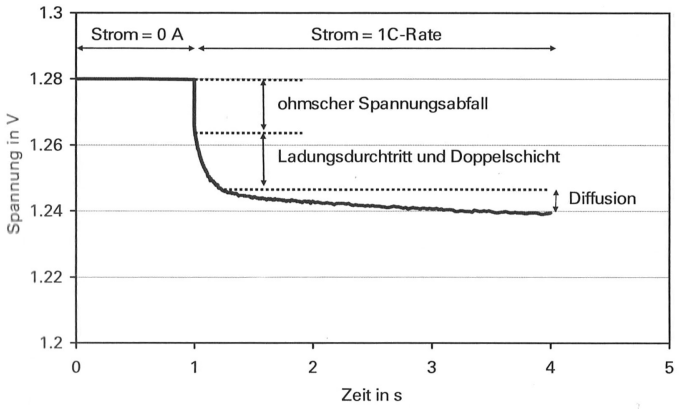


Abbildung 2.3: Spannungsverlauf bei einem Lastsprung. Bild aus [1]

2.4.2 Peukert Gleichung

Besonders bei NiMh- und PB-Zellen sinkt die entnehmbare Kapazität mit steigendem Strom. Dieser Effekt wird durch die Peukert Gleichung (2.1) approximiert. Diesen Effekt kann man am Beispiel von Bleiakkumulatoren in Abbildung 2.5 auf Seite 10 erkennen. Typische werte für einige Batterietypen sind in Tabelle 2.1 zusammengefasst.

Zelltype	Peukert Zahl (k)
Alkali-Mangan-Primärzelle	1.45
PB	ca. 1.1-1.3
NiMh	ca. 1.09
Li-Ion	ca. 1.05

Tabelle 2.1: Die Werte stammen von der deutschen Wikipedia. Auch wenn die Größenordnungen mit eigenen und allgemeinen Beobachtungen Übereinstimmen, konnte dafür keine wissenschaftliche Quelle gefunden werden.

$$t = \frac{C}{I^K} \tag{2.1}$$

2.5 Akku Technologien

Im Folgenden werden unterschiedliche Akku-Technologien vorgestellt. Abbildung 2.4 auf der nächsten Seite zeigt Energie- und Leistungs-Dichte unterschiedlicher Technologien.

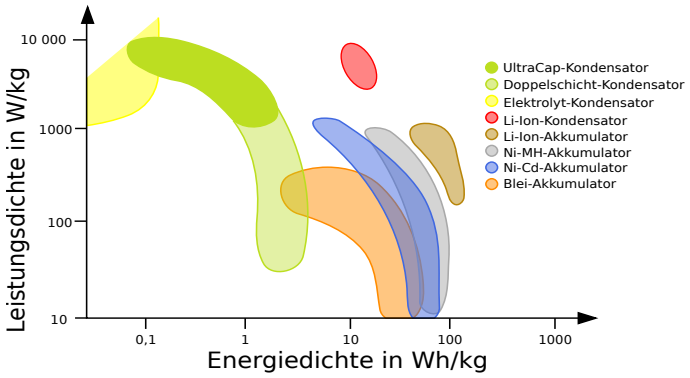


Abbildung 2.4: Vergleich Akku Technologien. Bild von `https://commons.wikimedia.org`

2.5.1 Bleiakku

Bleiakkus gehören zu den Älteren Akku Technologien. Sind aber aufgrund einiger interessante Eigenschaften immer noch im Einsatz.

2.5.1.1 Aufbau

Beim Bleiakku ist die positive Elektrode aus Reinblei (Pb) und die negative aus Bleidioxid (PbO_2). Als Elektrolyt kommt Schwefelsäure (H_2SO_4) zum Einsatz. Beim Entladen entsteht an beiden Elektroden Bleisulfat ($PbSO_4$). Das heißt, der Elektrolyt ist an der Speicherreaktion beteiligt. Dadurch verändert sich auch die Konzentration der Schwefelsäure und somit ihre Dichte abhängig von Ladezustand. Um die Oberfläche zu erhöhen wird das Aktivmaterial meist als Paste auf ein Bleigitter aufgebracht.

2.5.1.2 Eigenschaften

Das Funktionsprinzip des Bleiakkus ist eine ältere Technologie und zeichnet sich vor allem durch einen niedrigen Preis aus. Einer der Nachteile des Bleiakkus ist das hohe Spezifische Gewicht, welches den Anwendungsbereich limitiert.

Eine wesentliche Einschränkung ist, dass die Lebensdauer bei hoher Zyklentiefe stark abnimmt. So ergibt sich als Extremfall, dass Starterbatterien nach etwa 10 vollständige Lade/Entladung Zyklen Defekt sind.

Da beim Laden Wasserstoff entsteht, ist besonders bei größeren Anlagen eine Belüftung der Akkumulatoren notwendig.

Aufgrund der veränderten Säurekonzentration entsteht eine vom Ladezustand abhängige Ruhespannung. Diese kann zur Bestimmung des Ladezustandes verwendet werden. Abbildung 2.5 zeigt den Spannungsverlauf des Akkumulators

beim Entladen mit unterschiedlichen Strömen. Hierbei zeigt sich auch noch einmal die in Punkt 2.4.2 (Peukert Gleichung) diskutierte Abhängigkeit der Kapazität vom Ladestrom.

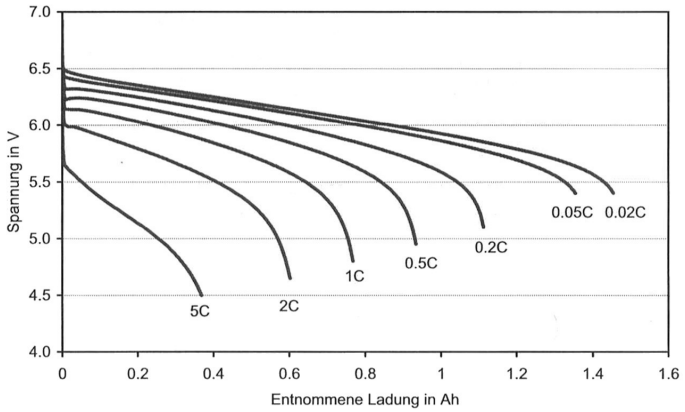


Abbildung 2.5: Entladevorgang Bleiakku. Bild aus [1]

2.5.1.3 Ladeverfahren

Aufgrund der Veränderung der Ruhespannung in Abhängigkeit des Ladezustandes bietet sich eine spannungslimitierte Konstantstromladung (Punkt 2.6.1) an.

2.5.1.4 Einsatzbereiche

Haupteinsatzbereich von Bleiakkumulatoren sind momentan Starterbatterien bei Kraftfahrzeugen. Ein weiterer großer Einsatzbereich sind Traktionsbatterien für Stapler, da in diesem Bereich das hohe Gewicht des Akkus keinen Nachteil darstellt. Außerdem werden Bleiakkumulatoren für unterbrechungsfreie Stromversorgungen eingesetzt, da in diesem Bereich normalerweise keine hohen Zyklenzahlen zustande kommen.

2.5.1.5 Alterung

Einer der primären Alterungsmechanismen bei Bleiakkumulatoren ist die Sulfierung. Im normalen Betrieb des Akkumulators bilden sich beim Entladen Sulfatkristalle. Aber bei längerer Lagerung mit niedrigen Ladezustand kommt es jedoch vermehrt zur Bildung größerer Kristalle, die sich kaum wieder auflösen. Folglich kommt zu einem Kapazitätsverlust.

Ein weiterer Alterungsfaktor ist die Korrosion der Bleigitter, auf denen das Aktivmaterial aufgebracht ist. Aufgrund der dabei entstehenden Reduzierung des Querschnittes des Gitters kommt es zu einer Erhöhung des Innenwiderstandes und im Extremfall sogar zum Abbrechen der Gitter.

Bei manchen Bleiakkumulatoren ist der Elektrolytverlust ein wesentliches Problem. Beim Laden wird als Nebenreaktion das Wasser durch Elektrolyse aufgespalten wodurch es zum Austrocknen der Zelle kommt. Verschlossene Akkumulatoren versuchen den dabei entstehenden Sauerstoff und Wasserstoff zu rekombinieren und somit den Wasserverlust zu minimieren.

2.5.2 Alkalische Akkumulatoren (NiCd und NiMh)

Nickel-Cadmium (NiCd) und Nickel-Metallhydrid (NiMh) Akkumulatoren besitzen einen sehr ähnlichen Aufbau und vor allem ein sehr ähnliches elektrisches Verhalten, deswegen werden sie hier gemeinsam behandelt. Weiters ist der NiCd-Akkumulator eine kaum noch eingesetzte Technologie, da Cadmium wegen seiner Giftigkeit durch die RoHS-Richtlinie (Restriction of Hazardous Substances) für die meisten Anwendungen verboten wurde. Auch in den verbleibenden Anwendungen wie Powertools wurden sie großteils durch NiMh- und Li-Ion- Technologien verdrängt. Mit Ende 2016 werden auch diese Ausnahmen auslaufen[1].

2.5.2.1 Aufbau

Auf der positiven Elektrode findet beim Entladen der Zelle eine Umwandlung von Nickel-Hydroxid $NiOOH$ in zweiwertiges Nickel Hydroxid $Ni(OH)_2$ statt (Gleichung 2.2). Das dabei freiwerdende OH^--Ion wird auf der negativen Elektrode entweder durch Umwandlung von Cadmium Cd in Cadmiumhydroxid $Cd(OH)_2$ gebunden (Gleichung 2.3) oder durch die Reaktion mit Wasserstoff aus einem Metall-Hydrid zu Wasser (Gleichung 2.4).

$$NiOOH + H_2O + e^- \rightarrow Ni(OH)_2 + OH^- \tag{2.2}$$

$$Cd + 2OH^- \rightarrow Cd(OH)_2 + 2e^- \tag{2.3}$$

$$MH + OH^- \rightarrow M + H_2O + e \tag{2.4}$$

In beiden Fällen kommt Kalilauge(KOH) als Elekrolyt zum Einsatz.

[1] http://www.europarl.europa.eu/news/de/news-room/20131004IPR21519/Parlament-stimmt-f%C3%BCr-Verbot-von-Cadmium-in-Batterien-f%C3%BCr-Elektrowerkzeuge Aufgerufen 10.2.1016

2.5.2.2 Eigenschaften

Die Ruhespannung liegt bei $1.30V$ für NiCd- und $1.32V$ für NiMh-Zellen.

Viele Zellen weisen eine hohe Selbstentladung von einigen Prozent pro Woche auf. Dieses Problem wird aber bei bestimmen neuen Zellen weitgehend verhindert.

Ein Problem für die Ladetechnik ist, dass die Zellen eine Hysterese der Ruhespannung aufweisen. Die Ruhespannung der Zelle liegt währende des Ladevorgangs etwa $70mV$ über der des Entladevorgangs bei gleichem L. Dies macht eine Bestimmung des Ladezustands anhand eines Spannungskriteriums schwierig.

Der oft diskutierte klassische "Memory Effekt"tritt nur beim langen Überladen von NiCd-Zellen auf. Eine Art "Memory Effekt", der sogenannte "Lazy Effekt", tritt auch bei NiMh-Zellen auf. Dabei kann bei wiederholter teilweiser Entladung der Zelle eine Spannungsstufe von bis zu $50mV$ entstehen. Diese Spannungsstufe führt zwar zu keiner Verringerung der Kapazität, kann aber zu einem frühzeitigem Abschalten der Ladeelektronik führen. Diese Spannungsstufe lässt sich durch einen einzigen vollen Ladezyklus beseitigen.

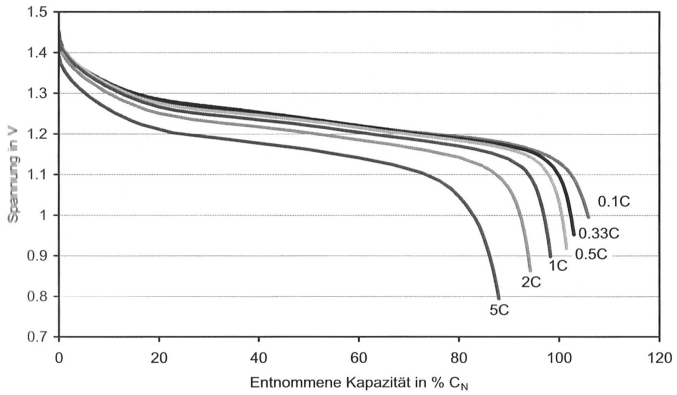


Abbildung 2.6: Entladevorgang einer NiCd Zelle mit unterschiedlichen Entladeströmen. Bild aus [1]

2.5.2.3 Ladeverfahren

Reine spannungsbasierte Verfahren funktionieren aufgrund der vielen Störfaktoren für die Ladung nicht. Früher kam oft ein langsames, zeitbasiertes Ladeverfahren mit konstantem Ladestrom zum Einsatz. Da dies aber häufig zu einer Überladung und somit langfristig zu einer Schädigung der Zellen führt, ist heute das $-\Delta U$ -Verfahren üblich. Weiters können auch komplexere spannungsbasierte Verfahren beziehungsweise temperaturbasierte Verfahren zum Einsatz kommen. Die Abschaltkriterien werden im Punkt 2.6.7 näher diskutiert.

2.5.2.4 Eisatzbereiche

NiMh-Akkumulatoren werden heute vor allem in Consumer-Geräten eingesetzt. Dabei sind sie vor allem im Formfaktor der AA-, AAA- Batterie gebräuchlich. Aufgrund der höheren Energiedichte und sehr ausgereiften Ladetechnik werden sie aber heute in vielen Einsatzbereichen von Li-Ionen Akkumulatoren verdrängt. Dasselbe gilt für den Bereich von Elektro- und Hybrid-Fahrzeugen.

2.5.2.5 Alterung

Bei Überladung kommt es zur Veränderung der Kristallstruktur in der Nickel-Elektrode. Dabei kommt es zur Volumsänderung und somit zur mechanischen Schädigung der Zellen.

Auch auf der Metallhydrid-Elektrode des NiMh-Akkumulators kommt es beim Laden zu einer Volumsänderung. Diese mechanische Beanspruchung führt zu einer von der Zyklentiefe abhängigen Alterung.

Mit der Zeit verlieren die Zellen auch Wasser. Es kommt somit zum Austrocknen des Elektrolyten und folglich zum Anstieg des Innenwiderstandes.

2.5.3 Lithium-Akkumulatoren

Lithiumbasierte Akkumulatoren sind momentan in fast allen Bereichen im Vormarsch. Mit Ausnahme von Standard-Batterien (der Größen AA, AAA) und PowerTools, wie Akkuschraubern werden sie aufgrund ihrer hohen Energiedichte in immer mehr Geräten eingesetzt.

2.5.3.1 Aufbau

Das Funktionsprinzip der Zellen unterscheidet sich etwas von Nickel- und Blei-Technologien. Bei Lithium-Technologien agiert Li^+ selbst als Ion und wird zwischen positiver und negativer Elektrode transportiert. Die Speicherung erfolgt meist auch nicht durch ein herkömmliche chemische Bindung, sondern durch die Einlagerung im Kristallgitter der Elektroden.

Auch ist zu beachten, dass es bei Lithium eine sehr große Anzahl von Subtechnologien gibt. Diese unterscheiden sich in den dabei verwendeten Elektrodenmaterialien und Elektrolyten. Besonders die verschiedenen Aktivmaterialien sind wichtig, da sie unterschiedliche Potentiale und somit Zellspannungen aufweisen. Eine Darstellung einiger gebräuchlicher Aktivmaterial-Kombinationen ist in Abbildung 2.7 dargestellt. Dabei sei vor allem die für Primärbatterien, wie Knopfzellen, verwendete Materialkombination $MnO_2/Li(Metall)$ und die in

den klassischen Lithium-Rundzellen verwendete Kombination $LiCoO_2/Graphit$ hervorgehoben.

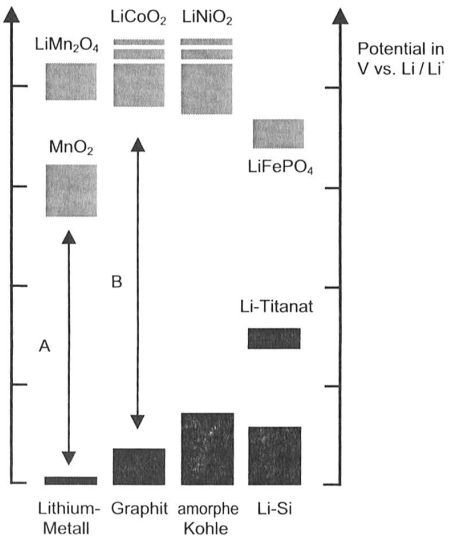


Abbildung 2.7: Potentiale der unterschiedlichen Materialkombinationen von Lithium Akkus. Bild aus [1]

Der Elektrolyt ist außerdem ein wesentlicher Faktor. Dabei werden zwei Technologien unterschieden. Die klassischen *Li-Ion*-Zellen, meist Rundzellen, besitzen einen flüssigen Elektrolyten. Dabei ist ein festes Metallgehäuse erforderlich um die mechanische Stabilität zu gewährleisten. Als Alternative gibt es die Lithium-Polymer-Zelle (*Li-Po*). Dabei kommt ein fester oder gelförmiger Elektrolyt zum Einsatz. Dies erlaubt die Nutzung einer metallbeschichteten Plastikfolie als Gehäuse und begünstigt dadurch eine quaderförmige Bauweise. Bei Lithium-Technologien kommen ausschließlich wasserfreie, organische oder anorganische Elektrolyten zum Einsatz. Eine wesentliche Herausforderung dabei ist, dass der Elektrolyt bei den hohen Potenzialdifferenzen nicht zersetzt wird.

2.5.3.2 Eigenschaften

Lithium-Akkus zeichnen sich vor allem durch die hohen Energiedichten (Typ. 140-180Wh/Kg) aus . Weiters besitzen sie hohe Zellspannungen. Somit kann die Versorgung moderner Low- Power-Elektronik meist durch eine einzelne Zelle erfolgen.

Zu beachten ist auch, dass Li-Ion Akkumulatoren keine reversiblen Nebenreaktionen aufweisen. Dies führt zu einer geringen Selbstentladung und hohen Speichereffizienz. Aufgrund der fehlenden Nebenreaktionen führt eine Überladung jedoch zur irreversiblen Beschädigung der Zelle.

Weiter werden mit Lithium-Zellen diverse Sicherheitsprobleme assoziiert. So besteht bei falscher Behandlung der Zellen Brandgefahr. Dies liegt an den hohen Energiedichten. Auch reagiert Lithium mit Wasser, wodurch ein Löschen eines Lithiumbrandes mit Wasser nicht zu empfehlen ist.

Neben der Problematik beim Überladen dürfen Lithium-Zellen auch nicht tief entladen werden. Bei Tiefentladung kommt es üblicherweise zu einer Zersetzung der Stromableiter.

Lithium-Akkumulatoren weisen eine Ruhespannung abhängig vom Ladezustand auf. Dies erlaubt einerseits die Abschätzung des Ladezustandes durch Spannungskriterien und andererseits das Laden durch das CC/CV-Verfahren (Punkt 2.6.5). Ein typischer Entladevorgang ist in Abbildung 2.8 zu sehen.

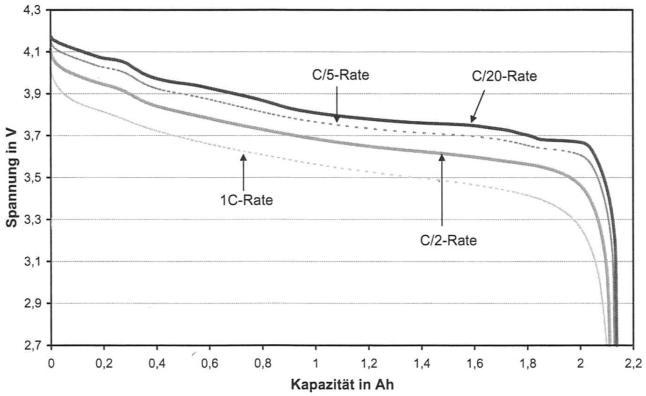


Abbildung 2.8: Entladevorgang einer LI-Ion Zelle bei unterschiedlichen Strömen. Bild aus [1]

2.5.3.3 Ladeverfahren

Bei Lithium-Technologien wird ein Konstant-Strom-Konstant-Spannung-Ladeverfahren (CC-CV) verwendet. Dieses wird später in Punkt 2.6.5 erklärt. Üblich sind dabei Laderaten von bis zu $1C$ bei Li-Ion. Für besondere Hochstromanwendungen kommen bis zu $4C$ zum Einsatz.

Als Ladeendspannung wird im Allgemeinen $4.2V$ verwendet, wobei eine Genauigkeit von $\pm 40mV$ gefordert wird.

Zu beachten ist, dass Lithium Akkus nicht überladen werden dürfen. Um ein Volladen aller Zellen zu ermöglichen ist ein Balancer nötig. Mehr dazu in Punkt 2.7.

2.5.3.4 Einsatzbereiche

Lithium-Akkus lösen in letzter Zeit vor allem die NiMh-Akkumulatoren ab. Besonders im Bereich der Consume-Elektronik und Mobile Devices sind nicht

mehr weg zu denken. Im Kfz-Bereich haben Lithium-Akkumulatoren für Hybrid- und Elektro-Fahrzeuge die NiMh-Akkumulatoren bereits abgelöst.

Auch kommen bei Anwendungen mit hohem Stromverbrauch, wie Taschenlampen, immer häufiger 18650er Lithium Rundzellen als Ersatz für die klassischen AA und AAA NiMh-Zellen zum Einsatz.

2.5.3.5 Alterung

Ein Nachteil von Lithium Technologien ist, dass es zu einer kalendarischen Alterung kommt. So liegt die Lebensdauer oft nur bei etwa 5 Jahren. Dabei wird die Alterung vor allem durch hohe Temperaturen und hohe Ladezustände beschleunigt.

Bei zu niedrigen Ladezuständen kommt es zur Korrosion der Elektroden. Aus diesem Grund wird bei längerer Lagerung empfohlen für die Kompensation der Selbstentladung ausreichend Ladungsreserve vorzusehen.

Bei hohen Spannungen kann es zum langsamen Zerfall des Elektrolyten kommen. Auch wenn ich dafür keine Angaben gefunden habe, gehe ich davon aus, dass dies ein wesentlicher Faktor für die kalendarische Alterung ist.

Als Ursache für die begrenzte Zyklenzahl ist dabei vor allem der mechanische Stress durch Volumenarbeit der Elektroden. Bei der Ladung/Entladung kommt es zur Ausdehnung der Elektroden von typisch 9.2% bei Graphit (5% bis zu 400% bei manchen Lithium-Legierungen), die aus diesem Grund nur für Primärbatterien zum Einsatz kommen.

Weiters entsteht an der Grenzschicht von Elektrode und Elektrolyt ein SEI-Film (Solid Elekctrolyte Interface). Diese Deckschicht bremst den Transport von Lithium-Ionen und erhöht somit den Innenwiderstand der Zelle.

2.5.4 Weitere Technologien

Neben den hier im Detail erwähnten Technologien gibt es auch noch ein Reihe anderer Technologien.

Rechargeabel-Alkali-Manganese (RAM)-Zellen, also wiederaufladbar Alkali Mangan Zelle, ist eine Sonderform der Alkali-Mangan-Primärbatterie die einige Ladezyklen bietet. Die Anzahl der Ladezyklen hängt dabei stark von der Zyklentiefe ab.

Nickel-Zinn(NiZn) ist eine andere Nickel basierte Technologie, die durch ihre im Vergleich zu NiMh höheren Zellspannung von $1.6V$ von Interesse ist. NiZn kommt momentan meist nur als wiederaufladbare alternative zu Alkaline Primehrbatterien zum Einsatz, dies ist besonders interessant wenn die Zellspannung

von rund $1.3V$ von NiMh nicht für den Betrieb eines Gerätes ausreicht. Bei aktuell verkauften Zellen liegt die Zyklenzahl jedoch noch deutlich niedriger als bei NiNh.

Doppelschichtkondensatoren, teilweise als Superkondensatoren bezeichnet, sind zwar technisch gesehen keine Batterien, besitzen jedoch einen ähnlichen Aufbau und zeigen in einigen Punkten ein ähnliches Verhalten. Sie erreichen meist Zellspannungen von rund $2.7V$. Die Energiedichte nähert sich zunehmend der von Akku Technologien. Doppelschichtkondensatoren bieten oft höhere Leistungsdichten und viel höhere Zyklenzahlen als Akkumulatoren. Auch sind teilweise Balancer-Schaltungen wie bei Li-Ion nötig. Der wesentliche Unterschied ist jedoch der lineare Zusammenhang zwischen Spannung und Ladung.

Weiters gibt es immer wieder neuen Entwicklungen und viele Technologien im Forschungsbereich diese sind zum Beispiel der Lithium-Luft-Akkumulator (Hohen energiedichten durch Nutzung des Luftsauerstoffs), oder Aluminium-Ion, Magnesium-Ion (Mehrere Elektronen pro Ion) [20]

2.6 Ladeverfahren

Die meisten Ladeverfahren können in zwei Teile unterteilt werden. Erstens in welcher Form wird die Ladeenergie zugeführt dies ist zum Beispiel Konstantstrom oder Konstantspannung. Der zweite Teil ist das Abschaltkriterium. Es entscheidet ab wann der Akku als voll geladen betrachtet wird.

Ein Ladeverfahren besteht logisch aus zwei Komponenten. Erstens in welcher Form wird der Strom / die Spannung angelegt. Zweitens das Abschaltkriterium, also wird der Ladevorgang beendet bzw. der Vollladezustand erkannt.

Einen guten Überblick über die gängige Ladeverfahren liefert Wikipedia [31]. Eine detailliertere Betrachtung mit mehr Fokus auf Implementierbarkeit für Ni* und Li* liefert [14]

Als Angabe für den Lade- und Entlade-Strom wird hier die C-Rate verwendet. Dabei ist $1C$ die Kapazität mal Stunde also $4300mA/h * 1h = 4.3A$. Für einen $2300mA/h$ Akku bedeutet eine Ladung mit $1/10C$ also einen Ladestrom von $230mA$. Die Angabe der C-Rate erlaubt eine Betrachtung weitgehend unabhängig von der Größe des Akkumulators.

2.6.1 Konstant-Strom (CC)

Konstantstrom laden ist eines der einfachsten Verfahren. Und stellt die Basis für viele andere Verfahren dar. Dabei wird der Akkumulator bis zum Ladeende mit einem Konstanten Strom beaufschlagt. Wesentlich dabei ist die Wahl des

Abschaltkriterium (Punkt 2.6.7). Für Ni-Cd/Mh wurde früher oft ein einfaches Zeitkriterium genutzt (z.B. 12h laden mit 1/10C). Dabei wird der Akku leicht überladen um sicher zu stellen dass der Vollzustand erreicht wird. Andere Abbruchverfahren sind z.B. Temperatur oder Spannung.

2.6.2 Pulsladeverfahren

Dabei erfolgt eine Ladung mit Konstantstrom. Der Ladestrom wird jedoch immer wieder unterbrochen. Je nach Anwendungsbereich können Stromstärke, Puls- und Pausen-Zeiten variieren. Eine wesentliche Anwendung ist dabei die Erhaltungsladung von Ni-Mh Akkus. Da bei diesen bei geringen konstanten Ladeströmen Dentrieten entstehen können, wir durch kurze Pulse mit hohem Strom der Vollladezustand erhalten. Dies beugt der Dentrietenbildung vor. Weiters wird es teilweise eingesetzt um Kosten für eine einstellbare Stromregelung zu sparen und den mittleren Ladestrom durch eine Pulsweitenmodulation zu realisieren. Ein weiterer Anwendungsbereich ist die kurzzeitige Unterbrechung der Ladung zur Messung der Ruhespannung.

2.6.3 Rückstromladen

Rückstromladen ist im Prinzip eine Erweiterung des Pulsladeverfahrens. Wobei auf jeden Ladeimpuls ein kurzer Entladeimpuls folgt. Dies soll die Sauerstoffbildung beim laden von NiMh Akkus reduzieren und dadurch eine schnellere Ladung ermöglichen [32].

2.6.4 Konstantspannung (CV)

Konstantspannungsladen ist ein sehr einfaches Verfahren, dass allerdings in reiner Form eigentlich nie zum Einsatz kommt. Das Problem dabei sind die extrem hohen Ladeströme die beim Anschluss von leeren Batterien zustande kommen können. Konstantspannungsladen kommt jedoch bei vielen Batterietypen als 2. Ladephase des CC-CV Verfahrens zum Einsatz.

2.6.5 Konstant-Strom und Spannung (CC-CV)

Dabei wird ein oberes Limit für Strom und Spannung verwendet. Als Resultat erfolgt zuerst eine Konstantstrom-Ladung und ab erreichen der Ladeschlusspannug eine Konstantspannungs-Ladung. Dieses Verfahren ist der Standard für Blei und Lithium Akkus. Ein Bespiel für einen Ladevorgang ist in Abbildung 2.9 zu sehen.

Bei billigen Geräten wird dies auch teilweise das Ladeverfahren durch eine Konstantspannungsquelle in Kombination mit einem hohen Innenwiederstand approximiert. Dies verlängert jedoch den Ladevorgang.

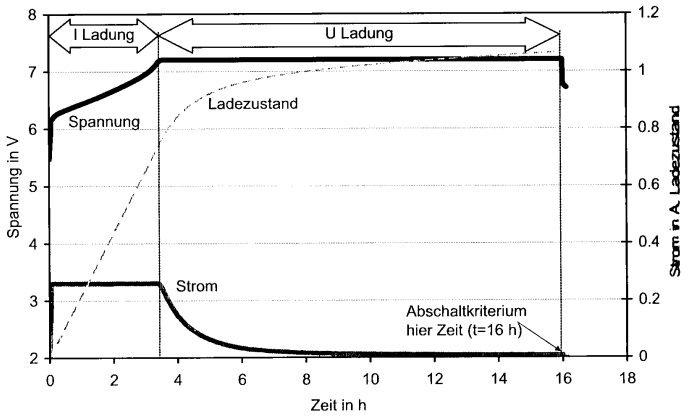


Abbildung 2.9: Batterie CCCV Ladevorgang. Bild aus [1]

2.6.6 Erhaltungsladung

Teilweise wird nach dem laden noch eine Erhaltungsladeung verwendet. Dies kann zum Beispiel eine Pulsladung mit Konstantstrom oder eine dauerhafte Ladung mit Konstantspannung sein. Durch ständiges laden besteht jedoch eine große Gefahr einer vorzeitigen Alterung. Und dies zu vermeiden wird z.B. bei Blei Akkus die angelegte Konstantspannung am Ende des Ladevorganges reduziert.

2.6.7 Abschaltkriterien

Die Aufgabe des Abschaltkriteriums ist zu entscheiden zu welchem Zeitpunkt der Akkumulator als voll geladen betrachtet wird. Daraufhin wird gegebenenfalls ein Wechsel in die Erhaltungsladung vorgenommen. Die Wahl des richtigen Abschaltkriteriums ist äußerst wichtig für die Lebensdauer der Akkus, und um Sicherheitsrisiken durch Überladung zu verhindern.

Das einfachste Abschaltkriterium ist ein einfaches Zeitkriterium. Dabei ist der Ladevorhang bei einer Ladung mit typischerweise 1/10 z.B. nach 10-14h beendet. Ein wesentliches Problem bei diesem Verfahren ist jedoch dass es beim Laden eines teilentladenen Akkumulators zu einer starken Überladung kommt.

Eine weitere Möglichkeit ist ein ladungsbasiertes Verfahren (Ladung basierend auf Charge-Faktor). Dabei ist es notwendig die entnommene Ladungsmenge des Akkus zu messen. Beim darauffolgenden Ladevorgang wird die entnommene

Ladung plus einer gewissen vom Ladefaktor bestimmten Ladereserve wieder nachgeladen.

Die Temperatur ist vor allem als Sicherheits-Abschaltung oft ein interessantes Kriterium. Es kann aber in seltenen Fällen auch als primäres Abschaltkritärium zum Einsatz kommen (z.B. für Ni-Mh). Dabei wird ausgenutzt dass bei erreichen des Vollladezustandes die gesamte geladene Leistung in Wärme umgesetzt wird, und es somit zu einem rapiden Temperaturanstieg kommt.

Das $-\Delta U$-Verfahren basiert darauf dass der Innenwiderstand, vor allem aufgrund einer verbesserten Ionenleitung, eines Ni* Akkus bei vollem Ladezustand sowie vor allem bei steigender Temperatur, sinkt. Die interne (chemische) Spannung bleibt jedoch gleich. Gesamt sinkt somit die äußere Zellenspannung bei konstantem Ladestrom leicht ab (NiCd: $\sim$ 30mV / NiMh: $\sim$ 10mV pro Zelle). Aus Sicherheitsgründen sollte dabei jedoch immer ein sekundäres Abschaltverfahren wie Temperatur oder Ladungsmenge zum Einsatz kommen.

Inflection-Point ist ein in ([14]) vorgestelltes Ladeverfahren. Es beruht auf dem gleichen Prinzip wie $-\Delta U$. Jedoch kommt es bei $-\Delta U$ immer zu einer leichten Überladung die dieses Verfahren zu reduzieren versucht. Dabei wird die 2. Ableitung der Spannung nach der Zeit gebildet und somit nach einen Wendepunkt in der Ladespannung gesucht. Dies erlaubt eine schnellere Erkennung des Vollladezustandes.

Die Verwendung von Akkupacks mit mehren Zellen stellt für die Abschaltkriterien teilweise ein Herausforderung dar. Da nicht alle Zellen gleichzeitig den Vollzustand erreichen, kann es zum Beispiel vorkommen dass $-\Delta U$ bei einem Akkupack den Vollladezustand nicht erkennt. Aus diesem Grund sind Sicherheitsmechanismen vorzusehen. Besonders bei Batterietypen wie Li-Ion die Überladung nicht bzw. nur schlecht vertragen müssen daher die einzelnen Zellspannungen überwacht werden.

2.7 Cell Balancing

Beim Cell-Balancing werden die Ladezustände der einzelnen Zellen eines Akkupacks einander angeglichen. Dies geschieht meistens während des Ladevorgangs. Primäres Ziel ist es ein Überladung der einzelnen Zellen zu verhindern. Cell Balancing wird meist nur bei Lithium Akkumulatoren verwendet, da bei diesen bei Überladung Explosionsgefahr droht. Weiters führt Überladung zur rapiden Alterung der Zellen. Eine Überladung führt aber auch bei den meisten anderen Akkumulator Technologien zu einer Reduktion der Lebensdauer. Eine wesentlich Ursache ist dabei der Abbau des Elektrolyten, zum Beispiel die Spaltung von wässrigen Elektrolyten in Wasserstoff und Sauerstoff. Problematisch dabei ist dass meist die Zelle mit der niedrigsten Kapazität als erste den Volladezustand erreich und darauffolgend überladen wird. Dies führt zu einem zusätzlichen

Kapazitätsverlust dieser Zelle und somit zum vorzeitigem Ausfall des gesamten Packs.

Um die Notwendigkeit von Cell-Balancing zu verhindern wird bei einigen Anwendungen eine Lösung mit einer Paralellschaltung der Zellen bevorzugt.

2.7.1 Passives Balancing

Die heute am meisten verbreitete Variante ist das passive Balancing. Dabei wird die Zelle/n mit dem höchsten Ladezustand/änden entladen und der Akkupack weiter aufgeladen. Dies erlogt in dem meisten Fällen durch einen Schalttransistor und einen Leistungswiderstand (siehe Abbildung 2.10). Vorteil dieser Variante ist die Einfachheit und somit die geringen Kosten.

Dabei wird beim Laden die überschüssige Energie einfach in Wärme umgewandelt. Dies wirkt sich jedoch negativ auf den Wirkungsgrad des Systems aus. Der Balancer bietet meist einen geringeren Strom als der erreichbare Ladestrom. Somit ist es am Ende des Ladevorganges oft nötig den Ladestrom zu Reduzieren und somit den Ladevorgang zu verlangsamen um eine Überladung zu verhindern. Grund für diese Dimensionierung ist die hohe anfallende Verlustleistung beim passiven Balancing, aber auch die geringere Belastbarkeit der Anschlussleitungen des Balancers.

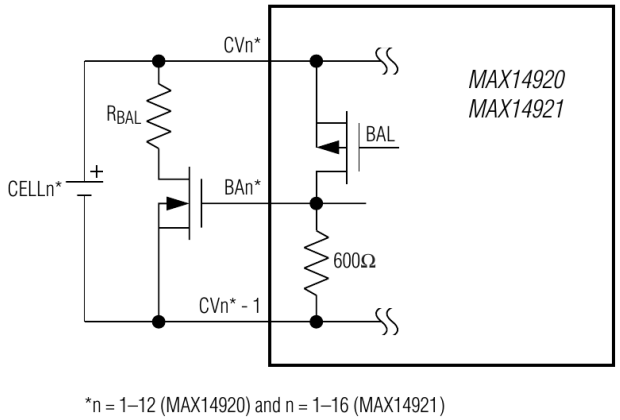


Abbildung 2.10: Schaltung einer einzelnen Zelle für Passives Balancing beim MAX14920 [17]

2.7.2 Aktives Balancing

Beim aktiven Balancing wird mit einer geeigneten Schaltung die Energie von einer Zelle in eine andere transferiert. Damit wird ein gleichmäßiger Ladezustand aller Zellen erreicht. Dies kann sowohl mit Kondensatoren als auch mit Induktivitäten erfolgen. Neben der Erhöhung des Wirkungsgrades, beziehungsweise der

Reduktion der Abwärme, erlaubt aktives Balancing auch eine bessere Nutzung der Kapazität eines Akkupacks. Die Kapazität eines Akkupacks in *Ah* ergibt sich normalerweise aus der Kapazität der schwächsten Zelle. Ist diese leer, muss der Entladevorgang abgebrochen werden um die Zelle nicht zu beschädigen. Alle anderen Zellen können jedoch noch Restladung besitzen. Im Fall von aktivem Balancing ist es jedoch möglich die leere Zelle mit Energie aus anderen Zellen wieder aufzuladen. Theoretisch ist es auf diese Art möglich die gesamte Energie des Akkupacks zu entnehmen. Dieser Zusammenhang wurde versucht in Abbildung 2.11 auf der nächsten Seite zu verdeutlichen.

Je nach Schaltungsvariante kann der Energietransfer zwischen unterschiedlichen Teilen des Akkupacks erfolgen. Man unterscheidet:

- **Cell to Cell Balancing:** Dabei erfolgt der Transfer der Energie zwischen unterschiedlichen Zellen des Akkupacks.
- **Cell to Pack:** Dabei werden einzelne Zellen entladen und die Energie der Zelle wird verwendet um den gesamten Akkupack zu laden.
- **Pack to Cell:** Dabei wird der Akkupack entladen und die Energie in eine einzelne Zelle transferiert.
- **Cell,Pack Bidirectional:** Das ist die Kombination von Cell to Pack und Pack to Cell.

Eine genaue Diskussion ist in [24] zu finden.

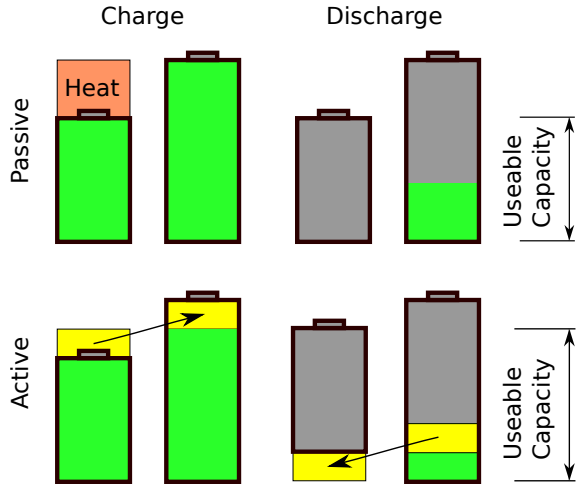


Abbildung 2.11: Vergleich Aktives und Passives Balancing. Beim Passiven Balancing wird beim Ladevorgang die überflüssige Energie in Wärme umgesetzt und beim Entladen verbleibt eine Restladung in einer der Zellen. Beim Aktiven Balancing wird die überschüssige Energie in die Zelle mit der höheren Kapazität übertragen. Beim Entladen wird die Energie zurück in die schwächere Zelle übertragen, es ergibt sich eine höhere nutzbare Kapazität.

2.7.2.1 Kapazitives Balancing

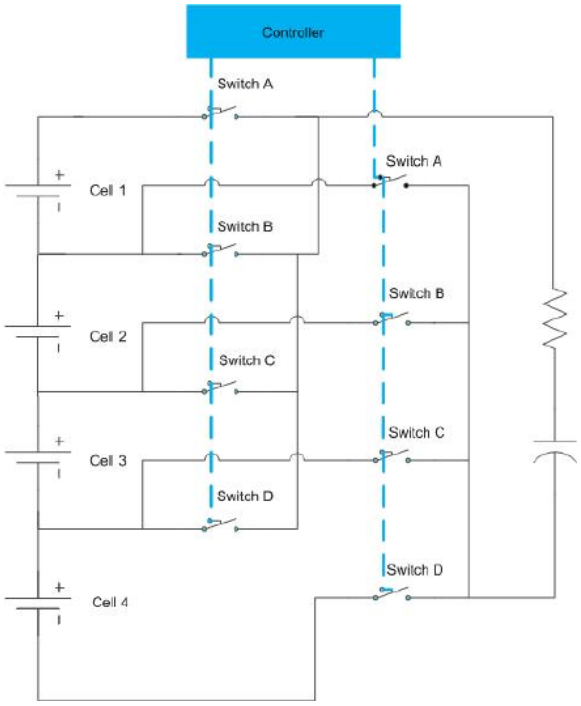
Kapazitives Balancing funktioniert durch abwechselndes Parallelschalten eines Kondensators mit unterschiedlichen Zellen des Akkupacks. Über die unterschiedlichen Zellspannungen erfolgt schließlich ein Ladungsaustausch. Ein angenehmer Effekt dieses Systems ist, dass keine Regelung des Balancers erforderlich ist, da der Stromfluss proportional zur Spannungsdifferenz der Zellen ist.

Eine sehr einfache Version ist in Abbildung 2.12a auf Seite 24 dargestellt. Dabei erfolgt der Energietransfer mittels eines einzelnen Kondensators, der mit den einzelnen Zellen des Akkupacks parallel geschaltet wird. Der Nachteil ist die komplexe Umsetzung der Schalter durch Halbleiter, da die Spannungsfestigkeit dem vollen Akkupack entsprechen muss und die Schalter in beide Richtungen sperren müssen.

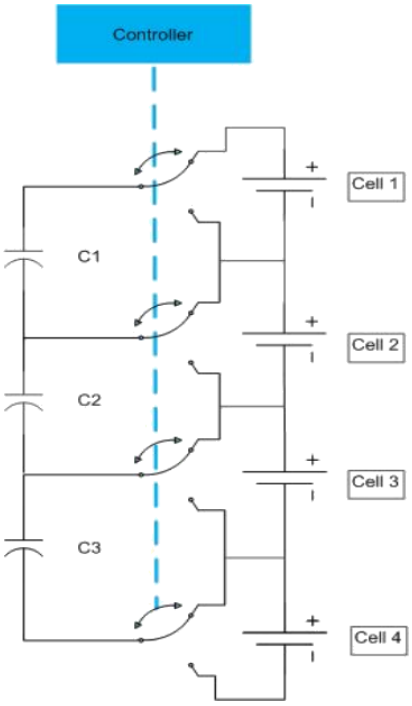
Eine alternative Architektur ist die Verwendung von mehreren Kondensatoren, siehe Abbildung 2.12b auf der nächsten Seite. Dabei erfolgt das Balancing immer nur mit der Nachbarzelle. Der Energietransfer über größere Strecken (zum Beispiel von Zelle 2 auf 5) erfolgt dabei nur indirekt. Vorteil dieser Schaltung ist jedoch, dass die Sperrspannung der Schalttransistoren nur einer Zellenspannung entsprechen muss. Weiters werden die Schalttransistoren immer nur mit einer Potentialdifferenz in einer Richtung beansprucht.

2.7.2.2 Induktives Cell Balancing

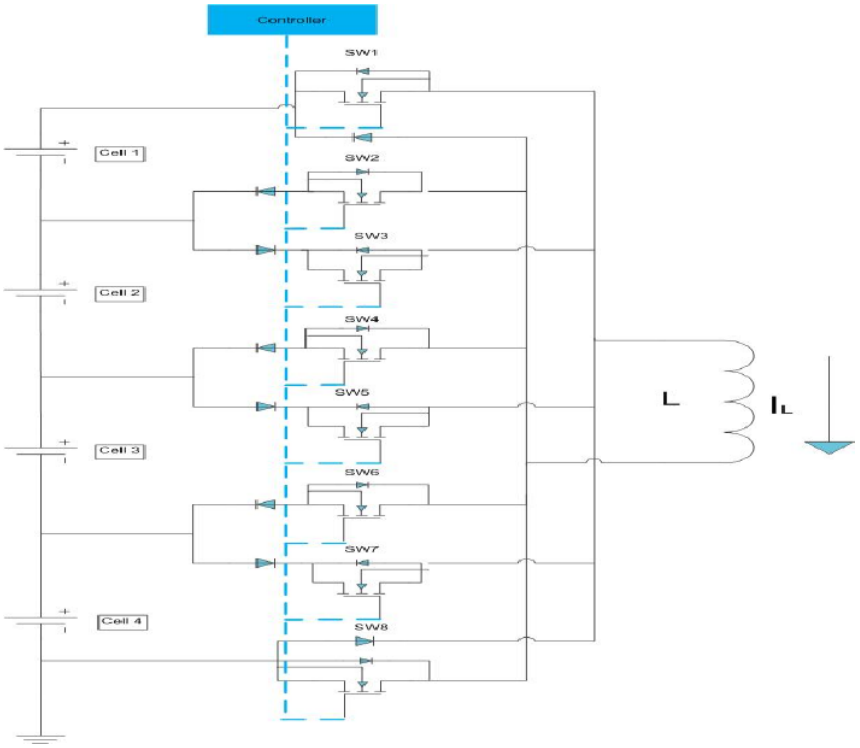
Beim induktiven Balancing gibt es eine Reihe von unterschiedlichen Ansätzen. Einige verwenden eine Speicherinduktivität, andere einen Transformator. Im Unterschied zum kapazitiven Balancing bietet sich hier nicht nur die Möglichkeit eines Balancings zwischen den Zellen, sondern auch ein Balancing mit dem gesamten Akkupack. Eine mögliche Schaltung für induktives Balancing ist in Abbildung 2.12c auf der nächsten Seite zu sehen.



(a) Beispiel für kapazitives Balancing mit einzelnem Kondensator. Bild aus [24]



(b) Beispiel für kapazitives Balancing mit mehreren Kondensatoren. Bild aus [24]



(c) Beispiel für induktives Balancing mit einer Speicherinduktivität. Bild aus [24]

Abbildung 2.12: Zusammenstellung einiger Schaltungen für aktives Balancing

3 Verwendete Technologien

Im folgenden Kapitel wird auf einige der verwendeten Technologien und Komponenten näher eingegangen.

3.1 Universal Serial Bus (USB)

Auch wenn USB aufgrund von Problemen momentan nicht zum Einsatz kommt, wurde versucht, die Kommunikation zwischen Steuercomputer und der Steuerplatine mittels USB zu realisieren. Bei der konkreten Umsetzung kam es leider immer wieder zu Problemen bei der Enummerierung durch den Linux-Host. Dabei wurde das Gerät grundsätzlich erkannt, jedoch ein Datentransfer war schlussendlich nicht möglich.

Einen guten Einstig in dem Bereich USB gibt der Artikel USB Grundkurs [29]. Für tieferer Einblicke in die Materie ist USBComplete [8] zu empfehlen.

3.1.1 USB Grundlagen

USB ist ein Master-/Slave-Bus. Dabei übernimmt der USB Host-Controller, meist Teil eines PCs, die zeitliche Steuerung des Buses. Der Datentransfer erfolgt dabei jeweils zum oder vom Host. Ein direkter Client zu Client Transfer ist nicht möglich. Um mehrere Geräte an den Bus anzuschließen kommen HUBs zum Einsatz. Diese repräsentieren Knotenpunkte im Bus und funktionieren logisch wie Schalter, die zum jeweiligen Zeitpunkt ein Gerät mit dem Host verbinden.

3.1.1.1 Versionen und Profile

Der USB Standard ist inzwischen in etlichen Versionen von 1.0 - 3.1 verfügbar. Die neuen Versionen erweitern den USB Standard meist um neue Profile (Geschwindigkeitsstufen) und gewährleisten im Allgemeinen Rückwärtskompatibilität. Eine Zusammenfassung der USB-Profile und ihrer Datenraten ist in Tabelle 3.1 zu finden.

Ab Version	Profil	Datenrate
1.0	Low Speed	1,5 Mbit/s
1.0	Full Speed	12 Mbit/s
2.0	Hi Speed	480 Mbit/s
3.0	SuperSpeed	4000 Mbit/s
3.1	SuperSpeed+	9697 Mbit/s

Tabelle 3.1: Unterstützte USB-Profile und Geschwindigkeiten für unterschiedliche USB-Standards.

Bei der Angabe der Geschwindigkeit ist jedoch teilweise auf die Details zu achten. So unterstützt der verwendete Mikrocontroller `XMEGA128A3U` zwar USB 2.0 jedoch nur Full Speed, und damit nur $12 MBit$.

3.1.1.2 Endpunkte und Übertragungsmodi

Ein USB-Gerät verfügt über einen oder mehrere Endpunkte. Diese repräsentieren voneinander unabhängige logische Datenkanäle zum oder vom Gerät. Der Endpunkt 0 hat dabei eine Sonderbedeutung und wird zur Konfiguration des Geräts verwendet.

Jeder Endpunkt hat dabei einen Übertragungsmodus: Control, Interrupt, Isochronous oder Bulk. Mit Ausnahme des Control-Übertragungsmodus erfolgt die Datenübertragung unidirektional. Die Transferrichtung wird als IN (Gerät zu Host) und OUT (Host zu Gerät) angegeben werden. Für eine bidirektionale Kommunikation sind zwei Endpunkte notwendig.

Control-Transfer dient in erster Linie dem Endpunkt 0, kann aber prinzipiell auch für weitere Endpunkte verwendet werden. Dabei werden Nachrichten mit optionalen Daten übertragen. Es erfolgt eine beidseitige Bestätigung um eine fehlerfreie Übertragung zu gewährleisten.

Interrupt-Transfer ist für das Senden geringer Datenmengen zu unbekannten Zeitpunkten gedacht, zum Beispiel USB-Maus oder Keyboard. Da USB keine Hardware-Interrupts unterstützt erfolgt eine zyklische Abfrage des Gerätes mit einer fixen Abtastrate von mindesten 10ms/1ms/125μs für Low/Full/High-Speed.

Isochronous-Transfer erlaubt das Übertragen mit einer fixen Datenrate und damit auch das reservieren von Bus-Bandbreite. Steht dem Bus nicht ausreichend Bandbreite zur Verfügung, wird die Konfiguration abgebrochen. Dies ist für Anwendungen wie Audio-/Video-Streaming interessant. Es erfolgt jedoch keine sichere Übertragung.

Bulk-Transfer ist für die Übertragung größerer Datenmengen gedacht, dabei wird allerdings keine Bandbreite reserviert. Dafür wird eine korrekte Übertragung

sichergestellt. Dieser Modus erlaubt die höchsten Datenraten und wird von USB-Sticks und auch von USB zu Seriellwandlern verwendet.

3.1.1.3 Geräteklassen

Um nicht für jedes Gerät einen neuen Treiber schreiben zu müssen gibt es für bestimmte Anwendungen vordefinierte Geräteklassen. Im Rahmen dieser Arbeit und allgemein im Embedded Segment sind vor allem 3 Klassen interessant. Jede dieser Klassen hat eine definierte Konfiguration von Endpunkten.

Communication Device Class (CDC) wird klassisch von USB zu Seriellwandlern verwendet. Dabei wird eine virtuelle serielle Schnittstelle bereit gestellt. Optionen wie die Einstellung von Baudraten, Parity-Bit, und Stop-Bits sind möglich, müssen jedoch in einem USB-fähigen Mikrocontroller nicht zwingend beachtet werden. Der wesentliche Vorteil liegt in diesem Fall an der einfachen Implementierbarkeit auf der Softwareseite des PCs, da der Zugriff auf serielle Schnittstellen in den meisten Fällen einfach möglich ist.

Human Interface Device (HID) ist die typische Deviceklasse für Mäuse und Tastaturen. Dies kann interessant sein um direkt Texteingaben am PC zu simulieren. Im konkreten Fall existiert ein Ziffernblock am Steuergerät, das auf diese Weise die Eingaben an die Steuersoftware übergeben kann.

Composite Device ist eine Deviceklasse die verwendet wird, falls ein USB-Gerät mehr als eine Klasse implementieren möchte. So würde sich im Idealfall das Controllerboard als zwei CDC- und ein HID-Device (Tastatur) registrieren. Dies hätte sowohl die normale Kommunikation CDC1, die Kommunikation mit einem externen NFC-Reader CDC2 und der Tastatur HID über ein einziges USB Kabel ermöglicht.

3.1.1.4 Power

USB hat zusätzlich zu seinem differentiellen Leitungspaar auch noch zwei Versorgungsleitungen. Mit diesen ist es möglich Geräte über USB zu versorgen.

Dabei wird zwischen Bus-Powered und Self-Powered unterschieden. Ist ein Gerät Self-Powered bezieht es seine Spannungsversorgung von einer externen Quelle, die Busspannung von 5V wird nur verwendet, um zu erkennen ob ein Host angesteckt ist.

Bus-Powered Geräte beziehen ihren Strom direkt über das USB-Kabel. Dabei sind für USB 2.0 bei 5V bis zu 500mA möglich. Zu beachten ist hierbei aber, dass in manchen Buszuständen, wie im Standby, weniger Strom zur Verfügung steht beziehungsweise darf das Gerät während der Initialisierung nur 100mA verbrauchen. Weiters erlaubt die USB-Spezifikation Spannungen im Bereich von

4.4V-5.25V am Ausgang eines passiven USB-Hubs. Dies reicht für klassische 5V Bausteine nicht aus. In diesen Fällen ist ein Boost-Konverter vorzusehen.

Details hierzu sind in [8] Kapitel 16 (Managing Power) zu finden.

3.1.1.5 Implementierung XMEGA

Der verwendete Mikrocontroller `ATXMEGA128A3U` besitzt einen eingebauten USB-Controller und Transceiver. Der USB-Anschluss kann direkt am Controller erfolgen, es werden nur Widerstände für die VBus-Detektion und optionale Schutzbeschaltung benötigt. Details dazu in der Application Note [2].

Weiters bietet Atmel im Rahmen seines Atmel Software Framework (ASF) [4] bereits Unterstützung für die gängigsten USB-Deviceklassen. CDC für die Emulation einer seriellen Schnittstelle [6], HID für die Emulation einer Maus oder einer Tastatur [5].

3.2 HD44780 kompatible LCD-Displays

Als Anzeige sind im Elektronik-/Mikrocontroller-Bereich einfache Text-Displays mit meist zwei Zeilen und 16-20 Zeichen Breite verbreitet. Dabei kommen fast ausschließlich Displays mit `HD44780` oder einem kompatiblen Controller zum Einsatz. Der große Vorteil dieser Displays ist, dass sie über einen internen Puffer verfügen und nur die anzuzeigenden Zeichen an das Display gesendet werden müssen. Es ist also keine komplexe und zeitkritische Ansteuerung nötig. Als Interface dient wahlweise ein 4 oder 8Bit breiter Bus mit einer Reihe von Steuerleitungen, die meist 5V Pegel verwenden. Obwohl der Bus grundsätzlich bidirektional ausgeführt ist kann die Ansteuerung unidirektional erfolgen.

Für Details empfehle ich einen Blick in das `HD44780`-Datenblatt oder eine kurze Online-Recherche.

3.3 RaspberryPi

Das RaspberryPi ist ein kleiner Einplatinen-Minicomputer, der sich vor allem im Hobby-Bereich sehr großer Beliebtheit erfreut. Da im Laufe der Entwicklung auf die zweite Version des Boards umgestellt wurde beziehen sich die folgenden Daten auf das RaspberryPi 2 Model B. Ausgestattet mit eine Quadcore ARM Cortex-A7 Prozessor mit $900Mhz$ erlaubt problemlos die Ausführung moderner Linux Distributionen inklusive grafischer Oberfläche. Den Kern der Platine stellt ein Broadcom `BCM2836` System on Chip. Weiters steht $1GB$ SD-RAM zur Verfügung. Als NVM-/HDD-Ersatz kommt ein Mikro-SD-Karte zum Einsatz. Als Schnittstellen stehen unter anderem USB, SPI, HDMI, I^2C, Ethernet, Seriell, sowie eine Reihe von General purpose Input-/Output-Ports zur Verfügung. Als Schnittstellenspannung kommt $3.3V$ zum Einsatz. Darüber hinaus gibt es noch die Möglichkeit über einen CSI-Anschluss (Camera Serial Interface) eine Kamera sowie über einen DSI-Anschluss (Display Serial Interface) ein Display zu verbinden. Die Spannungsversorgung erfolgt über $5V$ DC, wahlweise über einen Mikro-USB-Anschluss oder über die Stiftleisten der Platine.

Die große Stärke der Platine ist einerseits der günstige Preis und andererseits die große Community, die sich gebildet hat.

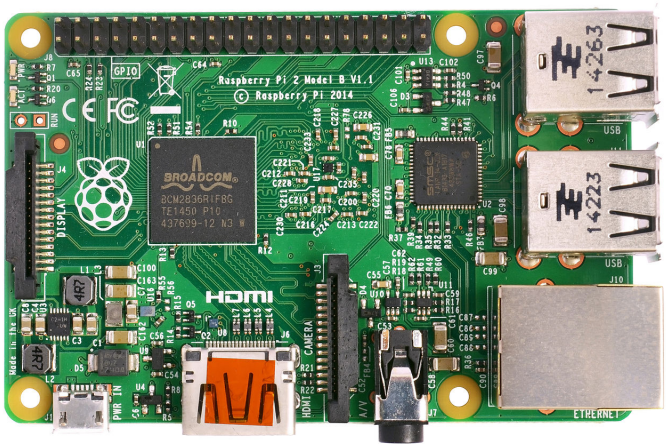


Abbildung 3.1: RaspberryPi 2. Image from `http://commons.wikimedia.org`

Display

Eine weiterer Aspekt ist, dass es seit einiger Zeit ein passendes Touch-Display für den RaspberryPi gibt. Dieses 7-Zoll Display mit $800x480$ Bildpunkten erlaubt die Erstellung moderner Benutzeroberflächen. Der Anschluss erfolgt dabei über den DSI-Anschluss. Der große Vorteil dieses Displays, im Vergleicht zu anderen über HDMI angeschlossenen Displays ist die etwas tiefere Integration und die damit verbundene kompaktere Bauweise. So ist der Umgang mit

dem DSI-Flachbandkabel einfacher als mit steifen HDMI-Kabeln und auch die Spannungsversorgung erfolgt über die selben $5V$ wie die des RasberryPis.

Leider ist die Qualität des getesteten Displays etwas mangelhaft. So löste sich bei Belastung mit dem Eigengewicht der Steuerplatine die Verklebung von Frontplatte (Touch) und LCD-Panel. Weiters war anfangs durch eine mangelhafte Ausrichtung des LCD-Panels auf der Frontplatte am Rand des Displays ein weißer Steifen des Backlights zu sehen.

4 Konzept und Design

4.1 Zielsetzung

Es soll ein modulares, scriptfähiges und leistungsfähiges Ladegerät konstruiert und aufgebaut werden. Damit soll es möglich sein neue Ladeverfahren einfach zu testen, Experimente im Bereich Spannungsversorgung und Batteriemanagement zu machen, sowie die Entwicklung der Akkus und Zellen über ihren Lebenszyklus zu verfolgen. Eine Logische Trennung zwischen der Steuerung und den Leistungsausgängen ist vorgesehen. Dies ermöglicht die Implementierung unterschiedlicher Module für spezielle Anwendungen.

Ziele:

- **Modularer Aufbau:** Es soll eine Trennung von Steuerlogik und den Leistungsausgängen stattfinden. Dies ermöglicht es an ein Steuermodul mehrere Leistungsmodule anzuschließen. Es soll auch möglich sein Leistungsmodule für unterschiedliche Anwendungen (Zellen Zahlen, Akku Technologien,...) zu bauen und diese gemeinsam zu betreiben.
- **Standard Ausgangsmodul:** Es soll ein Ausgangsmodul für das Laden von Akkupacks gebaut werden. Dabei ist auf die Anforderungen unterschiedlicher Akkutechnologien zu achten.
  - **Leistungsdaten:** Es soll möglich sein zumindest 6 Zellen Lithium- oder Blei-Akkus beziehungsweise 12 Zellen Nickel-Metallhydrid-Akkus zu laden. Als Ladestrom ist $10A$ wünschenswert, mindestens jedoch $5A$ zu erreichen. Weiters soll eine starke Entladefunktion vorgesehen werden. Hierbei liegt das Ziel bei $100W$ Entladeleistung bei einem maximalen Entladestrom von $10A$.
  - **Balancer:** Für die Ladung von Lithium-Akkus ist ein Balancer vorzusehen.
  - **Spannungsversorgung:** Als Spannungsversorgung ist ein externes Netzteil vorgesehen. Auf einem direkten Netzanschluss wird aus Sicherheits- und Komplexitätsgründen verzichtet.
  - **Geschwindigkeit und Regelbarkeit:** Der Lade- beziehungsweise Entladestrom soll schnell und in einem großen Dynamikbereich eingestellt werden können. Dies soll die Implementierung unterschiedlicher Lade- und Mess-Verfahren erlauben zum Beispiel Pulsladen, Reflexladen, 1kHz-Batterie-Impedanzmessung.

  - **Temperaturüberwachung:** Eine Möglichkeit zur Überwachung der Batterietemperatur ist vorzusehen.

- **Benutzerdefinierte Ladeverfahren:** Es soll dem Nutzer des Gerätes möglich sein eigene Ladeverfahren zu implementieren und zu verwenden. Eine Implementierung der Ladeverfahren in einer gängigen Skriptsprache ist vorgesehen.
- **Benutzerinterface:** Auf eine benutzerfreundliche Bedienung ist zu achten. Die Benutzung soll möglichst intuitiv erfolgen und gleichzeitig ein großes Maß an Kontrolle über das Gerät bieten. Die Ausgabe von Messwerten und Diagrammen von wesentlichen Batterieparametern ist vorzusehen.
- **Lautstärke:** Beim Betrieb des Ladegerätes soll unnötiger Lärm vermieden werden. Dies betrifft im Besonderen eine Regelung oder Abschaltung von Lüftern sowie das Vermeiden von pfeifenden Spannungsreglern.
- **Energieeffizienz:** Auf eine adäquate Energieeffizienz ist zu achten. Dies ist schon aufgrund der Reduktion der Abwärme sinnvoll.
- **Schutz des Gerätes:** Soweit möglich sollen Maßnahmen ergriffen werden um eine Beschädigung oder Funktionsstörung zu verhindern. Dabei ist speziell auf ESD, EMV sowie Verpolung und Kurzschluss zu achten.
- **Betriebssicherheit:** Das unkontrollierte Laden von Akkus kann zur Zerstörung der Akkus, Bränden oder sogar Explosion der Zellen führen. Es sind Schutzmechanismen zu integrieren, die das Risiko einer unkontrollierten Ladung minimieren.
- **Alternativlösungen berücksichtigen:** Im Laufe der Entwicklung treten immer wieder Probleme auf, beziehungsweise angestrebte Lösungsansätze funktionieren nicht wie erwartet. Um mit derartigen Situationen einfach umgehen zu können, sind Alternativlösungen mit zu berücksichtigen und gegebenenfalls vorzubereiten. Dies gilt im Besonderen für die Hardware-Komponenten, da hier Änderungen aufwändiger und langwieriger sind.

4.2 High-level Design

Im Rahmen des High-level Designs soll ein Schaltungskonzept erstellt werden. Um die Implementierbarkeit sicher zu stellen werden gleichzeitig mögliche Bauteile recherchiert.

4.2.1 Hardware versus Software

Aufgrund der höheren Flexibilität wird im allgemeinen möglichst viel in Software gelöst. Die Hardware stellt dabei die notwendige Basis zur Verfügung. Die Mikrocontroller-Software, die auf den Boards läuft, beinhaltet harte Echtzeitfunktionalität. Dieser Ansatz ermöglicht es die Hardware zu reduzieren. Höhere Logikfunktionen werden, soweit sinnvoll möglich, auf dem Steuercomputer gelöst,

da hier weniger Einschränkungen bei der Softwareentwicklung existieren und eine leichtere Fehlersuche und Fehlerbehebung möglich ist. Die Hardware stellt in vielen Fällen nur das Interface zum Mikrocontroller dar.

4.2.2 Allgemeines

Um ein modulares System zu erhalten wird erfolgt eine Trennung zwischen Steuer- und Bedien-Funktionalität sowie den Leistungsausgängen. Dabei kann ein Steuermodul mehrere Treibermodule ansteuern um mehrere Akkus gleichzeitig an zu schließen. Dieses System ist in Abbildung 4.1 veranschaulicht.

Um die Schaltungskomplexität möglichst gering zu halten kommt ein Mikrocontroller zum Einsatz. Dieser stellt das Zentrum der Hardware dar. Um den Entwicklungsaufwand zu reduzieren wird am Steuermodul und am Treibermodul derselbe Mikrocontroller eingesetzt. Eine Überdimensionierung wird dabei bewusst in Kauf genommen.

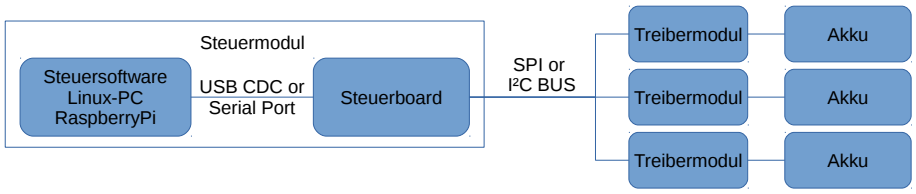


Abbildung 4.1: Moduldesign: Das Steuermodul besteht aus dem Steuercomputer und einem zusätzlichem Steuerboard. Über ein Businterface werden mehrere Treibermodule angeschlossen.

4.2.3 Spannungsversorgung

Bei der Spannungsversorgung ist logisch zwischen der Leistungsversorgung, also der Breitstellung der Ladeenergie für die Akkus und der Hilfsenergie zur Versorgung der Elektronik zu unterscheiden. Die Leistungsversorgung hat für jedes der einzelnen Module zu erfolgen. Die Hilfsenergie wird zwischen den einzelnen Modulen ausgetauscht. Der Anschluss der Spannungsversorgung (Hilfsspannungen) kann an einem beliebigen Modul erfolgen. Die höchste der Versorgungsspannungen wird als $VSupply$ an den Bus angelegt und an alle Module verteilt. $VSupply$ besitzt keine fix definierte Spannung, lediglich der Spannungsbereich (13-35V) muss eingehalten werden. Das Steuermodul leitet aus $VSupply$ nun eine $5V$- und $12V$-Hilfsspannung für die Treibermodule, sowie die Spannungsversorgung für den Steuercomputer ab. Um Störungen durch Spannungsschwankungen zu vermeiden wird die $3.3V$-Spannungsversorgung der einzelnen Module durch einen Linearregler von der $5V$-Versorgung abgeleitet. Die $3.3V$-Versorgung wird auch für Analogkomponenten verwendet und ist deshalb besonders empfindlich.

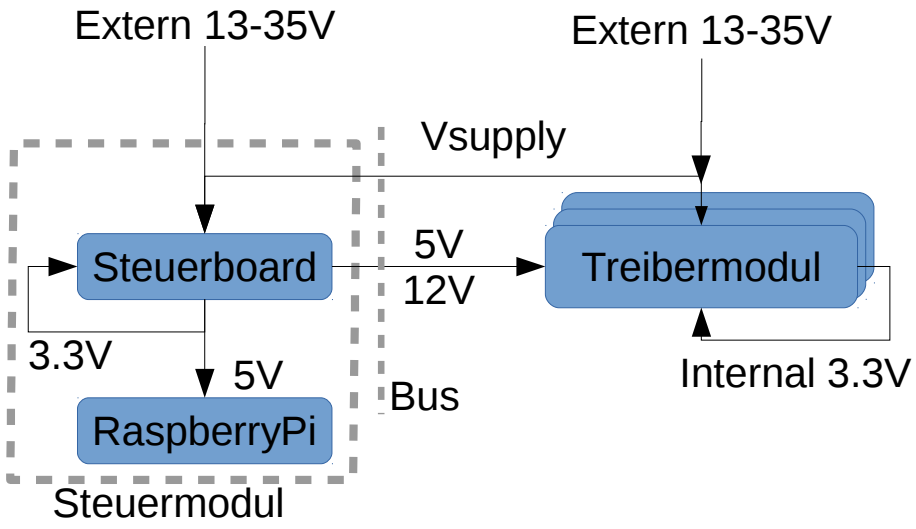


Abbildung 4.2: Überblick Spannungsversorgung (Hilfsspannung): VSupply ist die höchste der Versorgungsspannungen aller Module und wird über die Bus dem Steuermodul zu Verfügung gestellt. Das Steuermodul leitet daraus wiederum eine 5V und 12V Versorgung für die Treibermodule ab. Damit ist sichergestellt, dass bei Module ohne angeschlossene externe Spannungsversorgung die Steuerelektronik(Mikrocontroller) arbeitet und somit der Bus nicht beeinträchtigt wird.

4.2.4 Steuerboard

Das Steuerboard ist neben dem Steuercomputer der zweite wesentliche Teil des Steuermoduls. Es hat zwei wesentliche Aufgaben. Einerseits soll es die Verbindung zwischen Steuercomputer und den Treibermodulen herstellen, andererseits soll es die Spannungsversorgung des Steuercomputers übernehmen. Weiters wird auch eine Hilfs-Spannungsversorgung für die Treibermodule zur Verfügung gestellt. Dies erlaubt zumindest die Erkennung der Treibermodule ohne eigene Spannungsversorgung dieser. Aus Verlustleistungs- und Effizienz-Gründen werden die Versorgungsspannungen mittels Schaltreglern erzeugt. Weiters bietet das Steuermodul auch die Möglichkeit Tastatur, Drehgeber, NFC-Leser, Temperaturfühler sowie ID-Tokens anzuschließen. Dadurch ist es möglich die gesamte Hardware über einen einzigen USB-Port anzusteuern, wodurch weiters eine einfache Austauschbarkeit des Steuercomputers gewährleistet wird. Die Komponenten des Steuermoduls sind in Abbildung 4.3 auf Seite 36 dargestellt.

4.2.5 Treibermodul

Im Rahmen der Arbeit wird ein Treibermodul für Akkupacks entwickelt. Es implementiert einen Kanal des Ladegerätes. Dabei stellt es sowohl die Spannung als auch die Messfunktionalität zur Verfügung. Im Zentrum des Treibermoduls steht der Mikrocontroller, der alle Steuer- und Messaufgaben übernimmt. Dabei werden soweit als möglich interne Peripherieeinheiten des Mikrocontrollers

genutzt. Für den eigentlichen Ladevorgang ist eine Kombination aus einem Vorregler und einer Stromquelle verantwortlich. Der Vorregler ist ein Schaltregler und erzeugt eine Hilfsspannung, die leicht über der des Akkupacks liegt. Die Stromquelle selbst wird als lineare Stromquelle implementiert. Der Schaltregler reduziert die Verlustleistung gegenüber einer reinen Linearregelung dramatisch. Die Linear quelle bietet eine bessere Dynamik als eine direkte Regelung des Ausgangsstromes mittels eines Schaltreglers. Zum Entladen wird eine Stromsenke verwendet. Diese wird ebenfalls als Linearregler implementiert. Dabei ist auf die angestrebte Verlustleistungen von $100W$ zu achten. Der tatsächliche Lade- und Entladestrom wird zusätzlich vom Mikrocontroller gemessen.

Balancing und Zellspannungsmessung sind weitere Kernkomponenten. Dabei gibt es unterschiedliche Implementierungsoptionen, kombinierte Mikrochips, sowie selbst aufgebaute Lösungen für die jeweiligen Aufgaben.

Für die Temperaturüberwachung sind drei Systeme vorgesehen. Ein einfacher NTC-$10k$-Fühler, der nur geringe Genauigkeit bietet. Ein 1-Wire-Temperatursensor wie der `DS18B20`, der aufgrund seiner Unique-ID zusätzlich als Identifikations-Token für den Akku dienen kann. Als Option ist zusätzlich noch ein I^2C Anschluss für ein Infrarot-Thermometer `MLX90614` vorgesehen, dieser könnte als einfacher Sensor zur Überwachung auf einen AkkAupack gelegt werden.

Als kleine Anzeige wurde ein $2x16$-Zeichen LCD-Display vorgesehen. Dieses kann den aktuellen Status des Moduls unabhängig vom Steuermodul anzeigen. Weiters sind noch Status-LEDs sowie ein Button vorgesehen. Die geplante Funktionalität ist das Selektieren des Treibermoduls auf Knopfdruck als aktives Modul am Controller. Die LEDs zeigen an ob das Treibermodul aktiv ist, der Ausgang aktiviert ist, weiters gibt es eine Reserve-LED.

Für die Kommunikation ist natürlich auch die Bus-Verbindung zum Steuermodul zu implementieren. Es ist die Option einer direkten USB-Verbindung zum Steuer-PC, ohne das Bussystem zu verwenden, vorgesehen, wenn auch in diesem Fall Probleme durch die fehlende Spannungsversorgung $5V/12V$, die sonst über den Bus läuft, auftreten würden.

Die Komponenten des Treibermoduls sind in Abbildung 4.4 auf der nächsten Seite graphisch dargestellt.

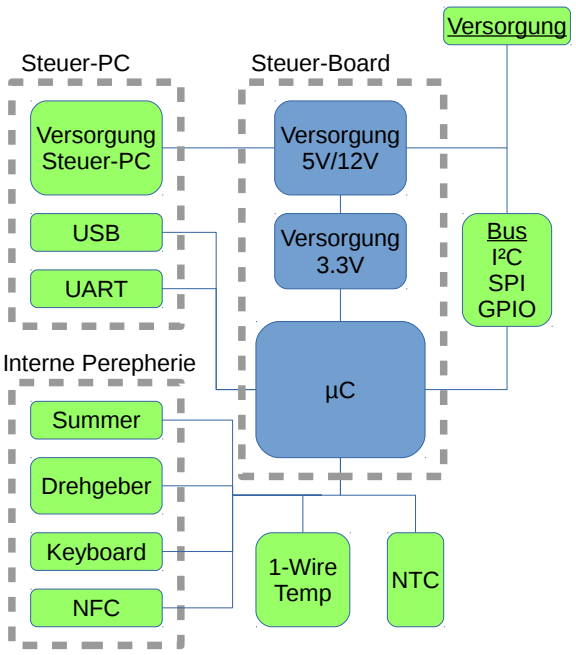


Abbildung 4.3: Highlevel-Design des Steuerboards. Die in grün dargestellten Komponenten sind in erster Linie Anschlüsse, die nur geringen Schaltungsaufwand erfordern.

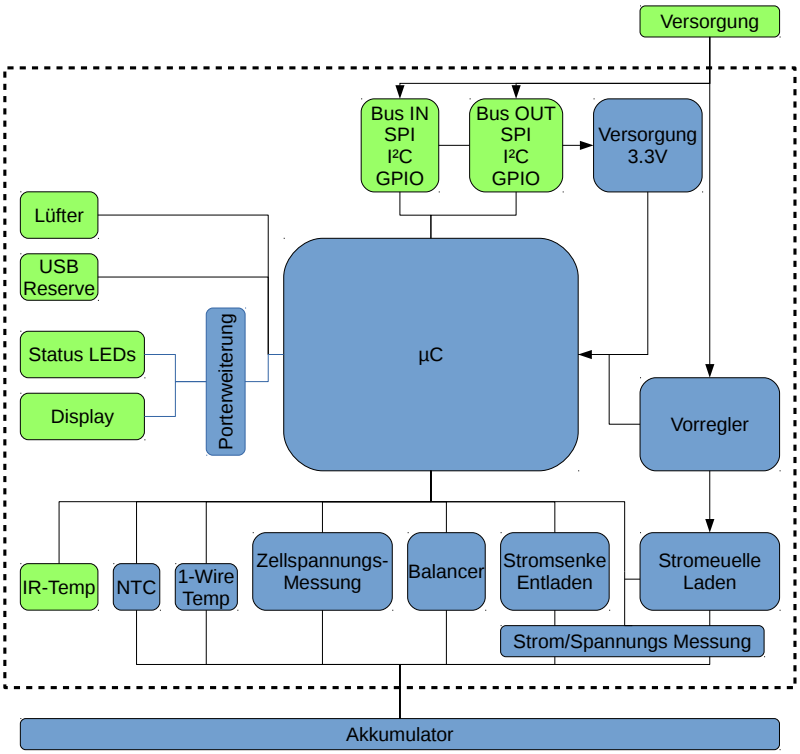


Abbildung 4.4: Highlevel Design des Treibermoduls

5 Implementierung

Im Rahmen diese Kapitels erfolgt die Entwicklung der Hardware basierend auf den Vorgaben aus dem Design. Als erstes werden allgemeine Themen die sowohl das Steuer- als auch das Treibermodul betreffen abgehandelt. Der Fokus des dieses Kapitels liegt bei den Schaltungsdetails der beiden Module.

5.1 Allgemein

im folgenden werden die Komponenten wie Mikrocontroller und Businterface, die in Steuer- und Treibermodul zu Einsatz kommen, diskutiert.

5.1.1 Mikrocontroller

Der Mikrocontroller stellt einen zentralen Baustein beider Module dar. Weiter soll, um die Entwicklung zu vereinfachen, auf beiden Modulen der gleiche Mikrocontroller zum Einsatz kommen. Die wesentlichen beiden Anforderungen sind ein möglichst hochauflösender ADC und eine Reihe von Schnittstellen für die Kommunikation zwischen den Modulen sowie zur Ansteuerung anderer Komponenten.

Die Entscheidung fiel auf den Mikrocontroller `XMEGA128A3U` [7] von Atmel. Dieser besitzt einen 8 Bit RISC-Prozessor mit 32 MHz, ist in einem 64 Pin TQFP Gehäuse und bietet 8 kByte RAM. Damit gehört er zu den stärksten 8 Bit Mikrocontrollern. Weiters hat er überdurchschnittlich viel und leistungsfähige interne Peripherie. Ein wesentliches Entscheidungskriterium ist, dass bereits Erfahrung mit diesem Mikrocontroller und der Atmel AVR Plattform existieren.

Peripherie:

- 2 stk. 12 Bit ADC mit 2 MSamle/sec mit 16 Kanälen in Kombination.
- 7 USARTs
- 2 I^2C Module
- 3 SPI Module
- 2 Kanal 12 Bit DAC
- 7 16 Bit Timer
- USB Full-Speed (12 MBit) inklusive der Bus-Treiber

Trotz der reichlich verfügbaren Peripherie kommt es zu Engpässen bei den SPI und USART Modulen.

5.1.2 Simulation

Das Aufbauen einer Schaltung zu Testzwecken ist mitunter zeit- und kostenintensiv. Die zunehmende Integration der Schaltungen und die damit verbundenen immer größer werdenden Pin-Zahlen, sowie der Umstieg auf SMT und der damit verbundene Schwenk weg von Steckbrettern, erschweren schnellen Aufbau von Testschaltungen. Dadurch ergibt sich oft nur die Möglichkeit der direkten Fertigung von PCBs.

Eine Alternative zur direkten Fertigung stellt die Simulation dar. Im Besonderen hat sich Spice als Industriestandard bei der Simulation durchgesetzt. Spice selbst ist ein reiner Simulator und besitzt weder eine grafische Bedienoberfläche noch Bauteilbibliotheken. Linear Technology bietet mit LTSpice [21] einen kostenloses Programmpaket, das Simulationsmodelle für die meisten ihrer Bauteile enthält. Weiters lassen sich für eine Vielzahl von anderen Bauteilen Spice und somit LTSpice kompatible Simulationsmodelle finden.

LTSpice bietet die Möglichkeit Subcircuits zu Zeichnen und in anderen Schaltungen wieder zu verwenden. Dieser Mechanismus wurde verwendet um eine hierarchische Simulation zu erlauben. So kann das Ladegerät in unterschiedliche Testschaltungen eingebunden werden. Weiters können im Ladegerät einfach Module ausgetauscht werden um z.B. den Schaltregler durch eine einfach Stromquelle zu ersetzen und damit die Simulationsgeschwindigkeit zu erhöhen.

5.1.3 Bus Interface

Das Businterface stellt die Verbindung zwischen dem Steuermodul und den Treibermodul her. Für die Kommunikation wurden als zwei Optionen I^2C und SPI implementiert. Aktuell wird jedoch nur SPI verwendet.

5.1.3.1 I^2C

I^2C hat den Vorteil, dass es von Haus aus als Bussystem konzipiert ist. Es besitzt bereits ein Adressierungsschema, eine Bus-Arbitrierung und ein Nachrichtenformat. Es ist allerdings als Bussystem innerhalb einer Platine vorgesehen und besitzt eine Open-Kollektor Treiber-Struktur. Somit ergibt sich eine geringe Störfestigkeit. Weiters ist eine dynamische Adressvergabe nicht im Standard gelöst. Zu diesem Zweck ist im Design eine eigene Board-Enable-Leitung (GPIO-CHAINED) vorgesehen. Damit kann sichergestellt werden, dass immer nur ein einziges, noch

nicht konfiguriertes Gerät am Bus aktiv ist und dieses ausgehend von Adresse 0 Konfiguriert wird.

5.1.3.2 SPI

SPI ist eigentlich nicht als Bus ausgelegt und hat aus diesem Grund immer einen aktiv getriebenen Ausgang pro Signalleitung. Dies erhöht die Störfestigkeit. Weiters ist die Signalrichtung für jede Leitung fixiert, was den Einsatz von Pegelwandlern und stärkeren Treibern vereinfacht. Um aus SPI einen Bus zu machen wird Daisy-Chaining verwendet. Jedes Treibermodul hat dabei ein 8 Bit breites Register. Im Rahmen einer Transaktion wird also ein Byte an jedes Modul gesendet und gleichzeitig ein Byte abgeholt. Abbildung 5.1 auf der nächsten Seite veranschaulicht diese Bus-Struktur. In der Software-Implementierung wird jede 1ms eine Transaktion ausgeführt. Dies ergibt eine konstante Bitrate für jedes Modul. Die logischen Übertragungskanäle sind voneinander unabhängig und die Module können sich gegenseitig nicht die Bandbreite wegnehmen. Jede SPI-Leitung hat eine Richtung und kann somit mit Push-Pull-Teibern verwendet werden. Dies führt zu einer höheren Störfestigkeit und ermöglicht höhere Datenraten. Aktuell wird die niedrigst mögliche SPI-Frequenz des Mikrocontrollers von 250kHz verwendet.

Momentan ist die maximale Anzahl von Treibermodulen in erster Linie durch die Treiberstärke auf den CLK und $/CS$ Leitungen begrenzt. Bei der aktuellen Hardware wird der Standardausgang des Mikrocontrollers verwendet. MOSI und MISO werden immer nur an das nächste Modul weitergeben und haben damit keine Probleme mit der Stärke der Treiber.

Durch den globalen $/CS$ erfolgt eine synchrone Taktung aller Module und auch ein synchroner Datentransfer. Diese Eigenschaft ist für diese Arbeit nicht näher von Bedeutung, könne aber bei anderen Applikationen, die eine Triggerung/Synchronisierung der Slaves erfordern, von Vorteil sein.

Da jedes Modul nur 1 Byte/ms empfängt, kann die Software Reaktionszeit relativ hoch bleiben. Es verbleibt der Mikrocontroller-Software die gesamte Bus-Idle-Zeit zwischen zwei Transaktionen für einen Austausch der Daten im Schieberegister des SPI-Moduls. Der Rest wird von der Hardware erledigt. Die Software-Triggerung erfolgt mittels Port-Interrupt auf die CS-Leitung.

Ein wesentlicher Nachteil dieses Systems ist, dass jeder Client in jeder Transaktion Daten übertragen muss. Dies wird gelöst, indem die Clients Null-Bytes senden, falls sie nichts übertragen wollen. Der Paketstart erfolgt dann wie üblich durch ein Start-Byte des Protokolls.

Über den Bus werden auch noch 5V und 12V vom Steuermodul zur Verfügung gestellt. Weiters kann über die V+ Leitung auch eine Versorgung des Steuermoduls durch die Treibermodule erfolgen. Die Masse-Leitungen wurden so platziert, dass

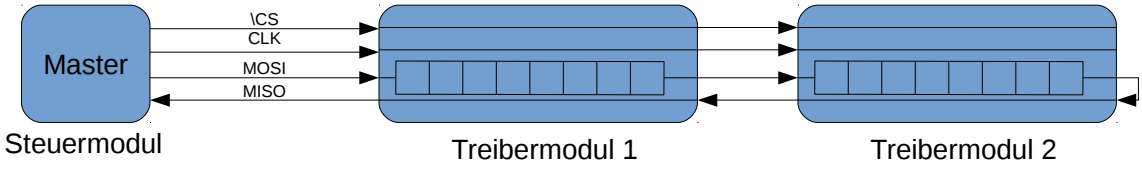


Abbildung 5.1: SPI-Interface Struktur: /CS und CLK werden an alle Module direkt weitergegeben. MOSI/MISO bildet eine Kette von Modulen.

eine Abschirmung zu den Datenleitungen entsteht. Die Vielzahl der Masseleitungen stellt sicher, dass die Masseverbindung in allen Fällen gewährleistet ist. Eine fehlerhafte Verbindung kann zu einem hohen Stromfluss über die Datenleitungen und somit zur Zerstörung das Gerät führen. Die Pinbelegung des Steckers ist in Abbildung 5.2 ersichtlich.

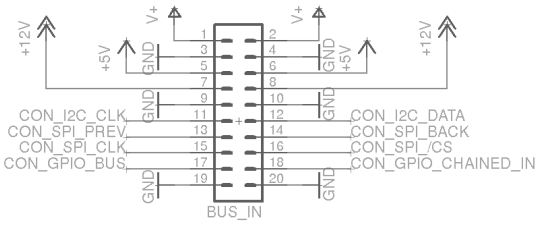


Abbildung 5.2: Pinbelegung des Bus-Steckers.

5.2 Steuermodul

Das Steuermodul besteht im Wesentlichen aus zwei Komponenten. Das sind die Datenübertragung zwischen den Modulen und die Erzeugung der Hilfsspannungen.

5.2.1 Spannungsversorgung

Für die Spannungsversorgung 5V/12V kommt ein `LT3514` von Linear Technologie zum Einsatz. Dabei handelt es sich um einen Drei-Kanal Step-Down Wandler mit internen Leistungstransistoren. Die Schaltung mit Voltage-Feedback mittels zweier Widerstände und einer Kombination aus Elektrolyt- und Keramik-Kondensatoren am Ausgang entspricht der üblichen Bauweise. Die Schaltung ist nach Datenblatt dimensioniert und simuliert. Abbildung 5.3 auf der nächsten Seite zeigt die Simulationsschaltung, Abbildung 5.4 auf der nächsten Seite zeigt

das das Simulationsergebnis. Bei der Dimensionierung erweisen sich die Induktivitäten als leicht problematisch in der Beschaffung.

Die 3.3V Versorgung für den Mikrocontroller besteht aus einem LM1117 Linear-Regulator und den zugehörigen Stützkondensatoren. Unabhängig vom Spannungsregler sind nahe der Versorgungspins einiger ICs noch 100nF Stützkondensatoren positioniert. Abbildung 5.5 auf der nächsten Seite zeigt die Schaltung.

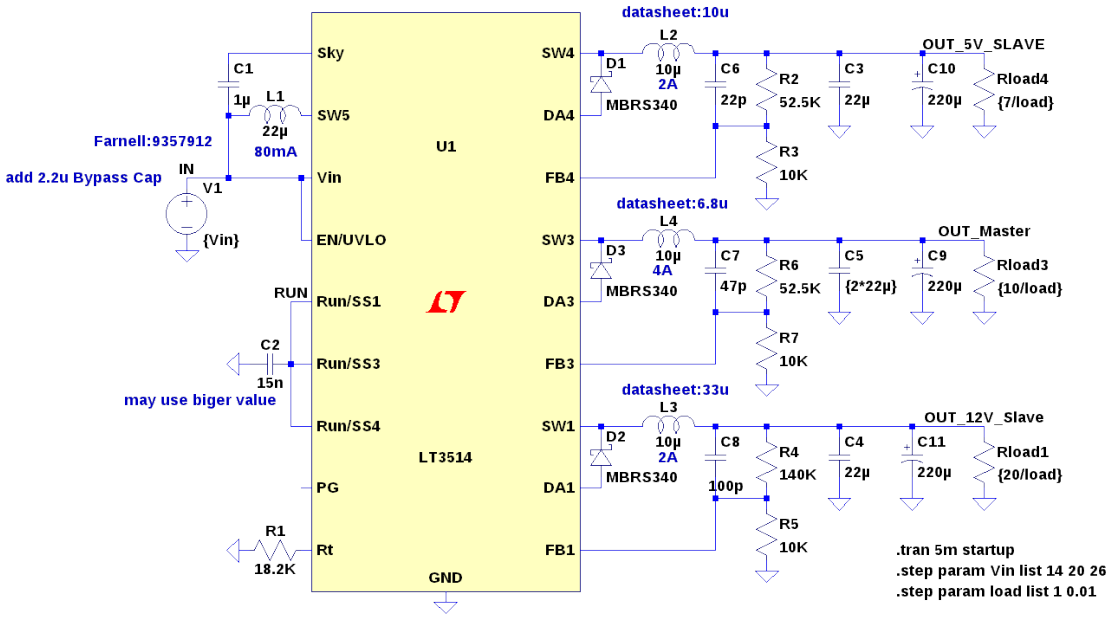


Abbildung 5.3: Simulationsschaltung für den LT3514

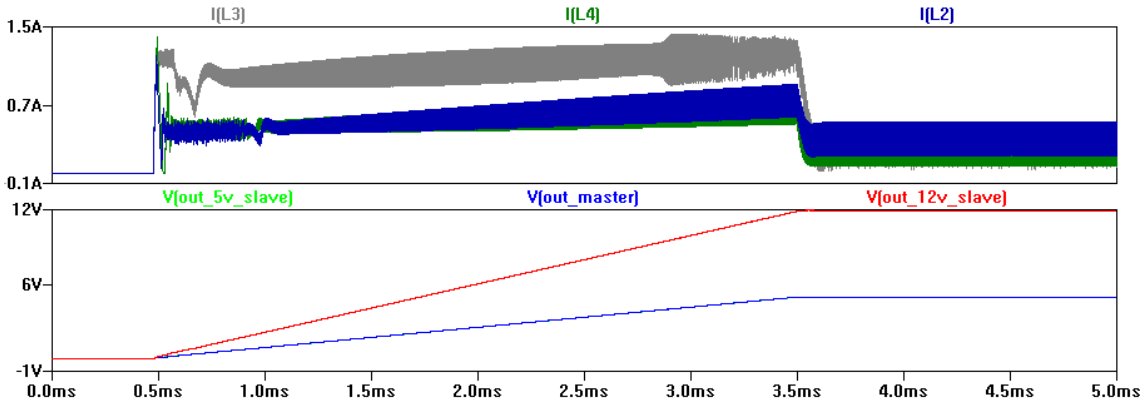


Abbildung 5.4: Simulationsergebnisse für anlaufen des den LT3514. Dabei sind die drei Ausgangsspannungen und Spulenströme dargestellt.

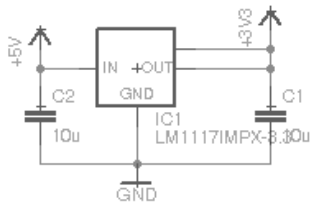


Abbildung 5.5: 3.3*V* Spannungsregelung basierend auf dem `LM1117`

5.2.2 Summer

Als Summer/Signaltongeber wird ein Piezo-Lautsprecher verwendet. Da dieser bei 3.3V keine ausreichende Lautstärke aufweist, wird er mit Hilfe eines FETs mit 12V angesteuert. Die Signalerzeugung erfolgt am Mikrocontroller mittels eines Timers, somit ergibt sich eine anpassbare Frequenz. Die Schaltung ist in Abbildung 5.6 zu sehen.

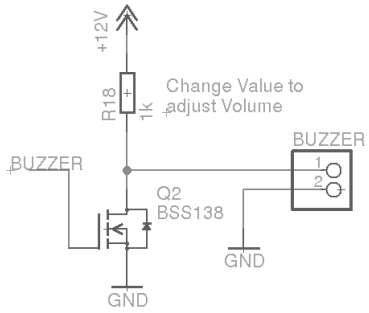


Abbildung 5.6: Schaltung des Piezo-Lautsprechers. Eine eingeschränkte Anpassung der Lautstärke ist durch die Variation des Widerstandes möglich

5.2.3 1-Wire Temperatur Sensor

Als digitaler Temperaturfühler kommt ein `DS18B20` [16] zum Einsatz. Dieser wird mittels 1-Wire-Interface angesteuert. 1-Wire kann dieselbe Leitung für die Versorgungs- und die Datenleitung verwenden. Da dazu Kommunikationspausen notwendig sind, um die Spannungsversorgung während des Messvorgenes sicherzustellen, wird dies nicht genutzt. Der Bus selbst funktioniert über eine Puls-/Pausenmodulation mit unterschiedlichen Pulslängen für logisch 0 und 1. Um dieses Signal am Mikrocontroller einfach zu erzeugen und zu messen wird ein USART-Modul des Mikrocontrollers verwendet. Die Implementierung basiert auf Applikation Note 318 [3] von Atmel. Die Schaltung ist in Abbildung 5.7 auf der nächsten Seite dargestellt. Der Anschluss des Sensors erfolgt über einen 3.5mm Klinkenstecker.

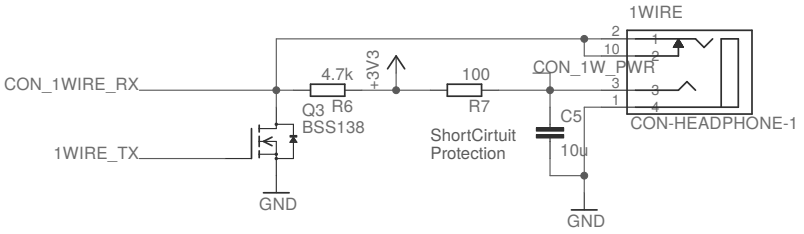


Abbildung 5.7: 1-Wire Anschluss für Temperaturfühler. *R6* und *Q3* stellen einen Open-Kollektor Ausgang zur Verfügung. *R7* verhindert einen Kurzschluss der Versorgung beim Einstecken des Klinkensteckens. *C5* stabilisiert die Versorgungsspannung

5.2.4 NTC-Temp

Als sehr einfache und kostengünstige Option zur Temperaturmessung wird ein Eingang für $10k\Omega$ NTC-Temperaturfühler eingebaut. Ein Spannungsteiler erzeugt dabei aus der 3.3V Versorungsspannung einen Pegel im 0-1V Bereich des ADCs. Als Anschluss wird wie beim 1-Wire Sensor ein 3.5mm Klinkenstecker verwendet. Die Schaltung ist in Abbildung 5.8 dargestellt.

Für die Umrechnung der ADC Werte auf Temperaturen werden Widerstandswerte eines der NTCs im Bereich von 8-95°C in einem Wasserbad ermittelt. Die Werte werden in ADC-Werte umgerechnet und mittels linearer Interpolation in einer Lookup-Tabelle gesucht.

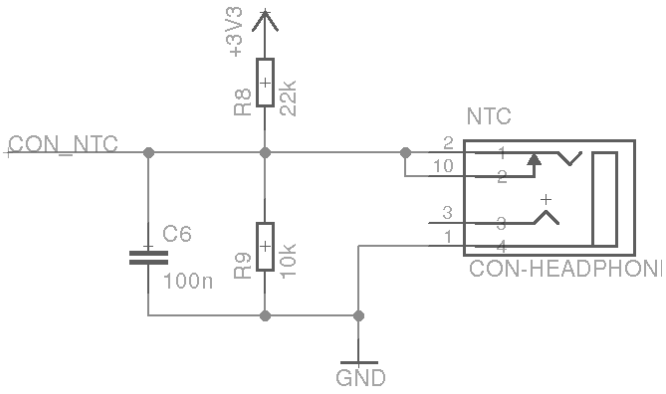


Abbildung 5.8: NTC-Temperatursensor

5.2.5 Electrostatic Discharge (ESD) Schutz

Zum Schutz vor elektrostatischen Entladungen kommen unterschiedliche Verfahren zum Einsatz. Digitale und analoge Schnittstellen am Mikrocontroller werden durch `ESD5V3U4RRS` [15] Schutzchip geschützt. Dieser bietet eine Kombination aus Ableitdioden und einer Supressordiode. Diese Schaltungsvariante führt zu

niedrigeren Leitungskapazitäten als die direkte Nutzung von Supressordioden oder Varistoren. Leider ist der verwendete Schutzchip für 5V-Anwendungen ausgelegt. Um diese Problem zu minimieren wird die Hiflsableitspannung auf 3.3V gelegt und mit Kondensatoren stabilisiert. Weiters werden Längswiderstände, die den Stromfluss in die zu schützende Schaltung begrenzen, verbaut.

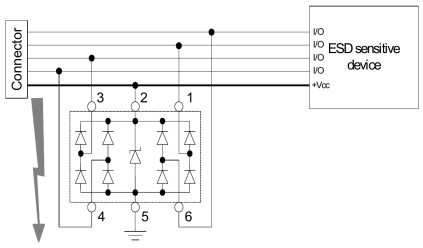


Abbildung 5.9: ESD Schutzchip `ESD5V3URRS`. Bild aus Datenblatt [15]

5.2.6 Inkrementalgeber

Der für das Verstellen von Zahlenwerten integrierte Inkrementalgeber besitzt eine Reihe von Schaltern, die gegen Masse gelegt werden. Theoretisch ist keine zusätzliche Beschaltung nötig, da eine Pullup-Eingangsstruktur im Mikrocontroller konfigurierbar ist. Um möglichen Problemen aus dem Weg zu gehen sind jedoch externe Pullup-Widerstände und Entprellkondensatoren vorgesehen. Weiters sind sowohl Anschlüsse für die Montage am PCB und einem externen Inkrementalgeber vorgesehen. Die Schaltung ist in Abbildung 5.10 auf der nächsten Seite zu finden. Die Auswertung am Mikrocontroller erfolgt wahlweise durch Port-Change-Interrupts oder Polling.

5.2.7 Keyboard

Als Keyboard kommt ein 16 Zeichen (4x4) Matrix Keyboard zum Einsatz. Der Anschluss erfolgt ohne weitere Komponenten an einem I/O-Port des Mikrocontrollers. Dabei werden 4 Pins als Eingänge mit Pulldown und 4 Pins als Push/Pull-Ausgängen am Mikrocontroller konfiguriert. Die Ansteuerung erfolgt zeilenweise. Es ist nicht möglich mehrere gleichzeitig gedrückte Tasten zu unterscheiden.

5.2.8 Universal Serial Bus (USB)

Für den USB-Anschluss wird einfach eine MicroUSB-Buchse direkt mit dem Mikrocontroller verbunden. Zur Erkennung der Busspannung wird ein Span-

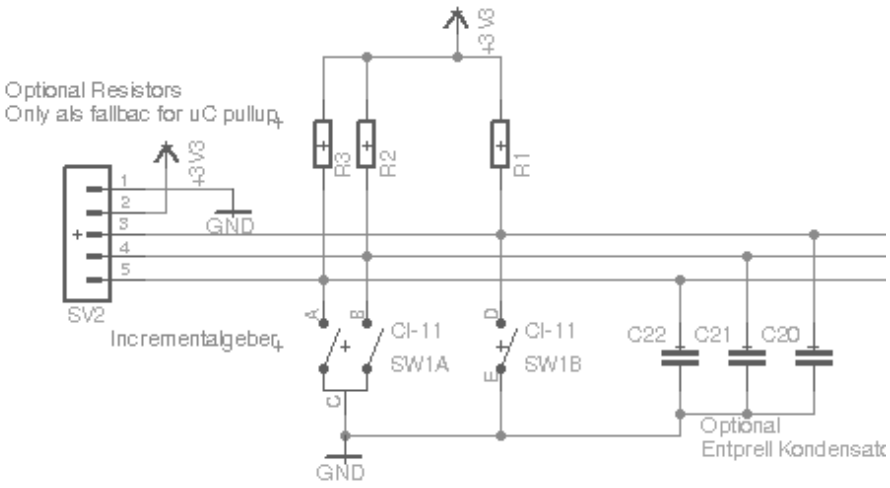


Abbildung 5.10: Schaltung Inkrementalgeber

nungsteiler von 5V auf 3.3V verbaut.

5.2.9 Weites

Es gibt eine Reihe weiter Schaltungsteile dir in erster Linie die Entwicklung vereinfachen.

- **Power LED:** Anzeige ob die Versorgungsspannung anliegt
- **Status LED:** Blinkt abhängig vom Betriebszustand des Mikrocontrollers
- **GND Jumper:** Erlaubt einfaches Anschließen von Messgeräten
- **Debug Header:** Erlauben einen einfachen Zugang zu den Busleitungen und somit den einfachen Anschluss eines Logic-Analysators zum Mitverfolgen des Bustransfers.
- **Reset Button:** Für den Neustart der Controller Software.
- **PDI Anschluss:** Für das Programmiergerät.

5.3 Treibermodul

Das Treibermodul stellt das komplexere der beiden aufgebauten Module dar. Es implementiert sowohl die Leistungselektronik für den Ladevorgang als auch die Messfunktionen zur Überwachung.

Um die Platinengröße zu reduzieren und die Modularität zu steigern, ist das Treibermodul in drei unterschiedliche Platinen aufgeteilt. Dies ist das Treiberboard, es beinhaltet den Vorregler, die Stromquelle und Stromsenke und einen Teil der Messfunktionalität. Weiters gibt es ein Balancerboard, welches die Messung der

Zellspannungen und das Ausbalancieren dieser übernimmt. Das Dritte ist das Steuerboard, dieses beinhaltet den Mikrocontroller das Businterface, die Temperaturfühler und die Ansteuerung des Displays. Die Aufteilung der Boards ist in Abbildung 5.11 dargestellt.

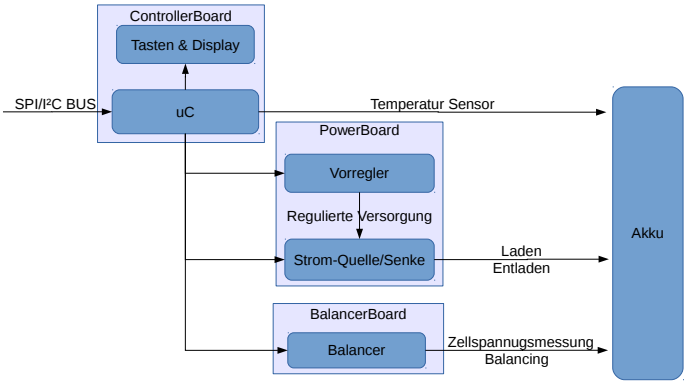


Abbildung 5.11: Platinen des Treiberdmoduls

Die Verbindung der Boards erfolgt über vier stapelbare 2x10 2.54mm Stiftleisten. Die Anordnung ist so gewählt, dass ein verdrehtes Einstecken sofort sichtbar wird. Um gegenseitige Störungen zu verhindern sind die Stiftleisten in logische Gruppen unterteilt. Top left (TL) beinhaltet die Spannungsversorgung, bottom left (BL) einige GPIOs, top right (TR) digitale Schnittstellen und bottom right (BR) analoge Schnittstellen.

5.3.1 Steuerboard

Auf dem Controller Board sitzt der Mikrocontroller, das Bus Interface sowie die Anschlüsse für Temperatursensoren und das Display. Der ESD-Schutz des 1-Wire-Interface und des NTC-Fühlers entspricht dabei dem des Steuermoduls. Es werden auch die IO-Ports für die weiteren Boards zur Verfügung gestellt.

Portexpander für Display und LEDs

Eine der wenigen Besonderheiten des Steuerboards ist die Ansteuerung des Displays. Dabei kommt ein `HD44780` kompatibles Display mit 2x16 Zeichen zum Einsatz. Da dieses mit 5V läuft ist eine Pegelumwandlung sinnvoll. Da nicht ausreichend IO-Ports für den Anschluss zur Verfügung stehen, wird das Display über eine Port-Erweiterung angesteuert. Dafür werden zwei `74HCT595` Serial-In/Parallel-Out Schieberegister mit TTL- kompatiblen Pegeln verwendet. Diese können problemlos durch die 3.3V Ausgänge des Mikrocontrollers angesteuert werden. Die freien Pins des Portexpanders werden auch für die Ansteuerung

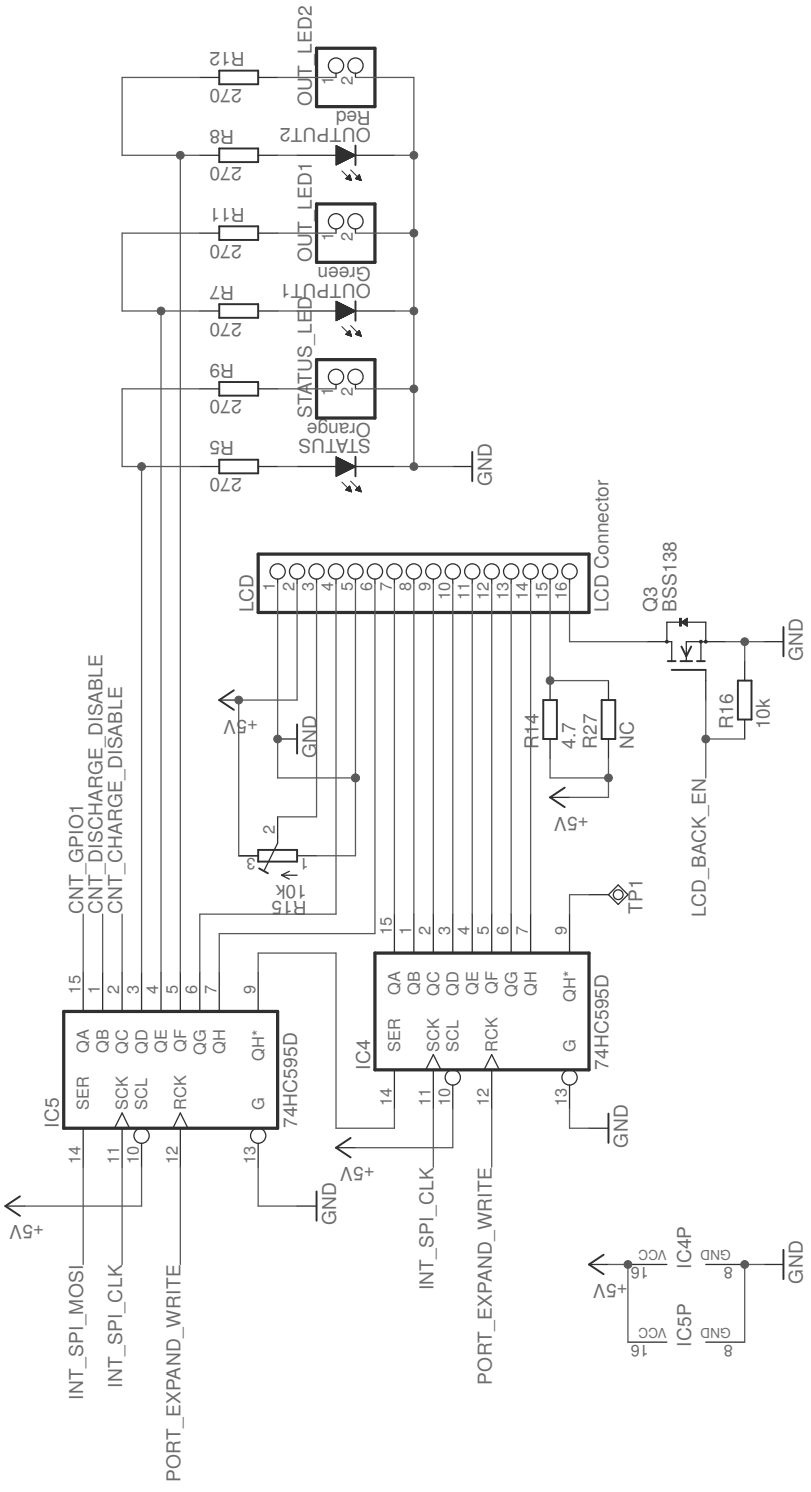


Abbildung 5.12: Beschaltung der beiden Portexpander/Schieberegister (74HCT595) und des LCDs. Die Portexpander werden seriellen über SPI Angesteuert. Das LCD wird über eine 16 Pin Stiftleiste angesteckt. Dabei ist eine Kontrasteinstellung (R15) ein Widerstand und ein Schalttransistor für die Hintergrundbeleuchtung (R14/R27/Q3) vorgesehen. Die LEDs sind doppelt ausgeführt. Jeweils eine direkt am PCB und ein Anschluss für eine externe LED.

von LEDs und der Steuerung der Verriegelung der Ausgangstreiber verwendet. In diesen Punkten stellt die geringere Geschwindigkeit kein Problem dar und die höhere Spannung ist vorteilhaft. Für das Display wird zusätzlich eine Ansteuerung der Hintergrundbeleuchtung und ein Potentiometer für die Kontrasteinstellung integriert. Bei den LEDs sind auch Anschlüsse für externe LEDs, die in ein Gehäuse eingebaut werden können, vorgesehen. Die Schaltung ist in Abbildung 5.12 auf der vorherigen Seite zu sehen.

5.3.2 Balancerboard

Für das Balancerboard werden unterschiedliche Schaltungsvarianten analysiert. Dabei muss sowohl die Messung der Zellspannungen als auch das Balancing gelöst werden. Aus Komplexitätsgründen wird dabei ein passiver Balancer eingesetzt.

Im Folgenden werden einige diskrete Schaltungsvarianten für Messung und Balancer diskutiert. Als letzter Punkt wird die kombinierte Lösung mittels eine Batterie Management ICs vorgestellt.

Die Wahl fällt schlussendlich auf den Battery-Frontend-Chip `MAX14920` von Maxim. Dabei ist vor allem die Reduktion der der Bauteile ein wesentliches Argument. Weiters besitzt er eine hohe Genauigkeit von 1mV (Analog). Das Synchrone Sampling aller Zellen ist weiters vorteilhaft für Impedanzmessungen.

Ein wesentlich Vorteil der diskreten Lösungen ist dass bei diesen bei Fehlern wie Verpolung deutlich robuster sind.

5.3.2.1 Zellspannungsmessung, Option 1: Widerstandsnetzwerk

Bei dieser Schaltung werden die einzelnen Zellen über Spannungsteiler auf ein für den ADC messbares Spannungsniveau gebracht. Durch die Verwendung des differenziellen Verstärkers des ADCs werden die Spannungsdifferenzen verstärkt und gemessen. Abbildung 5.13 auf der nächsten Seite verdeutlicht diese Schaltungsvariante. Für diese Schaltung werden $Zellenanzahl + 1$ ADC-Eingänge benötigt.

Das große Problem bei dieser Schaltungsvariante ist die hohe Anforderung an die Genauigkeit der Spannungsteilerwiderstände bei hohen Teilungsverhältnisen. Der maximale Fehler der Differenzspannung am Ausgang (U_{Err}) und der daraus resultierende Fehler bei der Zellspannungsmessung (U'_{Err}) ist in Gleichung 5.1,5.2 zu sehen.

Für ein Teilungsverhältnis (a:1) von 34:1, einer Offsetspannung der Zelle (U_{off}) von 21V, einer Zellenspannung (U_{cell}) von 4V sowie einem Widerstandsfehler (Δ)

von 0.01 (1%) ergibt sich ein maximaler Messfehler von U'_{Err} von 894mV. Dies ist weit entfernt von akzeptablen Messfehlern von etwa 40mV.

$$\begin{aligned} U_{Err} =& (U_{off} + U_{cell}) * \frac{1+\Delta}{1+\Delta+(a-1)*(1-\Delta)} \\ & - U_{off} * \frac{1-\Delta}{1-\Delta+(a-1)*(1+\Delta)} \\ & - \frac{U_{cell}}{a} \end{aligned} \tag{5.1}$$

$$U'_{Err} = U_{Err} * a \tag{5.2}$$

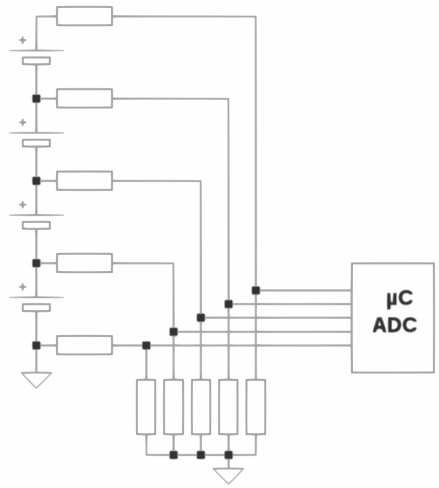


Abbildung 5.13: Schaltung Balancer Option 1 (Widerstandsnetzwerk). Dabei wird eine Reihe von Spannungsteilern verwendet um die Spannungspegel anzupassen. Die Auswahl erfolgt mit dem differenziellen Multiplexer des ADCs.

5.3.2.2 Zellspannungsmessung, Option 2: Analogmultiplexer

Eine weitere Option ist der Einsatz von Analogmultiplexern vor den Spannungsteilern. Ein mögliches Bauteil ist `DG507B` von Vishay Siliconix. Bei dieser Schaltungsvariante (Abbildung 5.14 auf der nächsten Seite) wird jeweils die obere und untere Zellenspannung über den Multiplexer auf einen von zwei Präzisionsspannungsteilern gelegt und anschließend die Differenz gemessen. Zur Verstärkung der Differenzspannung kann wahlweise der interne Verstärker des ADCs oder auch ein externer Subtrahierverstärker mit hoher Eingangsimpedanz zum Einsatz kommen.

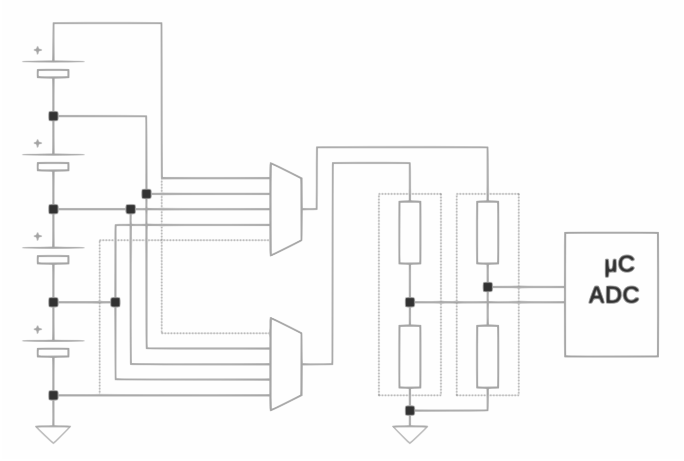


Abbildung 5.14: Schaltung Balancer Option 2 (Analogmultiplexer). Die Zellspannungen werden mit Analogmultiplexern ausgewählt und mit Präzisionsspannungsteilern auf die gewünschten Pegel gebracht. Die strichlierten Verbindungen zum Akku sind optional.

5.3.2.3 Balancer: Open Kollektor Ausgänge

Neben dem Messen der Zellspannungen ist auch noch eine Steuerung eines passiven Balancers notwendig. Dafür ist eine Lösung mit Open-Drain-Ausgängen, die eine Potentialverschiebung erlauben, möglich. Drei Schaltungsvarianten sind in Abbildung 5.15 auf der nächsten Seite dargestellt.

5.3.2.4 Kombination, Batterie Management IC (ausgewählt)

Auf diesem Sektor gibt es eine Vielzahl von unterschiedlichen Bauteilen. Diese sind im Allgemeinen für den fixen Verbau in Akkupacks von Notebooks und Elektroautos gedacht und damit nicht unbedingt optimal für den Einsatz in einem Ladegerät. Die Bauteile verwenden unterschiedliche Messverfahren und besitzen teilweise integrierte ADCs.

Die Entscheidung fällt auf den Analog Front-End Chip `MAX14920`. Dieser arbeitet nach den Flying-Capacitor Prinzip. Dabei wird ein Kondensator auf die Zellenspannung aufgeladen und anschließend dessen Potential auf Masse verschoben. Die Messung erfolgt massebezogen durch den ADC des Mikrocontrollers. Ein Vorteil des Chips ist, dass er synchrones Sampling unterstützt. So können alle Zellenspannungen gleichzeitig abgetastet und nacheinander gemessen werden. Dies ist besonders für die Messung der Spannungsantwort auf Stromimpulse interessant. Die aufgebaute Schaltung ist in Abbildung 5.16 auf Seite 52 zu sehen.

Weiters löst der IC auch die Potentialverschiebungsproblematik bei der Ansteuerung des Balancers. Die Kommunikation mit dem Chip erfolgt über SPI. Er bietet

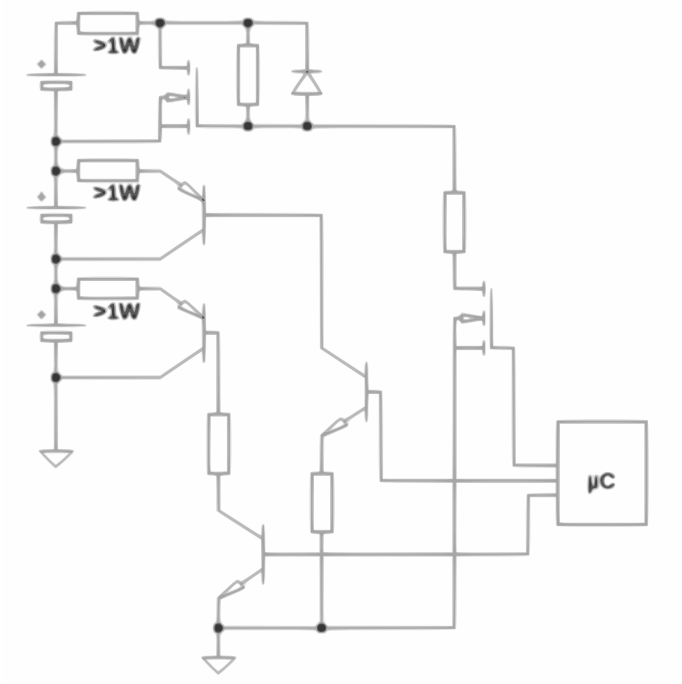


Abbildung 5.15: Drei exemplarische Schaltungsvarianten für den diskreten Aufbau eines Balancers.

für jede Zelle einen Ausgang zur Ansteuerung eines Balancing FETs. Die externen Beschaltung der Balancer-Kanäle ist in Abbildung 5.17 auf Seite 53 zu sehen.

Da der Chip eigentlich für den festen Verbau in einem Akkupack gedacht ist, bietet er einige Eigenheiten. Als minimale Zellenspannung ist 0.5V definiert. Dies Limit sollte keine größeren Probleme verursachen. Weiters ist laut Datenblatt die Versorgungsspannung mit der höchsten Zellenspannung zu verbinden. Als Mindestversorgungsspannung werden 6V benötigt. Um auch Akkus mit weniger als 6V Gesamtspannung laden zu können erfolgt die Spannungsversorgung wahlweise vom Akkupack oder der 12V Spannungsversorgung des Boards.

Laut Datenblatt sollen alle nicht verwendeten Eingänge mit der Versorgungsspannung des Chips verbunden werden. Da dies jedoch bei variabler Anzahl an Zellen nicht ohne weiters möglich ist, wird dies mittels der Widerstandsschaltung in Abbildung 5.18 auf Seite 53 gelöst.

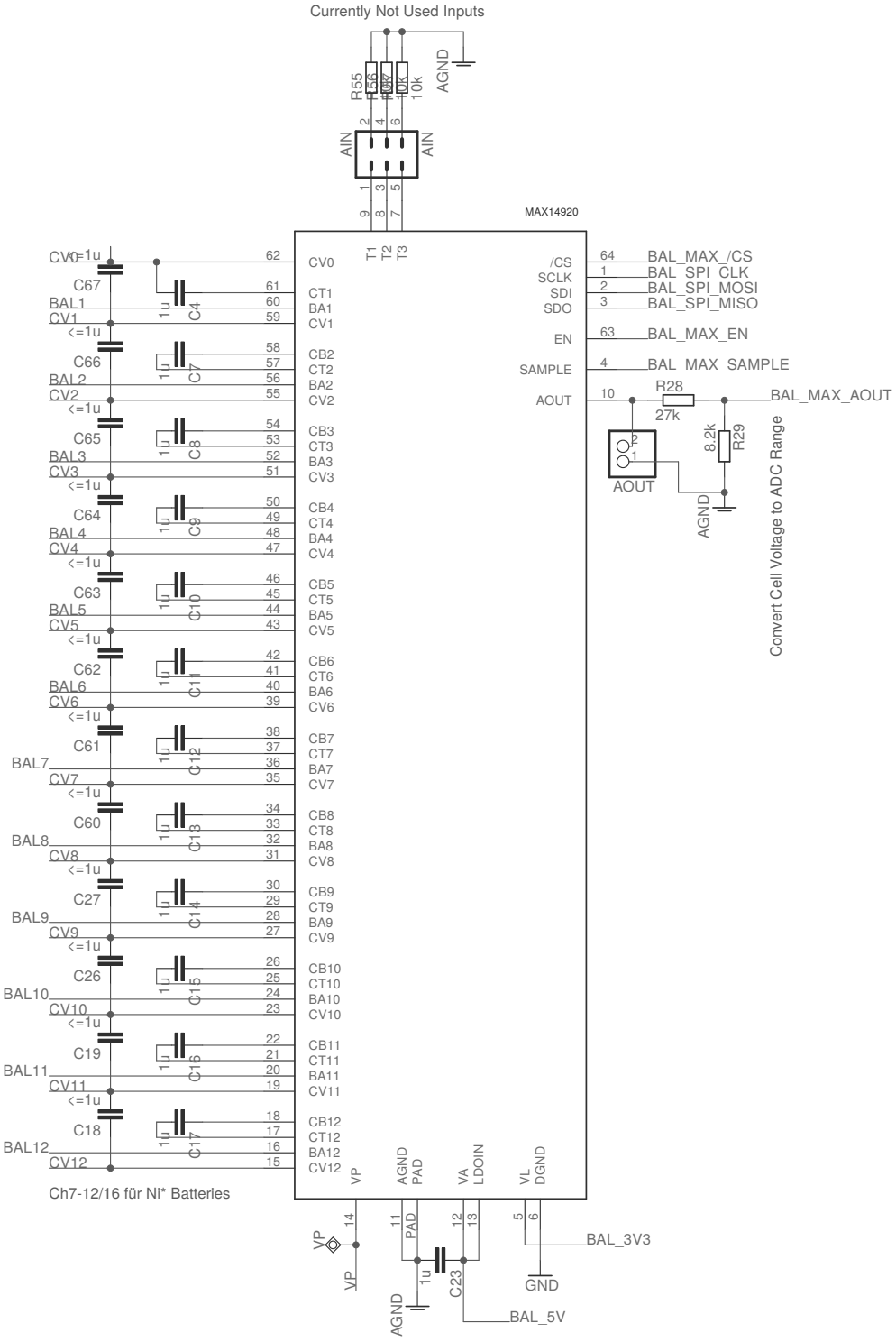


Abbildung 5.16: Beschaltung des Batteriemanagement ICs. Es sind nur die Messkomponenten dargestellt. Die Beschaltung der Balancer ist in Abbildung 5.17 auf der nächsten Seite zu sehen.

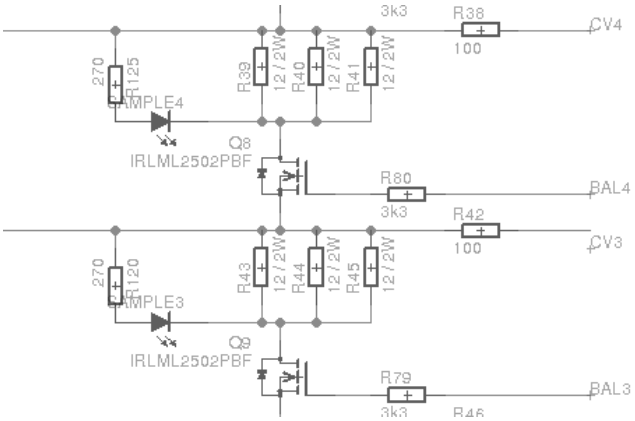


Abbildung 5.17: Schaltung zweier Balancer Ausgänge. Links der Ausgang zu den Batterien, Rechts der Anschluss am BalancerIC.

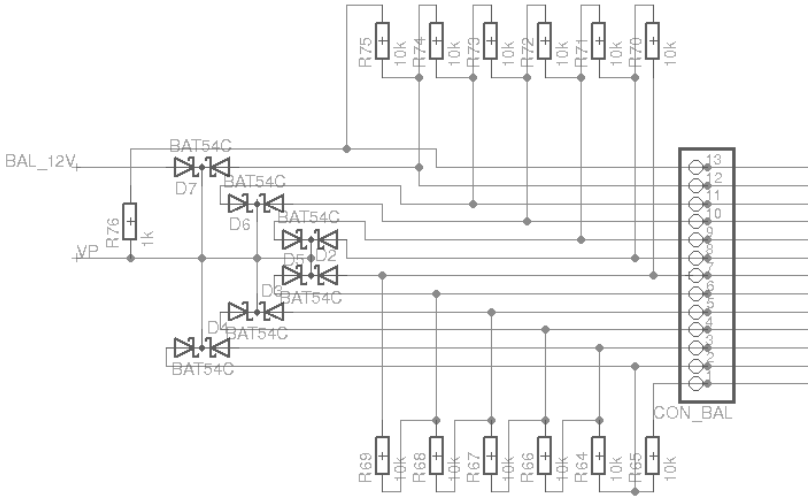


Abbildung 5.18: Vp ist die Versorgungsspannung des Chips. Diese ergibt sich durch das Diodennetzwerk als die näherungsweise höchste Spannung aller Kontakte des Balancer-Anschlusses und der 12V Versorgung. Das Netzwerk aus Widerständen teilt wiederum die Spannungsdifferenz von Vp zur höchsten Zellspannung auf. Erfolgt also ein Versorgung durch den Akkupack sind alle nicht verwendeten Eingänge auf dem Potential von Vp.

5.3.3 Treiberboard Version 1

Das Treiberboard ist das komplexeste der drei Boards. Es beinhaltet den Vorregler und Stromquelle und Stromsenke sowie einige Messfunktionen. Beim Designen der Schaltung und des Boards wurde darauf geachtet die den Schaltregler und die Stromquelle getrennt zu halten. Dies erlaubt es sie einzeln zu testen und im Fehlerfall einzeln ersetzen zu können. Die Verbindung der beiden Schaltungsteile erfolgt über Lötjumper.

5.3.3.1 Lüfter

Als Lüfter kommt ein drehzahlregelbarer 4-Pin PWM-Lüfter aus dem PC-Bereich zum Einsatz. Diese Art Lüfter besitzen einen PWM-Eingang über den ihre Drehzahl eingestellt werden kann. Die Spannungsversorgung des Lüfters erfolgt dabei mit 12V, für das PWM-Signal kommen jedoch 5V zum Einsatz. Als Tachosignal besitzt der Lüfter einen Open-Collector Ausgang. Die Ansteuerung ist in [18] spezifiziert. Die verwendete Schaltung wird in Abbildung 5.19 gezeigt.

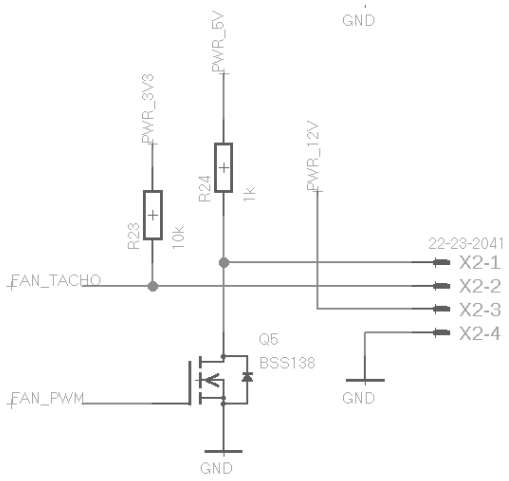


Abbildung 5.19: Schaltung für den Anschluss des PWM-Lüfters

5.3.3.2 Verpolungsschutz

Auf der Versorgungsseite ist ein Verpolungsschutz eingebaut. Abbildung 5.20 auf der nächsten Seite zeigt den Verpolungsschutz bestehend aus einem P-Kanal FET. Bei korrekter Polarität erfolgt zuerst eine Leitung über die Substrat-Diode. Bei Erreichen einer ausreichend hohen Spannung auf PWR_V+ schaltet der FET selbst ein und es entsteht eine niederohmige Verbindung. Die Widerstände limitieren die Gesamtspannung auf das zulässige Niveau von $\pm$ 20V

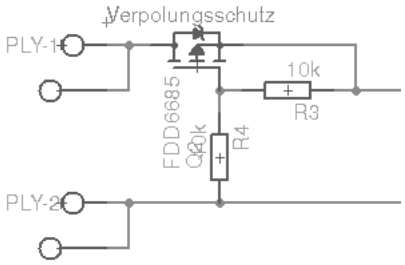


Abbildung 5.20: Der Verpolungsschutz des Versorgungseinganges.

5.3.3.3 Vorregler

Für die Vorregelung der Spannung kommt ein `LT8705` zum Einsatz. Dabei handelt es sich um einen Buck-Boost Converter. Er erlaubt es bei dem gewünschten Eingangsspannungsbereich von 10-30V die angepeilte Ausgangsspannung von 2-30V bei 10A zur Verfügung zu stellen. Der Chip benötigt vier externe Leistungstransistoren, eine Spule, Glättungskondensatoren und eine Reihe von kleineren Peripheriebauteilen, wie zum Beispiel Boost-Kondensatoren für die Erzeugung der Gate-Spannung der Highside-Switches. Die Schaltung ist in Abbildung 5.21 auf der nächsten Seite zu sehen.

Da der Regler im Betrieb als Boost-Konverter Ströme von etwa 45A (Spitze) schaltet, ist die Auswahl der Komponenten eine Herausforderung. Es ist schwierig Spulen diese hohen Ströme und Induktivitäten als Standardbauteil zu bekommen. Schließlich wird eine `VER2923-103KL` (10μF 26A-RMS) von Coilcraft eingesetzt. Dies bedeutet allerdings, dass bei Betrieb mit geringen Eingangsspannungen nicht die maximale Ausgangsleistung erreicht wird. Auch die Auswahl der Kondensatoren ist problematisch, da im Datenblatt meist nur Werte für den Rippelstrom bei 105°C angegeben werden. Verwendet man diese Werte ergibt sich die Anforderung von bis zu 15 Elektrolytkondensatoren für die Glättung. Dieser Faktor wird anschließend weitgehend ignoriert und die Auswahl erfolgt auf Basis des ESR. Zur zusätzlichen Dämpfung werden Keramikkondensatoren eingesetzt. Zu Überprüfung wird die Schaltung in LTSpice [21] simuliert.

Um die Ausgangsspannung des Vorreglers einstellen zu können wird die in Abbildung 5.22 auf Seite 57 gezeigte Schaltung verwendet. Da die beiden DACs des Mikrocontrollers bereits für die Steuerung des Lade- und Entladestroms verwendet werden, wird dafür ein eigener DAC benützt. Hierfür kommt ein `MCP4922` [23] zum Einsatz. Dieser wird über SPI am Mikrocontroller angeschlossen. Der DAC läuft dabei mit 3.3V, der höhere Signalpegel ist ebenfalls von Vorteil.

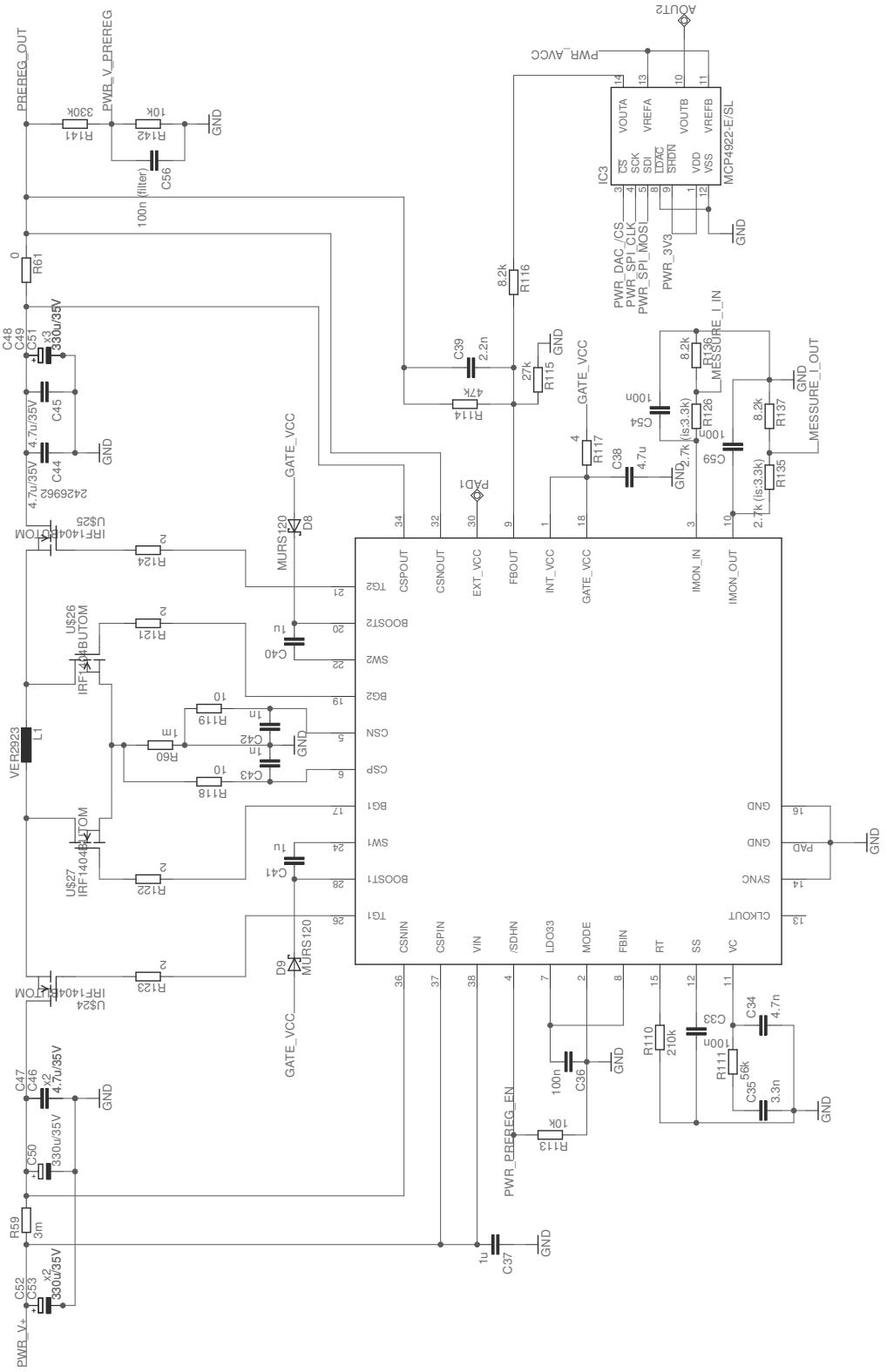


Abbildung 5.21: Schaltung des Vorreglers mit LT8705 (Buck Boost Controller) und MCP4922 (DAC)

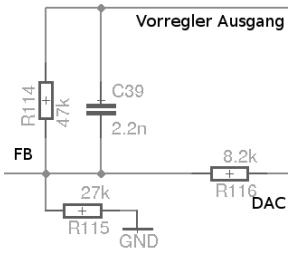


Abbildung 5.22: Schaltung zur Spannungsregelung des Vorregler mittels DAC.

5.3.3.4 Stromquelle für die Ladung

Die Stromquelle erlaubt es den Ladestrom genau und schnell einzustellen. Um eine Verschiebung des Massepotentials des Akkus so weit wie möglich zu vermeiden wird die Strommessung auf der positiven Versorgung implementiert. Dafür kommt der Current-Sense-Amplifier `LT6106` zum Einsatz. Dieser spiegelt den abgeschwächten Strom gegen Masse, wo er durch einen Widerstand als Eingangssignal für den OPV dient. Als Leistungstransistor kommt ein P-Kanal FET zum Einsatz.

Um sicherzustellen, dass es zu keinem ungewünschten Stromfluss kommt, ist eine Verriegelung der Schaltung implementiert. Dabei werden über die beiden Transistoren $Q12$ und $Q14$ Gate und Source des Leistungstransistors kurzgeschlossen. Die Schaltung ist in Abbildung 5.23 zu sehen.

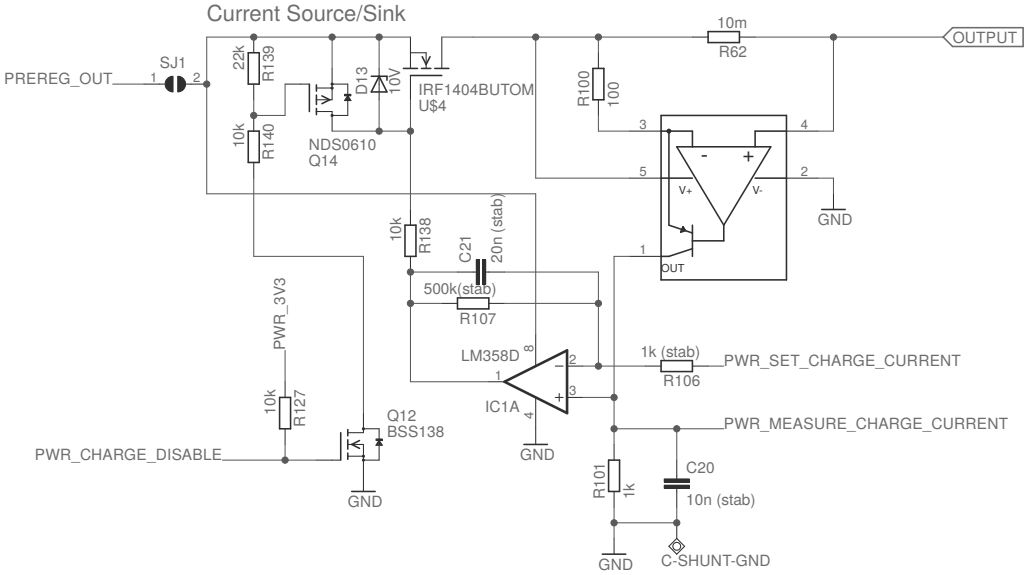


Abbildung 5.23: Schaltung Stromquelle (Laden). Treiberboard Version 1

5.3.3.5 Stromsenke für die Entladung

Zum Entladen kommt eine Standard OPV-Stromsenke zum Einsatz. Dabei ist zu beachten, dass die gesamte Entladeleistung am Leistungstransistor umgesetzt wird. Die verwendeten Transistoren im TO220-Gehäuse erlauben dabei teilweise eine Verlustleistung von bis zu 125W. Aufgrund der thermischen Übergangswiderstände ist es jedoch sinnvoll zwei Transistoren parallel zu schalten um das Ziel von 100W zu erreichen. Um ein sicheres abschalten der Schaltung zu gewährleisten wurde eine Verriegelung vorgesehen. Die Entladeschaltung ist in Abbildung 5.24 zu sehen.

Für eine genauere Messung des Entladestroms wäre es vorteilhaft einen höheren Spannungsabfall am Shunt zu erreichen. Der optimale Spannungsabfall von 1V bei 10A führt jedoch zu 10W Verlustleistung am Shunt, was zu Problemen bei der Kühlung führt.

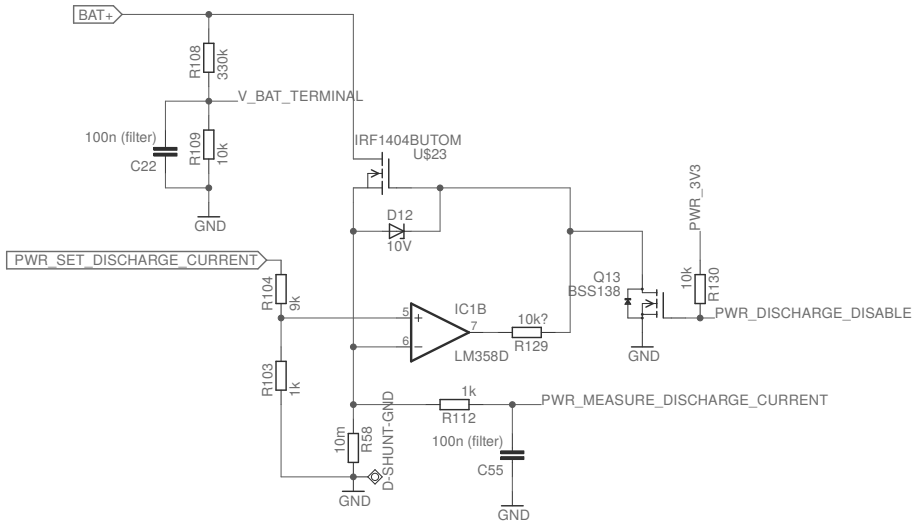


Abbildung 5.24: Stromsenke (Entladeschaltung). Treiberboard Version 1

$$R_\Theta = R_{\Theta JC} + R_{\Theta CS} + R_{\Theta SA} \tag{5.3}$$
$$R_\Theta = 0.45°\mathrm{C\,W^{-1}} + 0.5°\mathrm{C\,W^{-1}} + 0.3°\mathrm{C\,W^{-1}}$$

$$T_J = T_A + P_D * R_\Theta \tag{5.4}$$
$$T_J = 25°\mathrm{C} + 100\mathrm{W} * 1.25°\mathrm{C\,W^{-1}}$$
$$T_J = 150°\mathrm{C}$$

5.3.4 Treiberboard Version 2

Beim Treiberbord in Version 1 kommt es zu einer Reihe von Problemen.

- **Keine Funktion des Schaltreglers**: Im ersten Aufbau verursacht der Schaltregler ein Problem durch das gleichzeitige Ansteuern von highside- und lowside-FET. Dies führt zum Kurzschluss der Versorgungsspannung. Daraufhin wird der Chip getauscht und das Board ein zweites Mal komplett neu aufgebaut. Leider kommt es auch hier nie zu Schaltvorgängen. Da sich die implementierte Schaltung leicht von den anfänglichen Simulationen unterschied, wird auch die Simulation noch einmal mit den aufgebauten Bauteilwerten durchgeführt. Laut Simulation sollte die Schaltung funktionieren, was aber nicht mit der aufgebauten Schaltung übereinstimmt.
- **Current-Sense-Amplifier:** Der Current-Sense-Amplifier benötigt mindestens 2.7V um korrekt zu arbeiten. Dies führt zu einem Versagen der Stromregelung sowohl im Kurzschlussfall als auch wenn nur eine einzelne NiMh-Zelle angeschlossen ist.
- **Verhindern von Rückströmen:** Es kommt zum Zurückfließen von Strömen vom Akku zum Vorregler. Das ist beim Buck-Boost Regler noch vertretbar, führt aber spätestens bei der Verwendung des reinen Buck-Reglers zu einer Versorgung des Gerätes aus dem zu ladenden Akku.
- **Minimale DAC Ausgangsspannung:** Der DAC des Mikrocontrollers besitzt eine minimale Ausgangsspannung von 30mV (gemessen) beziehungsweise 150mV laut Datenblatt. Dies führt leider zu einem minimalen Ausgangsstrom von 300mA beziehungsweise 1.5A.

5.3.4.1 Stromquelle

Bei der Stromquelle wird der Current-Sense-Amplifier vor den Regeltransistor verschoben. Dadurch können jetzt beliebig geringe Ausgangsspannungen erreicht werden. Einziges Limit ist, dass die Vorregler-Ausgangsspannung nicht unter 2.7V fallen darf.

Es werden auch die Leistungstransistoren umgestellt. Die Schaltung besitzt nun 2 SMT Leistungstransistoren am Board. Dabei erfolgt die Kühlung durch die Printplatte. Da auf diese Weise nur eine geringe Verlustleistung möglich ist, kann auch ein externer Transistor verwendet werden. Die Auswahl zwischen internem und externem Transistor erfolgt über Lötjumper.

Zusätzlich wird eine Schottky-Diode eingebaut. Diese verhindert einen Stromfluss in das Gerät im ausgeschalteten Zustand. Dies ist notwendig, da der Buck-Converter einen zurückfließenden Strom nicht aufhalten kann.

5.3.4.2 Neuer Schaltregler

Nachdem der eigentlich geplante Buck-Boost-Schaltregler nicht funktioniert, wird ein alternativer Regler verbaut. Dabei wird ein einfacher Buck-Konverter (LT3840) verwendet. Der Vorteil ist, dass dabei nur zwei Schalttransistoren nötig sind. Die Schaltung (Abbildung 5.25 auf der nächsten Seite) entspricht auch hier wieder weitestgehend den Vorschlägen im Datenblatt. Es wird versucht diese Schalttransistoren auf SMT umzustellen. Um mögliche Probleme mit den SMT Transistoren, wie eine zu hohe Wärmeentwicklung umgehen zu können, sind zusätzliche Anschlüsse für externe Transistoren im TO220 Gehäuse vorgesehen.

5.3.4.3 Spannungsinversion

Da der DAC nicht 0V erreicht, werden Spannungsteiler mit einer negativen Referenzspannung verwendet. Um einen eigenen Spannungsinverterchip zu sparen ist ein kapazitiver Spannungsinverter diskret aufgebaut. Alle dafür benötigten Bauteile sind bereits in anderen Schaltungsteilen im Einsatz. Dabei erzeugt der Mikrocontroller ein Rechtecksignal mit 3.3V. Dieses wird mittels eines Kondensators im Potential verschoben und anschließend gleichgerichtet. Ein RC-Glied dient als Dämpfung, eine LED zur Erzeugung einer Referenzspannung und als Indikator.

5.3.5 Electrostatic Discharge (ESD) Schutz

Die ESD Schutzbeschaltung der Temperaturfühler und des Businterfaces einspricht der im Steuermodul (Abschnitt 5.2.5 auf Seite 43).

Der Balancer-Anschluss verwendet nur die interne Schutzbeschaltung des Balancer-ICs. Einer weiterer Schutz wäre nur mit verhältnismäßig hohem Aufwand möglich.

Die Anschlüsse für die Spannungsversorgung verfügen über starke Kondensatoren, die Pulse aufnehmen können sollten. Der Leistungsausgang besitzt starke Leistungstransistoren, deren Substratdioden sollten eine Ableitung gegen Masse beziehungsweise den Ausgangs-Puffer-Kondensator erlauben.

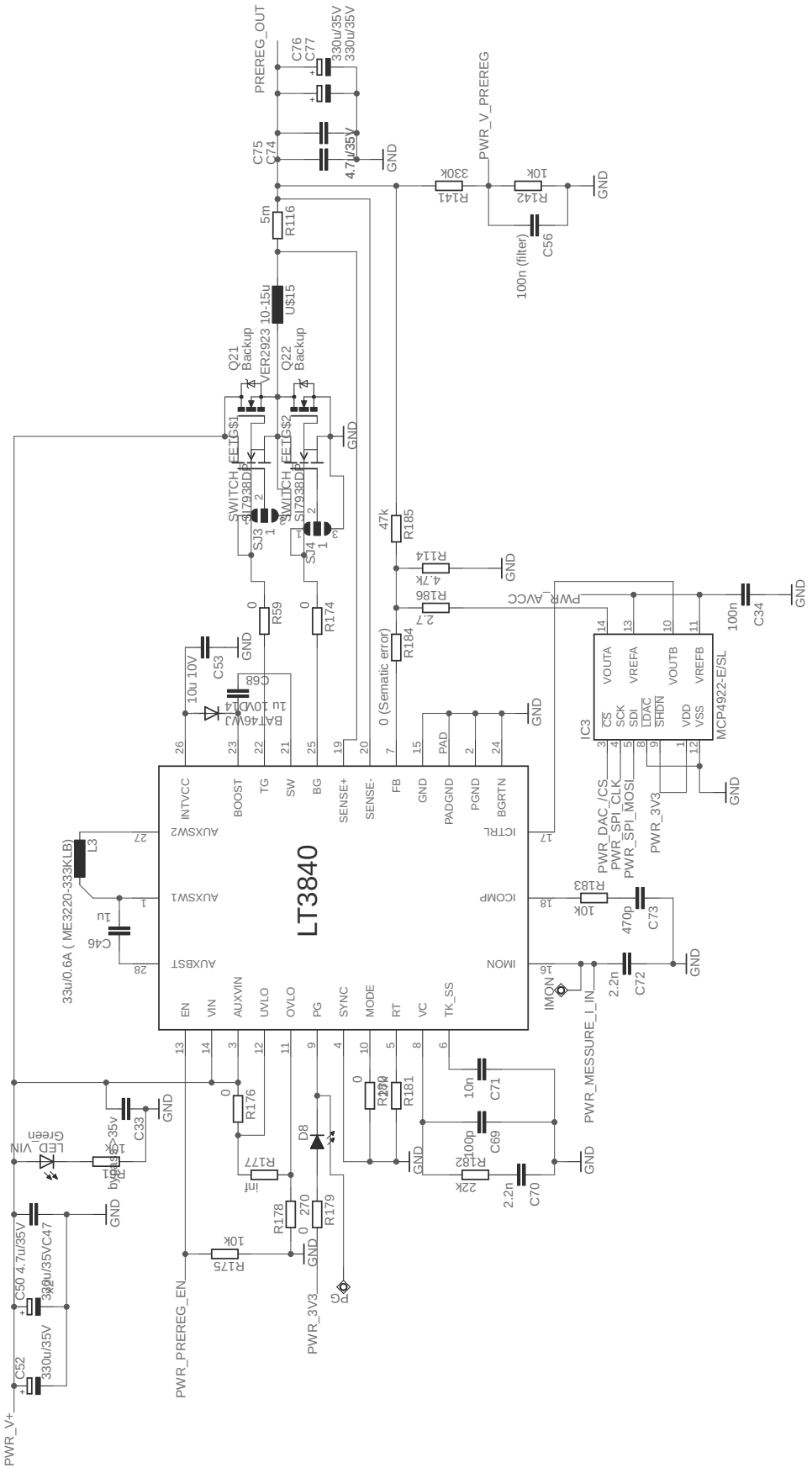


Abbildung 5.25: Schaltung Vorregler, Treiberboard Version 2

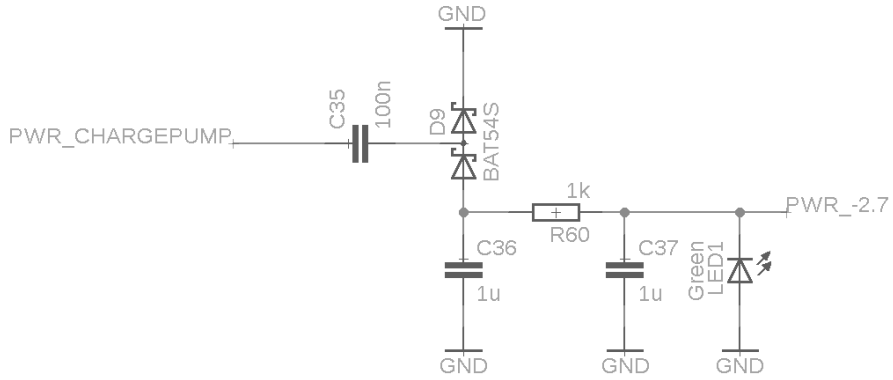


Abbildung 5.26: Schaltung diskret aufgebauter Spannungsinverter

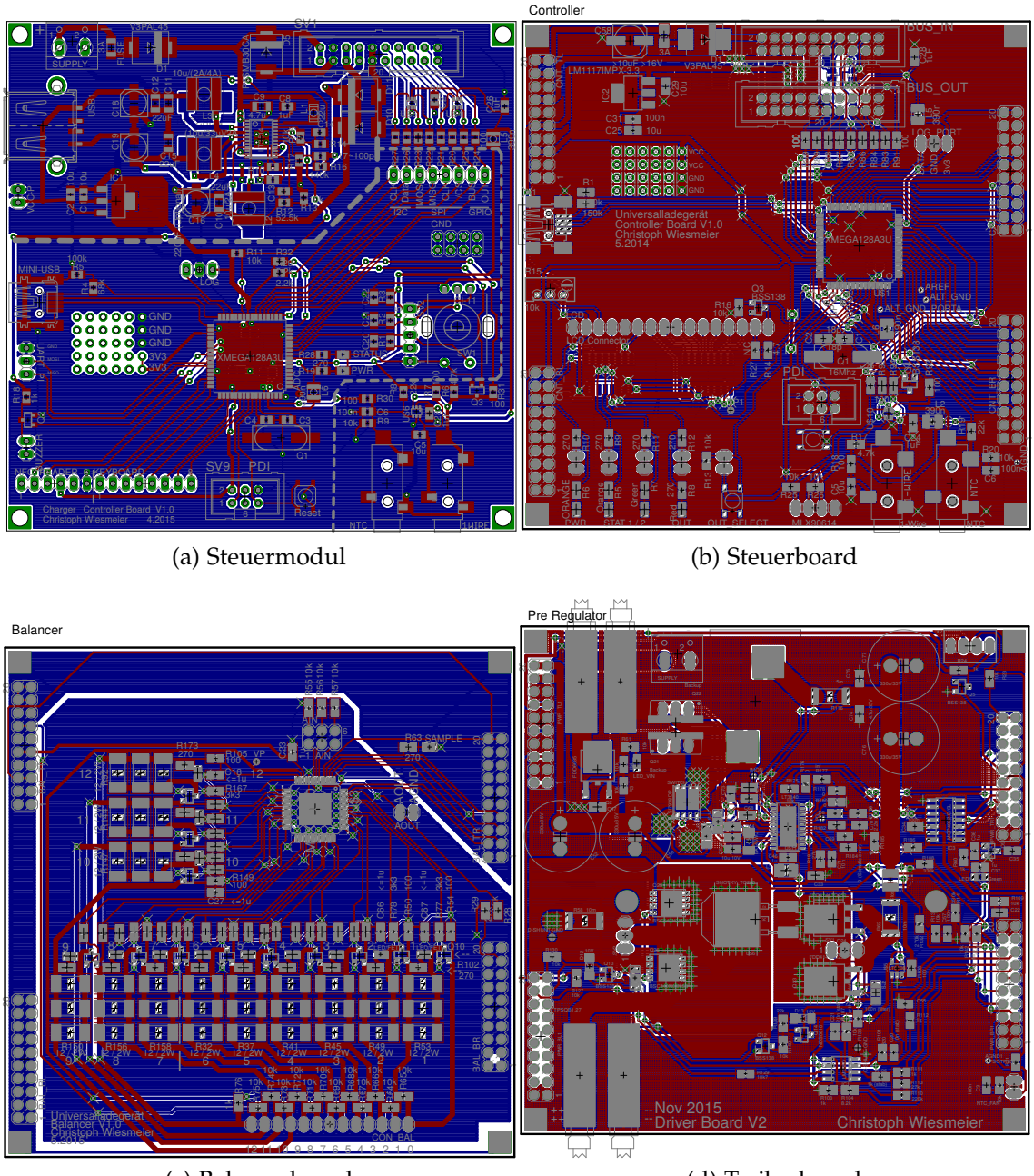
5.4 Layout (PCB)

Beim Layout wurde eine einheitlichen Platinen mit 10x10cm und 2 Layern verwendet. Die Bestückung erfolgt großteils mit SMT Bauteilen auf der Oberen Seite. Alle Anschlüssen externen Anschlüssen wurden auf der Vorder- oder Hinterseite platziert. Die Unterseiten der Boards wurde jeweils so gut wie möglich als Massenflächen verwendet. Das Balancerboard besitzt dabei eine Trennung von analoger und digitaler Masse. Das Treiberboard besitzt eine Reihen von Leistungstransistoren. Um diese sind Vias angebracht um die werme auf einen Kühlkürper auf der Unterseite Übertragen zu können. Die Layouts aller vier finalen Printplatten sind in Abbildung 5.27 auf Seite 64 dargestellt.

5.5 Ergebnisse, Verbesserungpotential, Bugs

- **Genauigkeit Entladestrommessung:** Die Messung des Entladestroms besitzt nur 100mV Full-Range. Daraus ergibt sich Messfehler von etwa 50mA. Eine mögliche Verbesserung könnte hierbei der Einsatz eines Vorverstärker-OPVs bieten. Eine weitere Möglichkeit wäre der Einsatz eines hochohmigen Shunts um damit einem höheren Signalpegel von beispielsweise 1V zu erreichen. Um die dabei entstehende Verlustleistung von 10W abzuführen, müssen mehrere Widerstände zum Einsatz kommen .
- **Konstantstromregelung mit Schaltregler:** Der verwendete Vorregler besitzt grundsätzlich die Möglichkeit einer Konstantstromregelung. Dies würde erlauben bei normalen Ladevorgängen die lineare Stromregelung zu umgehen und somit die Verlustleistung zu reduzieren. Aus nicht näher bekannten Gründen ist es jedoch weder möglich den vom Schaltregler ausgegebenen Strom am dafür vorgesehenen Ausgangspin zu messen, noch das Stromlimit einzustellen.

- **Platinen Aufteilung:** Leider macht es der Schaltregler auf der Platine fast unmöglich mit dem Oszilloskop Messungen auf der Platine durchzuführen. Die durch den Masseclip entstehende Schleife führt oft zu Störeinkopplungen, die weit größer als die zu messenden Signalpegel sind.



(a) Steuermodul (b) Steuerboard

(c) Balancerboard (d) Treiberboard

Abbildung 5.27: BCB Layouts. Top-Layer: Rot, Bottom-Layer: Blau.

6 Software

Ein zentraler Punkt dieser Arbeit ist die Software. Es wird zwischen zwei wesentlichen Komponenten unterschieden. Erstens: Die Mikrocontroller Software, diese läuft direkt auf den Boards und übernimmt die Ansteuerung der Hardwaremodule. Zweitens: Die Steuersoftware, die auf dem RaspberryPi oder dem Entwickler-Computer läuft. Diese übernimmt die höheren Steueraufgaben, wie das User Interface und die Ladeverfahren.

Aufgrund des modularen Designs spielt das Kommunikationsprotokoll eine zentrale Rolle. Es wird sowohl für die Kommunikation zwischen der Steuersoftware und Steuermodul als auch für die Kommunikation zwischen Steuer- und Treibermodulen verwendet.

6.1 Mikrocontroller

Mit dem Erstellen der Mikrocontroller-Firmware wird auch ein kleines Framework aufgebaut. Ziel ist die Wiederverwendung einiger Komponenten, wie Logging oder des Kommunikationsprotokolls. Diese Komponenten werden sowohl am Treiber- als auch am Steuermodul verwendet und sollen in Zukunft auch bei anderen Projekten zum Einsatz kommen.

6.1.1 Buildsystem

Das Buildsystem ist in CMake [9] erstellt. Dieses moderne Tool erlaubt es komplexe Buildprozesse zu definieren. Dabei werden die Entwicklerwerkzeuge wie Compiler, Linker und andere Tools automatisch erkannt und nach Benutzervorgaben Makefiles erstellt.

Das erstellte Buildsystem bietet folgende Funktionen:

- Erzeugen der Firmware für unterschiedliche Controller (Apps) aus der selben Quellcodesammlung.
- Einfaches Erstellen einer zusätzlichen Firmwareversion durch Anlegen einen neuen Ordners.
- Erstellen von Bibliotheken und das Verwenden dieser in den unterschiedlichen Applikationen. Dabei werden Abhängigkeiten korrekt aufgelöst.

- Erstellen von Sourcecode-Bibliotheken, die erst im Kontext der Applikation gebaut werden. Dies erlaubt das Erstellen von hoch optimierten Bibliotheken und ermöglicht die Integration des Atmel Software Famework als Bibliothek.
- Direktes Übertragen der Firmware auf den Mikrocontroller
- Integration zusätzlicher Tools wie das Logging Modul.
- Integration von Tests nach einem ähnlichem Schema wie Applikationen.

6.1.2 C++

Bei der Entwicklung wird auf C++ gesetzt. Dies ist momentan im Embedded Bereich leider noch nicht üblich. C++ bietet jedoch im Vergleicht zu C einige Zusatzfunktionen, die für den Embedded Bereich sehr nützlich sind.

Im Embedded Bereich wird auf die Verwendung von dynamischem Speicher (Heap) meist verzichtet. Grund dafür ist die sehr limitierte Speichergröße der Systeme und das Risiko von Fehlern wie durch Speicherfragmentierung. Leider basiert ein wesentlicher Teil der C++ Standard-Template-Library (STL) auf dynamischem Speicher und ist somit nicht für die Embedded-Entwicklung verwendbar. Die C++ Standard-Library von gcc ist ohnehin noch nicht auf AVR-Mikrocontroller portiert. Somit fallen auch Library-Funktionen, die für den Embedded Bereich nützlich sind, weg.

Einen kleinen Vorteil bieten Klassen und Namensräume. Dieses bieten eine bessere Strukturierung des Codes. Dabei ist zu sagen, dass Klassen in ihrem üblichen Anwendungsbereich eher selten sind, da der Großteil der Applikation mit Hardware interagieren muss und eine mehrfache Instanziierung einer Klasse somit nicht sinnvoll ist.

Einer der größten Vorteile sind C++ Templates. Diese erlauben es Codesegmente zu erstellen, die nachträglich für spezielle Datentypen oder Werte erstellt werden. Templates erlauben das Erstellen von Datenstrukturen. Im konkreten Fall wird ein Ringpuffer implementiert und an diversen Stellen eingesetzt.

Weitere interessante C++ Funktionen sind Konstante-Ausrücke (constexpr) und Template Metaprogramming. Beide Technologien ermöglichen es Berechnungen zur Kompilierzeit auszuführen und somit Rechenzeit am Controller zu sparen. Dabei ist anzumerken, dass constexpr eine neue C++11/14 Funktion ist. Template Metaprogramming ist einerseits zwar sehr leistungsfähig, andererseits aber eines der komplexesten Themen in C++.

```
1  LOG_MSG( "Starting up" , 1 );
2  LOG_WARNING( "A Warning", 2 );
3  LOG_ERROR( "An Error", 3 )
4  int i = 0;
5  LOG_VALUE(i,4);
6  LOG_ASSERT( i== 0, "Initialisation failed", 5 );
```

Abbildung 6.1: Beispiele für Logging Kommandos am Mikrocontroller.

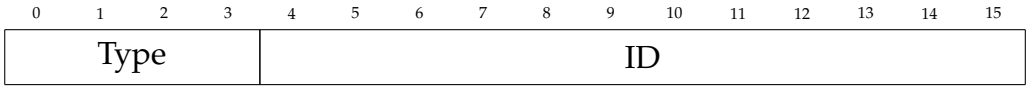


Abbildung 6.2: Format eines Message-Packets.

6.2 Logging

Das Debuggen von Mikrocontroller-Programmen ist schwierig. Die Softwareunterstützung für das Debugging von AVR-Mikrocontroller unter Linux ist sehr beschränkt. Dafür wird spezielle Hardware wie das JTAG-ICE benötigt. Weiters ist das Stoppen der Ausführung des Programms in vielen Fällen problematisch, da dadurch mögliche Echtzeitanforderungen verletzt werden.

Um eine Alternative zum traditionellen Debugging mit Breakpoints zu schaffen, wird ein Logging implementiert. Das Logging folgt dem selben Prinzip wie Ausgaben auf den Standardoutput bei einem PC-Programm (Codebeispiel Abbildung 6.1). Die Daten werden dabei über eine serielle Schnittstelle auf den PC übertragen um sie dort zu betrachten.

Um Speicher am Mikrocontroller und Bandbreite bei der Übertragung zu sparen, sendet das Logging-Protokoll nicht den Text, der in den Logging-Instruktionen angegeben ist, sondern nur die ID (letzter Parameter).

Während des Build-Prozesses erzeugt ein Pythonscript zusätzlich zum ausführbaren Programm eine csv-Datei und überprüft die Eindeutigkeit der IDs. Die csv-Datei wird vom Logging-Client (Abbildung 6.4 auf der nächsten Seite) am PC eingelesen und verwendet um die Nachrichten zu decodieren.

Für alle Pakete außer LOG_VALUE werden 16 Bit übertragen. 4 Bit für den Nachrichtentyp und 12 Bit für die ID (Abbildung 6.2).

LOG_VALUE Pakete besitzen einen 32 Bit Header (Abbildung 6.3 auf der nächsten Seite), der zusätzlich den übertragenen Datentyp und die Länge enthält .

Auf der Seite des Mikrocontrollers werden zwei Übertragungsarten unterstützt.

Erstens eine gepufferte Übertragung. Diese schreibt die Daten in einen Sendepuffer und sendet diese dann unter Verwendung von Interrupts auf die serielle Schnittstelle.

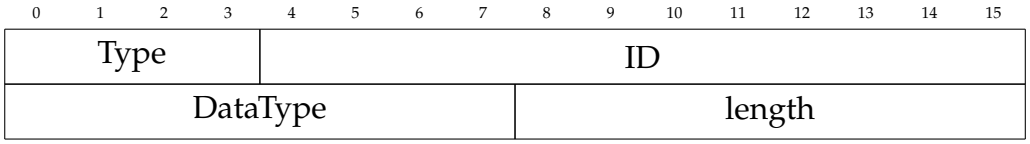


Abbildung 6.3: Header eines Value-Packets. Es folgen *length* Bytes and Daten.

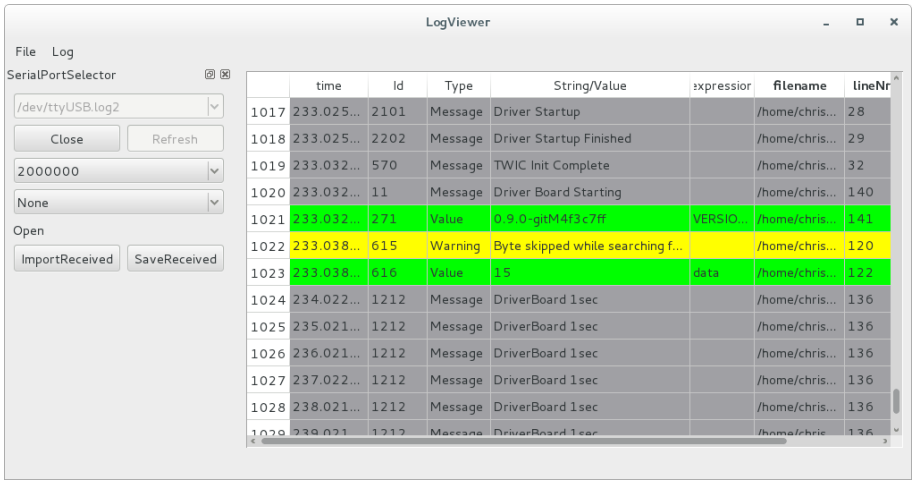


Abbildung 6.4: GUI des LogViewers, welcher die Nachrichten des Mikrocontrollers dekodiert und Anzeigt.

Alternativ gibt es eine Blocking-Implementierung. Diese ist in Verbindung mit hohen Baudraten wie 2 MBaud sinnvoll. Obwohl die Implementierung den ausführenden Thread während des Sendens blockiert, sollte sie in den meisten Fällen kaum Zeitverzögerung verursachen. Der Hintergrund ist, dass der USART der XMAGAs zusätzlich zum Sendeschieberegisters einen 1 Byte Transmit-Puffer besitzt. Das heißt, dass bei leeren Hardware-Puffern die typischen 2 Byte Nachrichten in diese passen. Dies geht vermutlich sogar schneller als das Schreiben der Daten in den Ringpuffer der Buffered-Variante. Die hohen Baudraten sind ratsam damit der Puffer auch tatsächlich den größten Teil der Zeit frei ist und die Übertragung von Value-Paketen, die in dieser Situation auf jeden Fall blockieren, nicht zu lange dauert.

6.3 System Timer

Auf dem Mikrocontroller sollten meist mehrere Aufgaben in Unterschiedlichen Zeitintervallen ausgeführt werden. Es wäre nun möglich für jede davon einen Hardware Timer zu starten und gegebenenfalls eine gegenseitige Unterbrechung zu erlauben. Diese Implementierung ist jedoch komplex in der Konfiguration und setzt eine menge Timermodule voraus. Die verwendete Implementierung

kommt jedoch mit nur einem Hardware Timer aus und bietet für die meisten Anwendungen ausreichend Funktionalität.

Nicht zeitkritische Tasks werden in einer Schleife im Hauptprogramm ausgeführt. Hierbei gibt es kein Limit für die Ausfuhrzeit oder die maximale Anzahl der aufrufe eines Tasks. Eine hier eingetragen Routine kann also Hundertausend mal pro Sekunde oder auch nur 10 mal Aufgerufen werden. In dieser Zeitklasse läuft momentan die Abarbeitung vor Host-Befehlen des Bus-Protokolls.

Für zyklische Aufgaben bei denen eine konstante Ausführrate erwünscht ist gibt es einen einzelnen System-Timer. Dieser verwendet einen Hardware-Timer und wird alle 500μs aufgerufen. Da ein wesentlicher Teil der Zeit dieser Timer-Interrupt ausgeführt wird, läuft er auf der niedrigsten Interrupt-Priorität. Aus diesem 500μs Interrupt werden nun unterschiedliche Intervall-Klassen abgeleitet. Durch Bitmaskierung ist es möglich Zweier-Potenzen der Ausführungsrate zu erzeugen. Wobei bei jedem Timer-Interrupt nur eine einzige Intervall-Klasse ausgeführt wird. Jede Intervall-Klasse hat somit 500μs Zeit zur Ausführung zur Verfügung. Die Einhaltung dieses Intervalls wird am Ende des Interrupts überprüft und Überschreitungen protokolliert. Der Sourcecode der Interrupt Routine wird in Abbildung 6.6 auf der nächsten Seite gezeigt.

```
1  ISR( SYSTEM_TIMER_ISR ){
2      static uint16_t counter = 0;
3      counter++;
4
5      if( counter & ( 1 << 0 ) ){// 1msec
6          SYSTEM_TIMER_EVENT_1MS
7      }
8      else if( counter & ( 1 << 1 ) ){ // 2msec
9          SYSTEM_TIMER_EVENT_2MS
10     }
11     ...
12     else if( counter & ( 1 << 15 ) ){ // 32768msec
13         SYSTEM_TIMER_EVENT_32768MS
14     }
15
16     if( SYSTEM_TIMER.INTFLAGS & 0x01u ){
17         LOG_ERROR("Real Time Error, Timer Interrupt not finished
               before new one occurred", 1005 );
18     }
19 }
```

Abbildung 6.5: Ausschnitt aus der System Timer Interrupt Routine. Zeile 6 bis 15 Zeigt die Zeitklassen. Zeile 17 überprüft die Einhaltung der Ausführungszeit.

Der große Vorteil dieser Architektur ist eine relativ gute Lastverteilung und das sich die Routinen gegenseitig nicht unterbrechen. Das führt Einerseits zu geringerem Stack bedarf und Andererseits zu weniger Locking. Weiters ist die

Implementierung äußerst einfach. Ein Nachteil ist jedoch dass für alle Intervall-Klassen das selbe Zeitlimit von 500μs gilt.

6.4 Weitere Module

Für jedes weitere verwendete Hardwaremodul oder Externen IC wird eine Klasse oder namespace angelegt und die Funktion darin abstrahiert. Sofern sinnvoll möglich wurde die Klasse als template implementiert um sie gegebenenfalls Mehrfach zu nutzen. Ein Beispiel hierfür ist der USART des XMEGA Controllers. Dieser kann für jeden Hardware USART neu instantiiert werden. Die Nutzung von Templates erlaubt hierbei eine Implementierung ohne laufzeit-overhead. Da der Compiler den Code für die spezifischen Registeradressen im Mikrocontroller instanziiert.

In anderen Fällen wurde zum Beispiel ein SPI-Modul instantiiert und dieses um die benötigte Chi p-Funktionalität erweitert.

```
1  ISR( SYSTEM_TIMER_ISR ){
2      static uint16_t counter = 0;
3      counter++;
4
5      if( counter & ( 1 << 0 ) ){// 1msec
6          SYSTEM_TIMER_EVENT_1MS
7      }
8      else if( counter & ( 1 << 1 ) ){ // 2msec
9          SYSTEM_TIMER_EVENT_2MS
10     }
11     ...
12     else if( counter & ( 1 << 15 ) ){ // 32768msec
13         SYSTEM_TIMER_EVENT_32768MS
14     }
15
16     if( SYSTEM_TIMER.INTFLAGS & 0x01u ){
17         LOG_ERROR("Real Time Error, Timer Interrupt not finished
               before new one occurred", 1005 );
18     }
19 }
```

Abbildung 6.6: Ausschnitt aus der System Timer Interrupt Routine. Zeile 6 bis 15 Zeigt die Zeitklassen. Zeile 17 überprüft die Einhaltung der Ausführungszeit.

- **Clock:** Konfiguriert die Takterzeugung des Mikrocontrollers durch Quarz, PLL und Prescaler.
- **SystemTimer:** Generiert Timerfunktionen mit statischen Ausführraten.
- **DAC:** Steuert den digital zu analog Konverter des Mikrocontrollers an.

- **ADC:** Misst Werte für Spannungen, Ströme und Temperaturen auf den Zahlreichen Anarlogen Eingängen.
- **Watchdog:** Konfiguriert den Watchdog-Timer
- **LED:** Vereinfacht die Ansteuerung von angeschlossenen LEDs
- **Buzzer:** Verwendet einen Timer um am Piezo-Lautsprecher Töne unterschiedlicher Frequenz und Länge aus zu geben.
- **Keyboard:** Übernimmt die zeilenweise Abfrage des $4x4$ Matrix-Keyboards.
- **SPIInterconnectMaster:** Übernimmt die Steuerung des Master-/Slave-Buses für die Kommunikation mit den Treibermodulen.
- **SPIInterconnectSlave:** Slave Komponente des implementierten Buses.
- **RotaryEncoder:** Dekodiert die Signale des Inkrementalgebers.
- **Fan:** Übernimmt die Steuerung und Drehzahlmessung des 4-Pin PWM-Lüfters.
- **Portexpander:** Erlaubt gepufferte Schreiboperationen auf eine SPI Port Expander. Des weiteren werden maskierte Schreiboperationen unterstützt, sodass sich unterschiedliche Komponenten die den Portexpander verwenden nicht beeinflussen.
- **LCD:** Übernimmt die Steuerung eines LCD-Displays und greift im Hintergrund auf den Portexpander zurück.
- **MAX14920:** Übernimmt die Steuerung des Batteriemanatgement-Chips wie Kanalauswahl, Zellspannungsmessung und Balancer.
- **MLX90614:** Zugriff auf einen Infrarot-Thermometer.
- **DS18B20:** Auslesen eines digitalen 1-Wire Temperaturfühlers.

6.5 Steuersoftware

Die Steuersoftware selbst läuft auf einer PC basierten Plattform. Dabei wird sowohl am Entwicklungsrechner als auch am Embedded-Computer (RaspberryPi) Debian/Linux [11] in Version 8 eingesetzt. Aufgrund der unterschiedlichen CPU-Architekturen x86 versus ARM muss die Software jedoch für das RaspberryPi eigens kompiliert werden.

Die Steuersoftware wird in C++ geschrieben. Dies geschieht im Wesentlichen aus persönlicher Erfahrung mit C++ und Qt. Weiters ist der Linux-Support besser als bei anderen Sprachen wie Java oder C#.

6.5.1 Qt Framework

Das Qt-Framework [27] bietet sowohl die Möglichkeit der Erstellung von graphischen Oberflächen als auch viele weitere Funktionen, wie den Zugriff auf Netzwerke, die verwendete Javascript-Engine [26], den Zugriff auf serielle Schnittstellen. Als umfangreiches Framework ersetzt es einen Kombination aus unterschiedlichen Bibliotheken wie GTK für GUI, Boost für Netzwerk, Systemfunktionen

für serielle Schnittstellen, LUA oder V8 als Scriptsprache. Qt löst so gut wie alle Library-Probleme auf einen Schlag und ist zudem besser dokumentiert als die meisten anderen Bibliotheken. Dabei werden die folgenden Module Verwendet:

- **widgets:** Zum Zeichnen von graphischen Oberflächen
- **serialport:** Für die Kommunikation mit den Hardware Modulen über virtuelle COM-Ports (/dev/ttyUSB*)
- **script,scripttools:** Für die Javascript Ladeverfahren.
- **printsupport:** Für das speichern von Plots im PDF-Format.

6.5.2 Javascript

Als Scriptsprache für das implementieren von Ladeverfahren wird Javascript verwendet. Die Sprache ist besonders aus dem Bereich Webbrowser bekannte. Aufgrund dessen wird sie von vielen Leuten beherrscht. Weiters erlauben die meisten Implementierungen ein Sandboxing, also eine vollständige Abtrennung von allen Systemfunktionen. Dies ist sinnvoll damit ein Ladeverfahren nur auf die gewünschten Funktionen zugreifen kann. Weiters besitzt das Qt-Framework die Funktion um Javascript in die eigene Applikation zu integrieren. Ein Beispiel Ladeverfahren ist in Abbildung 7.6 auf Seite 87 zu finden.

Die größte Alternative im Anwendungsbereich der Eingebetteten Sprachen ist Lua [22]. Lua wird vor allem im Bereich von Computerspielen für das Scripting eingesetzt.

6.5.3 Debian Linux

Als Betriebssystem für den Steuercomputer kommt Raspbian [28] ein Debian/-Linux Derivat zum Einsatz. Dies ist einerseits das Standard Betriebssystem des RaspberryPi und andererseits mit dem von mir am PC eingesetzten Debian/Linux nahezu identisch. Diese Übereinstimmung vereinfacht die Entwicklung.

Grundsätzlich erlaubt der Einsatz einer Linux basierten Umgebung die Nutzung von allen Funktionen eines modernen Betriebssystems. So ist es mittels X-Server forwarding möglich die Applikation auf der finalen Hardware auszuführen, die graphische Oberfläche inklusiver der erweiterten Entwickleroptionen jedoch auf dem Entwicklercomputer anzuzeigen.

6.6 SerialProtokoll

Für die Kommunikation zwischen Steuer-PC, Steuerboard und Treibermodul wird ein eigenes Protokoll definiert.

Funktionen:

- Erkennen und Zusammenfügen von Paketen
- Sicherstellen der Datenintegrität mittels CRC
- Identifikation von Kommandos und decodieren der angehängten Daten

6.6.1 Header

Das Protokoll besteht aus einem Header und einem Block von Nutzdaten. Der Header ist in Abbildung 6.8 auf der nächsten Seite zu sehen.

Der Header enthält ein Start-Byte(0x55). Dies vereinfacht es den Paketstart in einem Datenstrom zu erkennen.

Weiters ist eine 16 Bit-CRC enthalten. Da das entsprechende CRC-Hardwaremodul des Mikrocontrollers noch nicht integriert wurde kommt keine Standard CRC zum Einsatz. Das verwendete Verfahren in Abbildung 6.7 liefert jedoch auch gute Ergebnisse.

Die Instruktion ist eine 8 Bit Enumeration und dient der Identifikation des Kommandos. In den meisten Fällen gibt es eine fixe Zuordnung zu einer Struktur und damit auch eine fixe Länge des Pakets.

Das Längenfeld gibt die Länge der Nutzdaten in Bytes an. Auch wenn in den meisten Fällen die Länge bereits aus der Instruktion bekannt wäre, erlaubt ein eigenes Längenfeld eine bessere Trennung des Protokolls von den Nutzdaten und auch eine flexiblere Nutzung.

```
1 void Crc16::addByte( uint8_t data ){
2     _state += data + ( (uint16_t)data << 8 );
3     _state %= 0xF047; // prime number 0xF047
4 }
```

Abbildung 6.7: Prüfsumme des Headers

6.6.2 Instruktionen

Die Instruktionen und Paketformate werden in einer einzelnen Headerdatei definiert. Dieser Header wird sowohl von der Steuersoftware als auch von der Mikrocontroller-Firmware inkludiert. Dies stellt sicher, dass beide Programme dieselben Paketdefinitionen und Instruktionen verwenden. `REGISTER_PACKET_STRUCT` wird verwendet um eine Verknüpfung zwischen der Struktur und der Instruktion herzustellen. Eine Beispiel ist in Abbildung 6.9 auf der nächsten Seite zu sehen.

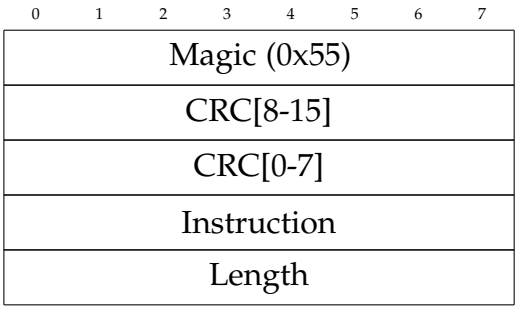


Abbildung 6.8: Nachrichtenheader eines Pakets

```
1  enum class Instruction : uint8_t {
2      RESERVED = 0, //Standard Instuctions
3      ERROR = 255,
4      PING = 254,
5      PONG = 253,
6      ...
7      DRIVER_SET_DISPLAY = 51, //User Instuctions
8      ...
9  }
10
11 struct __attribute__( ( __packed__ ) ) DriverSetDisplay {
12     struct {
13     uint8_t line:1;
14     uint8_t pos:4;
15     uint8_t clear:1;
16     } flags;
17     char data[17];
18 };
19 REGISTER_PACKET_STRUCT( DriverSetDisplay, Instruction::
       DRIVER_SET_DISPLAY )
```

Abbildung 6.9: Beispieldefinition eines Pakets. Instruction definiert alle Instruktionen.DriverSetDisplay definiert die Struktur einer Instruktion. REGISTER_PACKET_STRUCT verknüpft die Struktur und die Instruktion.

```
1  switch( packet.instruction ){
2     case PacketStructs::Instruction::DRIVER_SET_FAN_SPEED:{
3         PacketStructs::DriverSetFanSpeed msg;
4         packet.get( msg );
5         FAN::set( msg.speed );
6         return true;
7     }
8   ...//Other case Blocks
9
10    default: return false;
11    }
```

Abbildung 6.10: Beispiel für das Dekodieren der Pakete am Mikrocontroller. Zeile 3 Legt eine für die Daten passende Struktur an. Zeile 4 überprüft ob Struktur und Pakte zusammen Passen und Dekodiert das Paket.

6.6.3 Anwendungsbeispiel

Als Resultat ist das Decodieren von Nachrichten am Mikrocontroller sehr einfach. Der wesentliche Aufruf ist dabei ”packet.get(msg)”. Dabei überprüft die Bibliothek ob Instruktion und Struktur zusammenpassen, die Paketlängen übereinstimmen und befüllt anschließend die Struktur mit den empfangenen Daten. Ein Sourcecode ist in Abbildung 6.10 zu sehen.

In der Steuer-Applikation sieht das Ganze sehr ähnlich aus. Einziger Unterschied ist, dass sich hierbei jedes QObject beim Protokoll Stack registrieren kann um empfangen Pakete zu erhalten.

6.7 Messfunktionen

Für die Messung von Batterieparametern wird eine Reihen von Funktionen eingebaut.

- Messung von Akku und Zellenspannungen
- Messung des Lade- und Entladestroms
- Messung der Kapazität beim Laden und Entladen
- Aufzeichnung der Spannungen und Ströme. Daraus ergibt sich auch die Möglichkeit von kombinierten Auswertungen wie Zellenergie
- Kommando zur lastfreien Messung der Spannungen
- Kommando zur Messung der Spannungen und Ströme im Impulsbetrieb für die Impedanzmessung

6.7.1 Kontinuierliche Messung

Während des Ladevorganges wird kontinuierlich die Packspannung, die Zellspannungen sowie der Lade- und Entladestrom gemessen. Diese Messwerte bilden die Basis für die meisten Ladeverfahren. Die Messung aller Parameter erfolgt dabei unter den eingestellten Lastbedingungen zum Beispiel bei einem Ladestrom von 5A.

6.7.2 Lastfreie Messung

Für einige Ladeverfahren ist die Messung der Ruhespannung von Vorteil. Um dies dem Anwender einfach und flexibel zu ermöglichen, ist ein eigens Messkommando integriert. Dabei können drei Zeiten angegeben werden. Während der lastfreien Messung werden auch alle Balancer-Ausgänge lastfrei geschaltet.

- **sleepTime:** Zeit für die der Akku stromlos geschaltet wird bevor eine Messung erfolgt. Dies gibt einerseits de Elektronik Zeit den Strom auf null zu regeln und andererseits dem Akku Zeit sich dem chemischen Potential anzunähern.
- **measureTime:** Während dieses Zeitintervalls erfolgt eine Messung pro Millisekunde, der Mittelwert aller Messungen ergibt den finalen Messwert.
- **relaxationTime:** Mit dem Beginn der Erholzeit wird der vorherige Ladestrom sowie die Balancer Konfiguration wieder hergestellt. Diese Zeit gibt an wie lange das Kommando noch weiter läuft und damit eine normale zyklische Messung unterbindet. Dies soll verhindern, dass im Laufe der zyklischen Messung Störungen durch die Ausführung von Kommandos auftreten.

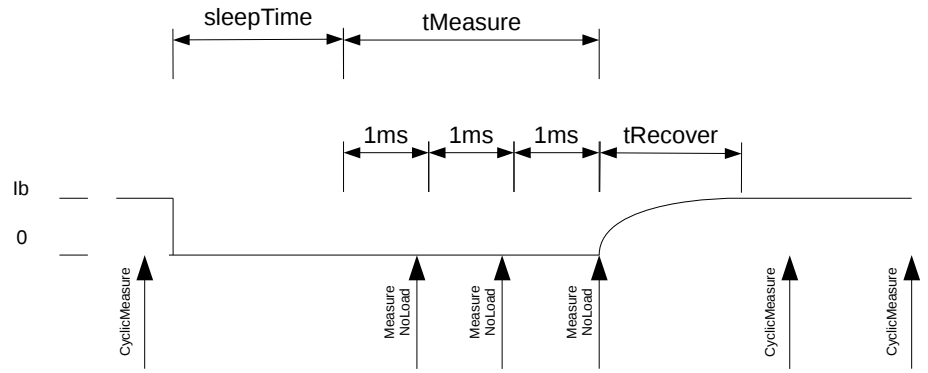


Abbildung 6.11: Lastfreie Spannungsmessung

6.7.3 Impedanzmessung

Für die Impedanzmessung wurde analog zur lastfreien Messung ein Kommando implementiert, das den momentanen Ladevorgang unterbricht. Dabei stellt das Ladegerät nur ein flexibles Messkommando zur Verfügung. Die Auswertung muss vom Nutzer beziehungsweise dem Ladeverfahren erfolgen. Das Kommando legt dabei einen rechteckförmigen Strom an den Akku an und misst die Spannungen und Ströme.

- **t1:** Erste Pulszeit
- **t2:** Zweite Puslzeit
- **current1:** Strom während des ersten Pulsabschnittes, der positiv und negativ sein kann
- **current2:** Strom während des zweiten Pulsabschnittes, der positiv und negativ sein kann
- **repititions:** Anzahl der Wiederholungen. Pro Wiederholung erfolgt eine Messung. Die Ergebnisse werden gemittelt.
- **tRecover:** Zeit zur Wiederherstellung der Bedingungen vor dem Messkommando.

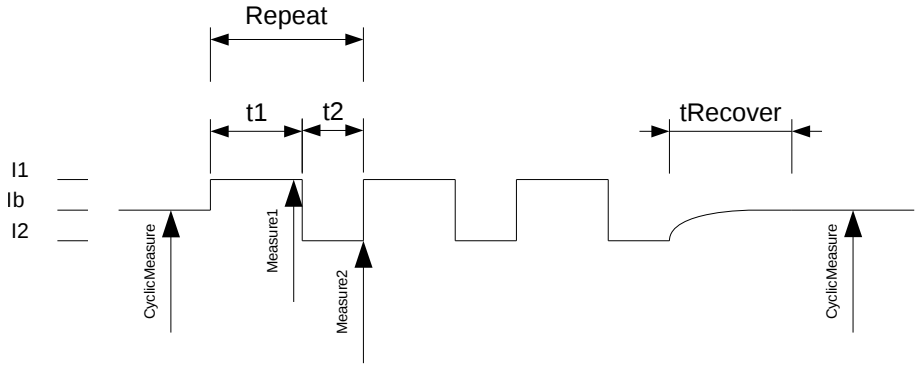


Abbildung 6.12: Schema der Impedanzmessung. I1 und I2 sind die beiden Messströme. Ib ist der Ladestrom vor dem Messkomando.

7 Ergebnisse

Im folgenden wird die aufgebauten Hard- und Software gezeigt. Weiters werden die Erfahrungen bei der Implementierung diskutiert und exemplarische Ladevorgänge gezeigt.

7.1 Testaufbau

Zu Testzwecken wurde die Hardware in doppelter Ausführung aufgebaut.

Dabei wird ein Hardwaresatz für die Entwicklung verwendet. Die Steuersoftware wird dabei auf dem Entwickler-PC ausgeführt. Dies erlaubt eine einfachere Fehlersuche der Steuersoftware. Die Testhardware ist in Abbildung 7.1 auf der nächsten Seite zu sehen und kommt dabei als offener Satz von Platinen zum Einsatz. Der offene Aufbau erleichtert dabei den Zugang zu Signalen und die Überwachung der Temperaturen von Leistungsbauteilen.

Mit dem zweiten Hardwaresatz wurde ein vollständiges Gerät inklusive Gehäuse aufgebaut (Abbildung 7.2 auf der nächsten Seite). Dieses dient für Demonstrationen, für längere Tests und zum Test der Benutzeroberfläche unter realen Umständen.

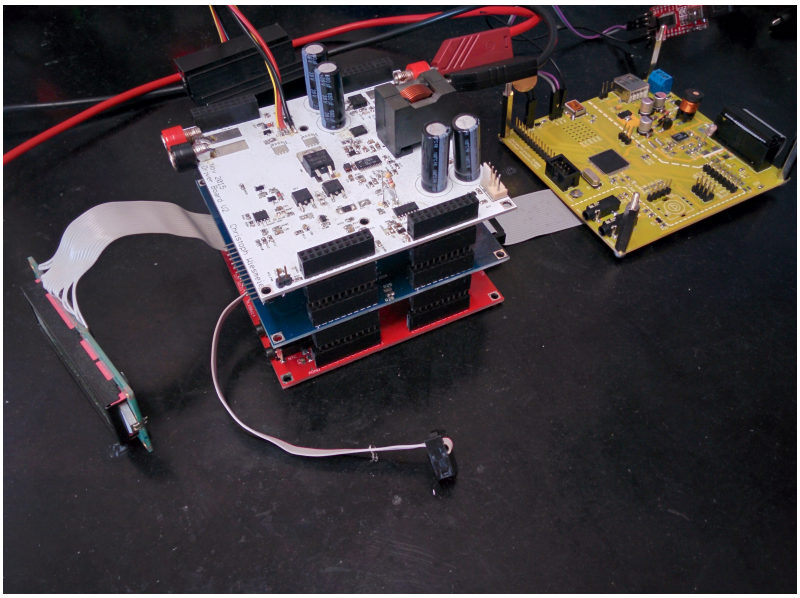


Abbildung 7.1: Entwicklungsaufbau der Hardware. Die gelbe Platine ist Teil des Steuergerätes. Über ein Flachbandkabel ist der Platinensatz des Treibermoduls angeschlossen. Das Treibermodul selbst beseht aus der weißen Platine (Oben) mit dem Leistungsteil, der blauen Platine (Mitte) mit dem Balancer und der Zellspannungsmessung, sowie der roten Platine (unten) für Kommunikation und Logik.

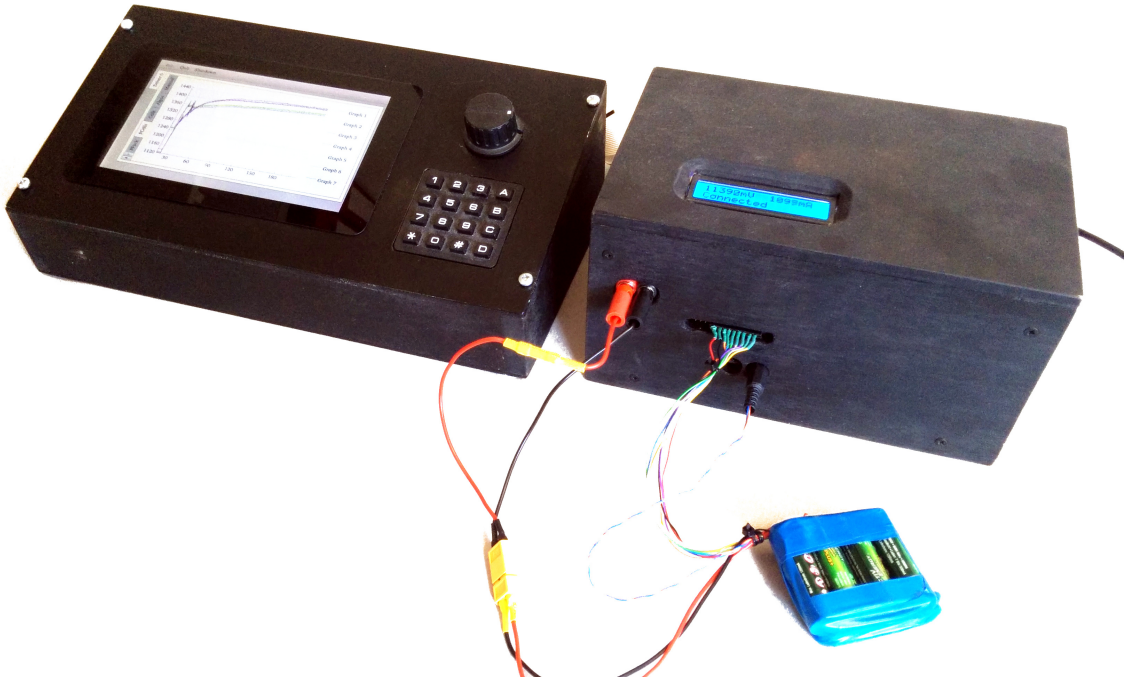


Abbildung 7.2: Demonstrationshardware: Links das Steuergerät und rechts das Treibermodul.

7.2 Hardware

Wie bereits im Kapitel 5 (Implementierung) erwähnt, wird bei der Hardware ein hoher Wert auf Fehlersuche und Flexibilität bei Problemen gelegt. Dieser Ansatz hat sich sehr bewährt. Es ermöglichte das nur eingeschränkt funktionierende Treiberboard Version 1 neu zu erstellen ohne die anderen Platinen ersetzen zu müssen. Im Falle eines Defektes ist es möglich eine defekte Platine zur Verifikation des Fehlers zu ersetzen. Weiters erlauben die Steckverbinder zwischen den Modulen ein Umordnen. Die aktuell zu messende Platine kann einfach an die oberste Stelle gesteckt werden, was den Zugang zu den Bauteilen erleichtert. Es bietet sich auch die Möglichkeit von Teilaufbauten. Es können entweder nur die Steuerboards verwendet werden, um die Buskommunikation zu testen, oder auch ein vereinfachtes Ladegerät ohne Balancer aufgebaut werden.

7.2.1 Funktionen

In der Hardware sind soweit alle geplanten Funktionen umgesetzt. Bei der Geschwindigkeit der Stromquelle und Stromsenke müssen jedoch Abstriche gemacht werden, da es ohne ausreichend starke Dämpfungsnetzwerke zu Schwingungen in der Stromregelung kommt.

7.2.2 Genauigkeit

Die Genauigkeit stellt leider eines der größten Probleme dar. Aufgrund des relativ großen Leistungsbereichs der Hardware von 0-10A und 0-30V kommt es notgedrungener weise zu Einschränkungen bei der Messgenauigkeit. So sind 1% Full-Range Fehler bereits 100mA bzw. 320mV. Etwas besser sieht die Situation bei den Zellspannungen aus, da der Eingangsspannungsbereich bei nur $0 - 5V$ liegt.

Ein großer Schritt zur Steigerung der Genauigkeit ist die Implementierung einer Kalibrierung. Durch die Steigungs- und Offset-Kalibrierung können die Fehler teilweise dramatisch reduziert werden. Die Kalibrierdaten werden dabei im EEPROM des Treibermoduls abgelegt. Es hat sich gezeigt, dass die Module eine ähnliche Kalibrierung benötigen. Daraus ergibt sich, dass auch wenn nicht jedes Gerät einzeln kalibriert wird, eine wesentliche Verbesserung der Genauigkeit.

Während des Ausbalancieren der Zellen kommt es zu Spannungsverschiebungen von einigen 100mV. Dies ist unvermeidlich aufgrund der Spannungsabfälle bei der oftmals dünnen Balancer-Leitungen bei Strömen von bis zu 1A. Um dies zu umgehen muss für das Ausbalancieren das lastfreie Messkommando zum Einsatz kommen. Dies deaktiviert den Balancer wehrend des Messvorganges.

Weiters treten bei Berührung und teilweise auch bei starken Temperaturschwankungen immer wieder Messschwankungen von einigen $100mV$ auf. Als Ursache werden dabei Änderungen der Übergangswiderstände zum Akku vermutet. Bei hohen Strömen führen hier Änderungen der Kontaktwiderstände schnell zu Schwankungen bei der Anschlussspannung. Weiters ist zu befürchten, dass das gewählte Design mit Steckverbindungen zwischen den Platinen anfällig für Kontaktprobleme ist.

Balancer Ground Offset

Die Masse-Anschlüsse des Balancerchips entsprechen der globalen, analogen und digitalen Masse des Moduls. Aufgrund von Spannungsabfällen auf den Anschlussleitungen des Akkus kommt es beim Laden zu einer Verschiebung des Potentiales vom Minuspol der Zelle 0 ($CV0$) und der Gerätemasse. Das Datenblatt des `MAX14920` würde hier eine Verschiebung von $\pm 0.3V$ erlauben. Ein Test mit Offsetspannungen von 0-750mV hat jedoch gezeigt dass diese direkt als Offsetspannung am Ausgang auftreten. Dies lägt nahe, dass der analoge Ausgang des ICs nicht auf $AGND$ sondern $CV0$ bezogen ist. Durch Spannungsabfälle auf den Zuleitungen des Akkus und am Messshunt kommt es beim Laden und Entladen teilweise zu massiven Offsets in der Zellspannungsmessung. Dies könnte mittels differenzieller Messung durch den ADC kompensiert werden.

Zur Lösung der Potentialverschiebung kann eine Potentialtrennung zwischen `MAX14920` und dem restlichen Board erfolgen. Dies könnte als tolerantes System nur durch Ausnutzung der Reserven auf den digitalen Signalpegeln oder durch eine saubere Potentialtrennung realisiert werden.

Im Allgemeinen zeigt sich jedoch, dass die Wahl des Chips möglicherweise nicht optimal ist, da der Chip kaum vor Fehlbeschaltung zu schützen ist. Vor allem eine Zellspannungsüberwachung bei NiMh bringt den Chip an die Grenzen der Spezifikation. Man könnte hier die alternative Schaltungsvariante Abschnitt 5.3.2.2 auf Seite 49 mit Analogmultiplexern verwenden.

7.2.3 Messproblematik durch Schaltregler

Die meisten analogen Spannungssignale liegen im Bereich der ADC und DAC Spannung von 0-$1V$. Aufgrund der Anforderungen an die Verlustleistung am Messshunt sogar nur bei $100mV$. Versucht man diese Signale extern zum Beispiel mit einem Oszilloskop zu messen, kommt es dabei zu massiven Störungen durch den Schaltregler. Als Hauptursache wird eine magnetische Einkopplung in die Schleife, die zwischen Spitze und dem Masseclip des Tastkopfes entsteht, vermutet. Eine Störung der am Board integrierten Messschaltungen scheint sich jedoch aufgrund der Masseflächen in Grenzen zu halten.

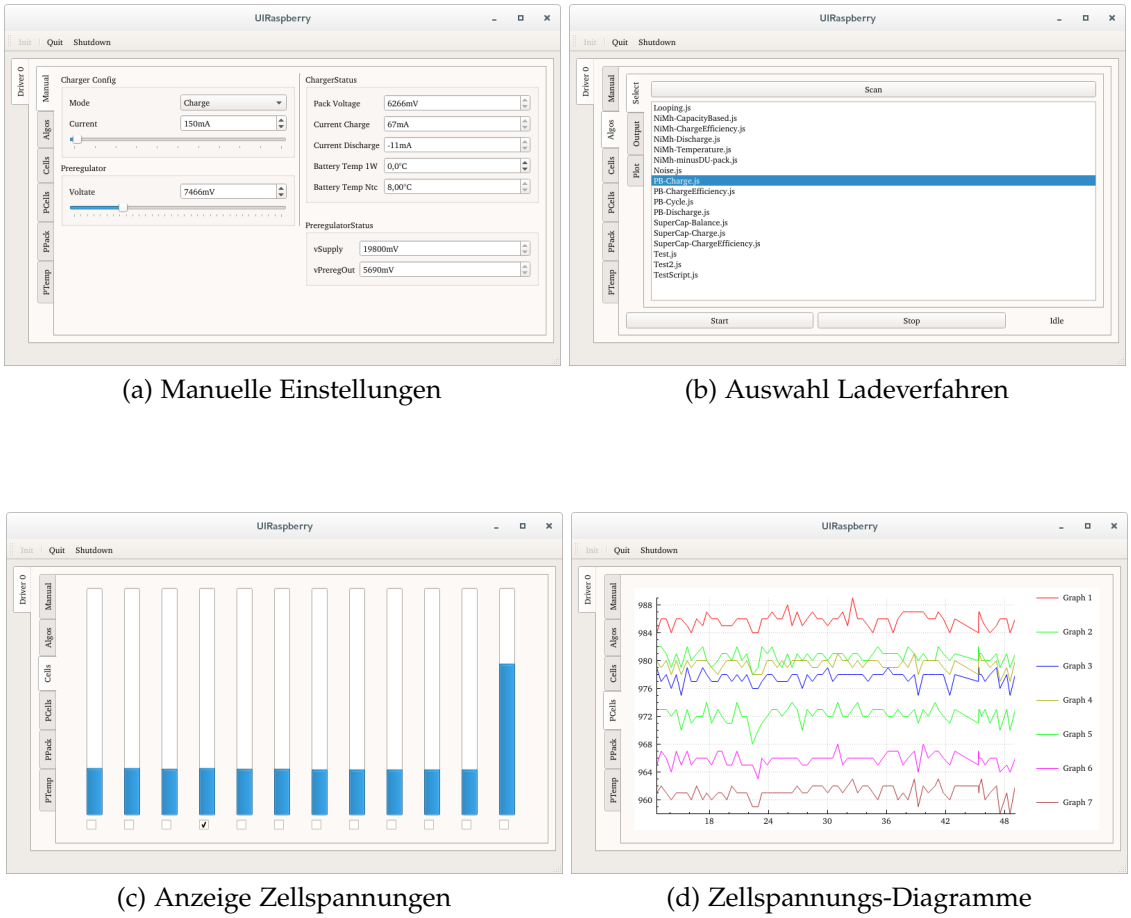
7.3 Software

Im folgenden wird die Entwickelte Steuersoftware im Vorgestellt. Dabei liegt der Fokus auf der graphischen Oberfläche und den Benutzerdefinierten Ladeverfahren.

7.3.1 Benutzeroberfläche GUI

Es wurden zwei Benutzeroberflächen implementiert. Eine reine Entwickler-GUI Abbildung 7.4 auf der nächsten Seite und eine für das fertige Ladegerät Abbildung 7.3.

Die Entwickleroberfläche ist nicht auf die Größe des Gerätedisplays begrenzt und bietet einige zusätzliche Einstellmöglichkeiten. Beide Oberflächen können gleichzeitig verwendet werden.



(a) Manuelle Einstellungen

(b) Auswahl Ladeverfahren

(c) Anzeige Zellspannungen

(d) Zellspannungs-Diagramme

Abbildung 7.3: Benutzeroberfläche des Ladegeräts.

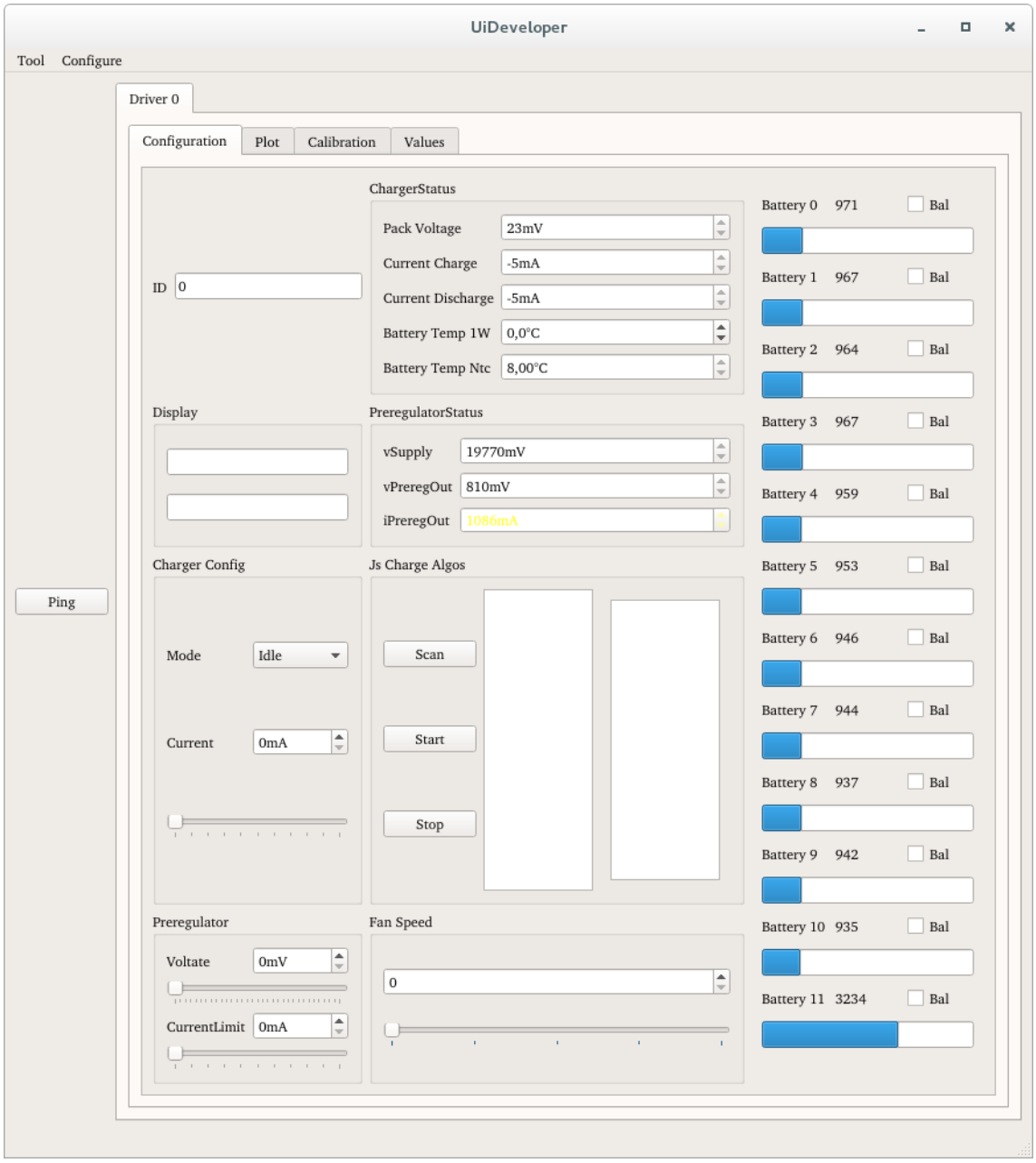


Abbildung 7.4: Benutzeroberfläche für Entwickler. Erlaubt einen den zugriff auf mehr Funktionen als die Oberfläche des Geräts.

7.3.2 Benutzerdefinierte Ladeverfahren

Die Ladeverfahren können mit Hilfe von Javascript vom Benutzer erstellt werden. Zu diesem Zweck werden die Eigenschaften und Funktionen der DriverModule-Klasse im Javascript zur Verfügung gestellt. Damit verfügt das Javascript im Wesentlichen über die selben Kontrollmöglichkeiten wie die GUIs. Durch die Implementierung mit QProperties erfolgt eine Synchronisation zwischen den Einstellwerte, Anzeigen, dem Javascript und der GUI. Durch diese Synchronisation erlaubt ein Blick auf die Einstellregler der GUI auch einen Einblick auf die Steuervorgaben des Javascriptes.

Die Programmierschnittstelle für Javascript ergibt sich im Wesentlichen aus den Möglichkeiten von Qt. Dabei erfolgt die Implementierung eventbasiert als asynchrone Schnittstelle. Dabei werden Funktionen für gewisse Events wie `startCharge`, `stopCharge`, `measurementFinished`, `timer` und weitere registriert. Da diese Implementierung direkt im Haupthread der Applikation läuft muss, diese asynchrone Implementierung eingehalten werden. Wird eine Endlosschleife in Javascript erstellt kommt es zum Stillstand der gesamten Applikation. Dass dies unabsichtlich passiert, ist jedoch äußerst unwahrscheinlich, da sich aufgrund der vorgegebenen Programmstruktur keine Notwendigkeit für die Verwendung von Schleifen ergibt. Werters weiters wird es solches Szenario vom Treibermodul mittels Watchdog-Timer erkannt und es erfolgt eine Sicherheitsabschaltung.

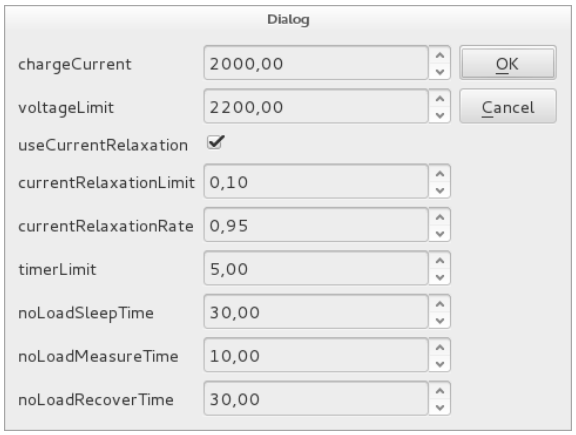


Abbildung 7.5: Javascript Einstellungs-Dialog. Wird automatisch vom Ladescript Erzeugt.

Ein Ladescript besteht im Wesentliche aus den folgenden Punkten:

- **Includes:** Lädt andere Javascriptfiles. Damit können zum Beispiel Zusatzfunktionen wie sprintf (Text Formatierung) oder CCCV.js (Standard Konstantstrom-Konstatspannung-Ladeverfahren) importiert werden.
- **Battery:** Ist eines von zwei Konfigurationsobjekten. Aus den Elementen dieses Objektes wird eine GUI erstellt und die Werte vor Beginn des Ladevorgangs von Benutzer abgefragt. Dialog siehe Abbildung 7.5.
- **Configuration:** Gleich wie Battery, nur für andere Konfigurationsvariablen.

- **onStartCharge:** Funktion, die beim Start des Ladevorganges aufgerufen wird. Erst in dieser Funktion sollten die restlichen Eventhandler registriert werden. Dies verhindert, dass sie bereits in der Konfigurationsphase aufgerufen werden.
- **Register Start Handler:** Registrieren einer Start- und Stopp-Funktion.

7.3.3 Nicht Implementierte Funktionen

Eine Reihe von in Betracht gezogenen Funktionen wurde nicht implementiert. Die Gründe dafür sind meist Zeitmangel sowie technische Probleme bei der Umsetzung.

7.3.3.1 NFC

Bei der Implementierung des NFC-Readers kam es zu einer ganzen Reihe von Problemen. Der NFC-Reader sollte ursprünglich über eine virtuelle, serielle Schnittstelle am Board des Steuermoduls angebunden werden. Da es aber bei der Implementierung von USB zu Problemen kam, wäre ein eigener USB-Seriell-Konverter nötig gewesen. Weiters implementiert die ursprünglich geplante libNFC [19] nur sehr elementare Funktionen. Vor allem ist es damit nicht ohne Weiteres möglich NDEF-Records zu schreiben. NDEF ist ein Format mit dem die gespeicherten Daten auch von Handys aus gelesen werden können.

Als Alternative käme die im März 2016 veröffentlichte Version 5.6 von Qt in Frage. Diese unterstützt die gewünschten Funktionen. Auf Linux basiert dies auf der Neard-Library [25]. Diese Library unterstützt aber den geplanten NFC-Reader `PN532` (NXP) nur in der Form von speziellen, kommerziellen USB-Sticks.

7.3.3.2 USB

Bei der Implementierung von USB kam es zu einer Reihen von Problemen. Um die USB-Funktionalität des Mikrocontrollers ohne komplette Implementierung eines USB-Protokoll-Stacks nutzen zu können, benötigt man des Atmel-Software-Framework (ASF). Dieses ist jedoch stark in die Entwicklungsumgebung Atmel Studio integriert, was eine Entwicklung unter Linux erschwert. Die Integration der relevanten Komponenten in das bestehende System war schließlich möglich. Leider kam es unabhängig von der verwendeten Entwicklungsumgebung immer wieder zu Problemen bei der Erkennung des Mikrocontrollers durch den PC. Windows konnte mit dem passenden Treiber mit den seriellen Schnittstellen kommunizieren. Unter Linux, der eigentlichen Zielplattform, kam es jedoch während der Enummerierung zu Problemen. So wurden die seriellen Schnittstellen erkannt, eine Kommunikation scheiterte jedoch trotz anfänglicher erfolgreicher

```
include("CCCV.js")
include("Prereg.js")
include("Sleep.js")
include("sprintf.js")

Battery = {
        cellcount : 3,
        chemistry : "PB",
        capacity : 4000 }
Configuration = {
    chargeRate : 0.5,
    ... }

function onStartCharge () {
    IO.log("Start Charger PB");
    Driver.resetDataStorage();

    var chargeAlgo = new CCCV(Battery.cellcount*Configuration.
       voltageLimit, Battery.capacity*Configuration.chargeRate)

    chargeAlgo.hooks.started = function(){
        Driver.resetDataStorage();
    }
    chargeAlgo.hooks.finished = function(){

        Driver.saveDataStorage(sprintf("PB charge from %s",(new
           Date).toLocaleString()));
        sleep.start();
    }

    var sleep = new SleepPackVoltageSattled(0.1,60);
  ... //Hooks für Sleep Phase

    var preregulation = new Prereg(1200);
    preregulation.start();
    chargeAlgo.start();
}

function onStopCharge(){ IO.log("StopCharge"); }

Charger.startCharge.connect(onStartCharge);
Charger.stopCharge.connect(onStopCharge);
```

Abbildung 7.6: Ladescript für Bleiakkus

```
1  IO.log("loaded CCCV.js")
2  include("sprintf.js")
3
4  CCCV = function CCCV(voltage,current){ //units are mV and mA
5      var that = this;
6      that.voltageLimit = voltage;
7      that.currentLimit = current;
8      that.rampupTime = 60 //sec
9      that.currentRelaxationLimit = 0.1;
10
11     that.hooks = {
12         started : new Function,
13         finished : new Function};
14
15     that.results = {
16         secondsCC : 0,
17         secondsCV : 0,
18         seconds : 0,
19         chargeAdded : 0,
20         chargeAddedCC : 0,
21         chargeAddedCV : 0}
22
23     that.mode = "idle";
24     that.time = NaN;
25     that.currentCurrent = 0;
26
27     that.start = function() {
28         that.mode = "rampup";
29         that.time = Date.now();
30         Driver.measuredPackVoltageChanged.connect(that.
               onNewMeasuredVoltage);
31         IO.log("Start CCCV charching");
32         that.hooks.started();};
33
34     that.stop = function() {
35         Charger.idle(); //stop charching but not the script
36         Driver.measuredPackVoltageChanged.disconnect(that.
               onNewMeasuredVoltage);
37         IO.log("Stop CCCV charching");
38         that.hooks.finished();};
39
40   ... // Implementierung in Abbildung 7.8 auf der nächsten Seite
41 }
```

Abbildung 7.7: Constantstrom/Constantspannungs Ladevefahren (CCCV). Setup Code. Der Kern des Verfahrens ist in Abbildung 7.8 auf der nächsten Seite

```
1  that.onNewMeasuredVoltage = function() {
2    var voltage = Driver.measuredPackVoltage;
3    if(that.mode === "rampup"){
4      if(voltage > that.voltageLimit){
5        that.mode = "CV";
6        IO.log("VoltageLimitReached during Rampup Phase, Battery
             proberbly already charged");
7        return;
8      }
9      else{
10       var timeElapsed = (Date.now() - that.time)/1000 ;
11       if( timeElapsed < that.rampupTime)
12       {
13         that.currentCurrent = that.currentLimit * timeElapsed/that.
             rampupTime;
14         Charger.charge(that.currentCurrent/1000);
15         IO.log(sprintf("Rampup: %d%% , %dmA",timeElapsed/that.
             rampupTime*100,that.currentCurrent));
16       }
17       else{
18         that.mode = "CC";
19         that.currentCurrent = that.currentLimit;
20         Charger.charge(that.currentCurrent/1000);
21         IO.log("Finised Rampup Phase, now switching to constant
             current phase");
22   } } }
23   if(that.mode === "CC"){
24     if(voltage > that.voltageLimit){
25       that.mode = "CV";
26       IO.log("Voltage limit reached continue with constant voltage
             charge");
27       return;
28   } }
29   if(that.mode === "CV"){
30     if(voltage > that.voltageLimit){
31       that.currentCurrent *= 0.99;
32       Charger.charge(that.currentCurrent/1000);
33
34       if(that.currentCurrent < that.currentLimit*that.
             currentRelaxationLimit){
35         IO.log("current limit reached, stop charching");
36         that.stop();
37 } } }    };
```

Abbildung 7.8: Constantstrom/Constantspannungs onMeasuredVoltage (Eigentliches Verfahren)

Tests. Es wurde schlussendlich der Alternativplan umgesetzt und ein USB-Seriell-Wandler FT232R (FTDI) eingesetzt. Das Keyboard und der Drehgeber konnten somit nicht direkt als Tastatur am PC erkannt werden. Somit sind diese nur in der eigenen Applikation funktionstüchtig.

7.3.3.3 Datenaufzeichnung über die Lebenszeit der Batterien

Die geplante Funktion, Batterieparameter über die Lebenszeit des Akkus zu verfolgen, wurde aus Zeitgründen nicht mehr implementiert. Dabei stellte insbesondere der Zeitaufwand für das Testen ein Problem dar. Um für Akkus interessante Ladezyklenzahlen mindestens 50 Zyklen zu erreichen, würde einen Zweitaufrand von mehreren 100 Stunden bedeuten. Aus Sicherheitsgründen wurde das Gerät jedoch nicht unbeaufsichtigt betrieben.

7.3.3.4 Webinterface

Im Anfangsstadium der Arbeit wurde ein Webserver für die Steuerung in das Ladegerät integriert. Dieser verwendete als Basis die Webserverimplementierung tufao [30]. Die Bibliothek wurde um eine an django angelehnte Template-Engine erweitert. Diese erlaubt mit Funktionen wie Template-Inheritence eine einfache Erzeugung moderner Webseiten mit konsistentem Design. Als im laufe des Projektes ein umstieg vom 2.8 Zoll auf ein 7 Zoll Display erfolgte, konnte eine guten Bedienbarkeit direkt am Gerät gewährleistet werden. Der Arbeitsaufwand für das zusätzliche Interface wurde daraufhin eingespart.

7.3.3.5 Temperaturüberwachung der Kühlkörper

Um einen leisen Betrieb des Gerätes zu ermöglichen, war eine Temperatur basierte Lüftersteuerung geplant. Jedoch besitzt die Hardware nur einen Eingang für einen Temperaturfühler. Die Regeltransistoren für Laden und Entladen sind jedoch auf unterschiedlichen Kühlkörpern angebracht. Weiters stellte sich heraus, dass die verwendeten 4-PIN PWM-Lüfter nicht bis zum Stillstand geregelt werden können. Da die Kühlung bei fixer, aber geringer Lüfterdrehzahl bereits ausreicht wird trotz des relativ geringen Aufwandes, auf eine Implementierung verzichtet. Unter anderem, da eine reine leistungsbasierte Steuerung bereits möglich ist.

7.4 Beispiel Ladevorgänge

Im folgenden wird beispielhaft der Ladevorgang eines NiMh-Akkupacks gezeigt. Abbildung 7.9 auf der nächsten Seite zeigt dabei die Parameter des Akkupacks und Abbildung 7.10 auf der nächsten Seite die Zellspannungen. In beiden Fällen

ist der für diesen Batterietyp charakteristische Verlauf zu erkennen. Am Ende erfolgte eine Abschaltung durch das $-\Delta U$ Ladeverfahren.

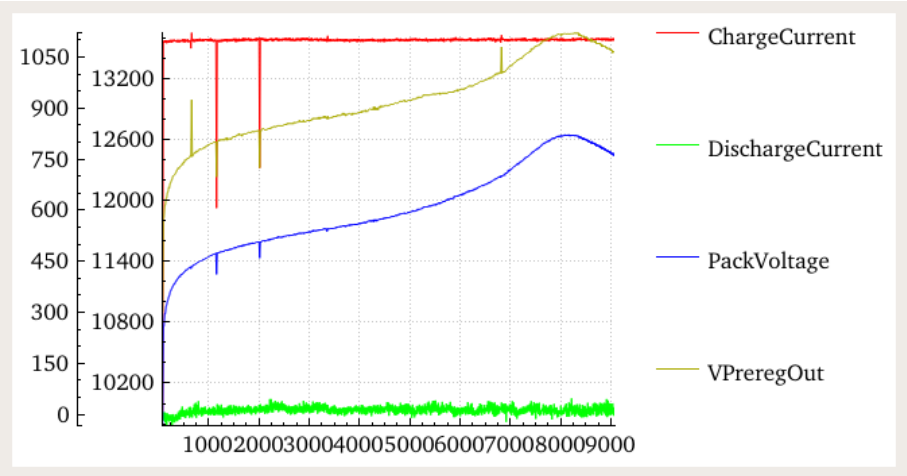


Abbildung 7.9: Ladevorgang eines NiMh Akkus. Linke Achse Strom in mA. Rechte Achse Spannung in mV. Unten Zeit in ms.

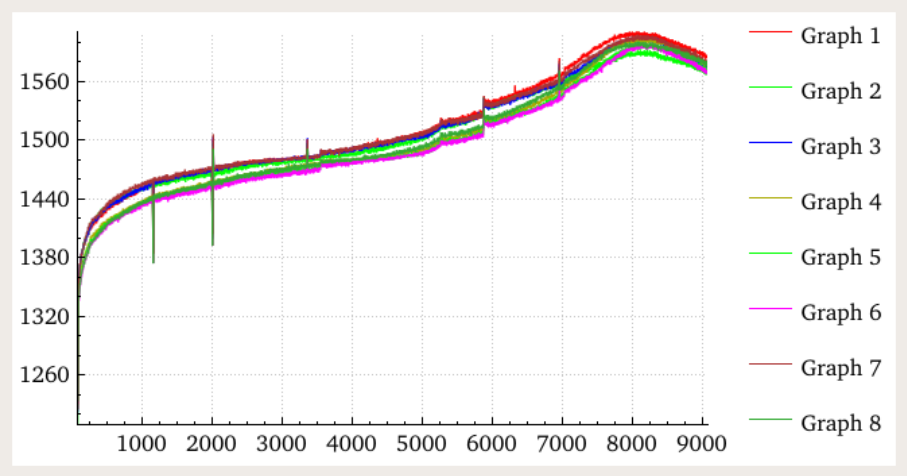


Abbildung 7.10: Ladevorgang eines NiMh Akkus (Zellenspannungen). Spannungen in mV über Zeit in Sekunden.

7.5 Ideen zur Weitereinwicklung

- **Lüftersteurerung:** Implementierung einer temperaturbasierten Lüftersteuerung. Dabei könnte als Trick für die Temperaturmessung eine Parallelschaltung aus zwei NTC-Fühlern verwendet werden. Aufgrund er nichtlinearen Charakteristik würde sich eine Art mittlerer Messewert ergeben der die höhere Temperatur stärker gewichtet. Im Rahmen einer Boardänderung wäre ein Transistor zum Abschalten des Lüfters vorzusehen.
- **Regelschleifen:** Momentan erfolgt eine reine Steuerung von Ladestrom, Entladestrom und Vorregler-Spannung. Da die Steuerkreise jedoch eine deutlich geringere Genauigkeit wie die Messkreise besitzen, wäre eine integrierte Regelung vorteilhaft. Als schnelle Umsetzung ist momentan eine Implementierung als Teil des Ladeverfahren möglich. Dies passiert momentan bei der Vorregler-Spannung.
- **Erweitertes Interface und Webserver:** Momentan sind die in Javascript geschriebenen Ladeverfahren, sowie die von ihnen exportierten Daten, nur in einem Verzeichnis gespeichert. Die aktuelle Software erlaubt kein verändern dieser Dateien. Als Möglichkeit für einen Zugriff könnte einerseits die GUI erweitert werden, der vermutlich bessere Weg für diesen Zweck wäre den integrierten Webserver in Betrieb zu nehmen um ein Arbeiten auf einem anderen Rechner zu ermöglichen.
- **Schutz vor Bedienfehlern:** Der verwendete Balancer-IC bietet zwar alle benötigten Funktionen in einem Chip, jedoch ist ein Schutz vor Verpolung oder falschem Anschluss mit diesem Bauteil nicht ohne weiteres möglich. Ein Austausch durch eine andere Schaltungsvariante wäre zu evaluieren.
- **ESD und EMV:** Zum Schutz vor elektrostatischen Entladungen wurden zwar Vorkehrungen getroffen, jedoch nicht getestet. Weiters wurde bei einigen Ausgängen auf die Schutzwirkung der Stützkondensatoren gesetzt. Die Wirksamkeit dieser Schutzmaßnahmen gilt es zu testen. Auch die Konformität des Gerätes mit den geltenden EMV und Sicherheitsrichtlinien gilt es zu überprüfen.
- **Alternatives Treibermodul:** Das aktuelle Treiberdmodul ist nur für das Laden von Akkupacks gedacht. Die grundsätzliche Struktur der Applikation ist jedoch nicht auf dieses eine Modul beschränkt. Ausgehend von diesem Standpunk wäre es interessant ein zusätzliches Modul zum Laden von einzelnen Zellen zu bauen und in das gegebene Framework zu integrieren. Es könnten auch andere Versionen des aktuellen Moduls aufgebaut werden, bei denen zum Beispiel der Balanceer oder die Entladefunktion weggelassen wird.

Literatur

[1] Wolfgang Weydanz Andreas Jossen. *Moderne Akkumulatoren richtig einsetzen*. Hrsg. von Philipp Küchler. 1. Aufl. Wolfgang Weydanz, 2006. ISBN: 978-3-939359-11-1 (siehe S. 3, 4, 8, 10, 12, 14, 15, 19).

[2] Atmel. *Application Note (AVR1017): USB Hardware Design Recommendations*. (Besucht am 20. 11. 2015) (siehe S. 28).

[3] Atmel. *Application Note (AVR318): Dallas 1-Wire master*. URL: www.atmel.com/images/doc2579.pdf (besucht am 06. 04. 2016) (siehe S. 42).

[4] Atmel. *Application Note (AVR4030): Atmel Software Framework - Reference Manual*. (Besucht am 20. 11. 2015) (siehe S. 28).

[5] Atmel. *Application Note (AVR4905): ASF - USB Device HID Generic*. (Besucht am 20. 11. 2015) (siehe S. 28).

[6] Atmel. *Application Note (AVR4907): ASF - USB Device CDC Application*. (Besucht am 20. 11. 2015) (siehe S. 28).

[7] Atmel. *Datasheet: XMEGA128A3U*. (Mikrocontroller). URL: http://www.atmel.com/Images/Atmel-8386-8-and-16-bit-AVR-Microcontroller-ATxmega64A3U-128A3U-192A3U-256A3U_datasheet.pdf (besucht am 20. 04. 2016) (siehe S. 37).

[8] Jan Axelson. *USB Complete*. Third Edition. Lakeview Research LCC, 2005 (siehe S. 25, 28).

[9] *CMake*. (Buildsystem für C++). URL: https://cmake.org/documentation/ (besucht am 21. 04. 2016) (siehe S. 65).

[10] Thomas b. Reddy David Linden. *Handbook Of Batteries*. Hrsg. von 3. McGraw-Hill, 2001. ISBN: 0-07-135978-8 (siehe S. 5).

[11] *Debian GNU/Linux*. (Linux Distribution). URL: https://www.debian.org (besucht am 19. 04. 2016) (siehe S. 71).

[12] Müller GMBH. *Der Aufbau der neuen AccuCell Batterien*. URL: http://www.reichelt.de/?;ACTION=7;LA=6;OPEN=0;INDEX=0;FILENAME=D400%2FAccuCell_Datenblatt.pdf (besucht am 06. 04. 2016) (siehe S. 5).

[13] Ke Gong u. a. »A zinc-iron redox-flow battery under $100 per kW h of system capital cost«. In: *Energy Environ. Sci.* 8 (10 2015), S. 2941–2945. DOI: 10.1039/C5EE02315G (siehe S. 3, 5).

[14] H.A.-H. Hussein und I. Batarseh. »A Review of Charging Algorithms for Nickel and Lithium Battery Chargers«. In: *Vehicular Technology, IEEE Transactions on* 60.3 (März 2011), S. 830–838. ISSN: 0018-9545. DOI: 10.1109/TVT.2011.2106527 (siehe S. 17, 20).

[15] Infineon. *Datasheet: ESD5V3U4RRS*. (Ultra-Low Capacitance ESD Diode Array). URL: http://www.infineon.com/dgdl/Infineon-ESD5V3U4RRS-DS-v01_01-en.pdf?fileId=db3a30431c69a49d011cad6ae0c10a10 (besucht am 20. 04. 2016) (siehe S. 43, 44).

[16] Maxim Integrated. *Datasheet: DS18B20*. (Termometer). URL: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf (besucht am 06. 04. 2016) (siehe S. 42).

[17] Maxim Integrated. *Datasheet: MAX14920*. (Battery Analog Frontend Chip). URL: https://datasheets.maximintegrated.com/en/ds/MAX14920-MAX14921.pdf (besucht am 11. 04. 2016) (siehe S. 21).

[18] Intel. *4-Wire Pulse Width Modulation (PWM) Controlled Fans*. URL: http://www.formfactors.org/developer%5Cspecs%5Crev1_2_public.pdf (besucht am 06. 04. 2016) (siehe S. 54).

[19] *libNFC*. (NFC Biblioteke). URL: http://www.libnfc.org/api/ (besucht am 21. 04. 2016) (siehe S. 86).

[20] Meng-Chang Lin u. a. »An ultrafast rechargeable aluminium-ion battery«. In: *Nature* 520.7547 (2015), S. 324–328. ISSN: 0028-0836. URL: http://dx.doi.org/10.1038/nature14340 (siehe S. 17).

[21] *LTSpice IV*. (Schaltungssimulationsprogramm). URL: http://www.linear.com/designtools/software/ (besucht am 20. 04. 2016) (siehe S. 38, 55).

[22] *LUA*. (Programmiersprache). URL: https://www.lua.org/ (besucht am 21. 04. 2016) (siehe S. 72).

[23] Microchip. *Datasheet: MCP4922*. (Dual Voltage Output Digital-to-Analog Converter with SPI Interface). URL: http://ww1.microchip.com/downloads/en/DeviceDoc/22250A.pdf (besucht am 20. 04. 2016) (siehe S. 55).

[24] Armin Najmabadi. »Evaluation of Active Balancing Algorithms and an Improved Method for a Deployed Active Battery Balancer as Well as Physical Implementation«. Magisterarb. McMaster University, 2013 (siehe S. 22, 24).

[25] *NEARD*. (Linux NFC System). URL: https://01.org/linux-nfc (besucht am 21. 04. 2016) (siehe S. 86).

[26] *QScriptEngine*. (Javascript Bibliothek). URL: http://doc.qt.io/qt-5/qscriptengine.html (besucht am 19. 04. 2016) (siehe S. 71).

[27] *Qt/Toolkit*. Version 5.5 (Graphic Toolkit). URL: https://www.qt.io/ (besucht am 19. 04. 2016) (siehe S. 71).

[28] *Raspbian (Debian/Linux)*. (Betriebsystem für RaspberryPi). URL: https://www.raspberrypi.org/downloads/raspbian/ (besucht am 21. 04. 2016) (siehe S. 72).

[29] Benedikt Sauter. »USB Grundkurs«. In: *Embedded Projects Journal* 1 (Juni 2008), S. 22–25. URL: http://journal.embedded-projects.net/index.php?module=archiv&action=pdf&id=1 (besucht am 06. 04. 2016) (siehe S. 25).

[30] *tufao*. (Webserver für Qt Programme). URL: https://github.com/vinipsmaker/tufao (besucht am 21. 04. 2016) (siehe S. 90).

[31] *Wikipedia / Ladeverfahren*. URL: http://de.wikipedia.org/w/index.php?title=Ladeverfahren&oldid=129677884 (besucht am 07. 05. 2014) (siehe S. 17).

[32] Phillip W Yury Podrazhansky. »Rapid Battery Charger, Discharger and Conditioner«. EN. 4829225. 1989 (siehe S. 18).

Glossar

ADC
: Analog to Digital Converter.

C-Rate
: Wird für die angaben von Lade-/Entladeströmen verwendet. Dabei ist 1C die Batteriekapazität durch 1 Stunde. Somit gilt 4C = 20A für einen Akku mit 5Ah.

CRC
: Cyclic Redundancy Check: Ist ein Verfahren zur Überprüfung der Dateninteg rität.

DAC
: Digital to Analog Converter.

EMV
: ElektroMagnetische Verträglichkeit: Fähigkeit eines Gerätes in seiner Umgebung fehlerfrei zu Funktionieren ohne andere Geräte durch Elektromagnetische Abstrahlung zu Stören oder Selbst gestört zu werden.

ESD
: ElektroStatic DischargeEin von den Meisten Mobiltelefonen unterstütztes Datenformat zum speichern Informationen wie Text, Webadressen oder Kontaktinformationen.

ESR
: Equivalent Serial Resistance: Gibt den parasitären Innenwiderstand eines Kondensators an.

FET
: Feld Effekt Transistor.

GPIO
: Gerneral Purose Input/Output: Bezeichnet einen normalen Ein-/Ausgang beim ICs und Mikrocontrollern. Unterscheidet sich dabei von Ausgängen mit spezieller Funktionalität wie Timer, SPI, I^2C.

I^2C
: Inter-Integrated Circuit bus: I Quadrat C ist ein Bussystem für die Kommunikation zwischen Mikrochips einer Schaltung.

IC
: Integrated Circuit: Mikrochchips bei denen komplexere analoge oder digitale Schaltungen eingebaut sind.

MISO
: Master In Slave Out: Ist eine vor allen bei SPI übliche Bezeichnung von Datenleitungen. Dabei wird die Datenflussrichtung beschrieben.

MOSI
: Master Out Slave In: Ist eine vor allen bei SPI übliche Bezeichnung von Datenleitungen. Dabei wird die Datenflussrichtung beschrieben.

NDEF
: NFC Data Exchange Format: Ein von den meisten Mobiltelefonen unterstütztes Datenformat zum speichern von Informationen wie Text, Webadressen oder Kontaktinformationen.

NFC
: Near Field Communication: Eine Kontaktlose Datenübertragung die überlicherweise nach den Standard ISO 14443 abläuft.

NTC
: Negative Temperature Coefficient: Wird häufig als Kurzform für einen NTC-Temperatur Fühler verwendet.

OPV
: OPerations Verstärker.

PCB
: Printed Circuit Board: Im Deutschen oft auch als Printplatte oder Leiterplatte bezeichnet.

PLL
: Phase Locked Loop: Eine üblicherweise zur Takterzeugung verwendete Schaltung mit deren Hilfe aus einer Frequenz eine höhere Frequenz abgeleitet werden kann.

RAM
: Random Access Memory.

RISC
: Reduced Instruction Set Computer: Beschreibt eine CPU-Architektur bei der die meisten Befehle meist nur einfache Adressierungsarten unterstützen.

SMT
: Surface Mounted Technology: Teilweise auch SMD für Surface Mounted Device. Heute übliche Befestigungsvariante für Bauteile auf Leitplatten.

SPI
: Serial Peripheral Interface: Ist ein Bussystem das typischerweise zur Kommunikation zwischen Chips auf einer Printplatte verwendet wird.

Spice
: Simulation Program with Integrated Circuit Emphasis: Ist ein weit verarbeitetes Simulationsprogramm für Elektronische Schaltungen.

TQFP
: Thin Quad Flat Package: Ist eine spezielle Art eines SMT Gehäuses.

USART
: Universal Synchronous and Asynchronous serial Receiver and Transmitter: Ein bei Mikrocontrollern übliches Peripheriemodul das für eine viel zahl von Schnittstellen wie RS232 oder SPI eingesetzt werden kann.

Zyklentiefe
: Beschreibt wie viele Prozent des Akkus bei jedem Ladezyklus entladen und wieder geladen werden.