HTWK

Hochschule für Technik, Wirtschaft und Kultur

Fakultät Ingenieurwissenschaften

Projektarbeit

Embedded Systems 3

Thema:	Entwurf eines Re-EMF-Ladegerätes für Bleiakkumulatoren
Autor:	Leon Arnecke, Johannes Trummer MatNr. 79511, 87051
Version vom:	29. September 2025
Prüfer:	Prof. Dr.-Ing Andreas Pretschner

Inhaltsverzeichnis

1 Einleitung

Bleiakkumulatoren gehören trotz ihres Alters nach wie vor zu den am häufigsten eingesetzten Energiespeichern. Ihre einfache Bauweise, robuste Charakteristik und vergleichsweise niedrigen Kosten machen sie für viele Einsatzzwecke attraktiv. Damit einher geht die Notwendigkeit einer zuverlässigen und effizienten Ladeelektronik, die sowohl die Lebensdauer der Akkumulatoren verlängert als auch eine sichere Regeneration tiefentladener Zellen ermöglicht.

Im Rahmen dieses Projekts wurde eine Re-EMF-Schaltung zur Ladung und Regeneration von Bleiakkumulatoren entwickelt. Ziel war es, den Ladevorgang durch Nutzung der Gegen-Elektromotorischen Kraft (Re-EMF) zu optimieren und ein schonendes, kontrolliertes Ladeverfahren zu realisieren. Die vollständige Schaltung wurde eigenständig entworfen und als Leiterplatte gefertigt.

Die Steuerung des Ladevorgangs übernimmt ein STM32MP135 Mikroprozessorboard. Es läuft unter einem Linux-Betriebssystem, das mit Hilfe des Yocto-Projekts speziell für diesen Anwendungsfall erstellt wurde. Der zugehörige Ladealgorithmus wurde in der Programmiersprache C implementiert und auf die zugrunde liegende Hardware abgestimmt.

Diese Arbeit dokumentiert die Entwicklung des Gesamtsystems – von der Konzeption der Ladeelektronik über die Realisierung der Steuerung bis hin zur Softwareumsetzung. Dabei liegt der Fokus auf dem Zusammenspiel zwischen Hardwaredesign und softwarebasierter Steuerung zur sicheren und effektiven Ladung von Bleiakkumulatoren.

2 Design des PCB

2.1 Designprozess

Die vorliegende Leiterplatte wurde mit der Software Fusion 360 im integrierten Modul PCB Design entworfen. Für die Umsetzung kam eine kostenlose Studentenlizenz zum Einsatz, die den vollen Funktionsumfang der Software für nicht-kommerzielle Bildungszwecke bereitstellt. Die Nutzung dieser Lizenz ermöglichte eine professionelle Umsetzung des Designs unter Berücksichtigung aktueller Industriestandards.

2.2 Entwurf des Schaltplans

Der Schaltplan wurde im Fusion 360 PCB Design-Modul erstellt und bildet die Grundlage für das gesamte Platinenlayout. Da viele der benötigten Bauteile nicht in den Standardbibliotheken enthalten waren, mussten sowohl die Schaltsymbole als auch die zugehörigen Footprints manuell erstellt werden. Die dafür erforderlichen Informationen wurden sorgfältig den jeweiligen Datenblättern der Hersteller entnommen. Diese Vorgehensweise stellte sicher, dass alle Bauteilabmessungen, Pinbelegungen und elektrischen Eigenschaften korrekt umgesetzt wurden. Trotz des erhöhten Aufwands bot Fusion 360 durch die enge Verknüpfung von Schaltplan und Layout eine effiziente Umgebung für die Entwicklung eines konsistenten und funktionsfähigen Designs.
In Abbildung 1 ist ein Ausschnitt des Schaltplans zu sehen, wie er in Fusion 360 PCB erstellt wurde. Die Symbole sind hauptsächlich eigens erstellt.

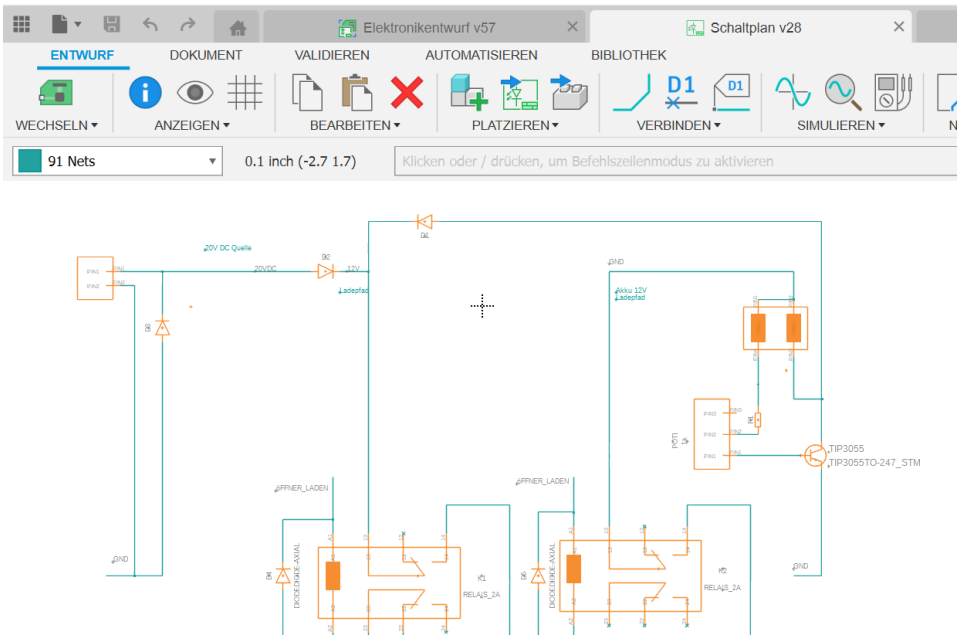

Abbildung 1: Ausschnitt des erstellten Schaltplans in Fusion 360 PCB Design

2.3 Übertragung des Schaltplans auf die Platine

Nach Fertigstellung des Schaltplans erfolgte die Übertragung in das Leiterplatten-Layoutmodul von Fusion 360. Dabei wurde die im Schaltplan definierte Netzliste, die alle elektrischen Verbindungen zwischen den Bauteilen beschreibt, automatisch in das PCB-Design übernommen. Diese Netze dienten im Layoutprozess als Grundlage für die spätere Leitungsführung (Routing).

Zunächst wurde die Grundfläche der Platine definiert, basierend auf den Platzanforderungen der Komponenten sowie den angestrebten mechanischen Abmessungen. Für den Entwurf wurde von einer industriellen Fertigung ausgegangen, weshalb eine zweilagige Platine (2-Layer PCB) angenommen wurde – ein gängiger Standard für viele elektronische Anwendungen.

Die Platzierung der Komponenten auf der Platine erfolgte unter Berücksichtigung sinnvoller Signalflüsse, minimaler Leitungslängen sowie thermischer und mechanischer Aspekte. Dabei wurde insbesondere auf eine übersichtliche und funktionsorientierte Anordnung geachtet, um spätere Tests, Fehlersuche und gegebenenfalls eine Bestückung per Hand zu erleichtern.

Die zuvor im Schaltplan definierten Netzverbindungen wurden im Layout als Luftlinien dargestellt, die als visuelle Hilfe für das Routing dienten. Diese Verbindungen wurden anschließend manuell auf den beiden verfügbaren Kupferlagen geroutet, wobei möglichst wenige Vias (Durchkontaktierungen) eingesetzt wurden, um Produktionskosten und mögliche Fehlerquellen zu minimieren.

Abbildung 2 zeigt die fertig erstellte Platine im PCB-Designer in Fusion 360 PCB. Die Löcher zum Befestigen am STM32MP135F-DK wurden aus den, von STM bereitgestellten, Unterlagen bezogen und auf der Platine platziert. Außerdem die Platine zum Schluss mit einem GND-Polygon versehen, um Störungen zu minimieren.

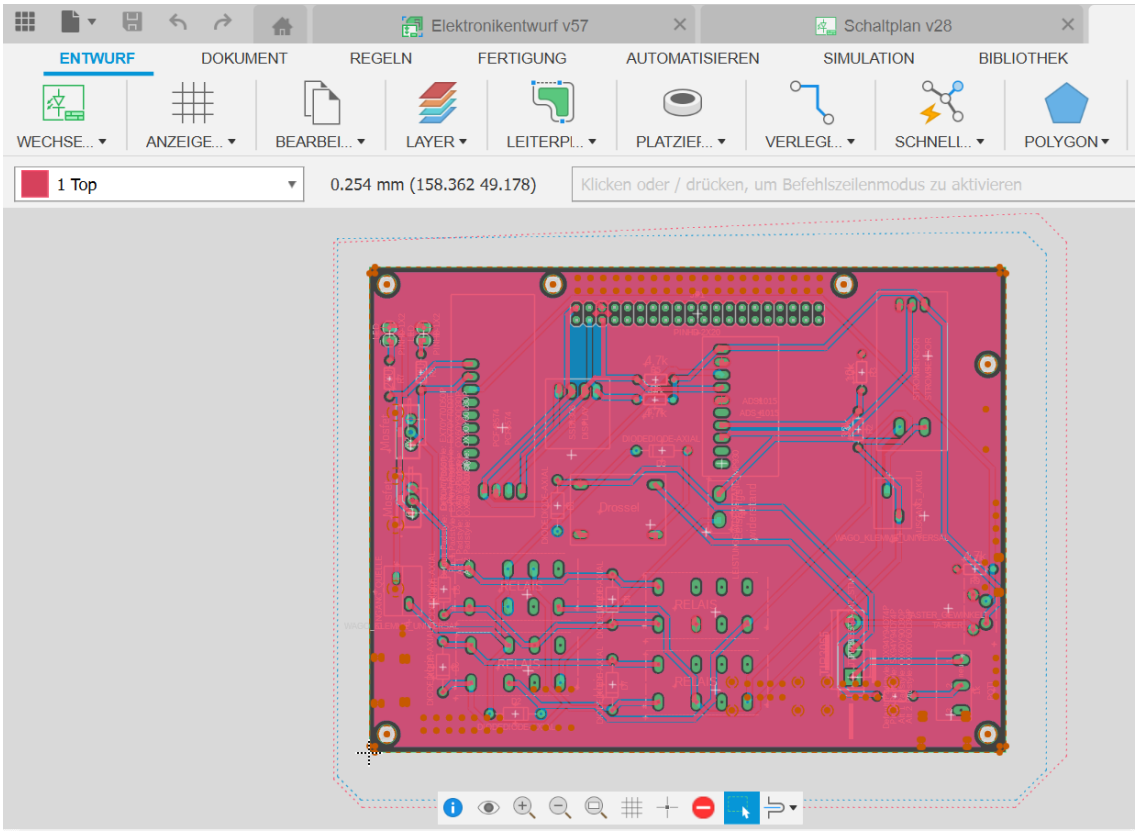

Abbildung 2: Fertig erstellte Platine mit Umrissen des Polygons

2.4 Bestellung der Platine und der Elektronikteile

Für die Fertigung der entworfenen Leiterplatte wurde der chinesische Hersteller JLCPCB gewählt. Dieser Anbieter bietet eine kostengünstige und zuverlässige Möglichkeit, Prototypen sowie Kleinserien professionell herstellen zu lassen. Die Übermittlung der Fertigungsdaten erfolgte in Form von standardisierten Gerber-Dateien, die direkt aus Fusion 360 exportiert wurden.

Die zur Bestückung benötigten elektronischen Bauteile wurden bei den bekannten Elektronikhändlern Conrad Elektronik und Reichelt Elektronik bestellt. Diese Anbieter boten eine breite Auswahl und eine schnelle Lieferung der benötigten Komponenten, wodurch eine zügige Inbetriebnahme der Platine ermöglicht wurde.

2.5 Bestückung der Platine

Nach Lieferung der Leiterplatte und der elektronischen Bauteile erfolgte die manuelle Bestückung. Dabei wurden die zuvor bestellten Komponenten gemäß dem Schaltplan und dem Platinenlayout in die dafür vorgesehenen Positionen eingesetzt. Anschließend wurden alle Kontakte sorgfältig per Hand verlötet, um eine zuverlässige elektrische Verbindung herzustellen.

Die meisten Bauteile passten exakt in die vorgesehenen Bohrungen. Lediglich bei einigen wenigen Komponenten traten kleinere Probleme mit dem Lochdurchmesser auf – in diesen Fällen mussten die Anschlussdrähte leicht nachbearbeitet oder vorsichtig angepasst werden. Abgesehen davon verlief die Bestückung weitgehend problemlos, und es konnte zügig mit der Inbetriebnahme der Schaltung fortgefahren werden.

3 Entwurf Linux-System

Für das STM32MP135F-DK wird ein Linux-Image mit Yocto gebaut, das eigene C-Programme bereits mitkompiliert im Root-Filesystem enthält. Die Builds laufen in einem Docker-Container unter Windows, um eine reproduzierbare und isolierte Umgebung zu erhalten.

3.1 Voraussetzungen

- Windows 10/11 mit Docker Desktop.
- Projektordner mit Dateien: `Dockerfile`, `hello.c`, `aus.c`, `laden.c`, `entladen.c`, `analogwerte.c` und `zustandsautomat.c`.
- Internetzugang (Yocto-Layer werden via `repo` synchronisiert).

3.2 Docker-Container bauen und starten

Bevor die eigentliche Konfiguration des Builds gestartet werden kann, muss die Buildumgebung aufgebaut und gestartet werden. Dafür wurden diese Befehle genutzt:

```
# Windows PowerShell
cd C:\Users\Leon\Documents\Yocto_Project

docker build -t yocto-stm32mp1 -f Dockerfile .

docker run -it --name yocto-dev -v
//c/Users/Leon/Documents/Yocto_Project:/work yocto-stm32mp1
```

Im Container sind die Projektdateien unter `/work` sichtbar.

3.3 Yocto-Quellen und Pakete holen

```
# Container-Shell
mkdir -p ~/stm32mp1-yocto

cd ~/stm32mp1-yocto

repo init -u https://github.com/
STMicroelectronics/oe-manifest.git -b kirkstone

repo sync -j"$(nproc)"
```

Beim ersten `envsetup.sh` prüft ST Pflichtpakete. Fehlende werden nachinstalliert.

```
sudo apt-get update

sudo apt-get install -y bsdmainutils gcc-multilib
libegl1-mesa libgmp-dev libmpc-dev pylint python3-git
python3-jinja2 socat
```

3.4 Build-Environment setzen

```
cd ~/stm32mp1-yocto

source layers/meta-st/scripts/envsetup.sh
```

Damit wird der Build-Ordner
`/stm32mp1-yocto/build-openstlinuxweston-stm32mp13-disco` angelegt.

3.5 Eigenen Layer und Sammel-Rezept erstellen

Wir bündeln alle C-Programme in einem Paket `battery-tools`, das die sechs Binaries nach `/usr/bin` installiert.

```
bitbake-layers create-layer ../meta-myapp

bitbake-layers add-layer ../meta-myapp

mkdir -p ../meta-myapp/recipes-apps/battery-tools/battery-tools

cp /work/hello.c
../meta-myapp/recipes-apps/battery-tools/battery-tools/

cp /work/aus.c
../meta-myapp/recipes-apps/battery-tools/battery-tools/

cp /work/laden.c
../meta-myapp/recipes-apps/battery-tools/battery-tools/

cp /work/entladen.c
../meta-myapp/recipes-apps/battery-tools/battery-tools/

cp /work/analogwerte.c
../meta-myapp/recipes-apps/battery-tools/battery-tools/
```

```
cp /work/zustandsautomat.c
../meta-myapp/recipes-apps/battery-tools/battery-tools/
```

Rezept `battery-tools.bb` erstellen.

```
nano ../meta-myapp/recipes-apps/battery-tools/battery-tools.bb
```

Inhalt (Kurzfassung):

- `LICENSE = MIT"` mit aktuellem `LIC_FILES_CHKSUM`.
- `SRC_URI` listet alle sechs `.c`-Dateien.
- `do_compile`: sechs Binaries bauen.
- `do_install`: nach `${bindir}` installieren.

3.6 Paket ins Image ziehen und bauen

```
echo 'IMAGE_INSTALL:append = " battery-tools "' >> conf/local.conf

echo 'ACCEPT_EULA_stm32mp13-disco = "1"' >> conf/local.conf

bitbake battery-tools
```

3.7 Image bauen (Weston-Variante)

```
bitbake st-image-weston
```

3.8 SD-Karten-Image erzeugen

Nach abgeschlossenem Build liegen die Artefakte in:

```
cd tmp-glibc/deploy/images/stm32mp13-disco
```

Das SD-Image wird mit dem ST-Skript aus einer TSV erstellt. Fehlende Tools wie `sgdisk` nachinstallieren:

```
sudo apt-get install -y gdisk

cd scripts

./create_sdcard_from_flashlayout.sh
../flashlayout_st-image-weston/extensible
/FlashLayout_sdcard_stm32mp135f-dk-extensible.tsv
```

Ergebnis: `FlashLayout_sdcard_stm32mp135f-dk-extensible.raw` im Deploy-Ordner.

```
# PowerShell (Host)
docker cp yocto-dev:/home/yocto/
stm32mp1-yocto/build-openstlinuxweston-stm32mp13-disco/
tmp-glibc/deploy/images/stm32mp13-disco/
FlashLayout_sdcard_stm32mp135f-dk-extensible.raw
C:\Users\Leon\Documents\Yocto_Project\
```

Abschließend kann die Datei mit Hilfe von BalenaEtcher unter Windows oder mit geeigneten Werkzeugen (z. B. dd oder spezialisierte GUI-Programme) unter Linux auf eine SD-Karte geschrieben werden.

4 Entwurf des Programms

4.1 Ziel und Modulübersicht

Das System umfasst mehrere eigenständige Testprogramme sowie ein Hauptprogramm mit Zustandsautomat. Die Module sind:

- **hello** – Konsolentest (`"Hallo vom STM32MP13!"`).
- **aus**, **laden**, **entladen** – Einzeltests für den I/O-Expander PCF8574. Sie initialisieren den Expander, setzen gezielt Pins und lesen einen Eingang.
- **analogwerte** – Einzeltest für den ADC ADS1015: liest AIN0 und AIN1 zyklisch und rechnet auf Volt bzw. mA um.
- **zustandsautomat** – Hauptprogramm: steuert Laden/Entladen über den PCF8574, misst Spannung/Strom via ADS1015, akkumuliert eine Ladungsgröße und verwaltet den Ablauf über eine explizite Zustandsmaschine.

4.2 Einzeübersicht

hello. Minimaler Konsolotest zum Funktionscheck der Toolchain/RootFS.

aus / laden / entladen. Diese drei Programme initialisieren den PCF8574, konfigurieren gewünschte Pins als Output oder Input und lesen zusätzlich Pin 2 als Eingang (Taster). Beispielhafter Ablauf:

```
// I2C initialisieren
pcf8574_begin();
// Pin0 als Output
pcf8574_pinMode(0, 0);
// LOW setzen
pcf8574_digitalWrite(0, false);
// Pin2 als Input
pcf8574_pinMode(2, 1);
// Tasterzustand lesen
pcf8574_digitalRead(2, &val);
```

Die konkrete Belegung: *Pin 0 = Laden*, *Pin 1 = Entladen*, *Pin 2 = Starttaster*.

analogwerte. `analogwerte` misst in einer Endlosschleife nacheinander Kanal 0 und Kanal 1 und gibt die Werte aus. Die Skalierung für AIN0 ist aktuell $U = 4{,}35 \cdot U_{\mathrm{ADC}}$, die Stromberechnung basiert auf $I\ [\mathrm{mA}] = 1000 \cdot (9{,}88 \cdot U_{\mathrm{ADC}} - 25{,}27)$. Die Skallierung des Spannungskanals ist nötig, da die zu messende Spannung über einen Spannungsteiler am ADC hängt. Die Parameter der Stromkurve wurden aus zwei Messpunkten berechnet.

4.3 Hauptprogramm: Zustandsmaschine

Das Hauptprogramm `zustandsautomat` kapselt die Steuerlogik in klar benannten Zuständen:

`Z_START`, `Z_SPANNUNG_MESSEN_1`, `Z_LADEN_AN`, `Z_LADEN_AUS`, `Z_ZAEHLER_ERHOEHEN`, `Z_ENTLADEN_AN`, `Z_ENTLADEN_AUS`, `Z_SPANNUNG_MESSEN_2`, `Z_STROM_MESSEN`, `Z_AUFSUMMIEREN`, `Z_ENDE`

Zustandslogik (verkürzt). Die zyklische Hauptschleife ruft `ausgangslogik(zustand)` und berechnet anschließend den Folgezustand mit `zustandsuebergang(zustand)`:

```
do {
  ausgangslogik(zustand);
  zustand = zustandsuebergang(zustand);
} while (!fertig);
```

In `ausgangslogik()` werden je nach Zustand Ausgänge gesetzt und Messungen durchgeführt, z.B.:

- `Z_START`:bei aktivem Taster Wechsel nach `Z_SPANNUNG_MESSEN_1`.
- `Z_SPANNUNG_MESSEN_1 / _2`: Einzelmessung ADS1015, Messung und Anzeige der Zellspannung.
- `Z_LADEN_AN`: Schalten des Lade-Relais; danach `sleep(60)` (Ladezeit).
- `Z_ENTLADEN_AN`: Schalten des Entlade-Relais.
- `Z_STROM_MESSEN`: ADS1015 messen, Strom ausrechnen.
- `Z_AUFSUMMIEREN`: Ladungsgröße fortschreiben, 1 s abwarten
- `Z_ZAEHLER_ERHOEHEN`: Wiederholungszähler erhöhen
- `Z_ENTLADEN_AUS`: Entladen abschalten, Zeit-/Akkumulatoren zurücksetzen und Zwischenergebnis ablegen.
- `Z_ENDE`: Messwertspeicher ausgeben, Ende.

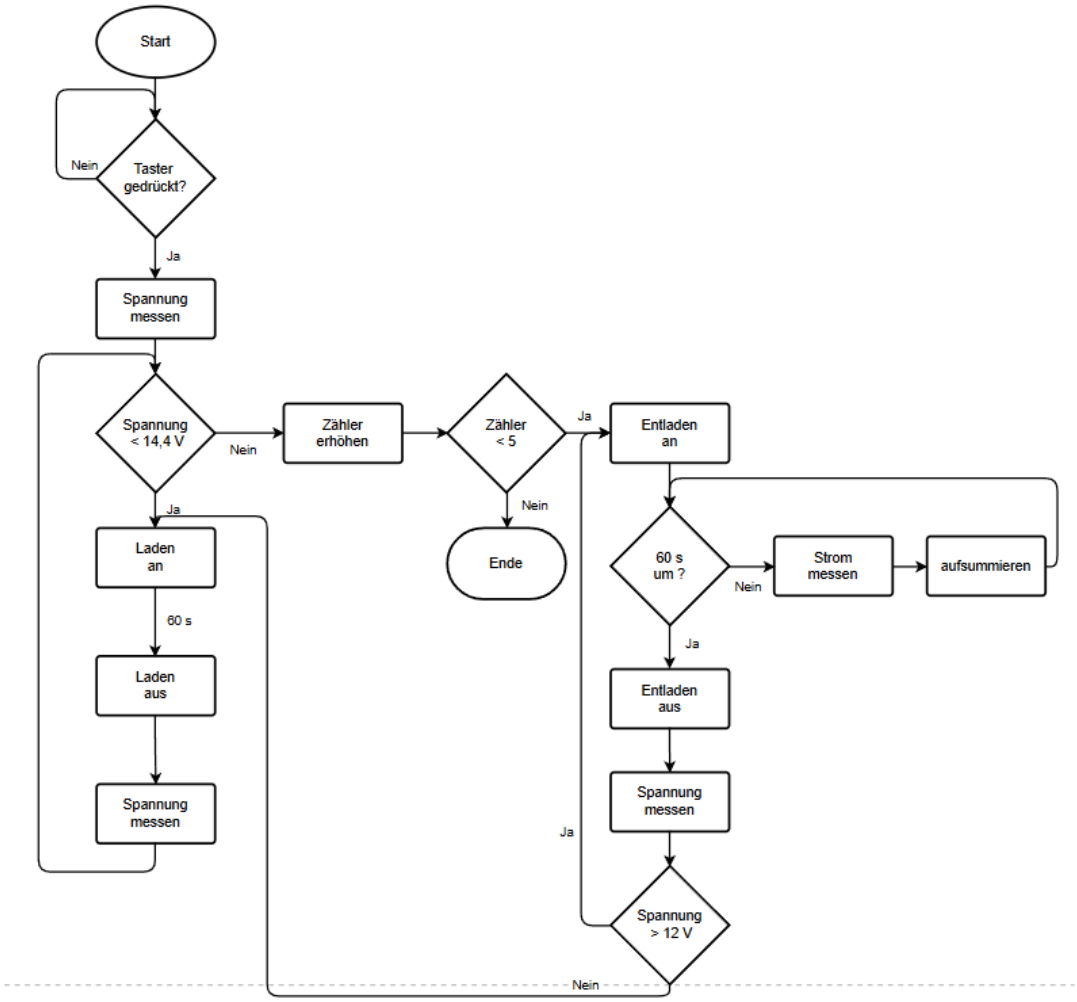

Abbildung 3: Ablaufdiagram

4.4 Zusammenfassung

Die Aufteilung in gezielte Einzeltests (PCF8574, ADS1115) erleichtert Verifikation und Fehlersuche; der Zustandsautomat orchestriert die Betriebsphasen (Laden/Entladen/-Messen) deterministisch. Das Open-Drain-Verhalten des PCF8574 (LOW = aktiv) ist dabei zentral für die Relais-/Laststeuerung, während der ADS1115 im Single-Shot-Betrieb reproduzierbare Messwerte liefert.

5 Test und Inbetriebnahme

5.1 Inbetriebnahme

Nachdem die Bestückung der Platine abgeschlossen war, konnte diese für die erste Inbetriebnahme vorbereitet werden. Hierzu wurde die Baugruppe mittels eines 20×2-poligen Flachbandkabels mit dem STM32-Controller verbunden. Über diese Verbindung werden sämtliche Steuersignale zwischen der Platine und dem Mikrocontroller übertragen.
Im Rahmen der initialen Inbetriebnahme liegt der Schwerpunkt zunächst darauf, die Baugruppe und den Controller ohne die Aktivierung spezifischer Funktionen in einen stabilen Betriebszustand zu versetzen. Auf diese Weise soll sichergestellt werden, dass die Schaltungsarchitektur sowie die Verdrahtung grundsätzlich fehlerfrei ausgeführt wurden und keine unvorhergesehenen Betriebszustände auftreten.
Nachdem die Baugruppe über einen Zeitraum von mehreren Minuten fehlerfrei betrieben wurde und weder Rauchentwicklung noch auffällige Geruchsbildung festgestellt werden konnten, kann im nächsten Schritt mit der schrittweisen Überprüfung der vorgesehenen Funktionen begonnen werden. Ziel ist es, die einzelnen Komponenten systematisch zu testen und die einwandfreie Funktionalität der gesamten Schaltung zu verifizieren.

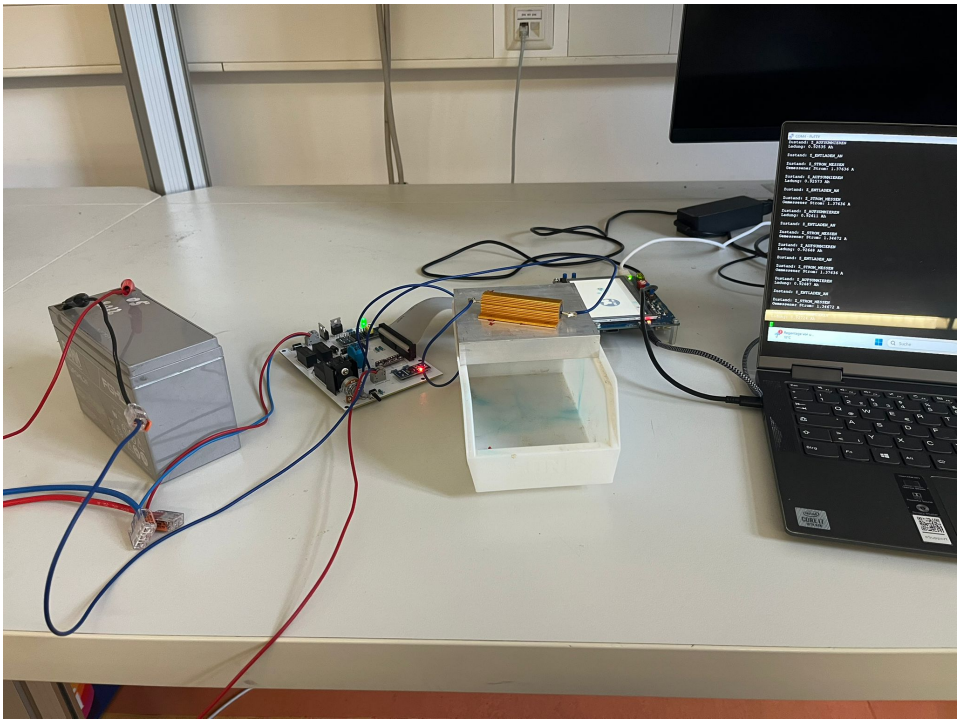

Abbildung 4: Platine und STM32-Controller während des ersten Akku-Zyklus

5.2 Test der Funktionen

Um eine ordnungsgemäße Funktion der Applikation sicherzustellen, müssen sämtliche Funktionen systematisch und präzise getestet werden. Dabei ist es erforderlich, sowohl die zentralen Hauptfunktionen – wie den Lade- und Entladevorgang sowie den automatischen Betriebsablauf – zu überprüfen, als auch kleinere, jedoch nicht weniger wichtige Teilfunktionen zu validieren. Hierzu zählen beispielsweise die Kapazitätsmessung sowie die korrekte Ansteuerung der Status-LEDs.

Ein weiterer wesentlicher Aspekt betrifft die Ausgangsspannung der Schaltung. Da der primäre Zweck dieser Schaltung in der Regeneration von Bleigel-Akkus mittels gepulster Spannung liegt, ist es notwendig, die erzeugte Ausgangsspannung mithilfe eines Oszilloskops zu überwachen und aufzuzeichnen. Auf diese Weise kann sichergestellt werden, dass die Pulsparameter den spezifizierten Anforderungen entsprechen und die Schaltung den Akku schonend und effektiv regeneriert.

Laden

Um die Funktion des Ladens zu überprüfen, wird ein 12V-Blei-Gel-Akku an die Schaltung angeschlossen. Dazu wird ein Oszilloskop parallel zum Akku angeschlossen, um

die Ausgangsspannung der Ladeschaltung aufzuzeichnen. Gleichzeitig muss der Ablauf des Steuerungsprogramms den richtigen Ausgang setzen, sodass der Ladepfad geschaltet wird.

Entladen

Um den Akku zu entladen, muss der Programmablauf ebenfalls wieder den richtigen Pfad schalten, sodass der Strom aus dem Akku herausfließen und sich dieser damit entladen kann. Nun fließt der Strom aus dem Akku über den Entladewiderstand und die Strommessung. Die Steuerung sollte nun zyklisch einen Strom messen und diesen so aufsummieren, dass die Kapazität des Akkus berechnet werden kann.

Ausgangsspannung aufzeichnen

Die Ausgangsspannung der Ladeschaltung ist ein wichtiger Aspekt. Durch die gepulste Spannung soll im Bleiakku die Sulfatschicht um die Elektroden abgebaut werden, wodurch sich die Kapazität des Akkus in der Theorie wiederherstellen lässt. Somit ist das Pulsen der Spannung äußerst wichtig und muss mit Hilfe eines Oszilloskops aufgenommen werden, um die korrekte Funktion sicherzustellen.

Ladungsmessung prüfen

Die Kapazitätsmessung an sich kann schwierig geprüft werden, da man dafür ein Messgerät bräuchte, welches die Kapazität eines Akkus misst. Die beiden Werte müssten dann verglichen werden. Da ein solches Messgerät nicht vorhanden ist, wurde die Kapazitätsmessung in zwei Teilschritten überprüft: erst die Überprüfung der Strommessung, dann die des zeitlichen Intervalls der Abtastung.
Die Strommessung konnte relativ einfach überprüft werden, indem das Entladen durch eine Spannungsquelle als Akku simuliert wurde und ein geeichtes Multimeter zur Strommessung in Reihe geschalten wurde. Über den Entladewiderstand auf der Platine wurde dann ëntladen". Der gemesssene Strom wurde nun an zwei Punkten mit dem gemessenen Strom des Multimeters verglichen, sodass die Kennlinie überprüft werden konnte.
Das Testen des zeitlichen Intervalls stellte sich anfangs als nicht so einfach dar. Eine gute Möglichkeit wurde allerdings gefunden, indem die Anzahl der Abtastungen über einen großen Zeitraum gezählt wurde. So sollte die Anzahl der Abtastungen bei einer Stunde Laufzeit beispielsweise 3600 betragen, da 3600 Sekunden. Dies war eine gute Möglichkeit, das zeitliche Intervall sehr genau einzustellen.

Simulation des gesamten Ablaufs

Um den gesamten Programmablauf zu testen, muss sich die Spannung des Akkus entsprechend dem Lade- und Entladevorgang verändern. Da das Laden und Entladen eines

Akkus einige Zeit in Anspruch nimmt, wurde an dieser Stelle wieder eine Spannungsquelle als Simulation des Akkus eingesetzt. Die Startspannung liegt bei 12,5V.
Der Ladevorgang wurde nun gestartet. Die Lade-Eingangsspannung wurde nicht angelegt, da die erwähnte Spannungsquelle sonst die Ladespannung abbekommen würde. Dadurch werden im trockenen Zustand einfach die Relais geschaltet und der Programmablauf kann getestet werden.
Die Spannung der Spannungsquelle wird nun auf 14,5V erhöht, wodurch die Steuerung im nächsten Messschritt (alle 60 Sekunden) einen fertig geladenen Akku feststellt, das Laden beendet und das Entladen startet. Auch im Zustand des Entladen wird die Spannung des Akkus alle 60 Sekunden gemessen. Dazwischen wird der Strom jede Sekunde gemessen.
Die Spannungsquelle wurde nun auf 11,8V gestellt. Jetzt wurde im nächsten Messschritt der entladene Akku festgestellt und wieder das Laden gestartet. Aktuell sind im Programm fünf Zyklen eingestellt. Der beschriebene Ablauf wurde also insgesamt fünfmal durchlaufen, bevor das Programm sich selbst beendet. Der Programmablauf wurde damit korrekt getestet.

Test des Zyklus mit Akku

Zu guter Letzt werden die Schaltung und der Ablauf mit einem 12V Bleiakku betrieben. Die Voraussetzung dafür war, dass alle vorherigen Tests erfolgreich durchgeführt werden konnten, was der Fall war. Nun wurde also am Akku-Port der 12V Akku angeschlossen und eine 20V DC-Quelle am Port für die Eingangsspannung.
Nach dem Start des Zyklus wird als Erstes entladen. Dabei wird auf dem Serial Output bei jeder Messung der Strom sowie die kumulierte Ladung angezeigt. Erreicht der Akku im Leerlauf eine Spannung von unter 12V, so schaltet die Steuerung nun in den Modus laden um. Nun wird der Akku geladen, bis er im Leerlauf eine Spannung von über 14,4V erreicht.
Abschließend konnte durch diesen Test die Funktion sowohl der Schaltung als auch der Steuerung validiert werden. Es war in beiden Modi Laden und Entladen ersichtlich, dass die erwarteten Ergebnisse eingetreten sind und der Akku sich jeweils wie erwartet verhielt.

6 Zusammenfassung und Ausblick

Zusammenfassend wurde während des Projekts eine Applikation entwickelt, mit der ein 12V Akkumulator in einem vollautomatischen Zyklus geladen und anschließend wieder entladen werden kann. Dabei wird die Leerlaufspannung des Akkus konstant überwacht und mittels Strommessung die Kapazität/ Elektrische Ladung des Akkus gemessen. Anhand der Messung und der zyklischen Wiederholung kann am Ende mehrerer Zyklen verglichen werden, wie sich die elektrische Ladung nach dem Laden des Akkus mit der Regenerationsschaltung entwickelt hat.
Die gepulste Spannung, die für die Desulfatierung der Elektroden dienen sollte, konnte während der Messungen nicht aufgezeichnet werden. Allerdings könnte dies daran liegen, dass der Widerstand des Akkus so gering ist, dass der Akku die Spannungsspitzen sofort wegzieht und das Oszilloskop zu träge ist, um die Spitzen zu messen.
Auch ohne Pulse kann festgehalten werden, dass der Akku zuverlässig geladen werden konnte. Eine genaue Aussage, ob die Akkus regeneriert werden, ist zum aktuellen Zeitpunkt noch nicht möglich. Während des Projekts wurden Akkus mit ursprünglich 7,2 Ah verwendet. Bei realistisch noch möglichen 4-5 Ah und einem Entladestrom von 1,4 A dauert ein einziger Entladevorgang fast drei Stunden. Um eine ordentliche Aussage treffen zu können, sollten mindestens fünf Zyklen durchlaufen werden. Die nötige Zeit zum fünfmaligen Laden und Entladen konnte während des Projekts nicht zusammenhängend aufgebracht werden.
Aus diesem Grund soll in Zukunft zunächst ein kleinerer Akku verwendet werden, da die Zeiten dadurch deutlich verkürzt werden können. Dies gilt als wesentlicher Verbesserungsvorschlag.
Weitere Verbesserungsmöglichkeiten sind:

- **genauere Strommessung** - die Genauigkeit der Strommessung kann verbessert werden, da der aktuelle Messbereichsendwert 5 A beträgt
- **graphische Darstellung des Verlaufs der elektrischen Ladung** - eine graphische Darstellung der elektrischen Ladung würde die Ergebnisse anschaulicher machen
- **Usability** - die Usability könnte beispielsweise durch Nutzung des Displays des STM-Controllers verbessert werden
- **Verwendung anderer Drossel** - durch die Verwendung einer anderen Drossel könnte das Pulsen der Spannung eventuell verstärkt werden, sodass es auf dem Oszilloskop sichtbar wird

Es wäre denkbar, diese Verbesserungen in Zukunft umzusetzen, um eine finale Aussage treffen zu können, ob die Re-EMF-Schaltung die elektrische Ladung von Bleiakkumulatoren effektiv regenerieren kann.

Abbildungsverzeichnis